REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE
Département d'Automatique

Projet de Fin d'Études
en vue de l'obtention du diplôme d'Ingénieur d'État en Automatique

THEME

Détection des objets mobiles et prédiction de leur mouvement dans un environnement dynamique

Etudié par :
- AMROUCHE Sadek
- OUALA Saïd

Dirigé par :
- M. ACHOUR Hakim (ENP)
- M. LAKROUF Mustapha (CDTA)

Juin 2015
École Nationale Polytechnique, 10 Avenue Hassen Badi, El-Harrach, Alger. Algérie

ENP : École Nationale Polytechnique
CDTA : Centre de Développement des Technologies Avancées

Remerciements

En premier lieu, nous remercions Dieu tout puissant de nous avoir donné le courage, la volonté et la patience pour réaliser ce travail.

Nous remercions nos parents, qui nous ont soutenus tout au long de nos études.

Nos remerciements les plus vifs s'adressent particulièrement à Monsieur A. ACHOUR, notre promoteur à l'ENP et à Monsieur M. LAKROUF, notre encadrant au CDTA. Sans oublier Monsieur R. TIAR (CDTA) et Monsieur A. KHELLOUFI (CDTA) pour leur aide et leur disponibilité.

Un grand merci à Madame O. AZZOUAOUI ainsi qu'à toute l'équipe de la division Productique et Robotique du CDTA auprès desquels nous avons acquis un savoir incommensurable.

Ce modeste travail, fruit de notre cursus, n'a été possible que grâce au concours de tous nos enseignants dont nous louons les efforts qu'ils ont consentis durant toutes ces années.

Dédicace

Je dédie ce modeste travail en premier lieu à mes très chers parents qui m'ont toujours soutenu, aidé et encouragé. Qu'ils trouvent ici le témoignage de ma reconnaissance la plus dévouée.

À mon grand-père.

À mes chers frères et à ma chère sœur à qui je souhaite beaucoup de réussites.

À toute ma famille : oncles, tantes, cousins et cousines,

À tous les membres du Club d'Activités Polyvalentes (CAP), source de ma motivation et auprès desquels j'ai appris à me surpasser et à prendre goût aux challenges,

À tous mes ami(e)s et camarades qui m'ont moralement soutenu,

À tous les enseignants qui m'ont transmis leur savoir,

À tous ceux qui m'ont aidé de près ou de loin pour réaliser ce modeste travail.

Sadek

Dédicace

Je dédie ce travail :

À mes chers parents ;

À mon frère et mes sœurs ;

À tous les membres de ma grande famille ;

À tous mes ami(e)s et camarades.

Saïd

Résumé

Un robot mobile autonome doit percevoir correctement l'environnement où il évolue. Autrement, l'interprétation qu'il en fait sera erronée et ne peut prendre des décisions correctes. La perception est également un point clé pour tout véhicule intelligent ou fournissant des fonctions d'assistance à la conduite. Dans ce cadre, la perception peut être divisée en deux parties: la première appelée SLAM (*Simultaneous Localization And Mapping*) s'intéresse à la construction d'une carte de l'environnement et à la localisation du véhicule dans cette carte, et la deuxième partie, appelée DATMO (*Detection And Tracking on Moving Objects*), traite la détection et du suivi des objets mobiles dans l'environnement.

L'objectif de ce travail est d'implémenter un module *grid-based* DATMO sur la plateforme expérimentale « Robucar » du CDTA utilisant les informations issues du télémètre laser. Pour ce faire, l'environnement est modélisé en utilisant le formalisme des grilles d'occupation et le filtrage est assuré par le filtre d'occupation bayésien (BOF).

Mots clés : Robot mobile, détection d'obstacles mobiles, suivi d'obstacles mobiles, filtre d'occupation bayésien, DATMO, grille d'occupation.

Abstract

An autonomous mobile robot must correctly perceive the environment in which it operates. Otherwise, the interpretation will be wrong and the robot can not make correct decisions. Perception is also a key point for any intelligent vehicle or even for a vehicle providing only driver assistance functions. In this context, perception can be divided into two stages: the first part called SLAM (Simultaneous Localization And Mapping) is concerned with building an online map of the external environment and localizing the host vehicle in this map, and second part, called DATMO (Detection And Tracking of Moving Objects), deals with finding moving objects in the environment and tracking them over time.

The objective of this work is to implement a grid-based DATMO module on the experimental platform « Robucar » working on the information provided by the laser rangefinder. To do this, the environment is modeled using the formalism of occupancy grids and filtering is provided by the Bayesian Occupancy Filter (BOF).

Keywords : Mobile robot, detection of moving obstacles, tracking of moving objects, Bayesian Occupancy Filter, DATMO, Occupancy Grid.

ملخص

الروبوت المتحرك المستقل يجب ان يدرك المحيط الذي يعمل فيه بشكل صحيح والا سيكون التفسير المقدم منه خاطئا ولن يتمكن من اتخاذ قرارات صحيحة. كما يعتبر الإدراك مفتاح لأية مركبة ذكية أو لتلك التي تقدم مهام المساعدة للقيادة. و في هذا السياق، يمكن تقسيم الإدراك إلى قسمين : SLAM و DATMO.

الهدف من هذا العمل هو الكشف عن العقبات باستخدام المعلومات من أجهزة الإستشعار بالليزر ثم تطوير طريقة لتقدير و توقع تحركاتها مستقبلا و قد تم تنفيذ هذا العمل و اختباره على المنصة التجريبية Robucar.

كلمات مفتاحية : الربوت المحمول، كشف العقبات، رصد العقبات.

Table des matières

Introduction générale....1

1. Généralités sur la robotique mobile 3

1.1 Robot mobile autonome 3

1.2 Les véhicules autonomes 4

1.3 Systèmes d'assistance à la conduite 5

1.4 Navigation autonome des robots mobiles 6

1.5 Présentation et définition du DATMO 7

1.5.1 Moving Objects Detection (MOD) 8

1.5.2 Moving Objects Tracking (MOT) 8

1.5.3 Bref aperçu historique du DATMO 8

1.6 Conclusion 9

2. Détection et suivi d'objets mobiles....10

2.1 La perception dans les véhicules intelligents....10

2.2 DATMO : position du problème 13

2.3 Modélisation de l'environnement 15

2.3.1 Cluster Based Model 15

2.3.2 Modèles géométriques....17

2.3.3 Grid Based Models 17

2.4 Les différentes approches de détection et de suivi d'objets mobiles (DATMO)....19

2.4.1 L'approche traditionnelle....21

2.4.2 Approche basée sur un modèle géométrique....25

2.4.3 Approche basée sur la grille d'occupation 26

2.5 Conclusion....28

3. Grid-based DATMO....29

3.1 Introduction....29

3.2 Programmation bayésienne : concepts de base....29

3.2.1 Les probabilités comme alternative à la logique classique 29
3.2.2 Définitions fondamentales 30
3.2.3 Formule de Bayes 32
3.2.4 Principe de l'inférence bayésienne 32
3.3 Programmation bayésienne 34
3.3.1 La description 34
3.3.2 La question 36
3.3.3 Inférence 37
3.4 Grille d'occupation : concepts de base 38
3.4.1 Hypothèse d'indépendance des cellules 38
3.4.2 Estimation d'une grille d'occupation 39
3.4.3 Utilisation de plusieurs capteurs 43
3.4.4 Limites de l'estimation statique des grilles 44
3.5 Filtre d'Occupation Bayésien (BOF) 45
3.5.1 Le filtre bayésien 45
3.5.2 Filtre bayésien adapté aux grilles d'occupation 46
3.5.3 Programme bayésien 47
3.5.4 Modélisation du capteur 51
3.5.5 Les modèles dynamiques 55
3.5.6 Champ des vitesses 56
3.6 Conclusion 57
4. Implémentation du grid-based DATMO 58
4.1 Présentation de l'environnement logiciel 58
4.1.1 Description d'un système d'exploitation pour robot 58
4.1.2 Robot Operating System (ROS) 59
4.1.3 Caractéristiques 60
4.1.4 Notions de base de ROS 60
4.2 Présentation de la plateforme expérimentale 62
4.3 Implémentation du *grid-based* DATMO 69

4.3.1 Acquisition des données laser 70
4.3.2 Construction de la grille d'occupation 71
4.3.3 Fast Classification of Static and Dynamic Environment 72
4.3.4 Implémentation du Bayesian Occupancy Filter (BOF) 84
4.3.5 Filtre d'occupation bayésien utilisé avec l'algorithme de classification rapide 91
4.4 Conclusion 93
Conclusion générale et perspective 94
Bibliographie 96

Table des figures

Figure 1.1 Illustration par le constructeur allemand BMW d'un véhicule autonome circulant sur une autoroute 4
Figure 1.2 : Système ADAS Renault « *Lane keeping system* » 5
Figure 1.3 : Les deux composantes de la perception : SLAM et DATMO 7
Figure 1.4 : L'Audi RS7 Sportback Adaptive Cruise Control. L'ACC maintient une distance de sécurité avec le véhicule du devant en actionnant automatiquement l'accélérateur ou le frein. 7
Figure 2.1 : L'objet de la perception en robotique. A droite : l'environnement réel. A gauche : le résultat de l'inférence des données capteur par le robot 11
Figure 2.2 : La perception utilise les données des capteurs en entrée et produit en sortie : l'état du véhicule (position et vitesse, ...), la carte de l'environnement et une liste des objets mobiles et leurs caractéristiques cinématiques 12
Figure 2.3 : Architecture PDA d'un système robotique 13
Figure 2.4 : Représentation graphique du problème du DATMO 14
Figure 2.5 : Grandeurs intervenant dans le module de perception 14
Figure 2.6 : Représentation d'un obstacle par la méthode du regroupement de points 16
Figure 2.7 : Inconvénient de la représentation par *clustering* dû aux parties non observées de l'objet. 16
Figure 2.8 : Inconvénient de la représentation par *clustering* dû à l'alignement des objets 16
Figure 2.9 : Avantage de la modélisation avec des formes géométriques. A droite : sans modèle géométrique. A gauche : avec modèle géométrique. 17
Figure 2.10 : Représentation de l'environnement avec une grille d'occupation. 18
Figure 2.11 : Les trois classes d'approches du DATMO. Les carrées rouges sont obligatoires tandis que les carrés bleus ne le sont pas [7]. 19
Figure 2.12 : Pipeline DATMO typique [7]. 20
Figure 2.13 : Utilisation de la méthode CRFs (Conditional Random Fields) pour modélisation des obstacles [15]. 22
Figure 2.14 : Structure d'un système de suivi selon l'approche traditionnelle 23
Figure 2.15 : L'étape de fenêtrage dans l'approche traditionnelle du DATMO 23
Figure 2.16 : Boucle de prédiction-estimation d'un filtre d'occupation bayésien 27
Figure 2.17 : Utilisation de l'algorithme « Fast Clustering-Tracking » avec un filtre d'occupation bayésien 27
Figure 3.1 : Résultat de l'inférence bayésienne en fonction du choix de l'a priori 34

Figure 3.2 : Canevas d'une représentation d'un programme bayésien.....37
Figure 3.3 : Une grille d'occupation construite à partir de données laser. Les zones en noir correspondent aux cellules avec une probabilité d'occupation élevée. Les cellules en blanc correspondent à l'espace libre.....39
Figure 3.4 : Etude comparative des méthodes de construction de grilles d'occupation d'une scène de *25cm* × *25cm*.....40
Figure 3.5 : La probabilité $P(m_i | z_{t,n})$ en fonction de la distance $dist(x_t, m_i)$.....43
Figure 3.6 : Grille d'occupation modélisant la salle des robots du CDTA.....43
Figure 3.7 : Le filtre Bayésien vu comme une boucle prédiction-estimation.....46
Figure 3.8 : Programme bayésien du *Bayesian Occupancy Filter* (BOF).....50
Figure 3.9 : Filtrage bayésien appliqué à l'estimation de l'occupation et de la distribution de probabilité sur les vitesses des cellules d'une grille d'occupation.....50
Figure 3.10 : Modèle capteur télémétrique : programme bayésien.....54
Figure 3.11 : Exemple de modèle probabiliste d'un capteur télémétrique laser.....55
Figure 4.1 : Quelques plateformes robotiques prisent en charge par ROS.....59
Figure 4.2 : Fonctionnement d'un programme sous ROS.....62
Figure 4.3 : La plateforme expérimentale « Robucar » du CDTA.....62
Figure 4.4 : A gauche : Le CyCab. A droite : Un Robucar TT : déclinaison tout terrain du Robucar.....63
Figure 4.5 : Schéma globale du Robucar.....64
Figure 4.6 : Les trois modes de braquage du Robucar.....64
Figure 4.7 : Schéma résumant la commande des moteurs et des actionneurs du Robucar par le PC embarqué.....65
Figure 4.8 : Schéma illustrant l'organisation de la communication dans le Robucar.....66
Figure 4.9 : Le capteur laser SICK LMS 511 monté sur l'avant du Robucar.....66
Figure 4.10 : Plage de balayage du capteur SICK LMS 511.....67
Figure 4.11 : Schéma descriptif du fonctionnement du LMS.....67
Figure 4.12 : Utilisation du temps de vol pour calculer la distance des obstacles.....68
Figure 4.13 : Caractéristiques technique du télémètre laser SICK LMS 511.....68
Figure 4.14 : Schéma descriptif du fonctionnement des odomètres.....69
Figure 4.15 : Structure de détection et de suivi des obstacles mobiles en *grid-based* DATMO.....69
Figure 4.16 : Acquisition des données laser issues d'un balayage du hall du CDTA.....70
Figure 4.17 : Grille d'occupation modélisant le hall du CDTA.....71
Figure 4.18 : Grille d'occupation modélisant un environnement extérieur.....72
Figure 4.19 : *Grid-based* DATMO avec utilisation d'un algorithme de classification rapide.73
Figure 4.20 : Mise en correspondance des grilles $OGt[i]$ et $OGt-1[i]$.....74

Figure 4.21 : Résultats obtenus du déroulement du scénario 1 77
Figure 4.22 : Résultats obtenus du déroulement du scénario 2 80
Figure 4.23 : Résultats obtenus du déroulement du scénario 3 83
Figure 4.24 Champ des vitesses choisis 84
Figure 4.25 : Scène observée par le robot et décrite dans le scénario 4 86
Figure 4.26 : Résultats obtenus du déroulement du scénario 4 87
Figure 4.27 : Résultats obtenus du déroulement du scénario 5 88
Figure 4.28 : Résultats obtenus du déroulement du scénario 2 89
Figure 4.29 : Résultats obtenus du déroulement du scénario 2 en utilisant le BOF + FC....92

Introduction générale

« *Tout ce qu'un homme est capable d'imaginer, d'autres hommes seront un jour capables de le réaliser.* »

Jules Verne (1828-1905)

La conception des robots mobiles autonomes (appelés aussi *véhicules autonomes*) est un domaine de recherche en pleine expansion. En effet, d'après *l'International Federation of Robotics*, le nombre de robots mobiles recensés dans le monde a dépassé le million d'unités fin 2010. Ces véhicules sont largement utilisés dans l'industrie, comme moyen de transport et même dans l'exploration planétaire.

Actuellement, les milieux occupés par les robots ont fortement tendance à s'extrapoler à des environnements de bureaux ou à des environnements domestiques. Les types d'applications possibles deviennent alors innombrables. Cela peut être des tâches de nettoyage et d'entretien, une assistance à une personne handicapée dans des tâches d'exploration et de préhension, le guidage lors d'une visite de musée, etc. On parle alors, de façon générale, de « robotique d'intérieur ».

Un tel cadre d'utilisation requiert un niveau minimum d'autonomie et de facilités de navigation pour le système robotisé. Pour ce faire, le robot est doté de capacités de perception et d'information sur son environnement qui lui permettent de se mouvoir en autonomie, sans se perdre, tout en évitant les obstacles. La perception apparaît ainsi comme la composante la plus essentielle dans l'autonomie d'un robot étant donné que les décisions et les actions qu'il peut prendre en dépendent directement.

Dans sa formulation la plus simple, la perception consiste à obtenir des informations sur le monde qui entoure le robot par le biais des capteurs et construire ensuite une représentation (modèle) de cet environnement.

Problématique

Un robot qui se déplace dans un environnement dynamique et inconnu pour accomplir une mission a besoin de percevoir le monde qui l'entoure et d'en faire une analyse cohérente. Le DATMO « *Detection And Tracking of Moving Objects* » est une composante essentielle de la perception et sert de brique de base à une multitude d'autres tâches en robotique notamment en

navigation autonome et en assistance à la conduite (ADAS) « *Advanced Driver Assistance Systems* ».

En effet, il est essentiel de détecter et d'anticiper le mouvement des objets présents dans l'environnement ainsi que leurs propriétés cinématiques afin de permettre au robot de raisonner et de planifier correctement des actions en fonction de ce qui l'entoure.

Dans ce mémoire, il est question de traiter la problématique du DATMO et de réaliser pour la première fois, sous l'environnement ROS, un module permettant la détection des objets mobiles et du suivi de leur mouvement dans un environnement dynamique en utilisant un télémètre laser 2D.

Ce travail intervient dans le cadre du projet de recherche du CDTA (Centre de Développement des Technologies Avancées) visant à doter la plateforme robotique expérimentale « Robucar » de modules lui assurant une navigation autonome complète dans un environnement dynamique.

Organisation du mémoire

Ce mémoire s'étend sur quatre chapitres encadrés par une introduction générale et une conclusion générale.

Dans le premier chapitre, on s'intéressera au contexte général dans lequel s'insère notre travail en présentant les notions de base de la robotique mobile tout en insistant sur le lien de dépendance fort entre la navigation autonome et le DATMO.

Le second chapitre expose un état de l'art des différentes résolutions du problème du DATMO. Nous remarquerons qu'une approche de résolution de ce problème se démarque fortement des autres approches.

Le troisième chapitre a pour but de présenter la solution adoptée et les différents prérequis théoriques nécessaires à son implémentation. Nous reviendrons notamment sur certaines notions des probabilités et nous présenterons le formalisme de programmation bayésienne utilisé dans les implémentations présentées au chapitre suivant.

Le dernier chapitre est consacré à l'implémentation et la présentation des résultats expérimentaux obtenus du fonctionnement du robot en temps réel dans des environnements différents.

Chapitre 1

Généralités sur la robotique mobile

De l'inauguration, en 2013, de la première ligne d'assemblage 100% robotisée aux explorations du rover « Curiosity » sur Mars, de l'essor des opérations chirurgicales assistées par ordinateur à la démocratisation des drones et des robots aspirateurs vendus à plusieurs millions d'exemplaires. Chaque jour, la liste des applications issues de la robotique s'allonge, tandis qu'elles s'ancrent un peu plus dans notre quotidien.

La robotique a atteint aujourd'hui son plus grand succès dans le monde de l'industrie manufacturière. Les robots manipulateurs à eux seuls représentent un marché de 2 milliards de dollars tandis que les robots mobiles connaissent un essor très important notamment les véhicules autonomes dont le marché économique semble immense et très prometteur.

1.1 Robot mobile autonome

Dans sa thèse, Nicolas Morette [1] précise qu'en robotique, on distingue les robots en deux principaux types : *les robots manipulateurs* et *les robots mobiles*. Les robots manipulateurs sont fixés au sol et ne peuvent se déplacer, contrairement aux robots mobiles.

De manière générale, on regroupe sous l'appellation *robots mobiles* l'ensemble des robots à base mobile. Le plus souvent, lorsqu'on parle de *robots mobiles*, on sous-entend *robots mobiles à roues*. Les autres types de robots mobiles sont généralement désignés par leur type de locomotion (robots mobiles à pattes, robots mobiles volants, robots mobiles marins ...).

On distingue également dans [1] deux principaux modes de fonctionnement pour un robot mobile : télé-opéré et autonome. En mode télé-opéré, une personne contrôle le robot à distance en envoyant ses ordres via une interface de commande (généralement un joystick ou un clavier) et ceux-ci sont ensuite transmis au robot via une interface de communication (réseau local, internet, satellite, etc.). Les robots sont rarement placés dans ce mode de fonctionnement et d'ailleurs, selon le niveau de télé-opération, le terme « *robot* » est plus au moins justifié.

Inversement, en mode autonome le robot doit prendre ses propres décisions. Ce qui veut dire qu'il doit être en mesure de percevoir son environnement et de savoir comment réagir en conséquence. Cela va de la localisation et la cartographie de son environnement à la planification

de son parcours et la détermination des mouvements qui vont lui permettre d'atteindre son objectif.

1.2 Les véhicules autonomes

Les véhicules autonomes (appelés aussi *véhicules intelligents*) sont une application typique du domaine de la robotique mobile. Aussi appelés *véhicules intelligents*, les véhicules autonomes sont des robots mobiles capables de rouler automatiquement et en toute autonomie (accélérer, freiner, tourner, éviter les obstacles, etc.) dans le trafic réel et sans l'intervention d'un être humain.

La construction des véhicules autonomes a été un objectif central de la recherche en intelligence artificielle lors des dix dernières années. Aujourd'hui, l'IEEE (« *Institute of Electrical and Electronics Engineers* ») avance dans sa dernière étude en date que 75 % des voitures qui circuleront aux États-Unis en 2040 seront des voitures autonomes. Le géant Google va plus loin en estimant, à travers son cofondateur Sergey Brin, que le grand public devrait y avoir accès dans quelques années, avant que la technologie ne se démocratise [2].

Figure 1.1 Illustration par le constructeur allemand BMW d'un véhicule autonome circulant sur une autoroute

En France, les premiers essais sur routes ouvertes sont attendus pour octobre 2015 tandis qu'en Allemagne, les constructeurs Audi, BMW et Mercedes ont tous déjà dévoilé des projets de voitures autonomes. Des tests en conditions réelles sont déjà réalisés en Suède, dans le cadre du projet « Drive Me » de Volvo, qui prévoit de mettre sur le marché une voiture capable de suivre un itinéraire préenregistré sans aucune intervention du passager pendant tout le trajet. Bref, dans le monde entier, les tests des constructeurs engagés sur ce créneau prometteur se multiplient et visent à faire de cette révolution bientôt une réalité.

Nous l'avons compris, les véhicules totalement autonomes ne sont pas pour tout de suite, mais progressivement, plusieurs phases et fonctions de conduite gagnent en autonome. On peut

retrouver aujourd'hui des véhicules qui peuvent se déplacer en convoi de façon autonome ou des systèmes de freinage d'urgence totalement automatisés. Nombre de voitures de marque peuvent aujourd'hui se garer toutes seules et présentent plusieurs fonctions autonomes : allumage automatique des feux et des essuie-glaces, régulation de vitesse, assistance au changement de file, freinage assisté, régulation de distance, correcteur électronique de trajectoire ... etc.

Ces technologies sont appelées « *Advanced Driver Assistant Systems (ADAS)* » par les constructeurs ou « Systèmes d'assistance à la conduite » en français et permettent notamment de libérer le conducteur et de prendre le relai sur un certain nombre de tâches de conduite tout en assurant une sécurité accrue.

1.3 Systèmes d'assistance à la conduite

Les systèmes ADAS assistent le conducteur dans différentes tâches de conduite. En conséquence, l'utilisation de ces systèmes permet d'améliorer considérablement la sécurité routière. Le confort peut également être amélioré mais le but principal des systèmes d'assistance à la conduite reste avant tout la sécurité routière. En effet, les dernières études en date [3] montrent que l'implémentation des systèmes ADAS dans les récents véhicules a conduit à une baisse de mortalité de 40% dans les accidents de la route.

Trois principaux acteurs interviennent dans le développement des technologies ADAS ; les gouvernements, les institutions de recherche et les constructeurs automobiles. Chaque partie prenante a son propre objectif dans le développement de ces systèmes. Les gouvernements tentent de résoudre le problème de la sécurité routière en palliant les déficiences et les erreurs des conducteurs. Les instituts de recherche travaillent sur les technologies innovantes, et les constructeurs automobiles y voient un instrument de marketing non négligeable [4].

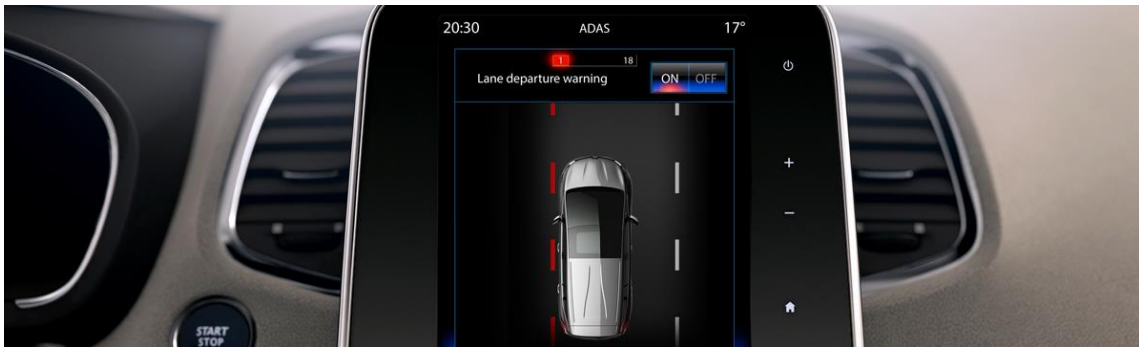


Figure 1.2 : Système ADAS Renault « *Lane keeping system* »

Aujourd'hui, beaucoup d'équipes travaillent sur la recherche et le développement de nouveaux systèmes ADAS et leur commercialisation. En Europe, les projets « RESPONSE » et « eSafety » jouent un rôle important dans la conception des systèmes ADAS qui seront utilisés dans la voiture du futur en 2020.

Le développement de tout système d'assistance à la conduite repose fondamentalement sur un système de perception de l'environnement capable de fournir toutes les informations nécessaires à la tâche de conduite. En particulier, les caractéristiques dynamiques (position et vitesse) des objets qui doivent être estimés de manière précise et robuste.

On constate donc d'emblée que les systèmes d'assistance à la conduite sont étroitement liés aux systèmes de perception de l'environnement. La perception et le raisonnement fiable et efficace dans des environnements dynamiques et encombrés représentent toujours des défis majeurs pour les systèmes d'assistance à la conduite.

1.4 Navigation autonome des robots mobiles

Le développement des véhicules autonomes a soulevé un grand nombre de problèmes dont la plupart ne sont toujours pas résolus. Ceux-ci viennent essentiellement du fait que ces véhicules sont destinés à évoluer de manière autonome dans environnements inconnus, peu ou pas structurés. Les principaux travaux dans ce domaine portent essentiellement sur comment développer des techniques et des technologies permettant au robot mobile de « *percevoir* » plus de choses dans son environnement et de « *penser* » davantage et plus vite.

Un des problèmes les plus importants de la robotique mobile autonome est la navigation autonome. L'objectif est de développer des techniques permettant au robot mobile de se déplacer sans intervention humaine dans un environnement inconnu et en présence d'obstacles mobiles. Cela consiste en l'exécution d'un certain nombre d'actions élémentaires (déplacement, manipulation d'objets...) qui nécessite une localisation précise, ainsi que la construction d'un bon modèle de l'environnement.

La construction du modèle de l'environnement est une étape essentielle et primordiale pour une multitude de fonctions en robotique mobile. Elle est le résultat de la résolution d'un problème plus général : la perception. En effet, un véhicule observe le monde externe à l'aide de capteurs et construit un modèle interne de l'environnement extérieur. Il met à jour continuellement ce modèle en utilisant les dernières données des capteurs [5].

Dans ce cadre, la perception peut être divisée en deux étapes : la première partie, appelée SLAM (*Simultaneous Localization And Mapping*) s'intéresse à la construction d'une carte de l'environnement extérieur et à la localisation du véhicule lui-même dans cette carte, et deuxième partie traite la détection et du suivi des objets mobiles qui s'y trouvent (DATMO pour *Detection And Tracking of Moving Objects*).

Ce dernier est un point clé pour le fonctionnement d'un véhicule autonome comme pour un véhicule fournissant des fonctions d'assistance à la conduite (ADAS). En effet, il est vital de détecter et d'anticiper le mouvement des objets présents dans l'environnement ainsi que leurs

propriétés cinématiques afin de permettre au robot de raisonner et de planifier correctement des actions en fonction de ce qui l'entoure. De ce fait, l'utilité du module DATMO s'avère incontournable que ce soit dans le cadre d'une application ADAS ou en navigation autonome.

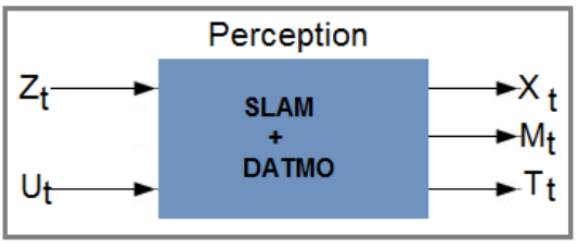


Figure 1.3 : Les deux composantes de la perception : SLAM et DATMO.

1.5 Présentation et définition du DATMO

Le paragraphe précédent nous a permis de conclure que les systèmes de détection et de suivi des objets mobiles sont un élément majeur dans la perception et la robotique en général. Les industriels commencent à exprimer un intérêt fort pour ces technologies. Un exemple typique est l'« *Adaptive Cruise Control* » *(Radar de régulation de distance)* où il est question de réduire les accidents de la route en utilisant un système de détection de collision très performant. L'exigence principale d'un tel système est un module DATMO robuste.

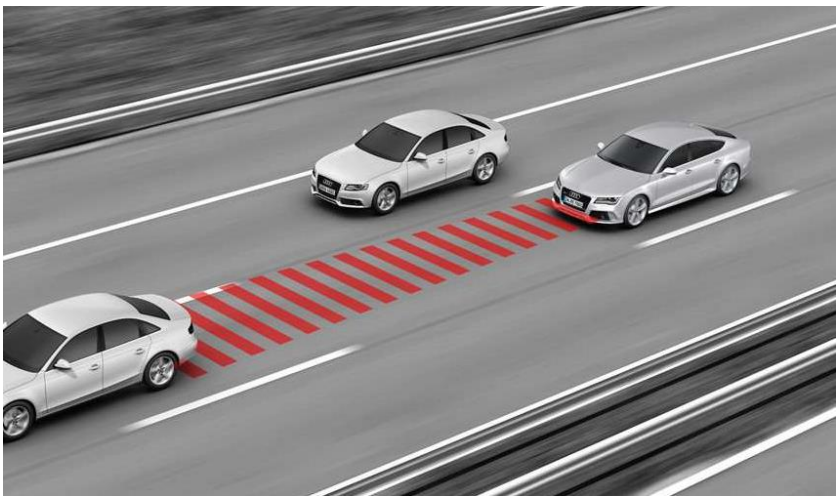


Figure 1.4 : L'Audi RS7 Sportback Adaptive Cruise Control. L'ACC maintient une distance de sécurité avec le véhicule du devant en actionnant automatiquement l'accélérateur ou le frein.

L'estimation des caractéristiques dynamiques des objets mobiles sur la route est essentiellement un problème de détection et de poursuite multi-cibles. L'objectif est de recueillir des observations, à savoir, des données des capteurs, sur les nombreux obstacles présents dans l'environnement du véhicule, puis d'estimer à chaque pas de temps les positions et les vitesses de ces obstacles [6].

En DATMO, la détection des objets mobiles est traitée séparément de la partie *tracking*. La liste des objets détectés est alors considérée comme l'entrée de la partie poursuite.

1.5.1 Moving Objects Detection (MOD)

Une énorme quantité de travail a été faite pour la détection des objets en mouvement depuis le début des années 80. Les premières techniques étaient basées sur l'extraction des objets en comparant les données reçus des capteurs à l'instant t avec la carte construite avec le SLAM afin de déterminer la liste d'obstacles [5]. Puis avec l'évolution de la technologie des capteurs qui deviennent de plus en plus précis, de nombreuses autres techniques ont été développées notamment celle utilisant les grilles d'occupation. Aujourd'hui, le problème du SLAM est séparé du DATMO, les deux modules fonctionnent indépendamment mais interagissent et se complémentent.

1.5.2 Moving Objects Tracking (MOT)

Les objets détectés par la partie MOD sont envoyés à l'entrée d'un algorithme de *Moving Objects Tracking* (MOT). Habituellement, le problème MOT est divisé en quatre sous-problèmes principaux :

1. Gating : définition d'une zone autour de la position prédite d'une cible donnée où de nouvelles observations vont être recherchées.
2. Association de données : processus d'attribution des nouvelles observations aux pistes existantes. Cette association se fait dans la zone définie dans (1).
3. Gestion des *tracks* (pistes) : créer et gérer une liste d'objets poursuivis et la définition des règles de création d'une nouvelle piste (*track*) et les règles de suppression d'une piste existante.
4. Filtrage : un processus récursif d'estimation des propriétés de chaque cible poursuivit.

Deux des quatre sous-problèmes précédents, à savoir l'association des données et le filtrage, sont des domaines de recherche plus attractifs que les deux restants. De nombreuses techniques ont été développées notamment en association de données pour traiter les différents scénarios qui peuvent se présenter.

1.5.3 Bref aperçu historique du DATMO

L'intérêt pour le DATMO est né avec les premiers travaux dans le domaine des véhicules intelligents et autonomes au début des années 80. Un grand projet a été lancé en Europe sous le nom de PROMETHEUS dès 1986 suivi ensuite par un certain nombre d'initiatives similaires notamment au Japon et aux États-Unis (Bertozzi et al. 2000 ; Sun et al. 2006) [5].

Alimenté par les dernières innovations technologiques dans le domaine du développement des capteurs, les dernières avancées en DATMO ont essentiellement porté sur l'amélioration de la détection en utilisant des techniques de reconnaissance des formes, une robustesse accrue par la fusion de données et une modélisation plus précise des capteurs et de la scène (environnement). En utilisant des capteurs laser de grande précision, des résultats importants ont été obtenus par les chercheurs. Cependant, avec des capteurs laser de faible résolution et des données bruitées, le problème reste toujours ouvert.

1.6 Conclusion

A travers ce premier chapitre, nous avons présenté de façon générale la problématique du DATMO ainsi que le contexte général dans lequel elle s'insère. Nous avons notamment insisté sur la relation qui existe entre la perception, la navigation autonome et les systèmes d'assistance à la conduite. Cette relation fera l'objet d'une étude plus détaillée au chapitre suivant.

Nous soulignons également que l'utilisation du DATMO en navigation autonome ou dans un système ADAS est une approche très récente et en constante évolution. Bien que de nombreuses méthodes intéressantes et efficaces ont été mises au point, un système DATMO efficace et totalement fiable reste à développer [7].

