UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Projet Tutoré M1
Département informatique
2015/2016

SYSTÈMES ROBOTIQUES

GENERATION DE TRAJECTOIRES, SUIVI DE TRAJECTOIRES

Réalisé par : Abdelbasset MARZAQ
Abdelhakim MOURIK

Encadré par : Mme Lynda SEDDIKI

TABLES DES MATIERES

Remerciements

Avant toute chose, il nous parait opportun de remercier tous ceux qui ont contribué à la réussite de notre recherche par leur implication et leur sens de responsabilité.

Nous tenons à remercier tout particulièrement notre tutrice Mme. Lynda SEDDIKI, qui nous a accompagné tout au long de cette période de projet, pour son suivi régulier, ses conseils pertinents quant à l'organisation et l'avancement de notre travail ainsi que pour sa disponibilité pour répondre aux problèmes que nous avons pu rencontrés.

Aussi nous tenons à remercier notre responsable de formation, M. Vincent BOYER, pour son soutien et également pour son rôle important en terme de communication et de coordination.

1. Introduction

L'homme a toujours cherché à améliorer sa condition, alléger son travail, à suppléer à ses défaillances et à ses faiblesses. Depuis la Préhistoire, il a créé puis perfectionné ses outils prolongeant son bras pour la chasse, la pêche, la manipulation d'objets lourds. Dans l'antiquité grecque et byzantine, Archytas et Heron l'Ancien de l'alexandrie conçoivent déjà des automates.

En robotique, on est souvent amené à devoir planifier des trajectoires pour permettre à un robot mobile de se déplacer d'un point initial à un point final. La planification de trajectoire est un sujet qui est souvent traité dans le domaine scientifique. Il existe plusieurs algorithmes différents permettant de réaliser une telle tâche.

Durant cette première partie du projet tuteuré nous allons voir l'état de l'art de la robotique ainsi que l'application de cette science dans différents secteurs d'activités. Ensuite nous allons voir les méthodes phares permettant la génération de trajectoires en prenant en compte les obstacles présents dans l'environnement du robot.

1.1 Robotique et ses applications :

La robotique est une science qui fait intervenir un grand nombre de connaissances : la conception des systèmes, les mathématiques, la mécanique l'électronique, les asservissements, la fluidique, l'informatique, l'analyse de la gestuelle et de la scène dans un environnement. L'intelligence artificielle permet les prises de décisions et les modifications du comportement. On arrive à une image grossière de l'homme.

Il existe différents types de robotiques telles que la robotique industrielle, les humanoïdes et autre.

Dans notre recherche, nous nous sommes intéressé à la robotique mobile qui vise à concevoir des systèmes capables de se déplacer de façon autonome en suivant des trajectoires basées sur des méthodes et des algorithmes spécifiques.

Les applications directes de la robotique mobile se situent notamment dans les domaines de l'automobile, de l'exploration planétaire ou de la robotique de service par exemple.

On peut définir un robot comme une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a. Ainsi, le robot devrait être capable d'effectuer des tâches diverses de plusieurs manières et accomplir correctement sa tâche même s'il rencontre de nouvelles situations inattendues.

Cette définition s'illustre par un schéma classique des interactions d'un robot avec son environnement (Figure 1.1). Ces différentes interactions correspondent au cycle Perception /

Décision / Action. La manière dont un robot gère ces différents éléments est définie par son architecture de contrôle, qui peut éventuellement faire appel à un modèle interne de l'environnement pour lui permettre alors de planifier ses actions à long terme.

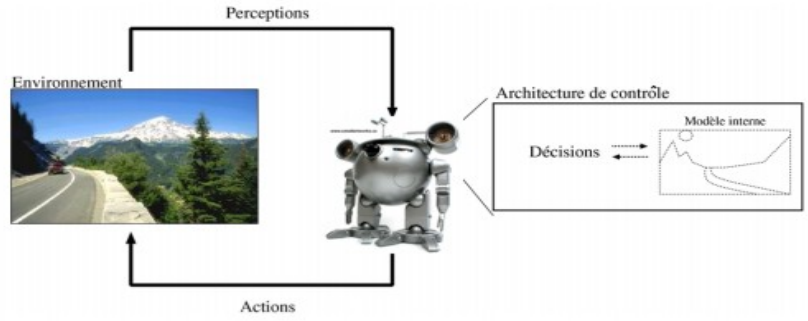


Figure 1.1. Interactions d'un robot avec son environnement[1]

1.2 Un bref aperçu historique

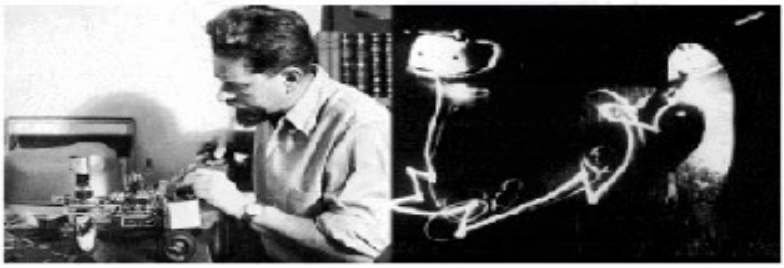


Figure 1.2. La tortue de Grey Walter : ELSIE

La Tortue construite par Grey Walter dans les années 1950[2], est l'un des tout premiers robots mobiles autonomes. Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours sujets de recherche pour les rendre de plus en plus génériques.

