Méthodologie d'évaluation du degré d'autonomie d'un robot mobile terrestre

Alexandre Lampe

Thèse

présentée

pour obtenir

LE TITRE DE DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

École doctorale : Système

Spécialité : Informatique

par

Alexandre Lampe

Méthodologie d'évaluation du degré d'autonomie d'un robot mobile terrestre

---

Soutenue le 14 décembre 2006 devant le jury composé de :

Rachid Alami	**LAAS-CNRS**	Président du jury
Raja Chatila	**LAAS-CNRS**	Directeur de thèse
Dominique Luzeaux	**DGA**	Rapporteur
René Zapata	**LIRMM**	Rapporteur
Catherine Tessier	**ONERA-CERT**	Examinateur
Aurélien Godin	**DGA**	Examinateur

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

Remerciements

Merci, merci, merci ...

... à tous ceux qui de près ou de loin ont contribué à ce travail. Et ils sont nombreux, même si parfois certains n'en ont pas conscience. Aussi je m'excuse par avance auprès de ceux qui ne seront pas nommés ici, je pense également à eux.

Pour commencer je voudrais remercier tous les membres de mon jury d'avoir accepter d'évaluer mes travaux. Merci à Rachid Alami pour avoir présidé ce jury et pour m'avoir accueilli dans le groupe RIS durant ces trois années. Ta sympathie constante et ton humour donnent du baume au cœur à chaque rencontre. Merci à Dominique Luzeaux et à René Zapata pour avoir été mes rapporteurs et m'avoir donné des conseils utiles pour améliorer ce manuscrit. Et un grand merci à mon directeur de thèse Raja Chatila. Tu a accepté de m'encadrer et de me faire confiance dans un domaine qui n'était pas le mien au commencement de cette thèse. Tu m'a supporté dans tous les sens du terme et je n'oublierais pas tes qualités humaines.

J'ai eu l'occasion de collaborer avec quelques thésards pendant ces années en particulier Sylvain Joyeux et Benjamin Lussier qui ont tout comme moi simulé (des robots bien sur). Jason Held avec qui j'ai pu apprécier le décalage horaire entre Toulouse et Sydney. Ma présentation de soutenance a bénéficiée des conseils avisés de Guillaume et de Nico.

Trois années de thèse c'est aussi des moments de pauses café et de détentes, afin de s'oxygéner un peu. Merci à mes camarades Seb, Blou, Nico, Thomas, Guillaume, Cédric, Max, Tony. Sans oublier les voyageurs du monde expatriés ou aventurier : Fred et Léo. Amitié pokeriste à tous les joueurs qui se reconnaîtront.

Une spéciale dédicace pour mon grand ami little-B qui est venu me soutenir avec le p'tit Ilan. Grâce à vous tous les bibelots sont en hauteur maintenant. Big Up également à la maman et à la petite dernière ainsi qu'à

tous les amis parisiens : Sylvain, Agnès, GrosBat, Alex et tous les membres du posse d'easy style.

Maintenant une pensée pour toute ma famille qui me soutiens depuis toujours et spécialement pour mes parents. Bises affectueuses à Max, Manu, Nico et tous les autres.

Enfin un grand merci à mon petit monde : ma chérie et mes deux matous. Alice dans peu de temps tu deviendra mon épouse. Merci de m'avoir supporté au quotidien pendant trois ans, même mes mauvais jours. Petit clin d'œil à tao et zorro qui m'ont fait comprendre en venant sur mon clavier ou devant mon écran qu'il était temps de faire une pause et de m'occuper d'eux.

Table des matières

Table des figures

Liste des tableaux

Abréviations

ALFUS Autonomy Levels For Unmanned Systems
BN Bayesian Network - Réseau Bayésien
DBN Dynamic Bayesian Network - Réseau Bayésien Dynamique
DAG Directed Acyclic Graph - Graphe Direct Acyclique
DARPA Defense Advanced Research Projects Agency
FCS Future Combat System : programme militaire américain
HRI Human Robot Interface - Human Robot Interaction
LAAS Laboratoire Analyse et d'Architecture des Systèmes
LADAR Laser Radar
MER Mars Exploration Rover
NIST National Institute of Standards and Technology
RSTA Reconnaissance, Surveillance and Target Acquisition
SMART Spacecraft Mission Assessment and Re-planning Tool
TRL Technology Readiness Level
UAV Unmanned Aerial Vehicle
VNH ou UMS Véhicule Non Habité - Unmanned System

Chapitre I

Introduction

I.1 Généralités

La robotique est une science à la croisée de plusieurs disciplines scientifiques et techniques (mécanique, électronique, informatique, ...). Ses applications sont actuellement principalement connues dans l'industrie. Ici nous nous intéresserons à la robotique mobile autonome, qui se différencie de la robotique industrielle par l'ajout de fonctions perceptuelles et décisionnelles permettant l'évolution du robot dans un environnement complexe.

Les missions martiennes de la NASA ont permis au grand public de découvrir des robots autonomes. Le robot *Sojourner* de la mission *Mars Pathfinder* est le premier rover semi-autonome envoyé dans l'espace. Il se posa sur Mars le 4 juillet 1997 et au cours de ses 83 jours de fonctionnement, il envoya 550 photos vers la Terre ainsi que des analyses de roches. Le robot dirigé depuis la Terre embarquait une fonction d'évitement d'obstacles autonome, sa vitesse de déplacement étant de 1cm par seconde.

La mission *Mars Exploration Rover* (MER), qui a pour objectif la recherche de traces de vie, est la plus ambitieuse à ce jour. Deux rovers *Spirit*

et *Opportunity*, de dimensions imposantes (1,5 m de haut, 2,3 m de large et 1,6 m de long pour 185 kg, en comparaison des 10,6 kg de Sojourner), se sont posés sur Mars les 3 et 24 janvier 2004. Les rovers sont dotés d'un planificateur de trajectoires et d'un algorithme d'évitement d'obstacles.

Si dans ces deux missions les robots ne planifient pas leurs actions (elles sont envoyées depuis la Terre), elles montrent en revanche le gain d'autonomie des robots d'exploration planétaire.

Dans le domaine civil, la robotique autonome commence à faire son apparition. Passons les robots commerciaux (robot Sony *AIBO*, robots aspirateurs ...) qui n'intègrent peu ou pas de capacités de décision, et regardons des applications souvent encore expérimentales. Le robot *Rackham*[1] du LAAS-CNRS a évolué pendant plusieurs semaines à la Cité de l'Espace à Toulouse en présence du public. Ce robot, doté d'un très grand nombre de fonctionnalités (capacités de décision, de perception, de locomotion, d'interaction avec le public ...), a servi de guide interactif pour les visiteurs.

De même des projets actuels de recherche concernant la robotique domestique, montrent l'intérêt croissant pour la robotique autonome. Ainsi le projet européen *COGNIRON*[2] s'intéresse au robot cognitif compagnon, l'objectif ultime étant d'arriver à concevoir des robots capables d'assister l'homme dans la vie de tous les jours. Ceci requiert des capacités cognitives avancées et une capacité à interagir en étroite collaboration avec l'homme.

Les recherches actuelles en robotique autonome visent toutes à donner plus d'autonomie au robot, que ce soit en développant de nouvelles capacités ou en rendant plus robustes celles existantes.

I.2 La notion d'autonomie

Tout d'abord, différencions les systèmes dit *automatiques* des systèmes dit *autonomes*. Dans [Clough, 2002], les systèmes automatiques, sont ceux qui ont été pré-programmés, qui n'ont pas le choix de leurs actions, en opposition aux systèmes autonomes qui peuvent décider. On voit très bien que la frontière entre les deux n'est pas franche. On pourrait être tenté de dire qu'un système déterministe est automatique, et que les autres sont autonomes. Mais si l'on considère par exemple, un robot purement réactif naviguant par la méthode des champs de potentiels, il peut être considéré comme déterministe (si l'on considère que les incertitudes attachées aux

[1] http ://www.laas.fr/∼sara/laasko/
[2] http ://www.cogniron.org/

capteurs ne sont pas trop grandes, sachant l'environnement, les points de départ et d'arrivée, on peut prédire son chemin), pourtant nous aurons tendance à le classer dans les systèmes autonomes, car il est capable d'agir sur sa trajectoire en fonction de l'environnement. *Nous considérerons qu'un système est autonome s'il est capable d'adapter son comportement à l'environnement*, tandis que les systèmes automatiques sont ceux qui évoluent dans des environnements complètement prévisibles. La distinction se fera sur la variabilité des conditions extérieures, le système automatique agissant dans un environnement assez maîtrisé, son comportement ayant été conçu en fonction de cela (on pense tout de suite aux tâches répétitives de l'industrie manufacturière, gourmande en automatismes).

Cette distinction faite, nous n'avons toujours pas défini l'autonomie en robotique. Concrètement il est très difficile de donner une définition à l'autonomie. La principale raison étant qu'il s'agit d'une notion très subjective et relative. Subjective, dans le sens où deux utilisateurs qui évaluent le même système, ne le verront pas de la même façon, selon les critères qu'ils privilégieront. Et relative car on ne peut évaluer et comparer l'autonomie d'un système sur deux tâches différentes. A cela s'ajoute une confusion dans la communauté IA qui est de comparer "l'intelligence"[3] et l'autonomie des systèmes. Si l'on qualifie souvent les systèmes autonomes comme étant des systèmes intelligents, la définition de l'intelligence étant elle même très vague, cela ne rend pas le terme *autonomie* plus clair. L'intelligence est une notion que l'on a déjà bien du mal à définir pour l'homme (en particulier, les tests de QI sont souvent critiqués). De plus dans notre cas, l'intelligence se rapporte bien souvent aux composants décisionnels d'un robot et à ses capacités de calcul. Or comme nous l'avons précisé, un robot purement réactif ne peut être considéré comme non autonome pour la seule raison qu'il possède une "intelligence" limitée.

Le grand nombre de définitions de l'autonomie dans la littérature, illustre bien le caractère subjectif de cette notion. Ici nous ne retiendrons que la définition donnée par le *National Institute of Standards and Technology* (NIST).

Autonomie : selon [Huang et al., 2004a]

1. Condition ou qualité à être auto-gouverné[4]

[3] voir à ce sujet la première série du workshop PerMIS : http ://www.isd.mel.nist.gov/research_areas/research_engineering/-PerMIS_Workshop/index.htm

[4] d'après : Houghton Mifflin Company, The American Heritage Dictionary.

> 2. Capacité propre d'un système sans équipage, à capter, percevoir, analyser, communiquer, planifier, prendre des décisions et agir afin d'atteindre les buts qui lui ont été assignés par un opérateur humain à l'aide d'une interface homme/-machine dédiée. L'autonomie est échelonnée sur plusieurs niveaux, qui sont caractérisés par des facteurs incluant la complexité de la mission, les difficultés environnementales et le niveau d'interaction homme/robot nécessaire à l'accomplissement de la mission.

Cette définition prend en compte toutes les capacités du robot, ce qui est totalement justifié, car toutes les fonctions d'un robot contribuent à son autonomie. On voit par ailleurs que l'autonomie est relative à l'objectif de la mission. En effet, pour une mission de navigation en milieu partiellement connu, on peut fixer dans un cas comme objectif d'aller d'un point à un autre le plus rapidement possible, et dans un autre cas de construire une carte de l'environnement de la manière la plus fidèle qui soit. Ces deux cas ne peuvent être comparés entre eux, car un robot peut très bien être performant sur un des scénarios et mauvais dans l'autre. La définition met également l'accent sur des paramètres qui influent sur l'autonomie : la complexité de la mission, l'environnement et l'interaction homme/robot. Il est évident que la complexité de la mission joue sur l'autonomie. Prenons par exemple notre mission de navigation précédente avec pour objectif à la fois d'aller le plus vite possible tout en cartographiant l'environnement, cette mission est plus complexe que les précédentes, et il semble raisonnable de penser que l'autonomie sera moins bonne (surtout si le robot n'est pas très bon dans une des deux missions simples). Les difficultés environnementales jouent elles aussi sur l'autonomie. Dans la mission de navigation, selon que l'environnement est un terrain plat découvert ou un environnement d'intérieur compliqué, l'autonomie ne sera pas la même. Enfin, [Goodrich et al., 2001, Crandall and Goodrich, 2003] montrent bien l'influence de l'homme sur les performances d'un robot.

Cette définition pose une bonne base pour l'étude de l'autonomie, en revanche elle ne précise pas comment procéder à l'évaluation. Plus précisément, il nous faut définir des outils pour mesurer quantitativement l'autonomie, ainsi que les paramètres qui l'influencent.

I.3 Ce qui se cache dans les pages suivantes

Le chapitre II donne un aperçu des méthodes et techniques utilisées en robotique, en particulier celles liées à l'évaluation des robots. Après un bref rappel sur les fonctionnalités communes des robots autonomes, nous discuterons des systèmes de classification par niveaux d'autonomie. Puis nous nous intéresserons aux benchmarks qui permettent soit de tester une partie du système, soit dans le cas des compétitions de robotique de tester l'ensemble du robot. Enfin nous verrons comment l'interaction entre un robot et une autre entité peut affecter l'autonomie.

La suite du manuscrit est consacrée à notre contribution et explique notre méthode pour l'évaluation de l'autonomie. Celle ci est basée sur la décomposition d'une mission robotique en termes de contraintes et paramètres mesurables (tels que des indicateurs de performances, des distances aux obstacles, la position du robot ...). Les contraintes sont soit imposées par les exigences de l'utilisateur, soit imposées par des conditions externes. L'évaluation se fait grâce aux paramètres mesurés pendant la mission du robot. Parmi ces paramètres, les performances du robot sont affectées par les conditions externes, telles que les difficultés environnementales et les connaissances sur l'environnement. Les difficultés environnementales et les connaissances sur l'environnement sont déduites de paramètres mesurés (par exemple les distances aux obstacles). Si le lien entre autonomie et complexité de l'environnement est un fait admis et démontré de manière empirique, le lien entre performances et informations est une nouveauté introduite par ce travail.

D'après ce cadre formel et avec l'identification des facteurs qui influencent l'autonomie, nous avons développé des métriques pour chaque composante. Nous utilisons des métriques pour rendre compte du niveau de performance du robot, de la complexité de l'environnement et pour quantifier l'information que le robot possède. Le chapitre III décrit la méthode et propose des métriques adaptées à une mission de navigation.

Afin de valider la méthode et les métriques, nous avons adapté un simulateur qui nous a servi de moyen d'expérimentation. Ce simulateur qui a nécessité plusieurs mois de développement, permet d'utiliser les mêmes algorithmes en simulation que ceux qui fonctionnent sur les robots du laboratoire. Le chapitre IV décrit brièvement ce simulateur ainsi que les scénarios de mission qui ont servi pour les expérimentations. Si la description du simulateur est succincte, c'est que ce document se focalise sur la méthodologie pour l'évaluation de l'autonomie. La simulation est le moyen que nous avons choisi pour valider notre approche, mais n'est pas le sujet qui nous concerne.

Les simulations fournissent un grand nombre de données qu'il faut traiter, mettre en forme et analyser. Des scripts ont été développés pour le traitement et la mise en forme. Pour l'analyse, deux méthodes ont été utilisées. La première, lorsqu'il s'agissait de lier entre eux deux ou trois paramètres, utilise des approximations linéaires selon le critère des moindres carrés. La deuxième méthode, qui permet de comprendre les liens entre un grand nombre de paramètres, est le fruit d'une collaboration avec l'université de Sydney. Le chapitre V décrit ces méthodes et en montre l'utilisation sur les données collectées lors des séries de simulations réalisées.

Pour finir, le chapitre VI clôt ce travail, présente nos conclusions, des voies d'amélioration et d'extension.

Chapitre II

Etat de l'art

L'évaluation de robots autonomes est une étape importante dans la phase de conception de ces systèmes. Cependant la littérature est assez pauvre sur ce sujet. Dans ce chapitre nous donnons, dans un premier temps, les caractéristiques que l'on rencontre sur les robots autonomes actuels. Puis nous évoquerons les travaux récents sur la classification des robots par niveaux d'autonomie. Ensuite nous nous intéresserons aux benchmarks qui sont très utiles lorsque l'on cherche à comparer plusieurs systèmes selon des critères communs. Après quelques exemples de benchmarks, nous verrons des exemples de tests de robots en conditions réelles. Enfin nous regarderons comment varie l'autonomie d'un robot lorsqu'il interagit avec une autre entité, qu'elle soit humaine ou qu'il s'agisse d'un autre robot.

II.1 Les capacités des robots

Classiquement on distingue deux types d'autonomie en robotique : l'autonomie opérationnelle et l'autonomie décisionnelle. Cette distinction vient

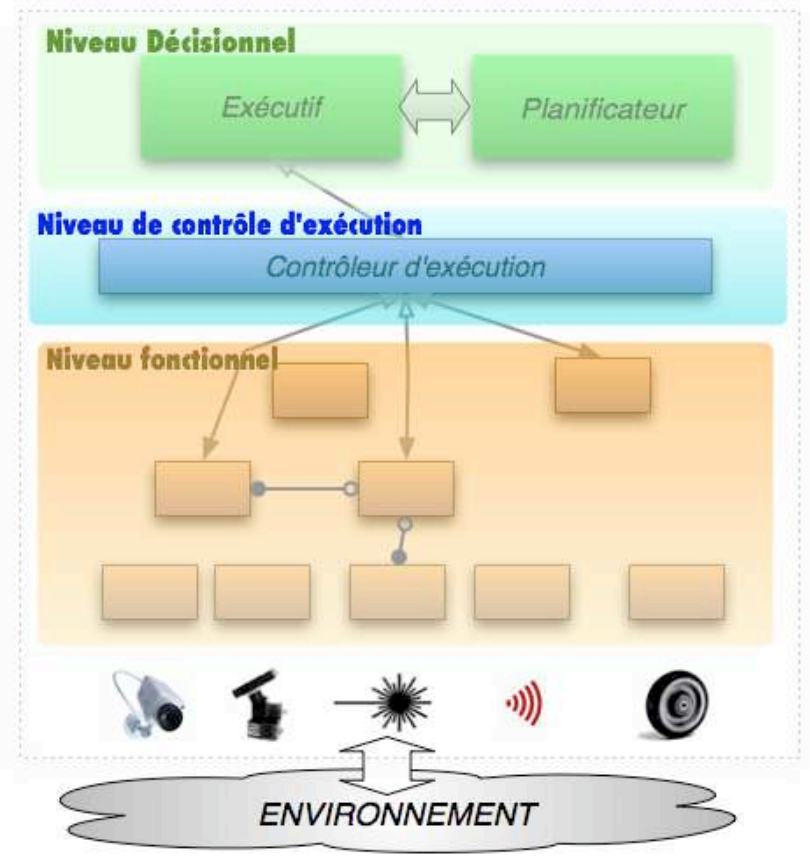


FIG. II.1 – Architecture générique trois niveaux du LAAS.

du découpage que l'on trouve dans les architectures de contrôle sur les robots. Par exemple la figure II.1 représente l'architecture développée, implémentée et utilisée au LAAS. On retrouve les capacités opérationnelles vers le bas de l'architecture, jusqu'au niveau fonctionnel. Les capacités décisionnelles se trouvent tout en haut. Par la suite nous prendrons cette architecture comme base pour illustrer la présentation des capacités des robots. Même si toutes les architectures robotique ne sont pas de type hiérarchique, les architectures les plus évoluées embarquent des composants fonctionnels et des organes décisionnels.

II.1.1 Capacités fonctionnelles

Les capacités fonctionnelles sont les briques de base qui permettent de faire fonctionner les robots. Elles servent de liens entre les composants physiques du robot (capteurs et actionneurs) et la partie qui sert à la supervision

et à la décision. Dans l'architecture *LAAS* les modules fonctionnels ont par exemple la charge de :

– convertir des informations de haut niveau en des commandes pour le matériel (consignes de position ou de vitesse converties en commandes moteur)

– convertir des données capteur en des informations exploitables (impulsions des capteurs odométriques convertis en des déplacements relatifs)

– fournir des services utilisables par d'autres modules fonctionnels ou par les organes décisionnels (planification d'une trajectoire)

– effectuer des traitements sur des données issues d'autres modules fonctionnels (localisation d'après une carte de segments grâce aux données issues du module qui gère le télémètre laser plan)

Chacune de ces capacités (ou modules) fournit un service spécifique qui traduit un besoin précis pour le robot (contrôle de la locomotion, évitement d'obstacles, planification de trajectoire, ...). Plus on dote un robot de capacités fonctionnelles différentes et variées et plus on augmente son potentiel et son spectre d'action. Par exemple un robot qui est capable de planifier des trajectoires est capable d'agir dans des environnements plus complexes que le robot qui ne le peut pas. De plus la redondance de capacités fonctionnelles peut être un avantage dans certains cas (par exemple avoir plusieurs moyens différents pour se localiser donne plus de chances de le faire correctement).

Dans l'architecture LAAS, l'outil GenoM a été spécifiquement développé afin de concevoir et développer des modules fonctionnels.

II.1.2 Capacités décisionnelles

Les capacités fonctionnelles doivent être coordonnées et organisées afin de garantir un bon fonctionnement du robot. On peut le faire de manière préétablie en figeant l'organisation des fonctions (quel module est client et/ou fournisseur d'un autre module ?) et en donnant des règles fixes pour gérer les différentes situations que l'on peut rencontrer. Mais pour avoir un robot plus autonome, on le dote de capacités décisionnelles plus élaborées qui lui permettent de faire face à un plus grand nombre de situations et qui permettent une organisation dynamique des modules fonctionnels selon la tâche.

Dans l'architecture LAAS, on distingue plusieurs capacités décisionnelles. Le niveau décisionnel qui est le plus haut dans la hiérarchie comporte un planificateur et un exécutif procédural. Le niveau de contrôle d'exécution sert de tampon entre les deux autres niveaux et peut être considéré comme

un organe décisionnel.

Le planificateur (nommé IxTeT) planifie l'ordre des actions qu'il faut exécuter afin de remplir une mission donnée. Il s'agit d'un planificateur symbolique qui prend en compte les contraintes temporelles.

L'exécutif procédural décompose les actions générées par le planificateur en des séquences de requêtes compréhensibles par le niveau fonctionnel. Par exemple pour une action de déplacement il peut demander une localisation, ensuite une planification de trajectoire, puis il demandera au module chargé du déplacement de s'initialiser et de suivre la trajectoire planifiée.

Le niveau de contrôle d'exécution a un rôle particulier, il s'agit surtout d'un organe pour la sûreté de fonctionnement. Il est chargé d'analyser la cohérence des ordres produits par l'exécutif procédural avec des règles préétablies et de vérifier le bon déroulement d'une séquence d'actions élémentaires.

II.2 Les niveaux d'autonomie

Parmi les méthodes pour l'évaluation des systèmes robotiques, la classification par des niveaux d'autonomie semble être la méthodologie la plus pratique pour l'utilisateur final. Le principe est simple : un tableau à niveaux définit les capacités requises pour un système, afin de correspondre à un niveau donné. Cependant nous verrons que cette méthode est difficile à mettre en œuvre.

II.2.1 Le système de classification ALFUS

Le système de classification ALFUS (Autonomy Levels For Unmanned Systems) est issu d'un groupe de travail créé en juillet 2003. Ce groupe de travail fait partie du NIST (National Institute of Standards and Technology). L'objectif d'ALFUS est de donner des métriques pour classer les systèmes par niveaux d'autonomie. Un premier travail important a été de définir la terminologie autour des systèmes autonomes [Huang, 2004].

De ce groupe de travail est né un système de classification générique (fig. II.2). Ce système se décline ensuite de manière plus spécifique selon la mission, de façon à être exploitable pour l'utilisateur final.

Les termes et définitions [Huang, 2004] servent de base à la description des systèmes non habités. Ceci est indispensable afin que chacun utilise un vocabulaire commun bien délimité. Les termes génériques comprennent entre autres les définitions : autonomie, environnement, fusion, interface homme robot (HRI), planification de mission, mode d'opération, perception,

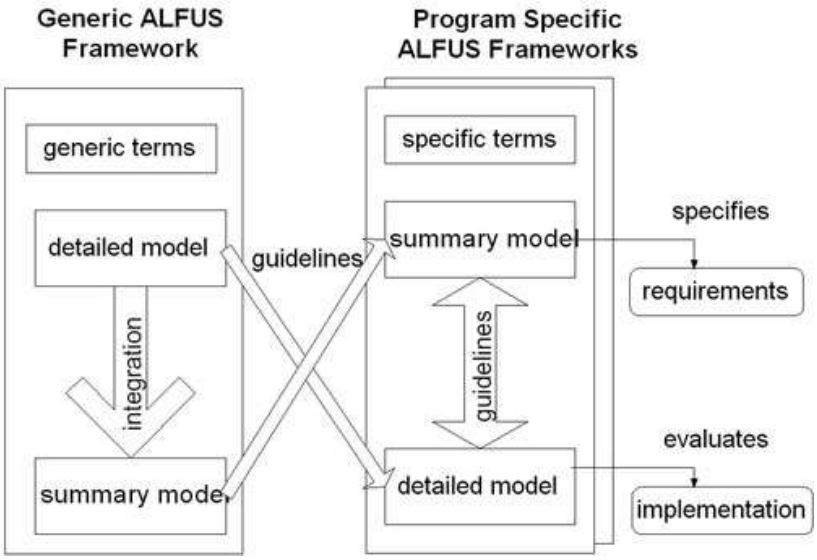


FIG. II.2 – Système de classification générique ALFUS

etc.