Ce premier chapitre sera compléter par une étude plus détaillées du problème de détection et de suivi des objets mobiles suivie d'une taxonomie des différentes classes d'approches de résolution existantes.

Chapitre 2

Détection et suivi d'objets mobiles

Dans la littérature de la robotique mobile, on retrouve plusieurs méthodes permettant d'assurer la détection des objets mobiles et le suivi de leur mouvement. Etant donné l'importance de ce module dans la tâche de perception, notamment en robotique autonome où la sécurité des personnes est hautement prioritaire, les axes de recherche en DATMO se sont multipliés afin de répondre à toutes les problématiques posées et d'apporter une constante amélioration aux performances de ce module.

Ce deuxième chapitre a pour but de présenter les différentes méthodes développées pour la détection et le suivi des objets mobiles (DATMO) en commençant par l'introduction des principes fondamentaux de la perception dans le cadre des véhicules intelligents. Ce chapitre présente également une taxonomie du DATMO telle qu'elle est décrite par Anna Petrovskaya dans « *Awareness of Road Scene Participants for Autonomous Driving* » [7].

2.1 La perception dans les véhicules intelligents

Littéralement la perception (du latin *perceptio*, *percipio*) est le processus de compréhension de l'environnement par l'organisation et l'interprétation des données qui proviennent des capteurs [5].

Dans le contexte de la robotique mobile (ou des véhicules intelligents), elle consiste à obtenir des informations sur l'environnement qui entoure le robot en utilisant les capteurs, un prétraitement sur les données de ces capteurs et en inférant une représentation (modélisation) de cet environnement. Concrètement, ceci est réalisé en obtenant les informations fondamentales suivantes :

- La position des objets par rapport aux autres objets. Ces informations sont appelées *carte de l'environnement* (ou *map*) ;
- La position du véhicule dans cette carte ;
- Distinction entre les objets statiques et les objets dynamiques ;
- Le type de chaque objet présent dans la carte, aussi appelé classification des informations (voiture, bicyclette, personne, ... etc.).

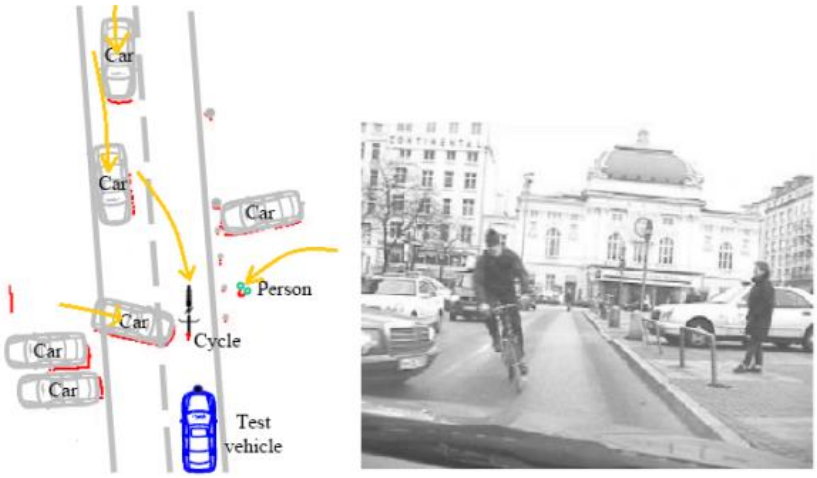


Figure 2.1 : L'objet de la perception en robotique. A droite : l'environnement réel. A gauche : le résultat de l'inférence des données capteur par le robot.

La figure ci-dessus, tirée de [8], a pour but d'illustrer l'objectif de la perception en robotique. L'image de droite montre l'environnement réel dans lequel se trouve le robot, tandis que l'image de gauche illustre la représentation de cet environnement en inférant les données des capteurs. Les données provenant du capteur laser sont représentés par les *points rouges* et le véhicule autonome (aussi appelé *véhicule de test* ou *véhicule hôte*) est représenté en bleu. On constate d'après cette figure que le robot a inféré les formes de la route et des objets qui s'y trouvent, la position de chaque objet et distingue entre les objets statiques et les objets dynamiques (illustré par les flèches jaunes). Une telle représentation est généralement obtenue en fusionnant les données de plusieurs capteurs.

La perception est le problème le plus important à résoudre en vue de permettre au robot de fonctionner sans intervention humaine. La perception, telle que définie plus haut, requiert la résolution des cinq sous-problèmes suivants :

- *Mapping* : permet d'obtenir les relations géométriques entre les objets statiques présents dans l'environnement. La résolution de ce problème est nécessaire pour la construction d'une carte en temps réel en utilisant les données des capteurs.
- *Localisation* : définie comme étant le processus permettant d'obtenir la relation entre le véhicule autonome et les objets statiques. Un véhicule autonome doit connaitre sa position dans la carte de l'environnement.
- *Détection des objets mobiles* : permet au robot de distinguer entre les objets statiques et les objets dynamiques.
- *Suivi des objets mobiles* : défini comme étant le processus permettant d'établir les relations géométriques et temporelles entre le robot autonome, les objets mobiles et l'environnement statique.

– *Classification des objets* : La résolution de ce problème permet de connaitre le type de chaque objet, notamment les objets dynamiques.

Les quatre premiers sous-problèmes sont regroupés dans la littérature en deux principaux problèmes : SLAM (*Simultaneous Localization And Mapping*) et DATMO (*Detection And Tracking of Moving Objects*) [5].

La résolution du problème SLAM permet d'obtenir la carte de l'environnement où évolue le robot ainsi que la position, à chaque instant, de celui-ci dans cette carte. Tandis que le DATMO lui permet de prendre en considération le mouvement des objets mobiles de sorte à ce que les actions appropriées soient exécutées.

La classification des objets peut être considérée comme une amélioration des résultats donnés par les deux modules SLAM et DATMO. Elle est particulièrement utile et utilisée en DATMO.

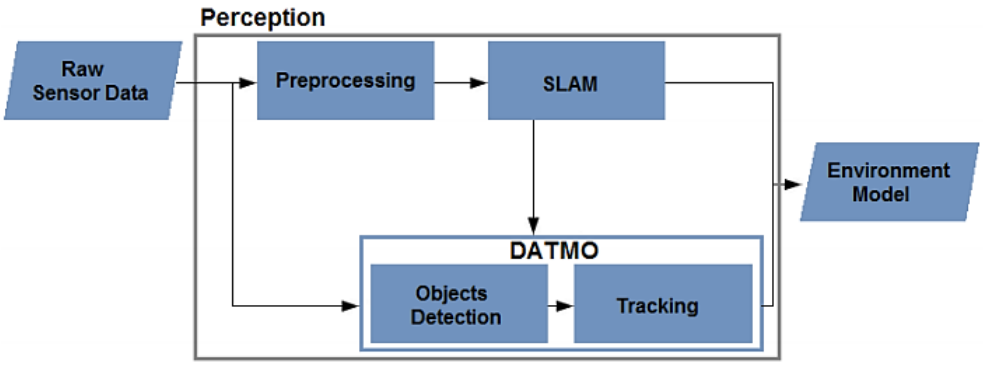


Figure 2.2 : La perception utilise les données des capteurs en entrée et produit en sortie : l'état du véhicule (position et vitesse, ...), la carte de l'environnement et une liste des objets mobiles et leurs caractéristiques cinématiques.

La carte de l'environnement incluant le comportement dynamique des objets mobiles (vitesses, accélérations, ...) donnée par la résolution du problème SLAM et DATMO est utilisée pour des applications de plus haut niveau telle que l'assistance à la conduite du véhicule. En effet, si le robot parvient à accomplir le SLAM et le DATMO d'une manière fiable et en temps réel, on peut utiliser le modèle généré pour, par exemple, détecter des situations critiques et alerter en avance le conducteur ou d'actionner automatiquement les freins en cas d'une collision inévitable.

Finalement, on peut dire que l'importance de la perception peut être vue à partir du fait qu'elle est le premier élément dans l'architecture d'un système robotique qui regroupe également la décision et l'action. Cette architecture est aussi appelée PDA (pour *perception*, *decision*, *action* en anglais) (voir la figure ci-dessous). On constate d'emblée que les actions du robot dépendent des décisions qu'il a prise, qui elles, dépendent directement du modèle de l'environnement produit par la perception.

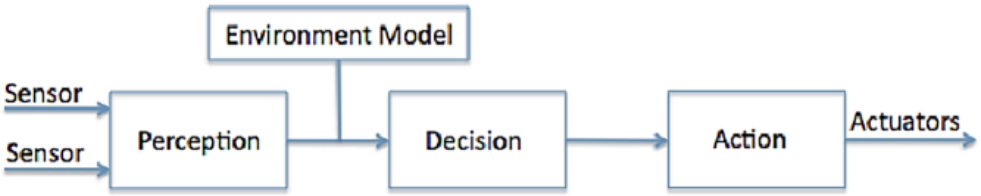


Figure 2.3 : Architecture PDA d'un système robotique

Plusieurs travaux ont été fait en perception et en utilisant différents capteurs. Dans le cadre de notre projet, nous avons utilisé un télémètre laser « SICK LMS 500 » monté sur l'avant du *Robucar* afin de résoudre le problème du DATMO.

Nous avons montré dans ce paragraphe le rôle important que joue la perception dans un système robotique et l'effet de ses performances sur les autres éléments de l'architecture PDA. La détection et le suivi des objets mobiles (*Detection And Tracking of Moving Objects*) est une étape à la fois complexe et cruciale dans la tâche de perception étant donné qu'elle permet de suivre l'évolution de l'environnement au cours du temps et de prédire ensuite son comportement.

2.2 DATMO : position du problème

Pour un problème d'estimation bayésienne général, l'objectif est d'inférer l'état S d'un système en se basant sur un ensemble de mesures Z. En raison de l'incertitude sur ces mesures, une représentation probabiliste est mieux adaptée pour résoudre le problème. En effet, dans des environnements complexes et non-prédictifs, on ne peut décrire les situations ou représenter les informations que d'une façon probabiliste ou statistique. Ainsi, nous nous plaçons dans le cadre d'une formulation dite *bayésienne* du problème (qui sera entièrement décrite dans le chapitre suivant) et on notera $bel \stackrel{def}{=} P(S|Z)$ la distribution de probabilité à postériori ou la *croyance bayésienne*.

Dans le cas d'un système bayésien dynamique, les états changent au cours du temps et sont supposés évoluer comme un processus de Markov. Le but étant d'estimer la croyance à l'instant « t » notée : $bel_t = P(S_t|Z_1, \dots, Z_t)$

Le comportement du système est entièrement décrit avec deux lois probabilistes principales, à savoir le modèle de mesure $P(Z|S)$ qui décrit la manière dont les capteurs calculent les grandeurs physiques mesurées et le modèle dynamique $P(S_t|S_{t-1})$ qui décrit la manière dont évolue le système entre deux pas de temps.

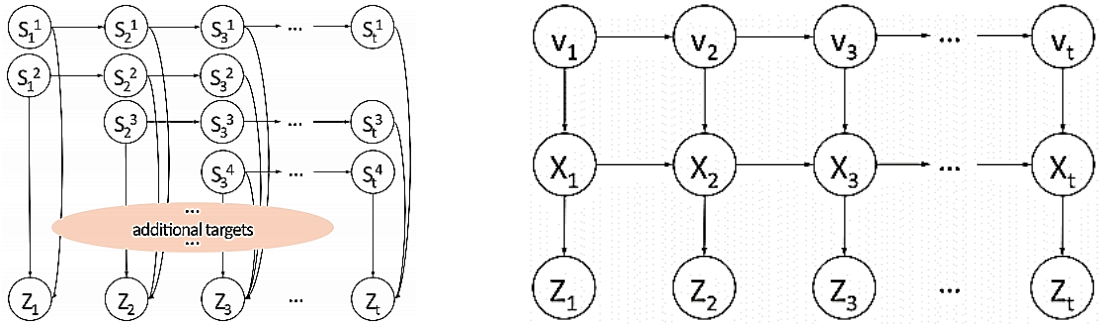
Etant donné ces deux distributions probabilistes, et les mesures obtenues à l'instant t, la croyance est calculée récursivement en utilisant l'algorithme de filtrage bayésien :

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

Où η est une constante de normalisation.

Dans le cadre du problème de détection et de suivi des objets mobiles, le système se compose d'un ensemble de cibles mobiles $T^1, \ldots, T^k$. Le nombre de cible k_t change au fil du temps étant donné que certaines cibles quittent la scène et d'autres y entrent. Pour chaque cible, les paramètres estimés, notés $X_t = (x_t, y_t, \theta_t)$, comprennent la position instantanée donnée par les deux coordonnées cartésiennes (x_t, y_t) et une orientation θ_t dans le plan 2D. Les paramètres comprennent également la vélocité v_t de chaque obstacle à l'instant t qui est considérée comme un scalaire étant donné que les objets sont supposés se déplacer le long du vecteur aligné avec leur orientation.

Une représentation graphique du problème du DATMO est illustrée dans la figure 2.4 ci-dessous où l'état d'une cible est noté $S_t \stackrel{\text{def}}{=} (X_t, v_t)$ bien qu'il arrive que des paramètres supplémentaires soient utilisés pour décrire la forme, le type ou le mouvement des objets. Il est à noter que durant le filtrage, les cibles sont généralement considérées indépendantes les unes des autres bien que certaines relations existent en réalité. Cette hypothèse d'indépendance permet au problème d'être divisé en plusieurs sous-problèmes plus simples et de réduire ainsi la complexité algorithmique. Les relations entre les cibles sont réintroduites comme contraintes imposées à chaque étape de filtrage comme nous le verrons plus loin.



(a) pour toutes les cibles (b) pour une seule cible

Figure 2.4 : Représentation graphique du problème du DATMO

On rappelle que la détection et le suivi des objets mobiles dans un environnement dynamique (DATMO) est une composante principale de la perception. Cette dernière utilise plusieurs grandeurs en entrée et après traitement produit plusieurs grandeurs composant le modèle de l'environnement. Ces variables seront illustrées dans la figure 2.5 ci-dessous.

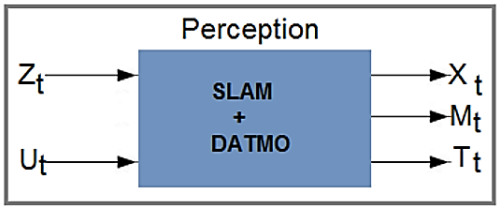


Figure 2.5 : Grandeurs intervenant dans le module de perception

Dans ce schéma, Z_t représente les données des capteurs accumulées jusqu'à l'instant t. U_t est la mesure du mouvement, habituellement identifiée aux données fournis par les odomètres ou par une centrale inertielle (IMU) :

$$Z_t = \{z_0, z_1, z_2, \dots, z_t\}$$

$$U_t = \{u_0, u_1, u_2, \dots, u_t\}$$

Où z_i est la mesure du capteur et u_i est la mesure du mouvement à l'instant i.

La sortie X_t représente l'ensemble des états estimés (position, orientation, ...) à l'instant t. Donc, si x_i est l'état du robot à l'instat i, on aura alors :

$$X_t = \{x_0, x_1, x_2, \dots, x_t\}$$

M_t est la carte de l'environnement comportant l'esemble des objets statiques détectés jusqu'à l'instant t. Elle est aussi représentée comme suit :

$$M_t = \{m_0, m_1, m_2, \dots, m_k\}$$

Enfin, l'ensemble des pistes jusqu'à l'instant t est l'ensemble des objets o_i en mouvement détectés :

$$T_t = \{o_0, o_1, o_2, \dots, o_m\}$$

Où m est le nombre d'objets dynamiques suivis à l'instant t.

2.3 Modélisation de l'environnement

Vu son importance, plusieurs méthodes de résolution du problème du DATMO ont été développées par les chercheurs. Anna Petrovskaya propose dans [7] de classifier ces méthodes en trois grandes classes d'approches suivant le type de modélisation de l'environnement adopté. L'environnement du robot peut être représenté de différentes manières. Dans la littérature, on retrouve principalement les modélisations suivantes : « *Cluster Based Models* », les *modèles géométriques* et les « *Grid Based Models* ».

2.3.1 Cluster Based Model

C'est l'une des représentations d'objets les plus simples et les plus communes [7] souvent utilisée avec les capteurs de distance (les télémètres laser notamment). Elle consiste à regrouper à partir des données brutes des capteurs, les mesures susceptibles d'appartenir à un même objet.

Dans cette représentation, la position de l'objet utilisée dans les calculs est considérée comme étant la moyenne métrique du regroupement des points (aussi appelé *point cloud*).

La simplicité de cette méthode n'est pas sans conséquences. En effet, il existe plusieurs situations conflictuelles qui peuvent induire le robot en erreur et causer une disfonctionnement dans la phase de perception, ce qui se répercutera inévitablement sur l'étape de décision.

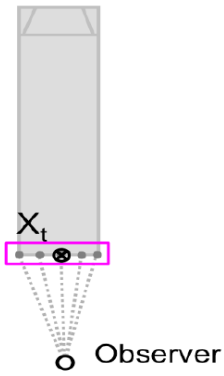


Figure 2.6 : Représentation d'un obstacle par la méthode du regroupement de points.

Le principal inconvénient est que cette représentation dépend directement du point duquel on observe l'objet vu que la moyenne géométrique ne tiens pas compte des parties non observées de l'objet. Une fois que ces parties non observées auparavant deviennent visibles suite au changement du point d'observation par exemple, la moyenne géométrique change et ce changement est perçu en conséquence comme étant un mouvement de l'objet observé (mouvement fantôme).

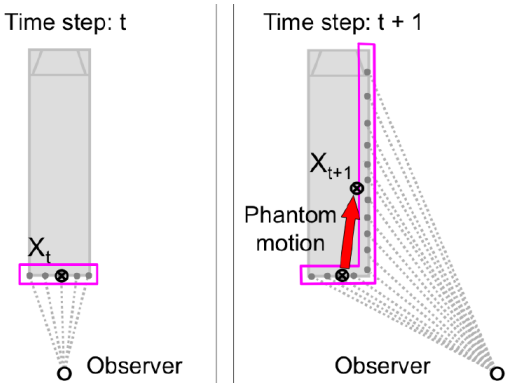


Figure 2.7 : Inconvénient de la représentation par *clustering* dû aux parties non observées de l'objet.

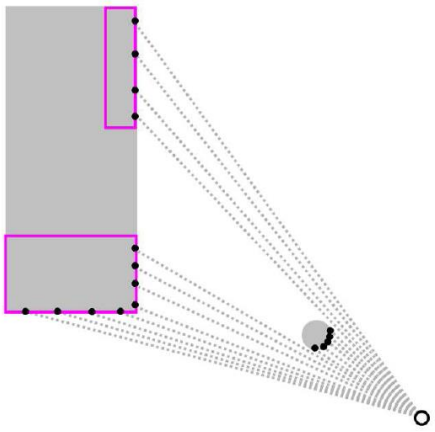


Figure 2.8 : Inconvénient de la représentation par *clustering* dû à l'alignement des objets.

Un autre inconvénient majeur et plus fréquent se présente lorsqu'un ou plusieurs objets sont les uns derrières les autres. L'algorithme de regroupement des points sera induit en erreur et donnera plusieurs « *point cloud* » appartenant au même objet comme l'illustre la figure 2.8 ci-dessus.

Ces inconvénients peuvent être évités en utilisant les modèles géométriques décrient dans la section suivante.

2.3.2 Modèles géométriques

Comme alternative à la méthode de regroupement des points, les objets peuvent être représentés par des formes géométriques. Les rectangles sont la forme géométrique la plus commune bien que les cercles et les ellipses sont également utilisés.

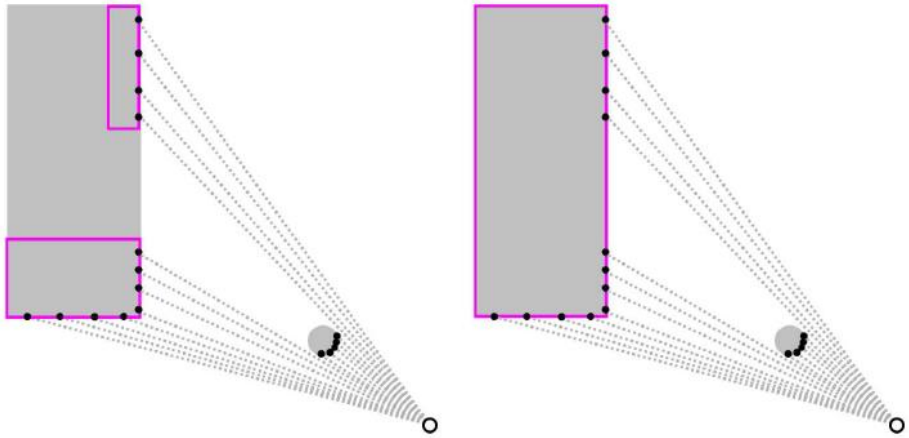


Figure 2.9 : Avantage de la modélisation avec des formes géométriques. A droite : sans modèle géométrique. A gauche : avec modèle géométrique.

Pour avoir des résultats plus précis, la forme géométrique des obstacles peut varier suivant leurs natures ou suivant la nature du regroupement de points les représentants. On peut par exemple représenter les piétons par des cercles, les voitures par des carrées et les bus par des rectangles larges.

Le problème relatif à l'apparition des *mouvements fantômes* est traité en attribuant à chaque forme géométrique un point appelé « *ancre* ». Cette méthode est détaillée dans [7].

2.3.3 Grid Based Models

Malgré les améliorations apportées par le modèle géométrique, une représentation plus robuste des environnements encombrés où de nombreux objets y manœuvrent en même temps est nécessaire.

C'est dans cette optique là que la représentation de l'environnement par une grille d'occupation fut introduite par A. Elfes dans [9] en 1989. Cette représentation se base sur la

division de la scène perçue par le robot en un ensemble de cellules de taille uniforme (souvent) ou variable. Elle attribue à chaque cellule une probabilité d'occupation notée $P(O_i)$ qui représente la probabilité qu'un objet occupe la cellule *i*. Cette probabilité est calculée à partir des informations fournies par les capteurs et sa valeur traduit l'état de la cellule :

$$P(O_i): \begin{cases} > 0.5 & si\ la\ cellule\ i\ est\ occupée \\ = 0.5 & l'état\ de\ la\ cellule\ i\ est\ inconnu \\ < 0.5 & si\ la\ cellule\ i\ est\ libre \end{cases}$$

Dans le contexte des véhicules intelligents, la grille d'occupation est la plus utilisée pour représenter l'environnement extérieur. Elle est particulièrement adaptée à la fusion des données fournies par plusieurs capteurs.

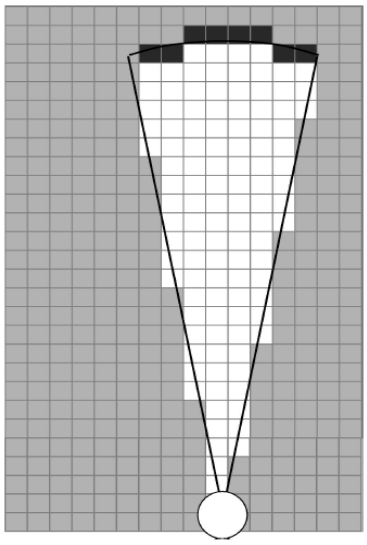


Figure 2.10 : Représentation de l'environnement avec une grille d'occupation.

Cette dernière modélisation se distingue des deux modélisations précédentes par le fait que des notions telles que « *objet* » et « *track* » n'existent plus. Elles sont remplacées par la notion de cellules et on ne s'intéresse qu'à l'estimation de leur occupation et de leur vélocité en faisant abstraction de la forme des objets (obstacles) qu'elles peuvent contenir.

En se basant sur les trois types de modélisations de l'environnement présentés dans ce paragraphe, les différentes approches de résolution du problème du DATMO peuvent être classifiées comme cela est illustré dans le paragraphe suivant.

2.4 Les différentes approches de détection et de suivi d'objets mobiles (DATMO)

Ce paragraphe a pour but de présenter les trois classes d'approches du DATMO telle qu'elles sont décrites par Anna Petrovskaya et al. dans [7].

Plusieurs méthodes de résolution du problème DATMO ont été développées par les chercheurs lors des dix dernières années et chaque méthode présente des particularités. Ces méthodes peuvent être classées en trois classes d'approches : l'approche traditionnelle, l'approche basée sur un modèle géométrique et l'approche basée sur la grille d'occupation.

- La première est l'*approche traditionnelle*. Elle comprend la segmentation de données, l'association de données et le filtrage en utilisant généralement des filtres de Kalman. Les travaux relatifs à cette classe d'approches visent principalement à améliorer les techniques de reconnaissance des formes.
- La seconde classe d'approches est basée sur un *modèle géométrique*. Elle effectue l'inférence directement sur les données des capteurs sans segmentation ni association de données. Cette approche repose sur l'utilisation des filtres non-paramétriques pour l'inférence.
- La troisième est l'approche basée sur la *grille d'occupation*. Elle repose sur une subdivision de l'environnement dynamique en un certain nombre de cellules. Cette approche sera étudiée en détail dans les paragraphes suivants.

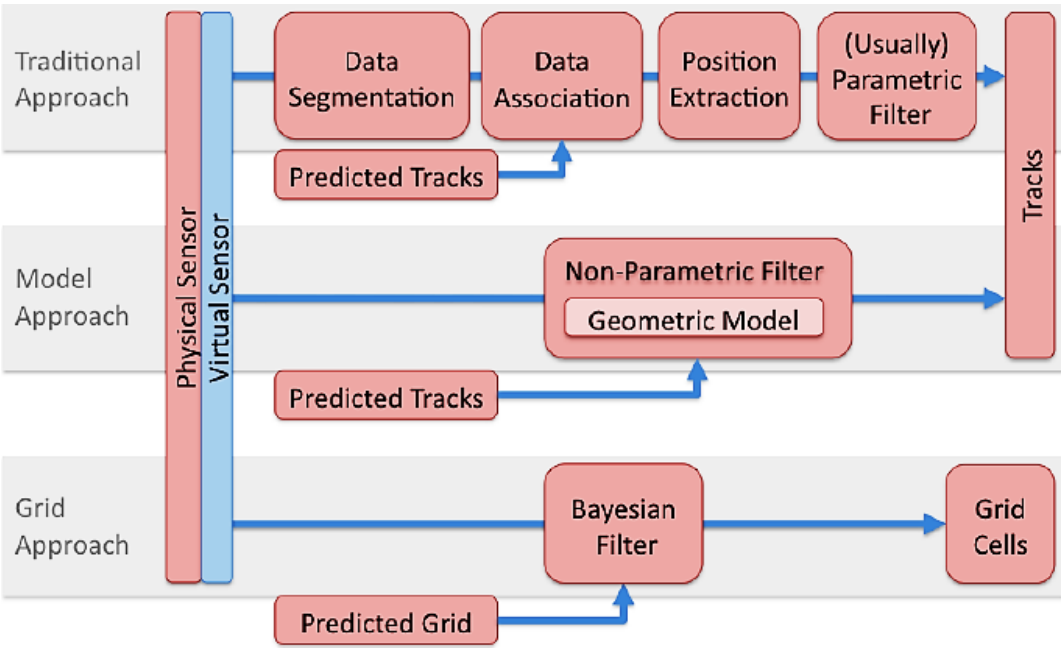


Figure 2.11 : Les trois classes d'approches du DATMO. Les carrées rouges sont obligatoires tandis que les carrés bleus ne le sont pas [7].

La figure 2.11 ci-dessus résume les trois approches précédentes. Chaque approche commence par une acquisition de données des capteurs (à gauche) et produit des « *tracks* » d'objets mobiles (à droite) en sortie.

Les deux premières approches représentent deux pistes de recherche qui travaillent sur le développement de techniques permettant la formation d'objets à partir des données des capteurs (brutes ou traitées) puis d'utiliser un filtre (paramétrique pour la première approche et non-paramétrique pour la seconde approche) pour le *tracking* de ces objets. Ces deux approches visent à reconstruire la scène puis de raisonner sur chaque objet formé. Ceci peut être considéré comme un avantage dans le cas où une modélisation des obstacles sous forme d'objets est nécessaire.

Dans plusieurs autres applications, la notion d'objet n'est pas nécessaire. Par exemple, en évitement d'obstacle, savoir quelle est la forme exacte des obstacles présents dans l'environnement n'est pas une information importante vu que l'objectif principal est de ne pas les heurter. C'est dans cette optique là que la troisième approche fut développée.

La troisième approche repose sur un principe différent des deux premières. En effet, l'environnement est modélisé avec une grille d'occupation et la notion d'objet n'existe plus, seules les cellules sont considérées. On dit que c'est une représentation des obstacles de bas niveau. Cette dernière approche s'avère être la plus intéressante et la plus prometteuse. Elle sera décrite en détail dans les sections qui vont suivre.

Ceci dit, il faut noter que ces trois approches partagent la même structure (voir la figure 2.12 ci-dessous). L'état des objets est récursivement estimé en utilisant un filtre bayésien qui utilise les informations accumulées pour réaliser une inférence sur l'état des objets ou des cellules.

Le filtrage est composé d'une *mise à jour dynamique* (*dynamic update*) qui observe les évolutions survenues entre deux pas de temps, et d'une *mise à jour des mesures* (*measurement update*) qui incorpore les données capteur les plus récentes. Ensuite, plusieurs filtres bayésiens sont utilisés pour le suivi des états de chaque objet mobile (ou cellule). La mise à jour du filtre est directement suivie par une phase de gestion des pistes (*tracks*) durant laquelle les *tracks* sont créés et supprimés. Le pipeline résultat est résumé dans la figure ci-dessous.

Figure 2.12 : Pipeline DATMO typique [7].

Notons qu'on distingue deux types de filtres bayésiens : les filtres paramétriques et les filtres non paramétriques. Les filtres paramétriques regroupent les différentes variantes du filtre de

Kalman (KF) et représentent *la croyance* avec une fonction paramétrique : une Gaussienne. Lorsque la croyance n'est pas gaussienne, des approximations peuvent être faites en utilisant le filtre de Kalman étendu (EKF) ou le filtre de Kalman inodore (UKF). Les filtres non-paramétriques regroupent les filtres à particules (PF) et *l'histogram filter* (HF). Ils représentent la croyance avec un ensemble de points positionnés aléatoirement dans le cas du PF (ils sont appelés *particules*) et d'une manière déterministes dans le cas du HF.

Les filtres paramétriques ont l'avantage d'être efficaces vu que leur complexité est polynomiale. Cependant, ils ne sont pas adaptés pour représenter des croyances complexes et leur performance s'appauvrit considérablement si une bonne estimation à priori de l'état n'est pas disponible. En revanche, les filtres non paramétriques ont l'avantage d'être en mesure de représenter n'importe quelle croyance mais leur complexité algorithmique est exponentielle par rapport à la dimension de l'état [7].

Ainsi, plusieurs approches du DATMO furent développées dans le but d'exploiter les avantages de chaque méthode de filtrage. Ces approches seront détaillées dans le paragraphe suivant.

2.4.1 L'approche traditionnelle

Les approches traditionnelles correspondent à la branche supérieure du pipeline de la figure 2.11 précédente. Elles reposent généralement sur des variantes du filtre de Kalman bien que récemment, on retrouve dans certains travaux l'utilisation du filtre à particules. La caractéristique distinctive de l'approche traditionnelle est qu'une grande partie du traitement est faite avant l'application du filtre.

On détaillera dans ce qui suite les différentes étapes qui constituent le pipeline du DATMO (illustré dans la figure 2.12) dans le cas des approches traditionnelles ainsi que leurs principales variantes.

2.4.1.1 Estimation dynamique

Dans le cas des approches traditionnelles, cette étape se divise en deux sous-étapes : la segmentation et la prédiction.

a) Segmentation de données

Il s'agit de la phase où des objets sont créés à partir des données fournies par les capteurs. La méthode la plus populaire est connue sous le nom de *clustering*, qu'on a vu précédemment et qui considère tout regroupement de points comme étant un obstacle à part entière.

Plusieurs autres méthodes furent développées, en se basant sur l'idée de base du *clustering,* dans le but d'améliorer les résultats obtenus. Par exemple, des méthodes de reconnaissance de lignes et de cercles et les autres formes standards ont été développées dans [10] et [11]. On

retrouve dans [11] une méthode de reconnaissance des jambes humaines. On retrouve également dans [12], [13] des travaux qui se sont focalisés sur la reconnaissance d'êtres humains.

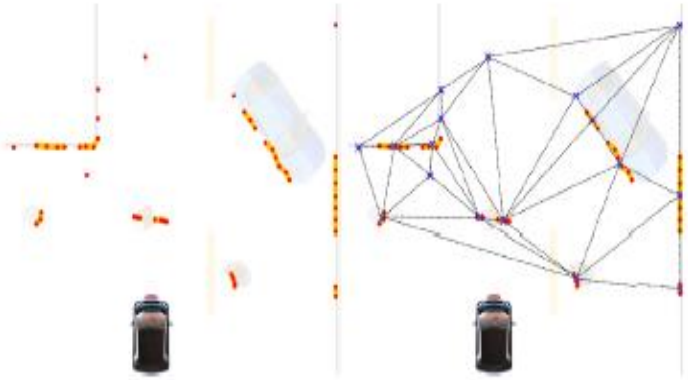


Figure 2.13 : Utilisation de la méthode CRFs (Conditional Random Fields) pour modélisation des obstacles.

b) Filtrage prédictif

Dans cette étape, il est nécessaire de choisir au préalable un modèle dynamique de mouvement (*motion model*) régissant le déplacement des obstacles. Plusieurs types de modèles ont été développés, on citera :

- Le modèle dynamique Brownien : C'est le modèle de mouvement le plus simple, il est généralement utilisé dans un environnement extérieur pour modéliser le mouvement des piétons caractérisé par des changements soudains de la direction de mouvement.
- Le modèle dynamique à accélération nul (ou *à vitesse constante*) : C'est le modèle le plus courant en robotique mobile. Il suppose que la vitesse de l'objet reste constante entre deux pas de temps.
- Le modèle dynamique *bicyclette* (*Bicycle Motion Model*) : Il est semblable au modèle à accélération constante dans la mesure où on considère une vitesse constante entre deux pas de temps, hormis que la vitesse de l'objet est représentée avec deux composantes : une vitesse linéaire et une vitesse angulaire.

Le modèle dynamique est choisi suivant l'environnement dans lequel se trouve le robot. Puis, un filtre prédictif est utilisé pour déterminer l'état de chaque objet aux instants suivants. Le filtre le plus utilisé dans les approches traditionnelles est le filtre de Kalman et ses différentes variantes.