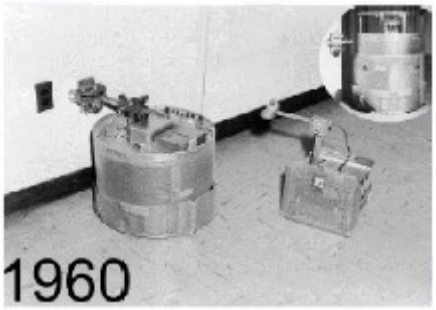


Figure 1.3. BEAST, John Hopkins University

Dans les années 60, les recherches en électronique vont conduire, avec l'apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure 1.3) de l'université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blancs) en utilisant des photo-diodes et de s'y recharger.

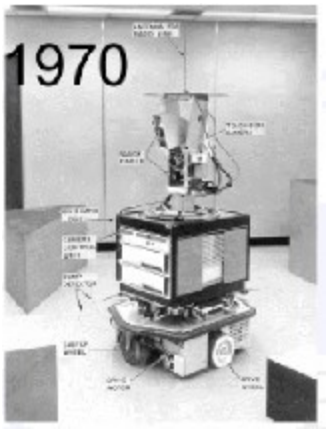


Figure 1.4. Shakey Stanford Research Institute

Les premier liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey[3]. Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle, qui à l'époque travaille essentiellement sur des approches symboliques de la planification. La perception de l'environnement, qui à l'époque est considérée comme un problème séparé, voire secondaire, se révèle particulièrement complexe et conduit là aussi à de fortes contraintes sur l'environnement.

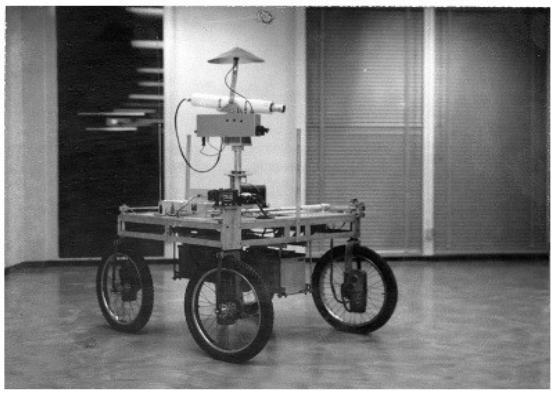


Figure 1.5. CartStanford[4]

Ces développements se poursuivent avec le StanfordCart dans les années 1980, avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement.

Figure 1.6. Genghis Robotique Réactive

Une étape importante est à signaler au début des années 1990 avec l'apparition de la robotique réactive, représentée notamment par Rodney Brooks.Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots, beaucoup plus réactifs et adaptés à leur environnement. Ces robots n'utilisent pas ou peu de modélisation du monde, problématique qui s'est avérée être extrêmement complexe. Ces développements ont continué depuis et l'arrivée sur le marché à partir des années 1990 de plates-formes intégrées a permis à de très nombreux laboratoires de travailler sur la robotique mobile et à conduit à une explosion de la diversité des thèmes de recherche. Ainsi, même si les problèmes de déplacement dans l'espace restent difficiles et cruciaux, des laboratoires ont pu par exemple travailler sur des approches multi-robot, la problématique de l'apprentissage ou sur les problèmes d'interactions entre les hommes et les robots.

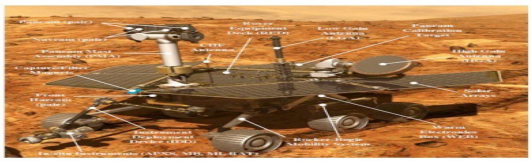
1.3 Robots mobiles autonomes

Il existe différents type de systèmes:

- **Machine télécommandée :** Seul l'opérateur commande et assure la réalisation de l'opération.

- **Machine télé opérée :** l'opérateur assure la décision en utilisant les perceptions provenant de la machine (maître/esclave).

- **Robot mobile,** autonome ou semi autonome : l'opérateur peut intervenir dans la décision



1.4 Application des robots mobiles

Sous la pression des forces économiques, il y a 3 grands domaines dans lesquels les robots sont utiles, voire indispensable :

- **La production :** Ses critères essentiels sont l'automatisation, la rapidité de reconfiguration, la flexibilité, l'apprentissage. L'environnement peut agir sur la gestuelle des robots ou être contraint pour faciliter la commande.
- **L'exploration :** Dans le sens le plus large, il s'agit de faire exécuter au robot des tâches dans les zones auxquelles l'homme ne peut pas accéder en raison du danger comme « les incendies, le nucléaire et déminage » ou de l'éloignement comme « les fonds marins, spatial ».
- **L'aide individuelle :** Le robot est un outil qui pourrait être un assistant dans les taches pénibles, répétitives ou dangereuses notamment dans les milieux hostiles, il décuple la force, augmente la précision, agit à distance comme en chirurgie par exemple(robot Da-Vinci). Des systèmes exosquelettes, prothèses ou bras sur fauteuil roulant sont des aides aux handicaps moteurs. Exemple Robot Jazz (figure 1.7).

Figure 1.7. Le robot d'assistance Jazz de Gostai

1.5 Composants d'un robot mobile

Un robot mobile est constitué de composantes matérielles et logicielles. Parmi les composantes matérielles, on retrouve une plate forme mobile à laquelle sont rattachées toutes les autres composantes comme les capteurs, les actionneurs et une source d'énergie.