Detailed Model for Autonomy Levels : Ensemble de métriques exhaustives qui représentent les multiples aspects d'intérêt. Ceci inclut la complexité de la mission, les difficultés environnementales et le niveau d'interaction homme-robot. Cette combinaison de métriques donne une indication du niveau d'autonomie d'un Véhicule Non Habité (VNH) (fig. II.3).

Summary Model for Autonomy Levels : Ensemble d'échelles linéaires, dont les valeurs sont comprises entre 0 et 10 (ou 1 à 10), utilisées pour indiquer le niveau d'autonomie. Ce modèle est la déclinaison du modèle détaillé du VNH.

[Huang et al., 2004b] et [Huang et al., 2005] décrivent comment sont détaillés et calculés les modèles ci-dessus. Le modèle de synthèse (Summary Model for Autonomy) est obtenu grâce à une somme pondérée des différentes métriques du modèle détaillé. Le modèle détaillé se décompose selon les trois axes de la figure II.3, chaque axe comporte plusieurs métriques pour le décrire.

L'axe **"complexité de la mission"** représente les missions que le robot est capable d'accomplir. Plus le score d'un robot sur cet axe est élevé et plus il est capable d'accomplir des missions difficiles. Pour cette mesure sont pris en compte les facteurs suivants :

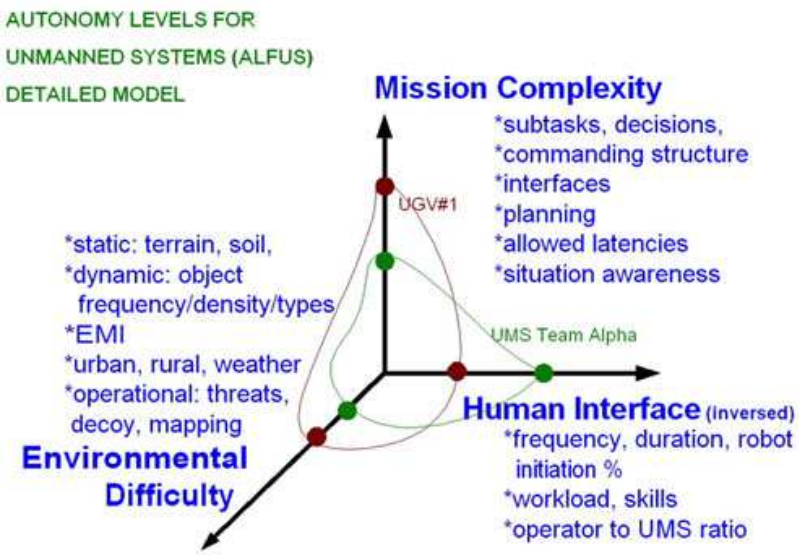


FIG. II.3 – ALFUS - Modèle détaillé selon 3 axes

– La capacité tactique se mesure d'après la structure et la composition des tâches invoquées. Cette mesure est tirée de la topologie de l'architecture hiérarchique 4D/RCS (voir [Albus, 2002a]). Elle prend en compte le nombre de sous tâches invoquées, le nombre de superviseurs et le nombre de points de décision.

– La coordination et la collaboration : des tâches compliquées requièrent plus de coordination et de collaboration entre les composants du robot ou les sous systèmes. Cette métrique inclut le nombre d'entités participantes et le nombre d'interfaces en service (type de données, nombre de canaux, ...).

– Le niveau de performance : si le robot est capable d'atteindre ses buts avec un haut niveau de performance, alors il possède une grande autonomie. Cette métrique se base sur la capacité du robot à planifier ses actions et sur la souplesse quant aux erreurs et aux contraintes temporelles.

– Le traitement perceptif et la modélisation de l'environnement : le niveau de perception requis et la complexité de modélisation de l'environnement nécessaires à l'accomplissement d'une mission sont des indicateurs de la complexité de la mission.

L'axe **"difficultés environnementales"** mesure la complexité de l'environnement que le robot est capable de gérer. Cet axe ne dispose pas de métriques exploitables de manière mathématique. Des considérations telles

que "environnement statique", "environnement dynamique" ou d'autre qualificatifs sont utilisées pour qualifier la difficulté environnementale.

L'axe **"niveau HRI"** (niveau d'interaction homme/robot) mesure quel est le niveau d'interaction nécessaire afin de remplir les objectifs de la mission. Cette mesure se base sur les interventions humaines non prévues (fréquence, durée et nombre), sur la charge opérateur (prévue ou non), sur le niveau de compétence requis pour l'opérateur (s'agit il d'un ingénieur ou d'une personne non qualifiée ?) et sur le ratio opérateurs/VNHs (combien de VNH un opérateur seul peut il contrôler en même temps ?).

A notre connaissance cette méthodologie n'a pas été utilisée en pratique. Le système ALFUS est un premier pas vers une définition de l'autonomie et vers la décomposition de l'autonomie en termes pratiques. Néanmoins les métriques de cette méthode mélangent la mesure de paramètres externes au robot avec des paramètres qui dépendent du robot. Ceci pose un problème par exemple pour la mesure de la complexité d'une mission qui devrait être indépendante du robot. Que le robot soit en difficulté ou non dans une mission particulière n'affecte en rien la nature de la mission. De plus pour la même mission un premier robot peut être à son aise tout au long et un autre se voir en difficulté. Cette mission sera facile pour le premier robot et complexe pour le second : lequel des deux robots "a raison" ?

Un autre point à noter est que certaines métriques font directement référence à la conception même du robot, en s'appuyant sur l'architecture 4D/RCS. Les métriques développées ne sont donc pas génériques et ne seront probablement pas adaptées à tous les robots que l'on peut chercher à évaluer.

II.2.2 FCS

Le programme militaire américain FCS (Future Combat Systems) s'appuie entre autres sur l'utilisation de robots autonomes. Ce programme est le plus avancé à ce jour en ce qui concerne l'utilisation de la robotique à des fins militaires. Dans [Kamsickas and Ward, 2003] une classification a été définie pour hiérarchiser les robots selon leurs niveaux d'autonomie (tab. II.1).

Ainsi 10 niveaux d'autonomie sont définis, allant de la simple téléopération filaire aux systèmes autonomes collaboratifs. Ce tableau décrit chaque niveau en termes de capacités que le robot possède.

Niveau	Description du niveau	Capacité de perception	Capacité de décision	Capacité générale	Exemple
1	Téléopération filaire	Aucune	Aucune	Opérateur commandant vitesse, direction et freins	Téléopération basique et filaire
2	Téléopération sans fil	+capteurs de conduite	Aucune	Opérateur commandant vitesse, direction et freins	Téléopération basique et sans fil
3	Téléopération avancée	+information sur état du véhicule	+compte-rendu de l'état du véhicule	Opérateur commandant vitesse, direction et freins et ayant connaissance de l'état du véhicule	Téléopération basique, sans fil, avec connaissance de l'état du véhicule
4	Semi-autonomie supervisée et planification d'itinéraire externe	+modèle du monde +capteurs simples	+suivi d'un itinéraire défini de manière dense par des points de passages	Suivi d'itinéraire, détection d'obstacles et de danger par l'opérateur	Déplacement par points de passage, téléopération assistée
5	Semi-autonomie supervisée et planification locale d'itinéraire	+détection d'obstacles et de danger	+(re)planification locale +corrélation des perceptions avec modèle du monde	Suivi de véhicule robuste avec l'aide d'un opérateur pour obstacles et danger	Suivi d'itinéraire en téléopéré, convoi de robot
6	Semi-autonomie non supervisée avec gestion des dangers	+perception locales corrélées avec modèle du monde	+planification d'itinéraire basée sur estimation du danger	Navigation semi-autonome basique sur terrain ouvert et roulant avec intervention d'un opérateur	Navigation basique sur terrain ouvert et roulant

Suite page suivante ...

... suite de la page précédente

Niveau	Description du niveau	Capacité de perception	Capacité de décision	Capacité générale	Exemple
7	Missions basiques en autonome	Utilisation des perceptions locales et du modèle du monde pour évitement et négociation dangers	+planification et gestion de terrains et d'objets complexes	navigation sur terrain ouvert avec négociation obstacles, vitesse limitée, aide ponctuelle opérateur	Navigation robuste sur terrain ouvert
8	Fusion autonome des données capteurs et des informations	+fusion des capteurs du véhicule	+planification robuste et négociation de terrains complexes, prise en compte environnement, dangers et objets	Navigation sur terrain complexe, vitesse réduite, aide ponctuelle de l'opérateur	Semi-autonomie basique sur terrain complexe
9	Missions autonomes en coopération	+fusion de données issues d'autres VNH (ex. drones)	+prise de décision complexes basées sur données obtenues par coopération	Navigation sur terrain complexe avec pleines capacités de vitesse et de mobilité, navigation coordonnée multi VNH avec supervision de haut niveau	Navigation robuste sur terrain complexe

Suite page suivante ...

... suite de la page précédente					
Niveau	**Description du niveau**	**Capacité de perception**	**Capacité de décision**	**Capacité générale**	**Exemple**
10	Missions autonomes en collaboration	+fusion de données de navigation et d'informations de RSTA issues des VNH	+décision, planif. et exécution en collaboration +comportements tactiques basés sur la connaissance de la situation	Réalisation de missions par planification et exécution en collaboration et sans supervision humaine	Réalisation de missions en autonome avec des buts individuels et différés

TAB. II.1 – Niveaux d'autonomie définis dans le Futur Combat System.

II.2.3 Difficultés - Limitations

On constate que la préconisation du système ALFUS concernant la classification par type de mission est utilisée dans le tableau FCS. En effet ce tableau concerne les missions de navigation pour les systèmes robotiques. Il faudra donc pour chaque catégorie de mission faire un tableau semblable. Par exemple pour une tâche de reconnaissance et de manipulation d'objet, le tableau proposé n'est pas vraiment pertinent.

La mesure de l'autonomie par des niveaux prédéfinis est certes ce que l'utilisateur final peut attendre en termes de simplicité pratique, mais dire qu'un robot atteint le niveau i plutôt que le niveau j n'est pas chose facile à trancher. Toute la difficulté est là : *selon quels critères peut-on décider du niveau d'autonomie réel d'un robot ?*

Chaque ligne dans le tableau II.1 donne, selon le niveau, les capacités que le robot doit posséder. Or nous savons que ce n'est pas parce qu'une fonction est implémentée, qu'elle fonctionne selon nos attentes. Il n'y a pas de garantie quant au domaine de fonctionnement du système. Plus gênant encore, des robots dont le niveau d'autonomie est faible peuvent être plus performants que d'autres de niveau plus élevé. Par exemple sur une mission de navigation, un robot purement réactif peut être plus rapide qu'un robot qui planifie ses trajectoires.

Il faut donc bien distinguer *niveau d'autonomie* et *performances*. Au final, sur une mission donnée, ce qui nous intéresse ce sont les performances du système (rapidité, robustesse, qualité des données collectées ...), et à la

limite peu importent les fonctionnalités mises en œuvre.

II.3 Benchmarking

Un *benchmark* est un banc d'essai permettant de mesurer les performances d'un système pour le comparer à d'autres[1].

II.3.1 Sur des algorithmes

Principe et intérêt

La création de benchmark en informatique n'est pas chose nouvelle. On en retrouve dans l'évaluation de composants (processeurs, mémoire, ...) ou dans l'évaluation d'algorithmes particuliers (par exemple en vision). L'idée de base est de tester les performances d'un système lorsqu'on lui applique un jeu de paramètres donné en entrée.

Dans le cas de test matériel, on fait effectuer un certain nombre d'opérations spécifiques et on regarde les performances. Pour un processeur, on pourra faire des calculs en nombres flottants et comparer le temps de calcul moyen. On peut également regarder les temps de lecture et d'écriture sur un disque dur ou dans de la mémoire volatile.

Dans le cas d'une évaluation d'algorithmes, les benchmarks consistent à définir des paramètres d'entrée génériques qui serviront de base de comparaison. Pour être comparable, il faut préciser quelle est la puissance de calcul qui a été utilisée pour faire le test. Car bien souvent les critères de performances comprennent en outre des critères de qualité, mais aussi le temps de calcul.

L'intérêt du benchmark est qu'il permet de comparer des systèmes différents selon des critères communs. L'inconvénient majeur est qu'il faut que les systèmes à tester puissent comprendre le formalisme utilisé en entrée (ceci est surtout vrai pour les algorithmes).

Exemple en vision

La banque d'images et de segmentation de Berkeley [Martin et al., 2001] est une base de données pour effectuer des comparatifs sur les algorithmes de segmentation d'images. La partie publique de cette base d'images comprend 300 images dont 200 pour l'apprentissage des algorithmes et 100 pour les tests (la moitié des images est en niveaux de gris, l'autre moitié en couleur).

[1] Source wikipédia

Les résultats consultables, montrent les comparaisons lors de la détermination de contours pour différents algorithmes ainsi que pour une segmentation manuelle (on demande alors à plusieurs personnes de tracer les contours principaux de l'image). Les figures II.4, II.5 et II.6 montrent un exemple de cette base de données. Un système de score a été défini pour quantifier les performances des algorithmes. Le score est basé sur la précision de la segmentation (probabilité qu'un pixel détecté comme frontière en soit effectivement un) et sur le paramètre "recall" (probabilité qu'une frontière soit détectée).

Exemple en planification

La communauté de l'intelligence artificielle a créé les compétitions en planification afin d'évaluer les performances des planificateurs développés (pour plus de détails voir les conférences internationales en planification : ICAPS[2]). Un formalisme (PDDL : Planning Domain Definition Language) a été mis au point pour permettre la comparaison des planificateurs. De plus, des problèmes type (proches des applications réelles) ont été conçus comme de véritables benchmarks. Au final les critères d'évaluation comprennent entre autres le temps nécessaire à la planification, le nombre de solutions, la taille des plans produits ...

Le principal inconvénient de cette compétition est que pour être comparés, les planificateurs doivent utiliser le formalisme développé. Ceci rend cette approche non générique et difficilement adaptable à tous les planificateurs. De plus dans le cas de la robotique autonome, nous sommes intéressé par l'action de l'ensemble du système et non par une seule de ses composantes.

II.3.2 Compétitions robotiques

On distingue deux types de compétition robotique : en premier lieu les compétitions où les robots (seuls ou en équipe) s'affrontent directement dans un match, et les compétitions dans lesquelles les robots doivent chacun à son tour accomplir une tâche précise.

Qu'il s'agisse d'une course organisée, d'épreuves ou de matchs, les compétitions robotiques permettent de comparer des robots entre eux. L'évaluation se faisant sur des critères communs, on peut les considérer comme de véritables benchmarks.

[2]http ://www.icaps-conference.org/

FIG. II.4 – Image de la base Berkeley en niveaux de gris

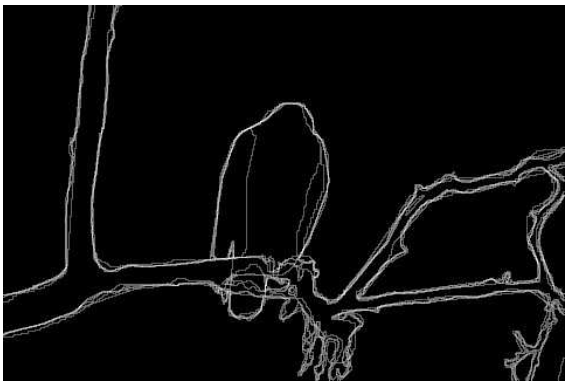


FIG. II.5 – Segmentation manuelle, score 0.92

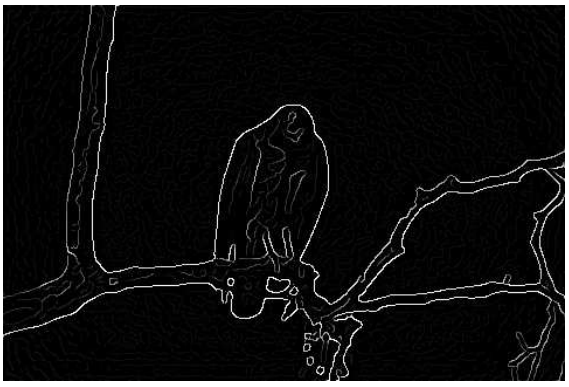


FIG. II.6 – Segmentation avec l'algorithme "gradient magnitude", score 0.90

Le DARPA grand challenge

Le grand challenge de la DARPA[3] est la compétition la plus ambitieuse à ce jour. Les "grand challenge" 2004 et 2005 consistent en une course à travers le désert, de véhicules entièrement autonomes. Les véhicules doivent emprunter une route pouvant comporter des passages dans l'eau, de la route pavée, des chemins de terre ou de sable, des passages obstrués naturellement ou artificiellement, ...

L'itinéraire est donné à chaque concurrent peu avant le départ, sous forme de points de passages (longitude, latitude, vitesse limite, taille du couloir de passage, temps limite). Un véhicule qui ne respecte pas les contraintes de passage est éliminé, de même s'il reste immobile trop longtemps.

La première session, qui a eu lieu en mars 2004, est longue de 200 miles qui doivent être parcourus en moins de 10 heures. Le véhicule *SandStorm* (fig. II.7) de la *Red Team* (Carnegie Mellon University) est celui qui a parcouru le plus de distance avec 7,4 miles. Sandstorm est un véhicule tout terrain. Il se localise par GPS et centrale inertielle, repère les obstacles grâce à des télémètres laser, une paire de caméras en stéréo-vision et un radar. Pour faire face à la quantité de calculs et de traitements, il embarque 7 processeurs (1 pentium III pour le contrôle de la direction et la supervision, 4 Itanium II sur serveur pour la planification de trajectoire et une plate forme bi-processeur à base de Xeon pour les lasers, caméras et radar).

La deuxième session du grand challenge a eu lieu en octobre 2005. Pour cette course de 10 heures, le parcours était de 132 miles dans le désert du Mojave. *Stanley*, le véhicule de l'université de Stanford (fig. II.8) a remporté la victoire en 6 heures et 54 minutes, suivi de près par les deux véhicules de Carnegie Mellon. En tout 6 véhicules ont réussi à terminer la course (dont 5 dans le temps imparti).

Le prochain grand challenge de la DARPA devrait avoir pour cadre la navigation en milieu urbain.

Ce type d'évènements permet de comparer les systèmes, mais aussi d'offrir un défi stimulant pour la communauté robotique. Il permet ainsi de faire progresser rapidement les technologies. Concernant l'évaluation des véhicules, on pourra reprocher à cette approche de ne pas prendre en compte en détail les échecs des véhicules. En effet un robot qui échoue sur un passage particulier est éliminé, ceci ne permet pas d'évaluer le spectre des capacités du robot (surtout si l'élimination intervient précocement). Dans ce sens on

[3]http ://www.darpa.mil/grandchallenge/index.asp

FIG. II.7 – Le véhicule “Sandstorm” de la Red Team (CMU) lors du grand challenge de 2004.

FIG. II.8 – Le véhicule “Stanley” de Stanford qui a remporté le grand challenge en 2005.

regrettera une évaluation incomplète des systèmes.

Les RoboCups

La RoboCupSoccer[4] dans laquelle des robots footballeurs s'affrontent dans différentes ligues est la plus ancienne des compétitions. Les différentes ligues comprennent : la simulation (des agents virtuels s'affrontent en équipes), des ligues en fonction des capacités physiques des robots (humanoïde, quadrupède, petits ou grands robots). Dans ce type de compétition, les robots (ou équipes) sont éliminés au fur et à mesure des défaites, le classement des systèmes se faisant par ordre éliminatoire.

Dans la RoboCupRescue[5] les robots ne s'affrontent pas directement, mais doivent localiser des mannequins dans des arènes encombrées. Les mannequins simulent des victimes d'une catastrophe (fig. II.9), ayant des conditions de santé différentes. Trois arènes de difficultés graduées sont utilisées dans cette compétition (fig. II.10). Pour comparer les robots sur des critères objectifs, un système de notation a été mis au point. Il se base sur le nombre de victimes découvertes, la précision de leur localisation et la qualité du diagnostic de l'état de la victime. Le score est pondéré par le niveau de difficulté de l'arène (les candidats doivent effectuer un parcours imposé et un parcours libre dans lequel ils choisissent le niveau de difficulté). A noter que dans cette compétition, il n'y pas obligation pour les robots à être autonomes : ils peuvent être téléopérés et/ou assistés sur certains points (le nombre d'opérateurs assistant le robot joue un rôle négatif sur le score final).

Les RoboCups donnent l'occasion de faire évoluer des robots sur le terrain et de les comparer entre eux. Elles ont l'avantage par rapport à d'autres types de benchmark, d'évaluer des systèmes intégrés et non pas une sous-partie de système. La RoboCupRescue est la compétition qui s'approche le plus d'un véritable benchmark de par son système de notation et ses métriques appropriés. On notera au passage que les arènes de test sont copiées sur le modèle de l'arène de référence du NIST. Le NIST a en effet développé une arène test pour évaluer les robots dans des missions de recherche et sauvetage urbain, dans le but de disposer d'un outil d'aide au développement des systèmes.

[4] www.robocup.org

[5] http ://www.rescuesystem.org/robocuprescue/

FIG. II.9 – Victimes simulées par des mannequins.

II.4 Autonomie partagée

Bien souvent, l'autonomie totale en robotique n'est pas l'objectif primordial, les robots doivent collaborer avec un ou des intervenants extérieurs dans bien des situations. De plus, la robotique autonome n'a pas encore la maturité nécessaire pour se passer complètement d'interventions extérieures. Un robot peut être amené à collaborer avec l'homme ou bien avec d'autres robots.

II.4.1 Interaction homme/robot

L'interaction homme/robot est un point essentiel en robotique, l'objet du robot est de satisfaire des ordres donnés par un opérateur humain. On peut regarder cet aspect sous l'angle de l'interface homme/machine, comme cela se fait dans d'autres domaines, mais on peut également étudier l'impact de la collaboration sur l'autonomie du robot.

SMART

Pour le projet SMART (Spacecraft Mission Assessment and Re-planning Tool), la Nasa a besoin d'outils pour spécifier le niveau d'autonomie de ses logiciels [Proud et al., 2003]. Ici les niveaux d'autonomie reflètent le partage des responsabilités entre la machine (ici l'ordinateur et non un robot) et l'homme : lorsqu'il est ici question de niveau d'autonomie il faut bien le voir comme un niveau de responsabilité. Plus le niveau est faible et plus

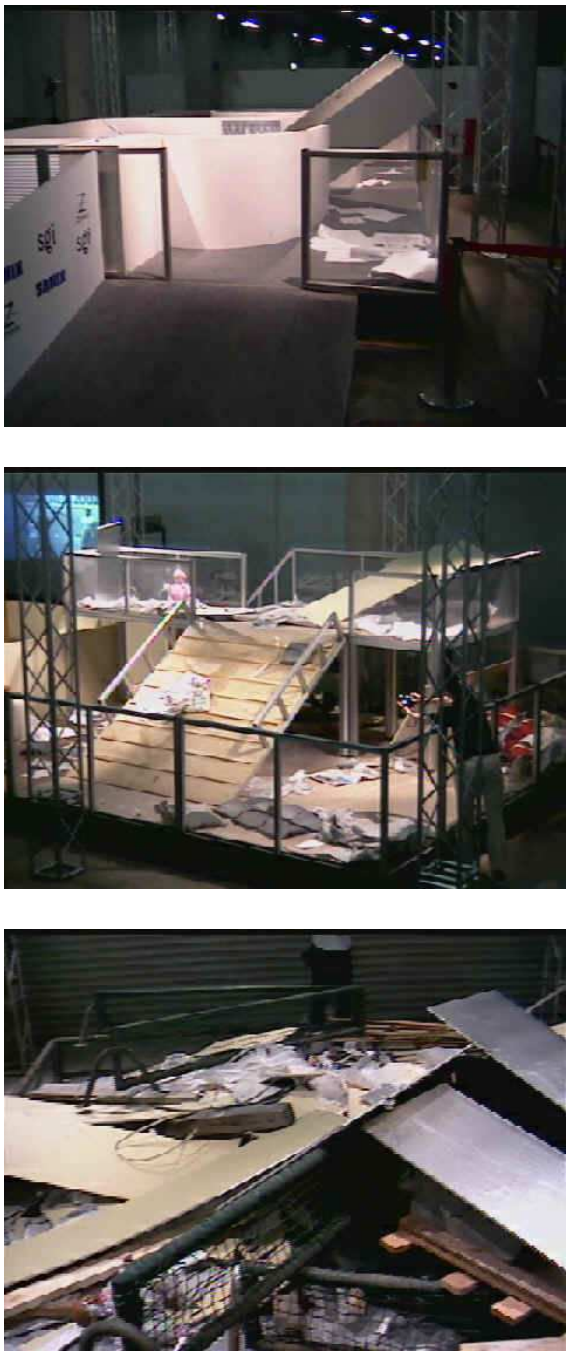


FIG. II.10 – Arènes de la RoboCupRescue. Le niveau de difficulté augmente de haut en bas.

l'homme est impliqué. Le tableau II.2 donne l'échelle d'autonomie dans ce projet. L'utilisation de ce tableau tranche sur le niveau de confiance que l'on a en l'homme ou en la machine. Si l'on considère que pour une tâche donnée l'ordinateur est plus sûr que l'homme, alors on lui donnera un grand niveau d'autonomie. A l'inverse, dans des tâches critiques pour lesquelles la machine n'est pas fiable, on lui confiera un petit niveau d'autonomie. La classification des systèmes se fait selon l'expertise des utilisateurs et concepteurs, c'est à dire qu'il n'existe pas de méthode d'évaluation quantitative pour de tels systèmes.