Cette étape dépend fortement du formalisme de filtrage qu'on désire adopter. Si on se place dans le cadre du formalisme bayésien (ce qui est souvent le cas en DATMO), le problème de

prédiction ainsi que le modèle dynamique adopté seront formulés en utilisant la théorie des probabilités.

2.4.1.2 Mise à jour en utilisant les données des capteurs

La mise à jour par les données des capteurs est une étape importante vu qu'elle permet de valider les résultats de la prédiction. Elle consiste à mettre en correspondance les mesures obtenues avec la liste des objets prédits. La méthode est expliquée et détaillée dans [14].

Cette étape se divise en deux sous-étapes : le fenêtrage (*gating*) et l'association de données.

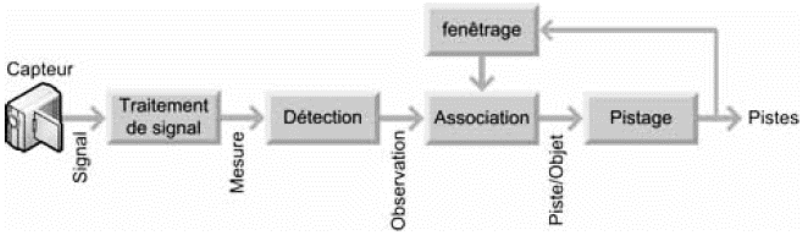


Figure 2.14 : Structure d'un système de suivi selon l'approche traditionnelle

a) Fenêtrage

Le but du fenêtrage (ou *gating*) est de réduire le temps de traitement de la phase d'association de données en minimisant le nombre d'hypothèses d'association possibles entre les pistes déjà existantes et les observations courantes.

Elle consiste en un test de proximité entre les objets prédits et les mesures (voir la figure 2.15 ci-dessous) ayant pour but d'éliminer les calculs inutiles relatifs à des associations non probables. Techniquement, il s'agit de chercher pour chaque piste, ses observations voisines localisées dans sa fenêtre d'association préalablement définie [14].

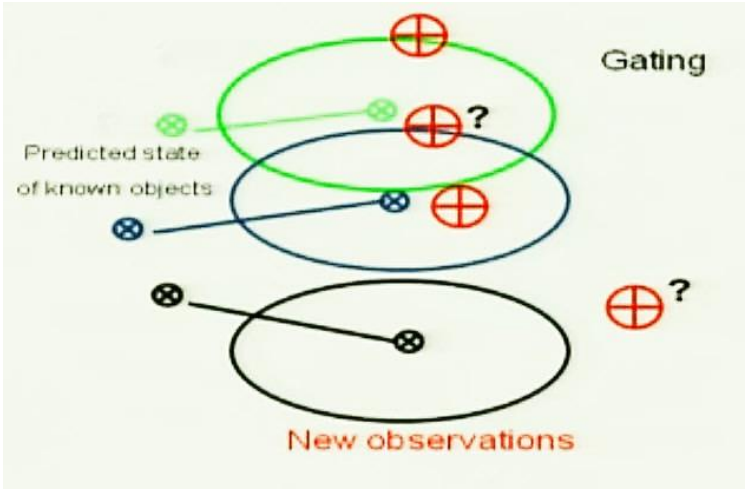


Figure 2.15 : L'étape de fenêtrage dans l'approche traditionnelle du DATMO.

b) L'association de données

Cette phase a pour but d'associer les données obtenues des capteurs avec la liste d'objets prédits en tolérant une certaine erreur de prédiction modélisée sous forme de fenêtre. Plusieurs méthodes ont été développées pour résoudre ce problème à grande complexité, notamment :

- La méthode du plus proche voisin « *Nearest Neighbor (NN)* » [15] : Cette méthode est la plus simple et la plus utilisée en pratique. Elle consiste à associer une piste avec l'objet le plus proche de sa position prédite à condition que chaque objet soit associé à, au plus, une seule piste.
- La méthode du *Global Nearest Neighbor* (GNN) : Cette méthode est une amélioration de la méthode précédente et se base sur une minimisation du coût globale d'association entre observations et prédictions (ou sur la maximisation d'une vraisemblance dans le cas d'une formulation probabiliste).

En pratique, deux ou plusieurs regroupements de points peuvent provenir d'un seul et même objet comme c'est le cas dans la détection des deux jambes d'un piéton par exemple ou la détection de plusieurs parties séparées d'un même objet à cause d'occultation. Ce genre de problème ne peut être résolu par les méthodes présentées en haut et nécessite des associations de plusieurs objets contribuant à la mise à jour d'une même piste [15]. C'est dans cette optique que les méthodes d'association probabilistes de données furent développées, on citera notamment :

- PDAF (*Probabilistic Data Association Filter*) : une méthode probabiliste bayésienne [16] [17] qui prend en considération toutes les hypothèses d'association possibles entre un objet prédit et les observations appartenant à son voisinage.
- JPDAF (*Joint Probabilistic Data Association Filter*) : cette approche a été développée pour pallier aux problèmes rencontrés avec la première méthode (PDAF) quand deux ou plusieurs fenêtres d'association se chevauchent [18] [19].
- MHT (*Multi Hypothesis Tracking*) : dans [20] une méthode d'association de données dite à hypothèses multiples est proposée. Dans cette méthode, au lieu de considérer les objets prédits et d'essayer de leur associer des regroupements de points, elle considère les observations et génère trois hypothèses donnant une information sur l'état de ces observations. Un calcul de probabilité de chaque hypothèse est ensuite entamé pour déterminer l'état d'appartenance de ces observations.

2.4.1.3 Gestion des pistes ou « tracks management »

Cette étape se décompose en quatre phases :

Imposition des contraintes relatives aux objets modélisés

Lors de l'étape de prédiction, les relations de dépendance entre les objets ne sont pas prises en considération et chaque objet est traité de façon indépendante, alors qu'en réalité, plusieurs contraintes les relient les uns aux autres. A titre d'exemple, on citera que deux objets ne peuvent être au même endroit en même moment, un objet ne peut pas disparaitre ... Ces contraintes sont vérifiées lors de l'étape de gestion objets.

Existence des objets

Cette phase se résume généralement au calcule d'un score d'existence qui augmente lorsque l'objet prédit se rapproche des mesures obtenues, et diminue dans le cas contraire.

Ajout de nouveaux objets

Après la phase de mise à jour par les données des capteurs, souvent, des regroupements de points ne correspondent avec aucun des objets déjà modélisés. Dans ce cas, on crée de nouveaux *tracks*. Cette création peut utiliser plusieurs trames de données à des instants antérieurs afin de bien modéliser l'objet.

Suppression des objets

Lorsque le score d'existence d'un objet va en dessous d'un certain seuil, l'objet sera supprimé.

2.4.2 Approche basée sur un modèle géométrique

L'approche basée sur un modèle géométrique correspond à la branche du milieu de la figure 2.11. Ces méthodes fonctionnent en modélisant directement le capteur physique et les objets en mouvement en utilisant un modèle physique du capteur et des modèles géométriques des objets. Dans ces méthodes, la quasi-totalité du traitement est assuré par le filtre lui-même.

Les étapes de segmentation et d'association de données ne sont pas nécessaires vu que la modélisation géométrique permet la reconstitution des objets à partir des données provenant des capteurs.

Le défi majeur de cette approche est de rendre le filtre suffisamment efficace pour répondre aux exigences de la navigation autonome notamment en termes de temps de calcul. Pour ce faire, plusieurs filtres ont été utilisés notamment : *Rao-Blackwellized particle filter* (RBPF) [21], *Scaling Series Algorithm* [22] et l'approche MCMC [23].

L'étape de gestion des *tracks* est réalisée de la même manière que dans les approches traditionnelles décrites plus haut.

2.4.3 Approche basée sur la grille d'occupation

L'approche du DATMO basée sur la grille d'occupation correspond à la branche inférieure de la figure (2.9) précédente. Le paradigme de filtrage bayésien est appliqué pour l'estimation et le filtrage de la grille d'occupation, ce qui permet de profiter pleinement de plusieurs avantages notamment :

- L'estimation de la vitesse de chaque cellule de la grille et donc une modélisation dynamique de l'environnement ;
- Vu que cette approche prend en considération l'historique des mesures des capteurs, elle permet ainsi la rétention de l'information sur l'occupation des régions occlues de la scène ;
- Cette approche peut éliminer par filtrage les erreurs survenues sur une trame de mesure.

En outre, dans l'approche traditionnelle présentée précédemment, les problèmes d'association de données et d'estimation de l'état des objets mobiles sont fortement couplés. Ainsi, une erreur dans une de ces deux étapes conduit à des résultats complètement erronés. L'approche basée sur la grille d'occupation permet de décomposer ce couplage en évitant le problème de l'association de données dans la mesure où cette étape est accomplie dans un niveau de représentation supérieur.

On retrouve dans la littérature plusieurs méthodes visant à adapter le filtrage bayésien aux grilles d'occupation. La méthode la plus populaire est incontestablement le *filtre d'occupation bayésien* (*Bayesian Occupancy Filter*, BOF) présenté pour la première fois dans [4], amélioré ensuite dans [24] en considérant des environnements à caractère fortement dynamique. C'est cette méthode qui sera illustrée dans le présent mémoire et implémentée sur la plateforme robotique expérimentale du CDTA (Robucar).

Le Filtre d'Occupation Bayésien (BOF)

Le filtre d'occupation bayésien (BOF) fonctionne, comme tout filtre bayésien, selon une boucle de *prédiction-estimation*. Cette structure, illustrée dans la figure ci-dessous, estime à chaque pas de temps l'état d'occupation et la vitesse de chaque cellule en combinant l'étape de prédiction (historique) à l'étape d'estimation (intégration des nouvelles mesures). La prise en compte de l'historique des mesures des capteurs permet une estimation robuste même dans le cas des environnements fortement dynamique (il permet de traiter les objets temporaires, les occlusions et les problèmes de détection).

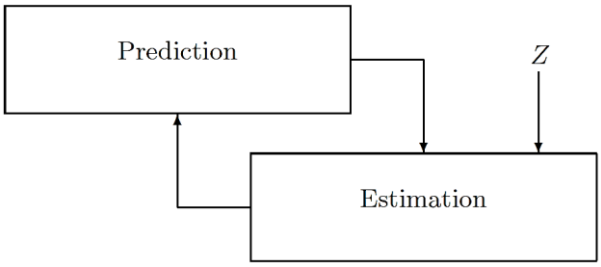


Figure 2.16 : Boucle de prédiction-estimation d'un filtre d'occupation bayésien.

L'intérêt du filtrage par le filtre d'occupation bayésien est d'estimer à chaque pas de temps l'occupation et les distributions probabilistes de vitesse de chaque cellule de la grille.

a) Prédiction

Un filtre d'occupation bayésien à 4 dimensions a été proposé par Christophe Coué dans [6] où chaque cellule possède un vecteur d'état à 4 composantes, à savoir : deux pour représenter la position de la cellule en coordonnées cartésiennes et deux autres pour sa vitesse. Ceci suppose donc qu'on connait d'emblée la vélocité de chaque cellule et l'étape de prédiction aura pour unique but la prédiction de l'état d'occupation des cellules.

Une amélioration à cette méthode fut introduite dans [25] où une grille d'occupation à 2 dimensions fut présentée, ce qui a permis d'estimer et de prédire l'état d'occupation et les vitesses possibles pour chaque cellule de la grille. Puis, dans [26], l'estimation des vitesses est assurée d'une manière implicite et déduite à partir de de l'estimation des antécédents de chaque cellule.

b) Estimation

Comme tout filtre bayésien, l'étape d'estimation permet de valider les résultats de la prédiction en utilisant les données des capteurs. Concrètement, cette étape calcule des croyances a postériori de l'état de chaque cellule (occupation et vitesse).

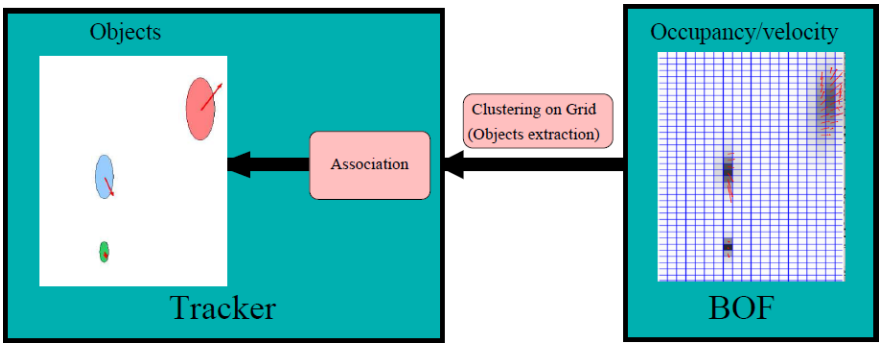


Figure 2.17 : Utilisation de l'algorithme « Fast Clustering-Tracking » avec un filtre d'occupation bayésien

Une étape de gestion des pistes peut être ajoutée au filtrage bayésien. En effet, on rappelle que dans une représentation de l'environnement par une grille d'occupation, les objets n'existent

pas et la segmentions et l'association de données n'ont donc plus aucune utilité. La reconstruction des objets et leur suivi se fera donc dans une *couche* supérieure.

On trouve à titre d'exemple l'algorithme illustré dans la figure ci-dessus et appelé « *Fast Clustering-Tracking* » présenté dans [26]. Cet algorithme assure une reconstitution des objets par regroupement de cellules répondant à certaines conditions.

2.5 Conclusion

Le DATMO a toujours été et continue d'être un problème qui suscite l'intérêt des chercheurs en Automatique et en intelligence artificielle durant la dernière décennie. Nous avons vu que plusieurs méthodes ont été développées, chacune ayant ses spécificités et répondant à une représentation de l'environnement donnée.

Nous avons particulièrement insisté sur l'approche basée sur les grilles d'occupation du fait qu'elle présente de nombreux avantages. On citera :

- L'incertitude est explicitement prise en compte en utilisant le paradigme du raisonnement probabiliste qui sera présenté au chapitre suivant ;
- Le problème complexe d'association de données présent dans les approches classiques est évité en utilisant la représentation par grille d'occupation qui ne considère plus des notions telles que *track* ou *objet* ;
- L'utilisation de plusieurs capteurs pour une meilleure représentation de l'environnement est possible. Plusieurs techniques de fusion de données se basant sur les grilles d'occupation existent.
- Le BOF permet une représentation riche en information sur la dynamique de l'environnement et procure ainsi une robustesse aux problèmes d'occultation, d'apparition et de disparition d'objets ;

Pour toutes ces raisons, et en se basant sur ce qui a été dit précédemment sur les deux autre approches, on peut conclure que le *grid-based* DATMO est l'approche la plus prometteuse. C'est d'ailleurs cette approche-là qui sera présentée en détail dans le chapitre suivant et qui sera ensuite implémentée sur la plateforme robotique expérimentale « Robucar » du CDTA.

Chapitre 3

Grid-based DATMO

3.1 Introduction

La navigation autonome dans des environnements encombrés est un problème qui doit être traité avec attention et une grande prudence. Le véhicule autonome doit percevoir non seulement l'environnement stationnaire, mais aussi des objets dynamiques tels que les véhicules et les piétons qui s'y trouvent. Pour chaque objet mobile, le véhicule autonome doit identifier son emplacement et sa vitesse de sorte qu'il puisse prédire sa position aux instants suivants.

Un tel cadre d'utilisation requiert que la représentation de l'environnement soit robuste aux défaillances des capteurs et aux problèmes d'incertitudes sur les modèles utilisés. De plus, il existe toujours des facteurs non pris en compte (variables cachées) qui font que le phénomène et le modèle ne se comportent jamais exactement de la même manière. En DATMO et en robotique en générale, la nécessité de gérer cette dimension incertaine est incontournable.

Dans ce chapitre, nous commencerons par traiter le problème de l'incertitude en présentant les concepts de base de la programmation bayésienne ; une méthode de modélisation et d'inférence permettant de prendre en compte les incertitudes. Puis, nous reviendrons plus en détail sur les grilles d'occupation et leur utilisation dans le cadre du *grid-based* DATMO.

3.2 Programmation bayésienne : concepts de base

Ce paragraphe a pour but de présenter le minimum théorique et les notions de base nécessaires pour la compréhension des principes du calcul bayésien. Nous nous placerons ainsi dans le cadre d'une théorie de raisonnement probabiliste appelée *Probability as Logic* (PaL) proposée par le physicien E.T Jayes dans [27] et qui considère le calcul des probabilités comme une généralisation de la logique formelle booléenne.

3.2.1 Les probabilités comme alternative à la logique classique

Olivier Lebeltel affirme dans [28] que la modélisation d'un phénomène réel est irrémédiablement incomplète, plusieurs variables non prises en compte dans le modèle font en sorte que le phénomène et le modèle n'ont jamais le même comportement. Tout système robotique est confronté à cette difficulté centrale et doit raisonner avec un modèle incomplet par rapport à son environnement pour percevoir, inférer, décider et agir efficacement.

E.T Jayes propose dans [27] une théorie permettant d'étende la logique « *classique* » à des propositions dont la vérité n'est pas connue avec certitude et de formaliser la notion de raisonnement plausible. Il a été démontré sous certaines hypothèses que l'unique façon de manipuler la notion de plausibilité est définie par la *théorie des probabilités*. Il est à noter qu'ici la notion de probabilité est prise dans son sens *subjectiviste* pour exprimer un état de connaissance représentant les informations dont on dispose sur le phénomène en question, par opposition au sens dit *fréquentiste* où elle caractérise la fréquence d'une mesure physique donnée [4].

Une méthode de programmation des robots, appelée *programmation bayésienne*, fut ensuite présentée dans [28] et sera étudiée dans la section suivante. Elle repose sur les principes de l'inférence et de l'apprentissage bayésien et traite formellement l'incertitude des informations et l'incomplétude des modèles utilisés.

3.2.2 Définitions fondamentales

3.2.2.1 Probabilité d'une proposition logique

Une proposition logique est un énoncé qui peut être soit vrai soit faux. La probabilité d'une proposition A notée $P(A)$ définie le degré de certitude accordé à sa véracité.

En réalité, on ne peut accorder un degré de certitude à la proposition A qu'au vu d'un ensemble de connaissances préalables qu'on dénotera C (les mesures d'un capteur par exemple). Il est alors plus convenable de parler d'une probabilité conditionnelle de la proposition A sachant les connaissances contextuelles C dont nous disposons et qu'on notera : $P(A|C)$.

On notera par $A \wedge B$ la conjonction des propositions logiques A et B et par $A + B$ leur disjonction. De même, on utilisera $\bar{A}$ et $\bar{B}$ pour noter leurs négations respectives.

La conjonction est souvent utilisée en DATMO notamment pour définir des distributions conjointes et des distributions marginales.

3.2.2.2 Règles de calcul

Etant donné les deux propositions logiques A et B, on définit les deux règles fondamentales à la base de toute inférence probabiliste, à savoir la règle du produit et la règle de normalisation.

Règle du produit

La règle du produit donne la probabilité d'une conjonction :

$$P(A \wedge B) = P(A)P(B|A) = P(B)P(A|B)$$

Règle de normalisation

La règle de normalisation exprime le fait que la somme des probabilités d'une proposition et de sa négation est égale à 1 :

$$P(A) + P(\bar{A}) = 1$$

On considère que les deux règles précédentes comme des postulats formant la base à partir de laquelle tous les calculs de probabilités dérivent. En effet, en partant de ces deux règles, nous pouvons déduire la probabilité d'une disjonction :

$$P(A + B) = P(A) + P(B) - P(A \wedge B)$$

Dans le cas où A et B sont indépendants, cette dernière règle s'écrit :

$$P(A + B) = P(A) + P(B)$$

3.2.2.3 Variable discrète

Par définition, une variable discrète X est un ensemble de propositions logiques x_i exhaustives et qui s'excluent mutuellement.

En pratique, on s'intéresse souvent à assigner des probabilités aux différentes valeurs numériques d'une variable discrètes X. Nous utiliserons alors la notation $P(X)$ pour désigner la distribution de probabilité sur la variable discrète X.

Les deux règles fondamentales énoncées précédemment (la règle du produit et la règle de marginalisation) restent valident et s'écrivent dans pour deux variables X et Y comme suit :

$$P(X \wedge Y) = P(X)P(Y|X) = P(Y)P(X|Y)$$

$$\sum_X P(X) = 1$$

3.2.2.4 Variable continue

De la même manière, on démontre dans [27] que les deux règles fondamentales énoncées précédemment restent valident dans le cas des variables continues et s'écrivent respectivement pour deux variables X et Y comme suit :

$$P(X \wedge Y) = P(X)P(Y|X) = P(Y)P(X|Y)$$

$$\int P(X)dX = 1$$

La distribution $P(X \wedge Y)$ est appelée *distribution conjointe* de X et Y.

3.2.2.5 Marginalisation et distributions marginales

On définit pour deux variables numériques X et Y la *distribution marginale* de X par rapport à Y comme suit :

$P(X) = \sum_Y P(X \wedge Y) = \sum_Y P(Y)P(X|Y)$ dans le cas discret ;

$P(X) = \int P(X \wedge Y)\mathrm{dY} = \int P(Y)P(X|Y)dY$ dans le cas continue.

3.2.3 Formule de Bayes

Le fondement de la théorie bayésienne s'est basé sur le théorème d'inversion des probabilités connu sous le nom du théorème de Bayes (1963) approfondit par Laplace (1774).

Une transformation mathématique directe de la règle du produit permet d'obtenir la formule dite de Bayes :

$$P(X|Y) = \frac{P(X)P(Y|X)}{P(Y)} = \frac{P(X)P(Y|X)}{\sum_X P(X)P(Y|X)}$$

Ce théorème permet de définir la manière dont nous changeons nos croyances subjectives (a priori) concernant un paramètre en y ajoutant des données expérimentales et obtenir ainsi des croyances a posteriori.

3.2.4 Principe de l'inférence bayésienne

On définit l'inférence comme étant un processus qui permet de passer d'une ou plusieurs assertions affirmées comme vraies et appelés prémisses, à une nouvelle assertion qui en est la conclusion en vertu de sa liaison avec les premières. Lorsque l'inférence prend une forme probabiliste, on parle d'*inférence bayésienne*.

Bien qu'elle ne soit pas très récente, l'inférence bayésienne est une méthode d'inférence très prometteuse et permet de déduire la probabilité d'un événement en se s'appuyant sur celles d'autres événements déjà évaluées. Elle repose principalement sur le théorème de l'inversion des probabilités énoncé par Bayes.

Ainsi, le but de l'analyse Bayésienne est de fournir une inférence probabiliste concernant le phénomène étudié ou plus exactement de faire une inférence sur un paramètre θ qui le caractérise au vue d'observations où subsiste une incertitude. En *grid-based* DATMO, on s'intéresse à inférer la position et la vitesse de chaque cellule composant la grille d'occupation en se basant sur les données fournies par les capteurs.

L'approche Bayésienne est caractérisée par le fait qu'elle considère le paramètre θ à estimer comme étant inconnu et aléatoire et définie par une loi de probabilité appelée *loi à priori*. Cette loi représente ce qu'on sait et ce qu'on ne sait pas sur θ avant d'observer les données. En statistique, on considère que c'est la meilleure façon de résumer l'information disponible ou manquante sur le paramètre et de cette manière toute l'information inexacte et incertaine sera incorporée dans le modèle.

Les *croyances a priori* sont ensuite actualisées à la lumière des nouvelles informations disponibles (appelées aussi fonction de vraisemblance) pour obtenir une *croyance a posteriori*. Cette transformation se fait à l'aide du théorème de Bayes qu'on peut énoncer comme suit :

$$Loi\ a\ posteriori \propto Vraissemblance \times Loi\ a\ priori$$

Cette relation peut également être exprimée comme suit :

$$Connaissances\ mise\ à\ jour = \frac{(Ce\ qu'on\ sait\ déjà) \times (Ce\ que\ nous\ apporte\ les\ données)}{Constante}$$

En DATMO, la connaissance à mettre à jour est la distribution de probabilité conjointe de l'état d'occupation d'une cellule et sa vélocité calculée à partir d'une distribution conjointe a priori et des dernières mesures fournies par les capteurs.

3.2.4.1 Distribution a priori

Comme mentionné précédemment, la distribution a priori résume toute l'information disponible ou manquante sur le paramètre à estimer. Le choix de cette distribution est crucial pour l'analyse bayésienne et peut se faire d'une manière *subjective* ou *objective*. Le premier choix, dit subjectif, repose sur l'information disponible sur le paramètre et obtenue des opinions d'experts. Cette information est exprimée par une loi de probabilité dite « distribution a priori informative ».

Dans le cas où on ne dispose pas d'informations a priori, la théorie d'inférence bayésienne peut toujours être appliquée et dans ce cas-là, nous faisons le choix d'une distribution de probabilité dite *objective* exprimée par une loi de probabilité non-informative (dite aussi *vague*) qui modélise le mieux cette absence d'information. Un bon exemple de loi non-informative est la loi uniforme.

3.2.4.2 Distribution a posteriori

La distribution a posteriori donne l'information dont nous disposons sur le paramètre θ estimé après l'observation. Elle est la combinaison de la loi a priori et la vraisemblance qui représente l'information « captée ». On remarque donc d'emblée l'effet d'un mauvais choix de la loi a priori sur l'estimation.

La figure 3.1 ci-dessous illustre le résultat d'une inférence bayésienne sur un paramètre θ. Nous remarquons clairement que la distribution a posteriori :

- sera proche de la vraisemblance si l'a priori est non-informatif ;
- sera proche de la distribution a priori si cette dernière est informative ;

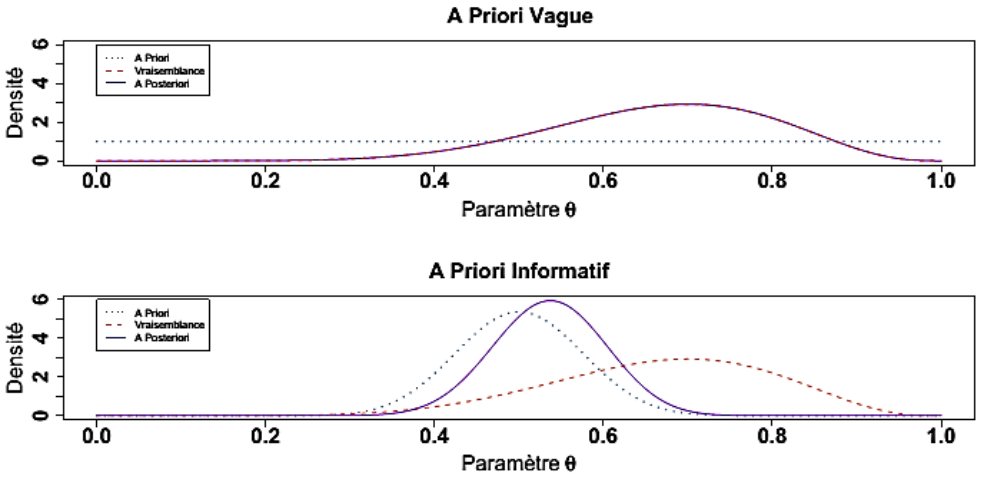


Figure 3.1 : Résultat de l'inférence bayésienne en fonction du choix de l'a priori

Un résultat important est donc à souligner ; le choix d'une distribution a priori informative n'est pas dominé par la vraisemblance et affecte donc d'une manière significative le résultat de l'inférence. Inversement, lorsque la distribution a priori est non-informative, son impact sur la distribution a posteriori est minimal. Ce choix donc est crucial et doit se faire avec la plus grande attention.

3.3 Programmation bayésienne

Dans [28] Olivier Lebeltel et son équipe présentent *la programmation bayésienne* comme une méthode originale et générique de programmation des robots fondée sur l'inférence et l'apprentissage bayésien.

Un programme bayésien est défini comme étant un moyen de spécifier une famille de distributions de probabilités. L'objectif étant d'utiliser de telles spécifications pour définir des tâches diverses en robotique, de résoudre des problèmes d'inférence et de commander effectivement un robot [28].

Dans la suite de ce paragraphe, nous présenterons les principes de base de cette méthode. Elle sera ensuite utilisée pour implémenter les différents programmes composant le *grid-based* DATMO.

D'un point de vue formel, un programme bayésien est composé de deux parties principales : *« la description »* et *« la question »*.

3.3.1 La description

Olivier Lebeltel explique que la description est une notion centrale du formalisme de programmation bayésienne. C'est l'objet formel faisant la synthèse des connaissances préalables et des valeurs des paramètres libres.

Une description est exprimée sous forme d'une distribution de probabilité conjointe $P(V_1 \wedge ... \wedge V_n | C \wedge Z)$ d'un ensemble de variables $V_1, ..., V_n$ spécifiées par le programmeur et déterminées à la lumière des connaissances préalables notées C et d'un ensemble de données expérimentales Z. Ainsi, l'ensemble des connaissances devant être fourni se trouve circonscrit par les connaissances préalables C et le jeu de données expérimentales Z.

La programmation d'un robot à l'aide d'une description se fait en trois phases :

- Spécification des connaissances préalables ;
- Identification des formes des distributions probabilistes et les valeurs des paramètres ;
- Utilisation de la description (la question).

3.3.1.1 Spécification

Au cours de cette phase, le programmeur doit énoncer clairement et explicitement l'ensemble des connaissances dont il dispose et celles qui résultent d'un jeu de données expérimentales. Cette étape est particulièrement délicate et est subdivisée en trois points :

- Le choix des variables pertinentes sur lesquelles sera définie la distribution conjointe ;
- Décomposer la distribution conjointe des variables retenues sous forme d'un produit de distributions élémentaires en utilisant les deux règles fondamentales énoncées plus haut. Des hypothèses d'indépendance entres les variables sont utilisées pour simplifier davanatge le produit résultant ;
- Définir les formes paramétriques associées à chacune de ces distributions.

Le choix des variables pertinentes

Commençons par le premier point qui consiste à définir les connaissances préalables dites structurelles. Ces connaissances permettent de définir l'ensemble des variables pertinentes $V_1, ..., V_n$ sur lesquelles va porter la description et de spécifier ensuite pour chacune d'elle son domaine de valeur et le nombre d'états possibles. Toute autre variable est supposée non pertinente pour le problème considéré.

Le choix d'une décomposition de la distribution conjointe

La dépendance entre les variables $V_1, ..., V_n$ définies précédemment est dénotée formellement par leur probabilité conjointe $P(V_1 \wedge ... \wedge V_n | C \wedge Z)$. Cette forme mathématique est une distribution de probabilité sur « n » dimensions et n'est souvent pas facile à spécifier. Ainsi, la structure de dépendance permet de décrire la probabilité conjointe sous la forme d'une décomposition en produit de distributions plus simples en utilisant notamment la règle du produit.

Supposons, à titre d'exemple, que nous avons retenu trois variables pertinentes pour une description : V_1, V_2 et V_3. L'application de la règle du produit nous permet d'écrire la probabilité conjointe de treize manières différentes.

D'un point de vue purement mathématique, les treize écritures sont équivalentes. En pratique, nous exprimons la probabilité conjointe sous une certaine forme de connaissance qui nous permet de mieux décrire les distributions élémentaires et les hypothèses d'indépendance entres les variables considérées. Pour les trois variables précédentes, on peut écrire :

$$P(V_1 \wedge V_2 \wedge V_3) = P(V_3)P(V_2|V_3)P(V_1|V_2 \wedge V_3)$$

L'expression des relations de dépendances ou d'indépendance entre les variables permet de réduire fortement les dimensions des termes apparaissant dans la décomposition et réduire ainsi la complexité des calculs. Si l'on suppose dans notre exemple que les variables V_1 et V_2 sont indépendantes sachant la valeur de V_3, on aura :

$$P(V_1 \wedge V_2 \wedge V_3) = P(V_3)P(V_2|V_3)P(V_1|V_3)$$

Le choix des formes paramétriques

Une fois la décomposition de la distribution de probabilité conjointe obtenue, il faut associer à chacun de ses termes une forme paramétrique. Cela consiste à fournir des *a priori* sur les valeurs des distributions de probabilités et la manière dont ces valeurs seront modifiées par l'expérience.

Les formes paramétriques choisies sont en générale des lois de probabilité classique, le plus souvent, des lois uniformes ou des lois normales.

3.3.1.2 Identification

Afin d'achever notre discription, il reste à fixer les valeurs numériques des différents paramètres intervenant dans les formes paramètriques choisies dans l'étape précédente, comme les écarts-type et les moyennes des distributions gaussiennes.

Ces paramètres sont généralement obtenus sur la base d'un jeu de données expérimentales provenant d'une observation spécifique ou sont, dans certains cas, fixés a priori.

3.3.2 La question

Au terme des phases de spécification et d'identification, nous disposons d'une description complètement définie. La phase d'utilisation va consister à mettre en œuvre les descriptions par le biais de questions probabilistes.

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. Si l'ensemble des variables est noté E on définit E_q, E_c et E_i respectivement le sous-ensemble des variables de E dont on cherche la distribution (*searched variables*), le sous-ensemble des variables de E dont on connait

les valeurs (*known variables*), le sous-ensemble des variables de E dont on ignore les valeurs (*free variables*).

Concrètement, cela consiste à chercher la distribution de probabilité donnée par la forme :

$$P(V_k \wedge ... \wedge V_l | V_m \wedge ... \wedge V_n)$$

Où $E_q = \{V_k, ..., V_l\} \neq \emptyset$ $E_c = \{V_m, ..., V_n\}$ et $E_i = \{V_0, ..., V_p\}$ sont les sous-ensembles de E définis précédemment et formant une partition de E.

La figure 3.2 ci-dessous résume les différentes étapes d'un programme bayésien. Nous utiliserons cette représentation pour résumer chacun des programmes que nous définirons dans le reste de ce document.

