وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة باجي مختار – عنابة
كلية علوم المهندس
معهد الإعلام الآلي

Université Badji-Mokhtar – Annaba
Faculté des sciences de l'ingéniorat
Département d'Informatique

MEMOIRE

Présentée en vue de l'obtention du diplôme de MAGISTER en Informatique

Nouvelle approche de recherche de trajectoire pour robot autonome dans un environnement inconnu

Option

Informatique Embarquée

Par

MEDDAH Farouk

DIRECTRICE DE THESE: Lynda DIB Professeur *U.B.M. ANNABA*

Devant le jury

PRESIDENT :	KHADIR Mohamed Tarek	Professeur	*U.B.M. ANNABA*
EXAMINATEUR :	TOLBA Cherif	MC-A	*U.B.M. ANNABA*
EXAMINATEUR :	GHANEMI Salim	Professeur	*U.B.M. ANNABA*

2014

UNIVERSITE BADJI MOKHTAR-ANNABA
Faculté des sciences de l'ingéniorat
Département d'informatique

MEMOIRE
En vue de l'obtention du diplôme **MAGISTER** en informatique

Option : Informatique embarquée

MEDDAH Farouk

THEME

Nouvelle approche de recherche de trajectoire pour robot autonome dans un environnement inconnu

Devant le jury

Président :	*Pr. KHADIR Mohamed Tarek*	*UBM-Annaba*
Rapporteur :	*Pr. DIB Lynda*	*UBM-Annaba*
Examinateur :	*Dr. TOLBA Cherif*	*UBM-Annaba*
Examinateur:	*Pr. GHANEMI Salim*	*UBM-Annaba*

Année : 2014

Dédicace

Je dédie ce travail à tous ceux qui croient en moi et à mes capacités.

Remerciement

Je tiens à exprimer mes sincères reconnaissances à Melle Lynda Dib, ma directrice de thèse, pour m'avoir encadré tout au long de mes travaux, pour sa patience, pour ses conseils et pour toutes les heures, parfois tardives, qu'elle a pu me consacrer malgré les dérangements.

Mes remerciements vont également à mes rapporteurs Mr. Khadir, Mr Tolba et Mr. Ghnemi pour avoir acceptés d'être membre de mon jury de soutenance de ma thèse.

Je remercie aussi tous ceux qui m'ont aidé durant ce travail même avec un mot ou un souhait.

Abstract

One of the most persistent problems in robotics is the problem of path-planning in an unknown environment. The robot has a limited knowledge of his environment defined by the surface perceived by its sensors (ultrasound, infrared or laser). The purpose of this memory of magister is the proposal of a new path-planning algorithm for an autonomous robot without environmental restrictions.

The proposed algorithm should solve the problems of endless oscillations that may exist in robot's path (when overcoming obstacles and even in the case of a local minimum or spiral).

Résumé

Un des problèmes les plus persistants dans le monde de la robotique est le problème de recherche de trajectoire dans un environnement inconnu. Le robot n'a qu'une connaissance limitée de son environnement défini par la surface de la trajectoire perçue par ses capteurs (ultrason, infrarouge ou laser). Le but de recherche de cette mémoire de magistère est la proposition d'un nouvel algorithme de recherche de trajectoire pour un robot autonome sans restriction d'environnement.

L'algorithme proposé devra résoudre les problèmes d'oscillations sans fin qui peuvent exister durant le chemin de parcours du robot (lors de dépassement des obstacles et cela même dans le cas d'un minimum local ou de spirale.

ملخص

أحد المسائل الأكثر إلحاحا في عالم الروبوتات هي مسألة العثور على المسار في بيئة غير معروفة. في هذه الحالة الروبوت ليس لديه عن محيطه سوى معرفة محدودة هي المساحة التي يمكن استكشافها من قبل أجهزة الاستشعار (الموجات فوق الصوتية والأشعة تحت الحمراء أو الليزر). الغرض من البحث المقدم في مذكرة الماجستير هذه هو اقتراح خوارزمية بحث جديدة لمسار روبوت مستقل دون قيود بيئية.

الخوارزمية المقترحة يجب أن تحل مشاكل التذبذبات اللا متناهية التي قد تتكون في طريق للروبوت (عندما تجنب العقبات وحتى في حالة وجود محل أدنى أو مسار حلزوني).

Sommaire

Liste des figures

Chapitre 1

Introduction

« La science informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes ».

Edsger Dijkstra

1. Introduction

Le terme "*Robot*" a été utilisé pour la première fois en1921 par Karel Capek[1] dans sa pièce de théâtre R.U.R. (*Rossums Universal Robots* : Les robots universels de Rossum) en quelle il décrit la révolte de robots. Il provient du tchèque "*robota*" qui signifiée corvée, travail obligatoire. Le terme robotique a été employé pour la première fois par Asimov en 1941 [55] [56].

Le premier robot industriel *Unimate* a commencé son travail dans les lignes d'assemblages de General Motors à New Jersey en 1961 (Figure 1). Depuis ce temps, les robots sont utilisés massivement dans le domaine de l'industrie, les progrès faits en ce domaine les ont introduits dans plusieurs autres domaines, du domaine chirurgical jusqu'à l'astronomie.

Figure 1: Le premier robot industriel Unimate. [57]

La firme Unimation qui a construit le premier robot, était presque la seule sur le marché américain jusqu'à quinze ans après, malgré ça elle n'a commercialisé que

1 - Ecrivain tchèque, (1890-1938).

1000 robots, tandis que JIRA (association japonaise de robotique industrielle) annonçait en 1975 à peu près 65000 robots ?!

D'après la JRA (Japan Robotics Association : Association du Robotique Japonaise) en 2010 le marché de la robotique a marqué 24.9 Milliard de dollars ?!

En 2011, le nombre total de robots industriels en service à travers le monde est estimé à 1 035 000 produits et 15,7 milliards de dollars [63].

Figure 2: Robot chef d'orchestre (image tiré de [57]).

Une des aptitudes critiques pour les robots mobile est la navigation. Son objective est de conduire un agent (ici le robot) à sa destination en évitant les obstacles statiques (les mures, l'eau, barrière…) et les obstacles dynamiques ou mobiles (autres robots, véhicules, passerelle flottante,…) [42].

Les algorithmes de navigation et de planification de mouvements sont aussi utilisés avec succès dans d'autres domaines, tel que le graphisme CAD/CAM, les jeux vidéo, la biologie...

Cette thèse est une contribution dans le domaine de la recherche de trajectoire pour les robots autonomes dans un environnement inconnu.

1.1.Définition du robot

Il existe plusieurs définitions d'un robot :

- ***Robot*** n.m. (du tchèque *robota*, travail forcé, corvée). Définition 1 : Dans les œuvres de science-fiction, machine à l'aspect humain, capable de se mouvoir, d'exécuter des opérations, de parler. Définition 2 : (Technique). Appareil automatique qui, capable de manipuler des objets ou d'exécuter des opérations selon un programme fixe ou modifiable, voir par apprentissage. [56].
- Un appareil automatique qui peut effectuer des fonctions normalement effectués par des humains. Traduit du dictionnaire Webster's [60]
- [Dans des œuvres de fiction sc]. Machine, automate à l'aspect humain capable d'agir et de parler comme un être humain… [61].
- [Domaine technique]. Appareil effectuant, grâce à un système de commande automatique à base de micro-processeur, une tâche précise pour laquelle il a été conçu dans le domaine industriel, scientifique ou domestique. [61].
- Machine automatique dotée d'une mémoire et d'un programme, capable de se substituer à l'homme pour effectuer certains travaux. [62].

1.2.La robotique

Ensemble des techniques permettant la conception et la réalisation de machines automatiques et de robots. Ce terme a été utilisé pour la première fois par le romancier Isaac Asimov dans son récit de science-fiction «Menteur !», publié en 1941 [55].

1.3.Evolution de la robotique

Le défi de DARPA (*Defense Advanced Research Projects Agency*) donne un bon aperçu de la vitesse de progression de la robotique. En 2004, l'objectif de ce défi était de réaliser une navigation autonome de 96 km, en ville, tout en respectant le Code de la route et en prenant en compte la circulation extérieure : piétons et autres voitures non automatiques. En 2004, aucune voiture n'a pu réaliser plus de 11.78 km. En 2005, cinq véhicules ont terminées cette couse dont l'équipe la plus rapide était celle de Stanford avec un temps total de 6 heures et 54 minutes.

En 2012, un autre défi a été réalisé par Google c'est de conduire un aveugle de son domicile à un commerce local par une voiture automatique.

Le rapport PIPAME sur Le développement industriel futur de la robotique personnelle et de service en France (Erdyn Consultants, 2012) estime qu'en 2015, au niveau mondial, le marché de la robotique de service personnelle représentera 8 milliards de dollars tandis que le marché de la robotique de service industrielle représentera, quant à elle, 18 milliards de dollars [45] (Figure 3).

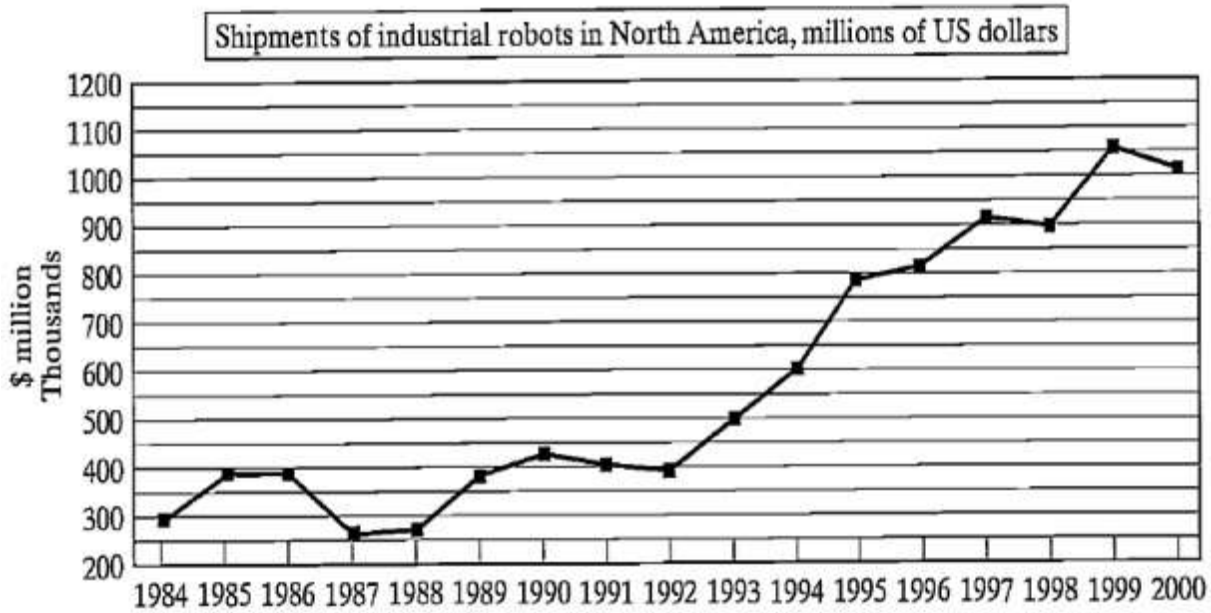

Figure 3: Marché des robots industriels dans l'Amérique du nord (Tiré de [58]).

1.4. Domaines d'applications

Le domaine de la robotique est en plein essor depuis quelques années. Les évolutions technologiques, dépassant sans cesse nos espérances, elles permettent de réaliser des solutions technologiques s'adaptant au moindre problème.

La robotique est utilisée dans beaucoup de domaines qui sont extrêmement rigoureux et exigeants. Nous allons explorer quelques domaines.

1.1.1. Exploration

NASA utilise les robots dans l'exploration d'autres planètes (exemple de Sojourner en 1997 (Figure 4), et Spirit en 2003) ; pour beaucoup de raison l'un est l'économie. Les robots sont utilisés aussi dans l'exploration sous-marine.

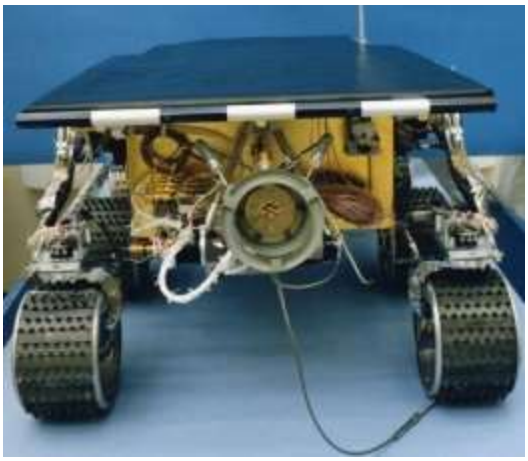

Figure 4: Sojourner, NASA, 1997 sur Mars (image tirée de [59]).

1.1.2. Industriel

Les robots sont utilisés très largement dans le domaine de l'industrie, l'assemblage des voitures et des cartes électroniques, la peinture, le transport, le nettoyage,…

Figure 5: Robot soudeur (image tirée de [59]).

1.1.3. Agriculture

L'agriculture est l'un des nouveaux domaines où la robotique est introduite, il existe maintenant des robots pour faire la majorité des travaux agricoles, allant de la plantation jusqu'à la récolte.

Figure 6: Tracteur autonome (image tirée de [59]).

1.1.4. Militaire

Les robots sont de plus en plus utilisés dans le domaine militaire. Les armés modernes se convergent vers l'utilisation des robots à la place des soldats, les robots sont utilisés dans les missiles intelligentes, les missions de reconnaissance et d'espionnage (guerre de golf 2000 et 2003, guerre de Lebanon 2006),…

Figure 7: Système automatique de détection des bombes (AMDS: The Automated Mine Detection System), développé par Carnegie Robotics, LLC. (Image tirée de [52]).

1.1.5. Civile

Dans ces jours, il n'est pas étonnant de trouver un robot qui fait le ménage, le nettoyage ou même la cuisine.

Figure 8: Aspirateur (image tirée de [59]).

1.1.6. Médical

Le domaine médical n'est pas une exception, des nouveaux techniques sont découvertes, les robots interviennent dans la chirurgie, ainsi que la télé-chirurgie.

Le robot « Da Vinci » est un robot médical qui peut opérer et diagnostiquer les patients. 1242 exemplaires étaient utilisés dans le monde en juin 2009 [63].

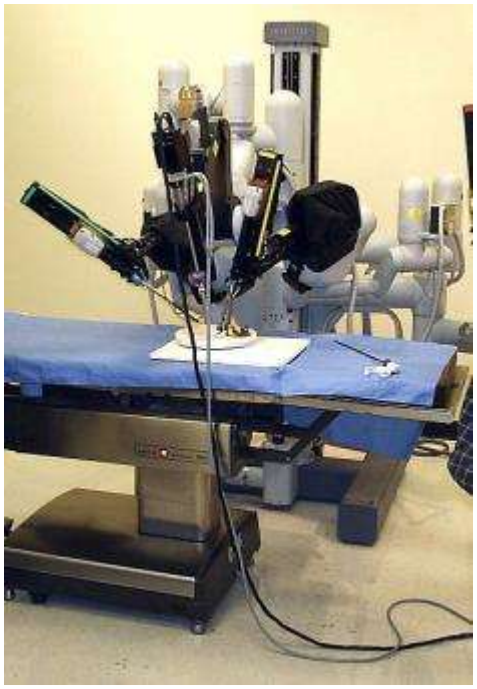

Figure 9: Le robot Da Vinci (image tiré de [63]).

On voit aussi apparaitre un nouveau domaine de recherche Cyborg [contraction de «CYBernetic ORGanism» (organisme cybernétique]. Le cyborg est la fusion de l'être organique et de la machine.

En 1964, l'université de Melbourne a attribué à Clynes le diplôme de «D. Sc» (Docteur en Science) [63].

1.1.7. Musique

Il n’est pas étonnant de voir un robot chef d’orchestre. En première mondiale le 17 mai 2008 un robot industriel FANUC ROBOTICS a dirigé deux morceaux classiques face à un ensemble instrumental à cordes à PARIS à la Cité des Sciences et de l'Industrie. Les morceaux joués étaient l'Andante festivo de SIBELIUS, et les Danses roumaines de BARTOK [63].

Figure 10: Le premier robot chef d'orchestre 17 mai 2008 (image tiré de [63]).

1.5.Types de robots

1.5.1. Robot manipulateur

Ce type de robot est généralement ancré et ne se déplace pas dans son lieu de travail, il est conçu généralement pour faire des tâches uniques et répétitives ou précises [55].

1.5.2. Robot mobile

Ce type de robot a la possibilité de se déplacer dans son lieu de travail. Contrairement au robot manipulateur ce type de robot peut évoluer de type autonome dans son environnement.

Les robots mobiles sont souvent désignés (catégorisés) par leurs types de déplacement, ci-dessous quelques types de robots mobiles.

1. **Mobiles à pattes** : ce type est utilisé sur des terrains dont il existe une grandes différence d'amplitude, son déplacement se fait en utilisant plusieurs pattes. Ce type de robot est plus performant par contre sa conception et commande est plus complexe.

Voir par exemple [43]

Figure 11: robot StarlETH quadrupel plateforme (Springy Tetrapod with Articulated Robotic Legs) image tirée de [43]

2. **Mobiles à chenilles** : Ce type de robot se caractérise par une meilleure adhérence au sol. Utilisées dans les terrains accidentés ou perturbés. Le changement de direction se fait en imposant des vitesses différences aux chenilles droite et gauche.

3. **Mobiles à roues** : Ce type de robot est le plus répandu à cause de la simplicité de conception et de commande. Pas structure mécanique particulière. Le robot est dit holonome s'il peut tourner à toutes directions.

4. **Mobiles se mouvant par reptation** : ils sont plus utilisés dans le domaine militaire. Ce type de robot se déplace par reptation dans des galeries ou des tuyaux.

5. **Volatile :** Parmi les robots volatiles existants les drones sont les plus répondus (faux-bourdon; sigle militaire : UAV = *Unmanned Aerial Vehicle*) est un aérodyne sans pilote embarqué et télécommandé qui peut emporter une charge. Ce type de

robots est destiné à des missions de surveillance, de renseignement, d'exploration, de combat, ou de transport [63] [29] [70]

Figure 12: Un drone AR.Drone en vol (image tirée de [63]).

6. **Robot flottant et sous-marine :** ils sont utilisés pour différent raisons tel que l'étude de vie sous-marine, l'étude de pollution, l'exploration, etc. voir par exemple [28] [44].

Malgré la diversité des types de robot les algorithmes de recherche de trajectoire sont presque les mêmes. Les différences se trouvent dans les commandes des mouvements et de direction.

Chapitre 2

Navigation

«Y a-t-il un pilote dans l'insecte ? ».

N. Franceschini

2. Navigation

Pour qu'un robot se déplace il doit avoir des informations à propos de son environnement local (il n'est pas obligé de connaitre la totalité de son environnement). Il faut distinguer un environnement totalement inconnu d'un environnement connu mais avec des obstacles dynamiques (environnement variable avec le temps).

Une des définitions de la navigation c'est la possibilité de connaitre sa position actuelle dans un cadre de référence et de planifier un plan vers une certaine destination [63].

Pour que le robot connaisse sa position, il existe différents types de capteurs utilisés suivant le type du robot et de l'environnement. Au fil des années on voit apparaitre de nouveaux capteurs.

2.1. Perception de l'environnement

Pour qu'un robot avance vers une destination quelconque, ou même faire n'importe quoi il doit avoir une connaissance de son environnement local (au minimum une partie de cet environnement). Pour cela les robots sont alimentés avec des capteurs de différents types voici les deux types de capteurs utilisés dans la perception :

2.2. Les différents types de capteurs

Il existe deux types de capteurs :

- Proprioceptifs : mesurent l'état du robot lui-même (capteur de position (GPS), capteur de vitesse, capteur de charge de batteries, ...).
- Extéroceptifs : renseignent sur l'état de l'environnement (capteur de température, télémètre (RADAR, LIDAR), boussole, détecteur de chaleur/lumière, ...) [55].

2.2.1. Capteurs proprioceptifs

Quelques exemples très utilisés dans l'actualité :

1. Capteurs ultrason

Les capteurs ultrason sont les premiers capteurs utilisés dans les robots. Le principe de ces capteurs est la mesure du temps de retour d'une onde sonore réfléchie par les obstacles pour estimer la distance des obstacles.

Les capteurs ultrason utilisent des ondes sonores non audibles par l'oreille humaine. La fréquence ultrason est trop élevée (le son est très aigu) supérieure à 20000Hz [63].

Figure 13: Exemple de capteur ultrason (image tirée de [63]).