Niveau	Observation	Interprétation	Décision	Action
8	L'ordinateur collecte, filtre et ordonne les données sans afficher la moindre information à l'homme.	L'ordinateur prédit, interprète et intègre les données en un résultat qui n'est pas montré à l'homme.	L'ordinateur ordonne les tâches. Il effectue l'ordonnancement final sans afficher les résultats à l'homme.	L'ordinateur exécute les actions automatiquement sans autoriser d'intervention humaine.
7	L'ordinateur collecte, filtre et ordonne les données sans afficher la moindre information à l'homme. Un message "programme fonctionnant" est affiché.	L'ordinateur prédit, interprète et intègre les données en un résultat qui n'est montré à l'homme que dans certains contextes.	L'ordinateur ordonne les tâches. Il effectue l'ordonnancement final et affiche un nombre réduit d'options d'ordonnancement sans justifier ses choix.	L'ordinateur exécute les actions automatiquement et n'informe l'utilisateur que dans certains contextes. Il autorise les changements après l'exécution. L'homme surveille les cas particuliers.

Suite page suivante ...

... suite de la page précédente

Niveau	Observation	Interprétation	Décision	Action
6	L'ordinateur collecte, filtre et ordonne les données affichées à l'utilisateur.	L'ordinateur remplace les prédictions avec les analyses et interprète les données. Tous les résultats sont montrés à l'homme.	L'ordinateur ordonne les tâches et affiche un nombre réduit d'options d'ordonnancement en donnant les raisons de ses décisions.	L'ordinateur exécute les actions automatiquement, informe l'utilisateur et autorise les changements après l'exécution. L'homme surveille les cas particuliers.
5	L'ordinateur collecte les données, mais n'affiche que les informations filtrées non ordonnées.	L'ordinateur remplace les prédictions avec les analyses et interprète les données. L'homme surveille les interprétations pour les cas particuliers.	L'ordinateur ordonne les tâches et affiche tous les résultats ainsi que leurs motivations.	L'ordinateur autorise l'homme, dans certains contextes et pendant un temps limité, à un droit de veto avant l'exécution. L'homme surveille les cas particuliers.
4	L'ordinateur collecte les données et les affiche. Il souligne les informations pertinentes non ordonnées à l'utilisateur.	L'ordinateur analyse les données et fait des prédictions. Cependant l'homme est responsable de l'interprétation des données.	L'ordinateur et l'homme participent tout les deux à l'ordonnancement des tâches. Les résultats de l'ordinateur ont la priorité.	L'ordinateur autorise l'homme, pendant un temps préprogrammé, à un droit de veto avant l'exécution. L'homme surveille les cas particuliers.
3	L'ordinateur est responsable de la collecte d'information et affiche les informations non filtrées et non ordonnées à l'homme. L'homme est surveillant principal pour toutes les informations.	L'ordinateur est la première source d'analyses et de prédictions, avec l'homme surveillant les cas particuliers. L'homme est responsable de l'interprétation des données.	L'ordinateur et l'homme participent tout les deux à l'ordonnancement des tâches. Les résultats de l'ordinateur ont la priorité.	L'ordinateur exécute les actions après la validation par l'homme. L'homme surveille les cas particuliers.

Suite page suivante ...

... suite de la page précédente				
Niveau	**Observation**	**Interprétation**	**Décision**	**Action**
2	L'homme est la première source de collecte et de surveillance des données. L'ordinateur surveille les cas d'urgences.	L'homme est la première source d'analyse et de prédiction, avec l'ordinateur surveillant les cas particuliers. L'homme est responsable de l'interprétation des données.	L'homme ordonne les tâches, l'ordinateur peut être utilisé comme outil d'assistance.	L'homme est la première source d'exécution, l'ordinateur surveille les cas particuliers.
1	L'homme est le seul à collecter et surveiller les informations.	L'homme est responsable de l'analyse de toutes les données, des prédictions et de l'interprétation des résultats.	L'ordinateur n'assiste pas dans l'ordonnancement. L'homme doit tout faire lui même.	L'homme est le seul lors de l'exécution.

TAB. II.2 – Partage de l'autonomie entre l'homme et la machine dans le projet SMART de la Nasa.

Interface homme/robot

Crandall et Goodrich ([Goodrich et al., 2001, Crandall and Goodrich, 2003]) ont étudié l'influence du temps que l'utilisateur consacre à assister un robot sur les performances. Leur étude porte sur différents modes de commande d'un robot dans le cas d'une mission de navigation :

– robot totalement autonome (il dispose d'un ensemble de points de passage au début de sa mission)

– robot téléopéré (l'utilisateur utilise un joystick)

– robot semi-autonome (à chaque intersection il demande à l'utilisateur la voie à emprunter)

La figure II.11 montre de manière qualitative les résultats de ces expériences. Des métriques ont étés conçues afin de rendre compte de l'efficacité du robot. Dans le cas d'une mission de navigation, il s'agit de la vitesse instantanée.

Ces travaux sont les premiers à mettre en évidence, et ce de manière quantitative, le lien entre les performances d'un robot et l'interface utilisa-

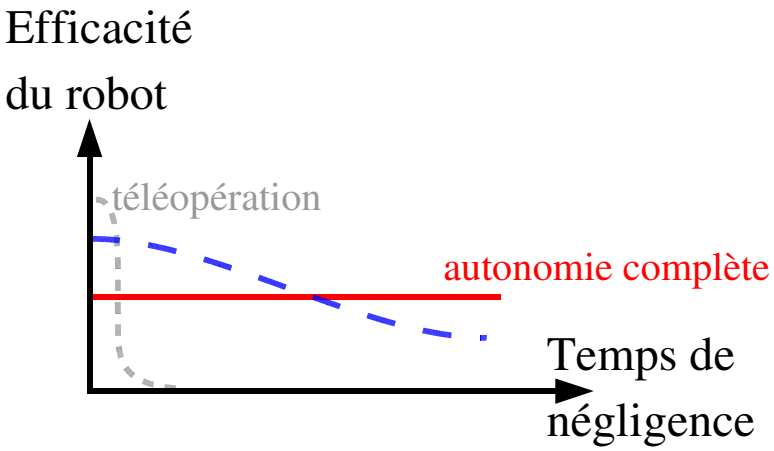


FIG. II.11 – Courbe de négligence. Efficacité du robot en fonction du temps pendant lequel l'opérateur néglige le robot. Trois cas de figures sont représentés : 1) le robot est entièrement autonome, 2) le robot est téléopéré et 3) un cas intermédiaire.

teur. Une utilisation de cette méthode est la prédiction du niveau d'interaction nécessaire entre l'homme et le robot, afin d'atteindre un niveau de performance donné, c'est à dire quel est le schéma d'interaction homme/robot le plus adapté pour un niveau de performance fixé. Une autre manière de faire est de modéliser la perte de performance au cours du temps lorsque le robot se trouve sans assistance, et de prédire le moment où l'opérateur devra intervenir pour augmenter les performances instantanées. Ceci se fait en fixant un seuil d'alarme sur l'efficacité, en dessous duquel l'homme doit intervenir. Cette façon de faire permet d'optimiser la mobilisation de l'opérateur dans les moments opportuns. La méthode s'inscrit naturellement dans le concept de l'autonomie ajustable, qui prévoit d'adapter l'autonomie des systèmes en fonction de la situation.

II.4.2 La coopération multi-robots

Ces dernières années, on voit de plus en plus de robots évoluer en équipe, collaborant à la réalisation d'une tâche. Par exemple le projet COMETS[6], utilise une flotte hétérogène d'UAV ayant des capacités et des modes de commande différents. Cette flotte est utilisée entre autres pour la surveillance

[6]http ://www.comets-uavs.org/

de feux de forêt ou la reconnaissance et la cartographie à partir du ciel.

Comment évaluer l'autonomie de telles équipes ? Comment se combine l'autonomie individuelle au sein de l'association ?

Brainov et Hexmoor [Brainov and Hexmoor, 2001] ont développé un cadre mathématique pour évaluer l'autonomie d'un agent ou d'un groupe d'agents, sous les angles suivants :

- l'autonomie décisionnelle et l'autonomie d'action d'un agent
- l'autonomie vis-à-vis d'un agent utilisateur
- l'autonomie vis-à-vis des autres agents ou groupes d'agents
- l'autonomie d'un groupe d'agents

L'autonomie est définie comme une relation entre 4 constituants :

- Le sujet (un agent ou un groupe d'agents) qui agit ou prend des décisions.
- L'objet de l'autonomie : un but ou une tâche à accomplir ou bien une décision à prendre.
- Les éléments influents : tout ce qui a un impact sur les actions et décisions du sujet, ce qui influe sur son comportement final (il peut s'agir de l'environnement, d'un autre agent, de l'utilisateur, ...). Ils peuvent aussi bien diminuer qu'augmenter l'autonomie.
- Une mesure de performance : mesure le succès, les performances du sujet en regard de l'objet de l'autonomie. Il s'agit d'une mesure dépendante du regard subjectif de l'utilisateur (il mesure ce qu'il juge important vis à vis de l'objet).

Autonomie décisionnelle et autonomie d'action

Pour mesurer l'autonomie, le problème est posé de la manière suivante par Brainov et Hexmoor [Brainov and Hexmoor, 2001] : un agent dispose à un instant t d'un ensemble C pour faire ses choix et d'une fonction de préférence P définie sur C. Dans la suite sont définis différents degrés d'autonomie.

Le degré d'autonomie préférentiel :

$$\frac{U[F(C,P')]}{U[F(C,P)]}$$

Où F(C,P) est la fonction de choix de l'agent s'il avait connaissance de toutes les préférences de l'utilisateur, F(C,P') est la fonction de choix de l'agent et U[.] est la fonction d'utilité de l'utilisateur (ie : la mesure de performance).

Un agent est autonome de manière préférentielle s'il a connaissance de toutes les préférences de l'utilisateur. Autrement dit, s'il fait les choix que ferait l'utilisateur.

Exemple

Prenons par exemple un robot dans une mission de navigation. Ce robot dispose d'un planificateur de trajectoires et d'un algorithme d'évitement d'obstacles. La situation est la suivante : le robot à un instant t se retrouve devant un obstacle non prévu dans sa trajectoire planifiée. Les choix C possibles sont :

1. de s'arrêter,
2. d'avancer vers l'obstacle (au risque d'une collision),
3. de faire appel au sous-système d'évitement d'obstacles,
4. de replanifier une trajectoire en prenant en compte cet obstacle.

La fonction d'utilité U[.] est le temps de la mission. L'utilisateur dans ce cas préférerait que le robot utilise l'évitement d'obstacles (la fonction F(C,P) conduit à l'action n^{o} 3). Le robot fait le choix de replanifier sa trajectoire (la fonction F(C,P') conduit au choix n^{o} 4). La fonction d'utilité, c'est à dire le temps de la mission est affectée par ces choix.

Le degré d'autonomie de choix :

$$\frac{U[F(C', P)]}{U[F(C, P)]}$$

Où C' est l'ensemble de choix de l'agent et C celui de l'utilisateur. Lorsque le nombre de choix est important, l'utilisateur en général n'est pas capable de lister tous les choix possibles. Dans ce cas, il est possible que le rapport soit supérieur à un et que donc l'agent soit capable de faire des choix plus performants que l'utilisateur.

Le degré d'autonomie décisionnel :

$$\frac{U[F(C', P')]}{U[F(C, P)]}$$

Si le rapport est égal à un, l'agent est complètement autonome du point de vue décisionnel, il est fidèle aux attentes de l'utilisateur. Si le rapport est inférieur à un, l'agent n'est pas autonome, il ne satisfait pas toutes les attentes de l'utilisateur. Si le rapport est plus grand que un, l'agent est super autonome, il dépasse les attentes de l'utilisateur.

Autonomie dans le contexte interaction agent/utilisateur

Soit v la mesure de performance. v_i^i est la performance de l'agent **i** et v_u^i est sa performance quand l'utilisateur le supervise. Le degré d'autonomie individuel A_u^i de l'agent **i** vis à vis de l'utilisateur u est défini comme :

$$A_u^i = \frac{v_i^i}{v_u^i}$$

A_u^i peut prendre toutes les valeurs sur $\mathbb{R}$.

Autonomie de groupe

Sont définies de la même manière, l'autonomie v_j^i de l'agent **i** vis à vis de l'agent **j** et l'autonomie v_{jk}^i de l'agent **i** vis à vis du groupe d'agents **(j,k)**. On peut alors définir une matrice d'autonomie. Par exemple, dans le cas de trois agents **(i,j,k)** :

$$\begin{pmatrix} v_i^i & v_j^i & v_k^i & v_{jk}^i \\ v_i^j & v_j^j & v_k^j & v_{ik}^j \\ v_i^k & v_j^k & v_k^k & v_{ij}^k \end{pmatrix}$$

De même sont définis les degrés d'autonomie suivants :

– degré d'autonomie de **i** vis à vis de **j** : $\mathrm{A}^i{}_j = \frac{v_j^i}{v_i^i}$

– degré d'autonomie de **i** vis à vis du groupe **(j,k)** : $\mathrm{A}^i{}_{jk} = \frac{v_{jk}^i}{v_i^i}$

– degré d'autonomie du groupe **(i,j)** : $\mathrm{A}^{ij} = \frac{v_j^i + v_i^j}{\max(v_i^i, v_j^j)}$

– degré d'autonomie du groupe S : $\mathrm{A}^S = \sum_{i \in S} A_{S-i}^i$; où S-i est le groupe S sans l'agent i

– degré d'autonomie du groupe S vis-à-vis de **k** : $\mathrm{A}^S{}_k = \frac{\sum_{i \in S}}{A_{S-i+k}^i A^S}$

Une fois toutes ces valeurs définies, on peut chercher le groupe qui a l'autonomie maximum. On a alors un problème NP complet à résoudre.

A notre connaissance ce cadre mathématique n'a pas été utilisé pour l'évaluation de systèmes concrets. Cependant cette formalisation semble intéressante pour l'étude de l'autonomie d'un groupe de robots.

Pour appliquer ce formalisme plusieurs problèmes surgissent :

– il faut être capable de lister les choix d'actions possibles selon chaque situation

– il faut connaître les choix que ferait l'utilisateur ferait dans une situation donnée

– il faut identifier les choix pris par le système en fonction de la situation

– il faut définir et mesurer les fonctions v qui sont les mesures de performances

Tout ceci rend cette méthode difficile à mettre en œuvre en pratique.

II.5 Conclusion

Comme on le voit les robots sont de plus en plus complexes et intègrent un grand nombre de fonctionnalités. Capacités opérationnelles et capacités décisionnelles se combinent pour donner de l'autonomie au robot. Mais comment évaluer de tel systèmes ? Le système de classification par niveau d'autonomie est la manière la plus pratique pour comparer des systèmes : une lecture dans un tableau donne directement les capacités du robot. Néanmoins il est nécessaire de définir des critères objectifs et quantitatifs afin de déterminer pour un robot donné quel est son niveau d'autonomie, ou plutôt quels sont ses niveaux d'autonomie. En effet nous avons vu que l'autonomie est relative à la tâche que l'on exécute, il semble donc que pour une tâche donnée un robot possède un certain niveau d'autonomie, qui ne sera pas le même sur une autre tâche. Le système ALFUS prévoit des critères et métriques afin de définir les moyens pour évaluer le niveau d'autonomie du robot. Mais les métriques sont mélangées sans prendre garde. Effectivement la complexité d'une mission ou les difficultés environnementales jouent sur l'autonomie d'un robot. Ces paramètres doivent faire partie du système de mesure, mais il faut les définir indépendamment du robot que l'on cherche à évaluer.

Le benchmarking est un bon moyen pour comparer les robots entre eux. Mais doit-on construire des benchmarks pour chaque composante du robot ou pour le robot dans son intégralité ? Si l'on cherche à évaluer les algorithmes et fonctions les uns après les autres, on risque de ne pas évaluer l'interaction entre ces fonctions. Or nous savons qu'une source de problèmes importante est l'interaction entre composants du robot. Néanmoins l'évaluation individuelle est une bonne pratique d'ingénierie et est par conséquent une phase importante du développement d'une fonctionnalité robotique.

Les compétitions robotiques permettent cette évaluation globale sur des robots intégrés. Le *Grand Challenge* est sans conteste la compétition la plus spectaculaire de par ses ambitions et dernières réussites. D'après les éléments rendus public, on peut lui reprocher de ne pas tester toutes les capacités du robot dans le sens ou un échec stoppe l'évaluation. Une question à laquelle nous n'avons pour le moment pas de réponse concerne les critères qui ont décidés du parcours. Est ce que le trajet est conçu comme un véritable benchmark avec un inventaire des situations que l'on souhaite tester ou est ce que seules quelques difficultés ont été placées sur le parcours sans réel étude globale ?

L'interaction homme/robot est un point important dans l'autonomie des robots. Le système de classification ALFUS le souligne en le prenant comme un des axes principaux d'étude. Les études de Crandall et Goodrich mettent également l'accent sur l'impact de la participation de l'homme sur l'autonomie. Bien que notre propre étude ne se focalise pas sur ce point dans un premier temps, nous montrerons comment on peut l'intégrer. La collaboration multi-robots est aussi un point intéressant à regarder. Là encore cela ne fait pas partie de notre étude, mais le cadre proposé par Brainov et Hexmoor peut permettre de comprendre l'autonomie de flotte hétérogène de robots.

Chapitre III

Méthodologie

L'autonomie d'un système est relative à la tâche à accomplir. Pour la robotique nous parlerons de missions. Il nous est donc nécessaire de définir un cadre formel pour décrire une mission robotique. Des critères pour quantifier l'autonomie du robot vis-à-vis d'une mission sont également à développer. Nous avons vu précédemment que la complexité de la tâche ainsi que les difficultés environnementales influencent l'autonomie. Pour mesurer l'efficacité (l'autonomie) du robot lors de l'exécution d'une tâche, nous nous focalisons sur la mesure des performances du système, seules quantités réellement mesurables.

Dans ce qui suit, on considère un robot qui ne bénéficie pas d'interventions extérieures pour l'aider (il n'y a pas d'interaction homme/robot). Comme facteurs influents de l'autonomie nous retenons la complexité de l'environnement, mais également la quantité d'information que le robot possède. Ce dernier point est une partie originale et nouvelle, qui à toute sa place dans l'évaluation de tels systèmes. En effet, plus le robot possède d'in-

formations exactes sur l'environnement et plus il a de chance de faire des choix conduisant à de meilleures performances.

Ce chapitre décrit nos propositions pour exprimer une mission en termes pratiques et mesurables. Des métriques liées aux performances sont explorées. Des métriques pour la mesure de la complexité d'un environnement ainsi que pour calculer la quantité d'information ont été développées. Pour illustrer la démarche et la définition de métriques, nous détaillerons une mission de navigation et expliciterons chaque composante.

III.1 Démarche générale

III.1.1 Spécification d'une mission

Lorsqu'il est question de se prononcer sur l'autonomie d'un système, une part subjective entre en ligne de compte. Un même système qui effectue une tâche précise, peut être vu comme autonome pour un premier utilisateur et non autonome pour un autre. Ceci est fonction du niveau d'exigence que l'on se fixe. Il est nécessaire de prendre en compte cette part subjective dans toute tentative de description formalisée d'une mission. Un autre point à considérer concerne les paramètres externes qui influent sur le déroulement de la mission, entre autres l'environnement.

La figure III.1 montre le raisonnement servant à la décomposition d'une mission ainsi que les différentes phases permettant une évaluation des systèmes.

La mission est définie comme un ensemble de contraintes. D'une part celles fixées en tant qu'objectifs pour la mission et d'autre part celles fixées par le cadre extérieur.

Contraintes liées aux objectifs de la mission

Dans cette partie, on retrouve le côté subjectif de l'évaluation. On y trouvera l'ensemble des contraintes et critères que l'utilisateur fixe pour la mission. On peut voir cette partie comme un cahier des charges que fixe l'utilisateur pour cette mission. Il décrit alors quelles sont ses exigences quant aux performances que le robot doit atteindre pour la mission.

D'une manière pratique pour l'utilisateur, on pourra y trouver des termes tels que :

- la robustesse
- la réactivité
- la durée de la mission
- l'utilisation de ressources

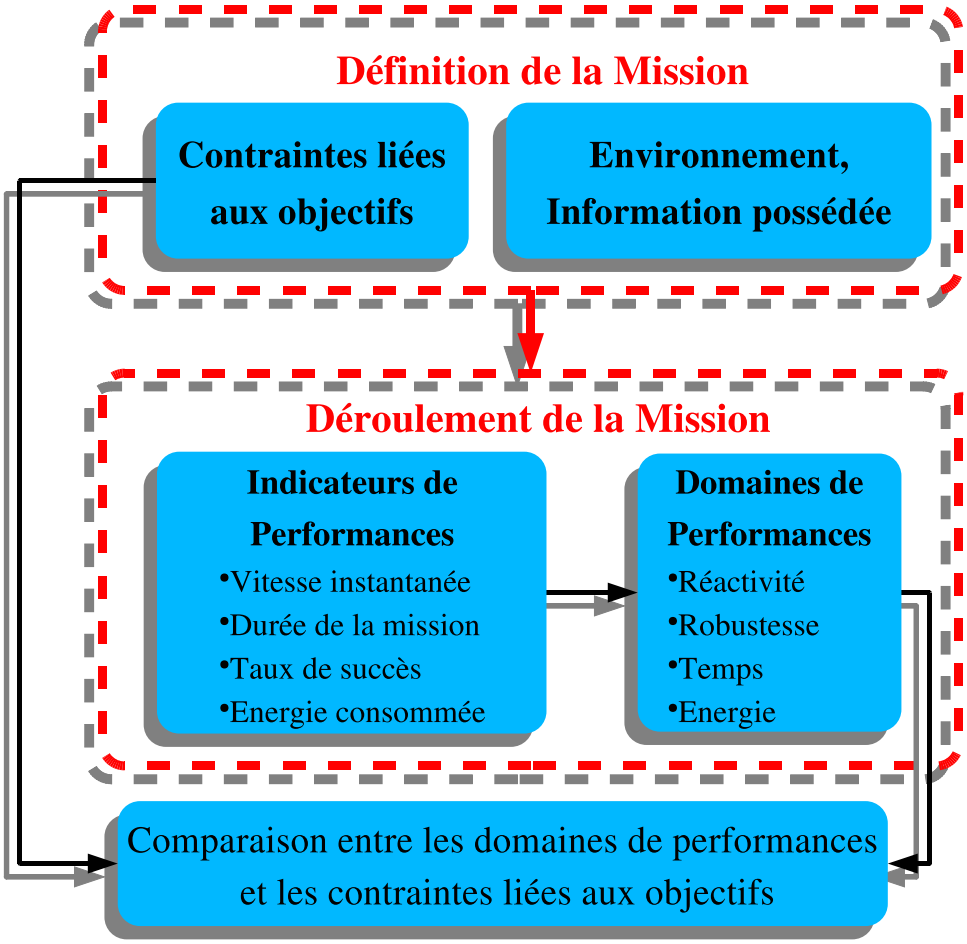


FIG. III.1 – Formalisme utilisé pour la description d'une mission robotique.

– des niveaux de performances
– le taux de succès

La *robustesse* est un terme très utilisé en robotique. On peut le définir comme la capacité d'un système à surmonter les variations externes. D'une manière pratique on peut l'extraire des performances mesurées du système. La figure III.2 représente les performances de deux systèmes en fonction d'un paramètre externe ayant un effet négatif (par exemple la vitesse instantanée pour le paramètre de performance et le nombre d'obstacles autour du robot pour le paramètre influent). Le système représenté par la courbe *a* voit ses performances chuter lorsque la valeur du paramètre néfaste augmente. Alors que le système représenté par la courbe *b* n'est pas sensible aux variations du paramètre externe : il est robuste vis-à-vis de ce paramètre. Si l'on considère les performances du système *a*, entre les deux lignes en pointillé ses performances ne varient pas beaucoup. Il est donc robuste sur cette plage de valeurs.

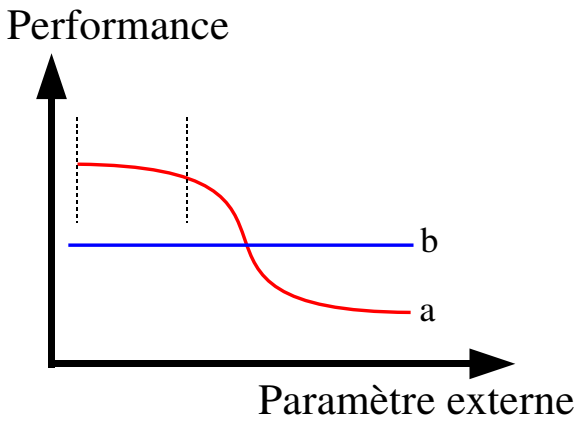


Fig. III.2 – Illustration de la robustesse.

La *réactivité* est le temps nécessaire au système pour réagir aux évènements. Le temps de réaction est décomposé en deux phases : la première est le temps nécessaire pour détecter l'évènement et la seconde le temps nécessaire pour traiter cet évènement. La figure III.3 montre la décomposition du temps de réaction.

Les autres termes sont relatifs aux métriques dont nous parlerons dans la section III.2. L'utilisation de ressources pourra être par exemple une contrainte sur l'utilisation de l'énergie. Des exemples concrets du calcul de la robustesse et de la réactivité se trouvent dans le chapitre V.