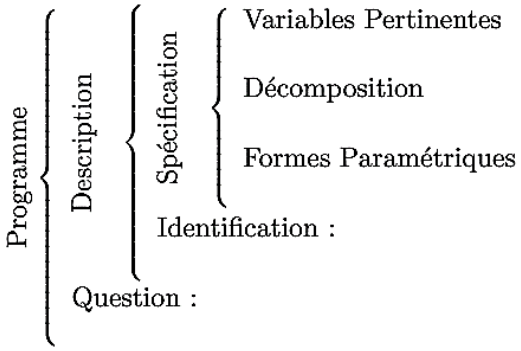


Figure 3.2 : Canevas d'une représentation d'un programme bayésien

3.3.3 Inférence

La distribution de probabilité $P(V_k \wedge ... \wedge V_l | V_m \wedge ... \wedge V_n)$ peut être écrit sous le forme suivante en utilisant les règles du produit et de la marginalisation énoncées précédemment. D'abord commençons par appliquer la règle de Bayes :

$$P(V_k \wedge ... \wedge V_l | V_m \wedge ... \wedge V_n) = \frac{P(V_k \wedge ... \wedge V_l \wedge V_m \wedge ... \wedge V_n)}{P(V_m \wedge ... \wedge V_n)}$$

En appliquant ensuite la règle de marginalisation, on peut exprimer le dénominateur et le numérateur de ce quotion en fonction de la distribution conjointe $P(V_1, ..., V_n)$ qu'on sait calculer.

$$P(V_k \wedge ... \wedge V_l | V_m \wedge ... \wedge V_n) = \frac{\sum_{V_0 ... V_p} P(V_k \wedge ... \wedge V_l \wedge V_m \wedge ... \wedge V_n \wedge V_0 \wedge ... \wedge V_p)}{\sum_{\substack{V_k ... V_l \\ V_0 ... V_p}} P(V_k \wedge ... \wedge V_l \wedge V_m \wedge ... \wedge V_n \wedge V_0 \wedge ... \wedge V_p)}$$

Cette dernière expression est une résolution purement symbolique et ne tient pas compte des simplifications qui peuvent apparaitre et des connaissances préalables fournies lors de la phase de spécification.

Ces simplifications peuvent être purement mathématiques (si par exemple une somme portant sur une variable V et il existe des termes dans lesquels V n'apparait pas) ou symboliques (dans

le cas où on ne s'intéresse qu'au maximum de la distribution de probabilité, il n'est pas nécessaire de calculer le dénominateur de l'expression de la question formulée précédemment).

Le résultat de l'inférence fournit une distribution de probabilité sur les variables recherchées. Ce résultat est rendu « effectif » en choisissant une valeur particulière pour chacune des variables, comme le choix de la valeur la plus probable par exemple, ou des heuristiques de décision plus sophistiqués.

3.4 Grille d'occupation : concepts de base

Ce paragraphe a pour but de présenter l'outil de base de notre modélisation de l'environnement du robot : la grille d'occupation. Nous rappelons que dans le contexte des véhicules intelligents, la grille d'occupation est le modèle le plus utilisé pour la construction de carte de l'environnement.

On définit la grille d'occupation (*Occupancy Grid* en anglais) comme un découpage de l'espace dans lequel évolue le robot mobile en un ensemble de cellules. Le but est d'estimer la probabilité qu'un objet quelconque (humain, voiture, murs, ...) occupe chacune de ces cellules en utilisant les mesures fournies par les capteurs.

3.4.1 Hypothèse d'indépendance des cellules

Afin de réduire la complexité de l'estimation de la grille complète, nous formulons une hypothèse fondamentale d'indépendance entre les cellules de la grille d'occupation. Chaque cellule est supposée indépendante des autres cellules. Ainsi, estimer la grille complète revient à appliquer N fois l'estimation d'une seule cellule (N étant le nombre de cellules composant la grille d'occupation).

Cette hypothèse permet essentiellement d'estimer la grille complète en un temps raisonnable. Cependant, les cartes obtenues souffrent d'un manque de précision notamment lorsque celles-ci sont construites à partir des mesures fournies d'un sonar [4].

En pratique, cette hypothèse est utilisée dans quasiment toutes les applications de *mapping* et les grilles obtenues se révèlent bien satisfaisantes. Ainsi, nous ferons cette hypothèse dans toute la suite de ce document.

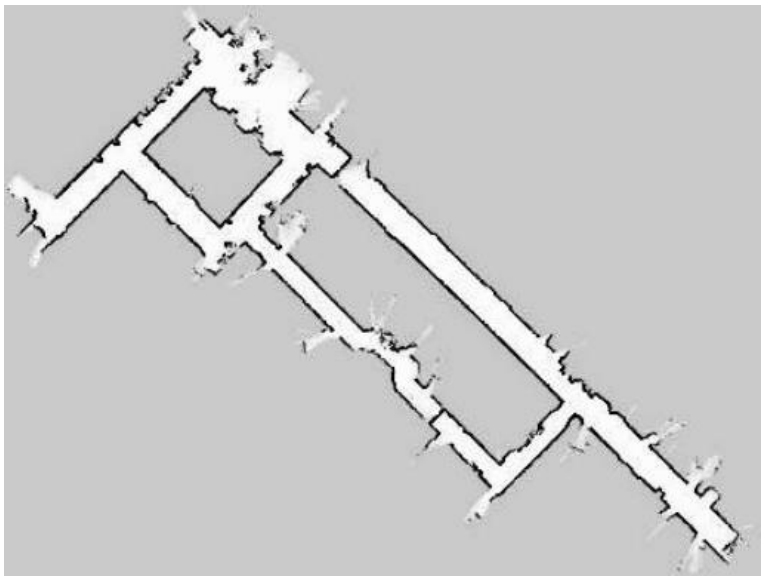


Figure 3.3 : Une grille d'occupation construite à partir de données laser. Les zones en noir correspondent aux cellules avec une probabilité d'occupation élevée. Les cellules en blanc correspondent à l'espace libre.

3.4.2 Estimation d'une grille d'occupation

Nous présentons dans cette section le mécanisme qui permet d'estimer une grille d'occupation à partir d'un jeu de données expérimentales provenant des capteurs. Dans la littérature, plusieurs méthodes de construction des grilles d'occupation ont été proposées, on citera notamment les travaux de :

- Moravec et Elfes : « *Probabilistic framework* » [29] ;
- Matthies et Elfes : « *Bayesian framework* » [30] ;
- Thrun : « *Neural network based approach* » [31] ;
- Konolige : « *Enhanced Bayesian framework* » [32];
- Thrun : « *Forward modeling approach* » [33].

Chacune des techniques citées ci-dessus vise à améliorer la précision des cartes obtenues ainsi qu'à démineur le temps d'estimation des cellules étant donné que la difficulté majeure des grilles d'occupation est le nombre important de cellules à estimer.

La figure 3.4 ci-après illustre les résultats d'une étude comparative réalisée par Thomas Collins dans [34] entre les cinq techniques citées plus haut. Cette étude s'est basée sur plusieurs critères de comparaison dont le nombre de fausses détections (positives ou négatives), la précision des cartes obtenues, le temps de calcul, ..., etc.

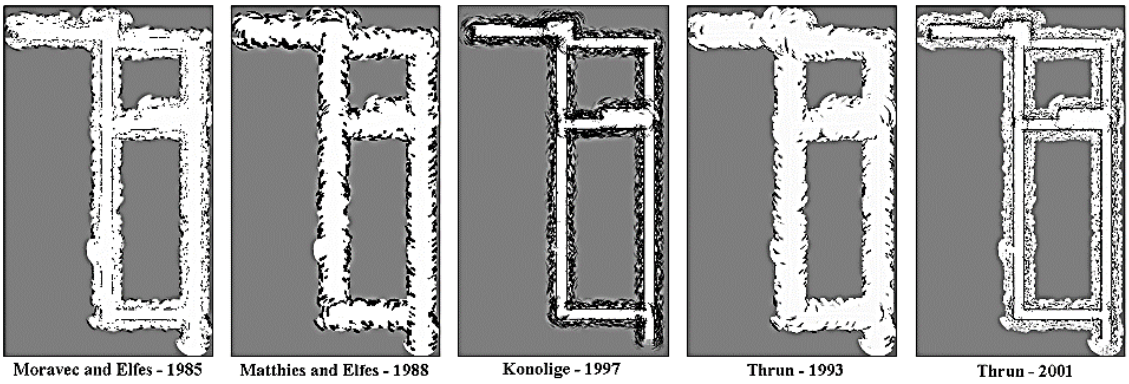


Figure 3.4 : Etude comparative des méthodes de construction de grilles d'occupation d'une scène de $25cm \times 25cm$

Il est à noter que l'estimation peut se faire sous la forme d'un programme bayésien mais vu la simplicité du problème, on peut adopter une démarche plus directe. C'est d'ailleurs le choix de S. Thrun dans [35] où il a illustré une démarche d'estimation récursive simple et peu coûteuse en termes de puissance de calcul. Pour ces mêmes raisons, nous ferons le choix d'une estimation directe (en utilisant le théorème de Bayes) dans le cadre de notre travail. Cette démarche sera expliquée ci-après.

On rappelle que le problème d'estimation de la grille d'occupation consiste à calculer la distribution de probabilité a posteriori sur les cellules de la grille :

$$P(m|z_{1:t})$$

Où m est la carte de l'environnement à estimer et $z_{1:t}$ est le jeu de données expérimentales donnés par les capteurs entre l'instant initial et l'instant t.

Nous rappelons que la grille d'occupation est un partitionnement de l'espace perçu par le robot en plusieurs cellules généralement de taille égale. Ainsi, en notant m_i la cellule d'indice i, on peut écrire que la grille d'occupation m est définie comme suit :

$$m = \{m_i\}$$

On attache à chaque cellule m_i un état d'occupation binaire qui spécifie si la cellule est occupée ou libre. Ainsi, les notations $P(m_i = 1)$ ou $P(m_i = Occ)$ font référence à la probabilité que la $i^{ème}$ cellule soit occupée. Par abus de notation, nous utiliserons dans la suite de ce document la notation $P(m_i)$.

Sous l'hypothèse d'indépendance des cellules énoncée précédemment, estimer une grille d'occupation revient à estimer la probabilité d'occupation de chacune des cellules en se basant sur les mesures des capteurs. Ainsi, le problème d'estimation de la distribution de probabilité a posteriori $P(m|z_{1:t})$ est décomposé en un ensemble de sous-problèmes indépendants ; à savoir l'estimation de $P(m_i|z_{1:t})$ pour chaque cellule m_i de la grille. Cette décomposition en sous-

problèmes n'est pas sans conséquences. En effet, elle ne nous permet pas de représenter les dépendances entre les cellules voisines qui sont, dans certains cas, particulièrement importantes.

La distribution de probabilités a posteriori sur les cellules de la grille est alors approximée par le produit de ses marginales :

$$P(m|z_{1:t}) = \prod_i P(m_i|z_{1:t})$$

Dans la suite de ce document, nous allons réduire le problème d'estimation de la grille d'occupation au problème d'estimation de la probabilité d'occupation de chacun des cellules la composant. En appliquant le théorème de Bayes avec $z_{1:t-1}$ comme connaissances a priori, on obtient :

$$P(m_i|z_{1:t}) = \frac{P(z_t|m_i \wedge z_{1:t-1}) \times P(m_i|z_{1:t-1})}{P(z_t|z_{1:t-1})}$$

En supposant que z_t est indépendant de $z_{1:t-1}$ on obtient :

$$P(m_i|z_{1:t}) = \frac{P(z_t|m_i) \times P(m_i|z_{1:t-1})}{P(z_t|z_{1:t-1})}$$

En appliquant le théorème de Bayes sur le terme $P(z_t|m_i)$ on obtient :

$$P(z_t|m_i) = \frac{P(m_i|z_t) \times P(z_t)}{P(m_i)}$$

En injectant ce terme dans l'équation précédente :

$$P(m_i|z_{1:t}) = \frac{P(m_i|z_t) \times P(z_t) \times P(m_i|z_{1:t-1})}{P(m_i) \times P(z_t|z_{1:t-1})}$$

En prenant en compte le fait que l'état de chaque cellule est binaire, on déduit de manière analogue l'équation suivantes :

$$P(\overline{m}_i|z_{1:t}) = \frac{P(\overline{m}_i|z_t) \times P(z_t) \times P(\overline{m}_i|z_{1:t-1})}{P(\overline{m}_i) \times P(z_t|z_{1:t-1})}$$

Où $P(\overline{m}_i|z_{1:t})$ représente la probabilité que la cellule m_i soit libre i.e. $P(m_i = 0|z_{1:t})$ ou $P(m_i = emp|z_{1:t})$.

En divisant les deux dernières équations obtenues, on obtient :

$$\frac{P(m_i|z_{1:t})}{P(\overline{m}_i|z_{1:t})} = \frac{P(m_i|z_t) \times P(\overline{m}_i) \times P(m_i|z_{1:t-1})}{P(\overline{m}_i|z_t) \times P(m_i) \times P(\overline{m}_i|z_{1:t-1})}$$

Finalement, en prenant en compte le fait que : $P(\overline{m}_i) = 1 - P(m_i)$ on obtient :

$$\frac{P(m_i|z_{1:t})}{1 - P(m_i)} = \frac{P(m_i|z_t)}{1 - P(m_i|z_t)} \times \frac{1 - P(m_i)}{P(m_i)} \times \frac{P(m_i|z_{1:t-1})}{1 - P(m_i|z_{1:t-1})}$$

On définit la notation $Odds(x)$ comme suit :

$$Odds(x) = \frac{P(x)}{1 - P(x)}$$

Ainsi, la dernière équation peut être écrite sous la forme :

$$Odds(m_i|z_{1:t}) = Odds(m_i|z_t) \times Odds(m_i)^{-1} \times Odds(m_i|z_{1:t-1})$$

Cette forme récursive nous permet d'écrire en utilisant une transformation *logOdds* :

$$logOdds(m_i|z_{1:t}) = logOdds(m_i|z_{1:t-1}) + logOdds(m_i|z_t) - logOdds(m_i)$$

L'usage de la notion *logOdds* est particulièrement avantageux pour calculer de manière efficace les probabilités d'occupation.

Notons qu'il est possible de retrouver, à partir de la notation *logOdds*, les probabilités d'occupation en utilisant la formule ci-après :

$$P(x) = \frac{Odds(x)}{1 + Odds(x)}$$

Il est à noter qu'en pratique, on prend souvent la probabilité d'occupation a priori d'une cellule $P(m_i)$ égale à 0.5 Ainsi, le terme $\frac{1-P(m_i)}{P(m_i)}$ peut-être simplifié de l'équation et on aboutit donc à la forme simple suivante :

$$logOdds(m_i|z_{1:t}) = logOdds(m_i|z_{1:t-1}) + logOdds(m_i|z_t)$$

Enfin, il ne reste plus qu'à décrire comment la probabilité d'occupation d'une cellule pour une *seule mesure* du capteur $P(m_i|z_t)$ est calculée. Ce terme dépend directement du type de capteur utilisé (télémètre laser, capteur à ultrason, …).

Dans le cas d'un télémètre laser, cette probabilité est assez simple à calculer et ne dépend que de la distance entre le télémètre et la cellule considérée. Pour une cellule m_i de la grille, traversée par le $n^{ème}$ faisceau du laser qu'on notera $z_{t,n}$ et provenant de l'observation à l'instant t notée z_t, nous définissons la fonction $dist(x_t, m_i)$ comme étant la distance entre le télémètre laser (positionné en x_t) et le centre de la cellule m_i considérée.

Lorsque cette distance est inférieure à la distance mesurée par $z_{t,n}$, la cellule est supposée libre et la cellule où s'arrête le faisceau est supposée occupée. Ceci peut être formulé comme suit :

$$P(m_i|z_{t,n}) = \begin{cases} p_{prior} & z_{t,n} \text{ est à sa valeur maximale} \\ p_{prior} & m_i \text{ n'est pas couverte par } z_{t,n} \\ p_{occ} & |z_{t,n} - dist(x_t, m_i)| < r \\ p_{free} & z_{t,n} \geq dist(x_t, m_i) \end{cases}$$

Où r est la résolution de la grille d'occupation. En outre, on doit avoir $0 \leq p_{free} < p_{prior} < p_{occ} \leq 1$. La figure 3.5 ci-dessous illustre le principe de cette méthode.

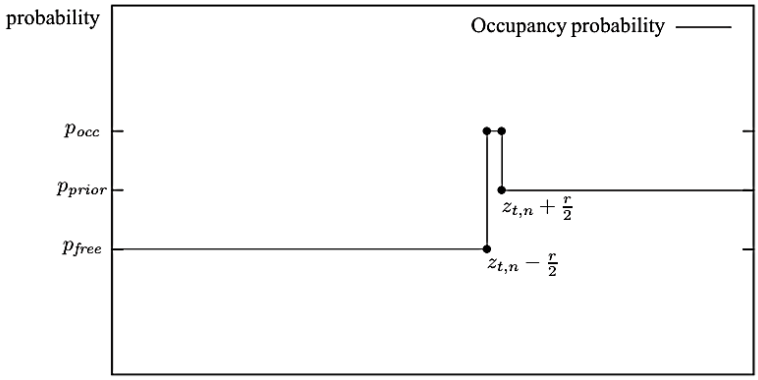


Figure 3.5 : La probabilité $P(m_i | z_{t,n})$ en fonction de la distance $dist(x_t, m_i)$.

Un exemple de construction d'une grille d'occupation est illustré dans la figure 3.6 ci-dessous où le « Robucar » modélise la scène devant lui (salle des robots au CDTA) en utilisant un télémètre laser SICK LMS 511.

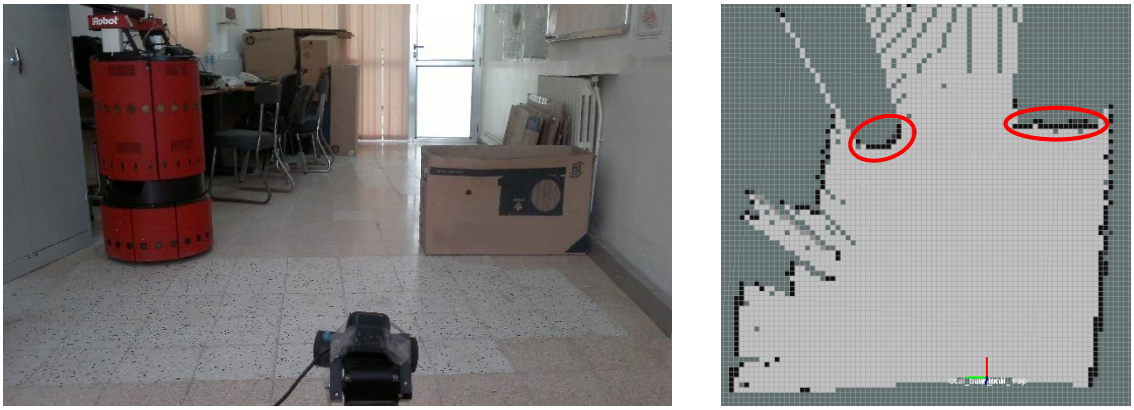


Figure 3.6 : Grille d'occupation modélisant la salle des robots du CDTA

Remarquons que les cellules entourées en rouge (marquage ajouté manuellement) représentent respectivement le robot « *iRobot B21r* » à gauche et un carton vide à droite.

Nous soulignons également que l'état de certaines cellules se trouvant loin du robot est inconnue (probabilité d'occupation égale à 0.5). Ceci est dû au fait que, pour ce test, la résolution angulaire du laser a été fixée à 1° et par conséquent, certaines cellules se trouvant loin du télémètre ne sont traversées par aucun des faisceaux émit. Nous reviendrons plus en détail sur ce point au chapitre suivant.

3.4.3 Utilisation de plusieurs capteurs

Une des grandes qualités des grilles d'occupation réside dans sa capacité à réaliser une fusion des données provenant de plusieurs capteurs. Jusque-là, nous n'avons traité que le cas où un unique capteur fournit une observation de l'environnement. Dans le cadre des véhicules autonomes, plusieurs capteurs sont utilisés afin d'augmenter la robustesse de l'observation et la justesse de la caractérisation de l'environnement. Ce raisonnement est calqué sur les êtres

humains et les animaux qui utilisent naturellement plusieurs systèmes de perception en même temps (les cinq sens). Par exemple, un prédateur caché derrière un arbre ne peut être détecté grâce à la vue, mais peut l'être grâce à l'odorat ou l'ouïe.

En outre, l'utilisation de plusieurs capteurs permet de s'affranchir des limitations propres à chaque capteur. A titre d'exemple, un télémètre laser est sensible aux conditions atmosphériques (pluie, neige, brouillard, ...) tandis qu'un radar est insensible à ces conditions. De plus, la redondance de l'information permet d'augmenter la précision de la perception mais surtout d'assurer une meilleure robustesse aux défaillances de fonctionnement d'un des capteurs.

3.4.4 Limites de l'estimation statique des grilles

Dans le formalisme des grilles d'occupation, il est facile de tenir compte de la succession des observations capteur lors de la construction de la grille. Cette qualité peut être utilisée pour représenter des environnements contenant des objets mobiles. Ainsi, en plus de contenir des informations sur la probabilité d'occupation, chaque grille contient des informations sur les caractéristiques dynamiques des cellules la composant.

Parmi les travaux phares qui ont été fait, on citera la méthode proposée par Arbuckle [36] et qui consiste à attacher à chaque cellule des informations sur le temps d'occupation de celles-ci. Ainsi, on peut classer les cellules suivant leur temps d'occupation et séparer les cellules occupées en permanence et qui correspondent à des objets immobiles (murs, meubles, ...) des autres cellules. Il est également possible de déduire les cellules correspondant aux lieux et passages fréquents.

D'autres techniques de modélisation des scènes dynamiques ont été développées par la suite, on citera notamment la méthode présentée par Prassler [37] et dont l'objectif est de reconstruire les caractéristiques dynamiques des objets en comparant des grilles d'occupation correspondant à des instants successifs.

Ceci dit, aucune des méthodes précédentes n'est robuste aux défaillances des capteurs et aux occultations d'objets. Dans sa thèse [4], Coué utilise une grille d'occupation de dimension 4 qui modélise les positions et les vitesses relatives des objets (ensemble de cellules) par rapport au véhicule en utilisant un capteur de position et un capteur de vitesse. Avec un exemple simple, il montre que l'estimation de la grille en tenant compte des seules dernières observations capteur ne permet pas d'être robuste aux occultations d'objets (lorsqu'un objet cache un autre) ni aux pannes des capteurs, et en général on ne peut rien dire en dehors du champ d'observation des capteurs. Il démontre ainsi qu'il est nécessaire de tenir compte de l'historique des observations et de la dynamique de la scène pour avoir une représentation complète de l'environnement. C'est dans cette optique là que le *filtre d'occupation bayésien* (en anglais *Bayesian Occupancy Filter*, BOF) fut développé et proposé par Christophe Coué pour la première fois dans [4].

3.5 Filtre d'Occupation Bayésien (BOF)

3.5.1 Le filtre bayésien

Le filtrage Bayésien, appelé aussi *estimation Bayésienne récursive* dans certaines références, est une approche très prometteuse d'estimation des systèmes dynamiques applicable sur une grande classe de systèmes réels et dans différents domaines (dont la robotique et notamment en *tracking*).

Estimer l'état d'un système dynamique à partir d'un ensemble d'observations bruitées est un problème classique dont la solution probabiliste [38] débouche sur la notion de *filtre Bayésien.*

Dans cette approche, on suppose que le système dont on veut estimer l'état est un système de Markov d'ordre 1. Cela signifie qu'on n'a pas besoin de garder une trace de toutes les observations précédentes et que toute l'information dont on a besoin est contenue dans l'état du système à l'instant précédent. En d'autres termes, on peut dire qu'entre deux pas de temps consécutifs, l'évolution du système est modélisable par une fonction bruitée de la forme :

$$S^k = f(S^{k-1}, \omega^k)$$

Dans cette équation S désigne l'état du système et les indices entiers k et $k-1$ désignent l'instant d'estimation et enfin ω^k désigne le bruit associé à la fonction f. Il faut noter ici que la fonction f peut ne pas être linéaire.

En pratique, l'état S^k du système est caché et ne peut être observé directement mais à travers un *vecteur d'observation* noté Z^k qui relate l'état du système entaché d'un *bruit d'observation.* Il est alors nécessaire de définir un modèle capteur comme suit :

$$Z^k = h(S^k, v^k)$$

Dans cette équation, v^k désigne le bruit associé à la fonction h qui peut être aussi une fonction non linéaire qui modélise l'incertitude de la mesure z^k de l'état du système S^k.

D'un point de vue probabiliste, définir les deux fonctions précédentes connaissant les bruits associés ω^k et v^k est équivalent à définir les distributions de probabilités respectives :

$$P(S^k|S^{k-1})$$

$$P(Z^k|S^k)$$

A partir de ces deux modèles, le filtre Bayésien estime de manière récursive, à chaque instant k, l'état du système $P(S^k|z^{0:k})$ connu sous le nom de *distribution de probabilité a postériori.* Cette estimation se fait sous forme d'une boucle *prédiction-estimation* comme montré sur la figure 3.7 ci-dessous :

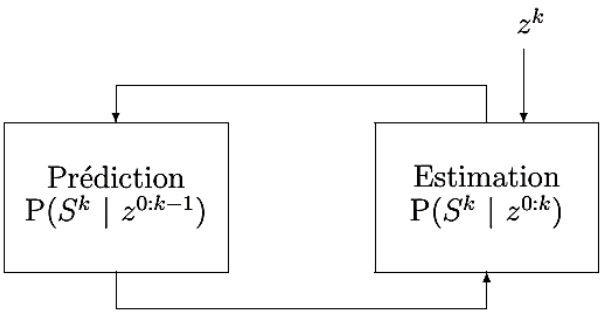


Figure 3.7 : Le filtre Bayésien vu comme une boucle prédiction-estimation

La phase de prédiction projette vers l'instant k le résultat de la phase d'estimation à l'instant $k-1$ grâce au modèle dynamique $P(S^k|S^{k-1})$. La phase d'estimation à l'instant k corrige cette prédiction en utilisant la nouvelle observation capteur z^k [4].

On appelle le résultat de la phase de prédiction *distribution a priori* et de même, le résultat de la phase d'estimation est appelé *distribution a posteriori*.

Une solution exacte de cette propagation récursive de la distribution de probabilité a postériori existe pour certains cas particuliers. A titre d'exemple, le filtre de Kalman est une solution optimale lorsque les fonctions f et h sont linéaires et les bruits ω et v sont Gaussiens.

En pratique, il est assez rare d'être placé dans ces conditions et une solution analytique ne peut être déterminée directement. En général, on fait plusieurs approximations afin d'obtenir une solution approximative.

C'est notamment le cas avec les grilles d'occupation où l'état du système est donné par l'état d'occupation de chaque cellule de la grille. Ainsi, les conditions requises pour appliquer une résolution exacte telle que le filtre de Kalman ne sont pas toujours vérifiées. De plus, vu la structure particulière du modèle (grille d'occupation) et les contraintes de temps-réel imposées à tout système robotique, il faut inévitablement développer un nouveau concept de filtre bayésien adapté aux grilles d'occupation.

3.5.2 Filtre bayésien adapté aux grilles d'occupation

L'adaptation des *filtres bayésiens* aux grilles d'occupation permet de définir le *filtre d'occupation bayésien* ou *Bayesian Occupancy Filter* (BOF). Le but de cette adaptation est d'intégrer à l'estimation de la grille d'occupation les propriétés temporelles du filtre bayésien.

Christophe Tay et al. présentent dans [25] deux formulations différentes du filtre d'occupation bayésien. La première formulation représente l'espace d'état avec une grille d'occupation à 4 dimensions [6] (deux dimensions pour la position et les deux autres pour les deux composantes orthogonales de la vitesse). La seconde formulation représente l'environnement avec une grille à 2 dimensions [24] où chaque cellule de la grille est associée à une distribution de probabilité sur

l'occupation et sur la vitesse. La différence entre ces deux formulations est subtile. Essentiellement, la formulation en 4D permet le chevauchement des objets avec des vitesses différentes tandis que dans la formulation 2D ce chevauchement n'est pas possible. De plus, dans la dernière formulation, il est possible d'inférer directement la distribution de probabilité sur les vitesses contrairement à la formulation 4D qui requiert la spécification de la dynamique de chaque cellule.

Pour ces raisons, nous avons choisi dans le cadre de notre projet d'utiliser la formulation à deux dimensions que nous allons ensuite exposer dans la suite de ce document. La formulation 4D est entièrement détaillée dans [25].

3.5.3 Programme bayésien

Le filtre d'occupation bayésien peut être exprimé sous la forme d'un programme bayésien. Dans le cadre du formalisme de la programmation bayésienne décrit plus haut, nous commencerons par définir les variables pertinentes comme suit :

- $C \in \mathcal{G}$: est une variable aléatoire qui identifie et localise chaque cellule de la grille $\mathcal{G}$.
- $A_C \in \mathcal{A}_C \subset \mathcal{G}$: est une variable aléatoire qui identifie une cellule potentiellement antécédente à la cellule C. L'ensemble des cellules antécédentes de la cellule C est noté $\mathcal{A}_C$ et contient toutes les cellules depuis lesquelles un objet peut provenir durant le pas de temps considéré. Pour une vitesse $v_k = \{v_x, v_y\}$ donnée et pour un pas de temps δt donné, on peut définir cet ensemble comme étant les cellules de coordonnées $(x - v_x \delta t \quad y - v_y \delta t)$ avec $(x \quad y)$ les coordonées cartésiennes de la cellule C considérée. Il faut noter que la cellule C appartient également à cet ensemble. C'est ce qui correspond au cas où la cellule est immobile durant deux instants successifs.
- O_C : est une variable aléatoire qui exprime l'état d'occupation de la cellule C et prend ses valeurs dans l'ensemble $\mathcal{O} = \{occ, emp\}$. De manière analogue, on définie O_C^{-1} pour exprimer l'état effectif de la cellule antécédente à la cellule C dans la phase d'inférence précédente. Cet état sera propagé à la cellule C si l'objet occupant la cellule A_C venait à se mouvoir vers la cellule C.
- V_C : est une variable aléatoire représentant la vélocité de la cellule C. Elle prend ses valeurs dans l'ensemble des vitesses discrétisées et préalablement définit et noté : $\mathcal{V} = \{v_0, \dots, v_N\}$.
- $Z_1, \dots, Z_S \in \{"Aucune\ détection"\} \cup \mathcal{Z}$: sont les mesures indépendantes provenant de S capteurs différents.

Ainsi, ces variables définissent la distribution de probabilité conjointe suivante :

$$P(C \wedge A_C \wedge O_c \wedge O_C^{-1} \wedge V_C \wedge Z_1 \wedge \dots \wedge Z_S)$$

3.5.3.1 Décomposition

Le but de cette décomposition est de faire apparaitre comme distributions élémentaires le modèle dynamique et le modèle du capteur, c'est-à-dire les distributions de probabilités $P(O_C|O_C^{-1})$ et $P(Z_i|V_C \wedge O_C \wedge C)$.

La décomposition de la distribution de probabilité conjointe donne :

$$P(C \wedge A_C \wedge O_c \wedge O_C^{-1} \wedge V_C \wedge Z_1 \wedge \ldots \wedge Z_S)$$
$$= P(A_C)P(V_C|A_C)P(C|V_C \wedge A_C)P(O_C^{-1}|A_C)P(O_C|O_C^{-1})\prod_{i=1}^{S} P(Z_i|V_C \wedge O_C \wedge C)$$

3.5.3.2 Formes paramétriques – identification

Les formes paramétriques suivantes seront associées aux distributions de probabilité élémentaires apparaissant dans la décomposition précédente :

- $P(A_C)$: est la distribution de probabilité sur tous les antécédents possibles de la cellule C. Etant donné qu'une cellule est supposée atteignable depuis n'importe quelle autre cellule de la grille, on choisit cette distribution comme étant uniforme.
- $P(V_C|A_C)$: est la distribution de probabilité sur toutes les vitesses possibles sachant que la cellule A_C est la cellule antécédente de la cellule C. Dans [24] on fait le choix de la représenter avec un histogramme sur l'ensemble des vitesses $\mathcal{V} = \{v_0, \ldots, v_N\}$.
- $P(C|V_C \wedge A_C)$: est la distribution de probabilité qui nous informe si la cellule C est atteignable depuis la cellule $[A_c = a]$ avec une vitesse $[V_c = v]$. Elle est prise égale à une distribution de Dirac unitaire si et seulement si $C_x = a_x + v_x\delta t$ et $C_y = a_y + v_y\delta t$. Dans certaines références, cette distribution est appelée *fonction d'atteignabilité*.
- $P(O_C^{-1}|A_C)$: est la distribution de probabilité qui exprime les états d'occupation probables de la cellule C à l'instant précédent.
- $P(O_C|O_C^{-1})$: est la distribution de probabilité conditionnelle sur l'état d'occupation de la cellule C qui dépend de l'état d'occupation de sa cellule antécédente. Dans le cas le plus général, elle est donnée par le modèle dynamique choisi. Pour une multitude de cas, il est possible de la définir à travers la matrice de transition : $T = \begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix}$ qui nous permet d'utiliser le modèle dynamique à accélération nulle « *null acceleration hypothesis* ». Le paramètre ϵ représente la probabilité que l'objet en C n'obéit pas à ce modèle.
- $P(Z_i|V_C \wedge O_C \wedge C)$: représente le modèle direct du capteur *i*. Souvent, les mesures des capteurs sont indépendantes de V_C. Ainsi, il est possible de simplifier l'écriture sous la forme : $P(Z_i|O_C \wedge C)$. La construction du modèle direct du capteur sera illustrée dans la suite de ce document.

3.5.3.3 La question

Nous rappelons que l'objectif de l'utilisation du filtre d'occupation bayésien est l'estimation de l'état d'occupation et la vitesse de chaque cellule de la grille. Ces deux informations sont contenues dans la distribution de probabilité conjointe $P(O_c \wedge V_C | Z_1 \wedge \ldots \wedge Z_S \wedge C)$. Ainsi, cette distribution représente la question de notre programme bayésien.