1.6 L'évolution du robot

Le degré d'évolution d'un robot est directement lié à l'information introduite dans son cerveau artificiel. Cette introduction constitue la phase d'apprentissage. A partir de cela on peut distinguer 2 groupes robots:

- Ceux qui, une fois la phase d'apprentissage terminée, accomplissent les tâches sans avoir recours à des informations extérieures. Ils sont aveugles et ont un comportement en boucle ouverte par rapport à leur environnement. Tout est connu d'avance, les robots industriels apprennent une suite de gestes ou trajectoires qu'ils reproduisent toujours dans le même ordre. Les seuls capteurs d'environnement sont ceux liés à la sécurité ou à la synchronisation avec d'autres machines. Ces systèmes fonctionnent d'une manière à ce qu'ils excluent la moindre adaptation aux modifications de l'environnement. Ce sont des manipulateurs dépourvu de tout sens.
- Ceux qui, après la phase d'apprentissage tiennent compte de l'environnement et s'adaptent. Les tâches sont effectuées en mode interactif entre le robot et son environnement. Le rebot doit extraire à chaque instant les paramètres réels de la tâche, les comparer aux paramètres désirés et se piloter avec les valeurs issues de cette comparaison. Ce sont ces machines que l'on peut nommer robots. C'est le début de l'intelligence artificielle.

1.7 Les robots d'aujourd'hui

Ils courent, marchent, volent, nagent, parlent, nous imitent, et tentent de nous comprendre. Ils sont minuscules, gigantesques, anthropoïdes ou informes, et parfois mous. Les robots sont de plus en plus présents dans les sociétés et commencent à intégrer la plupart des secteurs d'activités.

1.7.1 Les robots humanoïdes

Un robot humanoïde ou androïde est un robot dont l'apparence générale rappelle celle d'un corps humain. Généralement, les robots humanoïdes ont un torse avec une tête, deux bras et deux jambes, bien que certains modèles ne représentent qu'une partie du corps, par exemple à partir de la taille. Certains robots humanoïdes peuvent avoir un « visage », avec des « yeux » et une « bouche » comme le robot NAO sur la figure 1.8.

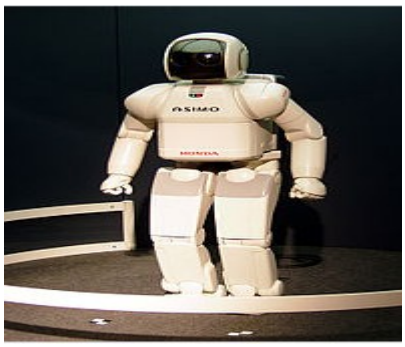


Figure 1.8. Robot humanoïde 'ASIMO'

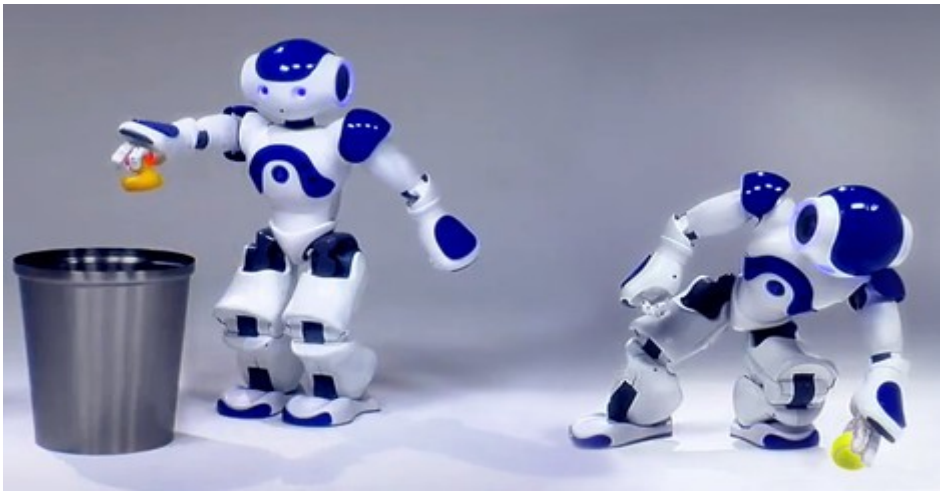


Figure 1.9. Robot humanoïde 'NAO'

2. Méthodes de génération de trajectoire

2.1 Problématique générale

La génération de trajectoire est un important problème de recherche en intelligence artificielle qui consiste à calculer une trajectoire géométrique d'un état initial à une configuration finale, pour un objet mobile donné.

Le problème de la planification de trajectoire est généralement formulé de la manière suivante : on considère un robot mobile A se déplaçant dans un espace de travail *CS*, l'objectif est de trouver les chemins qui relient la position du départ du robot P_i à sa position finale P_f.

Comment peut-on programmer des mouvements précis sur un robot ? Comment peut-on planifier un mouvement et suivre une trajectoire ? Comment un robot peut-il éviter des obstacles ?

Pour répondre à toutes ces questions on doit effectuer des recherches sur les méthodes qui permettent de planifier un mouvement ou suivre une trajectoire. Ces méthodes sont des algorithmes précis basés sur les mathématiques et l'informatique.

On peut distinguer 2 catégories différentes de ces méthodes : les méthodes probabilistes et les méthodes déterministes.

2.2 Les méthodes Probabilistes

Ce sont des méthodes qui permettent toujours de trouver un chemin s'il en existe mais ne trouve pas toujours le même chemin à chaque exécution contrairement aux méthodes déterministes.