La Figure **13** présente un capteur de Parallax, ce capteur est adaptable à la fois sur les Boe-Bot et les Sumobot. Les caractéristiques techniques de ce produit sont :

- Mesure de distance : entre 2 centimètres et 3,3 mètres.
- Erreur moyenne inférieure à 0,5 centimètres.
- Fréquence d'émission : 40 Hz.

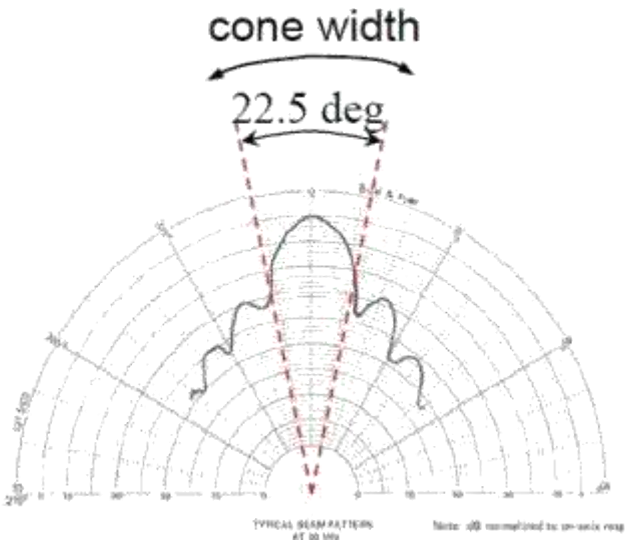

Figure 14: Forme typique de faisceau d'ultrasons.

Avantages

- Prix très bas.
- Très simple en utilisation
- Grand angle de détection
- Ce type de capteurs convient mieux pour des robots à petite distance et si on veut être sure que rien n'approche au robot.

Inconvénients

Les capteurs ultrason possèdent beaucoup d'inconvénients qui limitent leurs fiabilités et leurs utilisations, parmi ces inconvénients [53]:

- Temps de réponse et distance aveugle : ce type de capteurs possède une distance de quelques dizaines de centimètres où il ne peut pas détecter les obstacles en dessous à cause de la temporisation entre l'émission et le début de détection de l'onde sonore.
- L'onde réfléchie est très sensible aux conditions de l'environnement et aux matières des obstacles, par exemple un mur couvert par une moquette peut ne pas être détecté.
- Le résultat de mesure de ce type de capteurs peut varier suivant la forme d'obstacle rencontré. Le schéma suivant montre trois cas où les mesures retournés par ce type de capteurs seront fausses ou ne détecte pas l'obstacle.

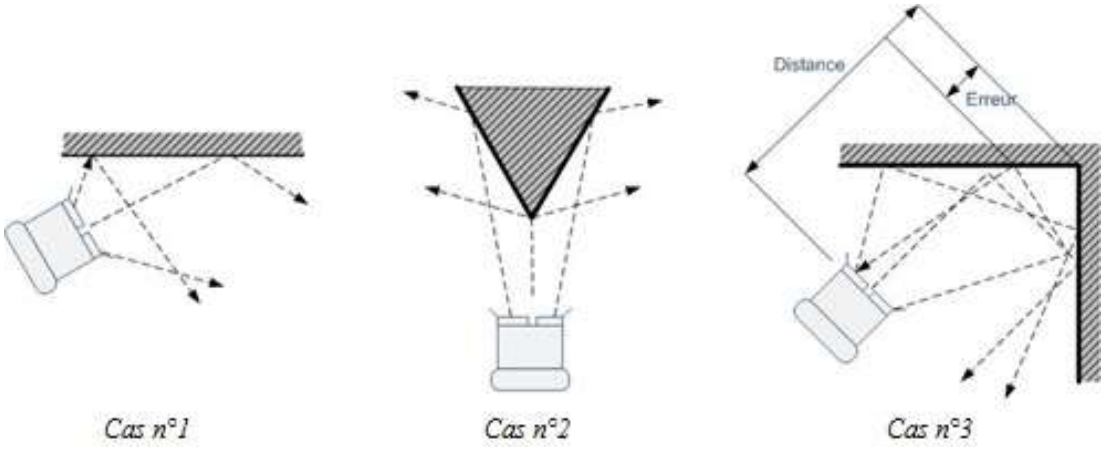

Figure 15: Exemple de quelques cas où les mesures des capteurs ultrasons sont fausses.

- En cas d'utilisation de plusieurs capteurs de ce type dans un même robot ou sur des robots différents un chevauchement survient, puisque on ne sait pas quel capteur a émis cette onde ou l'autre. On parle de phénomène de cross-talk.

2. Capteurs infra-rouge

Ces capteurs utilisent une lumière infrarouge au lieu d'une onde sonore. La lumière infrarouge émise rebondie sur un obstacle ou poursuit son chemin.

Quand cette onde est réfléchie vers le capteur dans un laps de temps donné, le capteur la perçoit et mesure l'angle entre l'émetteur, l'obstacle et le récepteur (sinon le capteur considère qu'il n'y a rien devant lui). L'angle varie suivant la distance à l'obstacle [63] [64] [53].

Pour cette raison, et à cause d'affaiblissement de l'onde émise on ne peut pas assembler ces capteurs en utilisant un émetteur avec un récepteur infrarouge normaux.

Figure 16: Capteur infrarouge Sharp GP2D120 (image tirée de [64]).

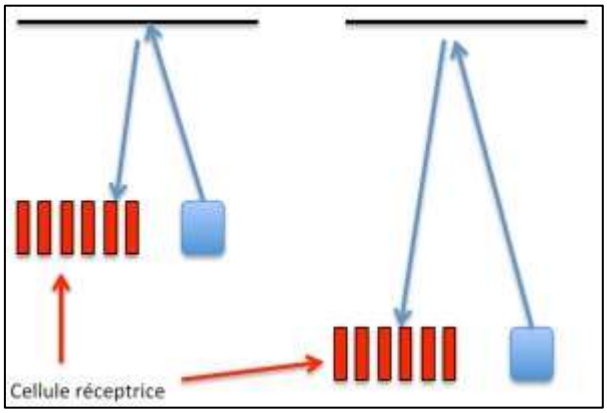

Figure 17: Exemple d'émission/réception d'une onde infra-rouge à des distances différentes.

Avantages

- Prix raisonnable (petite quantité sur le marché).
- Distance plus grande en comparaison avec celle des capteurs ultrason.
- Pas de chevauchement quel que soit le nombre de robots utilisé ou le milieu (même s'il y a des influences elles sont limitées), due à l'existence d'un filtre intégré.

Inconvénients

- Angle de détection faible il faut augmenter le nombre de capteurs pour palier la totalité du surface nécessaire pour le déplacement du robot.
- Le milieu et le type des obstacles peuvent influencer sur le rendement ou même la détection par exemple les vitres, l'eau, etc.

- Sensibles à la poussière.

3. Capteurs laser

Laser (acronyme de l'anglais : *Light Amplification by Stimulated Emission of Radiation*, en français : amplification de la lumière par émission stimulée de rayonnement) [63].

Ce type de capteur est de plus en plus utilisé dans les robots à l'heure actuelle. Ils utilisent un faisceau laser mis en rotation pour balayer un plan, généralement horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan [53].

Figure 18: Un exemple d'un capteur Laser à balayage, fournit 720 mesures réparties sur 360°, à 5 Hz (marque Ibeo) (tirée de [53]).

Avantages

- Distance de détection très grande, de quelques mètres à des dizaines de mètres (selon le modèle).
- Pas de problèmes de chevauchement.
- Très grande précision erreur à l'ordre de 1%.

Inconvénients

- Cher, fragiles, indisponible sur le marché.
- Angles très faibles, ce qui oblige le robot de faire des manouvres pour détecter l'existence des obstacles dans une surface plus importante (pour qu'il puisse avancer sans problème).
- Le milieu et le type des obstacles peuvent influencer sur le rendement ou même la détection par exemple les vitres, objets chromés, etc.

	Ultrason	**Infrarouge**	**Laser**
Portée	De 1 à 250 cm	De 5 à 80 cm	Plusieurs mètres à plusieurs dizaines de mètres selon les modèles.
Directivité	Cône d'environ 30°	Cône d'environ 5°	Les plus directifs (de l'ordre du degré, voire du demi-degré)
Précision	Relativement précis mais la précision diminue avec la distance, l'angle de mesure et les conditions de température et de pression.	Relativement précis mais la précision diminue avec la distance.	Sont précis avec un bruit de quelques centimètres sur des mesures de plusieurs mètres.
Coût	Peu chers	Peu chers	Relativement chers
Sensibilité aux interférences	Sensible à la température et à la pression. Egalement sensible aux autres robots utilisant la même fréquence ce qui peut poser problème dans une compétition.	Sont sensibles aux fortes sources de lumière qui contiennent un fort rayonnement infrarouge sont également sensibles à la couleur et à la nature des obstacles.	Ne peut pas détecter les objets réfléchissant la lumière laser (vitres, objets chromés,…)

Figure 19: Table comparatif des caractéristiques des trois types de capteurs les plus utilisés dans les robots mobiles.

4. Le GPS

Global Positioning System en français système de localisation mondial : un système de géolocalisation fonctionnant au niveau mondial. Entièrement opérationnel et accessible au grand public (avec certaines limitations comme la précision maximale).

Ce système comprend 24 satellites répartis sur 6 plans orbitaux tous inclinés d'environ 55° sur l'équateur (orbite circulaire de rayon environ 20000 km) [63].

Un capteur GPS capte les signaux d'au moins quatre satellites, la précision de ce système est à 10 mètres pour les capteurs qui ne disposent pas un code qui correspondant à la précision maximale (l'erreur de précision était presque 300m avant mai 2000) [63] [59].

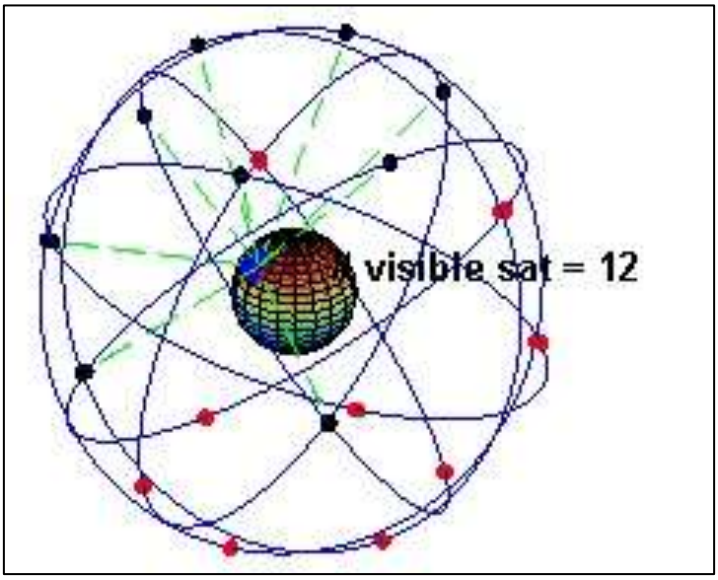

Figure 20: Constellation des satellites du GPS. (Image tirée de [63]).

A noter que l'utilisation de ces capteurs nécessite une puissance de calcul et une horloge de grande précision surtout si on utilise des capteurs à utilisation civile (qui ne disposent pas d'accès à la précision maximale des satellites), une erreur d'un millionième de seconde dans la synchronisation entre l'horloge du capteur et celle du satellite provoque une erreur de 300 mètres sur la position [63] [59].

Pour cette raison de précision (surtout dans certaines zones) on voit tendance de coupler ce type de capteurs avec d'autres types de capteurs tel que les capteurs sonar par exemple [68] ou des caméras RGB-D [29], d'autres utilisent plusieurs algorithmes de navigation en même temps [70].

2.2.2. Capteurs extéroceptifs

Ces types de capteurs renseignent sur l'état de l'environnement et non pas du robot, l'utilisation des caméras et la vision sont des éléments centraux. Les caméras produisent une quantité très importante d'informations pour les traiter il faut une puissance de calcul et une quantité importante de mémoire, ce qui rend leur utilisation dans les robots mobiles très limitée. Toutefois, les caméras sont utilisées de plus en plus [45] [71] [72].

D'autres types de capteurs très variés sont utilisés, par exemple les capteurs de chaleurs dans la recherche des feux à titre d'exemple [67] (ce projet est sponsorisé par le bureau de recherche naval américain), aussi les détecteurs de champs magnétiques ou autres types de champs [26]. On voit en outre l'utilisation des sonars dans les robots sous-marins [41] [44].

2.3.Localisation

Pour que le robot puisse naviguer dans son environnement, il doit se localiser dans l'environnement. La localisation est l'estimation de la position du robot par rapport à un repère fixe de l'environnement. La localisation est un facteur très important dans la conception des robots. Sans avoir la possibilité de se localiser un robot ne peut pas se déplacer.

Une des définitions de la localisation c'est l'ensemble des capteurs et des techniques permettant au véhicule de naviguer de manière autonome ou semi-autonome dans son environnement [46].

La localisation relative : permet au véhicule de naviguer à l'estime (*dead reckoning*) en utilisant uniquement les mesures de ses mouvements propres fournies par ses capteurs proprioceptifs.

La localisation absolue : utilise les mesures des capteurs extéroceptifs pour estimer la position du véhicule dans un repère lié à l'environnement. Les capteurs extéroceptifs permettent de recaler périodiquement la localisation obtenue par la mesure des mouvements [53].

Chapitre 3

Méthodes de recherche de trajectoire

“Il n’y a pas d’œuvre achevée, il n’y a que des œuvres abandonnées.”

Paul Valéry

3. Méthodes de recherche de trajectoire

Le problème de recherche de trajectoire a tiré l'attention des chercheurs depuis les années 70 et jusqu'à nos jours, beaucoup de travaux de recherches ont été effectués dans ce domaine qui englobent trois autres problèmes : localisation, planification et contrôle de mouvement.

Les années 80, c'est la naissance des algorithmes Bug pour recherche de trajectoire (Bug). Le premier algorithme est Bug1 créé en 1984 par Lamelsky [1] [2] [3].

Dans les années 1990-1999, beaucoup de méthodes classiques ont vu le jour : Bug3 hybridation de Bug1 et Bug2 1996 [24], Tangent Bug par Ishay Kamon en 1997 [49] la version finale en 1998 [12], ainsi que l'arrivée des méthodes du champ de potentielle Hussien en [48]. On voit aussi la naissance des méthodes heuristiques D* ou Dynamique A* inventé en 1994 par Anthony Stentz [20].

Dans ces dernières années (2000-2013), beaucoup de recherches s'attachent à l'extension de l'algorithme A* : A* modifié (connu Physical A*) [48], Focussed Dynamique A* par Stenz [20] [37], Extended A* par Matous Pokorny [19], ARA ARA+ [15], Improved D* par Jianming GUO en 2009 [17], Improved A* par Dongmei Zhang en 2010 [18].

Néanmoins l'arrivée de nouvelles approches de recherche de trajectoire, on voit toujours l'apparition de certains algorithmes de type Bugs des fois plus optimales tel que : HD1 Hiroshi Noborio en 2000 [27], Ave par Hiroshi Noborio en 2004 [21], K-Bug par Ricardo A. en 2007 [13], Point Bug par Buniyamin N. en 2011 [14], Dynamic Point Bug par B. Margaret Devi en 2013 [16].

Dans le présent chapitre on va exposer les différentes méthodes de recherche de trajectoire leurs principes, avantages et inconvénients.

Classification des méthodes de recherche de trajectoire

Il existe deux classes principales de méthodes de recherche de trajectoire pour les robots mobiles [63]:

1- Les méthodes déterministes : Elles trouvent toujours le même résultat (le même chemin) si on applique les mêmes conditions initiales. Les algorithmes classiques (Bug), Heuristiques (Compositions cellulaire), Champ de potentiel, Diagramme de Voronoï,…). Elles sont déterministes parce qu'elles garantissent de trouver une solution s'il existe une au moins.

2- Les méthodes probabilistes : Elles ne trouvent pas forcement le même chemin même en appliquant les mêmes conditions initiales. On trouve deux sous-catégories :

   a. Les méthodes d'échantillonnage : on prend un échantillon puis on lui applique des traitements jusqu'à l'arrivée à une solution.
   b. Les méthodes de diffusion : Effectuent une recherche aléatoire jusqu'à trouver une solution.

La plupart des méthodes exposées dans ce chapitre appartiennent à la première classe.

3.1.Méthodes Bug

Les méthodes Bug sont les premières méthodes de recherche de trajectoire, elles sont des méthodes basées sur deux comportements du robot (Figure 21) : (1) l'avancement en ligne droite vers la destination, (2) et l'évitement des obstacles. La grande différence entre ces méthodes se réside dans la condition de basculement entre ces deux comportements.

Autre caractéristique de cette famille d'algorithmes est leurs simplicités dans leurs principes et implémentations (plus ou moins) ce qui les rend plus adéquats pour être implémentées sur des robots à cartes électroniques de moyenne et faible gamme.

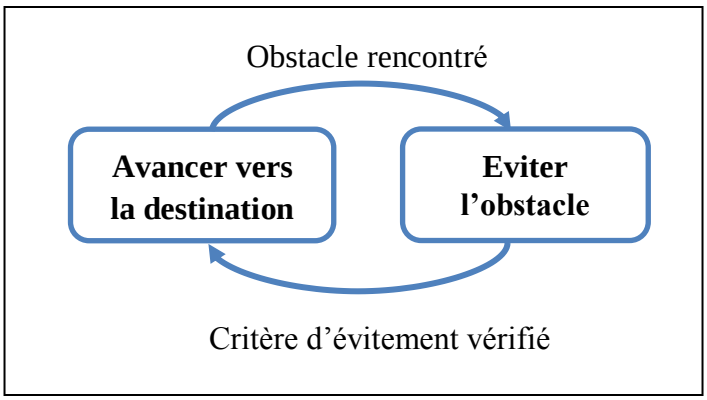

Figure 21: Le principe de la plupart des algorithmes Bug.

3.1.1. Avantages

- Les méthodes Bugs sont intéressantes parce qu'on peut les appliquer dans un environnement inconnus (elles sont indépendantes de la géométrie de l'environnement).
- La plupart d'entre elles sont convergentes qui veut dire elles trouvent un résultat.
- N'utilisent pas de fonctions mathématiques complexes (racine, sinus,..), et ne demandent pas d'espace mémoire énorme d'ailleurs certaines algorithmes de cette famille n'utilisent même pas une mémoire.
- Simple à implémenter sur des robots réels.
- Les résultats donnés par les tests réels prouvent l'efficacité de ces algorithmes.

3.1.2. Inconvénients

- Certains algorithmes de cette famille ne sont pas convergents.
- Dans la plus part des algorithmes la longueur du chemin peut-être soit meilleure ou très mauvaise mais pas moyenne.
- Le résultat de l'algorithme dépond de l'environnement, et la convergence de certains algorithmes ne peut pas être prouvée théoriquement.
- Certains algorithmes restent théoriques parce qu'ils supposent l'utilisation d'un matériel idéal ou des capteurs à distance illimités.

3.2.Méthodes heuristiques

Ces méthodes se basent sur des estimations du coût de trajectoire. L'algorithme utilisé essaye de minimiser la fonction du coût choisi (le tableau dans Figure **22** présente quelques fonctions de coût utilisées).

Dans les méthodes heuristiques l'environnement du robot (la carte géographique) est divisé en plusieurs cellules ou cases (*Grid*). Si une case contient un pourcentage d'obstacle que soit petit ou grand elle est déclarée comme obstacle sinon elle sera déclarée comme libre (pas d'obstacle) et le robot peut passer par elle.

Pour trouver son chemin le robot utilise une fonction de coût qui sera calculée à chaque case, l'objectif du calcul est de minimiser cette fonction.

La différence entre les algorithmes heuristiques réside dans cette fonction (et rien d'autre).

A chaque position du robot les coûts de toutes les cases voisines sont recalculés et la case ayant le coût minimal sera la prochaine position du robot [73].

3.2.1. Avantages

- S'il existe un chemin l'algorithme finira par le trouver (selon certaines conditions).
- Pas de boucles ou oscillations infinies, un chemin ne peut être parcouru qu'une seule fois.
- Peut-être adapter facilement pour qu'elles prennent en considération l'état et la nature du terrain [39].

3.2.2. Désavantages et problèmes

Les méthodes heuristiques comportent pas mal de problèmes. Beaucoup de recherches ont essayées de les éliminer ou du moins les minimiser, on va exposer quelques-uns de ces problèmes.