En utilisant la décomposition en distributions élémentaires précédente et sachant que $\prod_{i=1}^{S} P(Z_i | V_C \wedge O_C \wedge C)$ ne dépend ni de A_C ni de O_C^{-1} on peut alors écrire :

$$\overbrace{P(O_c \wedge V_C \wedge Z_1 \wedge \ldots \wedge Z_S \wedge C)}^{Estimation\ (probabilité\ conjointe)} = \underbrace{\prod_{i=1}^{S} P(Z_i | O_C \wedge C)}_{Observation} \underbrace{\sum_{A_C, O_C^{-1}} P(A_C) P(V_C | A_C) P(C | V_C \wedge A_C) P(O_C^{-1} | A_C) P(O_C | O_C^{-1})}_{Prédiction}$$

En appliquant les règles de marginalisation et le théorème de Bayes, on obtient la réponse à la question du programme bayésien :

$$\overbrace{P(O_c \wedge V_C | Z_1 \wedge \ldots \wedge Z_S \wedge C)}^{Estimation} = \frac{\overbrace{\prod_{i=1}^{S} P(Z_i | O_C \wedge C)}^{Observation} \overbrace{\sum_{A_C, O_C^{-1}} P(A_C) P(V_C | A_C) P(C | V_C \wedge A_C) P(O_C^{-1} | A_C) P(O_C | O_C^{-1})}^{Prédiction}}{\underbrace{\sum_{A_C, O_C^{-1}, O_C, V_C} P(A_C) P(V_C | A_C) P(C | V_C \wedge A_C) P(O_C^{-1} | A_C) P(O_C | O_C^{-1}) \prod_{i=1}^{S} P(Z_i | O_C \wedge C)}_{Normalisation}}$$

Notons que l'estimation de la probabilité d'occupation ainsi que la probabilité d'une vitesse donnée sont obtenues en utilisant la règle de marginalisation sur les variables libres comme suit :

$$P(O_c | Z_1 \wedge \ldots \wedge Z_S \wedge C) = \sum_{V_C} P(O_c \wedge V_C | Z_1 \wedge \ldots \wedge Z_S \wedge C)$$

$$P(V_C | Z_1 \wedge \ldots \wedge Z_S \wedge C) = \sum_{O_c} P(O_c \wedge V_C | Z_1 \wedge \ldots \wedge Z_S \wedge C)$$

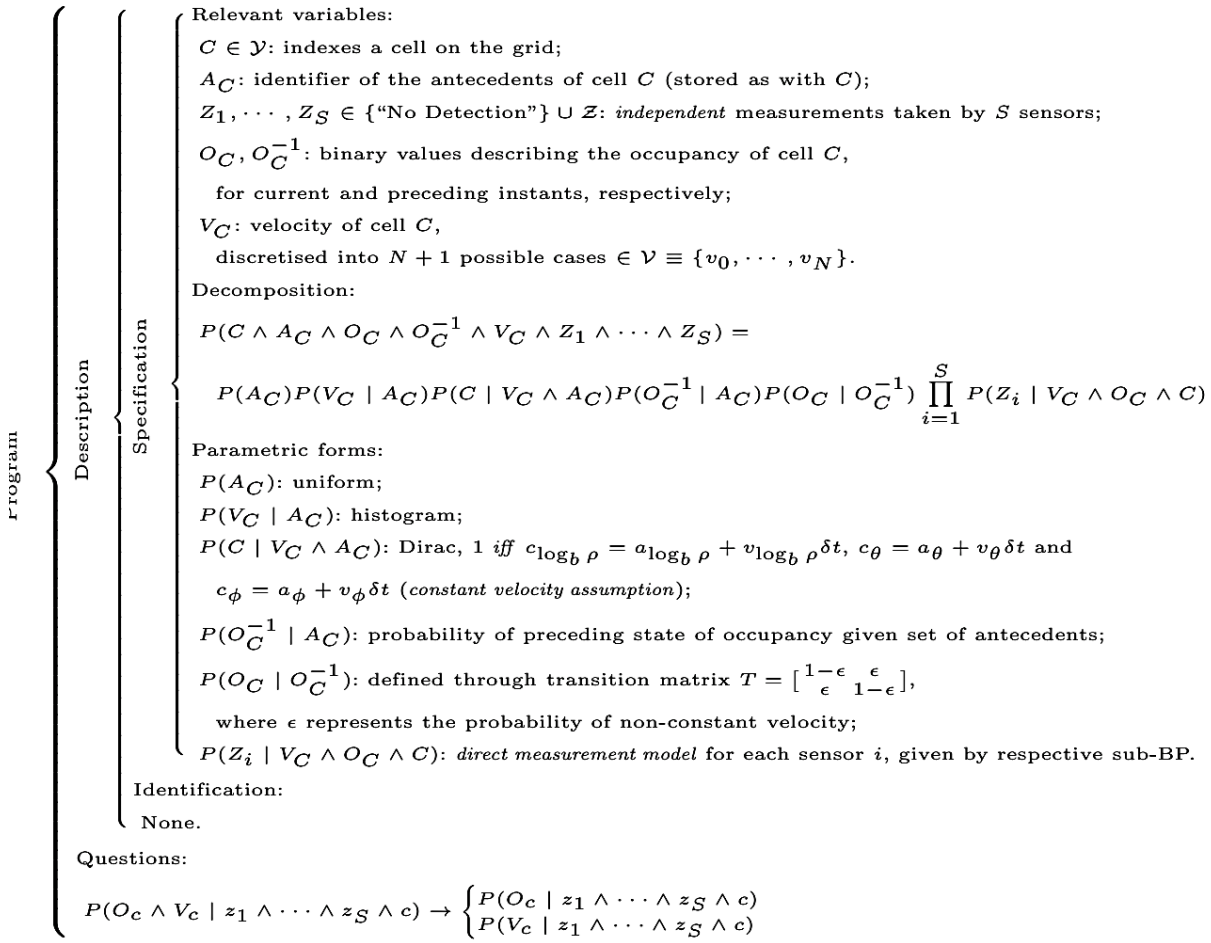


Figure 3.8 : Programme bayésien du *Bayesian Occupancy Filter* (BOF)

La figure 3.9 ci-dessous illustre comment le filtre d'occupation bayésien estime la distribution de probabilité conjointe $P(O_c \wedge V_C | Z \wedge C)$ en utilisant une boucle *prédiction-estimation* pour chaque cellule et dans le cas où un seul capteur est utilisé.

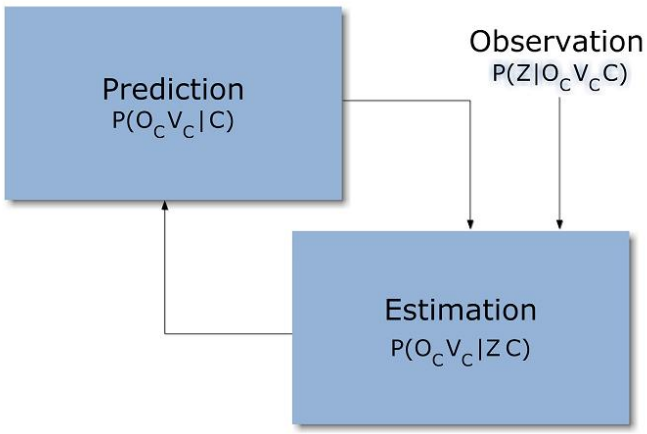


Figure 3.9 : Filtrage bayésien appliqué à l'estimation de l'occupation et de la distribution de probabilité sur les vitesses des cellules d'une grille d'occupation

Les deux phases prédiction et estimation sont accomplies à chaque itération. La phase de prédiction projette vers l'instant t la distribution de probabilité conjointe à partir des observations des instants 0 à $t-1$. Durant la phase d'estimation, $P(O_c \wedge V_C | Z \wedge C)$ est mise à jour en utilisant les dernières observations du capteur représentées par $P(Z|O_C \wedge C)$ afin d'obtenir l'estimation finale de $P(O_c \wedge V_C | Z \wedge C)$. Ce résultat est ensuite utilisé dans la phase de prédiction à l'itération suivante.

3.5.4 Modélisation du capteur

Le modèle de mesure, appelé aussi *modèle capteur* exprime le processus de mesure de manière probabiliste. En d'autres termes, sachant la distance réelle à laquelle se trouve l'objet ou l'obstacle, le modèle de mesure définit la probabilité d'obtenir certaines mesures spécifiques Z.

Il est évident qu'il n'est pas possible de déterminer un tel modèle de manière exacte étant donné que l'information récupérée des capteurs est limitée en précision et de nombreux facteurs peuvent influer la qualité des mesures : température, éclairage, humidité ... Il est donc impératif de tenir compte de ce manque de précision en se plaçant, comme expliqué au 2ème chapitre, dans un cadre de raisonnement probabiliste.

En DATMO, il est très fréquent que les robots utilisent des capteurs télémétriques pour percevoir leur environnement. C'est notamment le cas du Robucar où un télémètre laser SICK LMS 511 est placé à l'avant du véhicule. Ainsi, nous ferons le choix dans le présent rapport de nous restreindre à la description d'un modèle probabiliste de la réponse d'un capteur télémétrique. Ce modèle est similaire à celui décrit par Elfes dans sa présentation des grilles d'occupation [9].

Comme pour le BOF, il est plus commode de décrire le modèle probabiliste du capteur en utilisant le formalisme de la programmation bayésienne présentée dans les sections précédentes.

3.5.4.1 Variables

Les variables qui nous intéressent ici sont :

- La variable X qui définit la cellule de la grille à laquelle nous nous intéressons ;
- E_X : est une variable booléenne qui signifie « un objet existe dans la cellule X » ;
- D_X : est elle aussi une variable booléenne qui signifie « un objet a été détecté dans la cellule X » ;
- Et enfin les mesures capteur Z.

La conjonction des variables E_X et D_X permet d'expliciter les quatre situations possibles pour la réponse d'un capteur dont deux correspondent à un fonctionnement correct du capteur :

- $[E_X = 1] \wedge [D_X = 1]$: un objet occupe la cellule X et il est détecté par le capteur ;

– $[E_X = 0] \wedge [D_X = 0]$: la cellule X est libre et le capteur n'a rien détecté dans cette cellule.

Les deux autres possibilités correspondent à une défaillance du capteur :

– $[E_X = 1] \wedge [D_X = 0]$: un objet occupe la cellule X mais le capteur ne l'a pas détecté. Ceci peut être dû à plusieurs raisons : la cellule X se trouve en dehors du champ de vision du capteur, l'objet est masqué par un autre objet, ou le capteur est défaillant ;

– $[E_X = 0] \wedge [D_X = 1]$: la cellule X est vide alors que le capteur y détecte un objet. Dans ce cas, on parle communément de *fausse alarme*.

Ces variables définissent la probabilité conjointe :

$$P(Z \wedge X \wedge E_X \wedge D_X)$$

3.5.4.2 Décomposition

Dans sa thèse [4], Coué choisis la décomposition suivante :

$$P(Z \wedge X \wedge E_X \wedge D_X) = P(X)P(E_X|X)P(D_X|E_X \wedge X)P(Z|D_X \wedge E_X \wedge X)$$

3.5.4.3 Formes paramétriques – identification

Il faut à présent associer à chacune des distributions élémentaires précédentes une forme paramétrique :

– $P(X)$ et $P(E_X|X)$ représentent des *a priori* sur l'environnement du robot. Etant donné qu'on ne dispose pas de ces informations, nous ferons le choix d'un a priori non informatif et donc ces distributions seront choisies uniformes. Il faut noter que ces deux distributions seront analytiquement simplifiées dans lors de la formulation de la question, ce choix n'a donc aucune conséquence sur l'inférence.

Les distributions de probabilité sur la variable D_X représentent, suivant la valeur de la variable E_X, la capacité du capteur à détecter ou non une cible se trouvant à la cellule X ou bien à générer des fausses alarmes :

– $P([D_X = 1]|[E_X = 1] \wedge X)$: représente la probabilité de détection du capteur. Elle est souvent notée $P_D(X)$. $P([D_X = 0]|[E_X = 1] \wedge X) = 1 - P_D(X)$ représente la probabilité que le capteur manque une cible existante. Nous appelons cette probabilité : *probabilité de non-détection* et elle sera notée $P_{ND}(X)$.

– Inversement, $P([D_X = 1]|[E_X = 0] \wedge X)$ représente la probabilité de fausse alarme du capteur et elle sera notée $P_{FA}(X)$. $P([D_X = 0]|[E_X = 0] \wedge X)$ est alors égale à $1 - P_{FA}(X)$.

Ces distributions dépendent des capteurs utilisés et sont le plus souvent estimées expérimentalement dans des conditions particulières d'utilisation.

On définit les formes paramétriques associées à la famille de distribution $P(Z|D_X \wedge E_X \wedge X)$ suivant les quatre combinaisons que peuvent prendre les variables D_X et E_X.

- $P(Z|[D_X = 1] \wedge [E_X = 0] \wedge X)$ et $P(Z|[D_X = 1] \wedge [E_X = 1] \wedge X)$: dans les deux cas, une détection a eu lieu pour la cellule X. Ces deux distributions représentent donc la réponse du capteur à une détection en X et sont modélisées en utilisant, indifféremment de l'état de E_X, une famille de distributions normales :

$$P(Z|[D_X = 1] \wedge E_X \wedge X) = \mathcal{N}(Z, \mu(X), \Sigma(X))$$

Pour une cellule X donnée, cette distribution est de moyenne $\mu(X)$ égale à la réponse attendue du capteur et donc à la distance entre le capteur télémétrique et le centre de la cellule. La matrice de covariance $\Sigma(X)$ permet alors de représenter les variations possibles de la réponse du capteur autour de cette réponse attendue. Souvent, elle est qualifiée de grandeur qui représente *la précision du capteur.*

Notons également que la matrice de covariance est une fonction de X puisque justement la précision d'un capteur peut varier suivant la zone observée. Par exemple, plus on s'éloigne du capteur télémétrique, plus l'imprécision sur l'angle devient influente. Les valeurs propres de cette matrice sont obtenues par calibration du capteur.

Passons maintenant au cas où le capteur n'a détecté aucun objet dans la cellule X :

- $P(Z|[D_X = 0] \wedge [E_X = 0] \wedge X)$: cette distribution nous indique que nous savons que la cellule X n'est pas occupée et qu'une détection n'a eu lieu dans cette cellule. En considérant uniquement ces deux informations, il est difficile d'affirmer quelle serait la valeur de l'observation Z la plus probable. En effet, n'importe quelle valeur de Z est possible, on choisit donc, comme dans toutes les situations identiques, une distribution uniforme.
- $P(Z|[D_X = 0] \wedge [E_X = 1] \wedge X)$: cette distribution représente le cas particulier où une occultation d'objet a eu lieu. En effet, dans ce cas-là nous savons qu'un objet occupe la cellule X et qu'aucune détection n'a eu lieu dans cette cellule. Deux explications sont possibles : soit un autre objet masque la cellule X et a été détecté par le capteur, soit l'objet présent dans la cellule X n'est pas masqué mais n'a pas été détecté par le capteur. La forme paramétrique choisie pour cette distribution traduit ces deux cas de figure et on choisit donc une distribution dont la probabilité pour la mesure d'être dans une zone masquée par la cellule X différente de celle des zones non masquées.

Afin de traduit le fait que l'observation a plus de chance de provenir d'une zone non masquée par la cellule X, on attribue une probabilité supérieur à l'uniforme pour les mesures inférieures à la mesure attendue. La probabilité des autres mesures est fixée de manière à assurer une sommation sur toute la plage de mesure égale à 1.

3.5.4.4 Question

Le but du modèle capteur et donc de la description du programme bayésien présentée plus haut est de fournir une forme paramétrique à la distribution de probabilité $P(Z|E_X \wedge X)$ apparaissant dans l'estimation du filtre d'occupation bayésien. Nous devons donc poser la question suivante à notre description :

$$P(Z|e_x \wedge x)$$

Le résultat de l'inférence donne :

$$P(Z|e_x \wedge x) = \frac{\sum_{D_X} P(Z \wedge x \wedge e_x \wedge D_X)}{P(e_x \wedge x)}$$

Qu'on peut écrire sous la forme :

$$P(Z|e_x \wedge x) = \frac{P(x)P(e_x|x)\sum_{D_X} P(D_X|e_x \wedge x)P(Z|D_X \wedge e_x \wedge x)}{P(x)P(e_x|x)\underbrace{\sum_{Z,D_X} P(D_X|e_x \wedge x)P(Z|D_X \wedge e_x \wedge x)}_{=1}}$$

Finalement :

$$P(Z|e_x \wedge x) = \sum_{D_X} P(D_X|e_x \wedge x)P(Z|D_X \wedge e_x \wedge x)$$

Etant donné que D_X est une variable à deux états, on peut l'écrire la question sous sa forme éclatée :

$$\begin{aligned} P(Z|e_x \wedge x) = {} & P([D_X = 1]|e_x \wedge x)P(Z|[D_X = 1] \wedge e_x \wedge x) \\ & + P([D_X = 0]|e_x \wedge x)P(Z|[D_X = 0] \wedge e_x \wedge x) \end{aligned}$$

Programme

Description

Spécification

Variables Pertinentes

- X : cellule de l'environnement du véhicule ;
- Z : observation capteur ;
- D_X : un objet a été détecté dans l'état X ;
- E_X : un objet existe dans l'état X.

Décomposition

$$\mathrm{P}(X\ Z\ D_X\ E_X) = \mathrm{P}(X)\mathrm{P}(E_X \mid X)\mathrm{P}(D_X \mid E_X\ X)\mathrm{P}(Z \mid D_X\ E_X\ X)$$

Formes Paramétriques

- $\mathrm{P}(X)$: Uniforme ;
- $\mathrm{P}(E_X \mid X)$: Uniforme ;
- $\mathrm{P}(D_X \mid E_X\ X)$: probabilité de détection, de fausse alarme (suivant la valeur de E_X).
- $\mathrm{P}(Z \mid [D_X = 1]\ [E_X = 1]\ X)$, $\mathrm{P}(Z \mid [D_X = 1]\ [E_X = 0]\ X)$: fonction de vraisemblance ;
- $\mathrm{P}(Z \mid [D_X = 0]\ [E_X = 0]\ X)$, $\mathrm{P}(Z \mid [D_X = 0]\ [E_X = 1]\ X)$: « à la main ».

Identification :

calibration pour $\mathrm{P}(D_X \mid E_X\ X)$ et $\mathrm{P}(Z \mid X\ [D_X = 1])$.

Question :

$\mathrm{P}(Z \mid e_x\ x)$

Figure 3.10 : Modèle capteur télémétrique : programme bayésien

3.5.4.5 Illustration

Dans le cadre de notre application DATMO, un télémètre laser SICK LMS 511 est utilisé pour capter l'information sur l'environnement. La fiche technique de ce capteur précise qu'il a une précision de 3% dans les conditions normales d'utilisation.

Le télémètre laser est connu pour sa grande précision, il est donc très fréquent de prendre la probabilité de détection du capteur égale à 0.9 et la probabilité de fausse alarme égale à 0.1 [4]. Ainsi, en simulant un ensemble de mesures laser disposées à distance constante du capteur, nous obtenons la figure suivante :

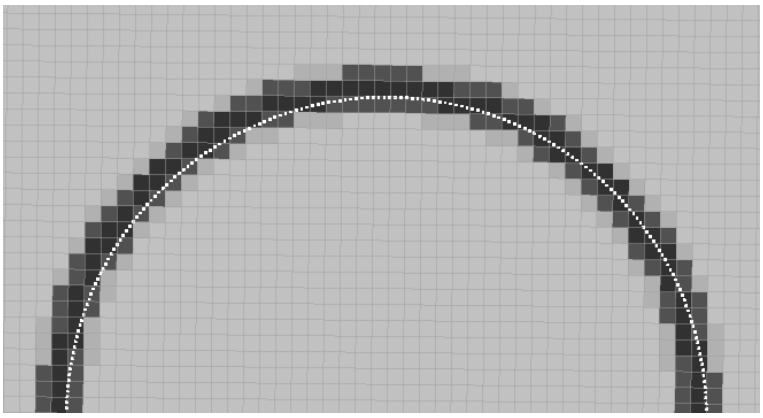


Figure 3.11 : Exemple de modèle probabiliste d'un capteur télémétrique laser

Sur cette figure, les points blancs représentent les scans laser et la probabilité $P(Z|e_x \wedge x)$ est représentée en utilisant les nuances de gris (noir pour une probabilité égale à un et blanc pour une probabilité égale à zéro). Remarquons ici que les cellules proches des mesures laser ont une probabilité $P(Z|e_x \wedge x)$ proche de un. Inversement, plus on s'éloigne, plus cette probabilité décroit.

Il est à souligner que la largeur de la *bande* de cellule ayant une probabilité $P(Z|e_x \wedge x)$ élevée dépend de la précision du capteur et donc ici de la valeur de l'écart-type donnée pour la fonction de vraisemblance lors de la modélisation.

3.5.5 Les modèles dynamiques

Dans un cadre général, les modèles dynamiques décrivent la manière dont un objet mobile évolue entre deux pas de temps successifs. Ceci se traduit par une distribution de probabilité de la forme $P(X^k|X^{k-1})$.

En DATMO, il est commun d'utiliser le *modèle à accélération nulle* (ou à *vitesse constante*) qui suppose que les cellules d'une grille d'occupation se déplacent avec une vitesse constante entre deux pas de temps successifs. De cette manière, on peut déduire la vitesse d'une cellule sachant la cellule d'où elle provient en calculant simplement la distance entre les centres des deux cellules. En effet, étant donné que l'accélération des cellules est supposée nulle, diviser cette

distance sur le pas de temps Δt donnerait la vitesse de cette cellule. Or, si cette hypothèse n'était pas respectée, une cellule se mouvant avec la vitesse calculée entre l'instant t_1 et l'instant t_2 (avec $t_2 - t_1 = \Delta t$) ne coïnciderait pas à l'instant t_2 avec une cellule de la grille.

D'autres modèles sont également utilisés. A. Petrovskaya en a cité trois autre dans [7] dont le modèle *Brownien*, le *modèle linéaire* et enfin le *Bicycle model*. Ces trois modèles ont déjà été présentés dans le chapitre précédent.

Ceci dit, on rappelle que lors de la présentation du BOF, nous avons eu besoin de définir la distribution de probabilité $P(O_c|O_c^{-1})$ qui représente l'état d'occupation de la cellule C sachant celui de sa cellule antécédente. Nous avons affirmé que pour un modèle à accélération nulle, cette distribution est définie à travers la matrice de transition :

$$T = \begin{bmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{bmatrix}$$

Où le paramètre ϵ représente la probabilité que l'objet en C n'obéit pas au modèle à accélération nulle.

Ainsi, la distribution de probabilité $P(O|O^{-1})$ fait intervenir quatre cas distincts :

- $P([O_c = occ]|[O_c^{-1} = occ])$: représente la probabilité que la cellule C soit occupée sachant que sa cellule antécédent est occupée. Elle est prise égale à $1 - \epsilon$ et donc si le modèle à vitesse constante est parfaitement respecté, l'état d'occupation de la cellule antécédente est entièrement transmit à la cellule C.
- $P([O_c = occ]|[O_c^{-1} = emp])$: représente la probabilité que la cellule C soit occupée sachant que sa cellule antécédente est libre. Elle est prise égale à ϵ vu que ça représente justement la probabilité que le modèle ne soit pas respecté.
- $P([O_c = emp]|[O_c^{-1} = occ])$: représente la probabilité que la cellule C soit libre alors que sa cellule antécédente est occupée. Elle est également prise égale à ϵ pour les mêmes raisons que dans le cas précédent.
- $P([O_c = emp]|[O_c^{-1} = emp])$: représente la probabilité que la cellule C soit libre sachant que sa cellule antécédent est aussi libre. Elle est prise égale à $1 - \epsilon$ pour les mêmes raisons que dans le premier cas.

Selon le cas, on remplace $P(O_c|O_c^{-1})$ par sa valeur dans l'expression du filtre d'occupation Bayésien.

3.5.6 Champ des vitesses

La discrétisation est un problème récurrent dans plusieurs domaines. Dans le cas d'une représentation de l'environnement avec une grille d'occupation, l'occupation d'une cellule est supposée la même pour la cellule entière, et pour un déplacement arbitraire de cette cellule, il est

probable que sa position finale ne coïncide pas exactement avec les limites géométriques d'une autre cellule. Ce problème est connu sous le nom de « *aliasing problem* ».

Permettre un tel mouvement introduit des erreurs dans l'étape de prédiction. Une façon de contourner ce problème est d'accorder plus de *croyance* aux mesures, c'est-à-dire donner plus de poids au modèle capteur. Cependant, ceci peut rendre notre filtre plus sensible aux fausses détections et diminuer ainsi sa qualité.

Un moyen efficace d'éviter ce problème est de choisir la vitesse d'une cellule lors de la prédiction de telle sorte qu'elle correspond à un déplacement égal à un nombre entier de cellule. Or, cela dépend fortement de la durée $\boldsymbol{dt}$ de l'étape de mise à jour du filtre, il est donc plus commode de considérer cette durée comme une constante. Par conséquent, la grille est régulière et en posant $\boldsymbol{dx}$ et $\boldsymbol{dy}$ les dimensions de chacune de ses cellules, on pourra caractériser le champ des vitesses comme suit :

$$\mathcal{V} = \left\{ \left(\frac{p}{n}\frac{dx}{dt}; \frac{q}{n}\frac{dy}{dt} \right) | (p,q,n) \in \mathbb{Z}^2 \times \mathbb{N} \right\}$$

Dans cette définition du champ de vitesse, le terme $\frac{1}{n}$ permet de considérer les mouvements des cellules qui ont besoin de plus d'un pas de temps pour atteindre une autre cellule (mouvements lents). A titre d'exemple, une translation de $\boldsymbol{dx}$ qui a besoin de deux pas de temps pour réaliser le mouvement correspond au vecteur $(\frac{dx}{2dt}; 0)$ et ainsi les coordonnées (p, q, n) correspondantes sont $(1,0,2)$.

Une conséquence de cette représentation est que les plans de vitesse ne sont pas tous vérifié à chaque pas de temps. En effet, les objets lents nécessitent moins d'attention que les objets en mouvement rapide, les plans de vitesse lentes sont donc observés moins fréquemment. Ceci permet notamment de réduire le temps de calcul lors de l'implémentation.

3.6 Conclusion

Dans ce 3ème chapitre, nous avons présenté le *grid-based* DATMO ainsi que les différentes notions théoriques nécessaires à son implémentation. Cela va de la justification de l'utilisation de la théorie des probabilités comme alternative à la logique classique à la présentation du formalisme des grilles d'occupation utilisé dans cette approche.

Nous avons pu constater que l'approche basée sur les grilles d'occupation n'apporte que des bénéfices. On cotera notamment que le filtrage des grilles d'occupation permet d'accroître la robustesse de l'estimation aux incertitudes et aux défaillances des capteurs. Dans cette approche, le modèle dynamique n'est plus attaché à un objet puisque cette notion n'existe plus dans le formalisme des grilles d'occupation.

Chapitre 4

Implémentation du *grid-based* DATMO

Dans ce chapitre, nous présenterons les différentes étapes de l'implémentation du *grid-based* DATMO décrites dans le chapitre précédent. Nous commencerons par présenter l'environnement logiciel ROS utilisé dans l'implémentation pour la première fois du DATMO sur la plateforme expérimentale « Robucar » du CDTA, également décrite dans le présent chapitre.

Les résultats expérimentaux seront détaillés et commentés dans le paragraphe (4.3).

4.1 Présentation de l'environnement logiciel

Avant de présenter les résultats expérimentaux, il est nécessaire de présenter l'environnement logiciel utilisé pour implémenter les différents programmes composant le module du DATMO. Comme nous l'avons mentionné en introduction, il s'agit de l'environnement ROS « *Robot Operating System* » également qualifié de *système d'exploitation pour robot*.

4.1.1 Description d'un système d'exploitation pour robot

En informatique, un système d'exploitation (souvent appelé OS pour *Operating System*) est un ensemble de programmes qui dirige l'utilisation des capacités d'un ordinateur. Il reçoit et gère les différentes demandes d'utilisation des capacités matérielles (stockage en mémoire, calcul du processeur, communication vers des périphériques, ...).

De manière analogue, un système d'exploitation pour robot est définit comme étant aussi un ensemble de programmes (*softwares*) qui gèrent l'allocation des ressources matérielles du robot composées essentiellement par un ensemble de capteurs et d'actionneurs.

Le développement des systèmes d'exploitation pour robot est essentiellement motivé par les difficultés liées à la conception des logiciels pour les robots. En effet, il est de plus en plus difficile pour les chercheurs de programmer les différentes tâches de la robotique étant donné que l'ampleur et la portée de la robotique continuent à croître. Aujourd'hui, on retrouve plusieurs types de robots et dans chaque type, des robots avec des architectures matérielles différentes existent. Ainsi, les chercheurs en robotique et en intelligence artificielle sont amenés à passer un temps non négligeable à écrire les programmes de base utiles au fonctionnement de leurs robots

tels que des pilotes pour les différents capteurs utilisés, programmes de gestion de la mémoire, des processus et de la concurrence … Ainsi, une telle démarche ne permet pas une réutilisation des programmes développés vu qu'ils sont fortement liés au matériel auquel ils ont été conçus.

Pour contourner ce problème et permettre ainsi aux chercheurs de gagner du temps et programmer les robots en faisant abstraction de son architecture matérielle, plusieurs systèmes d'exploitation pour robot furent développés dont *Robot Operating System* qu'on présentera dans ce document. Ces systèmes proposent des fonctionnalités suffisamment standardisée pour être utilisés sur différentes plateformes robotiques.

Dans [39] on explique que ROS n'est pas le meilleur OS pour toutes les applications possibles de la robotique. En réalité, vu la diversité des domaines où la robotique intervient, il est très difficile, voire impossible, de développer un OS suffisamment standard pour s'adapter à toutes les applications et à tout type de robot. Ainsi, le choix du système d'exploitation dépend de plusieurs facteurs et doit prendre en considération les caractéristiques de chaque OS.

4.1.2 Robot Operating System (ROS)

Comme son nom l'indique *Robot Operating System*, souvent abrégé en ROS et prononcé en un seul mot, est un système d'exploitation pour robot. Il a été développé en 2007 par la société américaine *Willow Garage* et a la particularité d'être open-source et donc soutenu par une communauté grandissante de chercheurs et de développeurs.

En toute rigueur, ROS est un *méta-système d'exploitation* pour robot, c'est-à-dire qu'il fournit les fonctionnalités attendues d'un OS (abstraction du matériel, gestion de la concurrence et des processus, …) tout en ayant les caractéristiques d'un *middleware* (un réseau d'échange d'informations entre différentes applications informatiques).

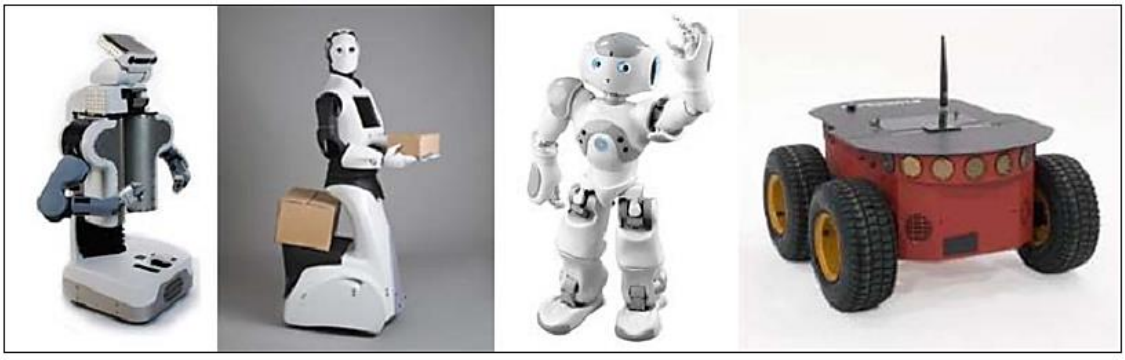


Figure 4.1 : Quelques plateformes robotiques prisent en charge par ROS

Il faut souligner que ROS a besoin d'un système d'exploitation traditionnel pour fonctionner. Les versions stables actuelles doivent être installées de préférence sur la distribution Ubuntu de Linux.

4.1.3 Caractéristiques

ROS se distingue des autres systèmes d'exploitation pour robot par son organisation qui se résume dans les cinq grandes lignes suivantes :

- Peer to Peer ;
- Basé sur des outils ;
- Multi-langages ;
- Léger ;
- Gratuit et open-source.

Reprenons chacun de ces points.

- Peer to Peer : Une tâche robotique suffisamment complexe est composée de plusieurs programmes simples qui s'exécutent en général de manière concurrente. Ces programmes, appelés nœuds, communiquent entre eux de manière synchrone ou asynchrone en utilisant l'architecture Peer-to-Peer.
- Multi-langage : ROS est qualifié de neutre d'un point de vue langage. Toutefois, il est très fréquent que les programmes sous ROS soient écrits en C++ ou en Python.
- Basé sur des outils : Afin de gérer au mieux la complexité de ROS, les concepteurs ont opté pour une architecture dite à micro-noyau (ou *microkernel*) où un grand nombre d'outils est utilisé pour assurer le fonctionnement des différents composants de ROS. Ces outils permettent essentiellement d'effectuer des tâches de navigation dans l'arborescence du code, d'obtenir les paramètres de configuration, de visualiser la topologie de la connexion peer-to-peer, de mesurer la bande passante, de tracer graphiquement des données de message, ... etc.
- Léger : Comme un OS pour robot a principalement pour but la réutilisation des programmes, les développeurs de ROS ont fait en sorte que les pilotes et autres algorithmes soient contenus dans des exécutables indépendants. Cela assure une réutilisation maximale mais surtout, le maintien d'une taille réduite étant donné que les parties communes entre les programmes ne sont stockées qu'une seule fois.
- Gratuit et open-source : comme nous l'avons déjà mentionné, ROS est entièrement gratuit et open-source. Le code est accessible au publique sur le site officiel du système d'exploitation.

4.1.4 Notions de base de ROS

Le principe de base d'un OS robotique est de faire fonctionner en parallèle un grand nombre d'exécutables qui doivent pouvoir échanger de l'information de manière synchrone ou asynchrone.

ROS assure un tel fonctionnement en définissant des notions de base simples telles que *master*, *nœud*, *topic*, *service* et *message*.

- Un nœud est une instance d'un exécutable. Il peut correspondre à un capteur, un moteur, un algorithme de traitement, ... etc. Pour faire simple, les nœuds sont les bouts de programmes simples que nous créons et qui, une fois lancés ensemble, réalisent une tâche complexe.
- Le master : le master est un service de déclaration et d'enregistrement des nœuds qui permet ainsi aux nœuds de se connaître et d'échanger de l'information essentiellement via des *topics* ou des *services*.
- Un topic est un système de transport de l'information basé sur le système d'abonnement/publication (*subscribe/publish*) ; un ou plusieurs nœuds pourront publier de l'information sur un *topic*, tandis que d'autres pourront la lire. C'est donc un système de communication asynchrone *many-to-many*. Les *topic* ont la particularité d'être fortement typé ; l'information publiée (le message) est toujours structurée de la même manière.
- Un message est une structure de donnée composite composée de données de types primitifs (chaînes de caractères, booléens, entiers, flottants, ...). En C++, la structure d'un message est définit en utilisant la notion de *class* et chaque message correspond à une instance de cette *class*.
- Un service permet une communication synchrone entre les nœuds. Ainsi, l'échange de l'information s'effectue soit de manière asynchrone via un *topic* ou de manière synchrone via un service.