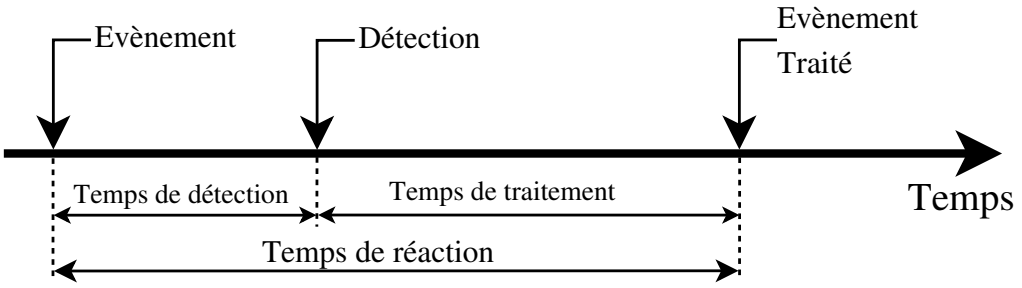


FIG. III.3 – Décomposition du temps de réaction : Réactivité du système.

Contraintes environnementales

Les contraintes environnementales comprennent l'ensemble des paramètres que nous ne maîtrisons pas. Y figure en premier plan l'environnement. Plus il est complexe et moins le robot risque d'être performant et autonome. Il est a priori plus facile d'évoluer dans un environnement statique comportant peu d'objets, que dans un environnement dynamique très encombré. Un autre paramètre influent est la quantité et la qualité de l'information que l'on possède sur l'environnement. Si l'on considère que l'on fournit toutes les connaissances nécessaires au robot ou qu'il possède les moyens de les obtenir par ses moyens de perception et de traitement, alors il a des chances de faire des choix conduisant à de meilleures performances et à plus d'autonomie. L'information est une contrainte externe seulement dans le cas où l'on dote le robot de toutes les données que l'on possède. Si l'on choisit d'occulter une partie de l'information au robot, alors il faut la considérer comme une contrainte supplémentaire de l'utilisateur.

III.1.2 Paramètres influents sur le déroulement de la mission

On essaiera de lister tous les paramètres qui peuvent influencer le déroulement de la mission et avoir un impact sur les performances du robot. Outre les contraintes environnementales citées plus haut, l'état interne du robot influence son comportement. La qualité de la localisation est fonction de l'environnement et de l'information, et affecte les performances du système. Une localisation très mauvaise peut entraîner par exemple un blocage du robot. Une analyse des états internes du robot peut aider dans la phase de développement à améliorer certaines fonctions en identifiant les problèmes éventuels.

En faisant varier les paramètres influents et en mesurant les perfor-

mances qui en résultent, nous serons capable d'évaluer des robots et de les comparer sur une même échelle. Ce faisant nous sommes en mesure d'extraire les domaines de performance du robot (fig. III.1 et III.4). Lorsque les domaines de performance ont été exprimés de manière similaire aux contraintes liées aux objectifs, alors on peut comparer les deux et voir dans quelle mesure le robot atteint les objectifs fixés par l'utilisateur.

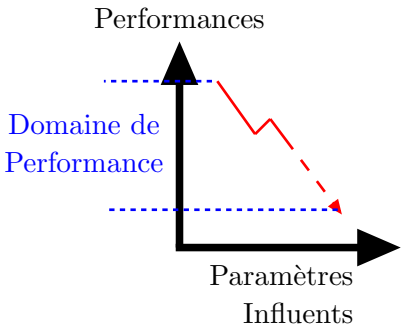


FIG. III.4 – Domaine de performance.

III.1.3 Une analogie : le système de l'éducation nationale

Pour bien comprendre la méthode d'évaluation proposée, une analogie avec le système d'évaluation de l'éducation nationale est possible.

Le sujet est un élève et sa mission est d'obtenir des diplômes d'examens (les diplômes constituent l'objectif). Les contraintes externes sont la difficulté des examens (brevet des collèges, BAC, ...) et les connaissances du sujet (les cours qu'il a suivis et la qualité de ces cours). Les critères de performance sont les notes obtenues à l'issue d'épreuves tests. A la fin des épreuves on compare les performances de l'élève vis-à-vis des attentes de l'examinateur (ici il joue le rôle de l'utilisateur). Le minimum demandé par l'utilisateur est une moyenne de 10/20 pour obtenir le diplôme et si l'élève dépasse les attentes de l'utilisateur (avec une moyenne supérieure), alors on peut lui attribuer une mention.

On retrouve ici une comparaison entre niveaux d'autonomie et niveaux scolaire. Pour définir les niveaux d'autonomie il nous suffirait alors de créer des examens de passage pour chaque niveau, comme cela est fait pour les diplômes de l'éducation nationale. Chaque niveau d'autonomie serait alors

validé par un test benchmark, le robot qui réussirait le test serait alors accrédité pour ce niveau d'autonomie.

III.2 Métriques liées aux performances

Les métriques choisies doivent être révélatrices du déroulement de la mission. Cependant pour un type de mission donnée il existe souvent plusieurs métriques possibles. On peut alors choisir de toutes les regarder, mais parfois seules certaines nous intéressent. De plus certains critères de performance peuvent être opposés.

Lorsque l'on évalue un robot pour une mission donnée, on peut le faire localement en regardant ses performances instantanées ou on peut le faire globalement en s'intéressant à l'ensemble de la mission (l'évaluation n'est alors possible qu'au terme de la mission).

Les critères de performance qui suivent sont ceux que l'on peut utiliser pour une mission de navigation :

- vitesse du robot (instantanée ou moyenne sur la mission)
- distance parcourue
- durée de la mission
- taux de succès de la mission (statistique sur plusieurs missions)
- ...

Au premier abord vitesse, temps et distance semblent corrélés, or ceci n'est pas forcément vrai. La vitesse est adaptée à l'environnement et à la proximité des obstacles. De plus chacune de ces métriques n'a pas la même signification selon que l'on cherche à naviguer en temps minimum ou avec une distance parcourue la plus petite possible. En effet des contraintes imposées au robot peuvent influencer son comportement.

On retrouve dans cette liste de performances des critères locaux (vitesse instantanée) et des critères globaux (durée de la mission).

III.3 Métriques liées à l'environnement

Pour une mission de navigation les paramètres externes qui affectent le plus les performances, sont liés à l'environnement : à sa complexité d'une part et à l'information que l'on possède à son sujet d'autre part. Nous proposons des métriques pour quantifier la complexité et l'information. La plupart de ces métriques sont inspirées de la théorie de l'information et de la mesure entropique.

Pour certaines métriques, nous utilisons comme base de départ une grille d'occupation. Cette grille représente l'environnement discrétisé dans laquelle

chaque cellule représente la probabilité d'occupation de l'espace par un obstacle (fig. III.5). Deux grilles sont utilisées : la première pour représenter l'environnement dans lequel le robot évolue et la seconde pour représenter les connaissances du robot.

III.3.1 Complexité de l'environnement

La littérature offre quelques exemples de métriques pour quantifier la complexité d'un environnement. Elles se focalisent essentiellement sur des environnements lors de mission de navigation.

[Crandall and Goodrich, 2003] propose une métrique locale qui se base sur le nombre d'intersections par unité de surface et sur le nombre d'obstacles par unité de surface. Le nombre d'intersections est calculé à partir des distances retournées par les sonars (ils calculent le nombre de chemins possibles pour une distance fixe autour du robot). Le nombre d'obstacles est plus compliqué à obtenir, il se base sur l'entropie directionnelle[1], l'accélération du robot et les variations de distances des sonars au cours du temps. On peut reprocher à cette métrique l'utilisation de données qui dépendent de la conception du robot. L'entropie directionnelle dépend directement des réglages d'asservissement de la direction. Un mauvais réglage qui conduirait à des instabilités dans la direction du robot ne serait pas l'image de la complexité de l'environnement mais d'une mauvaise conception. De même pour les accélérations du robot.

La DARPA pour vérifier que le démonstrateur *DEMO III XUVs* a atteint le niveau TRL-6 (Technology Readiness Level : voir le tableau II.1) a imaginé un moyen d'évaluation [Albus, 2002b]. Il s'agit de comparer un véhicule autonome à un véhicule piloté par un homme. Tous deux doivent effectuer la même mission de navigation (aller d'un point à un autre sans chemin imposé). Pour comparer les deux véhicules, une caractérisation du terrain parcourue est envisagée selon cinq scénarios. Les scénarios sont classés par ordre de complexité, un scénario plus complexe comprend tous les moyens d'évaluation des niveaux inférieurs :

– Une mesure basique consiste à faire noter de façon subjective la difficulté du terrain par un ou plusieurs hommes.

– Les accidents du terrain sont mesurés par une centrale inertielle, une caméra et un pare-choc instrumenté.

– Les mesures sont complétées par un relevé LADAR haute résolution.

– On regarde de plus les propriétés mécaniques du terrain (stabilité, ...).

[1] L'entropie directionnelle est l'image de la fréquence des changements de direction du robot.

– Le dernier scénario prévoit de caractériser le terrain aussi par des données aériennes.

Ces scénarios de caractérisation du terrain sont lourds à mettre en place et risquent de produire un trop grand nombre de données à interpréter. A notre connaissance cette méthode n'a pas été utilisée.

Nous proposons dans ce qui suit deux types de métriques pour mesurer la complexité d'un environnement lors d'une mission de navigation : une métrique globale qui rend compte de la complexité de la totalité de l'environnement et une métrique locale qui rend compte de la complexité dans une petite zone autour du robot.

Mesure de complexité globale.

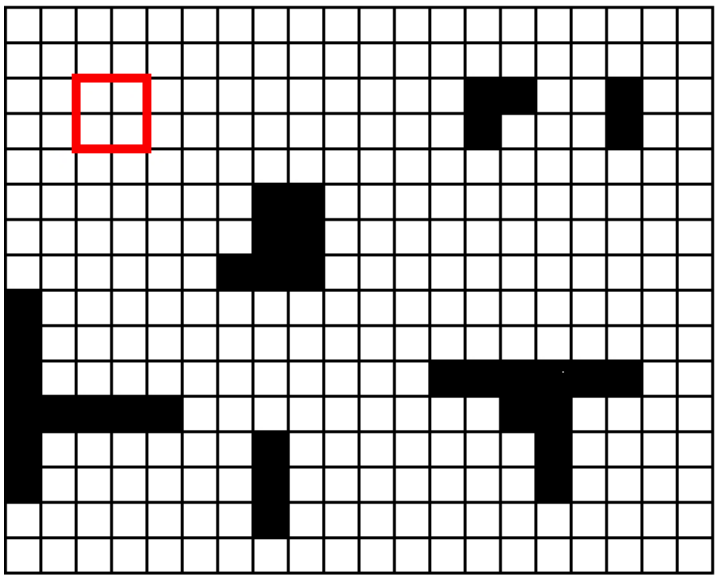


FIG. III.5 – Grille d'occupation avec un masque servant au calcul de la complexité globale.

Elle se base sur la carte d'occupation de l'environnement et sur une mesure entropique. Nous plaçons un masque sur la grille d'occupation (fig. III.5), il nous sert à calculer la densité d'obstacles dans cette zone. Pour un masque 2x2 et des probabilités d'occupation binaire (ce qui correspond à la description d'un environnement réel, qui intrinsèquement n'a pas d'incertitudes), nous avons cinq niveaux de densités possibles : $0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1$. Nous déplaçons le masque sur toute la grille et comptons le nombre d'occurrences de chaque niveau de densité. Ceci nous permet d'obtenir un histogramme

sur la répartition statistique des niveaux de densités. Nous pouvons alors faire un calcul entropique sur cet histogramme :

$$H = \sum_i -p(dens_i)\log(p(dens_i))$$

avec $p(dens_i)$ = fréquence du $i^{ème}$ niveau de densité dans la grille

$$\sum_i p(dens_i) = 1$$

Ce calcul évalue la répartition moyenne des obstacles dans l'environnement.

Nous avons :

$$H = 0 \Leftrightarrow \exists i \backslash p(dens_i) = 1$$

L'entropie est maximum quand il y a uniformité des niveaux de densité :

$$H = \log(n * m + 1) \Leftrightarrow dens_i = \frac{1}{n * m + 1}$$

$n * m$ est la taille du masque.

Si l'on considère une taille de masque petite devant la dimension des obstacles, alors le cas où il n'y a qu'un seul niveau de densité n'arrive pratiquement que lorsqu'il n'y a pas d'obstacle dans l'environnement. En effet, le cas où l'environnement est un obstacle géant n'a pas d'intérêt pour nous et n'avoir qu'une seule densité d'obstacle est très peu probable. L'obtention du maximum d'entropie suppose des fréquences égales pour tous les niveaux et dans un cas réel ceci n'arrive pratiquement pas.

Cette métrique peut être calculée de manière globale sur l'ensemble de l'environnement, ou de manière locale dans une zone limitée autour du robot. Seule la mesure globale a été utilisée lors de nos essais expérimentaux (mais il est probable que cette métrique donne des résultats satisfaisants lorsqu'elle est utilisée localement).

La figure III.6 montre un environnement de 10x40m qui est rempli d'obstacles générés de manière aléatoire. On voit que plus l'environnement est "chargé" d'obstacles et plus l'entropie augmente. Pour des environnements qui semblent de complexités semblables (ex. environnements n^{o} 6 et n^{o} 7), l'entropie est du même ordre de grandeur.

Les environnements n^{o} 7 et n^{o} 8 montrent l'intérêt d'une mesure entropique : ces deux environnements ont la même densité moyenne d'obstacles. Or de manière subjective on aurait tendance à penser que le premier est plus complexe que le second. L'utilisation de l'entropie nous dit que c'est le

cas. En fait, faire un calcul entropique avec un masque d'une certaine taille permet de prendre un considération que l'environnement est plus ou moins structuré (l'environnement n^{o} 8 est constitué de deux murs parallèles).

Mesure de complexité locale.

Pour rendre compte des difficultés locales du robot, nous avons conçu des métriques basées sur la proximité des obstacles. Nous avons développé deux métriques assez semblables dans l'esprit (fig. III.7).

La première mesure l'espace libre devant le robot. Un demi-cercle est tracé devant le robot. Dans ce demi-cercle on relève la valeur du plus grand angle libre d'obstacles ainsi que la position de sa bissectrice. Ces deux valeurs seront nos métriques concernant la complexité. Plus l'angle est grand et moins l'environnement est complexe localement (le robot a plus d'espace pour se déplacer). La position de l'angle peut nous renseigner sur le comportement symétrique du robot (selon que les obstacles sont à gauche ou à droite, est-ce que son comportement est le même ?), de plus on espère que les obstacles sur les côtés affectent moins le robot que les obstacles devant lui. Cette métrique un peu compliquée à ensuite été remplacée par une autre. Cependant elle peut être utile si l'on cherche à connaître l'influence de la position des obstacles sur le comportement du robot.

La deuxième métrique utilisée se base sur la surface de visibilité autour du robot (fig. III.7, partie grisée). Avec cette métrique, contrairement à la précédente, la position des obstacles par rapport au robot n'est pas prise en compte. Seule la surface occupée par l'environnement dans une zone proche du robot est mesurée.

L'utilisation de ces métriques est montrée dans le chapitre V. Elles sont également discutées, notamment en ce qui concerne la taille de la zone d'intérêt.

III.3.2 Information possédée sur l'environnement

La mesure de la complexité globale est tirée de la théorie de l'information. Cette théorie propose des outils pour calculer l'information mutuelle entre deux distributions de probabilités. C'est en s'inspirant de ceci que nous avons développé nos métriques.

Représentation des données.

La carte de l'environnement que le robot possède est représentée sous forme d'une liste de segments. Cette liste comprend pour chaque segment

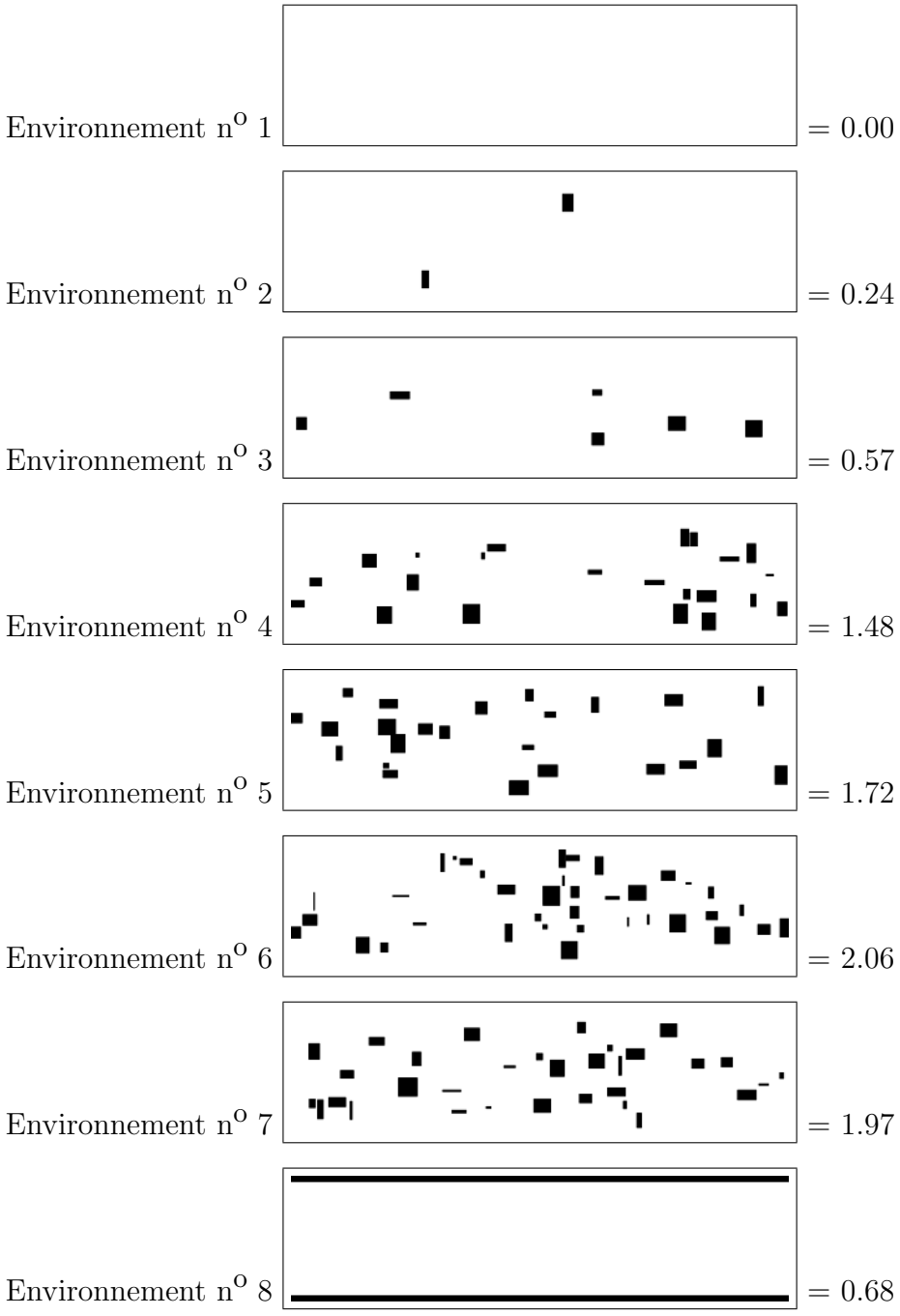


FIG. III.6 – Exemples d'environnements et entropie correspondante. Les environnements font 10x40m, la taille d'une cellule de la grille d'occupation est de 10x10cm. L'entropie est calculée avec un masque de 8x8 cellules.

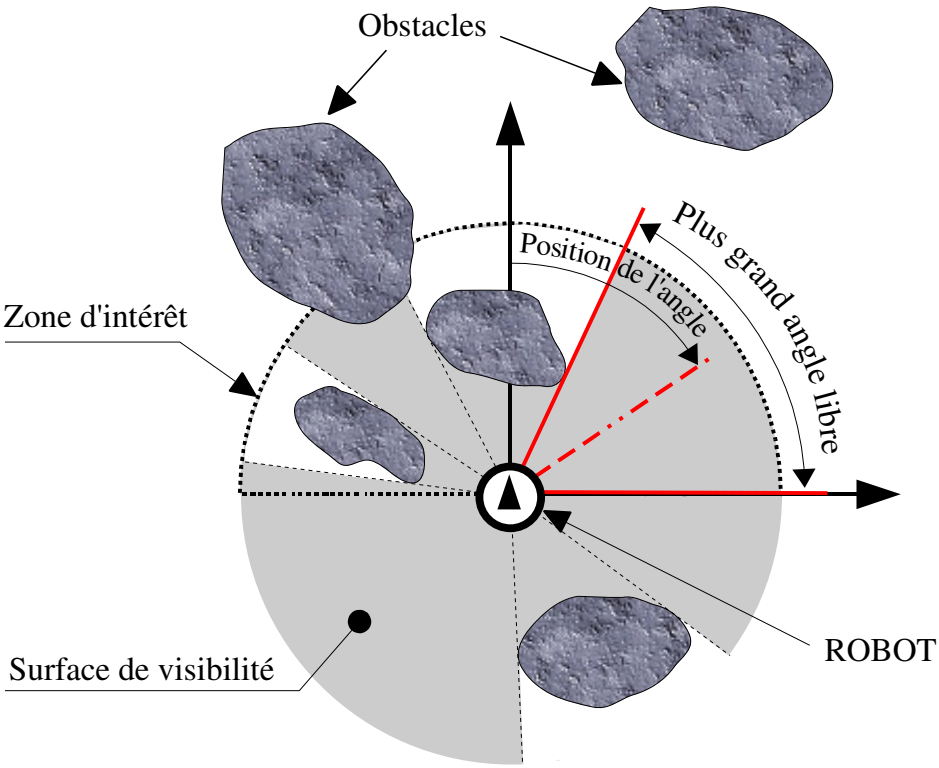


FIG. III.7 – Mesure de complexité locale. Une première métrique consiste à regarder le plus grand angle libre d'obstacles dans une zone d'intérêt (demi-cercle en pointillé) ainsi que sa position. Une seconde métrique est basée sur la surface de visibilité (en grise).

les coordonnées des extrémités, les coordonnées polaires de la droite porteuse ainsi que les incertitudes sur la position de cette droite. L'annexe B.1 montre comment sont représentés les cartes de segments dans le simulateur robotique. De plus cette annexe détaille les calculs nécessaires au passage d'une carte de segments à une grille d'occupation probabiliste.

Les figures III.8 et III.9 montrent respectivement la carte de l'environnement et un exemple de carte donnée au robot (elle est volontairement incomplète vis-à-vis de la carte de l'environnement). La figure III.10 montre un exemple de grille d'occupation obtenue à partir d'une carte de segments en prenant en compte les incertitudes.

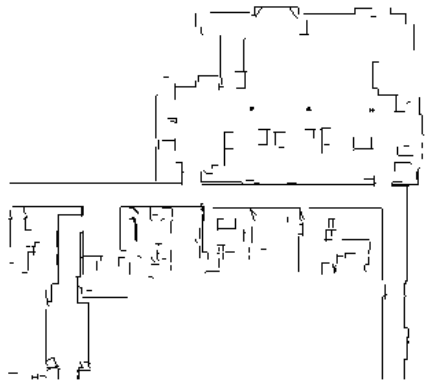


FIG. III.8 – Exemple de carte de l'environnement.

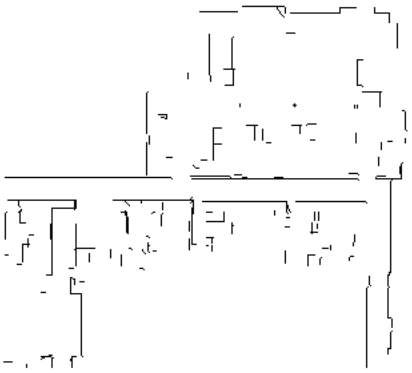


FIG. III.9 – Exemple de carte possédée par le robot.

Notation

Dans la suite lorsque l'on parlera d'information contenue dans une carte, il faudra considérer la grille d'occupation comme étant cette information.

Une grille d'occupation discrétise l'environnement et chaque cellule de la grille représente la probabilité que l'espace soit occupé dans cette cellule (ie. qu'il y ait un obstacle). La grille d'occupation peut être représentée dans un tableau :

$$\begin{array}{ccccc} p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & p_{m3} & \cdots & p_{mn} \end{array}$$

Dans un environnement de NxM mètres avec une discrétisation de dxd mètres, alors nous aurons un tableau de taille nxm avec : $n = \frac{N}{d}$ et $m = \frac{M}{d}$.

FIG. III.10 – Grille d'occupation obtenue à partir d'une carte de segments, les incertitudes sur les segments sont représentées : plus la couleur va vers le gris clair et plus la probabilité d'occupation est faible.

p_{12} est la probabilité que la cellule 12 soit occupé par un obstacle. Pour obtenir la probabilité que cet espace est vide il nous suffit de prendre le complémentaire à 1.

Une grille d'occupation peut être facilement visualisée en une image en niveaux de gris, il suffit de faire correspondre les probabilités à des niveaux de gris. $p_{ij} = 1$ correspond au noir, $p_{kl} = 0$ correspond au blanc et les valeurs intermédiaires à des nuances de gris (fig. III.10).

Calcul de la quantité d'information.