Elles disposent pour leur recherche d'une représentation globale du monde. Ces méthodes nécessitent la construction d'un espace de recherche qui permet l'obtention implicite d'un graphe valué. A l'aide de ce graphe, une recherche heuristique permet de trouver le trajet qui minimise les fonctions de coût utilisées.

La spécificité des méthodes probabilistes peut se résumer à un parcours aléatoire de l'espace de recherche, réduisant ainsi la complexité de la résolution.

On distingue 2 méthodes principales dans les méthodes Probabilistes :

- **les méthodes de l'échantillonnage :**

Ce sont des méthodes basées sur l'approche le Probabilisticroadmap « PRM ». Elles consistent à capturer la connexité de l'espace libre par un graphe (ou un arbre) appelé réseau construit à partir de configurations choisies aléatoirement dans l'espace des configurations. Une méthode locale permet de tester l'existence d'un chemin faisable entre deux nœuds donnés du graphe et ainsi créer des arcs entre les nœuds. Elle définit un chemin d'un robot entre ces deux configurations.

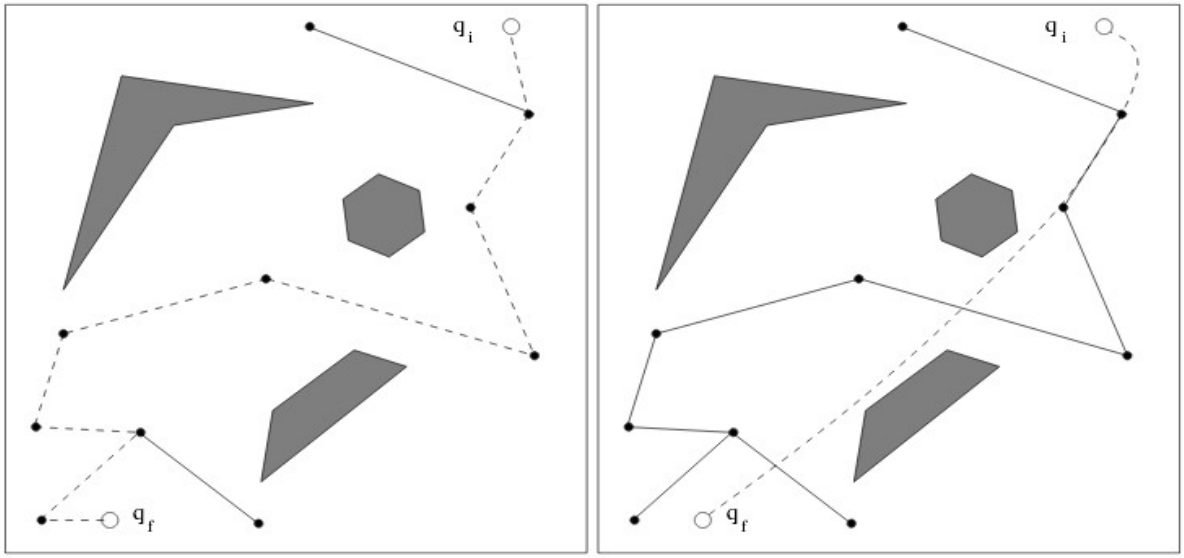


Figure 2.1. Principe de fonctionnement des méthodes de l'échantillonnage[6]

La figure de gauche représente deux configurations « q_i et q_f» qu'on veut relier par un chemin faisable s'il existe. L'algorithme réussit à connecter ces configurations au réseau et trouver ainsi un chemin faisable les reliant. Une procédure d'optimisation permet de lisser le chemin obtenu, figure de droite.

- **Les méthodes de diffusion :**

Ces méthodes qui sont devenues très populaires dernièrement, utilisent une stratégie de diffusion basée sur des algorithmes tels que « RRT » (RapidlyExploringRandomTree) qui se base sur la densité d'échantillonnage. Le nœud à étendre est choisi aléatoirement avec une probabilité inversement proportionnelle à la densité du graphe (nombre de nœuds situés dans un voisinage).

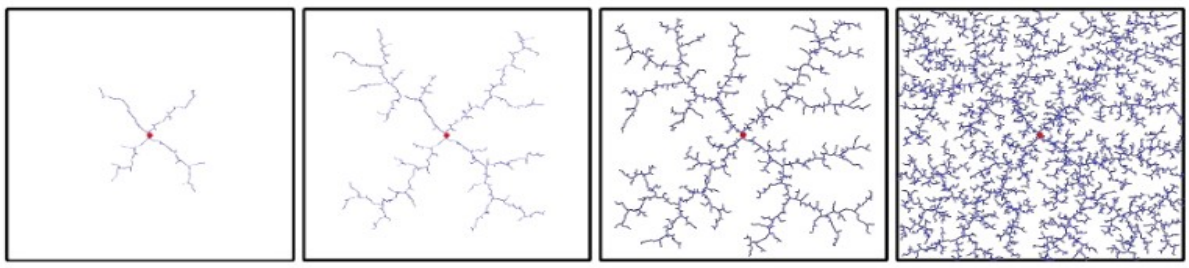


Figure 2.2. Evolution d'un arbre couvrant l'espace libre par la méthode RRT[7].

Les algorithmes RRT se composent d'une phase de construction et d'une phase de connexion. Une configuration est aléatoirement échantillonnée. La droite passant par cette configuration et le nœud de l'arbre le plus proche déterminant le nœud et la direction d'extension de l'arbre (voir Figure 2.2).