1. Chemin très proche ou très loin des obstacles

Ce problème fait qu'on est obligé d'appliquer des modification à la carte de l'environnement (filtre Blur de Gauss : *Gaussian Blur*) afin que le chemin soit un peu

plus loin des obstacles, c'est pour ça que la formule de base est modifiée d'un algorithme à un autre, quelques formules les plus utilisées sont présentées dans la Figure 22.

Métrique	Formule
Euclidien	$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
Manhattan	$\lvert x_1 - x_2 \rvert + \lvert y_1 - y_2 \rvert$
Manhattan diagonale	$max(\lvert x_1 - x_2 \rvert, \lvert y_1 - y_2 \rvert)$
Moore	$max(\lvert x_1 - x_2 \rvert, \lvert y_1 - y_2 \rvert) + min\left(\lvert x_1 - x_2 \rvert, \lvert y_1 - y_2 \rvert(\sqrt{2} - 1)\right)$

Figure 22: Liste de quelques formules de calcul de distance dans les méthodes heuristiques (fonctions de coût).

En plus de ce problème, le problème inverse se pose aussi, vu que la cellule sera déclarée comme obstacle si elle contient une partie d'un obstacle certaines cellules seront déclarées comme obstacles malgré que l'obstacle n'occupe qu'une très petite partie de ces cellules, ce qui fait que le chemin va être de plus en plus loin des obstacles dans ces cas.

2. Effet de la résolution

La résolution de la grille utilisée joue un rôle très important, si la résolution est faible le chemin sera long et il est très probable de ne pas trouver une solution.

Pour être sûre de trouver une solution (si elle existe bien sûr) il faut utiliser une résolution suffisamment grande, mais cela à des inconvénients : il faut utiliser un espace mémoire plus grand, les calculs augmentent par conséquence ils sont déjà grandes, en plus d'autres problèmes concernant la longueur du chemin et le temps de parcours.

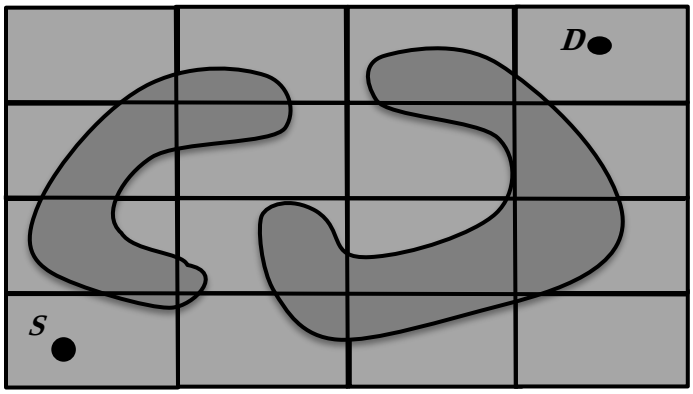

Figure 23: Exemple de carte divisée en 4x4 (ici pas de chemin).

Dans la figure ci-dessus (Figure 23) il n'existe pas de chemin reliant (*S*) avec (*D*), du point que toutes les cases de la troisième ligne contiennent des obstacles.

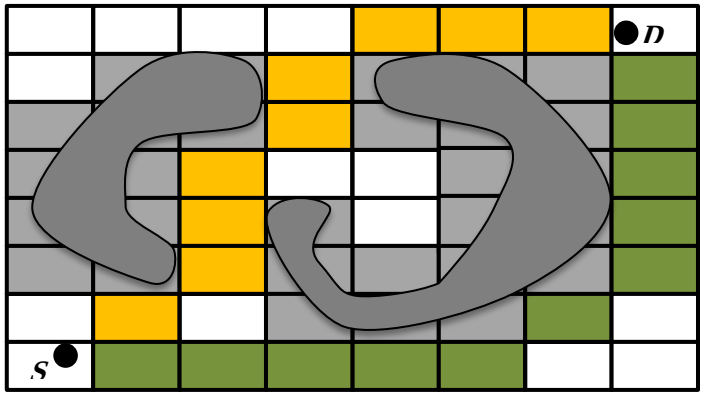

Figure 24: Exemple de carte divisée en 8x8 (ici deux solutions).

Dans cette figure (Figure 24) on a utilisé une résolution plus importante (le double en largeur et en hauteur que la précédente), on voit clairement qu'il existe deux solutions possibles.

Les deux figures ci-dessus montrent clairement l'effet de résolution sur le résultat de la recherche trouvée par ce type d'algorithmes.

Pour être sûre de l'efficacité de l'algorithme il faut utiliser une haute résolution, ce qui augmentera par la suite le volume des calculs.

3. Etat des cases de départ et d'arrivée

Un autre problème des méthodes heuristiques c'est l'état des deux cases de départ et d'arrivée si la case de départ contient un pourcentage d'obstacle alors le robot ne peut même pas bouger, par contre si la case d'arrivée contient un pourcentage d'obstacle alors le robot n'arrivera jamais à sa destination.

4. Autres problèmes

- Ce type d'algorithmes demande une mémoire des fois énorme pour enregistrer le chemin parcouru ainsi que les obstacles rencontrés, ce qui les rend inadéquates pour des environnements ouverts ou pour des grandes distances.
- Ces algorithmes demandent beaucoup de calcules à chaque mouvement du robot surtout en cas de rencontre d'un obstacle, ce qui oblige à utiliser un matériel de haute performance (ce qui augmente le coût des robots utilisés et par conséquence limite leurs utilisations), certaines équipes de recherches ont incorporé même des ordinateurs complets dans leur robots à titre d'exemple [30] [73] [38].
- Beaucoup d'algorithmes ne tiennent pas considération les dimensions du robot surtout dans le cas d'un chemin entre deux obstacles (ou plus), d'ailleurs c'est un des inconvénients de l'algorithme A* de base [30], la plupart des algorithmes dérivés ont été réalisés par simulation et peu d'entre eux sont implémentés sur des robots en monde réel.

Une question se pose dans ce type d'algorithme : « que se passe-t-il si le robot perd sa localisation ou si une erreur de localisation se produit ? » [66] Le robot dans ce cas n'arrivera jamais à sa destination, au contraire des autres algorithmes qui peuvent remédier à ce type de problèmes.

3.3.Champs de potentiel

L'algorithme des champs de potentiel inventé en 1986 par Khatib Oussama [36], a été développé initialement pour contrôler des robots manipulateurs puis il a été généralisé pour trouver une trajectoire pour robot mobile dans un environnement inconnu.

Les champs de potentiel est une méthode qui assimile le robot à une particule soumise à un champ de forces répulsives et attractives.

Un obstacle génère un champ de potentiel répulsif tandis que la destination du robot génère un champ de potentiel attractif. L'algorithme calcul donc un vecteur résultant qui indiquera au robot comment effectuer son déplacement [48].

$$U(p) = U_{destination}(p) + \sum U_{obstacles}(p)$$

La force appliquée sur le robot est l'inverse du gradient du potentiel total

$$F(p) = -\nabla U(p)$$

Exemple d'application de cet algorithme

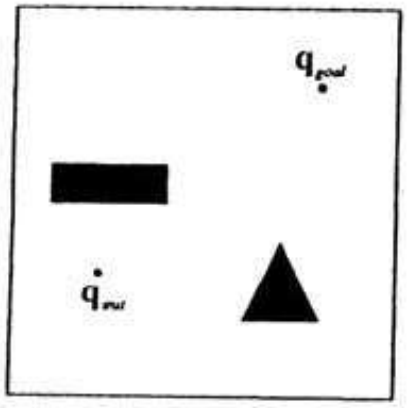

Figure 25: Environnement d'application de l'algorithme (image tirée de [65]).

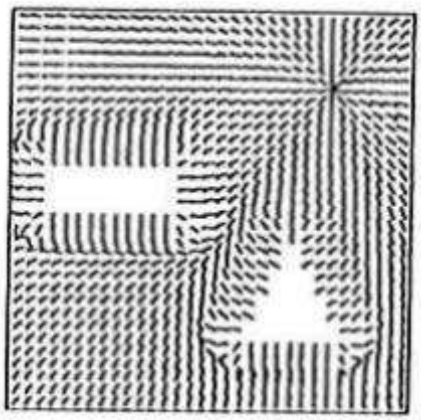

Figure 26: Gradient négatif (image tirée de [65]).

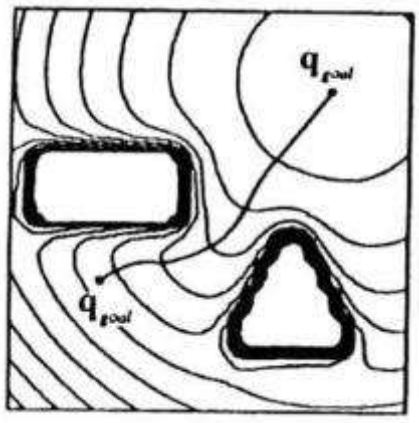

Figure 27: Total du potentiel ainsi que le chemin généré par l'algorithme (image tirée de [65]).

3.3.1. Avantages

- Cet algorithme est totalement réactif et peut donc être implémenté en temps réel très facilement.
- Les calculs se font directement et en temps réel et la génération du chemin ne nécessite aucune connaissance préalable de l'environnement, et même si l'environnement est connu seules les informations locales sont utilisées.
- Tous les calculs des forces de contrôle sont émergés dans une seule fonction.
- Le chemin généré est lisse ce qui minimise les manœuvres du robot ainsi que la consommation d'énergie.

- La vitesse du robot est adaptée suivant l'existence des obstacles, augmentée en cas d'absence d'obstacle et diminuée dans le cas contraire.
- A part des problèmes existants dans cette méthode, les calculs sont très simples et efficaces plus que ceux des autres méthodes, et le chemin généré est optimal.

3.3.2. Inconvénients

La méthode des champs de potentiel soufre de trois grands problèmes [48] [65].

1. Problème des minimums locaux

Cette méthode entraine facilement le robot dans des minimas locaux surtout dans un environnement peu convexe.

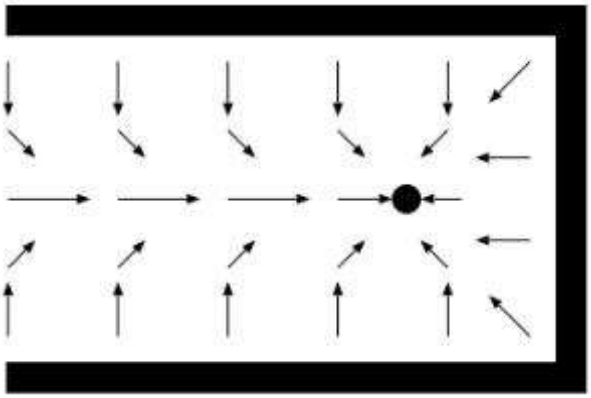

Figure 28: Exemple du problème de minimum local dans l'algorithme de champs de potentiel (image tiré de [53]).

Le cas du minimum local qui est très fréquent d'ailleurs met le robot dans un cas où il ne peut pas décider de la direction à prendre, La Figure 28 présente un exemple de ce type de problème, au meilleur des cas le robot entre dans un cas d'oscillation.

Pour sortir de ce cas il existe plusieurs façons par exemple faire déclencher un comportement particulier tel que : suivre le contour de l'obstacle, déplacer aléatoirement,…

Cependant la possibilité de rencontrer un minimum local (peut être le même minimum) reste posée, et l'évitement de ce cas ou même sa détection nécessite beaucoup de calculs des fois complexes [53].

Quelques méthodes de détection et d'évitement des minimums locales ont été étudiées dans les travaux de Michael Colin Hoy [41].

2. Problème d'instabilité

En plus, des problèmes d'oscillation peuvent survenus, dans certain cas tel que le cas d'une petite porte ou entre deux obstacles proches, ce qui rend le robot incapable d'avancer dans ce chemin.

Pour que le robot puisse atteindre sa destination dans ces cas il faut qu'il suit un chemin extrêmement précis (fin) puisque le gradient dans cet endroit (entre deux obstacle proche) est très minimal dans un intervalle très petit. En plus du chemin mince il faut une grande précision dans les calculs et malheureusement comme ce n'est pas le cas généralement avec les nombres flottants le robot ne peut pas passer avec précision entre deux obstacles.

Le problème d'oscillation peut être résolu en utilisant la méthode modifiée de Newton pour trouver la direction du robot qui est définie par [48]:

$$\dot{p} = -B(p)\frac{\partial U(p)}{\partial p}$$

Où $B(p)$ est une fonction définie par :

$$B(p) = (\varepsilon I + H)^{-1}, \varepsilon = \varepsilon(p) \geq 0$$

Tel que I est la matrice d'identité, Hest le Hessien (dérivée seconde Laplacienne) de $U(p)$, la valeur ε est la plus petite valeur qui rend la matrice $\varepsilon I + H$ positive et inversible avec son inverse positive.

3. Problème d'incapacité d'atteindre le but

Si la destination se trouve au près d'un obstacle le robot ne peut pas atteindre la destination à cause de la force répulsive généré par cet obstacle.

En plus de tous ces problèmes cités, cet algorithme ne tient pas compte des contraintes du robot [41].

3.4.Algorithmes Génétiques

Les d'algorithmes génétiques sont le fruit des recherches de John Holland et de ses collègues et élèves de l'Université du Michigan qui ont, dès 1960, travaillé sur ce sujet. La popularisation des algorithmes génétiques sera l'œuvre de David Goldberg à travers son livre (*Genetic Algorithms in Search, Optimization, and Machine Learning*) (1989) [63].

Les algorithmes génétiques se basent sur la notion de sélection naturelle et l'appliquent à une population de solutions d'un problème donné.

Etapes d'exécution d'un algorithme génétique [63] (voir Figure 29):

1. **Population de base** : Il faut choisir une population aléatoire de n chromosomes, chaque chromosome ici présente un emplacement prochain du robot (cette signification du chromosome est adaptée dans [35]), on peut aussi la changer pour que chaque gène désigne une prochaine direction du robot.
2. **Evaluation** : Mesurer la fitness de chaque chromosome dans la population.
3. **Sélection** : Créer une nouvelle population avec répétition des étapes suivantes jusqu'à ce que la population soit complète.
4. **Croisement et mutation** : Chaque couple génère deux enfants, dans cette opération deux chromosomes s'échangeant une ou plusieurs parties pour données des nouveaux chromosomes. S'il n'y a pas de mélange, le résultat est une copie exacte des parents. La mutation veut dire qu'un gène dans un chromosome peut substituer à un autre d'une façon aléatoire.
5. placer le résultat dans une nouvelle population qui sera utilisée dans les prochaines itérations de l'algorithme.
6. **Test du critère d'arrêt** : Si ce critère n'est pas vérifiée alors aller à l'étape (2).

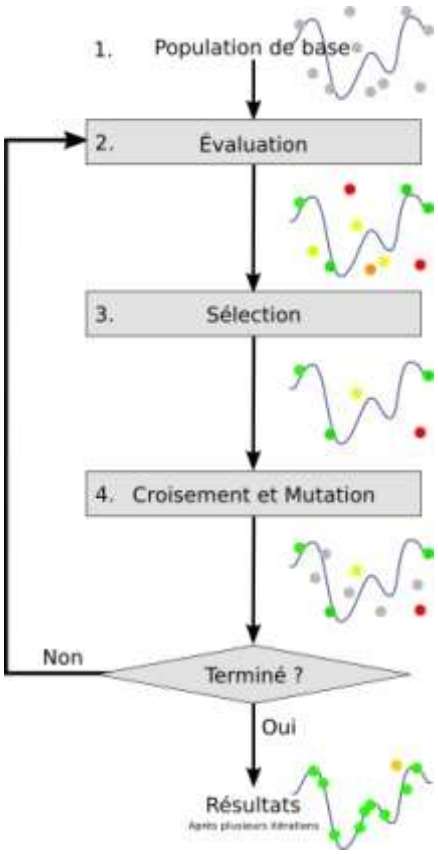

Figure 29: schéma présentant les étapes d'exécution d'un algorithme génétique (image tirée de [63]).

Exemple d'application d'un algorithme génétique

L'exemple suivant est publié en [35]. Chaque chromosome désigne un point ou emplacement du robot dans l'environnement, on a au départ une liste de points. A chaque itération on fait la modification d'un point (croisement voir Figure 32), l'ajout d'un nouveau point (mutation voir Figure 30) ou la suppression d'un mauvais point (disparition voir Figure 31).

Les étapes précédentes sont répétées jusqu'à la construction complète du chemin du robot de la source vers la destination, l'espace du choix des points est la totalité de l'environnement.

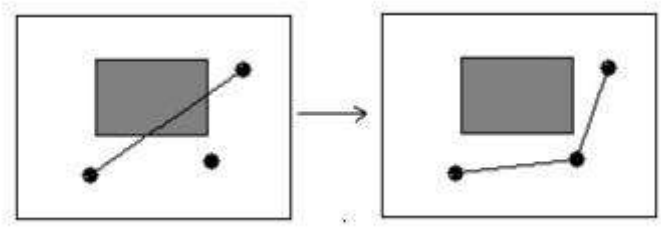

Figure 30: Ajout d'un prochain emplacement du robot (image tirée de [35]).

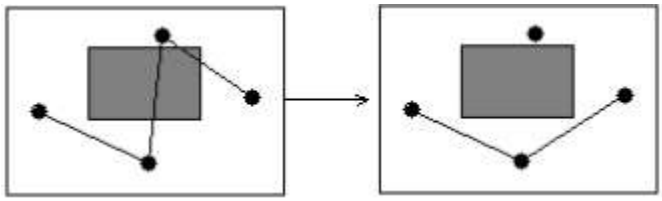

Figure 31: Suppression d'un mauvais emplacement du robot (image tirée de [35]).

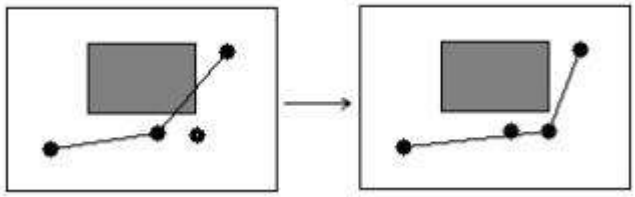

Figure 32: Déplacement d'un prochain emplacement du robot (image tirée de [35]).

3.4.1. Avantages

- Les algorithmes génétiques arrivent à trouver des résultats.
- Pas de problèmes de minimums locaux ou des contraintes sur l'environnement du robot.
- Il suffit de commencer par une solution quelconque, sans avoir besoin d'utiliser une solution optimale ou de définir des caractéristiques de la solution initiale.

3.4.2. Inconvénients

- Ces algorithmes sont coûteux en terme de volume de calcul, taille mémoire utilisée.
- Le chemin trouvé n'est pas optimal.

- Aucune garantie de convergence de l'algorithme ou façon de savoir le cas d'inexistence de solution.
- Problème des optima locaux, si dans la population à un temps donnée un individu devient majoritaire alors la population va converger vers cet individu et ne peut pas évoluée. Ce problème est lié au principe de l'algorithme lui-même et n'a aucune relation ni avec l'environnement ou avec le robot utilisé, il existe des travaux et quelques solutions pour remédier à ce problème.

Les résultats obtenues par l'utilisation des algorithmes génétiques dans le domaine de recherche de trajectoire pour robot autonome ne sont pas encourageante voir [31] [34].

En plus du temps et volume des calculs important, la convergence de cet algorithme n'est pas suffisante ce qui pousse à faire une hybridation de ce type d'algorithme avec d'autre algorithmes voir par exemple [32] [33].

3.5.Autres Méthodes

Il existe d'autres méthodes de recherche de trajectoire pour robot autonome dans un environnement inconnu, qui n'appartiennent à aucunes des approches citées précédemment, par exemple on voit l'apparition de quelques publications sur l'utilisation des probabilités dans la recherche de trajectoire dans un environnement inconnu (méthodes stochastiques). Voir [74] [75].

Chapitre 4

Méthodes Bug d'évitement d'obstacles

"Le plus grand obstacle à la découverte n'est pas l'ignorance, c'est l'illusion de la connaissance."

Michaël Aguilar

4. Méthodes Bug d'évitement d'obstacles

Appelé aussi (Algorithmes à base de règles *Rule based Algorithms*) [47]. Cette famille d'algorithmes est connue comme des algorithmes d'évitements d'obstacles plus que des algorithmes de recherche de trajectoire.

Dans ce chapitre on va exposer les différentes méthodes Bug d'évitements d'obstacles, depuis l'algorithme Bug1 (1984), jusqu'à l'algorithme Dynamique Point Bug (2013).