En plus de ces notions, les ressources de ROS sont organisées dans une structure hiérarchique sur disque. Deux concepts importants se détachent :

- le package : est l'unité principale d'organisation logicielle de ROS. Un package est un répertoire qui contient principalement plusieurs nœuds et des bibliothèques externes ;
- la *stack* : une *stack* est définie simplement comme une collection de packages ce qui permet de regrouper les package répondant à une même tâche principale de la robotique. Ainsi, on retrouve par exemple un *stack* dédiée à navigation, la localisation, une *stack* pour le DATMO, etc. Il faut noter que la notion de *stack* n'est plus utilisée dans les dernières distributions de ROS.

On définit également, comme dans Linux, la notion de distribution ROS comme un regroupement de *stack* de la même version. La dernière version en date est dénommée *Jade,* toutefois, pour des raisons de compatibilité, nous utiliserons « ROS Hydro » pour implémenter notre package DATMO.

La figure ci-dessous résume ce qui a été dit jusque-là et présente l'architecture de base de fonctionnement d'un programme sous ROS.

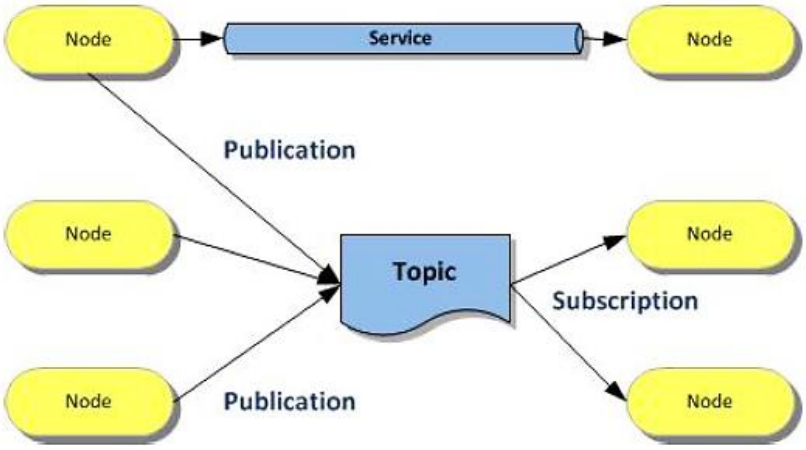


Figure 4.2 : Fonctionnement d'un programme sous ROS

4.2 Présentation de la plateforme expérimentale

Le Robucar est un véhicule électrique intelligent construit sur la base d'un châssis tubulaire des petites voitures utilisées dans les terrains de Golf. Il et commercialisé par l'entreprise française ROBOSOFT spécialisée dans les solutions de robotique, et est utilisé comme plateforme expérimentale par plusieurs instituts et centres de recherches en robotique dans le but de tester et valider les applications développées.

Figure 4.3 : La plateforme expérimentale « Robucar » du CDTA

Parmi ces centres de recherche, on citera particulièrement l'INRIA de Grenoble et son l'équipe de recherche « e-Motion » qui travaille sur le développement de méthodes algorithmiques

permettant de construire des systèmes artificiels dotés de capacités de perception, de décision et d'action. Leurs travaux sont le plus souvent testés et illustrés en utilisant la plateforme robotique « CyCab ; une déclinaison du Robucar.

Figure 4.4 : A gauche : Le CyCab. A droite : Un Robucar TT : déclinaison tout terrain du Robucar

Le Robucar, dirigé par un ordinateur de bord, est constitué de deux ponts identiques et indépendants. Chaque pont comporte deux roues entrainées chacune par un moteur électrique, un vérin électrique pour la commande de braquage et un dispositif de suspension. Il est également équipé de plusieurs batteries assurant son autonomie énergétique et d'un ensemble de capteurs différents (télémètre laser, capteurs ultrasoniques, caméra, odomètres, ...).

Ses principales caractéristiques sont :

- Longueur totale : 1836 mm ;
- Largeur totale : 1306 mm ;
- Hauteur : 616 mm ;
- Poids total : 310 kg ;
- Motorisation : 4 moteurs électriques de 1200 Watts ;
- 4 roues motrices et directrices ;
- Vitesse maximale : 18 Km/h (5 m/s) ;
- Autonomie : 2 heures d'utilisation continue ;
- Capacité d'accueil : 2 personnes avec bagages ;
- Conduite automatique ou manuelle ;

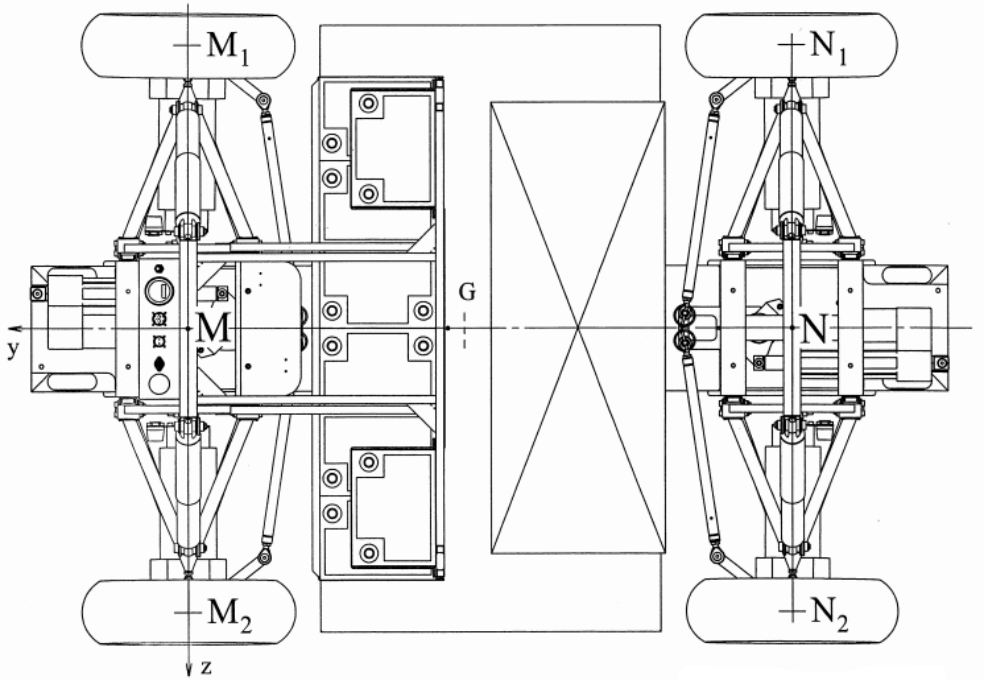


Figure 4.5 : Schéma globale du Robucar

Le Robucar est également doté de capacités cinématiques accrues comparé aux véhicules classiques ; il lui est possible d'utiliser, selon la situation, trois modes de braquage :

- Mode simple braquage (mode *Single*) : la propulsion est faite par les quatre roues mais la direction est faite seulement sur l'essieu avant ;
- Mode double braquage (mode *Dual*) : la propulsion et la rotation sont assurées par les quatre roues. Le sens de rotation des roues avant est opposé à celui des roues arrière ;
- Mode parking (mode *Park*) : la propulsion et la rotation sont assurées par les quatre roues. Le train avant et le train arrière tournent dans le même sens.

Figure 4.6 : Les trois modes de braquage du Robucar

4.2.1.1 Equipements de base

Les plateformes robotiques expérimentales telles que le Robucar sont conçues de manière à combiner les caractéristiques d'un véhicule traditionnel (transporter des personnes) et pouvoir

fonctionner en mode autonome sans intervention humaine. Ils sont donc essentiellement dotés de systèmes de perception et de moyens informatiques embarqués pour la prise de décision.

Nous allons dans cette sections détailler les deux équipements de base nécessaire à notre application DATMO à savoir l'ordinateur embarqué, le télémètre laser et les odomètres.

Ordinateur embarqué

Le Robucar est dirigé par un ordinateur de bord ou ordinateur embarqué équipé d'un microprocesseur « Intel Pentium MMX 233 MHz » et fonctionnant sous Linux 6.2 avec une couche RTAI permettant d'apporter des fonctionnalités de *real-time*. L'ordinateur embarqué permet essentiellement de commander l'architecture de bas niveau du robot (commande des moteurs et des vérins).

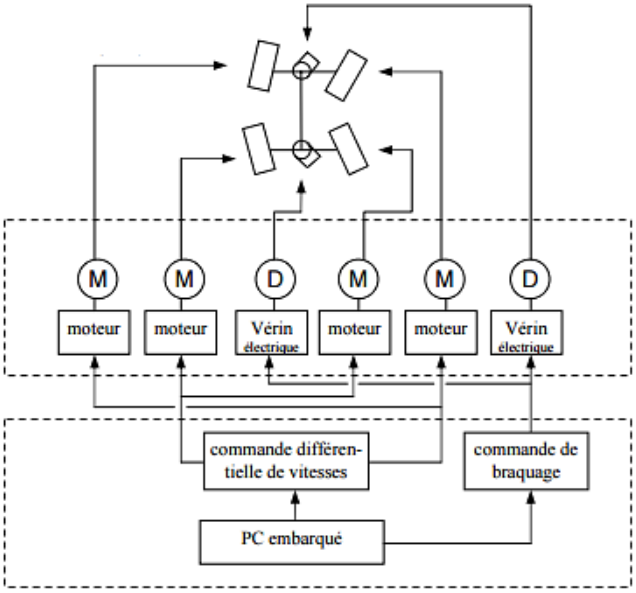


Figure 4.7 : Schéma résumant la commande des moteurs et des actionneurs du Robucar par le PC embarqué

Il faut noter que les caractéristiques du PC embarqué (taille de la mémoire et performances du microprocesseur) ne permettent pas de l'utiliser pour exécuter des programmes complexes. Il est donc exclusivement dédié à des tâches de bas niveau telles que la commande des moteurs et des capteurs ainsi qu'aux tâches de communication. On utilise donc, pour le traitement, un PC distant possédant de meilleures performances et relié au PC embarqué.

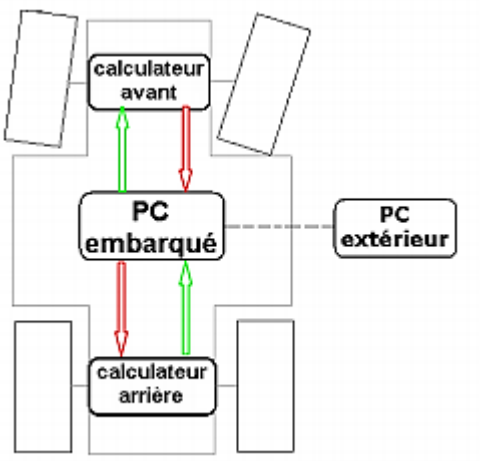


Figure 4.8 : Schéma illustrant l'organisation de la communication dans le Robucar

Télémètre laser SICK LMS 511

Le Robucar est équipé d'un télémètre laser de type SICK LMS 511 monté sur l'avant du Robucar à une hauteur de 30 cm du sol. Ce modèle est fabriqué par la société allemande SICK et commercialisé depuis 1994.

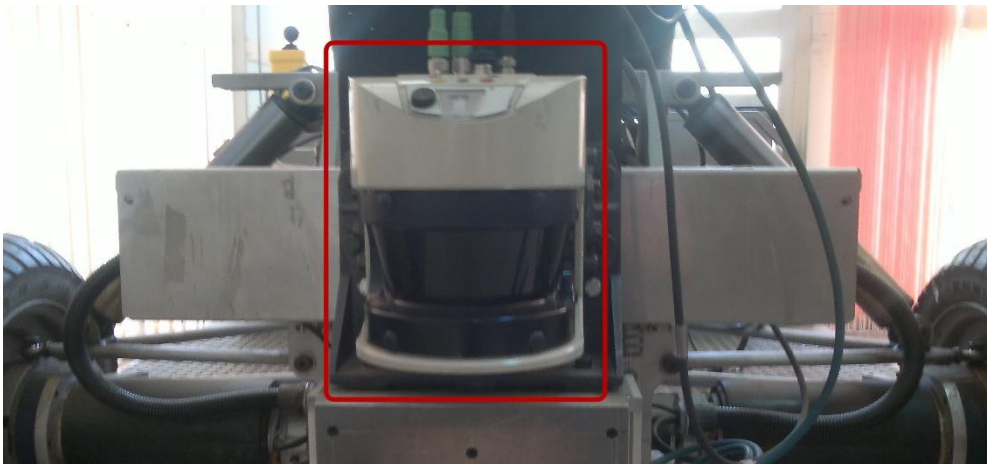


Figure 4.9 : Le capteur laser SICK LMS 511 monté sur l'avant du Robucar

La fiche technique du télémètre laser SICK LMS 511 affirme qu'il est capable de de balayer une zone allant de -95° à +95°, soit une plage de 190° avec une portée de 80 mètres. Ce balayage se fait dans le plan 2D horizontal des faisceaux lumineux envoyés et les résultats des scans sont transcrits en coordonnées polaires.

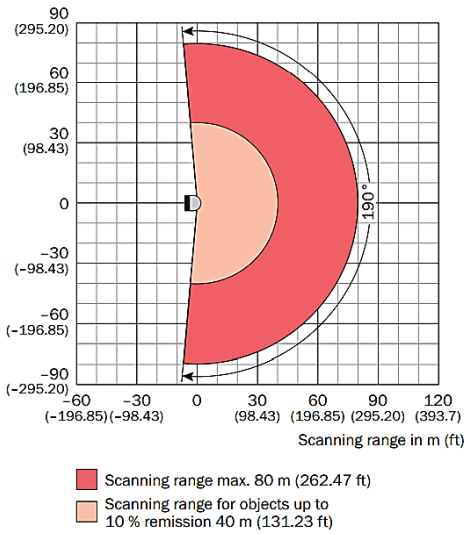


Figure 4.10 : Plage de balayage du capteur SICK LMS 511

La fonction première du capteur LMS est de balayer l'environnement et d'en extraire les obstacles détectés en renvoyant leurs positions polaires par rapport au laser.

Le principe de fonctionnement du LMS est présenté ci-après :

Une diode laser émet un faisceau qui sera réfléchi par un miroir interne et la rotation de la tête du senseur autour de son axe crée les angles de balayage.

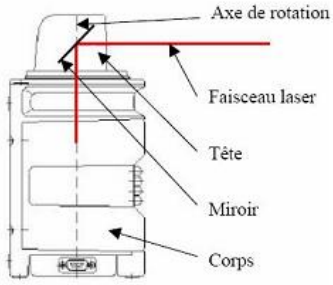


Figure 4.11 : Schéma descriptif du fonctionnement du LMS

Chaque fois qu'un faisceau est émis, un *timer* est automatiquement déclenché et ne s'arrête de compter que lorsque ce faisceau est réfléchit et reçu par le capteur. Le temps calculé représente le *temps de vol* et permet ensuite de calculer la distance entre le laser et l'objet en faisant une simple division. Dans le cas du non-retour de l'impulsion envoyée après un certain « temps mort » correspondant à la portée maximale du télémètre, aucun obstacle n'est à signaler dans la direction où le faisceau a été émis.

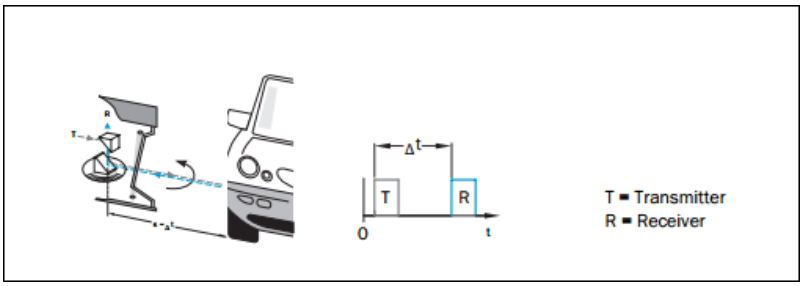


Figure 4.12 : Utilisation du temps de vol pour calculer la distance des obstacles

On retrouve dans la fiche technique plusieurs autres caractéristiques du laser résumées dans le tableau suivant :

Version	Mid Range
Modèle	Lite
Domaine d'application	Outdoor
Résolutions variées	Standard Resolution
Source lumineuse	Infrarouge (905 nm)
Classe laser	1, sans risque pour les yeux (IEC 60825-1 (2007-6))
Angle d'ouverture	190°
Fréquence de balayage	25 Hz / 35 Hz / 50 Hz / 75 Hz
Résolution angulaire	0,25° 0,5° 1°
Chauffages	Oui
Zone de fonctionnement	0 m ... 80 m
Portée pour 10 % de réflectivité	40 m
Taille de spot	11,9 mrad
Nombre d'échos évalués	2
Correction de brouillard	Oui

Figure 4.13 : Caractéristiques technique du télémètre laser SICK LMS 511

Pour notre application DATMO, on utilisera le capteur avec un angle de balayage de 180°, une résolution angulaire souvent fixée à 1° et une fréquence de 75Hz.

Les odomètres

Les odomètres sont des capteurs proprioceptifs qui mesurent la distance parcourue par le robot. Les odomètres les plus classiques sont constitués de codeurs incrémentaux entrainés par les roues du robot. A chaque tour de la roue, les codeurs s'incrémentent d'une unité. Ainsi, le déplacement est calculé en fonction du nombre de tours stockés dans le codeur et le diamètre de la roue en prenant en compte la structure mécanique et cinématique du robot.

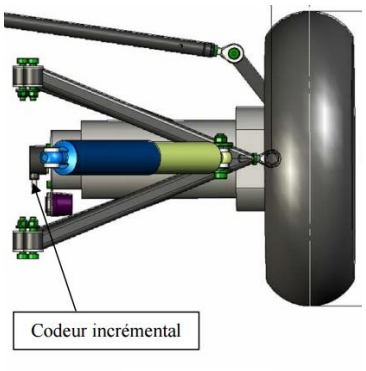


Figure 4.14 : Schéma descriptif du fonctionnement des odomètres

Ce système est très utilisé car très peu coûteux, fonctionne indépendamment de l'environnement et sa précision est bonne pour une distance parcourue moyenne. Cependant, si le robot parcourt une grande distance, les odomètres perdent considérablement leur précision car les erreurs de mesure s'accumulent continuellement et par conséquent, faussent complètement les résultats.

4.3 Implémentation du *grid-based* DATMO

La résolution du problème de détection et de suivi des objets mobiles en utilisant le filtre d'occupation bayésien a montrée plusieurs avantages par rapport aux approches classiques. En effet, le raisonnement basé sur l'abstraction de la notion d'*objet* et de *track* a permis de contourner plusieurs problèmes très complexes notamment celui de l'association de données.

La démarche d'implémentation du filtre d'occupation bayésien proposé dans [24] et décrite dans le chapitre précédant repose sur l'utilisation de la grille d'occupation de l'environnement observé par chaque capteur utilisé. Le filtre d'occupation bayésien prend ensuite le relai et estime pour chaque cellule la probabilité d'occupation et la distribution de probabilité sur les vitesses.

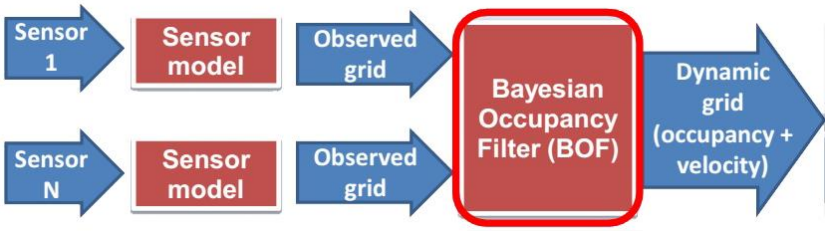


Figure 4.15 : Structure de détection et de suivi des obstacles mobiles en *grid-based* DATMO

En pratique, chacune des étapes de la figure ci-dessus sera implémentée sous forme d'un nœud ROS. Pour notre cas, on n'utilisera que le capteur laser et par conséquent cette structure sera décomposée comme suit :

- Nœud d'acquisition des données laser (existant) ;
- Nœud de construction de la grille d'occupation observée ;

– Nœud du filtre d'occupation bayésien (BOF).

Chacun des nœuds précédents sera décrit dans les sections qui suivent et son fonctionnement sera illustré par des simulations pour des expériences réelles (scénarios).

4.3.1 Acquisition des données laser

Le Robucar que nous utilisons pour l'implémentation du module DATMO est équipe d'un laser SICK LMS 511 placé sur la face avant à 30 centimètre du sol. Nous utiliserons ce capteur pour balayer le demi-plan horizontal situé entre -90° et +90° avec une résolution angulaire de 1°.

Pour fonctionner sous ROS, le télémètre laser a besoin d'un driver permettant de récupérer les mesures en coordonnées polaire et de les transcrire dans un type de données dédié à la représentation des données laser sous ROS. Ce type de données est développé par la communauté ROS et implémenté sous forme d'une class « LaserScan » en C++.

Ainsi, à chaque balayage de la scène, le driver (également développé par la communauté ROS) transcrit les mesures obtenues dans un objet (ou instance) de la class « LaserScan », objet qui sera ensuite publié dans un *topic* dénommé le plus souvent « */scan* ».

La figure 4.16 ci-dessous illustre un exemple d'acquisition de données à partir du laser (hall du CDTA). Ces données sont visualisées en utilisant « *rviz* » ; un outil intégré à ROS qui permet de représenter différents types de données notamment les scans laser et les grilles d'occupation.

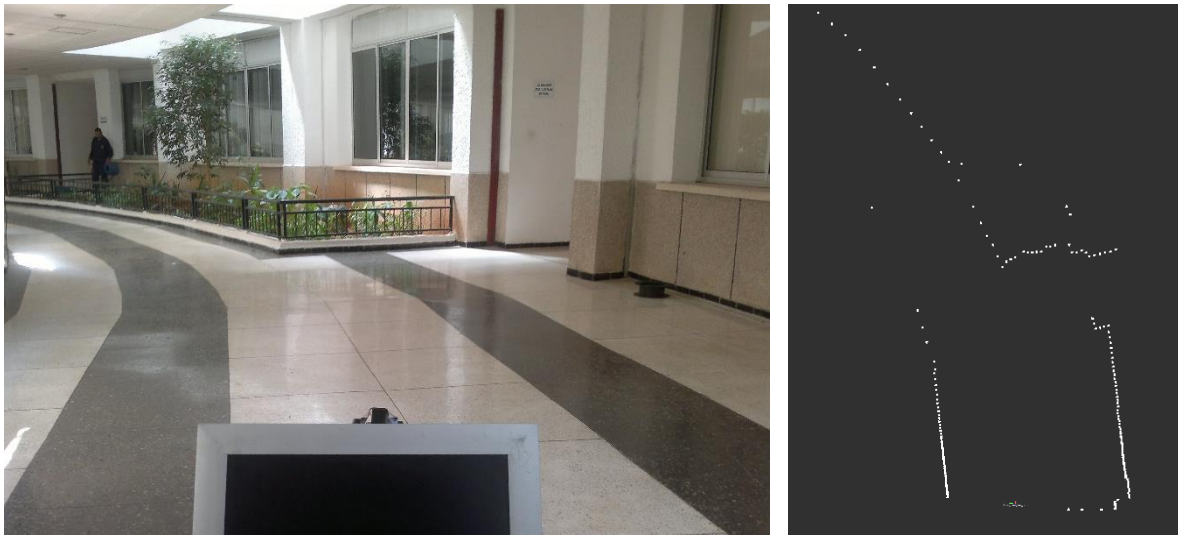


Figure 4.16 : Acquisition des données laser issues d'un balayage du hall du CDTA

Il est à souligner que l'utilisation d'un télémètre laser seul ne permet pas d'avoir une perception fiable de l'environnement étant donné que celui-ci est sensible à plusieurs facteurs dont les conditions atmosphériques. Il est très fréquent dans les véhicules autonomes d'utiliser plusieurs systèmes de perception en même temps et de fusionner ensuite les données obtenues.

Le plus souvent, on associe au télémètre laser un système de stéréovision (fusion des images de deux caméras) qui procure plus d'informations sur la géométrie de l'environnement.

4.3.2 Construction de la grille d'occupation

La scène perçue par le Robucar est modélisée en utilisant le formalisme des grilles d'occupation présenté dans le chapitre précédent. Ses dimensions sont choisies en fonction de la puissance de calcul dont nous disposons.

On utilise, pour la construction de notre grille d'occupation, la démarche proposée par S. Thrun dans [35] et détaillée dans le chapitre précédent. En résumé, cette démarche détermine l'état d'occupation de chaque cellule en se basant sur le faisceau laser le plus proche du centre de la cellule considérée.

La construction de la grille d'occupation locale de l'environnement du Robucar à partir des données fournies par le télémètre laser est réalisée par le nœud ROS « *local_map* ». La figure 4.17 ci-dessous représente un exemple de grille d'occupation construite à partir des données laser de l'expérience précédente (figure 4.16). L'environnement étant statique, on remarquera que l'état d'occupation des cellules est déterminé sans grande ambiguïté.

Figure 4.17 : Grille d'occupation modélisant le hall du CDTA

Comme nous l'avons souligné dans le chapitre précédent, l'état de certaines cellules se trouvant loin du robot est inconnu (probabilité d'occupation égale à 0.5). Ceci est causé par deux facteurs : la résolution angulaire du télémètre laser et la taille des cellules composant la grille d'occupation. Pour ce test, la résolution angulaire du laser a été fixée à 1° et la taille d'une cellule à 0.1m. Par conséquent, certaines cellules se trouvant loin du télémètre laser ne sont traversées par aucun des faisceaux laser émit ; il est donc impossible à l'algorithme d'estimer leur état d'occupation.

Il faut donc trouver un compromis entre la résolution angulaire et la taille des cellules en fonction de la portée qu'on veut atteindre avec le télémètre. Ce compromis doit également prendre en considération la puissance de calcul dont nous disposons.

Figure 4.18 : Grille d'occupation modélisant un environnement extérieur

La figure 4.18 ci-dessus représente un autre exemple de construction d'une grille d'occupation pour le cas d'un environnement extérieur. Dans la suite de ce document (sauf indication contraire), nous fixerons les paramètres de la grille d'occupation pour les environnements extérieurs comme suit :

- Longueur : 100 cellules ;
- Largeur : 80 cellules ;
- Taille d'une cellule : 0.20m × 0.20m.

Notre grille comporte donc 8 000 cellules et permet de couvrir une superficie de 320m^2.

4.3.3 Fast Classification of Static and Dynamic Environment

Dans la formulation *classique* du filtre d'occupation bayésien, les cellules sont traitées indifféremment de leur nature cinématique ; l'algorithme estime une distribution de probabilité de vitesse pour les cellules statiques et les cellules dynamiques, ce qui augmente le nombre d'hypothèses relatives aux cellules antécédentes d'une cellule donnée. De plus, le BOF est réputé pour être coûteux en terme de temps de calcul étant donné qu'il considère pour chaque cellule un nombre d'hypothèses égale aux nombre de cellule de la grille. A titre d'exemple, pour une grille de 10 000 cellules, le BOF considère pour chacune des 10 000 cellules au moins 10 000 hypothèses soit un total d'au moins 10^8 hypothèses.

On constate donc qu'il est inutile que le filtre d'occupation bayésien estime la distribution de probabilité sur les vitesses des cellules statiques. Qadeer Baig est partie du même constat et explique dans [40] [41] qu'il est possible d'alléger le traitement dans la structure générale de l'algorithme du DATMO (figure 4.15) en y ajoutant simplement une étape intermédiaire entre la construction de la grille d'occupation et l'application du filtre d'occupation bayésien. Cette étape consiste en un prétraitement sur la grille d'occupation en faisant une classification rapide des cellules statiques et des cellules dynamiques présentes dans l'environnement (*fast classification of static and dynamic environment*) comme illustré dans la figure 4.19 ci-dessous.

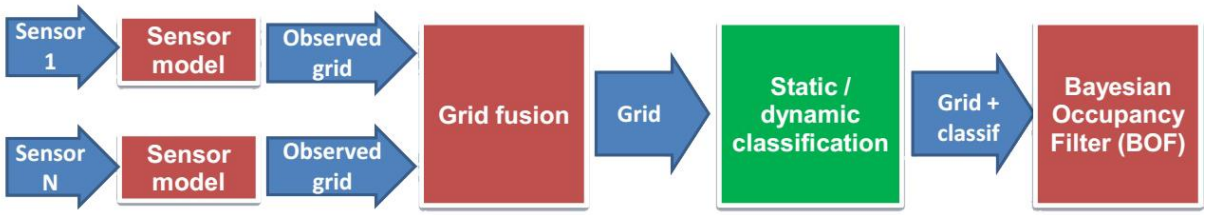


Figure 4.19 : *Grid-based* DATMO avec utilisation d'un algorithme de classification rapide

Dans cette nouvelle structure, le filtre d'occupation bayésien prendra en entrée le résultat du prétraitement réalisé par le module de classification rapide (aussi appelé *motion detection*) et seules les distributions de probabilité sur les vitesses des cellules dynamiques seront estimées par le BOF.

Le module de classification rapide des objets sera implémenté sous forme d'un nœud « *fast_classification_node* » sous ROS.

4.3.3.1 Présentation de l'algorithme

Comme expliqué précédemment, le module « *motion detection* » a pour objectif de classifier les cellules de la grille d'occupation préalablement construite par le nœud « *local_map* » en cellules statiques et cellules dynamiques. Le principe de cette méthode est illustré ci-dessous.

Notons OG_t la grille d'occupation construite par le nœud « *local_map* » à l'instant t. Notons également $OG_t[i]$ la probabilité d'occupation de la $i^{ème}$ cellule de la grille à l'instant t avec $0 \leq i \leq N$ et où N représente le nombre total de cellules dans la grille.

Enfin, notons $u_t = (v_t, w_t)$ le vecteur vitesse du robot à l'instant t. Ce vecteur est obtenu en utilisant des capteurs proprioceptifs tels qu'une centrale inertielle (IMU) ou par la technique ICP-SLAM par exemple.

L'algorithme du module de classification rapide des cellules consiste à exécuter à chaque instant t les étapes suivantes :

a) Création des compteurs

Pour chaque cellule de la grille d'occupation, deux compteurs sont créés : *OccupiedCount* et *FreeCount*. Ces compteurs enregistrent le nombre de fois qu'une cellule a été perçue respectivement comme étant occupée et comme étant libre.

La valeur de ces compteurs à l'instant t est déterminée à partir de la probabilité d'occupation de la cellule calculée par le nœud « *local_map* » :

$$OccupiedCount_t[i] = \begin{cases} 1 \ si \ OG_t[i] > 0.5 \\ 0 \ sinon \end{cases}$$

$$FreeCount_t[i] = \begin{cases} 1 \ si \ OG_t[i] < 0.5 \\ 0 \ sinon \end{cases}$$

b) Mise à jour des compteurs

La mise à jour des compteurs définis précédemment nécessite de mettre en correspondance la grille d'occupation à l'instant t avec celle de l'instant $t-1$. Cette mise en correspondance consiste à déterminer la transformation homogène qui ramènerait la grille d'occupation $OG_t[i]$ au même repère que $OG_{t-1}[i]$ en utilisant le vecteur $u_t = (v_t, w_t)$.

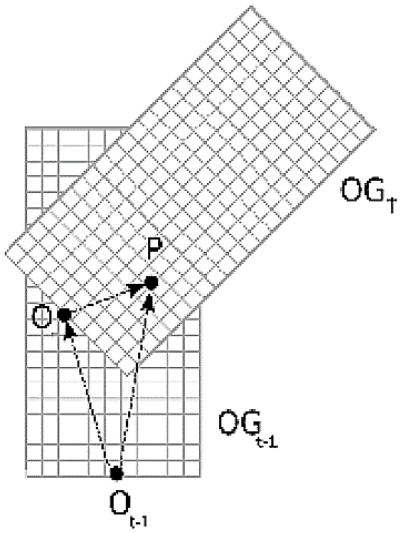


Figure 4.20 : Mise en correspondance des grilles $OG_t[i]$ et $OG_{t-1}[i]$

Cette transformation permet de mettre en correspondance une cellule i de la grille $OG_{t-1}[i]$ avec une cellule j de la grille $OG_t[i]$.

En notant $OccupiedCount_{t-1}[i]$ et $FreeCount_{t-1}[i]$ les compteurs définis précédemment mais à l'instant $t-1$, il est alors possible de mettre à jour ces compteurs comme suit :

$$FreeCount_t[j] = FreeCount_t[j] + FreeCount_{t-1}[i]$$

$$OccupiedCount_t[j] = OccupiedCount_t[j] + OccupiedCount_{t-1}[i]$$

c) Détermination des cellules dynamique

Une fois que les compteurs ont été mis à jour, on calcul une grille dénommée « *motionGrid* » comme suit :

$$motionGrid_t[i] = \begin{cases} 1 & si\ OG_t[i] \geq 0.5\ et\ FreeCount_t[i] > 2 \times OccupiedCout_t[i] \\ 0 & sinon \end{cases}$$

Ainsi, une cellule est considérée dynamique si elle est occupée $(OG_t[i] \geq 0.5)$ à l'instant t et qu'elle a été perçue libre deux fois plus qu'elle n'a été perçue occupée durant les instants précédents.

La grille *motionGrid* obtenue contient donc des « 1 » dans les cellules détectées comme étant dynamiques et appartenant à des objets en mouvement et des « 0 » ailleurs.

Nous soulignons qu'une amélioration de cette méthode a été proposée dans [41] en y ajoutant un *score* permettant un meilleur recalage des grilles $OG_t[i]$ et $OG_{t-1}[i]$.

4.3.3.2 Illustration

Dans cette section, nous allons tester les performances du nœud « *fast_classification_node* » développé sous ROS en le combinant au nœud « *local_map* » présenté précédemment. Le premier nœud récupère la dernière grille d'occupation publiée dans le *topic* « */local_map* » et exécute l'algorithme définit précédemment.

Afin d'illustrer les performances de l'algorithme, nous avons esquissé plusieurs scénarios représentant les différents environnements auxquels peut être confronté le robot.

Le premier scénario consiste à mettre le robot dans les conditions d'un environnement faiblement dynamique (un seul objet en mouvement) afin d'illustrer le fonctionnement de l'algorithme.