Pour comprendre le principe des méthodes de diffusion aléatoire, On suppose un arbre de racine « Qstart » dont tous les nœuds sont dans l'espace libre de l'espace configuration « *CS* » (c'est-à-dire CS privé des volumes représentant les obstacles).

- On tire tout d'abord un point « q_{rand} » de *CS* au hasard suivant une certaine loi de probabilité.

- On cherche parmi les nœuds de l'arbre le point le plus proche de q rand pour la distance dont on a munit *CS* ; on note ce point « q_{near} » .

- On calcule alors le plus court chemin entre « q_{near} » et « q_{rand} » grâce à une méthode appelée «Steering Method».

- On effectue l'opération d'extension : on parcourt ce chemin en partant de « q_{near} » et on s'arrête dès que l'on rencontre un obstacle. Ce point limite est noté« q_{new} » (cf. figure 2.3).

- On ajoute « q_{new} » à l'arbre en créant une arête entre « q_{near} » et « q_{new} »

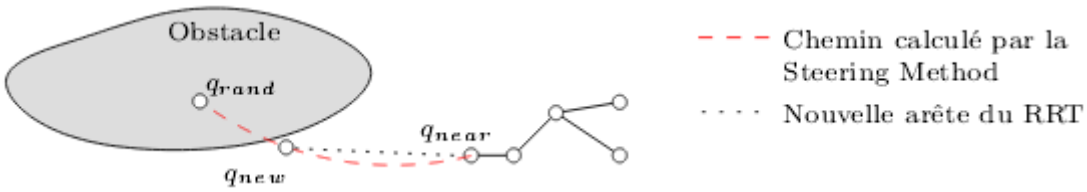


Figure 2.3. Arbre de diffusion aléatoire, opération d'extension[8]

On parcourt le chemin de « q_{near} » à « q_{rand} ». Si on rencontre un obstacle, « q_{new} » est le point limite ne créant pas de collision. Sinon, « q_{new} » est « q_{rand} ».

2.3 Les méthodes déterministes :

Les méthodes précédentes se basent toutes sur la structuration de l'espace des configurations et la modélisation a priori de sa connectivité.

D'autre méthodes existent leur principe consiste à déterminer les déplacements du robot en ne considérant qu'une représentation locale de l'environnement et à percevoir la planification de mouvement comme un problème d'optimisation.

Parmi ces méthodes on trouve :

- **champs de potentiel**

Ce sont des méthodes basées sur des stratégies réactives en suivant le principe « Perception / Action ». Le robot est soumis à un champ de forces répulsives et attractives.

Un obstacle génère un champ de potentiel répulsif tandis que l'objectif à atteindre génère un champ de potentiel attractif. L'algorithme calcul donc un vecteur résultant qui indiquera au robot comment effectuer son déplacement.

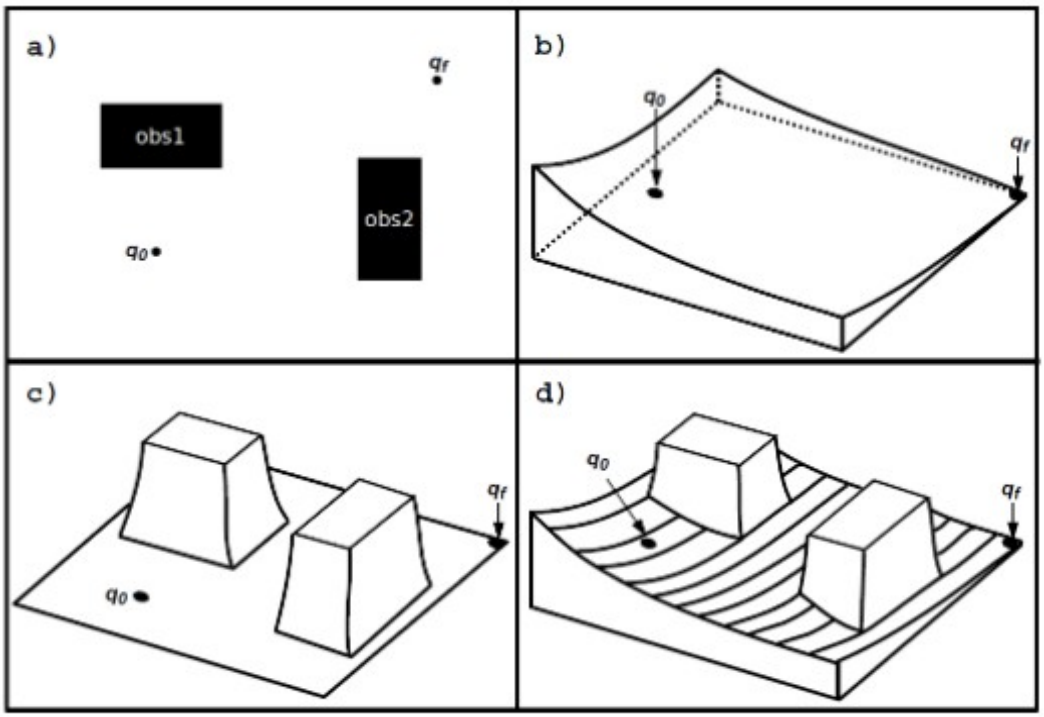


Figure 2.4. Calcul d'un chemin entre deux configurations « méthode de champs de potentiels »[9]

(a) Espace de configuration du robot et représentation des obstacles obs1 et obs2 dans celui-ci.

(b) Champ attractif généré par la position finale.

(c) Champ répulsif exercé par les obstacles.

(d) Combinaison des deux.