4.1. L'algorithme BUG1

Présentation et principe

C'était le premier algorithme de la famille BUG, inventé par V. Lumelsky et A. Stépanov [1] [2]. Lumelsky a commenté son algorithme Bug1 en disant [2]: «*L'algorithme Bug1 est le meilleur qu'on peut offrir aujourd'hui*».

Le principe de cet algorithme et que le robot avance vers la direction de la destination jusqu'au moment où il trouve un obstacle.

Quand le robot trouve un obstacle il tourne au tour de la circonférence de cet obstacle jusqu'à revenir au point de rencontre avec cet obstacle. Le but de cette tournée est de trouver le point le plus proche de la destination (ayant la distance minimale vers la destination).

Après le retour au point de rencontre de l'obstacle, le robot va directement au point le plus proche de la destination, puis il continue son chemin vers la destination.

Exemple

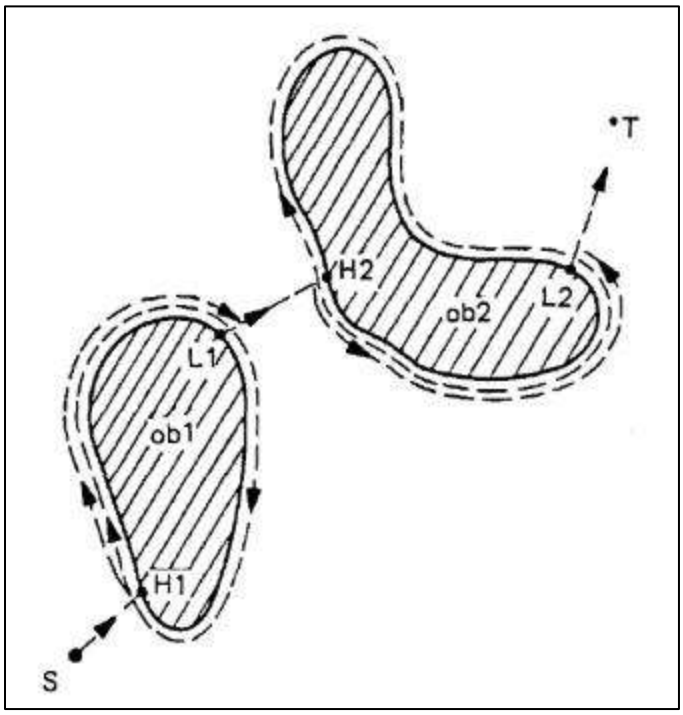

Figure 33: L'algorithme Bug1 (tiré de l'article [2]).

Dans l'exemple ci-dessus (Figure 33) le robot avance du point source (S) vers la destination (T), et quand il rencontre le premier obstacle (ob1) au point (H1), il tourne toute autour de cet obstacle. En même temps, le robot calcule à chaque point de la circonférence de l'obstacle (ob1) la distance entre le point en cours et la destination (T), et il sauvegarde le point ayant la distance minimale, (L1) dans cet exemple.

Quand le robot fait le tour complet de la circonférence de l'obstacle (ob1) en arrivant au point (H1) pour la deuxième fois, il va directement au point (L1), et continue son chemin vers la destination (T).

En avançant, le robot trouve le deuxième obstacle (ob2), il refait le même traitement déjà réalisé avec le premier obstacle (obs2). Le début de tournée au tour de cet obstacle depuis le point de rencontre (H2), pour trouver un nouveau point plus proche de la destination [le point (L2) cette fois]. En arrivant au point (H2) pour la deuxième fois, le robot va directement au point (L2) et continu son chemin vers la destination (T).

Remarques

Le point fort de cet algorithme est qu'il arrive toujours à atteindre la destination s'il existe un chemin, en plus tous les obstacles rencontrés sont explorés une seule fois seulement.

Néanmoins, cet algorithme a un très mauvais inconvénient c'est que le chemin parcouru est très long. En plus, il fait beaucoup de calcules qui consomment du temps (racine carré et multiplication), d'autre part il peut y avoir des problèmes dans l'application de cet algorithme en monde réel (surtout dans un environnement ouvert):

- Un exemple d'environnement fermé : Dans un immeuble si on a un des murs comme obstacle dans n'importe quelle partie du chemin, l'algorithme va parcourir tout la circonférence intérieure et extérieure de l'immeuble.
- Un exemple d'environnement ouvert : Si dans l'immeuble précédent on ouvre une porte vers l'extérieur le robot va parcourir cet immeuble et tous les immeubles des voisins.

4.2.L'algorithme BUG2

Présentation et principe

Cet algorithme est un premier essai d'amélioration de l'algorithme Bug1, inventé par Vladimir J. Lumelsky et Alexander A. Stepanov [1] [2] [3] [4] [5].

Le principe de cet algorithme est presque le même que le précèdent (l'algorithme Bug1) sauf qu'on tourne au tour de cet obstacle jusqu'au point de rencontre avec la ligne imaginaire (*m-line*) qui relie la source avec la destination, puis le robot continue son chemin vers la destination à partir de ce point.

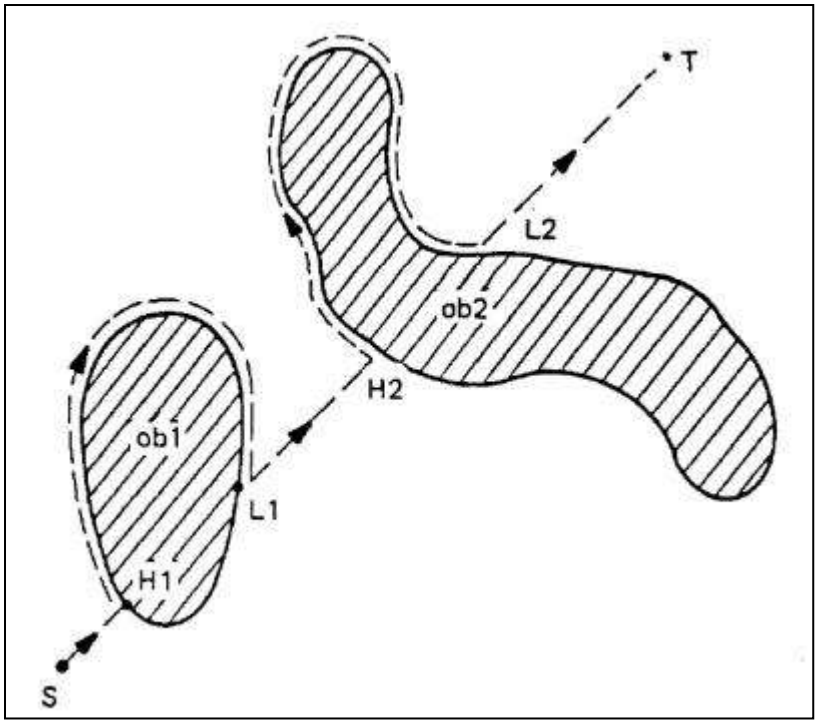

Figure 34: L'algorithme Bug2 (tiré de l'article [3]).

Dans cet exemple (Figure 34) le robot démarre du point (S), puis il avance vers la destination (T), il rencontre l'obstacle (Obs1) au point (H1), il tourne autour de (Obs1) jusqu'au point (L1) où il continue son chemin suivant (ST). Après, il rencontrera l'obstacle (Obs2) au point (H2), il tourne autour de (Obs2) jusqu'au point (L2) et finalement il arrive à sa destination (D).

Remarques sur cet algorithme

- Cet algorithme peut exploiter les obstacles deux fois ou même plus avant de donner une solution.
- La convergence de cet algorithme n'est pas sure, il ne donne pas toujours une solution, ci-dessous un des exemples où cet algorithme ne donne pas de solution.
- Si une solution est donnée par cet algorithme elle peut ne pas être optimale.

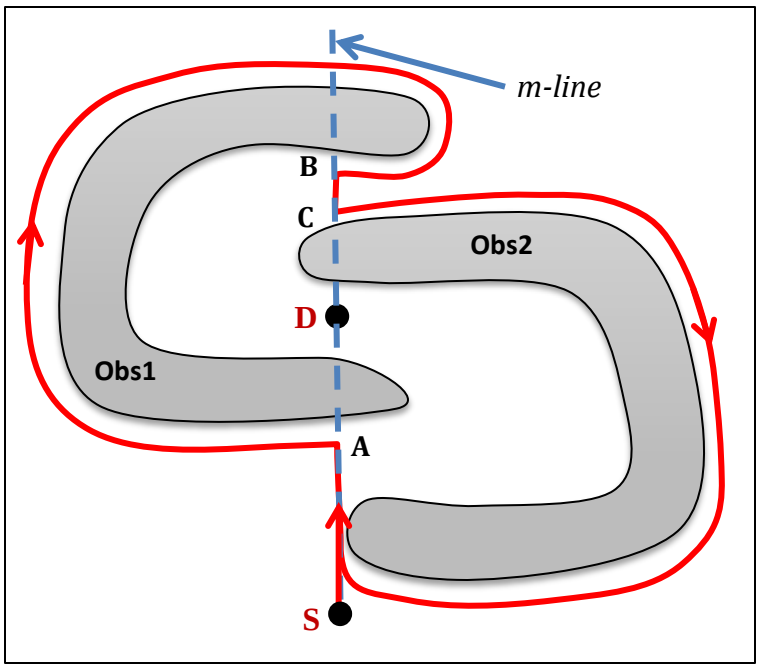

Figure 35 : Problème d'échec de l'algorithme Bug2.

4.3.L'algorithme ALG1

Cet algorithme est une extension de l'algorithme Bug2, inventé par Sankaranarayanan et Vidyasagar [6] [7] [8].

Un des problèmes de Bug2 est qu'il peut explorer un obstacle deux fois ou même plus, pour remédier à ce problème, Alg1 sauvegarde tous les points de rencontre des obstacles ainsi que les points de dépassements correspondants, et à chaque fois où le robot rencontre un point de dépassement ou de rencontre d'un obstacle une deuxième fois, il va directement au point de dépassement correspondant et continue son chemin depuis ce point mais en changeant le sens de tourné cette fois pour définir un nouveau point de dépassement (qui va remplacer l'ancien point de dépassement).

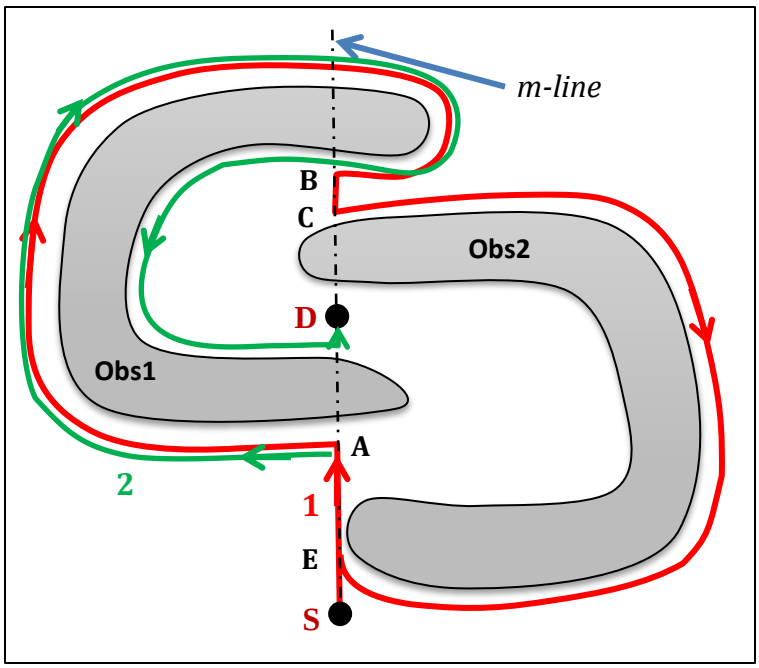

Figure 36: Application de l'algorithme ALG2 dans un exemple d'échec de l'algorithme BUG2.

L'exemple ci-dessus (Figure 36) montre l'application de l'algorithme Alg1 sur l'exemple précédent (cas d'échec de Bug2, voir Figure 35). Quand le robot arrive au point (E) pour la deuxième fois il va suivre la ligne (*SD*) de nouveau jusqu'à l'arrivée au point (A) une deuxième fois.

A ce point le robot va directement au point de dépassement correspondant à (A) qui est le point (B) [le chemin en vers]. Puis quand il arrive au point B, il tourne au sens inverse et continue son chemin pour atteindre sa destination (D).

4.4.L'algorithme ALG2

Cet algorithme est une amélioration de l'algorithme Alg1, inventé par Sankaranarayanan et Vidyasagar [7] [8].

La notion de *m-line* (la ligne imaginaire qui relie la source (S) avec la destination (D)) est abandonnée. Ici le robot teste les distance au lieu de rencontrer *m-line* il avance vers le point qui minimise la distance entre le point en cours et la destination.

Cet algorithme est plus optimisé que les algorithmes précédents, mais les calculs sont plus importants.

4.5.Autres variations des algorithmes BUG1 et BUG2

Beaucoup d'algorithmes ont été créés afin de plus optimiser les algorithmes Bug1 et Bug2, parmi ces algorithmes :

4.5.1. L'algorithme BUGM

Cet algorithme est un autre essai afin d'améliorer les algorithmes Bug1, il a été publié par les mêmes auteurs de l'algorithme Bug1 et Bug2 [3].

Il essaye d'éliminer les situations où le robot tourne infiniment autour de la destination dans l'algorithme Bug2 (voir Figure 35).

A chaque rencontre de la ligne *m-line* qui relie la source (S) avec la destination (D), il teste si la distance entre ce point et la destination est plus grande que celle entre le dernier point de rencontre avec *m-line* et la destination ? Si oui le robot fera un tour complet au tour de la circonférence de l'obstacle afin de définir un nouveau point d'évitement, puis il continue son chemin depuis ce point. Sinon il n'existe pas de chemin amenant vers la destination.

Appliquant cet algorithme à l'exemple précédent d'échec de Bug2 (Figure 37) : Quand le robot atteint le point (E) pour la deuxième fois [après avoir fait le tour de l'obstacle (Obs2)], il aperçoit que la distance entre (E) et (D) est supérieure à celle entre (C) et (D). Alors le robot continue tout le tour de la circonférence de (Obs2). A la fin de ce tour le robot arrive à atteindre la destination (D).

Cas d'inexistence de chemin vers (D) :
Si on suppose que le point (D) se situe un peu plus haut à l'intérieur de l'obstacle (Obs2).

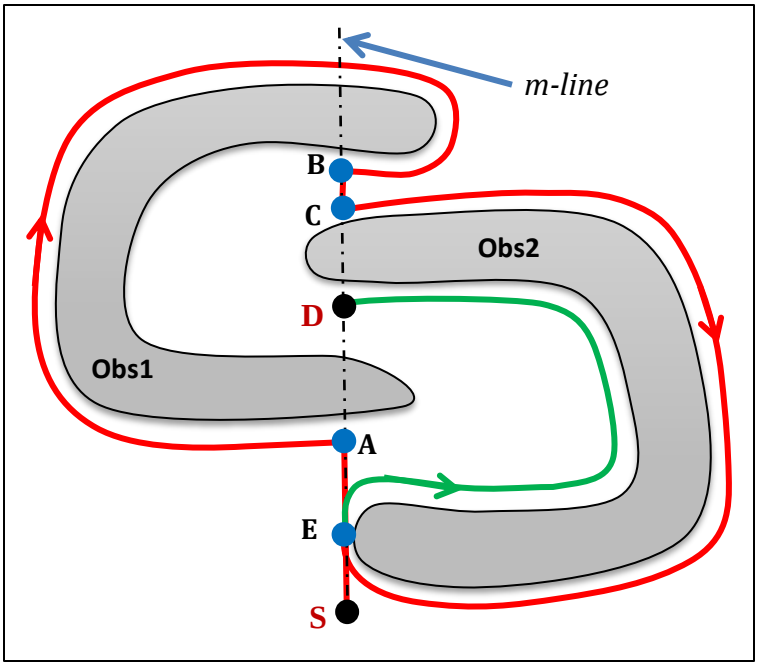

Figure 37: Algorithme BUGM1 appliquer dans un exemple d'échec de l'algorithme BUG2.

Malgré que cet algorithme arrive à atteindre la destination, toutefois il est très clair même dans cet exemple que le chemin produit par cet algorithme n'est pas optimal.

4.5.2. L'algorithme BUG3

Bug3 est inventé en 1996 par Giovanni Bianco et Riccardo Cassinis [24]. Cet algorithme est une hybridation des deux algorithmes Bug1 et Bug2. Le robot applique l'algorithme Bug2 qui est efficace mais il peut échouer. Quand l'algorithme Bug2 échoue le robot bascule vers Bug1 moins efficace mais converge toujours, un exemple est donné en Figure 38.

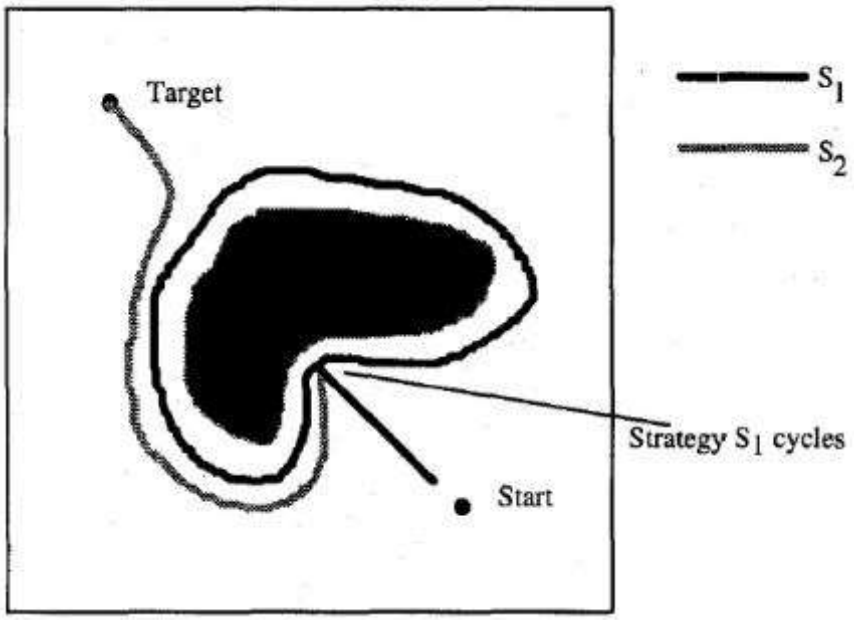

Figure 38: Exemple d'application de l'algorithme Bug3 {s1 :Bug1, s2 :Bug2} (image tirée de [24]).

La même remarque faite pour l'algorithme précédent se pose ici, en plus cet algorithme n'est pas fini (ils l'on écrit franchement) [24]; les auteurs n'ont donnés que deux exemples d'applications de cet algorithme dans leur article, et finalement les critères de basculement non pas été bien définis.

4.6.L'algorithme AVE

Il est inventé par Hiroshi Noborio, Ryo Nogami et Satoru Hirao [21], cet algorithme est presque pareil à l'algorithme Alg1 sauf qu'il utilise les points de rencontre (*hit point*) et les points d'évitement (*leave point*) pour construire un arbre contenant ces points au lieu de construire une liste comme l'algorithme Alg1 ou Alg2.

Cet algorithme utilise l'arbre construite puis il applique quelques règles pour éviter les minimums locaux ou de passer en un chemin deux fois. Les trois règles utilisées par cet algorithme sont : (1) Evitement simple (*Simple avoidance*), (2) Evitement complexe (*Complex avoidance*) et (3) Evitement de boucle (*Loop avoidance*).

Ces trois règles sont tirées depuis l'arbre qui représente le chemin parcouru, de façon à ne pas faire des cycles dans le chemin.