Scénario 1

« Le Robucar à l'arrêt observant la scène devant lui en utilisant les données fournies par le télémètre laser. Un piéton traverse la chaussée, va jusqu'à l'autre bout, s'arrête et repart dans la direction du Robucar. Les murs et les buissons représentent des obstacles statiques. »

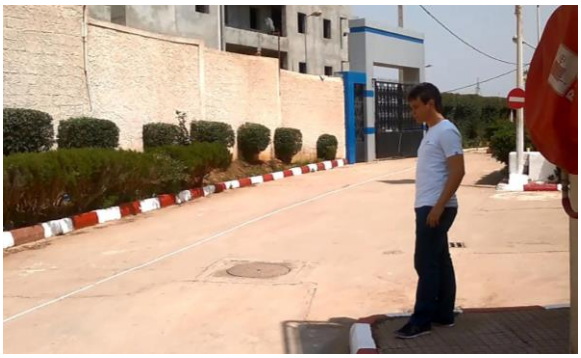
Cette première expérience nécessite que les nœuds ROS suivants soient lancés :

- Nœud d'acquisition des données laser « *sickLMS5xx* »;
- Nœud de construction de la grille d'occupation locale « *local_map* » ;
- Nœud de classification rapide des cellules statiques et dynamiques « *fast_classification_node* ».

Afin d'illustrer les résultats obtenus du déroulement de ce scénario, nous avons choisis quatre instants *clés* correspondant aux différentes phases du mouvement. Les petits points blancs dans les figures (*1-4.b*) et (*1-4.c*) représentent les scans laser. Les cellules statiques sont représentées dans les figures du milieu (*1-4.b*) et les cellules dynamiques sont représentées en jaune dans les figures (*1-4.c*).

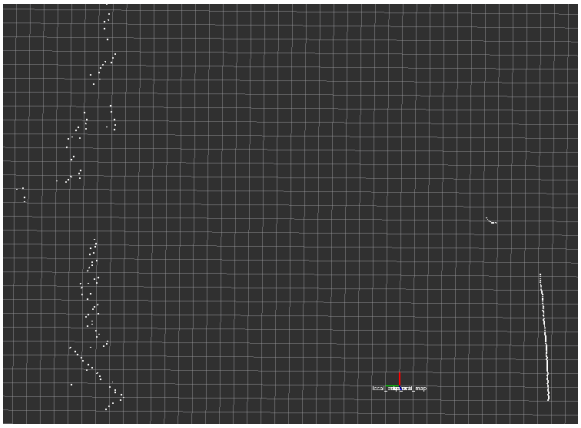
Etant donné que le scénario se déroule dans un environnement extérieur, la résolution du télémètre laser est fixée à 1° et la grille d'occupation est construite avec une résolution de 0.20 *mètre*.

instant t_1



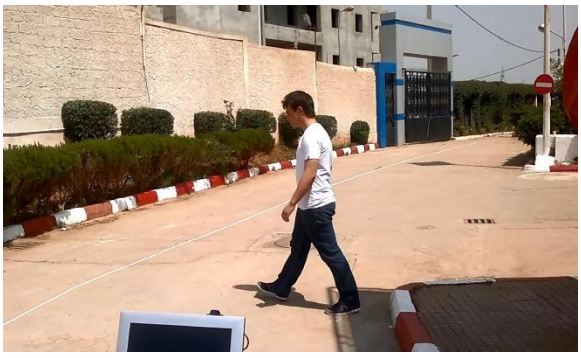
(1.a) : scène réelle

(1.b) : cellules statiques

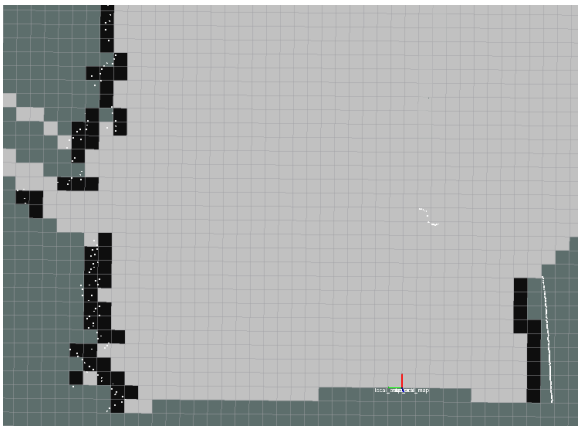


(1.c) : cellules dynamiques

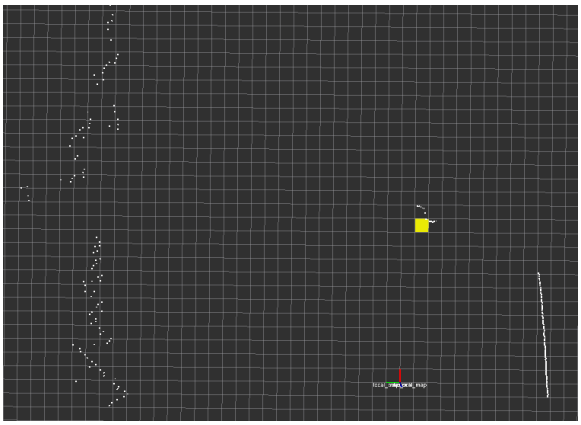
instant t_2



(2.a) : scène réelle

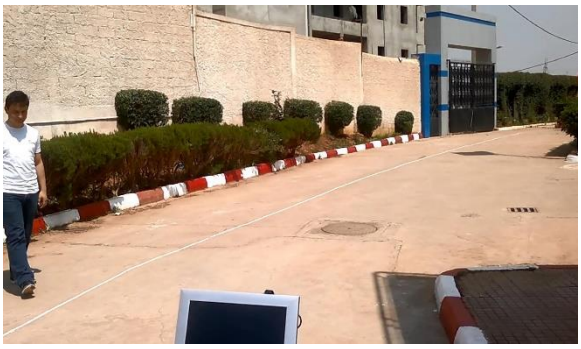


(2.b) : cellules statiques

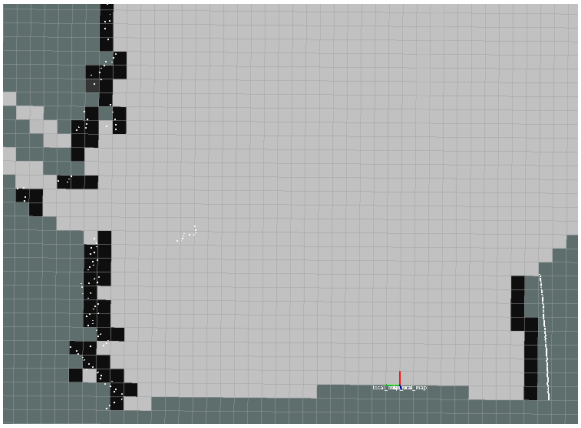


(2.c) : cellules dynamiques

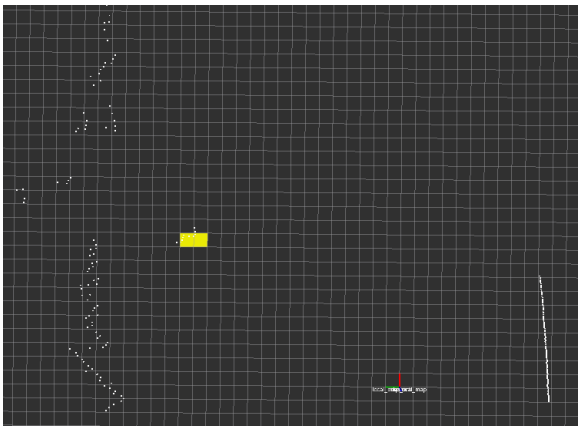
instant t_3



(3.a) : scène réelle



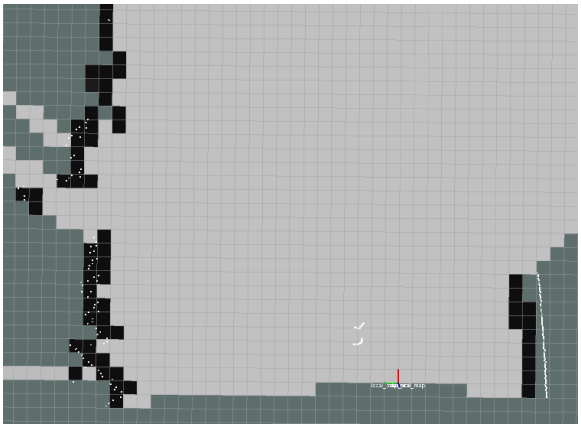
(3.b) : cellules statiques



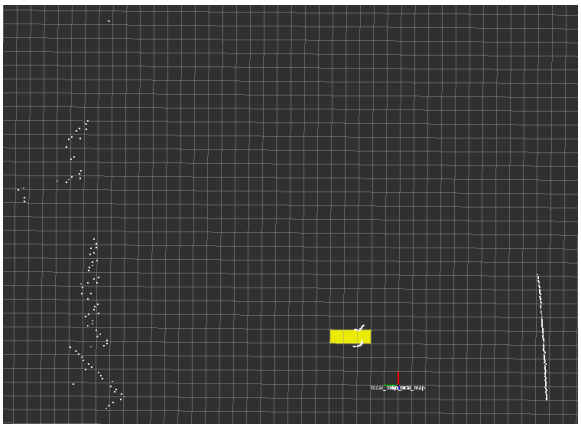
(3.c) : cellules dynamiques

instant t_4

(4.a) : scène réelle



(4.b) : cellules statiques



(4.c) : cellules dynamiques

Figure 4.21 : Résultats obtenus du déroulement du scénario 1

Commentaire

A l'instant t_1 le piéton n'ayant pas encore bougé est détecté comme étant un objet statique. Dès qu'il se met en mouvement à l'instant suivant (instant t_2) les cellules qu'il occupe disparaissent quasi-instantanément de la carte statique et apparaissent sur la carte dynamique. Durant tout le mouvement (instants t_2, t_3 et t_4) le piéton n'apparait que dans la carte dynamique.

On peut d'emblée constater l'efficacité de l'algorithme de classification rapide puisque le piéton (représentant un objet dynamique) a été détecté dès qu'il a commencé à se mouvoir et n'a pas été perdu durant toute l'expérience.

Nous allons à présent mettre le robot dans les conditions d'un environnement dynamique présentant des objets en mouvement permanent et des objets momentanément en mouvement.

Scénario 2

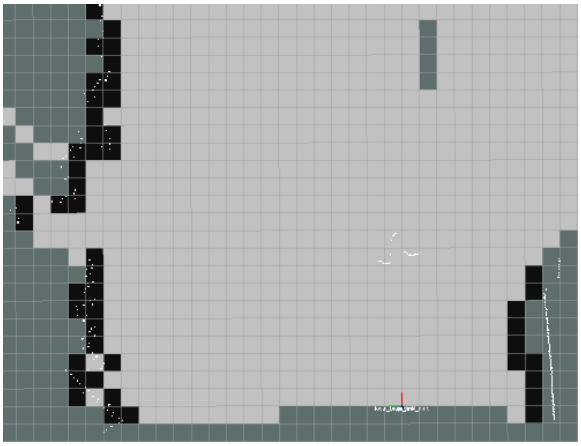
« Le Robucar à l'arrêt observant la scène devant lui en utilisant les données fournies par le télémètre laser. Un piéton tenant un plot dans sa main droite traverse la chaussée, parcourt une certaine distance et s'arrête, dépose le plot sur le sol et repart en le laissant derrière lui »

Cette deuxième expérience nécessite de lancer les mêmes nœuds ROS qu'on a lancé lors de l'expérience précédente. Nous gardons également les mêmes paramètres d'environnement (résolution angulaire du télémètre et résolution des cellules).

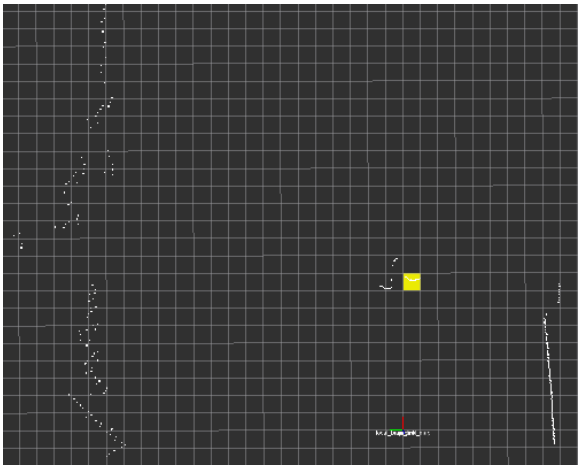
Les résultats obtenus du déroulement de ce scénario sont illustrés, en utilisant la même convention du scénario 1, dans la figure 4.22 ci-dessous.

instant t_1

(1.a) : scène réelle



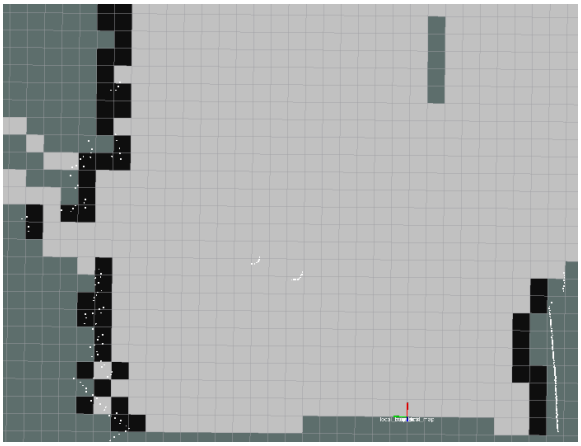
(1.b) : cellules statiques



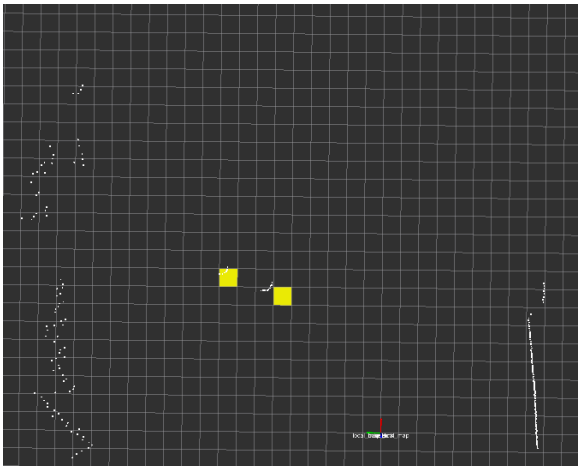
(1.c) : cellules dynamiques

instant t_2

(2.a) : scène réelle

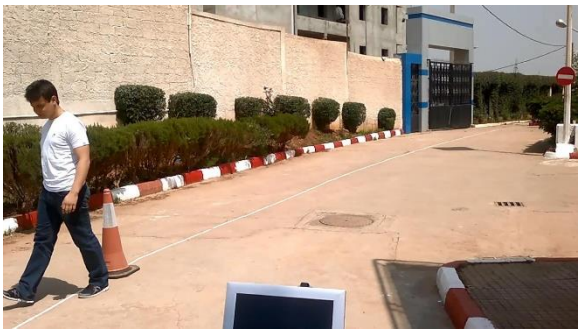


(2.b) : cellules statiques

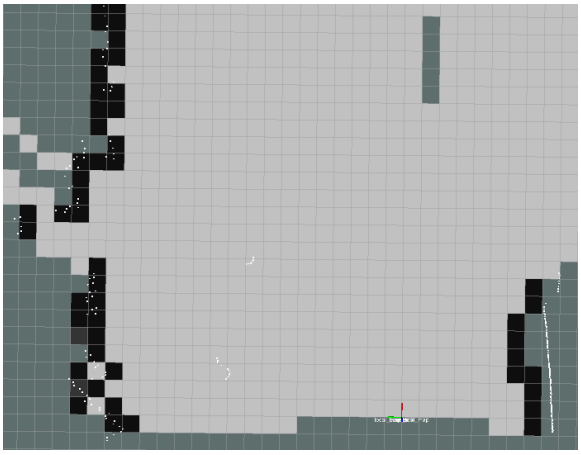


(2.c) : cellules dynamiques

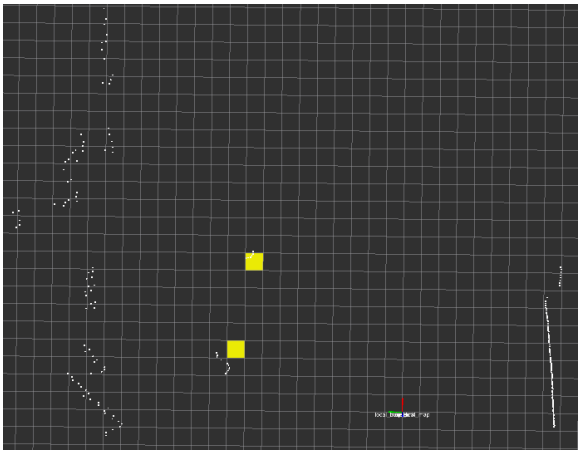
instant t_3



(3.a) : scène réelle



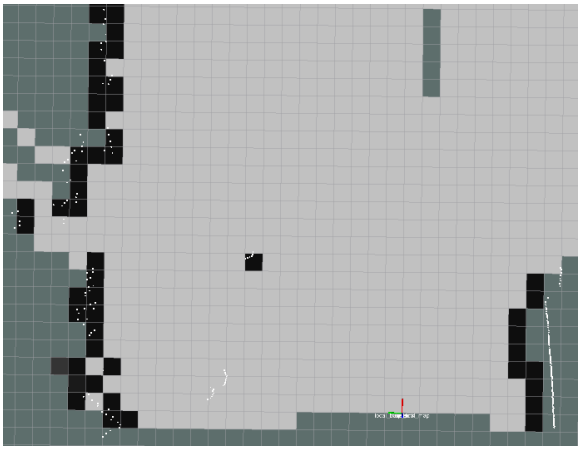
(3.b) : cellules statiques



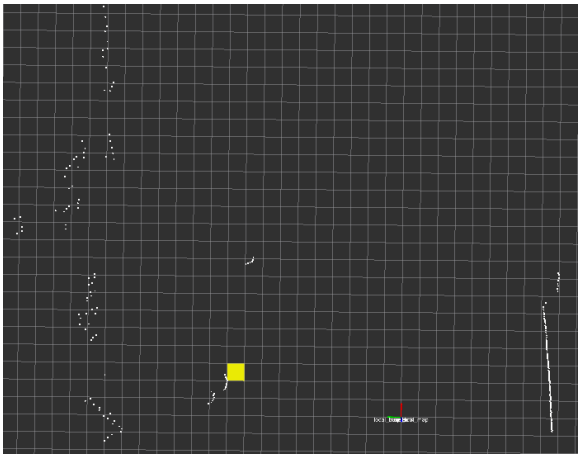
(3.c) : cellules dynamiques

instant t_4

(4.a) : scène réelle



(4.b) : cellules statiques



(4.c) : cellules dynamiques

Figure 4.22 : Résultats obtenus du déroulement du scénario 2

Commentaire

Comme lors de l'expérience précédente, les objets en mouvement sont détectés sans grande ambigüité et ne sont à aucun moment perdus de vue. De plus, un résultat important est à souligner ; le *plot* en mouvement à l'instant t_1 et déposé sur le sol par le piéton à l'instant t_2 apparait comme un objet dynamique durant ces deux instants. A l'instant t_3, le piéton continue son chemin laissant derrière lui le *plot* et les deux objets sont détectés comme étant dynamiques.

Un très bref instant plus tard (correspondant à environs deux pas du piéton), à l'instant t_4, le *plot* disparait de la carte dynamique et apparait comme un objet statique.

Ce dernier résultat est très important étant donné qu'il démontre que l'algorithme de « *motion detection* » est capable d'éliminer les obstacles momentanément à l'arrêt (ou momentanément en mouvement) de la carte dynamique.

Nous allons maintenant entamer un dernier test en mettant le robot dans les conditions d'un environnement fortement dynamique où plusieurs objets y manœuvrent dans des directions différentes.

Scénario 3

« Le Robucar à l'arrêt observant la scène devant lui en utilisant les données fournies par le télémètre laser. Deux personnes et une chaise se trouvent devant lui et exécutant des mouvements rapides et dans toutes les directions ».

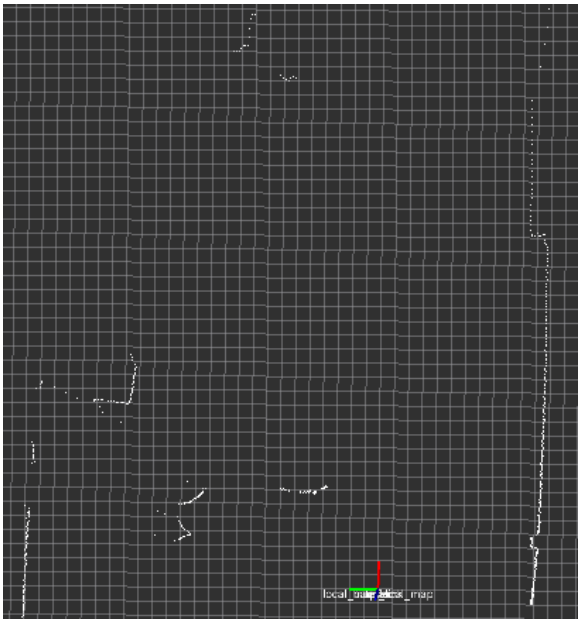
Cette expérience nécessite de lancer les mêmes nœuds ROS que lors des expériences précédentes. Néanmoins, étant donné que cette dernière expérience se déroule dans un environnement intérieur, et afin de déterminer efficacement l'état d'occupation de chaque cellule, nous règleront la résolution angulaire du laser à 0.25°.

Les résultats obtenus du déroulement de ce scénario sont illustrés, en utilisant la même convention que précédemment, dans la figure 4.23 ci-dessous.

instant t_1

(1.a) : scène réelle

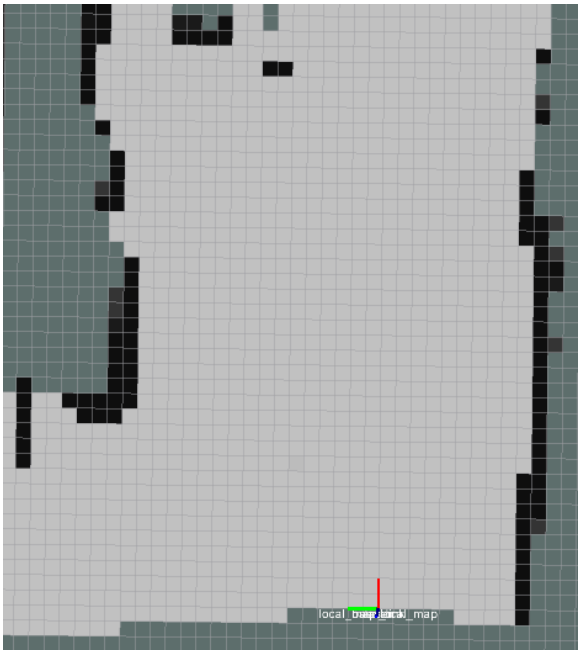
(1.b) : cellules statiques



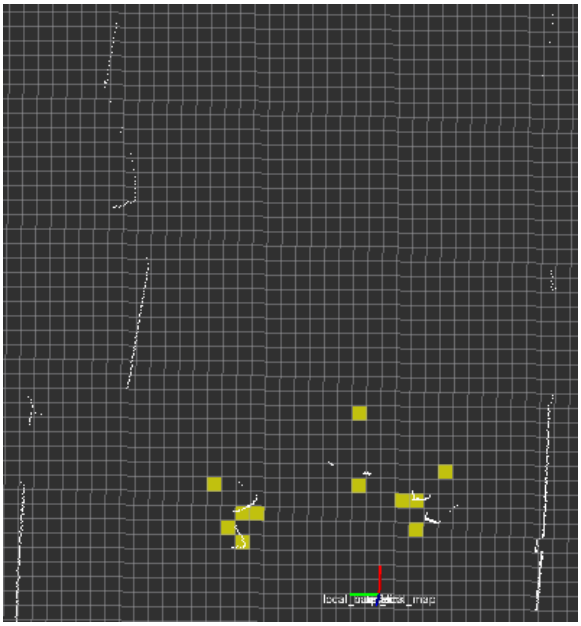
(1.c) : cellules dynamiques

instant t_2

(2.a) : scène réelle



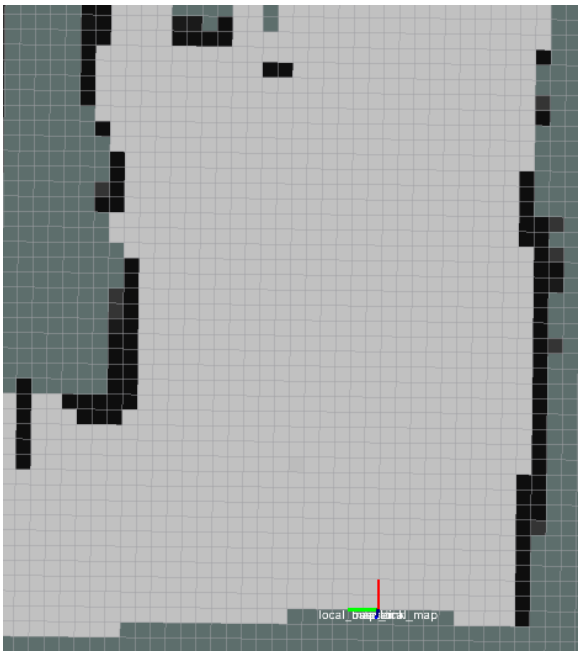
(2.b) : cellules statiques



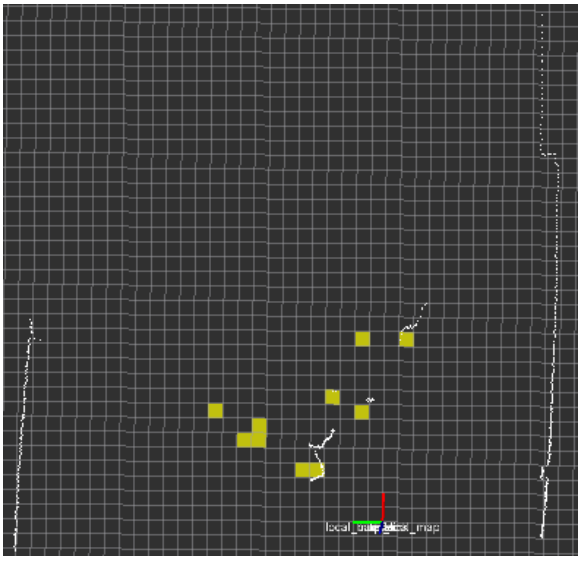
(2.c) : cellules dynamiques

instant t_3

(3.a) : scène réelle



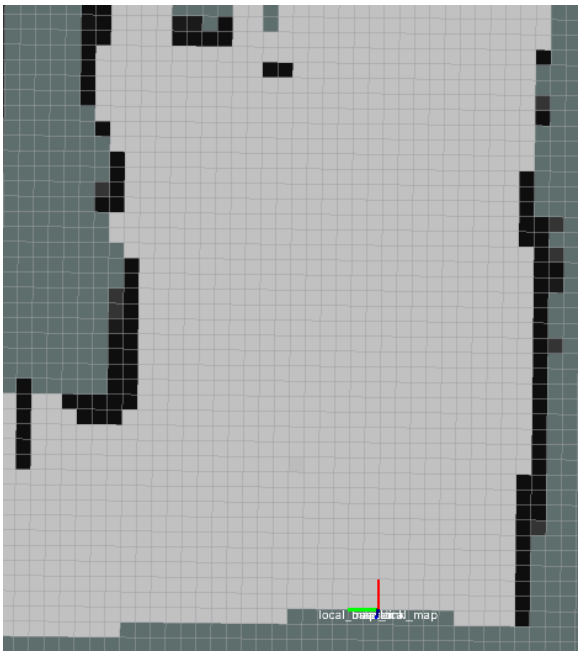
(3.b) : cellules statiques



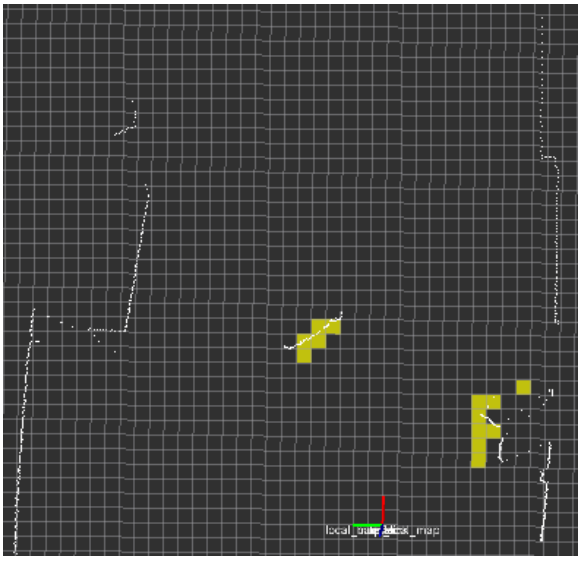
(3.c) : cellules dynamiques

instant t_4

(4.a) : scène réelle



(4.b) : cellules statiques



(4.c) : cellules dynamiques

Figure 4.23 : Résultats obtenus du déroulement du scénario 3

Commentaire

Cette dernière expérience nous a permis de constater la robustesse de l'algorithme lorsque l'environnement est non-structuré et contient des objets à dynamique variable. En effet, les cellules dynamiques ne sont jamais perdues de vu et à aucun moment elles ne disparaissent de la carte dynamique.

4.3.3.3 Conclusion sur l'algorithme de classification rapide des cellules

Le déroulement des trois expériences précédentes nous a permis de constater l'efficacité et la robustesse de l'algorithme de classification rapide des cellules statiques et des cellules dynamiques. Nous remarquons d'emblée que l'utilisation de cet algorithme permettra de réduire considérablement le nombre de cellules que le filtre d'occupation bayésien aura à traiter et de réduire ainsi l'inférence sur la vélocité des cellules uniquement.

4.3.4 Implémentation du Bayesian Occupancy Filter (BOF)

Le filtre d'occupation bayésien est implémenté sous forme d'un nœud ROS « *bof_node* » qui utilise les mesures fournies par le télémètre laser disponibles dans le *topic* « */scan* ».

Dans un premier temps, nous allons tester le filtre d'occupation bayésien appliqué directement à la grille d'occupation (sans prétraitement). Ceci nous permettra notamment de constater les performances du filtre appliqué seul et de justifier l'utilisation d'un algorithme de prétraitement.

4.3.4.1 Champ des vitesses

Comme nous l'avons mentionné au chapitre précédent, la valeur de vitesse que peut prendre chaque cellule est prise dans un champ de vitesses imposé par le concepteur du filtre suivant la connaissance qu'il a de l'environnement dans lequel le robot est amené à évoluer.

Ce champ est tel qu'il n'autorise que les vitesses correspondantes à un déplacement d'un nombre entier de cellules dans un pas de temps donné.

Pour notre cas, nous avons choisis un champ de vitesses à huit composantes correspondant à un déplacement d'un cellule à gauche, à droite, vers le haut, vers le bas et suivant les deux diagonales. En d'autres termes, entre deux itérations du filtre, une cellule dynamique ne doit bouger que d'une seule case au maximum et ce dans les 4 directions possibles.

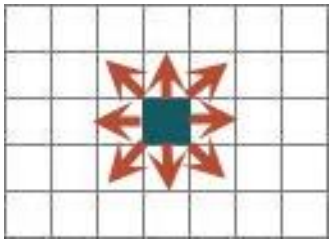


Figure 4.24 Champ des vitesses choisis

4.3.4.2 Illustration

Afin d'illustrer les résultats de l'implémentation du filtre d'occupation bayésien, nous procédons comme précédemment en esquissant plusieurs scénarios représentant différentes situations auxquelles peut être confronté le Robucar.

Le premier scénario consiste à mettre le robot dans les conditions d'un environnement statique afin d'illustrer le fonctionnement de la boucle *prédiction-estimation*.

Scénario 4

« Le Robucar à l'arrêt observant la scène devant lui (salle des robots du CDTA) en utilisant les données fournies par le télémètre laser. La scène observée contient un carton, un robot B21r, une armoire et des chaises ».

Cette première expérience nécessite que les nœuds ROS suivants soient lancés :

- Nœud d'acquisition des données laser « *sickLMS5xx* »;
- Nœud du filtre d'occupation bayésien « *bof_node* ».

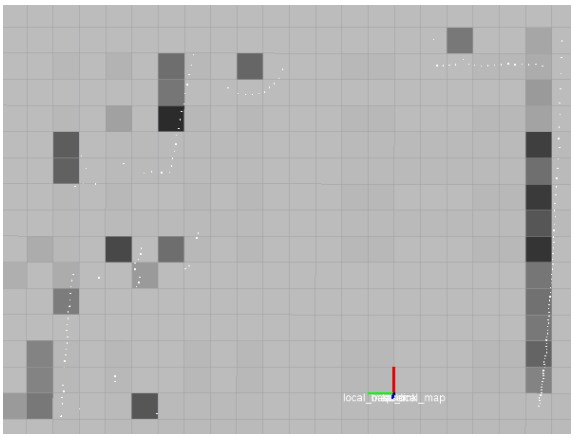
Afin d'illustrer les résultats obtenus du déroulement de ce scénario, nous avons choisis trois instants *clés* correspondant à trois itérations du filtre (trois boucles *prédiction-estimation*). Comme pour les expériences précédentes, les petits points blancs représentent les scans laser et l'état d'occupation des cellules est représenté avec des nuances de gris (blanc pour une cellule libre et noir pour une cellule occupée).

Etant donné que le filtre d'occupation bayésien est réputé pour être coûteux en termes de temps de calcul, nous allons régler pour cette expérience la résolution du télémètre à 1° et les dimensions de la grille d'occupation sont fixées comme suit :

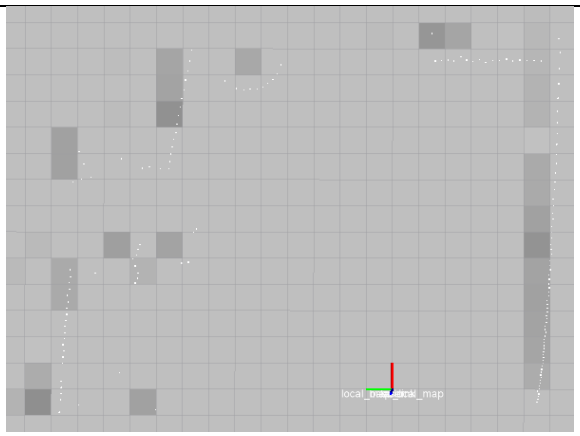
- Nombre de cellules en largeur : 30 cellules ;
- Nombre de cellule en hauteur : 30 cellules ;
- Résolution d'une cellule : 0.2 mètres.