- **Décomposition cellulaire**

L'espace de configuration libre est décomposé en un ensemble de cellules dont la connexité est représentée par un graphe d'adjacence. Les cellules sont en contact entre elles mais ne se chevauchent pas.

Elle consiste à partitionner l'espace des configurations libres du robot en un ensemble de régions connexes adjacentes. La description de la décomposition obtenue est alors capturée dans un graphe de connectivité dont les nœuds correspondent aux différentes cellules et les arcs aux relations d'adjacence entre elles. Le problème de planifier un mouvement entre deux configurations situées initialement dans deux cellules différentes est résolu en deux étapes :

a) exploration du graphe de connectivité et détermination d'un chemin reliant les cellules contenant les deux configurations initiales du graphe.

b) recherche de la solution au problème de planification à partir de l'enveloppe définie par la liste de cellules adjacentes trouvée en « a ».

Le diagramme de **Voronoï** est l'un des exemples de la décomposition cellulaire, il s'agit d'une décomposition d'un espace ou d'un plan à des cellules à partir d'un ensemble de points appelés « germes », Il permet de construire un réseau comportant des polygones convexes. Il est utilisé dans les calculs de trajectoires des robots mobiles.Le calcul du diagramme de Voronoï est un des problèmes célèbres de la géométrie algorithmique. Son intérêt s'explique par la remarquable diversité de ses propriétés. Il permet notamment de résoudre en temps optimal d'autres problèmes comme le calcul des plus proches voisins. Pour un ensemble de points, le diagramme de Voronoï est formé de segments de droite. Pour un ensemble d'obstacles, le diagramme de Voronoï est formé de segments de droites et d'arcs de paraboles.

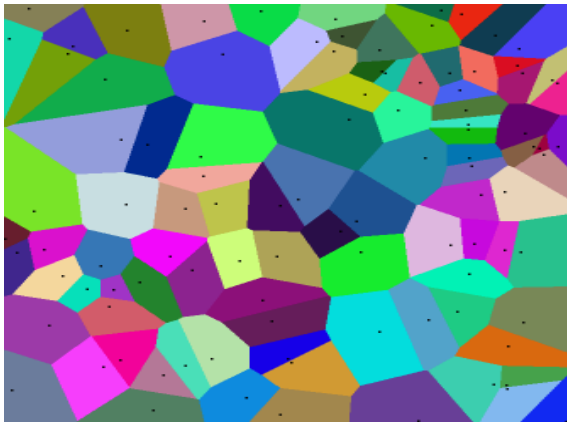


Figure 2.5. Diagramme de Voronoï [10]

La figure 2.5 représente les cellules « surfaces colorées » et chaque cellule représente la « zone d'influence » d'un germe « les points noirs ».

Enfin le diagramme de Voronoi permet de capturer la connectivité de l'espace libre et générer des solutions éloignées le plus des obstacles. Une telle méthode, bien qu'elle soit générale en théorie, reste limitée à des espaces de dimension peu élevée. Une autre variante de cette méthode est développée par Brooks pour des espaces de travail polygonaux et se base sur la représentation de ceux-ci par des cylindres généralisés.

Les chemins à suivre par le robot sont alors déterminés en considérant la connectivité entre les axes de ces cylindres.

3. Suivi de trajectoire

Le suivi de trajectoire consiste à asservir la configuration du robot sur une trajectoire de référence. L'évitement réactif d'obstacles, doit assurer que la trajectoire est sans collision et la détourner sur un intervalle quand une collision est détectée.L'évitement d'obstacles peut amener le suivi de trajectoire à ralentir, voire même à s'arrêter sur la trajectoire si la collision ne peut être évitée.

Le problème de suivi d'une trajectoire de référence pour un robot mobile est apparu comme un problème de premier ordre pour la communauté roboticienne dans ces dernières années. En effet, la forte utilisation des robots mobiles dans les domaines où l'être humain ne

peut pas être présent, notamment dans les sites nucléaires à haut risque ou dans le cas de l'exploration spatiale, nécessite la mise en œuvre de lois de commande autonomes et performantes pour assurer les tâches assignées aux robots.

3.1 Algorithme de Dijkstra

Le principe de cet algorithme est de diffuser le long des nœuds du graphe une fonction de coût qui évalue le coût du déplacement d'un nœud vers le suivant. Si cette fonction est, par exemple, la distance entre deux nœuds et que la diffusion commence à partir du nœud de départ, le chemin le plus court pour atteindre tous les nœuds du graphe peut être déterminé.

L'algorithme de Dijkstra[5] n'est valable que pour les graphes à valuation positive ou nulle, qui ne contiennent pas de circuits négatifs.A chaque itération, un sommet x reçoit son étiquette définitive, on dit qu'il est fixé. L'itération principale sélectionne le sommet x d'étiquette minimale parmi ceux déjà atteints par un chemin provisoire d'origine s. Pour tout successeur y de x, on regarde si le chemin passant par x améliore le chemin déjà trouvé de s à y : si oui on remplace V [y] par min(V [y], V [x] + W(x, y)) et on mémorise qu'on parvient en y via x en posant P[y] := x. Si tous les sommets sont accessibles au départ de s, l'algorithme se déroule en N itérations. En pratique, des sommets peuvent ne pas être accessibles. Si on veut calculer seulement un plus court chemin de s vers un 11 autre sommet t (problème B), il suffit d'arrêter l'algorithme dès que « t » est fermé, par exemple en modifiant le test de fin en "jusqu'à (V M in = +∞) ou x = t".

```
Entrée(G, list[Adj[G]], s)
Sortie(ℓ)
begin
    R ensemble de tous les noeuds;
    Initialiser le tableau V à +∞;
    Initialiser le tableau P à 0;
    V[s] := 0
    P := s while R ≠ ∅ do
        u :=Extraire-Min(V);
        P := P ∪ u;
        R := R − u;
        for All Noeud v voisin de u do
            if V[v] > V[u] + W(u, v) then
            V[v] := V[u] + W(u, v);
        end
    end
    return (ℓ)
end
```