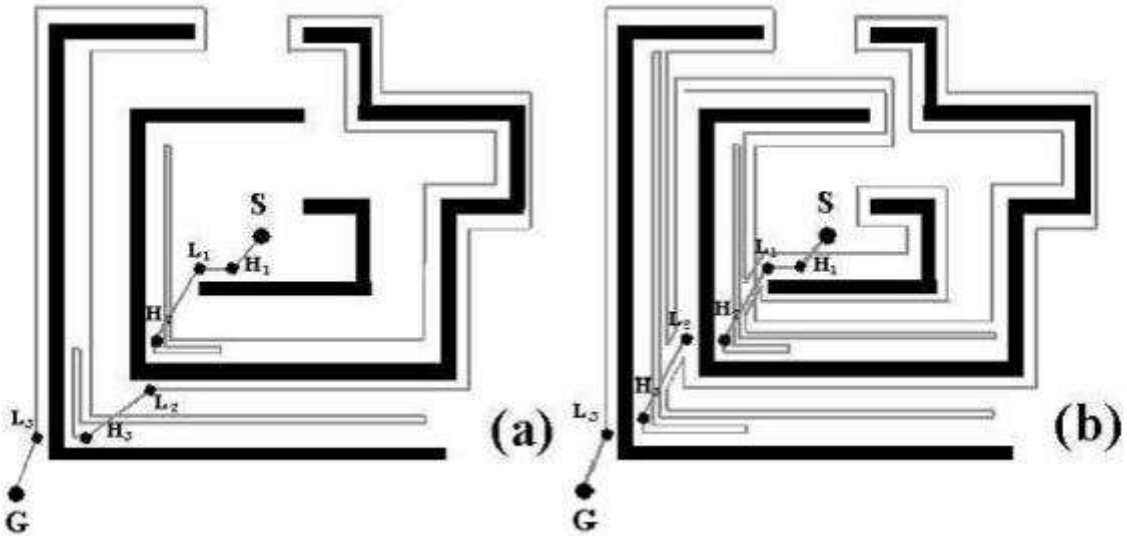

Figure 39: Exemple d'application de l'algorithme ave : (a) ave/wo (b) ave/lo (image tirée de [21]).

Cet algorithme arrive à trouver une solution, aussi à l'optimiser néanmoins en plus de tous les règles données par les auteurs des cet algorithmes il reste des améliorations et optimisations à faire (voir plus loin l'application de notre approche sur ce même exemple de la Figure 39).

4.7.L'algorithme DISTBUG

Inventé par Kamon and Rivlin en 1997 [10].

Le principe de cet algorithme est de choisir à chaque fois le point qui donne la distance minimum vers la destination, et dans le cas de non existence d'un tel point (ex : minimum local) il se comporte comme l'algorithme Bug1 en suivant la circonférence de l'obstacle. Lors du comportement comme Bug1 la distance vers la destination commence à s'augmenter le robot ne retourne à son premier comportement qu'au point ou cette distance commence à se diminuer.

4.8.Algorithme TangentBug

Cet algorithme est la meilleure amélioration des algorithmes de la famille Bug, inventé par Kamon and Rivlin.

Cet algorithme est utilisé (avec une extension) par NASA dans le robot Rokey7 (le robot explorateur de la planète Mars en 2002) [11], idée proposée en 1992 [25].

Le principe de cet algorithme est de trouver un chemin minimum local afin de faire avancer le robot vers la destination ou lui sortir d'un minimum local.

4.8.1. La version de base de l'algorithme Tangent BUG

Cet algorithme dans sa première version [49] n'était pas déterministe et peut y avoir des cas d'échec, ci-dessous un exemple (Figure 40).

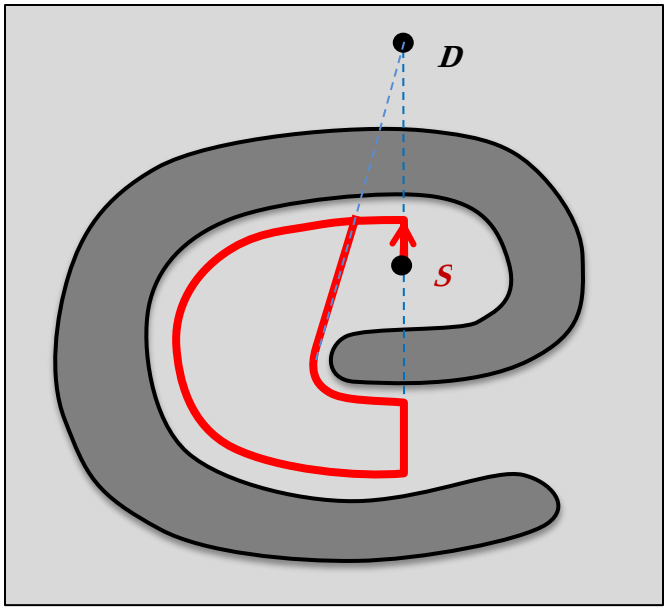

Figure 40 : Exemple d'un cas d'échec de l'algorithme TangentBUG.

Comme décrit dans [49], dans ce cas l'algorithme va s'arrêter et retourner (Destination injoignable = *The goal is not reachable*).

4.8.2. La version finale de Tangent BUG

Une amélioration apportée à cet algorithme lui rendant convergeant, c'est en changeant la condition de dépassement des obstacles, et le comportement du robot dans le cas d'augmentation successive de la distance amenant vers la destination [12].

Lors de l'augmentation successive de la distance entre le robot et la destination, le robot change son comportement en Bug1, ce qui fait qu'il suit la circonférence de l'obstacle jusqu'au moment où cette distance commence à diminuée il retourne à son comportement initial.

4.9.K-BUG

Il est inventé en 2007 par Ricardo A. Langer, Leandro S. Coelho et Gustavo H. C. Oliveira [13] [37]. Dans sa deuxième forme algorithme n'est pas destiné à la recherche de trajectoire dans un environnement inconnu malgré qu'il est cité comme un d'eux, parce

qu'il exige que l'environnement soit connu d'avance ce qui contredit avec un des principaux propositions de cette famille d'algorithmes [37].
L'algorithme K-Bug a deux formes :

1. **Première forme de K-Bug** : Le principe de K-Bug dans cette forme est d'avancer chaque fois vers le coin le plus proche puis continu son chemin vers la destination.
2. **Deuxième forme de K-Bug** : Dans cette forme l'algorithme considère la ligne (source, destination), cette ligne traverse les obstacles et coupe leurs circonférences en pairs de points (point d'entrée) et (point de sortie). Le coté ayant le petit périmètre sera élu. Dans ce côté le coin le plus loin de la ligne (coin en cour qui est la source, destination) sera ajouté à la liste des points qui forment la trajectoire du robot. Cette action sera répéter jusqu'à où la trajectoire résultant ne traverse aucun obstacle.

Malgré les bons résultats fournis par cet algorithme, néanmoins les résultats obtenus dépondent de l'environnement de simulation. L'utilisation de cet algorithme dans un environnement inconnu aura comme effet comme si la direction est choisie d'une façon aléatoire à cause de son principe erroné, un contre-exemple est donné dans la figure suivante (Figure 41).

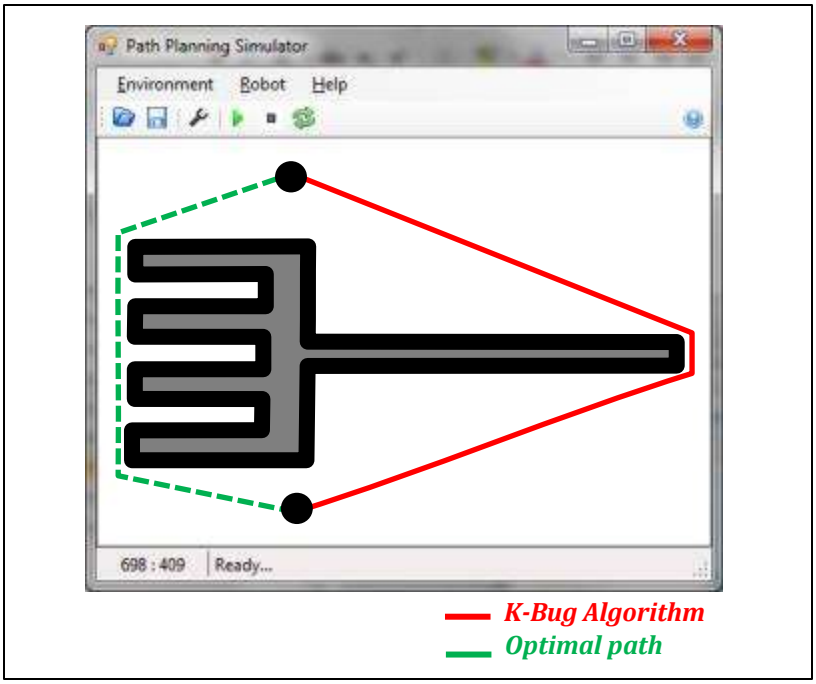

Figure 41: Exemple où l'algorithme K-Bug ne peut pas assurer le chemin optimal.

Même si cet on applique cet algorithme dans sa deuxième forme avec une connaissance complète de l'environnement le résultat reste le même et toujours non optimal.

Cet exemple montre que principe de K-Bug n'assure pas le chemin minimum.

4.10. Point BUG

Il est inventé en 2011 par Buniyamin N et al. [14].

Malgré que cet algorithme soit cité comme un algorithme de la famille Bug, il utilise un principe tout à fait différent dans la recherche de trajectoire.

Cet algorithme est purement théorique pour deux raisons : le premier parce qu'il suppose que les capteurs utilisés dans le robot sont à distances illimitées ce qui est inexistant (du moins avec la technologie existante à ce moment).

En plus du fait que cet algorithme est théorique, il possède beaucoup de points faibles et des fois il n'arrive pas à trouver le chemin vers la destination malgré son existence, il arrive aussi dans des cas où il tourne en infini (que soit il existe un chemin menant vers la destination ou non).

Cet algorithme sera étudié profondément dans le chapitre suivant.

4.11. L'algorithme Dynamic Point Bug

Il est inventé en 2013 par B. Margaret Devi et Prabakar S [16].

Cet algorithme est basé sur la réorientation permanente de la direction du robot (dans tous les cas), et à chaque rencontre d'un obstacle le robot commence à l'éviter en tournant à droite ou à gauche et continue son chemin.

La réorientation du robot se base sur des calculs trigonométriques afin de choisir la meilleure direction à chaque instant (voir Figure 42).

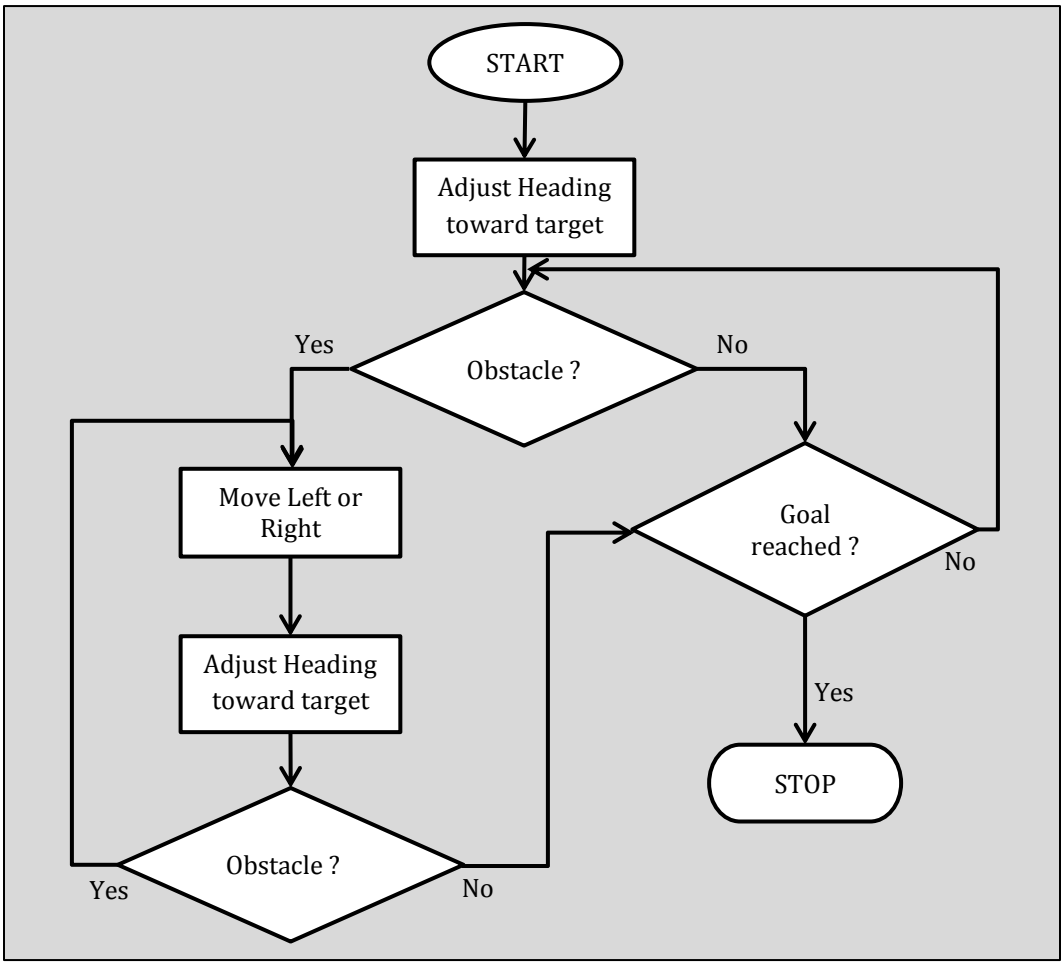

Figure 42: L'algorithme BUG Point dynamique (Tiré de [16]).

Cet algorithme contient deux points faibles :

1. La réorientation du robot lors de l'avancement vers la destination n'a aucun effet, puisque la direction reste la même (en ligne droite), ce sont des traitements inutiles. Dans l'exemple ci-après (Figure 43) la réorientation entre le point (*S*) et le point (*A*) est inutile, aucun changement de direction ne sera appliqué.
2. La réorientation permanente du robot lors de l'évitement des obstacles peut engendrée une oscillation infinie dans le cas d'un minimum local. Dans l'exemple ci-après (Figure 43), le point (*A*) présente un minimum local quel que soit la direction de la réorientation du robot (vers la droite ou vers la gauche), on aura une oscillation infinie (pour éviter ce cas le robot doit suivre la circonférence).

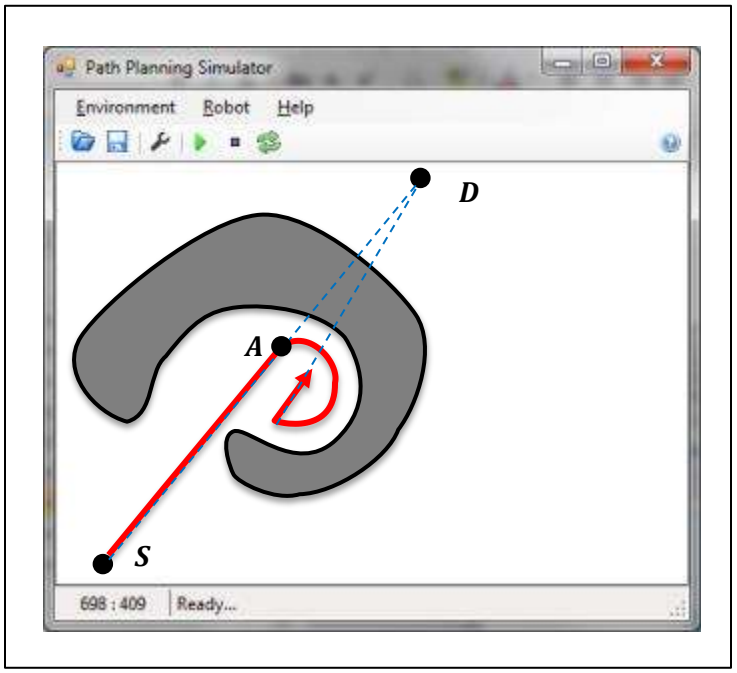

Figure 43: Exemple d'un cas où l'algorithme de point BUG oscille infiniment.

Chapitre 5

Approche proposée

“Si les gens savaient combien je travaille dur pour acquérir ma maîtrise, ça ne leur semblerait pas, après tout, tellement merveilleux.”

Michel-Ange

5. Approches proposées

Dans le cadre de notre travail de magistère, on a proposé une nouvelle approche

5.1. L'algorithme Point Bug

L'algorithme Point Bug repose sur une nouvelle définition c'est « les points soudains ».

1. Définition d'un Point soudain

Un point soudain est définie comme étant le point au quelle la distance détectée par le capteur change soudainement [14]. Il existe trois cas :

a) La distance détectée par le capteur change de l'infini à une certaine distance.
b) La distance change d'une façon inverse, d'une certaine distance à l'infini, voir Figure 44 le point (D)/(E).
c) La distance saute soudainement d'une certaine valeur à une autre valeur, voir Figure 44 le point (A)/(B).

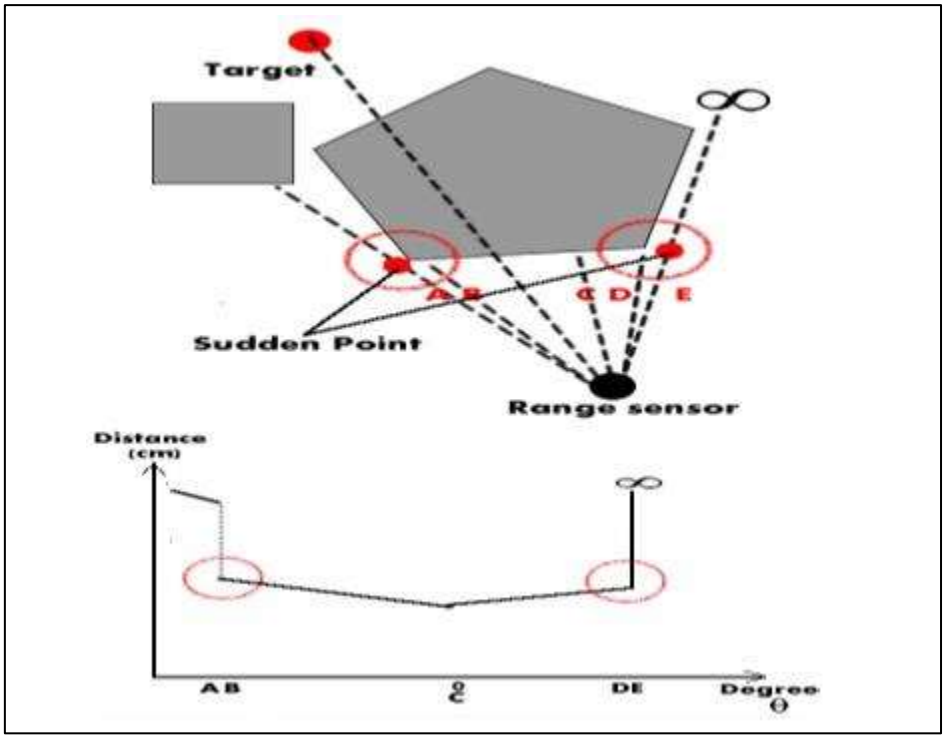

Figure 44: Différents types des points soudains (image tiré depuis [14]).

La différence du changement dans le troisième cas est définie préalablement lors de la simulation de l'algorithme.

2. Détail de l'algorithme Point Bug

Le principe de fonctionnement de l'algorithme Point Bug est décrit comme suit :

La position initiale du robot est en face de la direction de la destination, puis le robot tourne vers la droite ou vers la gauche pour chercher le premier point soudain.

Après avoir trouvé le premier point soudain, la direction d'avancement du robot est en ligne directe vers ce point soudain.

Lors de l'arrivé à chaque point soudain, le robot cherche le point soudain suivant en commençant de la direction d'avancement courante. Si un nouveau point soudain est détecté le robot continue son chemin, sinon il retourne au point précédent et continue la recherche de la même façon.

A noter que durant toute cette procédure l'angle de recherche ne doit pas dépasser 180° exception faite pour le point de départ dont la limite est de 360°.

Exemple de trajectoire d'un robot selon Point Bug :

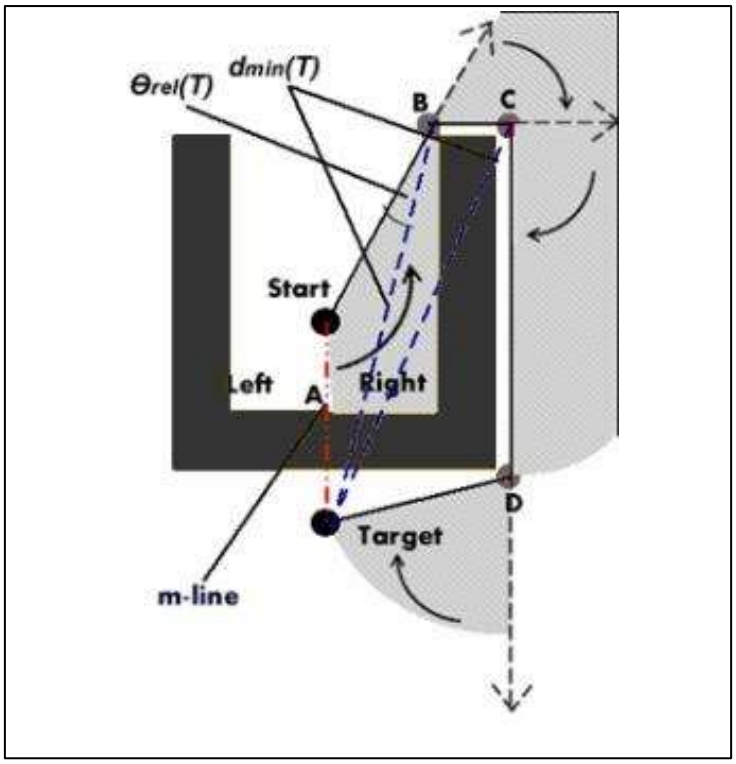

Figure 45: Trajectoire généré par l'algorithme Point Bug (image tiré depuis [14]).