La théorie de l'information donne un outil pour calculer la quantité d'information redondante dans deux messages distincts. C'est à partir de l'entropie conditionnelle, que l'on obtient cette quantité : c'est l'information mutuelle. Elle se calcule de la manière suivante :

Information Mutuelle. *Soit M l'information contenue dans la carte du monde (par exemple fig. III.8) et soit R l'information contenue dans la carte du robot (par exemple fig. III.9).*

Alors, l'entropie conditionnelle est la quantité :

$$H(M|R) = -\sum_{m,r} P(M = m, R = r) \log P(M = m|R = r)$$

Avec

$$P(M = m|R = r) = \frac{P(M = m, R = r)}{P(R = r)}$$

*Et **l'information mutuelle** est la quantité :*

$$I(M, R) = H(M) - H(M|R)$$

En pratique pour les calculs,$\sum_{m,r}$ est une somme sur chaque cellule de la grille. Pour une cellule de l'espace $P(M = m, R = r)$ prend successivement les valeurs : probabilité que la cellule de la carte du monde soit occupée et que la cellule de la carte du robot soit occupée, probabilité que la cellule de la carte du monde soit occupée et que la cellule de la carte du robot soit libre, probabilité que la cellule de la carte du monde soit libre et que la cellule de la carte du robot soit occupée, probabilité que la cellule de la carte du monde soit libre et que la cellule de la carte du robot soit libre.

Malheureusement, l'information mutuelle fait appel aux probabilités jointes ou conditionnelles. Or dans la plupart des cas réels, on ne dispose pas de ces distributions de probabilités pour faire les calculs. Nous avons donc utilisé une autre métrique pour comparer directement deux distribution de probabilités. Il s'agit d'une forme symétrique de la distance de *Kullback-Leibler*.

Distance de Kullback-Leibler. *Soit M l'information contenue dans la carte du monde (par exemple fig. III.8) et soit R l'information contenue dans la carte du robot (par exemple fig. III.9).*

La distance de Kullback-Leibler, également appelée entropie relative de M par rapport à R, pour la cellule i est :

$$D_i(m|r) = \sum_{x \in X} m_i(x) * \log \frac{m_i(x)}{r_i(x)}$$

Où $m_i(x)$ (resp. $r_i(x)$) est distribution de probabilités pour la $i^{ème}$ cellule de la grille du monde (resp. la carte du robot).

La forme symétrique, que nous utiliserons comme mesure de l'information *est :*

$$Info_i(m, r) = \frac{D_i(m|r) + D_i(r|m)}{2}$$

Alors la quantité d'information autour du robot est définie par :

$$INFO(M, R) = \frac{1}{Norm} * \sum_{i \in A} Info_i(m, r)$$

Où $Norm$ est un facteur de normalisation et A est une aire de taille fixe autour du robot.

Nous avons : $INFO(M, R) = 0 \Leftrightarrow$ le robot possède toute l'information potentiellement disponible dans l'environnement.

III.3.3 Mise en œuvre : les calculs en pratique

La complexité locale

Nous ne détaillerons pas le calcul du plus grand angle libre devant le robot. L'obtention de cette grandeur est triviale grâce aux données issues du télémètre laser plan.

La surface de visibilité n'est pas très compliquée à obtenir non plus, elle nécessite cependant d'instrumenter le robot avec un deuxième télémètre laser qui sera mis de telle sorte que l'on puisse mesurer les distances aux obstacles sur 360°. Dans notre cas nous utilisons la simulation (voir les sections IV.1 et IV.3) et donc il nous est facile d'ajouter un capteur.

Chaque nappe laser retourne 361 mesures de distance (résolution de 0,5°). Il suffit ensuite de faire une intégration pour obtenir la surface de visibilité normalisée. Avec d_i la i^{ème} distance mesurée et d_{max} le rayon de la zone d'intérêt (rayon maximum considéré) alors la surface normalisée se calcule de la manière suivante :

$$Surf = \frac{1}{d_{max}^2} * \sum_i \frac{d_i^2}{360}$$

La grille d'occupation

Deux grilles sont à calculer. La première est la grille d'occupation de l'environnement réel, qui peut être calculée une fois pour toutes (si l'on ne change pas d'environnement entre deux simulations). La deuxième grille concerne la carte du robot et devra être recalculée à chaque ajout de segment dans la carte. Les cartes sont calculées en coordonnées cartésiennes avec un pas de grille de 5cm. L'annexe B détaille le calcul d'une grille d'occupation.

Quelques remarques concernant ce calcul. La distribution de probabilité des segments est exprimée en coordonnées polaires et nous faisons un changement de repère pour obtenir une grille cartésienne. Ceci nous conduit à faire des approximations et peut être que la méthode gagnerait à être repensée en coordonnées polaires. Notamment la précision des calculs diminue lorsque l'on s'éloigne de l'origine. Une piste serait pour le calcul de la quantité d'information, de ne faire les calculs que localement autour du robot (ce qui pose le problème de recalculer la grille très régulièrement).

III.4 Une nouvelle méthode d'analyse : les "system maps"

Une système map [Held and Sukkarieh, 2005] est une représentation probabiliste d'un système. Elle permet de modéliser un système dynamique complexe en utilisant l'interaction qui peut exister entre plusieurs métriques. Le degré d'interaction entre ces métriques est le modèle qui permet de comprendre le système. On l'obtient grâce à la théorie des modèles graphiques et à l'inférence statistique. Les system maps s'appuient sur la théorie des réseaux bayésiens dynamiques [Dean and Kanazawa, 1990].

Le travail décrit ci-aprés est le fruit d'une collaboration avec Jason Held un étudiant de l'université de Sydney. Nous présentons ici le schémas général de cette méthode sans entrer dans les détails de l'implémentation.

Dans ce qui suit M est un ensemble de métriques considérées comme des variables aléatoires continues qui peuvent être exprimées sous forme de distribution gaussienne. M_t^i est la réalisation de la métriques M_i à l'instant t.

III.4.1 Les réseaux bayésiens dynamiques

Un réseau bayésien (BN) est défini formellement par un ensemble $B = \langle G, \theta \rangle$ où G est un graphe composé de nœuds représentant des variables aléatoires et d'arcs représentant les relations de dépendance entre variables (probabilités conditionnelles). θ est un ensemble de paramètres qui définissent les transitions entre les nœuds. Les réseaux bayésiens dynamiques (DBN) constituent une extension qui permet de modéliser des systèmes dynamiques en tant que processus stochastiques grâce à la prise en compte de l'évolution du système entre deux intervalles de temps. En théorie des graphes, on le représente comme un couple $(B_1, B_{\rightarrow})$, où B_1 est le réseau bayésien formé avec l'information a priori $P(M)$ (fig. III.11), et $B_{\rightarrow}$ est un réseau bayésien à deux pas de temps. Ce dernier définit la transition entre l'information $P(M)$ et l'information $P(M|M_{t-1})$ à l'instant suivant (fig. III.12). Il s'agit d'un graphe direct (l'influence d'une métrique sur une autre est une relation unidirectionnelle) et acyclique (il n'y a pas de transition pour revenir sur un état passé).

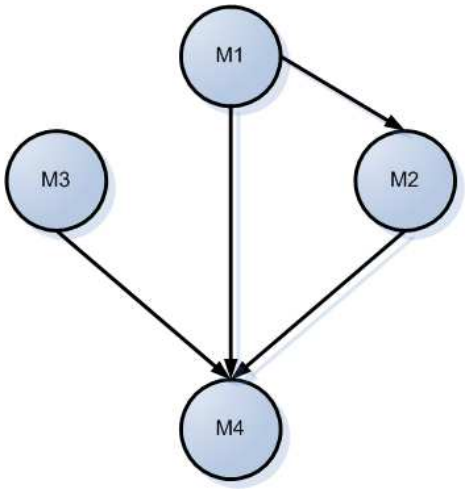


FIG. III.11 – Graphe Direct Acyclique (DAG) montrant l'interaction apprise entre quatre métriques

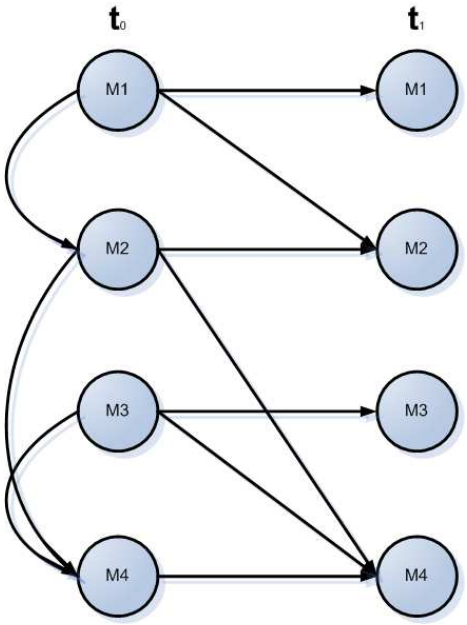


FIG. III.12 – Réseau bayésien à deux pas de temps pour t_0 et t_1. Cette structure se "déroule" en répétant le réseau t_0 à t_1 pour la transition entre t_1 et t_2.

On peut inférer la valeur des variables à l'instant t sachant leur valeur

à l'instant $t-1$ en donnant l'expression des probabilités jointes :

$$P(M_t|M_{t-1}) = \prod_{i=1}^{N} P(M_t^i|Pa(M_t^i)),$$

Où $Pa(M_t^i)$ sont les prédécesseurs de M_t^i et N est le nombre de métriques (variables aléatoires). Le réseaux peut être "déroulé" jusqu'à un instant futur T pour inférer les distributions de probabilités à cet instant. Cette inférence est de la forme :

$$P(M_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{N} P(M_t^i|Pa(M_t^i)).$$

III.4.2 Les system maps

Construire la system map d'un système dynamique revient à déterminer deux types de données. Premièrement il nous faut la structure du réseau (influences entre métriques). Ensuite il nous faut trouver les probabilités conditionnelles, qui représenteront le degré d'interaction plus ou moins fort entre les métriques.

Au départ la structure est inconnue, mais nous faisons l'hypothèse que le système est entièrement observable. Pour déterminer la structure nous recherchons un DAG optimal. Comme nous ne disposons pas d'information a priori sur les transitions, l'espace de recherche comprend $2^{N_2 log N}$ DAG possibles (N est le nombre de métriques). Cette recherche se fait sous forme d'apprentissage hors ligne à partir des données collectées lors de différents essais. La recherche du DAG optimal se fait grâce à l'algorithme K2 [Cooper and Herskovits, 1992], un algorithme glouton. Partant d'un ensemble vide (fig. III.13) une famille de DAG est comparée aux autres familles de DAG possibles selon le critère du maximum de vraisemblance dans une recherche en largeur. La famille avec le maximum de vraisemblance est alors ajoutée au DAG et cette connexion est supprimée de l'espace de recherche. Ce processus continue jusqu'à ce que toutes les transitions possibles pour une famille donnée soient testées. Lorsque toutes les combinaisons possibles ont été évaluées, l'algorithme s'arrête.

Le maximum de vraisemblance détermine le meilleur paramètre θ pour le jeu de métriques échantillonnés M_L :

$$\widehat{\theta}_{ML} = \arg\max_{\theta}[P(M_L|\theta)].$$

Le critère du maximum de vraisemblance dans un espace de recherche non restreint conduit à une system map complètement connectée. A la place

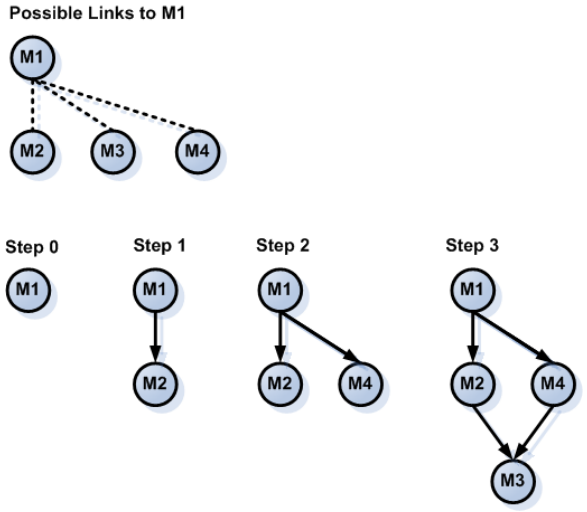


FIG. III.13 – Exemple de construction d'une system map avec l'algorithme glouton. Ici la famille {M1-M2, M1-M4} est plus vraisemblable que la famille {M1-M2, M1-M3}, ce qui conduit à la structure de l'étape 2. Ensuite M3 est ajouté dans la dernière étape.

le critère BIC (Bayesian Information Criterion) [Heckerman, 1996] est utilisé pour noter une nouvelle transition et ce à chaque pas de temps :

$$BIC_M = logP(D|G, \widehat{\theta}_{ML}) - \frac{d}{2} logN$$

Où $logP(D|G, \widehat{\theta}_{ML})$ est le log de vraisemblance de la structure particulière défini par G et $\frac{d}{2} \log N$ est la complexité du modèle (appelée densité structurale). d est le nombre de paramètres pour N échantillons et D est un échantillon aléatoire de données. Le calcul du critère BIC pour une system map demande de faire les calculs à chaque pas de temps en utilisant les paramètres vraisemblables de la manière suivante :

$$BIC(G) = \sum_i \sum_t logP(M_i|Pa(M_i), \widehat{\theta}_i, M_t^i) - \frac{d_i}{2} logN,$$

La system map est le $\widehat{\theta}$ qui obtient la notation la plus haute selon les critères précédents. On la représente comme une matrice d'interaction entre les différentes métriques. La valeur d'interaction est le paramètre du maximum de vraisemblance. Une system map doit comporter des zéros sur sa diagonale puisque le graphe est acyclique. Les paramètres peuvent être positifs ou négatifs, et ne sont pas forcement compris dans l'intervalle $[0, 1]$.

III.4.3 A quoi cela peut-il nous servir ?

Nous avons vu dans les sections précédentes un certain nombres de métriques, certaines liées aux performances d'un robot, d'autres liées à la complexité de l'environnement ou à l'information possédée sur cet environnement. Avec les system maps nous pouvons faire l'apprentissage d'un réseau qui représentera le poids d'une métrique sur les autres. Par exemple obtenir une valeur quantitative sur l'influence de la complexité de l'environnement sur les performances du robot, nous permettra de constater si le robot voit ses performances affectées par l'environnement. On pourra alors extraire une grandeur représentative de la robustesse du robot vis-à-vis de la complexité environnementale. La section V.2 montre l'utilisation des system maps sur des données issues de simulations et les conclusions que l'on peut tirer de ce type d'analyse.

III.5 Conclusion

Nous avons décomposé une mission robotique de telle sorte que l'on fasse la distinction entre ce qui est imposé par l'utilisateur et ce qui est imposé par les conditions externes. Chaque composante issue de cette décomposition est décrite de manière pratique et mesurable. Notre approche s'appuie sur une évaluation du robot par l'intermédiaire de ses performances, vis-à-vis de la variabilité de la complexité environnementale mais également vis-à-vis de l'information qu'il possède. De plus nous avons vu que le comportement du robot peut être étudié globalement ou localement. Nous avons proposé des métriques associées pour chaque cas d'étude et pour chaque composante.

Les métriques associées à la complexité globale de l'environnement et à la quantité d'information sont inspirées de la théorie de l'information. L'idée d'utiliser l'entropie conditionnelle et l'information mutuelle s'est confrontée dans notre cas au problème de l'accès aux probabilités conditionnelles. La distance de Kullback-Leibler surmonte cet inconvénient. Dans le cas où l'on disposerait de tous les éléments, l'information mutuelle peut être une mesure intéressante.

Ici nous nous préoccupons des métriques associées à une mission de navigation. Pour d'autres types de missions, il faudrait redéfinir des métriques pertinentes notamment en ce qui concerne les difficultés environnementales et les paramètres externes influents. Trouver des métriques relatives aux performances d'un système est souvent assez facile et par exemple les statistiques sur le nombre de missions réussies sont réutilisables dans beaucoup de cas. Sur la mission de navigation les métriques liées aux performances sont assez intuitives et naturelles. Par contre l'effort est plus grand pour

les autres métriques. Par exemple pour une tâche de manipulation d'objet, trouver des métriques concernant la complexité de l'environnement semble ardu (surtout si on recherche des métriques quantitatives).

Les system maps permettent de faire l'apprentissage des liens qui lient les métriques entre elles et donc de déterminer l'influence de l'environnement et de l'information sur les performances du robot. Elle nous serviront de méthode pour analyser l'interaction entre des métriques dès lors que le nombre de métriques est supérieur à trois et que l'on ne peut plus avoir de représentation graphique..

Chapitre IV

Mise en œuvre et expérimentation

Dans cette section nous décrivons notre approche pour valider expérimentalement les métriques et concepts décrits précédemment. Les tests ont été réalisés grâce à un simulateur robotique que nous avons développé en coopération avec Sylvain Joyeux.

IV.1 Simulation et benchmark

Pour être capable d'évaluer et de comparer des robots selon les mêmes critères nous avons besoin d'un protocole expérimental bien cadré. La mise en place d'un *banc d'essai (ou benchmark)* qui permette la maîtrise de l'environnement et la reproductibilité des expériences est nécessaire.

IV.1.1 Maîtrise de l'environnement

Notre hypothèse de travail est que l'environnement est un paramètre qui influe sur les performances du robot. Nous avons développé des métriques

à ce propos. L'utilisation de ces métriques impose une maîtrise de l'environnement. Maîtrise dans le sens où il nous est nécessaire de connaître les éléments qui constituent cet environnement afin d'appliquer les métriques. Il nous faut donc à un moment donné avoir accès aux dimensions spatiales. De plus pour pouvoir tester un grand nombre de configurations, il faut que l'environnement soit facilement reconfigurable ou qu'il soit très grand et qu'il représente plusieurs configurations.

Enfin, pour comparer deux robots selon les mêmes critères, il faut que l'environnement puisse être le même pour tous. Il est donc nécessaire d'assurer un minimum de reproductibilité.

Ces raisons conduisent à deux types de choix expérimentaux :

– soit on conçoit un champ de test comme c'est le cas dans les compétition robotique (voir section II.3.2).

– soit on utilise un environnement virtuel.

IV.1.2 Le choix de la simulation

Il est difficile dans un laboratoire de recherche de figer un environnement dans la durée. Il est également difficile de mesurer précisément cet environnement. C'est pourquoi nous nous sommes orientés vers la simulation robotique. Les avantages en termes pratiques sont nombreux :

– environnement facilement mesurable

– environnement reconfigurable à volonté

– possibilité de mémoriser un environnement et de le réutiliser plus tard

– mise en oeuvre expérimentale simplifiée

La simulation offre également un gain de temps non négligeable. En effet, une fois la simulation mise au point on peut réaliser un grand nombre de tests en un temps limité (une centaine par nuit). Chose impossible à faire lorsque l'on doit préparer un robot réel pour sa mission, puis à la fin le remettre dans un état opérationnel.

IV.2 Intégration dans une architecture robotique

Nous avons choisi le simulateur open source *GAZEBO*[1] comme base de départ.

Sylvain Joyeux a réalisé l'intégration du simulateur au sein de l'architecture robotique. Il a notamment programmé toute la partie qui s'occupe

[1] http ://playerstage.sourceforge.net/index.php ?src=gazebo

de la cohérence temporelle entre la simulation et les logiciels existants. Pour plus d'information, le lecteur pourra consulter son rapport de stage [Joyeux, 2004].

Gazebo est un simulateur qui intègre un moteur d'interaction physique (il s'appuie sur Open Dynamics Engine : ODE[2]). Il offre ainsi un certain réalisme quant à l'interaction entre objets simulés et permet une modélisation assez fine du robot (en particulier la répartition de masse, la puissance des moteurs ...).

Le choix a été fait d'intégrer la simulation de telle sorte que les logiciels qui font fonctionner les robots du laboratoire ne voient pas de différence entre robot réel et robot simulé. Si l'on reprend l'architecture LAAS, nous avons remplacé les bibliothèques qui constituent le système logique par des bibliothèques qui font le lien entre les capteurs et actionneurs simulés (fig. IV.1). Simuler uniquement le matériel et laisser telle quelle la partie logicielle permet de tester les algorithmes tel qu'ils sont utilisés sur les robots réels.

Deux robots du laboratoire ont été modélisés. Le premier est un robot d'intérieur (fig. IV.2) et le second est un robot d'exploration d'extérieur (fig. IV.3). De plus un modèle de nappe laser 2D (SICK) a été réalisé.

Ces deux robots utilisent le même microcontrôleur pour les commandes de mouvements, le retour odométrique et l'utilisation des sonars de proximité.

Dans les scénarios que nous avons étudiés nous n'utilisons que le niveaux fonctionnel de l'architecture LAAS (voir fig. II.1). Cependant toute l'architecture est capable de fonctionner en simulation.

Benjamin Lussier est un doctorant du groupe de recherche Tolérance aux fautes et Sûreté de Fonctionnement (TSF) du Laas. Il travaille sur l'architecture LAAS et sur les mécanismes de tolérance aux fautes que l'on pourrait y intégrer. Plus particulièrement il étudie le comportement du planificateur IxTeT grâce au simulateur robotique. Pour ces travaux toute l'architecture robotique est utilisée au sein du simulateur [Lussier et al., 2005].

IV.3 Scénarios étudiés

Deux scénarios ont été expérimentés afin de valider la méthode d'évaluation proposée. Le premier, très simple, consiste en la traversée d'un couloir encombré. Le second utilise plus de fonctionnalités, il consiste en une mission de navigation entre plusieurs endroits du laboratoire. Dans tous les scénarios, nous utilisons le modèle de robot d'intérieur et les algorithmes qui vont avec.

[2]http ://www.ode.org/

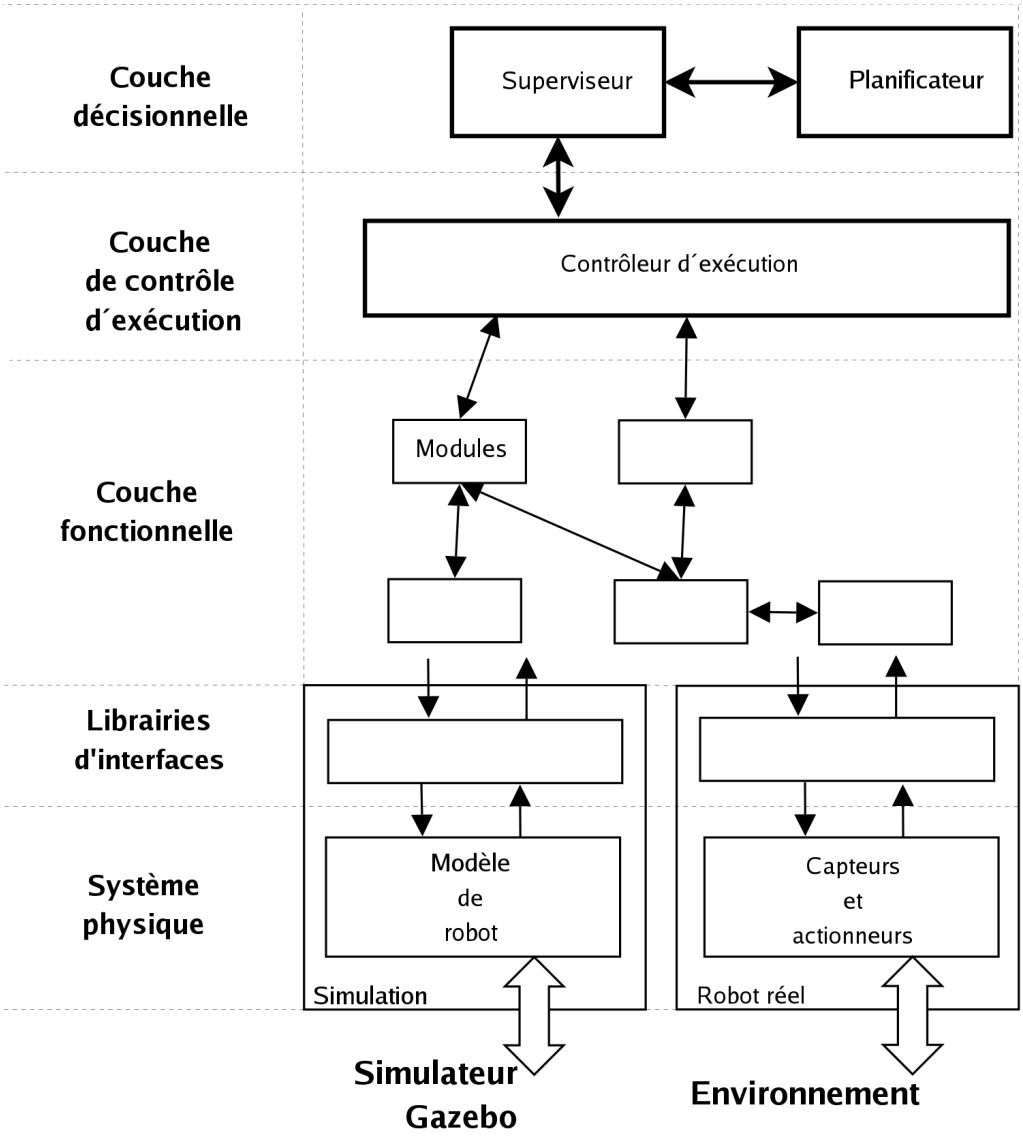


FIG. IV.1 – Architecture de contrôle robotique du LAAS : comparaison entre l'implémentation sur robot réel et sur robot simulé.

FIG. IV.2 – Modèle du robot Rackham.

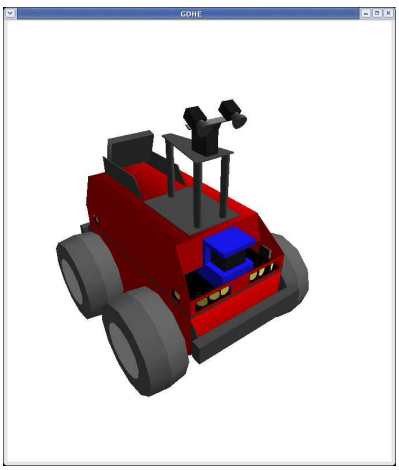


FIG. IV.3 – Modèle du robot Dala.