Figure 3.1 Algorithme DIJKSTRA[11]

L'inconvénient majeur de l'algorithme de Dijkstra c'est qu'il est insensible à la densité du graphe[11]. Le nombre d'itération de la boucle While, au plus N, ne peut pas être amélioré par construction de l'algorithme. En revanche l'essentiel du travail est dû à la boucle interne trouvant le prochain sommet i à fixer.

3.2 Algorithme de Bellman

Cet algorithme à correction d'étiquettes a été conçu dans les années 50 par Bellman et Moore[11]. Il est prévu pour des valuations quelconques et peut être adapté pour détecter un circuit de coût négatif. Il s'agit d'une méthode de programmation dynamique, c'est-à-dire d'optimisation récursive, décrite par la relation suivante :

$$\begin{cases} V_0(s) = 0 \\ V_0(y) = +\infty, y \neq s \\ V_k(y) = \min_{x \in \Gamma^{-1}(y)} \{V_{k-1}(x) + W(x,y)\}, k > 0 \end{cases}$$

Vk(x) désigne la valeur des plus courts chemins d'au plus k arcs entre le sommet s et le sommet x. Les deux premières relations servent à stopper la récursion. Le sommet s peut être considéré comme un chemin de 0 arc et de coût nul. La troisième relation signifie qu'un chemin optimal de k arcs de s à y s'obtient à partir des chemins optimaux de k − 1 arcs de s vers tout prédécesseur x de y. En effet, tout chemin optimal est formé de portions optimales, sinon on pourrait améliorer le chemin tout entier en remplaçant une portion non optimale par une portion plus courte. La formulation récursive étant peu efficace, on calcule en pratique le tableau V itérativement, pour les valeurs croissantes de k. Nous donnons ci-après un algorithme simple. Les étiquettes en fin d'étape k sont calculées dans un nouveau tableau à partir du V des étiquettes disponibles en début d'étape. Pour tout sommet y, on regarde si V [y] est améliorable en venant d'un prédécesseur de y. En fin d'étape on écrase V par le nouveau tableau et on passe à l'étape suivante. En l'absence de circuit absorbant, on peut se restreindre aux chemins élémentaires pour trouver un plus court chemin de s vers tout autre sommet. Or, un tel chemin n'a pas plus de N − 1 arcs. Les étiquettes sont donc stabilisées en au plus N − 1 itérations. En pratique, elles peuvent se stabiliser plus tôt, et un meilleur test de fin est quand Vk = Vk−1. La complexité est en O(N.M) : il y a au plus N − 1 itération principale, consistant à consulter les prédécesseurs de tous les sommets, c'est-à-dire les M arcs.

```
Entrée(G, s)
Sortie(ℓ)
begin
    R ensemble de tous les noeuds;
    Initialiser le tableau V à +∞;
    V[s] := 0
    for i=1 jusqu'à Nombre de sommets de G do
        for All lien (u,v) du G do
            if V[v] > V[u] + W(u,v) then
                V[v] := V[u] + W(u,v);
            end
        end
    end
    for All lien (u,v) du G do
        if V[v] > V[u] + W(u,v) then
            return Faux
        end
    end
    return (ℓ = P)
end
```

Figure 3.2. Algorithme de Bellman [11]

4. Simulation de diagramme voronoi

Pour qu'on puisse simuler le suivi d'une trajectoire entre un point initial et un point final, on a choisi la méthode de diffusion qui se base sur l'algorithme de voronoi.

Dans un premier temps en utilisant Matlab[12] on a réussi à tracer le Diagramme de voronoi avec la fonction voronoi (x,y) comme le montre la figure 4.1.