Dans cet exemple (Figure 45), le robot commence la recherche du premier point soudain depuis le point (*A*). Le robot tourne à droite (choix fait par l'auteur de l'article) à la recherche du point soudain suivant dans ce cas c'est le point (*B*). Le robot avance vers le point (*B*). En arrivant à ce point, le robot commence la recherche du point soudain suivant en tournant à partir de la direction d'avancement courante qui est la ligne (Start, B). Le prochain point soudain est le point (*C*).

L'algorithme applique le même traitement avec ce point (*C*). En arrivant au point (*D*), le robot commence la recherche du point soudain suivant, il trouve alors la destination. Le robot avance vers ce point et l'algorithme s'arrête.

5.2.Problèmes de l'algorithme Point Bug

5.2.1. Tours infinis

Les tours infinis faites par cet algorithme peuvent être classées selon leurs causes en deux types :

1. **Limitation de l'angle de détection à 180°** : Cette condition est imposée par l'auteur de l'algorithme afin d'éviter le traitement des points soudains plusieurs fois. Malgré cette condition, il peut arriver que l'algorithme traite des points soudains plusieurs fois ou même une infinité de fois et tombe dans le cas d'une oscillation infinie.

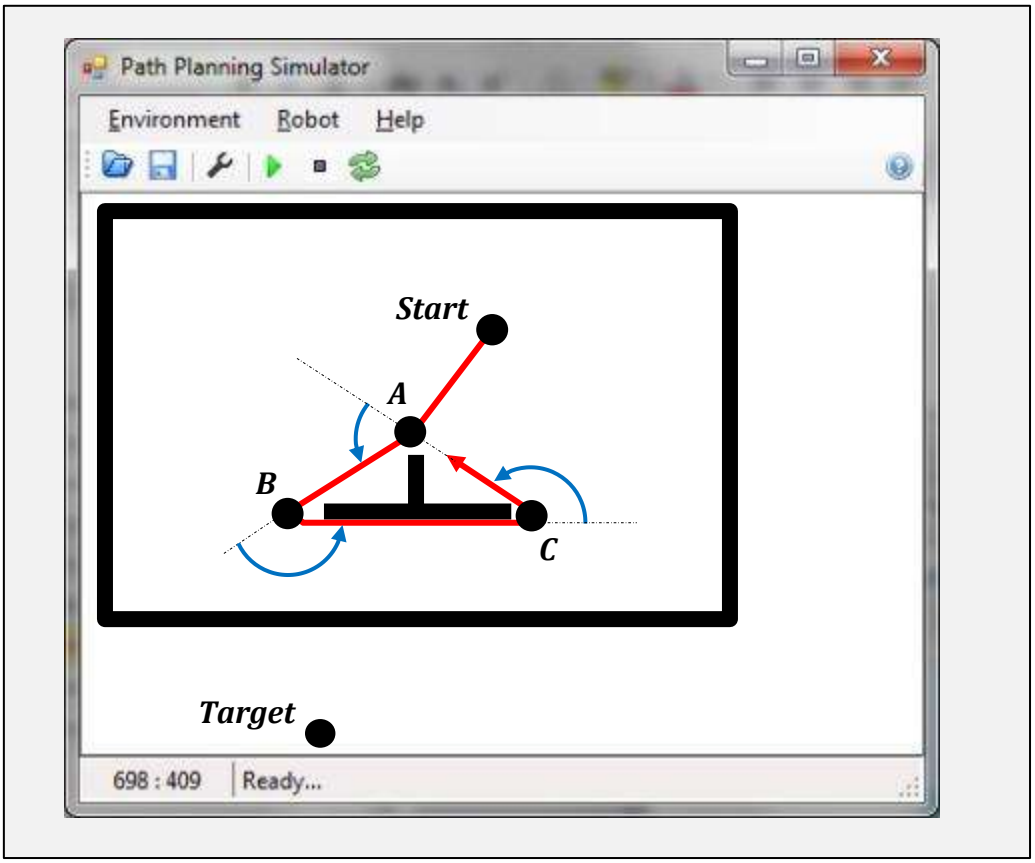

Figure 46: Exemple d'oscillation infinie dans l'algorithme Point Bug.

Dans cet exemple (Figure 46), l'application de l'algorithme Point Bug produit un cycle infini entre les trois points (*A*), (*B*) et (*C*). La cause de ce cycle est premièrement la limite de 180°, cette limite est affranchie ici parce que la somme des angles dans les coins A, B et C dépasse cette limite. La deuxième cause est l'absence des tests d'arrêts, l'algorithme a dû s'arrêter juste en atteignant le point (*A*) lors de la deuxième fois (ou du moins changer la stratégie).

2. **Utilisation des capteurs à distance illimités** : Contrairement à ce qu'il peut apparait, l'utilisation des capteurs à distance illimitée peut ne pas fournir des résultats optimaux dans plusieurs cas. Pire encore ils peuvent être la source de quelques problèmes.

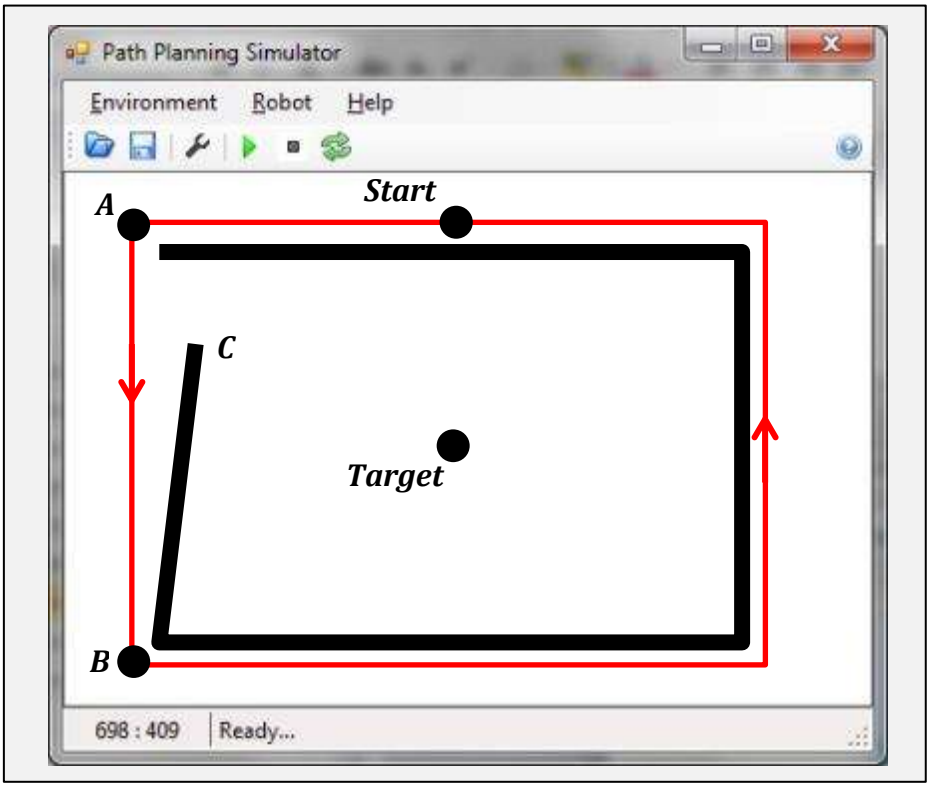

Figure 47: Exemple où l'algorithme Point Bug n'arrive pas à trouver la destination.

Dans cet exemple (Figure 47), En arrivant au point (*A*), au lieu d'avancer directement vers la destination le robot avance vers le prochain point soudain détectée qui sera le point (*B*). Le point (*C*) ne peut jamais être détecté à cause de la stratégie de l'algorithme qui détecte les points les plus loin d'abord.

5.2.2. Omission des chemins secondaires

Vue que cet algorithme suppose que les capteurs utilisés sont à distances illimitées, les points soudains les plus loin d'abord, et les points soudains les plus proches seront traiter après. De ce fait il peut arriver que l'algorithme Point Bug omit des chemins secondaires, un exemple est donné dans la section précédente (Figure 47).

5.2.3. Tests d'arrêts

Dans cet algorithme le seul test d'arrêt est le traitement de tous les points soudains dans un angle de 180°.

Deux cas peuvent se présentées :

1. L'absence totale des points soudains, si dans l'environnement de simulation il n'existe aucun point soudain l'algorithme ne finira jamais.

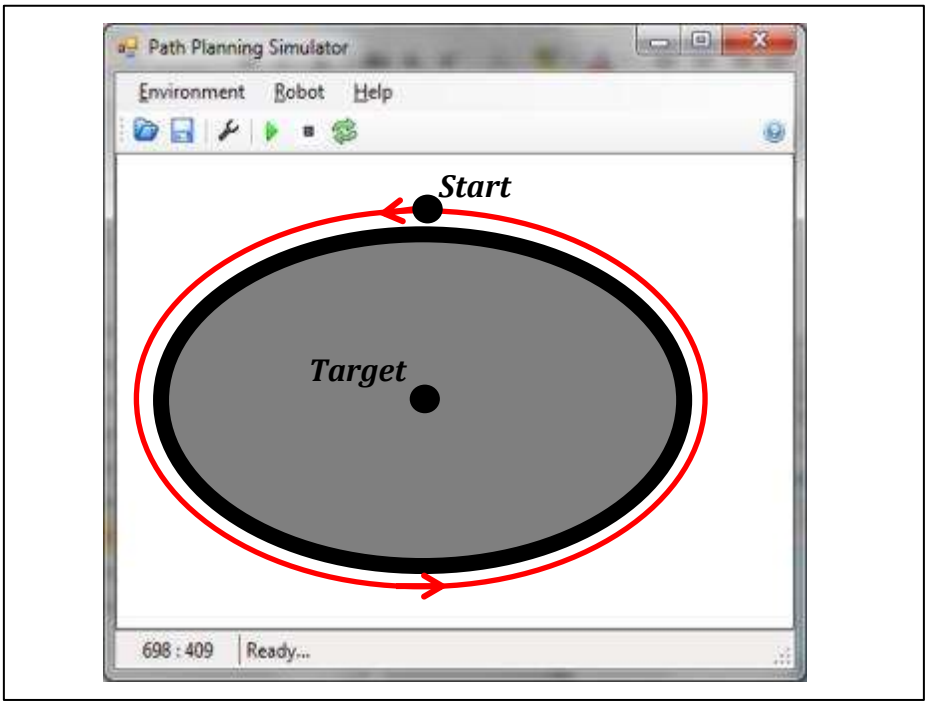

Figure 48: Cas où il n'existe pas de tests d'arrêt.

2. Il n'existe pas de test si un point est traité pour une deuxième fois, donc il peut exister des cycles dans la trajectoire.

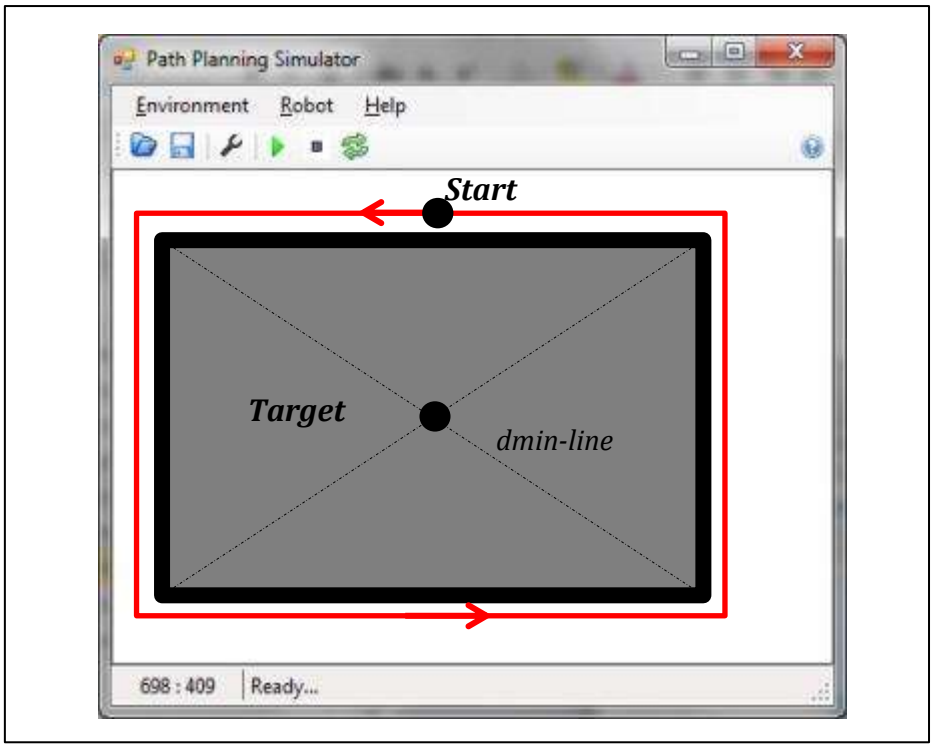

Figure 49: Cas où il n'existe pas de tests d'arrêt.

5.3.Solution proposée

La solution proposé est un nouveau algorithme P* (*P Star*) [76], en solvant les problèmes du Point Bug et introduisant des améliorations. Pour cela, en va enregistrer les points soudains rencontrés et à chaque fois où un point soudain est rencontré pour une deuxième fois tous les points rencontrés entre ces deux passes seront mises dans une liste de rejet et ignorer par l'algorithme. Le premier point de rencontre (*hit-point*) d'obstacle sera traité comme un point soudain jusqu'à la rencontre d'un point soudain.

Si depuis un point soudain aucun nouveau point soudain ne peut être détecté alors ce point sera mis dans la liste de rejet et le robot retourne vers le point soudain qui précède ce point. L'algorithme s'arrête si la destination est atteinte ou si tous les points soudains sont traités.

Algorithme P*

Début

A. FindNextDirection et avancer dans la direction sélectionnée, jusqu'à ce qu'un des évènements suivant arrive:

(1) The robot atteigne la destination Stop.

(2) Le premier point soudain est rencontré une deuxième fois et tous les points soudains sont traités. Stop, la destination ne peut être atteinte.

(3) Le robot doit respecter les règles suivantes:

- Si aucun point soudain n'est rencontré encore traiter le premier *hit-point* comme un point soudain.
- L'ajustement de l'angle de direction ne doit pas dépasser une certaine valeur défini lors du suivit du périmètre d'un obstacle (paramètre prédéfini).
- Ne retourne pas en arrière sauf dans le cas où il est impossible d'avancer.
- Le point soudain en cour est toujours traité ignore tous les points soudains qui suivent.

(4) Si un nouveau point soudain est détecté aller à (A)

Fin

FindNextDirection

If aucun soudain point ne peut être trouver Then /* capteur à distance limitée */

Le robot essaye au maximum d'avancer vers la destination dans une ligne droite,

if le robot commence à s'éloigner de la destination alors suit le périmètre de l'obstacle en cour

Else

Depuis le point soudain en cour énumérer tous les points soudains qui peuvent être détectés puis avancer vers le premier d'entre eux.

Endif

If un point soudain est toujours traité ignore le et passer au point suivant.

End

5.4.Simulation

Dans cette section, on va présenter les résultats de simulation générés par notre approche ainsi que des comparaisons faites avec les algorithmes les plus puissants parmi ceux présentés dans le chapitre précédent.

5.4.1. Simulateur utilisé : Path Planning Simulator

Dans le cadre de ce travail de magistère, un environnement de simulation a été réalisé en utilisant le langage C# (Microsoft Visual Studio 2012) et qui peut être recompilé sous Linux en utilisant l'environnement (Mono).

Cet environnement (nommé : **Path Planning Simulator**) est réalisé d'une façon que la simulation ne soit pas limitée seulement à l'algorithme (P*). Pour cela il suffit d'écrire une classe de l'algorithme qu'on veut simuler puis l'intégrer dans cet environnement.

Figure 50: L'environnement de simulation réalisé (Path Planning Simulator).

Ce simulateur permis à l'utilisateur plusieurs fonctionnalités :

- Charger une carte d'environnement sous forme d'image.

- Enregistrer le résultat de simulation sous forme d'image.
- Ajouter plusieurs robots et les simuler en même temps.
- Arrêter la simulation, faire des pauses, ou réinitialisé la simulation.
- Modifier les caractéristiques des capteurs du robot ainsi que certains paramètres de l'algorithme P* (direction de recherche par défaut, distance max et min des capteurs, seuil de distance pour les points soudains).

Figure 51: Fenêtre des paramètres du simulateur.

Pour simuler un algorithme en utilisant ce simulateur il suffit d'écrire une classe en C# qui doit implémenter quelques fonctions pouvant être appelées par le simulateur:

1. Initialisation : cette fonction est appelée lors du démarrage d'une simulation d'un robot sa déclaration est :

   public void init(Point srcPoint, Point distPoint){}

2. La position suivante du robot : cette fonction est appelée à chaque pas du robot, sa déclaration est :

   public Point getNextPosition(){}

3. Cette classe doit contenir trois variables de type Point :

   startPoint, targetPoint et **currentPoint.**

Reste à noter que ce simulateur est en phase de développement alpha, il reste à faire plusieurs améliorations et à introduire d'autres fonctionnalités.

5.4.2. Point Bug

Les simulations présentées ici sont divisées en deux cas et cela pour les problèmes présentés par l'algorithme Point Bug :

1. **Cas d'omission des chemins secondaires**

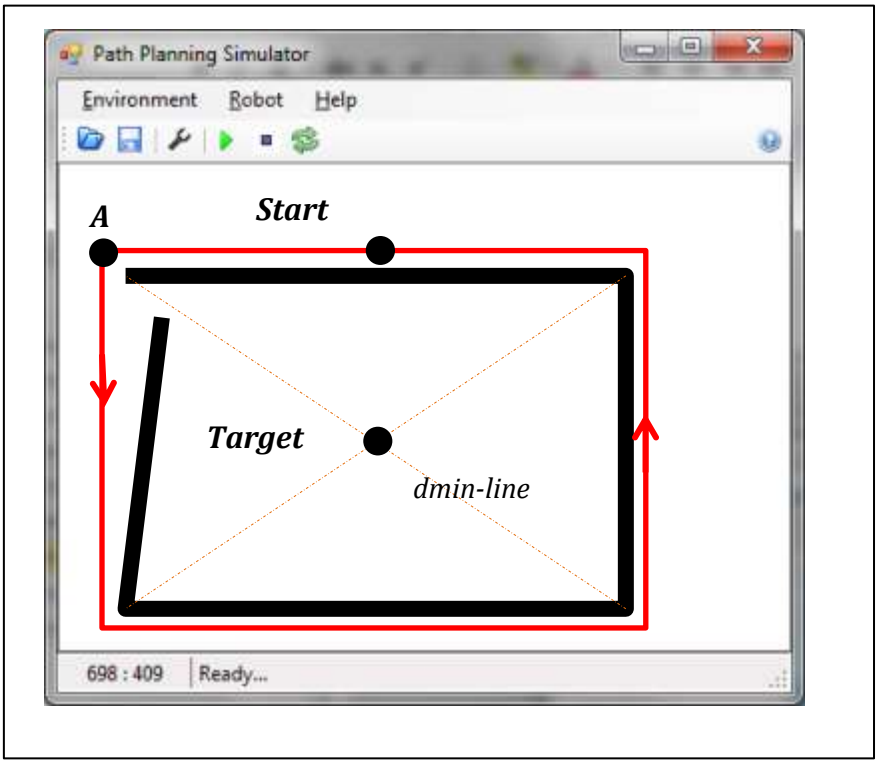

Figure 52: Simulation de l'algorithme Point Bug dans un cas de destination non atteignable.