IV.3.1 Corridor

L'environnement - La mission

L'environnement est un couloir encombré de 40m de long et 10m de large. La mission pour le robot est de traverser le couloir le plus rapidement possible. S'il est bloqué, la mission est considérée comme ayant échoué. A chaque nouvelle mission des obstacles sont placés aléatoirement dans le couloir. La figure IV.4 montre le robot au point de départ dans un tel environnement.

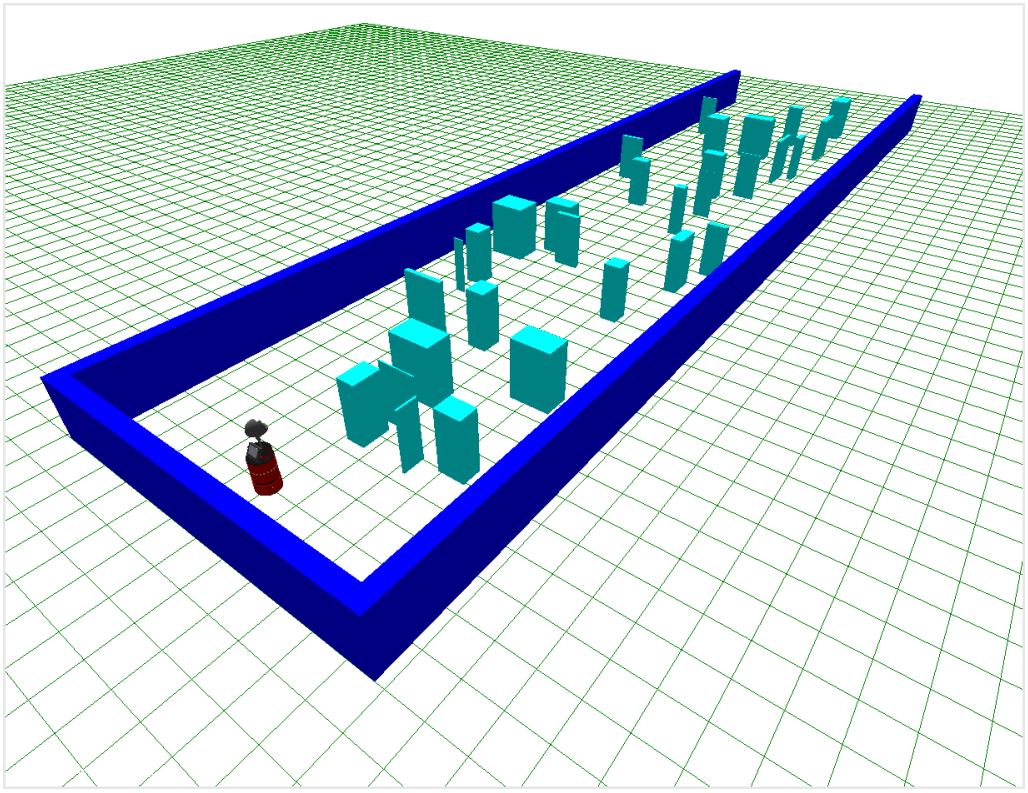


FIG. IV.4 – Robot virtuel dans l'environnement de simulation de type corridor.

Le robot

Dans ce scénario, le robot dispose de fonctionnalités de base :
- fonctions de locomotion
- retour odométrique (il s'agit de son seul moyen de localisation)
- un algorithme local d'évitement d'obstacles [Minguez et al., 2004]

Le robot ne dispose pas de contrôleur d'exécution et se localise relativement à son point de départ. Lors du début de la mission on lui envoie l'ordre d'aller au point $(41, 0)$ ce qui correspond à la traversée complète du couloir. Si l'ordre échoue ou si le robot est bloqué, alors il n'y a aucune possibilité de reprise et la mission échoue.

Afin de comparer des systèmes différents et comme nous ne disposions pas de plusieurs robots pour faire des comparaisons, nous avons fait varier les paramètres de l'algorithme d'évitement d'obstacles. Un paramètre en particulier a pris quatre valeurs différentes lors des séries de tests. Il s'agit de la distance de sécurité aux obstacles qui a pris les valeurs de 0.3, 0.6, 1.0 et 1.5m. Cet artifice nous a permis de varier le comportement du robot lors des campagnes de simulations.

Les métriques

Deux indicateurs de performance ont été utilisés avec ce scénario :
- le ratio $\frac{\text{vitesse instantanée}}{\text{vitesse maximale}}$ a servi d'indicateur de performance local
- le temps mis à traverser le couloir a servi d'indicateur global

Concernant l'environnement, seules les métriques de complexité ont été utilisées. La mesure globale s'appuie sur la mesure entropique (section III.3.1) et la mesure locale s'appuie sur la première métrique explicitée section III.3.1 (nous avons deux paramètres pour cette métrique : la valeur du plus grand angle libre d'obstacles et sa position).

La quantité d'information que le robot possède sur cet environnement est nulle (il n'a pas de carte du monde et ne possède pas de planificateur de trajectoires). Nous n'utilisons donc pas la métrique associée.

IV.3.2 Exploration du laboratoire

Ce scénario est plus complexe et correspond à une mission de navigation. L'environnement est plus complexe et le robot possède un plus grand nombre de fonctionnalités concernant la navigation.

L'environnement - La mission

L'environnement est issu d'une carte de segments acquis avec un robot réel lors du parcours d'une partie du laboratoire. Les segments qui sont représentés dans un plan 2D, sont étirés selon l'axe vertical afin de créer des murs ou autres obstacles dans un environnement à 3 dimensions.

La mission pour le robot est de se déplacer successivement à trois destinations tirées aléatoirement dans une liste qui en comporte 8. La figure

IV.5 montre l'environnement du laboratoire en 3D ainsi que les destinations possibles pour la mission. Si le robot échoue en cours de route, des routines sont prévues afin de le remettre dans un état opérationnel, lui permettre d'acquérir localement de l'information et de repartir. 10 tentatives sont autorisées pour chaque destination.

Les connaissances du robot sur son environnement sont incomplètes dans ce scénario. On retire aléatoirement des segments de la carte réelle de l'environnement (fig. III.8) afin d'obtenir une carte partielle qui sera l'information initiale du robot (fig. III.9).

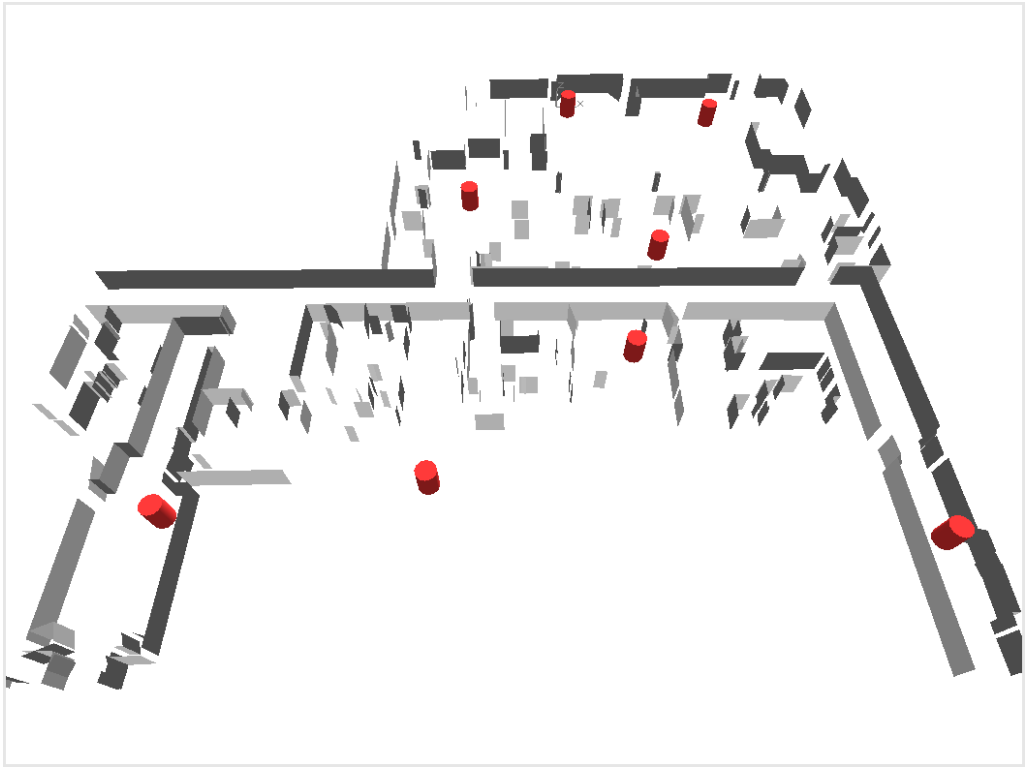


Fig. IV.5 – Environnement pour la mission "exploration du laboratoire". L'environnement correspond aux environs de la salle robotique du LAAS, cartographiés par un robot. Les cylindres représentent les destinations possibles lors de la mission. La taille des cylindres correspond approximativement aux dimensions du robot.

Le robot

Le robot dans ce scénario est le même que précédemment avec des fonctionnalités supplémentaires :

– le robot possède une carte de segments de l'environnement
– localisation sur segments
– fusion des estimateurs de position (localisation sur segments et odométrie)
– un planificateur de trajectoire utilise la carte de segments
– possibilité d'acquérir de nouveaux segments de manière locale et de les intégrer à la carte

Le contrôle d'exécution est très basique, il consiste en des scripts qui se lancent sur certaines valeurs de retour des requêtes de contrôle. Lorsque le robot est bloqué, une séquence de reprise est lancée (dans la limite des 10 tentatives autorisées) :
– arrêt du robot
– ré-initialisation des fonctions qui en ont besoin
– acquisition de segments dans la position d'arrêt
– replanification d'une trajectoire
– exécution de la nouvelle trajectoire

Les métriques

L'analyse du comportement du robot est locale dans ce scénario. Nous utilisons les métriques suivantes :
– état du robot : bloqué ou en mouvement (il s'agit d'une variable binaire)
– le ratio $\frac{\text{vitesse instantanée}}{\text{vitesse maximale}}$
– le deuxième indicateur de complexité locale décrit dans la section III.3.1 (surface libre autour du robot)
– quantité d'information commune entre la carte du monde et celle du robot (distance de Kullback-Leibler)

L'état du robot est déterminé après simulation, en regardant rétroactivement dans les logs. Lorsque le robot est bloqué, il renvoie un message d'alerte. Or dans la plupart des cas au moment où il retourne ce message, il était déjà bloqué depuis un certain temps cherchant un passage libre. D'un point de vue externe le robot est comme "hésitant" et ses déplacements ne sont pas utiles. Pour déterminer l'instant où le robot est réellement bloqué, on regarde les n derniers mètres parcourus et l'instant correspondant est celui où l'on considère le robot comme bloqué (fig. IV.6). De plus on peut mesurer le temps de réaction du robot vis-à-vis des blocages (voir la section III.1.1).

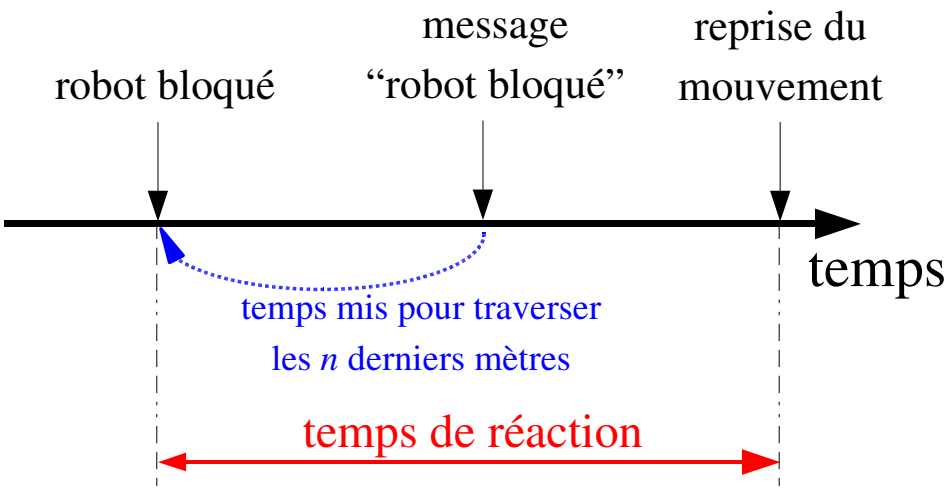


FIG. IV.6 – Méthode pour déterminer si le robot est dans un état “bloqué”. On peut également mesurer le temps de réaction du robot. Le temps de détection est le temps entre le moment où le robot est bloqué et le moment où le robot retourne le message de blocage. Le temps de recouvrement (de réaction) est le temps entre le moment où le message apparaît et le moment où le robot reprend son déplacement : c'est le temps nécessaire à l'ajout de segments dans la carte, à la replanification de trajectoire et la ré-initialisation des fonctions concernées.

Chapitre V

Analyse des données

V.1 Premier scénario : le corridor

Pour chaque valeur de la distance de sécurité (voir section IV.3.1) nous avons fait une série de 100 essais afin de collecter des données. Une série de 100 essais correspond environ à une dizaine d'heures de simulations. Pour chaque essai l'environnement est unique puisque généré aléatoirement en début de mission. Tous les calculs sont faits a posteriori, c'est à dire une fois les missions achevées. On utilise les données enregistrées pour déterminer les valeurs de toutes les grandeurs d'intérêt (grâce aux métriques développées).

V.1.1 Mesures globales

L'analyse des indicateurs globaux ne se fait qu'à partir des missions qui ont réussi (le robot a traversé le couloir). Les échecs étant dus essentiellement à des problèmes locaux, leur analyse ne peut permettre d'expliquer le comportement du robot sur l'intégralité de la mission.

La figure V.1 représente l'ensemble des résultats pour la série d'essais avec la distance de sécurité de 0,6m. Afin de faire une analyse moyenne

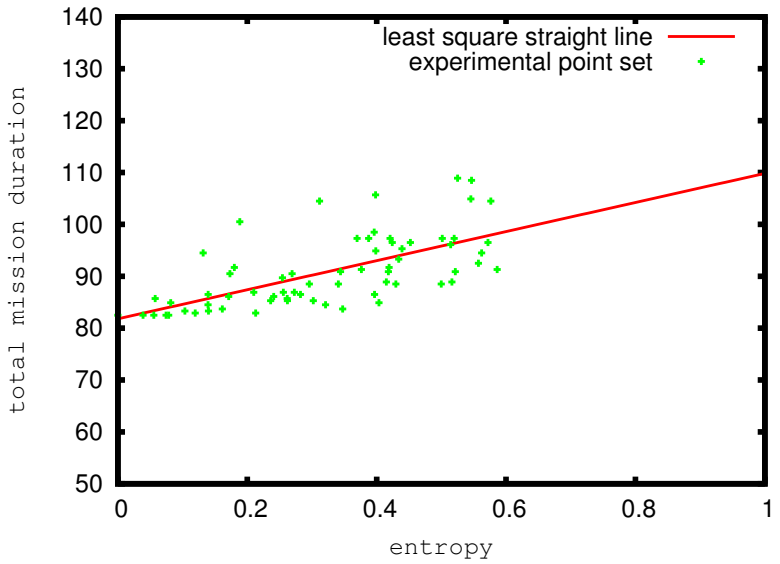


FIG. V.1 – Ensemble des points (entropie, temps de la mission) pour la série d'essais avec la distance de sécurité égale à 0,6m. Approximation par une droite des moindres carrés.

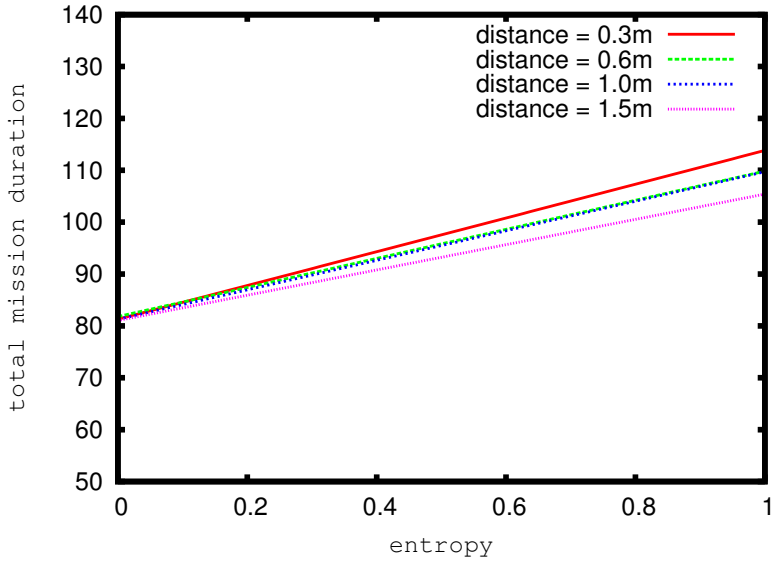


FIG. V.2 – Droites des moindres carrés pour les quatre séries d'essais.

sur l'ensemble des données, nous calculons la droite qui passe au mieux par ces points au sens des moindres carrés. La figure V.2 montre les droites de moindres carrés pour les quatre séries d'essais.

Toutes les droites coupent l'axe des ordonnées au même point. Ceci est tout à fait normal étant donné qu'une entropie nulle implique une absence d'obstacles dans le couloir. Le robot quelle que soit la valeur de la distance de sécurité traverse alors le couloir en ligne droite à la vitesse maximum.

La pente d'une droite peut être comprise comme étant la *robustesse* du robot à la complexité de l'environnement : plus la pente est grande et plus les performances du robot décroissent avec une augmentation de la complexité globale (i.e : l'entropie).

On constate que lorsque l'on diminue la distance de sécurité, la pente de la droite augmente (i.e : la robustesse diminue). Une explication possible est liée au fait que la distance de sécurité autorise plus ou moins le robot à s'approcher des obstacles. Si ce paramètre est petit le robot ira plus volontairement dans des endroits étroits. Or lorsqu'il est proche d'obstacles il diminue sa vitesse en évaluant la distance nécessaire pour s'arrêter avant collision. En moyenne un robot pour qui la distance de sécurité est petite mettra plus de temps dans un environnement encombré et donc à entropie élevée.

V.1.2 Mesures locales

La figure V.3 représente l'ensemble des points mesurés sur la série d'essais avec une distance de sécurité de 1m. Plus de 40000 points sont ainsi relevés sur l'ensemble des essais, ce qui correspond à plusieurs échantillons par seconde. Près de la moitié des points ont pour valeur $(\pi, 0, 1)$ ce qui se traduit par le robot va à la vitesse maximale quand il n'y a pas d'obstacles dans sa proximité.

Nous avons cherché quel est le comportement moyen du robot, pour cela nous avons calculé le plan qui passe au mieux par le nuage de points au sens des moindres carrés (fig. V.3).

$$h(x, y) = a_0 + a_1 * x + a_2 * y$$

est l'équation du plan. x est la valeur du plus grand angle libre, y sa position et $h(x, y)$ la vitesse normalisée.

a_2 nous renseigne sur le comportement symétrique du robot. Si ce coefficient est non nul, alors le robot ne se comporte pas de la même façon selon que les obstacles sont à sa gauche ou à sa droite.

a_1 peut être compris comme la sensibilité locale du robot à l'environnement (i.e : la robustesse). Si l'on fait varier la taille de la zone dans laquelle

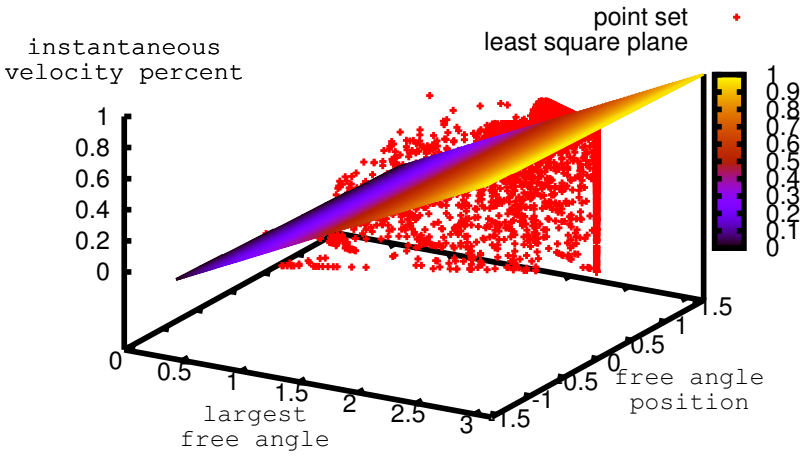


FIG. V.3 – Ensemble des points (plus grand angle libre, position de l'angle, vitesse normalisée). La distance de sécurité est égale à 1m et la zone d'intérêt est un disque de 1m de rayon. Approximation par un plan des moindres carrés.

on prend en compte les obstacles, ce paramètre change de valeur. La figure V.4 montre les relevés pour une distance de sécurité de 0,6m. Les points peuvent être approximés par une courbe de la forme :

$$f(x) = a + \frac{b}{x - c}$$

Si la distance de sécurité était considérée comme une barrière infranchissable alors c serait égal à la valeur de cette distance. Il faut voir les asymptotes verticale de la manière suivante : lorsque le rayon de la zone d'intérêt est égal à c, alors il faut peu d'obstacles pour que le robot soit bloqué (la vitesse instantanée décroît très vite vers zéro dès que la complexité locale augmente).

La figure V.5 montre les courbes obtenues pour les quatre séries d'essais. On constate que lorsque la taille de la zone d'intérêt est grande, a_1 est comparable pour les quatre séries. Le robot a alors la même sensibilité vis-à-vis de l'environnement. En revanche lorsque l'on diminue la taille de cette zone, on voit une nette différence sur les valeurs de a_1.

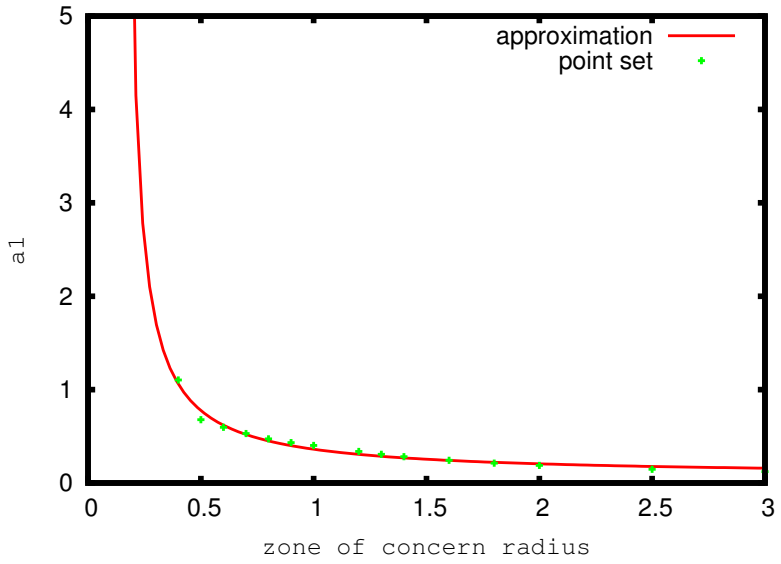


FIG. V.4 – Variation de a_1 en fonction du rayon de la zone dans laquelle les obstacles sont comptabilisés. Approximation par une courbe.

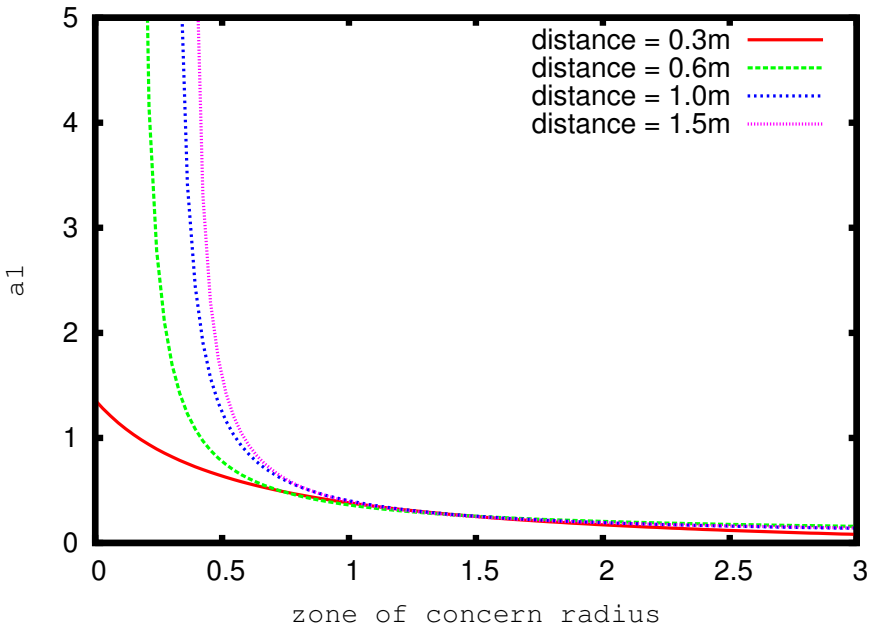


FIG. V.5 – Courbes approximant a_1 pour les quatre séries.