Les résultats obtenus du déroulement de ce scénario sont illustrés dans la figure 4.26 ci-dessous.

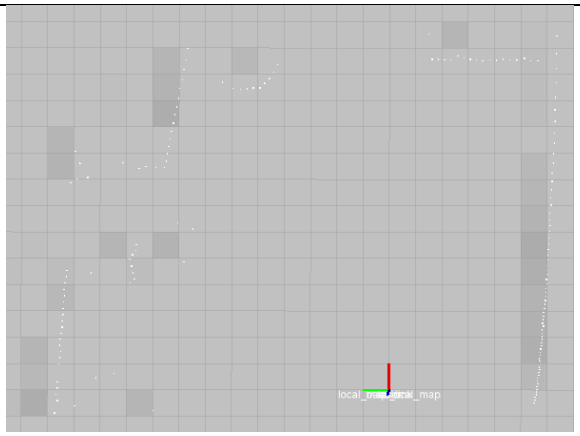
Figure 4.25 : Scène observée par le robot et décrite dans le scénario 4



(a) instant t_1



(b) instant t_2



(c) instant t_3

Figure 4.26 : Résultats obtenus du déroulement du scénario 4

Commentaire

Le filtre d'occupation bayésien élimine toutes les cellules correspondantes aux objets statiques en leurs attribuant des probabilités d'occupation faibles (inférieurs à 0.5) qui s'atténuent ensuite continuellement à chaque itération.

Il est à souligner que le processus de filtrage des cellules statiques est lent comparé aux résultats de l'algorithme de classification rapide illustré dans la section précédente. En effet, on remarque que les cellules statiques présentent à l'instant t_1 ne sont filtrées par le BOF qu'après plusieurs itérations alors que l'algorithme de classification rapide les élimine quasi-instantanément.

Les résultats de cette première expérience donnent donc une première justification quant à l'utilisation de l'algorithme de classification rapide en vue de l'amélioration du temps de calcul.

Nous allons maitenant mettre le robot dans les conditions d'un environnement dynamique avec un piéton en mouvement.

Scénario 5

« Le Robucar à l'arrêt, dans un environnement intérieur, observant la scène devant lui en utilisant les données fournies par le télémètre laser. Un piéton passe devant lui, parcourt une certaine distance, s'arrête, se retourne et repart dans la direction du Robucar ».

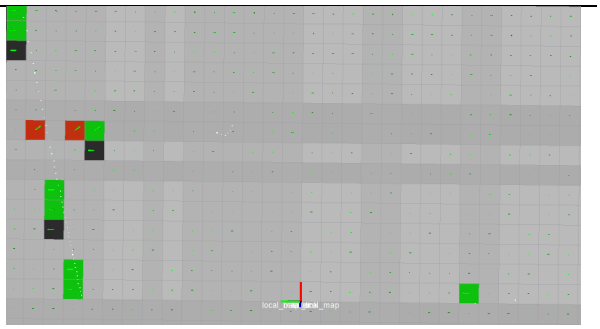
Cette deuxième expérience nécessite de lancer les mêmes nœuds ROS qu'on a lancé lors de l'expérience précédente. Nous gardons également les mêmes paramètres d'environnement que précédemment (résolution angulaire du télémètre et résolution des cellules).

Les résultats obtenus du déroulement de ce scénario sont illustrés dans la figure 4.27 ci-dessous. Nous soulignons que nous avons choisi de représenter, pour chaque cellule, uniquement

le vecteur vitesse ayant la plus grande probabilité. Ce vecteur est représenté par les petites flèches vertes.

instant t_1

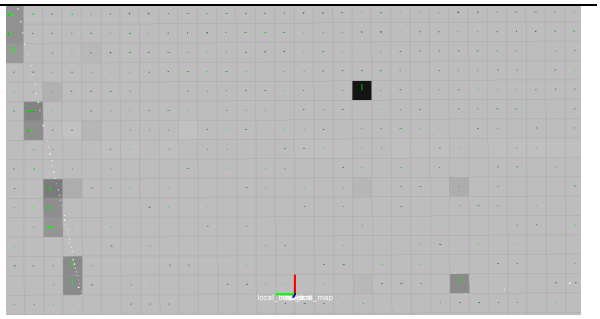
(1.a) scène réelle



(1.b) résultat du BOF

instant t_2

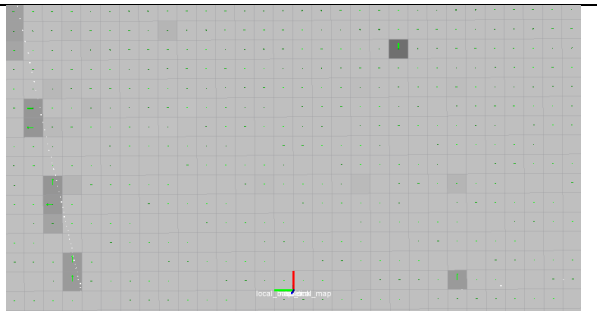
(2.a) scène réelle



(2.b) résultat du BOF

instant t_3

(3.a) scène réelle



(3.b) résultat du BOF

Figure 4.27 : Résultats obtenus du déroulement du scénario 5

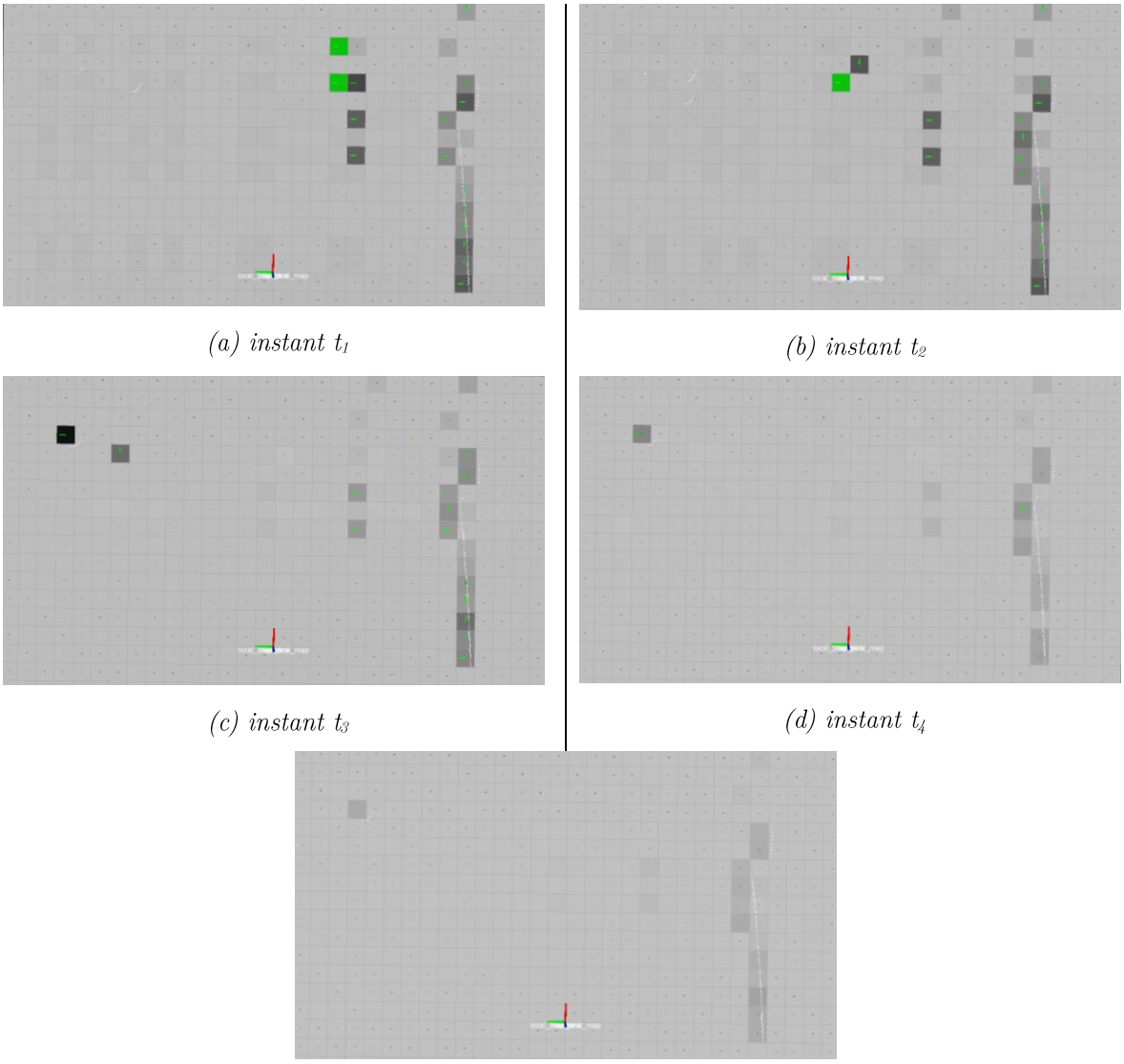
Commentaire

Les résultats donnés par le filtre d'occupation bayésien reflètent le déroulement du scénario ; l'estimation de l'occupation est correcte et l'estimation des vitesses s'accorde avec la direction du mouvement. Cependant, le temps d'exécution de la boucle *prédiction-estimation* est d'environ 2 secondes (exécution avec un processeur Intel i5 et 4Gb de RAM) et ne répond pas aux contraintes

de fonctionnement en temps réel. Ce dernier point sera abordé plus en détail dans les conclusions qui vont suivre.

Intéressons-nous à présent au comportement du filtre d'occupation bayésien lorsqu'un objet est momentanément en mouvement. Nous allons donc reprendre le scénario 2 précédent en utilisant le nœud ROS « *bof_node* ».

Afin d'illustrer les résultats obtenus du déroulement de ce scénario, nous avons choisis cinq instants *clés* correspondant à cinq itérations du filtre (cinq boucles *prédiction-estimation*). Les résultats obtenus sont illustrés dans la figure ci-après.



(a) instant t_1

(b) instant t_2

(c) instant t_3

(d) instant t_4

(e) instant t_5

Figure 4.28 : Résultats obtenus du déroulement du scénario 2

Commentaire

Les résultats de l'estimation de l'occupation et de la vitesse des cellules s'accordent avec le déroulement du scénario. Nous remarquons notamment que dès que le *plot* est à l'arrêt, il est tout de suite filtré par le BOF : sa vitesse devient nulle et sa probabilité d'occupation s'atténue aux itérations suivantes.

Il est à souligner que durant cette expérience, une erreur de prédiction est commise par le filtre d'occupation bayésien à l'instants t_2 où le filtre attribue à une cellule une vitesse dont la direction ne correspond pas au mouvement. Cette erreur est due à une projection par le champ de vitesse de l'occupation des cellules correspondantes aux murs. Néanmoins, l'erreur est corrigée dès l'itération suivante, ce qui reflète le mécanisme de fonctionnement du BOF qui apprend continuellement de ses expériences.

4.3.4.3 Conclusion sur le filtre d'occupation bayésien

Un résultat important de l'implémentation du filtre d'occupation bayésien est que ce filtre propose une solution complète au problème du DATMO dans la mesure où il permet de détecter les cellules dynamiques et d'estimer leur vitesse.

Comme nous l'avons souligné dans les commentaires des expériences précédentes, le filtrage bayésien est couteux en termes de temps de calcul (estimé à environ 2 secondes/itération avec un microprocesseur Inter i5 et 4Gb de RAM). Ceci est essentiellement dû au nombre important d'hypothèses considérées par le filtre (plus de 6 480 000 hypothèses dans notre cas) étant donné que le BOF est appliqué sur toutes les cellules quelle que soit leur nature (statique ou dynamique). Une telle implémentation ne répond pas aux exigences d'un fonctionnement en temps réel notamment dans le cas d'un environnement non-structuré ou fortement variable.

Il est donc nécessaire d'appliquer un prétraitement sur les cellules de sorte qu'elles ne soient pas toutes traitées par le BOF. En effet, dans la quasi-totalité des environnements, on retrouve plus de cellules statiques (murs, immobilier, bords de la route, ...) que de cellules dynamiques, il est donc indispensable de procéder à une classification des cellules avant d'appliquer le filtre d'occupation bayésien.

Ainsi, de ce manière, le problème du DATMO est résolu en deux étapes ; la détection des cellules et leur classification en cellules statiques et cellules dynamiques est assurée par l'algorithme de « *fast classification* » et l'estimation de la vélocité des cellules dynamiques est assurée par le filtre d'occupation bayésien.

Cette approche est utilisée dans quasiment toute les implémentations récentes du BOF et permet de réduire considérablement le temps d'exécution de la boucle *prédiction-estimation* et de profiter pleinement des avantages de la représentation de l'environnement par une grille d'occupation.

Les détails et les résultats expérimentaux obtenus de l'utilisation du filtre d'occupation bayésien avec l'algorithme de classification rapide sont illustrés dans la section suivante.

4.3.5 Filtre d'occupation bayésien utilisé avec l'algorithme de classification rapide

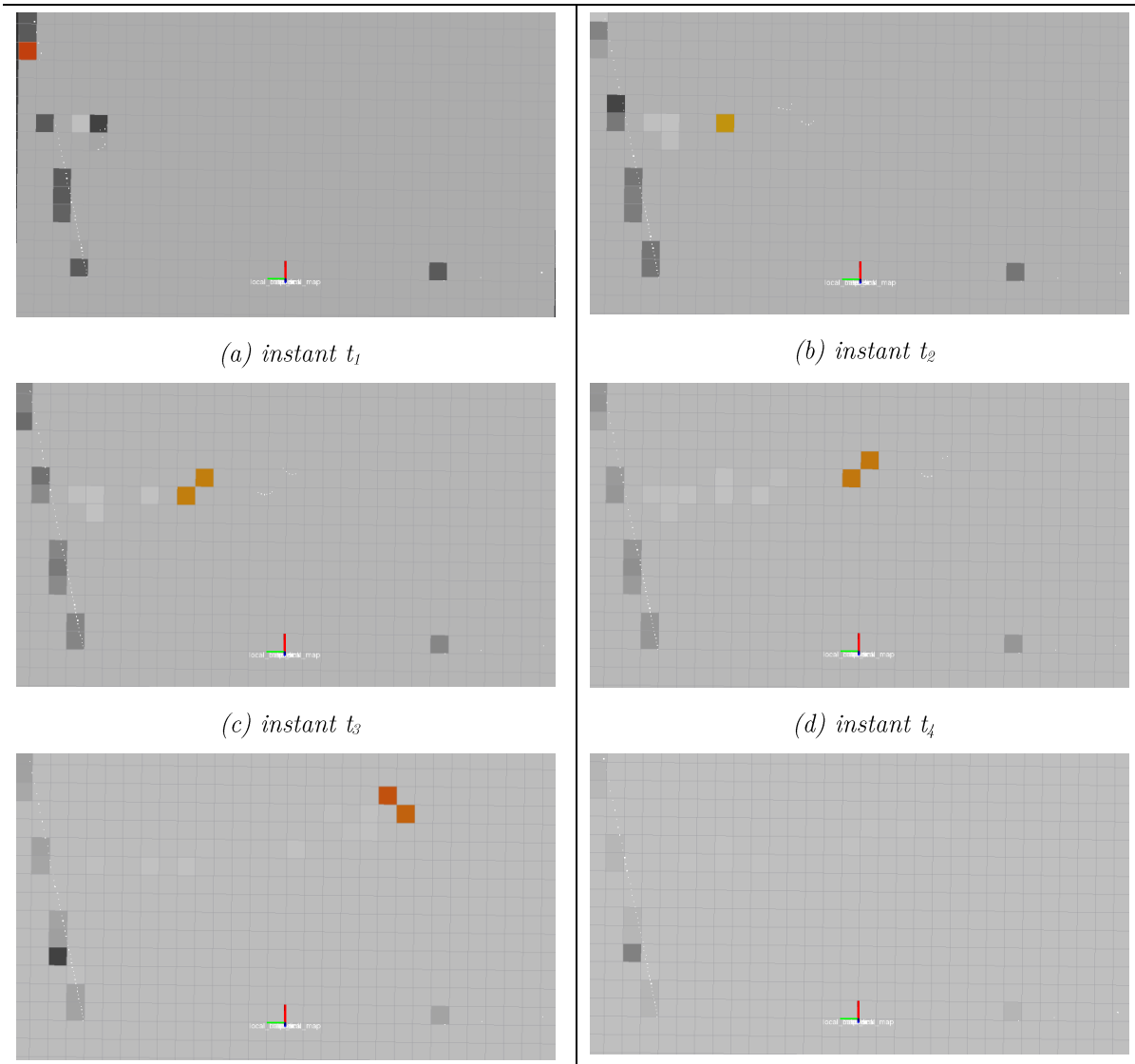
Nous venons de le voir, le filtre d'occupation bayésien est coûteux en terme de temps de calcul et ne répond donc pas aux exigences d'un fonctionnement en temps réel. Ainsi, associer le BOF avec l'algorithme faisant la séparation entre les cellules statiques et les cellules dynamiques présenté et illustré dans la section précédente devient une nécessité.

Cette démarche nous permettra d'éviter d'estimer la distribution de vélocité pour toutes les cellules mais uniquement celles des cellules classées comme étant dynamiques par l'algorithme de classification rapide. Nous verrons plus loin que le gain de temps est considérable.

4.3.5.1 Illustration

A présent, nous illustrons les résultats obtenues de l'association du filtre d'occupation bayésien à l'algorithme de classification rapide. Nous utiliserons le scénario 5 précédent (piéton dans un environnement intérieur) et nous comparerons ensuite les résultats à ceux obtenus en utilisant le BOF seul.

Les résultats obtenus du déroulement de ce scénario sont illustrés, en utilisant la même convention que précédemment, dans la figure ci-dessous.



(a) instant t_1 *(b) instant t_2*

(c) instant t_3 *(d) instant t_4*

(e) instant t_5 *(f) instant t_6*

Figure 4.29 : Résultats obtenus du déroulement du scénario 2 en utilisant le BOF + FC

Commentaires

Comme prévue, l'association de l'algorithme de classification rapide au filtre d'occupation bayésien a permis de réduire de manière très significative le temps de calcul. En effet, pour cette expérience, une boucle *prédiction-estimation* est exécutée dans un laps de temps égal à environs 0.5 secondes (avec la même configuration matérielle), contre 2 secondes dans le cas où le BOF est appliqué seul sans prétraitement.

Ce gain de temps est illustré par les résultats obtenus de cette dernière expérience (figures 4.29) qui montrent que l'état des cellules représentant le piéton en mouvement est estimé de façon beaucoup plus rapide, ce qui a permis un meilleur suivit du mouvement.

4.4 Conclusion

Au cours de ce chapitre, nous avons présenté et illustré la démarche d'implémentation d'un module DATMO basé sur le formalisme des grilles d'occupation, ce qui est une première pour la plateforme expérimentale « Robucar » du CDTA.

Nous avons montré que le choix des paramètres du modèle (grille d'occupation) est un compromis entre la précision de la carte et le temps de calcul. Lorsque le nombre de cellules est important, le temps nécessaire à sa création et à son actualisation en temps réel est aussi important. Un temps d'exécution relativement grand pourrait avoir des répercussions sur la qualité de la modélisation de l'environnement perçu et par conséquent sur le reste des éléments de la chaine PDA.

Nous avons relevé dans ce chapitre que les résultats obtenus de l'utilisation directe du filtre d'occupation bayésien ne sont acceptables que dans le cas des environnements faiblement dynamique où il est possible de suivre l'évolution des objets (cellules) dynamiques avec une faible cadence.

De plus, nous avons pu constater que l'utilisation d'un algorithme de prétraitement a permis de diviser par quatre le temps de calcul d'une itération du filtre et le ramener ainsi à 0.5 secondes/itération (en utilisant un calculateur équipé d'un microprocesseur Intel i5). De meilleurs résultats peuvent être obtenus en utilisant un calculateur plus puissant.

Par rapport à notre but initial, les résultats présentés dans ce chapitre sont satisfaisants. En effet, le Robucar initialement incapable de distinguer entre les objets statiques et les objets dynamiques est maintenant doté d'un moyen fiable de séparation et de perception complète de son environnement.

Conclusion générale et perceptives

Le domaine de la robotique mobile est devenu depuis quelques années un enjeu fondamental et stratégique pour de nombreux pays industriels ; les gouvernements et les constructeurs automobiles sont engagés sur ce créneau prometteur et visent à faire du véhicule complètement autonome bientôt une réalité. Aujourd'hui, plusieurs phases de conduite ont gagné en autonomie notamment avec la généralisation des systèmes d'assistance à la conduite dans les véhicules récents.

Ce mémoire s'inscrit dans le cadre du projet du Centre de Développement des Technologies Avancées (CDTA) visant à doter la plateforme robotique expérimentale « Robucar » de moyens lui permettant une navigation sûre autonome dans un milieu urbain. Notre apport consiste à développer un module de détection et de suivi des objets mobiles (DATMO) indispensable à la perception artificielle et à la modélisation de l'environnement extérieur entourant le robot.

La perception est une tâche complexe faisant intervenir plusieurs disciplines notamment l'instrumentation, l'informatique et la programmation, l'intelligence artificielle et une multitude d'autres domaines. Cette diversification est inévitable vu l'enjeu fondamental du problème de perception en robotique.

Dans ce mémoire, nous avons choisis d'implémenter le *grid-based* DATMO qui s'avère être l'approche la plus prometteuse et celle qui permet de construire une représentation de l'environnement en utilisant les données bruitées fournies par le capteur LMS. Cette approche utilise le filtre d'occupation bayésien pour l'estimation des paramètres dynamiques (vélocité) de façon probabiliste.

D'un point de vue purement pratique, la première étape dans la réalisation du module de DATMO est la construction d'une carte locale de l'environnement perçu par le robot et modélisé en utilisant le formalisme des grilles d'occupation. Nous avons vu que ce formalisme est très puissant et procure plusieurs avantages. Cependant, il faut choisir soigneusement les paramètres de cette grille (nombre de cellules et résolution) afin de vérifier les contraintes du fonctionnement en temps réel (fonctionnement entre 2 et 10Hz). Une fois la grille construite, la prochaine étape consiste à l'associer au filtre d'occupation bayésien afin d'estimer la distribution de probabilité sur la vélocité des cellules. Nous avons constaté que lorsque le filtre est appliqué seul, même sur une grille de taille moyenne, le temps de calcul d'une itération ne répond pas aux exigences du fonctionnement en temps réel, ce qui justifie l'utilisation d'un algorithme de prétraitement.

Ainsi, un algorithme de classification des cellules statiques et dynamiques est intercalé juste avant l'entrée du filtre d'occupation bayésien afin de soulager le traitement effectué par ce dernier. Les résultats obtenus sont acceptables étant donné que le module est capable de suivre le mouvement des cellules à 2Hz. Néanmoins, il est possible d'améliorer encore le temps de traitement en procédant à une optimisation des algorithmes développés d'une part et en utilisant un calculateur plus puissant d'autre part. C'est la perspective à court terme envisagée.

A moyen terme, ce travail peut être amélioré en utilisant un moteur d'inférence bayésienne pour l'implémentation des programmes. En effet, l'implémentation sous l'environnement ROS en utilisant le langage de programmation C++ s'est avéré être inappropriée à la manipulation des vecteurs et des matrices (2D et 3D) à grande dimension représentant les différentes distributions de probabilités. Une solution spécialement conçue au calcul probabiliste est donc fortement recommandée. Parmi les solutions existantes, on citera notamment ProBT®, un moteur d'inférence développé par l'équipe de recherche « e-Motion » de l'INRIA et « *Smart Sensors* », une solution d'analyse probabiliste des données générées par les capteurs, développée par l'entreprise française ProBayes®. L'utilisation de ces solutions permettra de gagner considérablement en temps de calcul étant donné qu'elles ont été spécialement conçues et optimisées pour la manipulation des distributions de probabilité.

Une implémentation *hardware* sur une carte dédiée peut également être envisagée. En effet, Coué explique dans [6] que le filtre d'occupation bayésien est particulièrement adapté à une implémentation sur des systèmes embarqués tels que les systèmes sur puce SoC (*System on Chip*).

Enfin, on conclue par souligner que l'utilisation d'un télémètre laser seul pour observer l'environnement est insuffisant surtout que le LMS 511 utilisé est un capteur balayant uniquement un plan 2D. Il est donc nécessaire d'augmenter les capacités de perception du Robucar et le choix des capteurs se doit se faire selon leur pertinence vis-à-vis de l'application considérée. En pratique, les caméras (stéréovision) semblent les plus adaptées et les plus utilisées. C'est la perspective à long terme envisagée.

Bibliographie

[1] N. Morette, «Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive,» Thèse de doctorat à l'Université d'Orléan, 2009.

[2] R. Moonzur, «Les voitures autonomes, bientôt sur nos routes ?,» *Article Techniques de l'ingénieur,* 05 octobre 2012.

[3] V. Berge, «Contribution à la gestion des incertitudes en fusion multicapteurs : Application à la perception du contexte de conduite,» Mémoire pour l'obtention de l'Habilitation à Diriger des Recherches, Université de Technologie de Compiègne, 2009.

[4] C. Cristophe, «Modèle bayésien pour l'analyse multimodale d'environnements dynamiques et encombrés : Application à l'assistance à la conduite en milieu urbain,» Thèse de doctorat. Institut Nationale Polytechnique de Grenoble, 2003.

[5] B. Qadeer, «Multisensor Data Fusion for Detection and Tracking of Moving Objects From a Dynamic Autonomous Vehicle,» Doctoral dissertation, Ph. D. dissertation. University of Grenoble, 2012.

[6] C. Coué, C. Pradalier, C. Laugier et T. Fraichard, «Bayesian Occupancy Filtering for Multitarget Tracking : an Automotive Application,» The International Journal of Robotics Research, SAGE Publications (UK and US), 2006, 25, 25 (1), pp.19{30}, 2006.

[7] A. Petrovskaya, P. Mathias, L. Oliveira, S. Luciano, R. Triebel, A. Makris, J.-D. Yoder, C. Laugier, U. Nunes et P. Bessiere, «Awareness of Road Scene Participants for Autonomous Driving,» *In Handbook of Intelligent Vehicles (pp. 1383-1432). Springer London,* p. 37, 2011.

[8] K. C. Fuerstenberg, D. Klaus, S. Eisenlauer et V. Willhoeft , «Multilayer laserscanner for robust object tracking and classification in urban,» *In 9th World Congress on Intelligent Transport Systems,* p. 8, 2002.

[9] A. Elfes, «Using Occupancy Grids for Mobile Robot Perception and Navigation,» Computer, 22(6), 46-57, 1989.

[10] L. John, M. Andrew et P. Fernando, «Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,» *University of Pennsylvania, Department of Computer & Information Science,* 2001.

[11] B. Douillard, D. Fox et F. Ramos, «Laser and Vision Based Outdoor Object Mapping,» *In Proceedings of Robotics: Science and Systems (RSS), Zurich, Switzerland,* 2008.

[12] Mataric, J. Ajo, F. Andrew et H. Maja, «A Laser-Based People Tracker,» *In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on (Vol. 3, pp. 3024-3029). IEEE,* 2002.

[13] D. Schulz, W. Burgard, D. Fox et A. B. Cremers, «People Tracking with Mobile Robots Using Sample-based Joint Probabilistic Data Association Filters,» *The International Journal of Robotics Research, vol. 22, no 2, p. 99-116,* 2003.

[14] B. Véronique, «Contribution a la Gestion des Incertitudes en Fusion Multicapteurs, Application à la perception du contexte de conduite,» *Université de Technologie de Compiègne, Mémoire pour l'obtention de l'habilitation à diriger des recherches,* 2009.

[15] C. Ingemar, «A review of statistical data association techniques for motion correspondence,» *International Journal of Computer Vision, Springer,* 1993.

[16] Y. Bar-Shalom, «Tracking in a Cluttered Environment With Probabilistic Data Association,» *Automatica, vol. 11, no 5, p. 451-460,* 1975.

[17] Y. Bar-Shalom et A. G. Jaffer, «Adaptive nonlinear filtering for tracking with measurements of uncertain origin,» *In Proceedings of the 1972 IEEE Conference on Decision and Control and 11th Symposium on Adaptive Processes (No. 11, pp. 243-247),* 1972.

[18] T. Fortmann, Bar-Shalom et Scheffe, «Multi-target tracking using joint probabilistic data association,» *In Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on (pp. 807-812),* 1980.

[19] Fortmann, Y. Bar-Shalom et M. Sheffe, «Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association,» *Oceanic Engineering, IEEE Journal of, 1983, vol. 8, no 3, p. 173-184,* 1983.

[20] B. R. Donald, «An algorithm for tracking multiple targets,» *Automatic Control, IEEE Transactions on, 24(6), 843-854,* 1979.

[21] Z. Khan, T. Balch et F. Dellaert, «A Rao-Blackwellized Particle Filter for EigenTracking,» *In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on (Vol. 2, pp. II-980). IEEE,* 2004.

[22] A. Petrovskaya et O. Khatib, «Global Localization of Objects via Touch,» *IEEE Transactions on, 27(3), 569-585,* 2011.

[23] C. Andrieu, N. d. Freitas, A. Doucet et M. I. Jordan, «An Introduction to MCMC for Machine Learning,» *Machine learning, vol. 50, no 1-2, p. 5-43,* 2003.

[24] C. Tay, K. Mekhnacha, C. Chen, M. Yguel et C. Laugier, «An Efficient Formulation of the Bayesian Occupation Filter for Target Tracking in Dynamic Environments,» *International Journal of Vehicle Autonomous Systems, 6(1-2), 155-171,* 2007.

[25] C. Tay, K. Mekhnacha, M. Yguel, C. Coué, C. Pradalier, C. Laugier, T. Fraichard et P. Bessiere, «The Bayesian Occupation Filter,» *In Probabilistic Reasoning and Decision Making in Sensory-Motor Systems (pp. 77-98). Springer Berlin Heidelberg,* 2008.

[26] K. Mekhnacha et D. Raulo, «Robust multi-target sensing/tracking in the Bayesian Occupancy Filter framework,» *In Journées Francophone sur les Réseaux Bayésiens,* 2008.

[27] E. T. Jaynes, «Probability Theory: The Logic of Science,» Cambridge University Press, 1995.

[28] O. Lebeltel, P. Bessiere, J. Diard et E. Mazer, «Programmation bayésienne des robots,» Revue d'Intelligence Artificielle, Lavoisier (Hermes Science Publications), 2004.

[29] H. Moravec et A. Elfes, «High resolution maps from wide angle sonar,» Proceedings. 1985 IEEE International Conference on (Vol. 2, pp. 116-121). IEEE, 1985.

[30] L. Matthies et A. Elfes, «Integration of sonar and stereo range data using a grid-based representation,» Proceedings of the 1988 IEEE International Conference on Robotics and Automation, 1988.

[31] S. Thrun, «Exploration and model building in mobile robot domains,» Proceedings of IEEE International Conference on Neural Networks, Seattle, Washington, USA, 1993.

[32] K. Konolige, «Improved occupancy grids for map building,» Autonomous Robots, no. 4, pp. 351–367, 1997.

[33] S. Thrun, «Learning occupancy grids with forward models,» Proceedings of the Conference on Intelligent Robots and Systems (IROS'2001), 2001.

[34] T. Collins, J. Collins et C. Ryan, «Occupancy Grid Mapping: An Empirical Evaluation,» Mediterranean Conference on Control and Automation, July 27 - 29 , 2007, 2007.

[35] T. Sebastian, «Probabilistic Robotic,» Intelligent Robotics and Autonomous Agents series, The MIT Press, 2005.

[36] D. Arbuckle, A. Howard et M. M., «Temporal occupancy grids : a method for classifying the spatio-temporal properties of the environment,» In Pro c. of the IEEE-RSJ Int. Conf. on Intel ligent Rob ots and Systems, Lausanne, (CH), 2002.

[37] E. Prassler, J. Scholz et A. Elfes, «Tracking Multiple Moving Objects for Real-Time Robot Navigation,» Journal of Autonomous Robot, 8(2), 105-116, 1999.

[38] A. H. Jazwinsky, «Stochastic Processes and Filtering Theory,» New York : Academic Press, 1970.

[39] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler et A. Ng, «ROS: an open-source Robot Operating System,» In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5), 2009.

[40] Q. Baig, M. Perrollaz, J. B. D. Nascimento et C. Laugier, «Fast classification of static and dynamic environment for Bayesian Occupancy Filter (BOF),» In Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on (pp. 656-661). IEEE, 2012.

[41] Q. Baig, M. Perrollaz, J. B. D. Nascimento et C. Laugier, «Using Fast Classification of Static and Dynamic Environment for Improving Bayesian Occupancy Filter (BOF) and Tracking,» In Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on (pp. 656-661). IEEE, 2012.

[42] B. Kluge, C. Kohler et E. Prassler, «Fast and Robust Tracking of Multiple Moving Objects with a Laser Range Finder,» IEEE, 2001.

[43] T.-D. Vu et O. Aycard, «Laser-based Detection and Tracking Moving Objects using Data-Driven Markov Chain Monte Carlo,» IEEE International Conference on Robotics and Automation, 2009.

[44] A. Staranowicz et G. L. Mariottini, «A Survey and Comparison of Commercial and Open-Source Robotic Simulator Software,» In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments (p. 56). ACM, 2009.

[45] A. Mendes, L. C. Bento et U. Nunes, «Multi-target Detection and Tracking with a Laserscanner,» In Intelligent Vehicles Symposium, 2004 IEEE (pp. 796-801). IEEE, 2004.

[46] C.-C. Wang, C. Thorpe et S. Thrun, «Online Simultaneous Localization And Maapping with Detection And Tracking of Moving Objects : Theory and Results from a Ground Vehicule in Crowded Urban Areas,» In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on (Vol. 1, pp. 842-849). IEEE, 2003.

[47] C.-C. Wang, C. Thrope et A. Suppe, «Ladar-Based Detection and Tracking of Moving Objects form a Ground Vehicule at High Speeds,» In Intelligent Vehicles Symposium, 2003. Proceedings. IEEE (pp. 416-421). IEEE., 2003.

[48] A. Censi, D. Calisi, A. D. Luca et G. Oriolo, «A Bayesian framework for optimal motion planning with uncertainty,» In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (pp. 1798-1805). IEEE, 2008.

[49] P. Kondaxakis, S. Kasderidis et P. Trahanias, «Tracking Multiple Targets from a Mobile Robot Platform using a Laser Range Scanner,» Institute of Computer Science Foundation for Research and Technology, 2008.