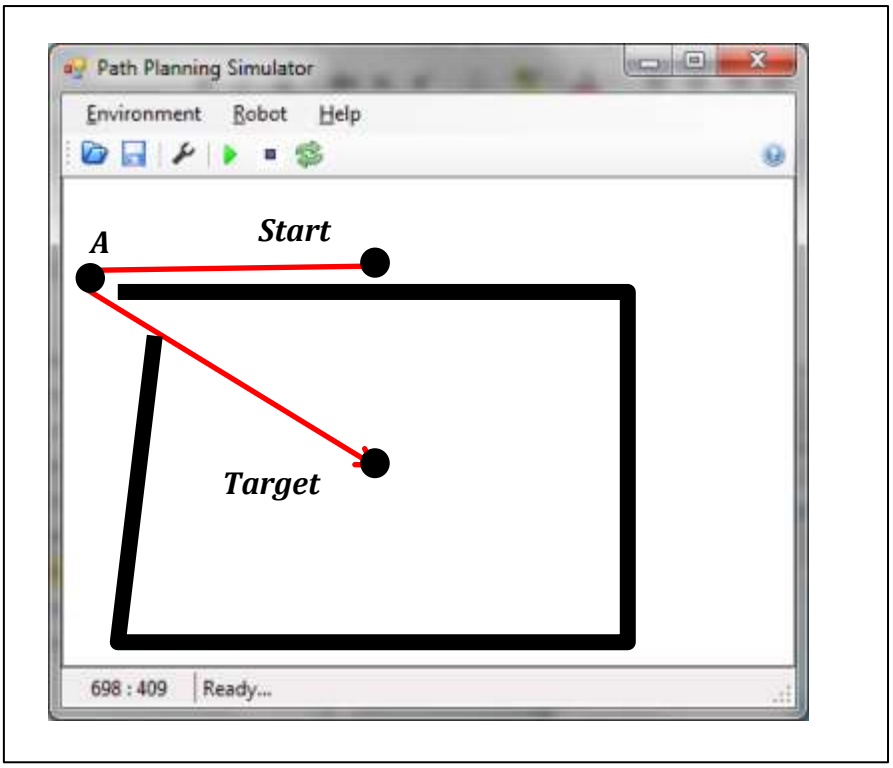

Figure 53: Simulation de l'algorithme P* dans le même environnement précédent en utilisant un robot avec des capteurs de très faibles distances.

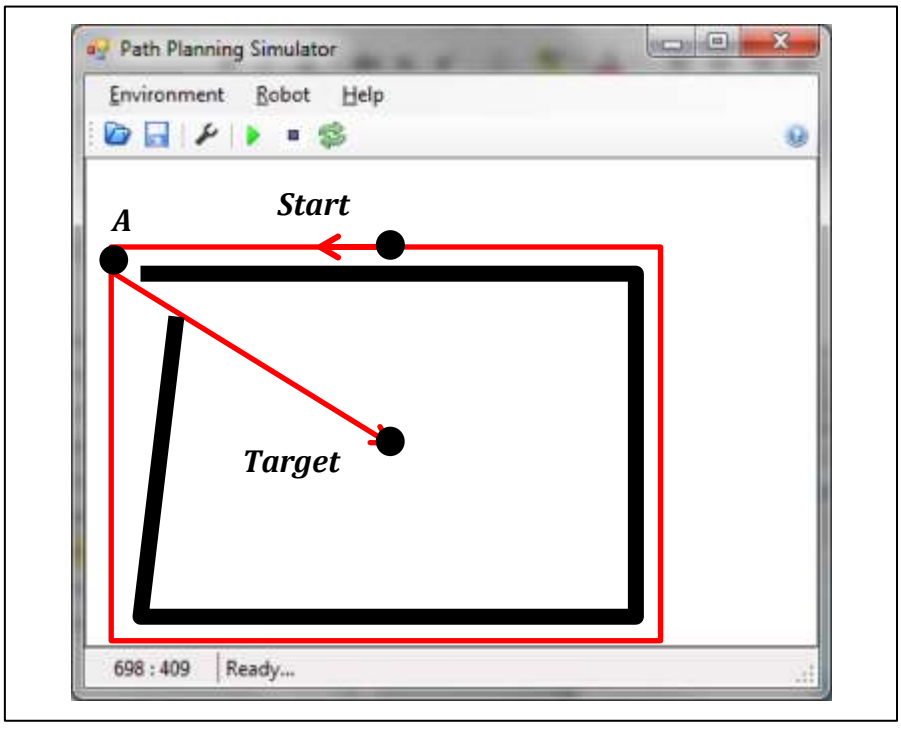

Figure 54: Simulation de l'algorithme P* dans le même environnement précédent en utilisant un robot avec des capteurs de grandes distances

Dans cet exemple il est très clair que l'algorithme Point Bug n'arrive pas à trouver le chemin vers la destination (Figure 52). Contrairement à ça, l'algorithme proposé P* arrive toujours à atteindre la destination soit directement (sans faire un tourné au tour de l'obstacle) soit dans le cas où le capteur utilisé détecte une distance très faible (Figure 53), et aussi après avoir fait un tour complet si cette distance de détection est plus grande (Figure 54).

2. Cas de destination non atteignable

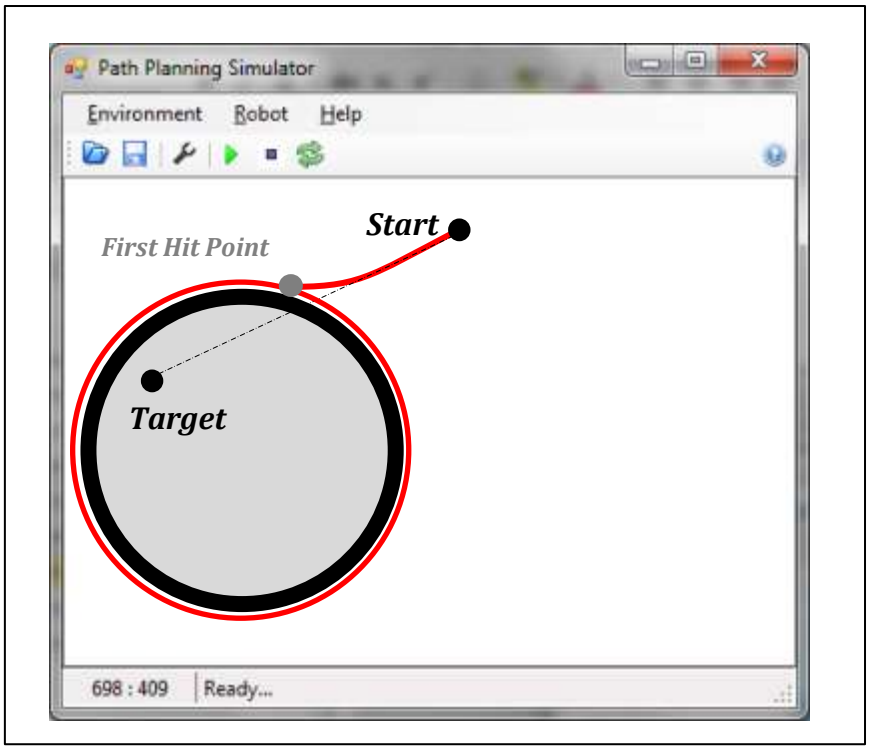

Figure 55: Simulation de l'algorithme P* dans un cas où l'algorithme Point Bug présente une boucle infinie.

Dans l'exemple ci-dessus (Figure 55), si on applique l'algorithme Point Bug le robot continue à tourner infiniment au tour de la destination, par contre l'algorithme P* s'arrête de tourner après un seule tour.

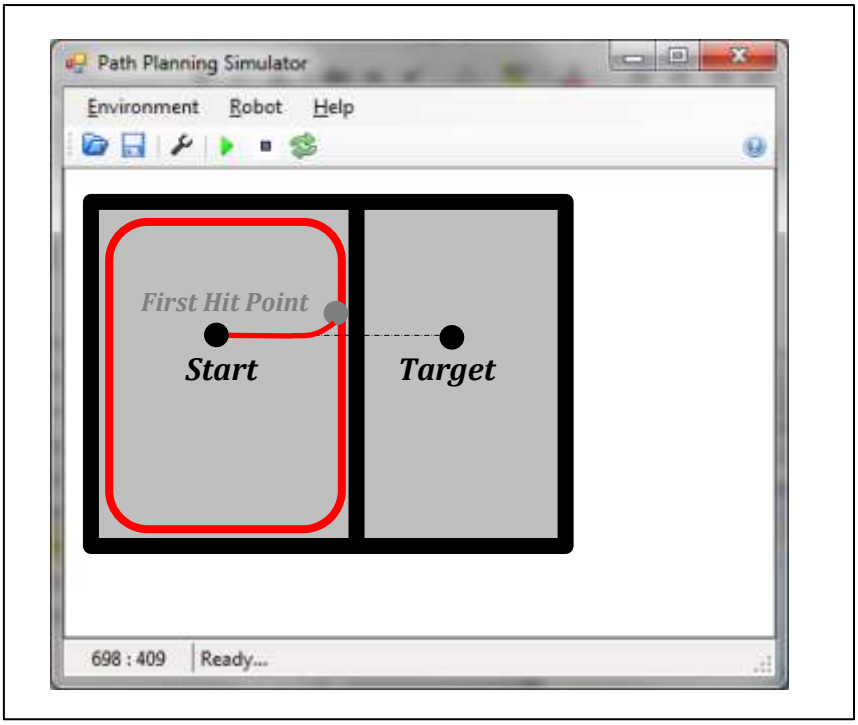

Figure 56: Simulation de l'algorithme P* dans un cas où aucun point soudain ne peut être détecté.

Dans cet exemple (Figure 56), l'algorithme P* arrive au point de rencontre avec l'obstacle puis si la distance du capteur est aussi grande le robot s'arrête sinon il fait un tour complet puis s'arrête et renvoi « destination non atteignable ». Ici l'algorithme Point Bug ne peut détecter aucun point soudain au premier lieu et continu à tourner infiniment.

5.4.3. Ave

L'algorithme Ave étant un des algorithmes les plus puissants surtout dans le cas de labyrinthe (*maze*). On va présenter ici une comparaison entre cet algorithme en appliquant tous ses règles d'évitement et l'algorithme P*.

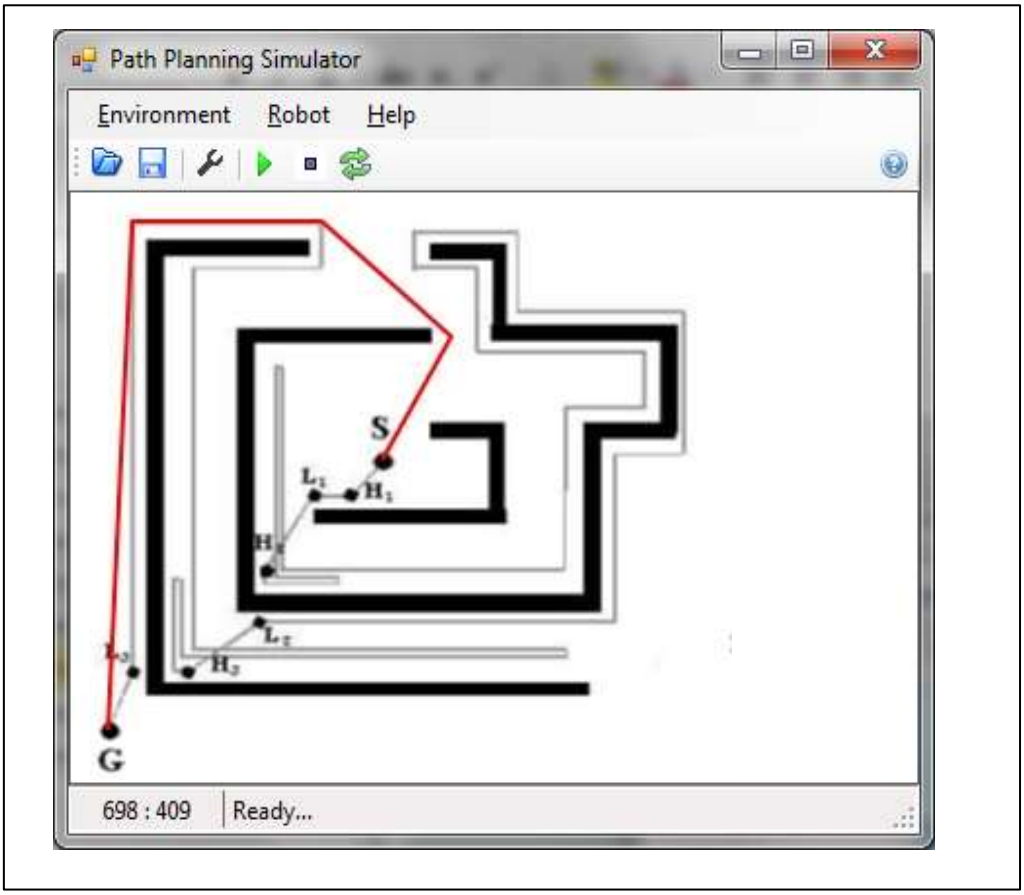

Figure 57: Comparaison entre l'algorithme P* et ave.

Cet exemple (Figure 57) montre que même après l'application de tous les règles de l'algorithme Ave l'algorithme P* arrive à fournir des résultats plus optimaux.

5.4.4. Tangent Bug

Dans cette section on va présenter une comparaison entre les meilleurs résultats fournis par l'algorithme Tangent Bug et l'algorithme P*.

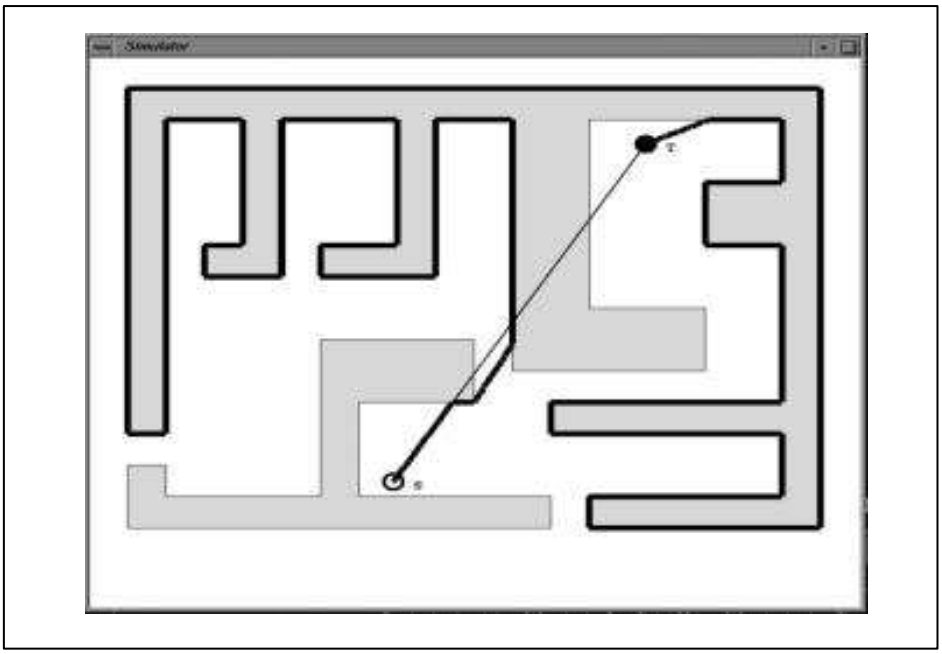

Figure 58: Simulation de l'algorithme Tangent Bug dans un environnement World2, ici le robot utilise un capteur à distance très faible (simulation tiré de [12]).

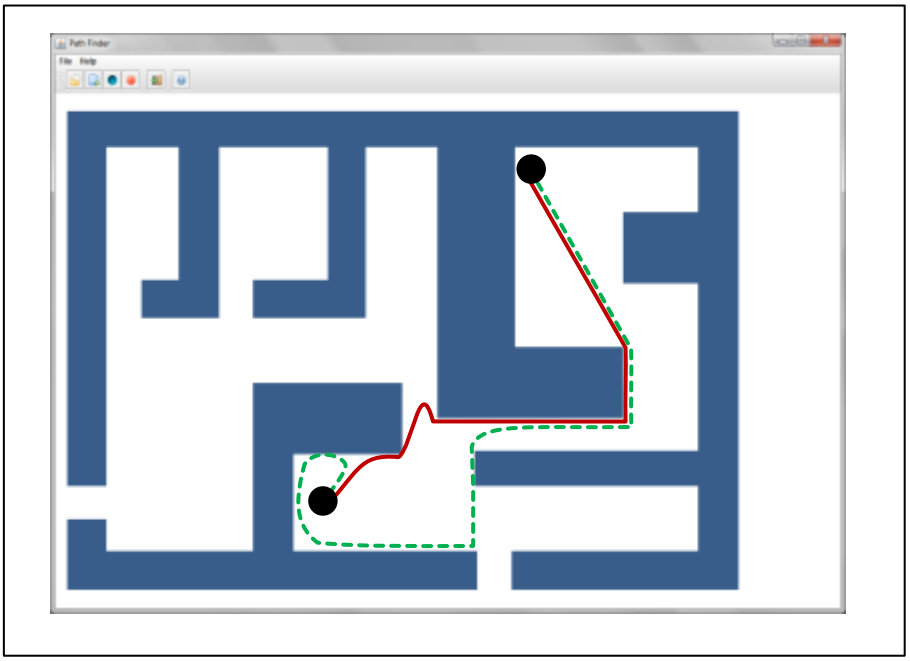

Figure 59: Simulation de l'algorithme P* dans le même environnement précédent et avec un robot de mêmes caractéristiques.

La différence entre les deux résultats est très claire. En appliquant l'algorithme P* (Figure 59) et quel que soit la première direction du robot à droite (trajectoire en

rouge continue) ou à gauche (trajectoire en vert pointillé). La longueur de la trajectoire est inférieur à celle générer par l'algorithme Tangent Bug (Figure 58).

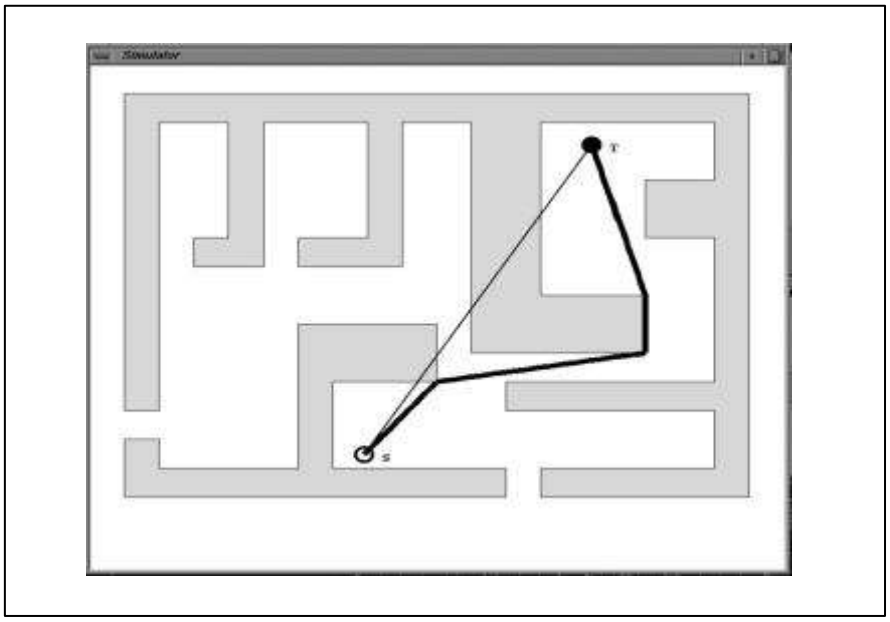

Figure 60: Simulation de l'algorithme Tangent Bug dans le même environnement en utilisant des capteurs à distance infinies (simulation tiré de [12]).

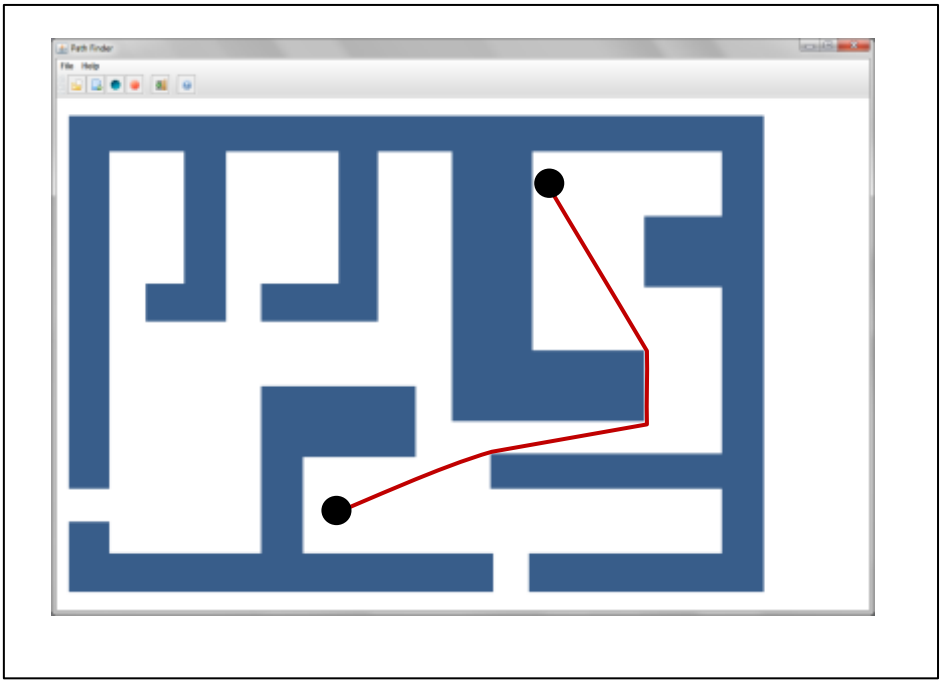

Figure 61: Simulation de l'algorithme P* dans le même environnement en utilisant des capteurs à distance illimités.