V.1.3 Confrontation des deux analyses

Pour les petites valeurs de la distance de sécurité la vitesse instantanée du robot ne décroît pas trop vite avec l'augmentation de la complexité locale. Or nous avons vu que au contraire la vitesse moyenne diminue (car le temps de traversé augmente). La conclusion est que lorsque la distance de sécurité augmente, le robot s'éloigne plus des obstacles et donc sa vitesse instantanée reste en moyenne plus élevée. Il pourrait être intéressant de voir quel compromis offre à la fois de la robustesse localement et globalement.

V.2 Second scénario : exploration du laboratoire

V.2.1 Les System Maps

Dans le scénario précédent nous faisions une analyse moyenne grâce à la méthode des moindres carrés. Ceci est envisageable pour trouver une fonction entre deux ou trois paramètres mais il devient difficile de trouver des fonctions d'approximation lorsque l'espace des paramètres devient plus grand. La méthode qui a été retenue est une analyse par les "system maps". Les "system maps" sont basées sur des réseaux bayésien dynamiques. Ils permettent d'apprendre les liens qui existent entre différentes métriques (voir la section III.4).

D'une manière schématique il faut voir les system maps comme un réseaux de relations entre différentes métriques. Chaque métrique et chaque indicateur est normalisé afin que toutes les valeurs soient comprises dans l'intervalle $[0, 1]$. Ceci permet de considérer chaque paramètre avec le même poids. Le réseau est initialisé, puis on fait un apprentissage hors ligne avec les valeurs collectées. A la fin de l'apprentissage, le poids des arcs dans le réseau est représentatif de l'influence d'une métrique sur une autre.

Le poids d'un arc entre une métrique sur les conditions externes et un indicateur de performance doit être compris comme une mesure de la robustesse de cette performance vis-à-vis de cette métrique. Un arc entre deux métriques ou entre deux indicateurs de performances montre la corrélation entre ces deux éléments.

On peut résumer l'intérêt des system maps avec le tableau ci-dessous. On cherche en fait les relations entre les différentes métriques, soit à remplir les cases du tableau :

	mouvement	vitesse	complexité	information
mouvement				
vitesse				
complexité				
information				

Ceci peut également être vu par l'équation suivante :

$$M_i = \mu_i + \sum_j w_j.M_j$$

Où M_i est une métrique que l'on cherche à exprimer en fonction des métriques M_j. Une system map permet de déterminer les w_j. Par contre nous n'avons pas accès aux μ_i.

V.2.2 Analyse des données

Deux séries de tests ont été conduites avec des valeurs différentes pour la distance de sécurité pour l'algorithme local d'évitement d'obstacles. Les valeurs ont étés fixées successivement à 0,7m puis à 1,5m. Chaque série comporte 25 simulations.

Les résultats pour une system map sont représentés dans une matrice antisymétrique. Les tableaux V.1 et V.2 montrent les valeurs des deux system map obtenues (une par série de simulations). Plus les valeurs sont grandes et plus l'interaction entre deux métriques est marquée. La diagonale ne nous intéresse pas (il s'agit des μ_i).

Pour rappel, la métrique "mouvement" est une variable binaire qui vaut 1 quand le robot n'est pas bloqué, la métrique "vitesse" est la vitesse instantanée normalisée.

Si l'on reprend le tableau V.1 on peut représenter la system map correspondante grâce à un graphe (fig. V.6).

Partant du graphe, exprimons l'information en fonction du mouvement :

$$M_i = \mu_i + \sum_j w_j.M_j$$

$$\begin{aligned} \text{Information} &= \mu_{Info} - 0,3 * \text{Compléxité} - 0,03 * \text{Vitesse} + 0,02 * \text{Mouvement} \\ &= \mu_{Info} - 0,15 * \text{Mouvement} \end{aligned}$$

On constate que l'interaction entre les métriques "mouvement" et "vitesse" décroît lorsque la distance de sécurité augmente. En effet avec une

moyenne				
	mouvement	vitesse	complexité	information
mouvement	0	0.8442	-0.0172	0.0222
vitesse	0	0	0.2224	-0.0324
complexité	0	0	0	-0.3072
information	0	0	0	0

variance				
	mouvement	vitesse	complexité	information
mouvement	0	0.0024	0.0191	0.0151
vitesse	0	0	0.0100	0.0055
complexité	0	0	0	0.0914
information	0	0	0	0

Tab. V.1 – Résultats pour la distance de sécurité de 0.7m.

grande distance de sécurité, le robot est plus souvent bloqué dans des zones exiguës. En revanche comme il s'engage moins souvent dans de telles zones, l'interaction avec les métriques de complexité diminue. Les interactions avec la quantité d'information décroissent avec l'augmentation de la distance de sécurité, les obstacles étant vus de plus loin.

On note que l'interaction entre la complexité et l'information est très dépendante de l'environnement, la variance étant très grande. De plus on voit que plus il y a d'obstacles (grande complexité) et plus il y a possibilité de gagner de l'information (d'où une décroissance dans la métrique). Cependant cette valeur est très incertaine et beaucoup de cas de figure ne suivent pas cette tendance générale.

Des deux robots quel est celui qui est le plus performant ? Avec une distance de sécurité de $0,7$m, le robot est plus fortement influencé par l'environnement. En effet les interactions mouvement/vitesse, mouvement/complexité et vitesse/complexité sont plus fortes pour ce robot.

V.2.3 Extensions

Dans ce scénario nous avons utilisé seulement deux indicateurs de performance et deux métriques liés aux conditions externes. Nous avions commencé l'extension de cette étude expérimentale à plus de métriques, mais faute de temps nous n'avons pu la mettre en œuvre.

Nous avons vu qu'il est possible d'utiliser des métriques binaires comme celle relative au mouvement du robot. On pourrait envisager une extension

moyenne				
	mouvement	vitesse	complexité	information
mouvement	0	0.8094	-0.0727	0.0194
vitesse	0	0	0.1413	-0.0558
complexité	0	0	0	-0.3878
information	0	0	0	0

variance				
	mouvement	vitesse	complexité	information
mouvement	0	0.0036	0.0131	0.0065
vitesse	0	0	0.0133	0.0082
complexité	0	0	0	0.1066
information	0	0	0	0

Tab. V.2 – Résultats pour la distance de sécurité de 1,5m.

en incluant dans la system map des nœuds qui montreraient la présence ou l'absence de certains composants (par exemple le robot dispose t-il d'un planificateur de trajectoire), afin de voir l'influence de certaines fonctions sur les performances.

Les états internes du robot ont aussi une influence sur son comportement. Nous avions pensé à intégrer dans la system map la qualité de la localisation du robot. On peut mettre un nœud par estimateur de position, la métrique correspondante serait alors basée sur la différence entre la position estimée et la position réelle (que l'on connaît facilement avec le simulateur).

Un autre facteur important est l'intervention humaine. Dans nos scénarios nous considérons les robots comme entièrement autonomes, or on peut légitimement se demander quel est l'impact de la coopération homme/robot sur les performances du système. Pour cette étude, on peut par exemple ajouter un nœud qui représente le pourcentage de temps que l'homme consacre au robot, dans le cas où l'on fait une étude de performances globale. Dans le cas d'une étude locale le nœud pourrait être une variable binaire mise à 1 quand l'homme intervient.

Concernant l'information, nous n'avons considéré que la différence d'information entre la carte du robot et l'environnement. Nous pourrions également regarder la différence entre les segments perçus et l'environnement afin de voir si le robot perçoit correctement l'environnement. En regardant la différence entre les segments perçus et la carte du robot nous pourrions mesurer la quantité d'information gagnée de manière instantanée.

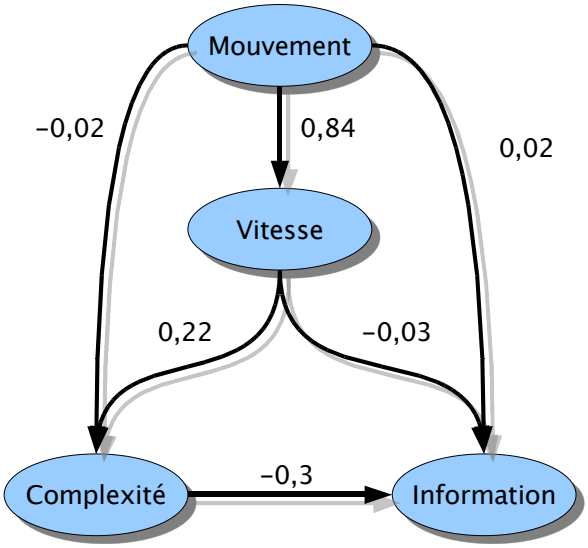


FIG. V.6 – Graphe de la system map obtenue pour la distance de sécurité de 0.7m.

V.2.4 Remarque sur la réactivité

Comme nous l'avons fait remarquer, on peut calculer la réactivité du robot lorsqu'il se trouve bloqué. La chaîne de calcul n'étant pas implémentée lors des tests précédents, nous en avons conduit de nouveau selon le même principe et avec la même configuration pour le robot. Le tableau V.3 montre les résultats statistiques pour environ 80 situations de blocage.

	Détection	Traitement	Temps de réaction
Moyenne	24.42	3.5	27.92
Variance	10.13	1.07	10.30

TAB. V.3 – Statistiques sur le temps de réaction aux situations de blocage.

Il faut prendre quelques précautions avec ces valeurs : le temps de traitement est de moins de 4 secondes en moyenne or la précision des relevés est de l'ordre de la seconde. Pour faire des mesures plus précise il nous aurait fallu prendre ces mesures à l'intérieur du simulateur. Néanmoins ici nous montrons comment quantifier la réactivité d'un robot. On constate que la variance sur le temps de détection est assez importante. Probablement qu'en

modifiant la distance parcourue qui sert de référence pour déterminer les derniers instants de blocage, on peut faire varier la variance des mesures (ici nous avons considéré que lorsque le robot déclare être bloqué, il l'était également dans les 2 derniers mètres).

Les données en elles mêmes peuvent nous renseigner sur le temps de reprise moyen. Il serait intéressant de pouvoir comparer ces valeurs à d'autres (faire le même test avec un autre robot).

V.3 Conclusion

Cette section montre l'intérêt de la simulation pour évaluer des systèmes robotiques. La simulation permet de faire rapidement des campagnes d'essais et de collecter beaucoup de données. Elle permet également de maîtriser l'environnement que se soit du point de vue d'une reconfiguration rapide de l'environnement de simulation ou du point de vue de la connaissance de l'environnement. En effet l'accès aux mesures dimensionnelles de l'environnement nous évite d'avoir recours à des moyens externes comme cela serait nécessaire sur un site réel. Nous pouvons également connaître à tout moment la position réelle du robot et la comparer à sa position estimée.

Le grand nombre de données collectées lors de ces séries d'essais nous à permis d'analyser le comportement du robot vis-à-vis des paramètres que nous avons fait varier. Ainsi nous avons pu montrer qu'il est possible établir et de mesurer le lien qui existe entre les diverses métriques. Notamment nous avons vu et quantifié pour un robot donné l'influence de la complexité de l'environnement sur les performances du robot. De tous ces résultats ont peut extraire des critères tels que la robustesse.

Deux méthodes ont servi à l'analyse, elles sont cependant semblables dans l'esprit car elles relatent de l'influence moyenne d'un paramètre sur un autre. La méthode des moindres carrés nous donne le meilleur estimateur pour approximer notre nuage de points. Les system maps quant à elles font un apprentissage incrémental d'un réseau bayésien.

Alors quel est le robot le plus autonome (ou dans notre cas quel est le meilleur réglage pour l'algorithme d'évitement d'obstacles) ? En l'état nous ne pouvons pas répondre à cette question. En revanche nous pouvons dire quel robot est le plus performant et sous quelles conditions. Pour répondre à la question de l'autonomie il nous aurait fallu définir les critères que nous avons appelés *contraintes utilisateur* et qui cadre nos exigences en matière de performance à atteindre. Ainsi que l'on privilégie les performances locales ou globales du robot, que l'on envisage des environnement très complexes ou au contraire très simples, le classement du robot le plus autonome ne

sera pas le même.

Chapitre VI

Conclusion - Perspectives

L'intitulé de ce mémoire résume de lui même les objectifs que nous nous étions fixés : définir une méthode pour l'évaluation de l'autonomie en robotique. Dans ce chapitre nous faisons le bilan de notre contribution, puis verrons quelles peuvent être les perspectives à ces travaux.

VI.1 Bilan - Contribution

Comme nous avons pu le voir, l'autonomie est une notion complexe à définir. Sa nature subjective rend son interprétation difficile et aléatoire en fonction des critères que chacun privilégie. Au cours des discussions que nous avons pu avoir avec des roboticiens, il est apparu que chacun a sa propre définition. Hélas aucune ne pouvait servir de manière pratique. La définition proposée par le NIST peut servir de référence et elle semble être raisonnable quant à son cadre. Cependant nous avons pu constater que cette définition ne rendait pas plus pratique l'évaluation de l'autonomie.

Il s'agit là de la principale difficulté de ce thème de recherche. Comment passer de la notion d'autonomie à son évaluation ?

VI.1.1 La méthode

La méthode proposée tente de répondre à la question posée ci-dessus. Tout d'abord nous avons décomposé une mission selon des termes pouvant être mesurés indépendamment les uns des autres. La partie relative aux contraintes imposées par l'utilisateur tente de prendre en compte la part subjective que l'utilisateur introduit forcément lorsqu'il juge de l'autonomie d'un système. La partie relative aux contraintes externes relate quant à elle de tout ce que l'on ne maîtrise pas directement lors d'une mission robotique mais qui influence le comportement du robot.

L'objectif de cette décomposition est de comparer ensuite le comportement du robot en fonction des contraintes externes, vis-à-vis des contraintes fixées par l'utilisateur.

D'une manière pratique nous avons proposé des métriques pour chaque composante. En s'appuyant sur la mesure des performances du robot nous avons pu évaluer son comportement en fonction de l'environnement et également de l'information qu'il possède.

Pour cela nous avons développé des métriques locales et globales spécifiques à une mission de navigation. Ces métriques ont montré qu'il est possible de faire le lien entre les performances d'un système robotique et son environnement.

Il nous semble aujourd'hui évident que l'évaluation de l'autonomie ou plus précisément des capacités d'un robot, ne peut se faire que par l'intermédiaire de l'évaluation de ses performances. Nous avons utilisé certains indicateurs de performances, mais il en existe d'autres et sur ce sujet seul l'utilisateur final peut décider de la pertinence des indicateurs en fonction du comportement qu'il cherche à évaluer. De plus nous ne prétendrons pas avoir trouvé des métriques universelles, même si nous les avons construites dans un souci de généricité. Probablement que d'autres métriques sont utilisables, mais elles restent à définir. Notamment selon le type de représentation choisi pour les données ou pour d'autres types de missions, il faudra adapter ou créer les métriques appropriées (par exemple, nos métriques relatives à l'environnement étant basées sur une représentation métrique de l'environnement, ne conviennent pas pour une représentation par graphe de connexité entre zones).

VI.1.2 L'expérimentation

Pour vérifier nos hypothèses et métriques nous avons contribué au développement et à l'implémentation d'un simulateur robotique. Nous sommes convaincu que ce genre d'outil a toute sa place dans le processus de dévelop-

pement et d'évaluation de systèmes complexes tels que des robots. Premièrement nous avons pu constater les avantages que cela apporte en termes de rapidité d'expérimentation, de maîtrise environnementale et de facilité pour la collecte de données. La simulation offre également la possibilité lors de la conception de nouvelles fonctionnalités, de pouvoir les tester au plus tôt, et ce sans danger pour le robot. Mais il ne faudrait pas se contenter de cette évaluation. Pour un robot nous savons que le monde réel avec son lot d'incertitude et d'aléas est une source d'ennuis. De plus certaines fonctionnalités restent difficiles à simuler comme par exemple tout ce qui concerne la vision par caméras. Si la simulation donne une idée du comportement d'un robot, il est indispensable de le tester également en conditions réelles.

VI.1.3 L'analyse

La quantité de données collectées pendant les simulations devient vite impressionnante. Pour traiter cette masse nous nous sommes servi dans un premier temps d'une analyse par moindres carrés. Ce qui nous a permis de mettre en évidence (et ce de façon mesurable) l'influence de la complexité de l'environnement sur les performances du robot. De cela nous avons pu extraire la robustesse du robot, terme très usité dans le domaine mais peu ou pas mis en évidence de façon rigoureuse.

Les system maps que nous avons utilisées par la suite en collaboration avec l'université de Sydney ont permis d'étendre l'analyse à plus de métriques et d'indicateurs. Il s'agit là sans doute d'une voie prometteuse pour décortiquer toutes les données collectées et faire une analyse multidimensionnelle. En calculant les liens et l'influence qu'ont les métriques entre elles, les system maps autorisent l'extraction de la robustesse d'un indicateur de performance vis-à-vis des paramètres externes.

Nous avons montré grâce à cet outil qu'il est possible d'utiliser des métriques différentes lors de l'analyse. Ainsi nous pouvons intégrer des mesures continues (par exemple la vitesse instantanée entre $[0, 1]$) ou des mesures binaires (état du robot). Ceci ouvre de nombreuses perspectives quant au nombre et aux types de métriques utilisées, puisque nous disposons d'une méthode pour les analyser.

VI.2 Perspectives

VI.2.1 Extension

Dans la section V.2.3 nous avons évoqué les développements entamés pour étendre notre analyse. D'une part en ajoutant d'autre métriques et

indicateurs tels que ceux concernant les estimateurs de position. D'autre part en ajoutant un mode de contrôle par un opérateur humain.

L'ajout de métriques devrait permettre de comprendre de manière plus fine ce qui se passe pour le robot. Quelles sont les situations qui lui posent problème et quel est le composant du robot qui manque de robustesse dans cette situation ? Inclure des indicateurs de performances concernant des parties ou sous-parties du robot peut aider à répondre à ce type de question. Dans ce cas nous disposerions alors d'un outil efficace pour évaluer le robot, corriger et améliorer ses fonctionnalités et capacités.

L'autre point important que nous n'avons pas eu le temps de tester est l'interaction du couple homme-robot. Certains ont déjà montré que l'homme influe sur les performances d'un robot. En suivant cette idée il nous semble intéressant d'intégrer ce facteur dans l'évaluation du robot. En effet les systèmes robotiques sont rarement conçus pour évoluer en complète autonomie. L'homme bien souvent assiste la machine dont le but est d'accomplir une tâche qui lui a été désignée. Il s'agit là d'une voie qui devrait être étudiée et analysée. Ce type d'analyse s'intègrerait tout à fait dans le concept de l'autonomie ajustable.

VI.2.2 Faire du benchmarking

Le benchmark est un bon outil dans l'évaluation d'un système quel qu'il soit. La méthode proposée peut servir à la construction de benchmark, c'est ce qui a été fait lors de nos campagnes de simulations. Elle peut certainement être utilisée également pour des tests en conditions réelles.

A notre connaissance la Délégation Générale de L'Armement (DGA) construit un terrain d'essai pour tester les robots qu'elle conçoit et développe. Construire ce type d'environnement revient à concevoir un benchmark puisque tout les robots seront testés selon les mêmes conditions. Pour aller plus loin dans le concept du benchmark, il serait envisageable de disposer de métriques rendant compte des difficultés du terrain de manière quantitative. Connaître de façon précise et quantitative l'ensemble du champ de test permet alors d'analyser rigoureusement le comportement du robot ainsi que ses capacités effectives.

VI.2.3 Et l'autonomie dans tout ça ?

Notre but premier est la définition de l'autonomie. N'ayant pas fixé de critère de performances à atteindre lors de nos tests, nous n'avons pas terminé la boucle pour décider de l'autonomie ou de la non-autonomie du robot.

Pour définir l'autonomie par rapport aux performances du robot, nous avons donné des pistes avec notre analogie de la section III.1.3 (page 54). Ainsi si l'on considère les niveaux d'autonomie comme des diplômes donnés en fonctions des performances du robot, il nous faut construire les examens de passages. Ces examens pourraient être tout simplement construits à partir de benchmarks. Selon le niveau de performance du robot on lui attribuera ou pas le niveau d'autonomie donné.

Il nous semble que cette approche pragmatique de la définition des niveaux d'autonomie est plus facile à utiliser et moins aléatoire que l'approche qui consiste à dire qu'un robot disposant de plus de fonctionnalités est forcement plus autonome. Les critères quantitatifs sont là pour trancher sur la question sans que l'interprétation subjective de l'homme n'entre en ligne de compte.

Références bibliographiques

[Albus, 2002a] Albus, J. S. (2002a). 4D/RCS : A Reference Model Architecture for Intelligent Unmanned Ground Vehicles. In *Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, Orlando, FL.

[Albus, 2002b] Albus, J. S. (2002b). Metrics and Performance Measures for Intelligent Unmanned Ground Vehicles. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[ALFUS, 2005] ALFUS (2005). http ://www.isd.mel.nist.gov/projects/autonomy_levels.

[Balakirsky and Messina, 2002] Balakirsky, S. and Messina, E. (2002). A Simulation Framework for Evaluating Mobile Robots. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Barber and Martin, 1999] Barber, K. S. and Martin, C. E. (1999). Agent Autonomy : Specification, Measurement, and Dynamic Adjustement. In *Proceedings of the Autonomy Control Software Workshop*, pages 8–15, Seatle, WA.

[Borenstein and Feng, 1994] Borenstein, J. and Feng, L. (1994). UMBmark - A Method For Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robots.

[Bradshawa et al., 2004] Bradshawa, J. M., Feltovich, P. J., Jung, H., Kulkarni, S., Taysom, W., and Uszok, A. (2004). *Dimensions of Adjustable Autonomy and Mixed-Initiative Interaction*, pages 17–39. Springer-Verlag.

[Brainov and Hexmoor, 2001] Brainov, S. and Hexmoor, H. (2001). Quantifying Relative Autonomy in Multiagent Interaction. In *IJCAI-01 workshop on Autonomy, Delegation, and Control : Interacting with Autonomous Agents*, pages 27–35, Seattle, WA.

[Carlson and Murphy, 2003] Carlson, J. and Murphy, R. R. (2003). Reliability Analysis of Mobile Robots. In *International Conference on Robotics and Automation*, Taipei, Taiwan. IEEE.

[Cleary et al., 2000] Cleary, M. E., Abramson, M., Adams, M. B., and Kolitz, S. (2000). Metrics for Embedded Collaborative Intelligent Systems. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Clough, 2002] Clough, B. T. (2002). Metrics, Schmetrics ! How The Heck Do You Determine A UAV's Autonomy Anyway ? In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Cooper and Herskovits, 1992] Cooper, G. F. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. *Mach. Learn.*, 9(4) :309–347.

[Crandall and Goodrich, 2003] Crandall, J. W. and Goodrich, M. A. (2003). Measuring the Intelligence of a Robot and its Interface. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Dalgalarrondo, 2001] Dalgalarrondo, A. (2001). *Integration de la fonction perception dans une architecture de contrôle de robot mobile autonome*. PhD thesis, Université Paris-Sud, centre d'Orsay.

[Dalgalarrondo, 2003] Dalgalarrondo, A. (2003). A propos de l'autonomie des robots. Technical Report CTA/02 350 108/RIEX/807, DGA/CTA.

[Dean and Kanazawa, 1990] Dean, T. and Kanazawa, K. (1990). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3) :142–150.

[Deplanques, 1996] Deplanques, P. (1996). *Vers le Test de l'Autonomie des Robots : une Ontologie de la Robotique*. PhD thesis, Académie de Montpellier, Université de Montpelier II, Sciences et Techniques du Languedoc.

[Friedlander et al., 2000] Friedlander, D., Phoha, S., and Ray, A. (2000). Domain Independent Measures of Intelligent Control. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Gat, 1995] Gat, E. (1995). Towards Principled Experimental Study of Autonomous Mobile Robots. In *Experimental Robotics IV, the 4th International Symposium*, chapter 9, pages 390–401. Springer.

[Goodrich et al., 2001] Goodrich, M., Olsen, D., Crandall, J., and Palmer, T. (2001). Experiments in Adjustable Autonomy. In *Seventeenth International Joint Conference on Artificial Intelligence*, Seatle, USA.

[Heckerman, 1996] Heckerman, D. (1996). A Tutorial on Learning With Bayesian Networks. Technical Report MSR-TR-95-06, The Microsoft Corporation.

[Held et al., 2006] Held, J., Lampe, A., and Chatila, R. (2006). Linking Mobile Robot Performances With the Environment Using System Maps. In *International Conference on Intelligent Robots and Systems*.

[Held and Sukkarieh, 2005] Held, J. and Sukkarieh, S. (2005). System Maps : A Data Driven Model for Systems of Systems. In *Conference Proceedings Infotech@Aerospace, AIAA 2005-6989*.

[Hexmoor and Vaughn, 2002] Hexmoor, H. and Vaughn, J. T. (2002). Computational adjustable autonomy for NASA Personal Satellite Assistants. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 21–26. ACM Press.

[Huang, 2004] Huang, H.-M. (2004). Autonomy Levels for Unmanned Systems (ALFUS) Framework - Volume I : Terminology. Technical Report Version 1.1, NIST.

[Huang et al., 2004a] Huang, H.-M., Albus, J., Messina, E., Wade, R., and English, W. (2004a). Specifying Autonomy Levels for Unmanned Systems : Interim Report. In *Proceedings of the 2004 SPIE Defense and Security Symposium*.

[Huang et al., 2003a] Huang, H.-M., Messina, E., and Albus, J. (2003a). Autonomy Level Specification for Intelligent Autonomous Vehicles : Interim Progress Report. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Huang et al., 2003b] Huang, H.-M., Messina, E., and Albus, J. (2003b). Towards a Generic Model for Autonomy Level for Umanned Systems (ALFUS). In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Huang et al., 2004b] Huang, H.-M., Messina, E., Wade, R., English, R., Novak, B., and Albus, J. (2004b). Autonomy Measures For Robots. In *Proceedings of IMECE : International Mechanical Engineering Congress*.