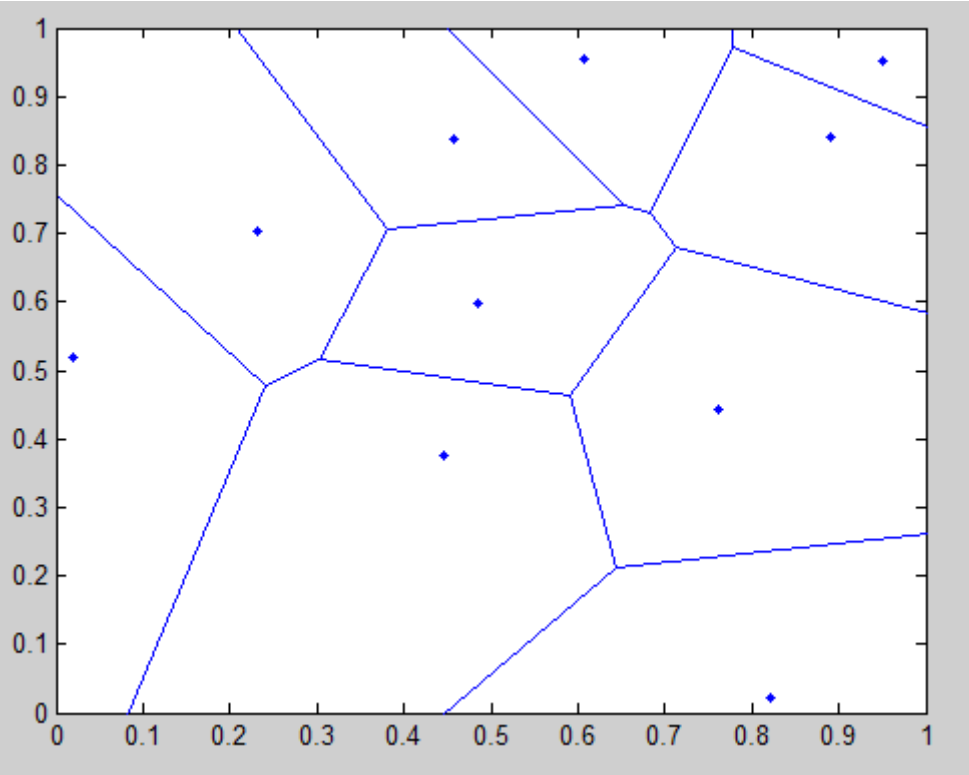


Figure 4.1: Diagramme de vornoi sous Matlab

Ce diagramme a été créé avec 10 points générés aléatoirement, chaque point est inséré dans une cellule sous forme de polygone, ce qu'on appelle le polygone de Voronoi.

Le code permettant la création du diagramme :

```
x = gallery('uniformdata',[1 10],0);
y = gallery('uniformdata',[1 10],1);
voronoi(x,y)
```

5. Conclusion et perspectives

Dans cette première partie de projet tuteuré, nous avons présenté une étude bibliographique sur la robotique en général et son application dans certains domaines. Cette recherche nous a permis de nous familiariser avec le monde de la robotique puis approfondir nos recherches afin de comprendre les méthodes permettant la génération de trajectoires.

Maintenant qu'on a compris le principe de ces méthodes nous allons mettre en pratique toutes les connaissances acquises durant cette première partie du projet, et cela en effectuant des simulations de certains algorithmes tels que le « diagramme de Voronoi » sur l'environnement Matlab Simulink.

Nous souhaitons également mettre en pratique ce que notre professeur Mr Vincent Boyer nous a appris par rapport à la gestion des développements et du versioning qui consiste à centraliser le code – dans notre cas ce serai des fonction qu'on va traduire en code sur Matlab – grâce à l'outil de gestion de versions « Git ».

6. Bibliographie

[1] Cours robotique mobile paristech

[2] (Cybernetic Tortoise) – Paul-Alain Amouriq (French)

[3] Stanford Research Institute (SRI) from 1966 to 1972

[4] cours robotique1 master 1 informatique paris 8

[5] P.Chrétienne, C.Hanen, A. e. C. (1994). Introduction à l'algorithmique. DUNOD.

[6] A Choaib Malti Planification et exécution de mouvement référencés sur des amers

[7] M Gharbi, these, Planification de mouvements et et manipulation d'objets par des torses humanoides

[8] V Delaitre, rapport, stage planification de mouvement en robotique

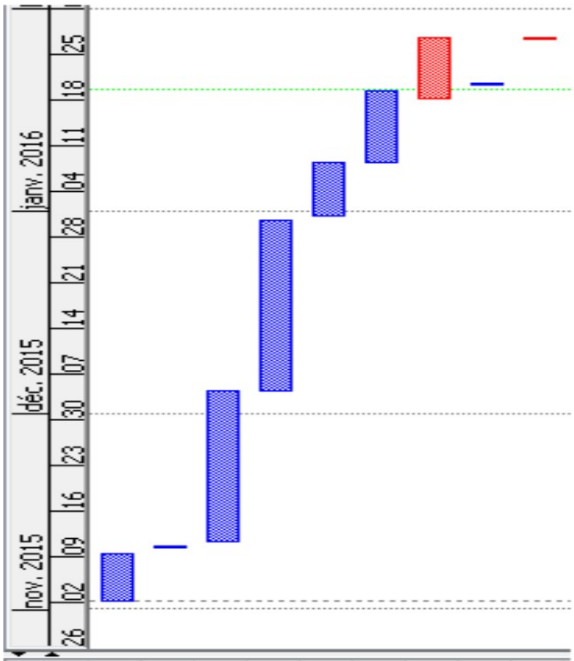
[9] V Delsart navigation autonome en environnement dynamique : une approche par déformation de trajectoire . Automatique /Robotique Univ Grenoble 2010

[10] wikipédia

[11] A kenzai DEA PLANIFICATION DE TRAJECTOIRES DE ROBOTS MOBILES VIA DES METHODES ENSEMBLISTES

[12] www.mathworks.com

7. Annexe : Diagramme de Gantt



	Nom	Durée	Début	Fin
1	Recherche & Découverte: Documentation	6 jours	02/11/15	09/11/15
2	Point de suivi: Réunion	0,125 jours	10/11/15	10/11/15
3	Etat de l'art: systèmes robotiques	18 jours	11/11/15	04/12/15
4	Méthodes de planification de trajectoires	19 jours	04/12/15	30/12/15
5	Algo de suivi de trajectoire	7 jours	31/12/15	08/01/16
6	Rédaction du rapport	8 jours	08/01/16	19/01/16
7	Application sur Matlab	8 jours	18/01/16	27/01/16
8	Retour Rapport	1 jour	20/01/16	20/01/16
9	Présentation 1ère partie	1 jour	27/01/16	27/01/16