Le tableau suivant montre en valeur la différence entre les résultats générés par les deux algorithmes.

Distance de capteur	Tangent Bug	P*
Zéro (0)	9452	Droite: 993 Gauche: 1401
Infinie (∞)	733	583

Résultats de simulations générées par les différents algorithmes (mm).

Les deux derniers résultats de simulation (Figure 60 et Figure 61) montrent une caractéristique très importante de l'algorithme P* : c'est la stabilité de cet algorithme par rapport aux autres algorithmes. Malgré le changement des caractéristiques des capteurs utilisés dans le robot le chemin généré par l'algorithme P* ne change pas beaucoup (ni au niveau de sa longueur ni de sa forme.

Chapitre 6

Conclusion et perspectives

“Si on croit être arrivé, c’est qu’on n’avait pas l’intention d’aller plus loin.”

Claude Chabrol

6. Conclusion et perspectives

6.1. Conclusion générale

Dans le travail présenté dans cette thèse, on présenté la plupart des algorithmes Bug existant, ainsi que leurs problèmes, failles et points forts. Basant sur étude détaillée nous avons pu proposer un nouveau algorithme P*, qui est une implémentation en monde réel d'un algorithme très puissant mais qui est théorique, en plus des améliorations effectuées à cette algorithme.

Les résultats de simulation de l'algorithme proposé P* prouvent sa puissance, ainsi que sa stabilité par rapport aux algorithmes Bug les plus puissant qui existent actuellement.

Les résultats fournis par l'algorithme proposé (P*) prouvent que malgré le matériel de performance très faible P* a pu donner des résultats comparables et des fois plus meilleures que ceux des plus puissants algorithmes existants.

6.2. Perspectives

Beaucoup de perspective peuvent être envisagés suite à notre travail :

- La plupart des méthodes de navigation actuelles utilisent des capteurs qui renvoient la distance entre le robot et l'obstacle mais n'utilisent pas une vision réelle de l'environnement, cette vue (image) de l'environnement contient beaucoup d'autre informations très utiles, il serait très pratique et intéressant de les prendre en considération dans le travail futur.
- Faire introduire l'aspect d'environnement dynamique (obstacle en mouvement) dans l'approche proposée, cette approche telle qu'elle à donner des bonnes résultats mais serais-t-il le cas dans un environnement dynamique (il faut introduire d'autres tests, et résoudre les problèmes qui peuvent exister).
- Etendre les approches proposées pour un ensemble de robots, en utilisant les systèmes multi-agent et ajoutant l'aspect d'intelligence.

Bibliographie

“Le seul endroit où le succès précède le travail est le dictionnaire.”

Vidal Sassoon

7. Références

[1] V. Lumelsky et A. Stepanov. "*Navigation Strategies for an Autonomous Vehicle with Incomplete Information on the Environment*". General Electric Company Corporate Research and Development. No 84CRD070. Apr. 1984.

[2] V. J. Lumelsky et A. A. Stépanov. "*Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment*". IEEE Transactions on Automatic Control, Vol AC-31, No 11, Nov. 1986. Pages 1058-1063.

[3] Vladimir J. Lumelsky et Alexander A. Stepanov. "*Path-Planning Stategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape*". Algorithmica (1987) 2: pages 403-430.

[4] Lucas, C.; Lumelsky, V.; Stepanov, A. Comments on "*Dynamic path planning for a mobile automation with limited information on the environment*" [with reply]. IEEE Transactions on Automatic Control, Vol. 33, No 5, May. 1988. Pages 511-512.

[5] V. Lumelsky, "*Algorithmic Issues of Sensor-Based Robot Motion Planning"*. IEEE 26th International Conference on Decision and Control, Los Angles, December 1987. Pages 1796-1801.

[6] A. Sankaranarayanan et M. Vidyasagar. "*A New Path Planning Algorithm for Moving a Point Object Amidst Unknown Obstacles In A Plane"*. IEEE Conference on Robotics and Automation, Pages 1930-1936, 1990

[7] A. Sankaranarayanan et M. Vidyasagar. "*Path Planning for Moving a Point Object Amidst Unknown Obstacles In A Plane: The Universal Lower Bound On Worst Case Path Lenghts And A Classification of Algorithms"*. IEEE International Conference on Robotics and Automation, Sacramento, California, April 1991. Pages 1734-1741.

[8] A. Sankaranarayanan et M. Vidyasagar. "*Path Planning for Moving a Point Object Amidst Unknown Obstacles In A Plane: A New Algorithm And A General Theory For Algorithm Development"*. IEEE 29th International Conference on Decision and Control, Honolulu, Hawaii, December 1990. Pages 1111-1119.

[9] A. Sankaranarayanan et Isao Masuda. "*A New Algorithm For Robot Curve-Following Amidst Unknown Obstacles, And A Generalization Of Maze-Searching"*. IEEE International Conference on Robotics and Automation, Nice, France, May 1992. Pages 2487-2494.

[10] Ishay Kamon et Ehud Rivlin. "*Sensor-Based Motion Planning with Global Proofs"*. IEEE Transactions on Robotics and Automation, Vol 13, No 6, Dec. 1997. Pages 814-822.

[11] Terry Huntsberger, Hrand Aghazarian, Yang Cheng, Eric T. Baumgartner, Edward Tunstel, Chris Leger, Ashitey Trebi-Ollennu, and Paul S. Schenker. "*Rover Autonomy for Long Range Navigation and Science Data Acquisition on Planetary Surfaces"*. Proceedings of the 2002 IEEE 1 International Conference on Robotics & Automation Washington, DC May 2002

[12] Ishay Kamon, Elom Rimon et Ehud Rivlin. "*TangentBug: A Range-Sensor-Based Navigation Algorithm*". The international journal of robotics research. Vol. 17, No 9, Sept. 1998. Pages 934-953.

[13] Ricardo A. Langer, Leandro S. Coelho et Gustavo H. C. Oliveira. "*K-Bug, A New Bug Approach For Mobile Robot's Path Planning"*. 16th IEEE International Conference on Control Applications, Part of IEEE Multi-conference on Systems and Control. Singapore, Oct. 2007. Pages 403-408.

[14] Buniyamin N., Wan Ngah W.A.J., Sariff N., Mohamad Z. "*A Simple Local Path Planning Algorithm for Autonomous Mobile Robots"*. International Journal of Systems Applications, Engineering and Development. Issue 2, Volume 5, 2011. Pages 151-159.

[15] Bo Li, Jianwei Gong, and Yan Jiang, Hany Nasry, Guangming Xiong: "*ARA*+: Improved path planning algorithm based on ARA**". IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology. 2012. Pages 361-365.

[16] B. Margaret Devi et Prabakar S.: "*Dynamic Point Bug Algorithm For Robot Navigation"*. International Journal of Scientific and Engineering Research, Volume 4, Issue 4, Apr. 2013. Pages 1276-1279.

[17] Jianming GUO; Liang LIU; Qing Liu et Yongyu Qu: "*An Improvement of D* algorithm for Mobile Robot Path Planning in Partial Unknown Environment"*. IEEE 2009 Second International Conference on Intelligent Computation Technology and Automation. Pages 394-397.

[18] Dongmei Zhang, Min Xie, Gengyu Wei, Shidong Zhang: "*Improved A* Algorithm For Robot Path Planning In Unknown Environment"*. Proceedings of 2010 IEEE/IC-BNMT2010 International Conference on Broadband Network and Multimedia Technology. Pages 1154-1158.

[19] Matouš POKORNY: "*Collision-free Path Planning for Mobile Robots Based on Extended A* Algorithm"*. 16th International Student Conference on Electrical Engineering. Poster 2012, Prague May 17.

[20] Anthony Stentz: "*Optimal and Efficient Path Planning for Partially-Known Environments"*. Proceedings of the IEEE International Conference on Robotics and Automation *(ICRA '94)*, May 1994. Pages 3310-3317.

[21] Hiroshi Noborio, Ryo Nogami et Satoru Hirao: "*A New Sensor-Based Path-Planning Algorithm whose Path Length is Shorter on the Average"*. Proceedings of the IEEE International Conference on Robotics and Automation *(ICRA '04)*, May 2004. Pages 2832-2839.

[22] Vladimir J. Lumelsky: "*A Comparative Study on the Path Length Performance of Maze-Searching and Robot Motion Planning Algorithms"*. IEEE Transactions On Robotics And Automation, Vol. I, No. 1, Feb. 1991. Pages 57-66.

[23] Alpaslan Yufka et Osman Parlaktuna : "*Performance Comparison of Bug Algorithms for Mobile Robots"*. 5th International Advanced Technologies Symposium (IATS'09), May 13-15, 2009, Karabuk, Turkey.

[24] Geovanni Bianco, Riccardo Cassinis: "*Multi-Strategic approach for robot path planning in an unknown environment"*. Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT '96), 1996. Page(s): 108 – 115.

[25] Brian Wilcox; Larry Matthies; Donald Gennery; Brian Cooper; Tam Nguyen; Todd Litwin; Andrew Mishkin; Henry Stone. "*Robotic vehicles for planetary exploration"*. Proceedings of the IEEE International Conference on Robotics and Automation, 1992.

[26] Kamilah Taylor and Steven M. LaValle: "*I-Bug: An Intensity-Based Bug Algorithm"*. IEEE 2009 International Conference on Robotics and Automation, Kobe International Conference Center, Kobe, Japan, May 12-17, 2009. Pages 3981-3986.

[27] Hiroshi Noborio, Keiichi Fujimura, Yohei Horiuchi: "*A Comarative Study of Sensor-Based Path-Planning Algorithms in a Unknown Maze"*. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), 2000. Volume 2. Pages 909-916.

[28] S. M. Masudur Rahman Al-Arif, A. H. M. Iftekharul Ferdous, Sohrab Hassan Nijami: "*Comparative Study of Different Path Planning Algorithms: A Water based Rescue System*". International Journal of Computer Applications (0975 – 8887). Volume 39– No.5, February 2012.

[29] Gerardo Flores, Shuting Zhou, Rogelio Lozanoy, and Pedro Castillo: "*A Vision and GPS-Based Real-Time Trajectory Planning for MAV in Unknown Urban Environments*". 2013 International Conference on Unmanned Aircraft Systems (ICUAS). May 28-31, 2013, Grand Hyatt Atlanta, Atlanta, GA.

[30] Basem M. ElHalawany, Hala M.Abdel-Kader, Adly TagEldeen, Alaa Eldeen Elsayed, Zaki B.Nossair: "*Modified A* Algorithm for Safer Mobile Robot Navigation*". U2013 Proceedings of International Conference on Modelling, Identification & Control (ICMIC). Cairo, Egypt, 31 Aug.- 2 Sept. 2013.

[31] Metin Ozkan, Ahmet Yazici, Muzaffer Kapanoglu, Osman Parlaktuna: "*Hierarchical Oriented Genetic Algorithms for Coverage Path Planning of Multi-robot Teams with Load Balancing*". GEC'09, June 12–14, 2009, Shanghai, China.

[32] Yuming Liang, Lihong Xu: "*Mobile Robot Global Path Planning Using Hybrid Modified Simulated Annealing Optimization Algorithm*". GEC'09, June 12-14, 2009, Shanghai, China.

[33] Luis E. Gonzalez Moctezuma, Angelica Nieto L., Jose L. Martinez Lastra: "*A Genetic Algorithm for Optimizing Vector-based Paths of Industrial Manipulators*". 11th IEEE International Conference on Industrial Informatics. 2013.

[34] Pu Shi, Yujie Cui: "*Dynamic Path Planning for Mobile Robot Based on Genetic Algorithm in Unknown Environment*". 2010 Chinese Control and Decision Conference. pp 4325-4329. 2010 IEEE.

[35] Lin Lei, Houjun Wang and Qinsong Wu: "*Improved Genetic Algorithms Based Path planning of Mobile Robot Under Dynamic Unknown Environment*". Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation. Pages 1728-1732. June 25 - 28, 2006, Luoyang, China

[36] Michael A. Hurni. "*An information-centric approach to autonomous trajectory planning utilizing optimal control techniques"*. Ph.D. thesis, Naval Postgraduate School, Monterey, California. Sept. 2009.

[37] James Ng. "*An Analysis of Mobile Robot Navigation Algorithms in Unknown Environments"*. Ph.D. thesis, School of Electrical, Electronic and Computer Engineering. Feb. 2010.

[38] Carlos Favis Ezequiel: "*Real-Time Map Manipulation for Mobile Robot Navigation"*. Ph.D. thesis, University of South Florida. March 2013.

[39] Braden Edward Stenning. "*Path/Action Planning for a Mobile Robot"*. Graduate Department of Institute for Aerospace Studies, University of Toronto. 2013.

[40] Walter Hangartner. "*Navigat-Soar a path-finding agent using Soar"*. Ph.D. thesis, University of Zürich. Oct 1994.

[41] Michael Colin Hoy. "*Methods for Collision-Free Navigation of Multiple Mobile Robots in Unknown Cluttered Environments"*. Ph.D. thesis. School of Electrical Engineering and Telecommunications, University of New South Wales. January 2013.

[42] Yi Li : "*Real-Time Motion Planning of Multiple Agents and Formations in Virtual Environments"*. Thèse PhD. Université Simon Fraser. 2008.

[43] Marco Hutter: "*Design and Control of Legged Robots with Compliant Actuation*". Thèse PhD. Université ETH ZURICH. Mars 2013.

[44] Liam Paull: "*Robust Online Adaptive Sensor-Driven Survey Planning for Single and Multiple Autonomous Underwater Vehicles*". Thèse PhD. Université de New Brunswick. Octobre 2013.

[45] Thomas Moulard: "*Optimisation numérique pour la robotique et exécution de trajectoires référencées capteurs*". Thèse de doctorat. Université Toulouse 3 Paul Sabatier. Septembre 2012.

[46] Olivier Lefebvre : "*Navigation autonome sans collision pour robots mobiles non holonomes*". Thèse de doctorat de l'Institut National Polytechnique de Toulouse. 2006.

[47] Michael Wilson Otte: "*Any-Com Multi-Robot Path Planning*". Thèse PhD. Université de Colorado. 2011.

[48] C. B. Crous : "*Autonomous Robot Path Planning*". Thèse Master en Science. Université de Stellenbosch. 2009.

[49] Ishay Kamon, Elon Rimon et Ehud Rivlin. « *A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots"*. Technical Report CIS-9517, Technion, Department of Computer Science. Sept. 1995.

[50] Nageswara S. V. Rao, Srikumar Kareti, Weimin Shi et S. Sitharama Iyengar. "*Robot Navigation in Unknown Terrains: Introductory Survey of Non-Heuristic Algorithms"*. Technical Report ORNL/TM12410, Prepared by the OAK Ridge National Laboratory U.S Department of Energy and Robotics and Machine Intelligence Program National Science Foundation. Jul 1993.

[51] Robert J. Szczerba, Danny Z. Chen, John J. Uhran Jr.: "*A Grid-Based Approach For Finding Conditional Shortest Paths in an Unknown Environment"*. Technical Report: #94-34. Department of Computer Science and Engineering, University of Notre Dame, Indiana, USA. Nov 1994.

[52] *A Roadmap for U.S. Robotics From Internet to Robotics*. Distribution non classifie. Version approuvée pour public. U.S. Army RDECOM CERDEC NVESD. Release Date: March 6, 2012.

[53] David FILLIAT. "*Robotique Mobile"*. Cours C10-2. ENSTA: École Nationale Supérieure de Techniques Avancées ParisTech. Oct. 2004. http://www.ensta-paristech.fr/~filliat/ (dernière mis à jours: Oct 2011).

[54] J.C. Alvarez A. Shkel V. Lumelsky. "*Active Sensing In Sensor–Based Motion Planning With Dynamics*".

[55] Laëtitia Matignon: *"Introduction à la robotique"*. GREYC-CNRS. Université de Caen, Basse-Normandie. France. (http://lmatigno.perso.info.unicaen.fr/L1robotique). 2011.

[56] *Le petit Larousse Illustré*. Éditions Larousse 2007. ISBN 978-2-03-582503-2. Page 894

[57] O. Labbani-Igbida, E. Mouaddib: "*Une histoire de la robotique*". Cours Robots Future. Université Picardie. 2010.

[58] John J. Craig: "*Introduction to Robotics - Mechanics and Control*". Third Edition. Pearson Education International, Inc. Edition Pearson Prentice Hall. 2005.

[59] David Daney : "*Cours de robotique fondamental*". Projet Coprin. INRIA Sophia Antipolis. 2007.

[60] *Merriam-Webster Online Dictionary and Thesaurus*: http://www.merriam-webster.com/dictionary/robot

[61] CNRL: http://www.cnrtl.fr/. *Centre National de Ressources Textuelles et Lexicales*. http://www.cnrtl.fr/lexicographie/robot.

[62] *Dictionnaire Hachette*. Hachette éducation Edition 2013. ISBN 978-2-01-281493-6. Page 1407.

[63] Wikipédia. L'encyclopédie libre. http://fr.wikipedia.org/.

[64] Vikram Kapila: "*Introduction to Robotics*". 2003. http://mechatronics.poly.edu.

[65] Václav Hlaváč: "*Motion planning methods*". Czech Technical University in Prague. 2010

[66] Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Giorgio Grisetti, Kai Arras: "*Introduction to Mobile Robotics Path Planning and Collision Avoidance*". Université de Freiburg. 2010.

[67] Jong-Hwan Kim, Brian Keller, Brian Y. Lattimer: "*Sensor Fusion Based Seek-and-Find Fire Algorithm for Intelligent Firefighting Robot*". 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) Wollongong, Australia, July 9-12, 2013.

[68] Anjan Kumar Ray, Laxmidhar Behera and Mo Jamshidi: "*GPS and Sonar Based Area Mapping and Navigation by Mobile Robots*". 7th IEEE International Conference on Industrial Informatics (INDIN 2009). 2009. P801-806

[70] Lefteris Doitsidis, Stephan Weiss, Alessandro Renzaglia, Markus W. Achtelik, Elias Kosmatopoulos, Roland Siegwart, Davide Scaramuzza: "*Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard*". Springer Science+Business Media, LLC 2012. Published online: 31 March 2012. Pages 173-188.

[71] Constantin Purcaru, Radu-Emil Precup, Daniel Iercan, Lucian-Ovidiu Fedorovici, Bogdan Dohangie, Florin Dragan: "*nRobotic Mobile Robot Navigation Using Traffic Signs in Unknown Indoor Environments*". 8th IEEE International Symposium on Applied Computational Intelligence and Informatics • May 23–25, 2013 • Timisoara, Romania. pp29-34.

[72] László Somlyai: "*Mobil robot localization using RGB-D camera*". ICCC 2013 IEEE 9th International Conference on Computational Cybernetics • July 8-10, 2013. Tihany, Hungary. pp131-136.

[73] Dave Ferguson, Maxim Likhachev, and Anthony Stentz: "*A Guide to Heuristic-based Path Planning*". American Association for Artificial Intelligence (www.aaai.org). 2005.

[74] K. MARTI and S. QU: "*Path Planning for Robots by Stochastic Optimization Methods*". Journal of Intelligent and Robotic Systems 22: pages 117–127, 1998.

[75] Syed Atif Mehdi, Karsten Berns: "*Probabilistic Search of Human by Autonomous Mobile Robot*". ACM ISBN 978-1-4503-0772-7/11/05. PETRA'11, May 25-27, 2011, Crete, Greece.

[76] Farouk Meddah and Lynda Dib: "*P*: A new path planning algorithm for autonomous robot in an unknown environment*". Second International Conference On Advances in Computing, Electronics and Communication - ACEC 2014. Zurich, Switzerland. Pages 42-46.