[Huang et al., 2005] Huang, H.-M., Pavek, K., Albus, J., and Messina, E. (2005). Autonomy Levels for Unmanned Systems (ALFUS) Framework : An Update. In *Proceedings of the 2005 SPIE Defense and Security Symposium*, Orlando, Florida.

[Jacoff et al., 2001] Jacoff, A., Messina, E., and Evans, J. (2001). Experiences in Depoying Test Arenas for Autonomous Mobile Robots. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Joyeux, 2004] Joyeux, S. (2004). Architecture distribuée de simulation multi-robots. Stage de DEA de système informatique. http ://www.laas.fr/∼sjoyeux/doc/sim-architecture.pdf.

[Kamsickas and Ward, 2003] Kamsickas, G. M. and Ward, J. N. (2003). Developing UGVs For The FCS Program. In *Proceedings of SPIE*, volume 5083, Orlando, USA.

[Lussier et al., 2005] Lussier, B., Lampe, A., Chatila, R., Guiochet, J., Ingrand, F., Killijian, M.-O., and Powell, D. (2005). Fault Tolerance in Autonomous Systems : How and How Much ? In *4th IARP/IEEE-RAS Joint Workshop on Technical Challenge for Dependable Robots in Human Environments*, Nagoya, Japan.

[MacKenzie and Arkin, 1998] MacKenzie, D. and Arkin, R. (1998). Evaluating the Usability of Robot Programming Toolsets. *The International Journal of Robotics Research*, 17(4) :381–401.

[Madhavan and Messina, 2003] Madhavan, R. and Messina, E. (2003). Quantifying Uncertainty Towards Information-Centric Unmanned Navigation. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Martin et al., 2001] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

[Matthies et al., 1995] Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R., and Litwin, T. (1995). Mars Microrover Navigation : Performance Evaluation and Enhancement. *Autonomous Robots Journal*, 2(4) :291–311.

[Messina et al., 2005] Messina, E., Jacoff, A., Scholtz, J., Schlenoff, C., Huang, H.-M., Lytle, A., and Blitch, J. (2005). Statement Of Requirements For Urban Search And Rescue Robot Performance Standards. Technical Report Preliminary Version, Department of Homeland Security Science and Technology Directorate and National Institute of Standards and Technology.

[Messina et al., 2001] Messina, E., Meystel, A., and Reeker, L. (2001). White Paper : Measuring Performance and Intelligence of Intelligent Systems. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Minguez et al., 2004] Minguez, J., Osuna, J., and Montano, L. (2004). A "Divide and Conquer" Strategy based on Situations to achieve Reactive Collision Avoidance in Troublesome Scenarios. In *ICRA*, New Orlean, USA.

[Proud et al., 2003] Proud, R. W., Hart, J. J., and Mrozinski, R. B. (2003). Methods for Determining the Level of Autonomy to Design into a Human

Spaceflight Vehicle : A Function Specific Approach. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Robotcup, 2005] Robotcup (2005). http ://www.robocup2005.org/robo-rescue/default.aspx.

[Schipani, 2003] Schipani, S. P. (2003). An Evaluation of Operator Workload, During Partially-Autonomous Vehicle Operations. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Scholtz et al., 2003] Scholtz, J., Antonishek, B., and Young, J. (2003). Evaluation of Operator Interventions in Autonomous Off-road Driving. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Yanco, 2001] Yanco, H. A. (2001). Designing Metrics for Comparing the Performance of Robotics Systems in Robot Competitions. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Yanco, 2002] Yanco, H. A. (2002). Evaluating the Performance of Assistive Robotic Systems. In *Proceeding of the Performance Metrics for Intelligent System Workshop*.

[Yriarte, 1997] Yriarte, L. (1997). *Modélisation et Validation de Robots Autonomes : une Méthodologie des Outils*. PhD thesis, Académie de Montpellier, Université de Montpelier II, Sciences et Techniques du Languedoc.

Annexe A

Les données dans le simulateur

A.1 Collecte des données dans le simulateur

Pour comprendre comment la collecte des données est faite, il faut faire un bref rappel sur les structures utilisées par l'outil GenoM.

Les échanges de données entre modules fonctionnels se font par l'intermédiaire de posters. Les posters sont composés de structures C et peuvent être partagés en lecture et en écriture. La simulation permet d'enregistrer toutes les données écrites dans un poster sous format binaire. Un outil nommé *logviewer* permet ensuite de lire les données ainsi collectées.

Exemple de sortie de logviewer, la commande info permet de lister le nombre et le type de posters enregistrés :

```
# logviewer info bridge.log
Found 7 streams
Stream 0: robotTruthPos [p6d]
        39441 samples
      From  316336:37:44.471 to  316336:50:53.251 [ 0:13:08.780]
Stream 1: robotPos [move_status]
        7889 samples
      From  316336:37:44.471 to  316336:50:53.171 [ 0:13:08.700]
Stream 2: laserSick [laserScan]
        7889 samples
      From  316336:37:44.551 to  316336:50:53.251 [ 0:13:08.700]
Stream 3: laserSick_back [laserScan]
```

```
        7889 samples
      From  316336:37:44.551 to  316336:50:53.251 [ 0:13:08.700]
Stream 4: seglocMap [SEGLOC_MAP]
        22 samples
      From  316336:37:47.450 to  316336:49:15.181 [ 0:11:27.731]
Stream 5: seglocSegs [SEGLOC_ACQ_SEGS]
        3880 samples
      From  316336:37:47.451 to  316336:50:53.208 [ 0:13:05.757]
Stream 6: robotPosCmd [move_cmd]
        1608 samples
      From  316336:37:49.953 to  316336:50:52.236 [ 0:13:02.283]
```

Dans cet exemple on voit que 7 types de posters ont été enregistrés. Prenons le premier flux de données pour détailler la sortie :

```
Stream 0: robotTruthPos [p6d]
        39441 samples
      From  316336:37:44.471 to  316336:50:53.251 [ 0:13:08.780]
```

Le nom du poster est *robotTruthPos*, le nom de la structure C associée est *p6d*. 39441 enregistrements dans le poster ont été collectés (nous avons donc 39441 enregistrements de la structure *p6d*). L'enregistrement a débuté au temps d'horloge `316336:37:44.471` et s'est arrêté au temps `316336:50:53.251`. La duré d'enregistrement correspond à 13 minutes et 8,780 secondes.

A.2 Les types de données collectées et leur exploitation

Les posters enregistrés sont les suivants :

– robotTruthPos : position du robot dans le simulateur. C'est la position exacte du robot.

– robotPos : position estimée du robot. C'est la position que le robot obtient à partir de ses différents estimateurs de position (odométrie, localisation sur segments).

– laserSick : information retournée par la nappe laser avant. C'est elle qui sert au robot pour estimer les obstacles et se localiser.

– laserSick_back : information retournée par la nappe laser arrière. Elle ne sert qu'au calcul de la surface de visibilité et le robot n'a pas accès à ces données.

– seglocMap : carte de segments dans la mémoire du robot. A chaque nouvelle acquisition de segments par le robot, ils sont ajoutés dans cette structure. L'acquisition de segments fait l'objet d'une requête particulière.

– seglocSegs : segments perçus par le robot. Ils ne sont pas ajoutés à la carte en continu. Le robot se sert de ces informations pour se localiser au cours de ses déplacements.

– robotPosCmd : commande en vitesse du robot. Il s'agit des ordres de commande envoyés au contrôleur de vitesse.

Il faut bien noter que la collecte d'informations ne se fait pas sur des intervalles de temps réguliers. On enregistre les données uniquement lorsqu'une écriture dans un poster intervient. Même de cette manière la quantité de données est importante, par exemple la taille du fichier de cette simulation d'environ 13 minutes dépasse les 87Mo. Rien d'étonnant car par exemple pour une nappe laser chaque enregistrement contient 361 données de type double (codées sur 8 octets) pour les distances mesurées, plus quelques informations annexes.

Le fait d'avoir des enregistrements de données qui ne se font pas aux mêmes instants demande un effort particulier pour remettre toutes ces informations en correspondance dans le temps. Dans cette optique, la position du robot dans le simulateur sert de référence à chaque fois, car c'est ce poster qui est initialisé en premier et qui dispose de la plus grande fréquence d'échantillonnage.

La commande :

```
logviewer show bridge.log 0 -r 30435:30439
39441 samples in stream, will fetch 4
read 4 samples
logtime .pos[0] .pos[1] .pos[2] .rot[0] .rot[1] .rot[2]
 316336:47:53.151
  9.958804037491211    -12.15911822243338     0.001855707072961482
 -0.0001397777159406103 -9.071901252907017e-05  0.9947518304009385
 316336:47:53.171
  9.958807397492839    -12.15911822242684    -0.001855722275078978
  0.0001420110119221372  9.231219422682065e-05  0.9947518306781945
 316336:47:53.191
  9.958804037175693    -12.15911822241906    -0.001855708511276019
 -0.0001398049616595811 -9.073500501598362e-05  0.9947518303773211
 316336:47:53.211
  9.958807397921159    -12.15911822241242    -0.001855723858639732
  0.0001420461214317897  9.233675521235324e-05  0.9947518306557815
```

montre les données enregistrées dans le poster *robotTruthPos* entre les index 30435 et 30439. Chaque enregistrement comprend le temps correspondant à l'instant de la capture, puis la position du robot sous forme de six coordonnées. En fait dans le cas d'un robot d'intérieur qui n'utilise que des données 2D, seules les valeurs pos[0] (coordonné X), pos[1] (coordonné Y) et rot[2] (rotation) sont utiles. D'ailleurs on voit que les valeurs de pos[3], rot[0] et rot[1] sont quasi nulles (elles ne le sont pas à cause de la souplesse dans les liaisons entre solides et les approximations dans le simulateur).

Annexe B

Calcul de la grille d'occupation

B.1 Données d'une carte de segments

La commande :

```
logviewer show bridge.0.log 4 -r 5:5
22 samples in stream
Will fetch 0 samples
Read 1 samples
 316336:39:40.381
 pos x 0.106909 y 0.00172136 th -0.00858699
seg 0 0.0214124 0.58892 -0.242217 0.591728
var 0 0.589114 -1.58145 8.90003e-05 0.00395353 0.000471913
seg 1 11.0156 0.0209967 10.9497 0.699088
var 1 10.9659 -3.0447 0.00114452 0.00117773 -0.00110537
seg 2 3.79954 -0.453477 3.80232 0.200424
var 2 3.80143 3.13735 4.92519e-05 0.00063152 -7.24795e-05
seg 3 0.493857 1.1665 0.0903532 0.549843
var 3 0.225456 -0.579423 0.000274064 0.000367493 -0.000299208
seg 4 11.5648 -1.92332 11.5678 -0.128336
var 4 11.568 3.13993 0.000180623 0.000109101 -0.00010565

...

seg 197 -14.5704 -24.296 -14.3731 -24.7927
```

```
var 197 22.5121 0.378203 0.0126786 4.1375e-05 0.000723981
seg 198 -12.0448 -24.705 -12.4001 -24.6059
var 198 -27.0328 -1.84281 0.350062 0.0148162 0.0719238
```

donne la cinquième carte de segments enregistrée. Après la ligne qui donne le temps d'horloge de l'enregistrement on voit une ligne qui donne la position estimé du capteur dans le repère du robot. Ensuite viennent les coordonnées des segments. Chaque segment est décrit par deux lignes. La première donne les coordonnées cartésiennes des extrémités du segment (X_1, Y_1, X_2, Y_2). La seconde donne les coordonnées polaires statistiques de la droite porteuse (rayon moyen, angle moyen, variance sur le rayon, variance sur l'angle et covariance). La figure B.1 montre la représentation de ces grandeurs pour un segment.

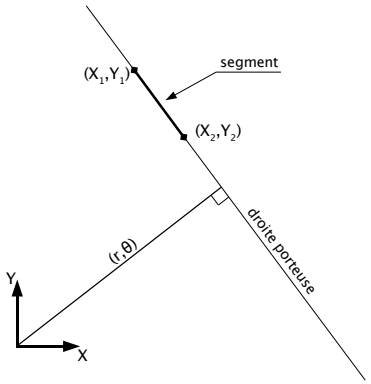


FIG. B.1 – Coordonnées des segments de la carte. Droite porteuse du segment en coordonnées polaires.

La distribution de probabilités pour une droite porteuse est exprimée par :

$$p = \frac{\mathbf{e}^{\left(\frac{-1}{2(v_\rho v_\theta - v_{\rho\theta}^2)}\left(v_\theta(\rho_0-\rho)^2+v_\rho(\theta_0-\theta)^2-2\left(v_{\rho\theta}(\rho_0-\rho)(\theta_0-\theta)\right)\right)\right)}}{2\Pi\sqrt{v_\rho v_\theta - v_{\rho\theta}^2}}$$

Avec ρ_0 le rayon moyen, θ_0 l'angle moyen, v_ρ la variance sur le rayon, v_θ la variance sur l'angle et $v_{\rho\theta}$ la covariance.

B.2 Calcul de la densité de probabilité : obtention de la grille d'occupation

B.2.1 Méthode utilisée

Pour un segment donné, nous avons les coordonnées moyennes de sa droite porteuse. Autour de ces coordonnées nous plaçons une grille polaire discrète et nous calculons la probabilité que la droite passe par chaque cellule de la grille. Ce calcul se fait par intégration de l'expression de la densité ci dessus sur la surface d'une cellule. Une hypothèse pratique permet de faire cette intégration plus simplement.

Si l'on suppose que la covariance est nulle, l'expression de la densité de probabilité devient :

$$\begin{aligned} p &= \frac{\mathbf{e}^{\left(\frac{-1}{2(v_\rho v_\theta)}\left(v_\theta(\rho_0-\rho)^2+v_\rho(\theta_0-\theta)^2\right)\right)}}{2\Pi\sqrt{v_\rho v_\theta}} \\ &= \frac{1}{2\Pi\sqrt{v_\rho v_\theta}} \cdot \mathbf{e}^{\frac{-(\rho_0-\rho)^2}{2v_\rho}} \cdot \mathbf{e}^{\frac{-(\theta_0-\theta)^2}{2v_\theta}} \end{aligned}$$

Ceci permet alors de faire deux intégrales simples au lieu d'une intégrale double.

Le schémas ci dessous montre les étapes pour le remplissage de la grille d'occupation cartésienne à partir d'un segment.

Algorithme 1 Intégration :

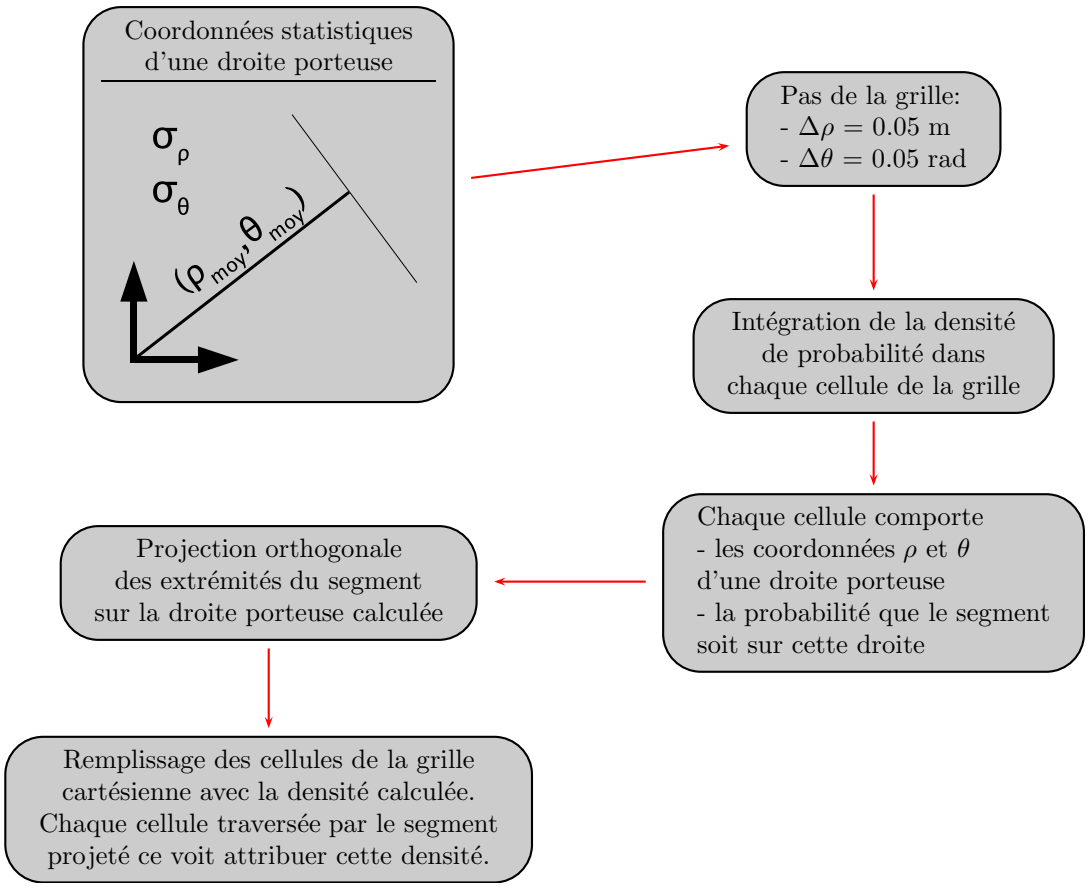
$proba_{min} = 0.01$
tantque $proba > proba_{min}$ **faire**
  pour $\rho = \rho_{moy}$ jusqu'à $\rho_{moy} + 3\sigma_\rho$ **faire**
    pour $\theta = \theta_{moy}$ jusqu'à $\theta_{moy} + 3\sigma_\theta$ **faire**

$$proba(\rho,\theta) = \frac{1}{2\Pi\sqrt{v_\rho v_\theta}} \int_{\rho-\frac{d\rho}{2}}^{\rho+\frac{d\rho}{2}} \mathbf{e}^{\frac{-(\rho_{moy}-\rho_i)^2}{2v_\rho}} d\rho_i \int_{\theta-\frac{d\theta}{2}}^{\theta+\frac{d\theta}{2}} \mathbf{e}^{\frac{-(\theta_{moy}-\theta_i)^2}{2v_\theta}} d\theta_i$$

      $\rho = \rho + \Delta\rho$
      $theta = \theta + \Delta\theta$
    fin pour
  fin pour
  pour $\rho = \rho_{moy}$ jusqu'à $\rho_{moy} - 3\sigma_\rho$ **faire**
    pour $\theta = \theta_{moy}$ jusqu'à $\theta_{moy}3\sigma_\theta$ **faire**

$$proba(\rho,\theta) = \frac{1}{2\Pi\sqrt{v_\rho v_\theta}} \int_{\rho-\frac{d\rho}{2}}^{\rho+\frac{d\rho}{2}} \mathbf{e}^{\frac{-(\rho_{moy}-\rho_i)^2}{2v_\rho}} d\rho_i \int_{\theta-\frac{d\theta}{2}}^{\theta+\frac{d\theta}{2}} \mathbf{e}^{\frac{-(\theta_{moy}-\theta_i)^2}{2v_\theta}} d\theta_i$$

      $\rho = \rho - \Delta\rho$
      $theta = \theta - \Delta\theta$
    fin pour
  fin pour
fin tantque



Le code perl associé à l'intégration et au remplissage de la grille est le suivant :

```
sub compute_density{
my ($class, $map, $varM, $densLimit) = @_;

my $dr = $data->{dr};
my $dth = $data->{dth};
print "compute density for map $map\n";
for (my $i = 0; $i < $data->{$map}->{nbPt}; $i++)
  {
    my $x1 = $data->{$map}->{segs}[$i]->{x1};
    my $y1 = $data->{$map}->{segs}[$i]->{y1};
    my $x2 = $data->{$map}->{segs}[$i]->{x2};
    my $y2 = $data->{$map}->{segs}[$i]->{y2};
    my $r  = $data->{$map}->{var}[$i]->{r};
    my $o  = $data->{$map}->{var}[$i]->{o};
    my ($vr, $vo, $vro);
```

```
if ($map eq 'envMap')
  {
    $vr  = 0.0000001;
    $vo  = 0.0000001;
    $vro = 0.000000;
  }
else
  {
    $vr  = $data->{$map}->{var}[$i]->{vr};
    $vo  = $data->{$map}->{var}[$i]->{vo};
    $vro = $data->{$map}->{var}[$i]->{vro};
  }

# start to compute and set density map
my $dens = 1;
my $pas = 30;
for (my $rho = $r; $rho <= $r + $varM*sqrt($vr);
   $rho += $dr/$pas)
  {
    for (my $th = $o; $th <= $o + $varM*sqrt($vo);
       $th += $dth/$pas)
      {
        my $dens = calculUtils::dens_ro($dr, $dth,
            $rho, $th, $r, $o, $vr, $vo,$vro);
        my ($X1, $Y1, $X2, $Y2) = calculUtils::
           compute_line_sup($rho, $th, $x1, $y1,
           $x2, $y2);

        if ($dens > 1) {$dens = 1;}
        if ($dens > $densLimit)
          {
            my ($Xi1, $Yi1) = $data->rob_to_img(
               $X1, $Y1);
            my ($Xi2, $Yi2) = $data->rob_to_img(
               $X2, $Y2);
            if ($map eq 'envMap')
              {
                $data->set_density($map, $dens,
                   $Xi1, $Yi1, $Xi2, $Yi2);
              }
            else
```

```
                {
                  $data->set_density('missionMap',
                     $dens, $Xi1, $Yi1, $Xi2, $Yi2);
                }
            }
          else
            {
              $th = $o + $varM*sqrt($vo);
              $dens = 1;
            }
        }
    }
  $dens = 1;
  for (my $rho = $r ; $rho >= $r - $varM*sqrt($vr);
     $rho -= $dr/$pas)
    {
      for (my $th = $o ; $th >= $o - $varM*sqrt($vo)
         ;$th -= $dth/$pas)
        {
          my $dens = calculUtils::dens_ro($dr, $dth,
              $rho, $th, $r, $o, $vr, $vo, $vro);
          my ($X1, $Y1, $X2, $Y2) = calculUtils::
             compute_line_sup($rho, $th, $x1, $y1,
             $x2, $y2);

          if ($dens > 1) {$dens = 1;}
          if ($dens > $densLimit)
            {
              my ($Xi1, $Yi1) = $data->rob_to_img(
                 $X1, $Y1);
              my ($Xi2, $Yi2) = $data->rob_to_img(
                 $X2, $Y2);
               if ($map eq 'envMap')
                 {
                   $data->set_density($map, $dens,
                      $Xi1, $Yi1, $Xi2, $Yi2);
                 }
               else
                 {
                   $data->set_density('missionMap',
                      $dens, $Xi1, $Yi1, $Xi2, $Yi2);
```

```
                    }
                }
                  else
                {
                  $th = $o - $varM*sqrt($vo);
                  $dens = 1;
                }
            }
        }
    }
print "...    done\n";
}
```

La fonction *dens_ro* fait l'intégrale à proprement parlé, nous effectuons deux intégrales simples successivement. La méthode d'intégration est celle *Romberg*.

La fonction *compute_line_sup* calcule les extrémités projetées du segment sur la droite porteuse calculée.

Nous nous servons d'une image en niveaux de gris pour stocker les probabilités d'occupations. La fonction *rob_to_img* passe du repère robot au repère image.

La fonction *set_density* remplit les pixels de l'image qui sont traversées par le segment avec la plus grande probabilité calculée (les pixels sont initialement vides et à chaque nouveau segment ont regarde si la valeur augmente).

B.2.2 Discussion

Effectuer les calculs selon une discrétisation polaire pose le problème suivant : à $d\theta$ constant lorsque le rayon est grand, la taille de cellule augmente. Ceci a pour effet après passage en coordonnées cartésienne, "d'étaler" les segments (voir la figure III.10 page 63). Le phénomène est d'autant plus marqué que les dimensions de l'environnement sont importantes (ici il mesure plus de 20x30m).

Mais le calcul en coordonnées polaires permet, lorsque l'on fait une hypothèse de covariance nulle, de tirer profit des particularités de la distribution gaussienne et ainsi de simplifier l'intégration.

On peut voir dans le code proposé que nous avons dû rediscrétiser les cellules pour le calcul (nous avons divisé les incrémentations sur l'intégration par la variable *pas*). Sans cela les segments éloignés de l'origine étaient mal représentés dans la grille.

Une solution à ce phénomène serait de disposer d'une bonne méthode d'intégration double et de faire les calculs directement en coordonnées cartésiennes grâce à un changement de variables.

Les calculs ont été implémentés en *Perl*, langage qui n'est pas rapide en terme de calculs. La condition d'arrêt sur la probabilité minimum sert à éviter de faire des calculs non significatifs et donc à augmenter la rapidité. Même si les données sont interprétées hors ligne et après la simulation, on pourrait envisager de coder les parties calculatoires avec un langage plus efficace comme le *C*. Ceci est même presque obligatoire si l'on souhaite calculer l'information contenue dans les segments perçus à chaque instant, car alors il faut calculer une nouvelle grille très fréquemment.

A titre indicatif, une mission d'une dizaine de minutes avec une vingtaine de cartes acquises par le robot, nécessite une vingtaine de minutes de calculs pour interpréter toutes les données et les mettre sous forme exploitable pour une system map.