République Algérienne Démocratique et Populaire

Ministère de l'enseignement Supérieur et de la Recherche Scientifique

Université Ammar Telidji – Laghouat

Faculté Des Sciences et Sciences de l'ingénierie

PROJET DE FIN D'ETUDES

Pour L'obtention du Diplôme

D'INGENIEURE D'ETAT EN INFORMATIQUE

Option : Systèmes Parallèles et Distribués

Thème :

Réalisation D'un Simulateur Des Algorithmes Génétiques Et De Champ Du Potentiel Artificiel Pour La Planification D'une Trajectoire du Robot KHEPERA

Réalisé par :

- **Saadaoui Aboubakr**
- **Sebkhaoui Mohamed**

Encadré par :

- **M^{elle} : TAABA Kheira**

Année Universitaire : 2008 / 2009

Dédicace

Je dédie ce modeste travail:

A la source de l'amour et de la tendresse à ma chère mère au meilleur souvenir

qui a été gravé dans mon esprit « Zohra ».

A celui qui souhaite et qui j'aie souhaité moi-meme sa presence dans ces précieux moments.

Pour qu'il assiste à ma joie et à mon bonheur de ma réussite à mes

fins d'études mon cher père « Sebkhaoui ».

A mes adorables frères

A toute ma famille …

A mes amies les plus fidèles: Fares, Dahou, Messoud, Amer, Abderrahman,

Rabouh, Said, Sliman, Belkhier …

A tous mes chères amies de « Elite School Training »

A tous mes amies de département d'informatique

Et a d'autres que je ne peux tous les citer …

Mohamed

Je dédie ce modeste fruit:

Aux êtres les plus chers a mon cœur, mes parents qui m'ont soutenu, aidé,

encouragé tout au long de ma vie scolaire

A tous mes frères …

A toute ma famille …

A tous mes amies …

Aboubakr

Remerciement

Tout d'abord des plus grands remerciements faits pour notre dieu qui

est donne la force et la santé pour accomplir notre travail.

Nous exprimons nous profonds remerciement à notre

Encadreur **M^{elle} TAABA Khiera** Pour l'aide

Compétence qu'il nous a apportée

Pour sa patience et son

Encouragement à finir ce travail.

Sommaire

Chapitre II : Les Algorithmes génétiques

Chapitre III : Présentation du simulateur

Liste des tableaux et figures

Introduction Générale

INTRODUCTION GENERALE

Les recherches récentes en robotiques s'orientent vers des architectures dotées de moyens de perception, d'action et de décision permettant une certaine autonomie. Cette autonomie est liée aux capacités de génération de trajectoire dans des milieux contraignants, aux moyens de prise de décision à partir des informations imprécises et/ou incertaines des capteurs et d'apprentissage à partir de l'interaction robot /environnement.

L'intérêt indéniable de la robotique mobile est d'avoir permis d'augmenter considérablement nos connaissances sur la localisation et la navigation de systèmes autonomes. La gamme des problèmes potentiellement soulevés par le plus simple des robots mobiles à roues en fait un sujet d'étude à part entière et forme une excellente base pour l'étude de systèmes mobiles plus complexes.

Les robots mobiles sont destinés à travailler dans des espaces relativement grands et non structurés. Certes, depuis quelques années le problème de la planification de trajectoire a reçu une attention considérable au sien de la communauté robotique, mais le problème de l'évolution des robots mobiles dans des environnements (connus/inconnus) est loin d'être complètement résolu malgré la diversité des méthodes (commandes) proposées.

Parmi ces méthodes on distingue les algorithmes évolutionnaires qui sont basé sur le principe d'optimisation et de recherche combinatoire tels que les algorithmes génétiques et les réseaux de neurones, et d'autre tel que le champ de potentiel artificiel.

Dans ce cadre, nous présentons notre travail qui a pour objectif la planification d'une trajectoire du robot mobile par deux méthodes qui sont : les algorithmes génétiques et le champ de potentiel artificiel. Ce mémoire est organisé en trois chapitres.

Nous introduisons dans le premier chapitre, des généralités sur la robotique mobile telle que les types des robots mobiles existants que ce soit dans l'industrie ou dans la recherche et les applications de ces robots et les méthodes de perception.

Dans le deuxième chapitre, nous décrivons le principe des algorithmes génétiques, nous analysons l'intérêt des différentes phases de traitement : sélection des individus, croisement et mutation. Nous décrivons enfin plusieurs méthodes de représentation des

informations traitées par un algorithme génétique sous la forme d'un codage binaire, ou par des nombres réels ou en base n.

Enfin, dans le troisième chapitre, nous représentons l'application qui a été réalisée et implémenté en MATLAB 7.1. Les résultats seront discutés et illustrés par des exemples.

Nous terminons ce travail par une conclusion générale résume notre étude et deux annexes contient la commande de champ potentiel artificiel et une présentation générale de l'interface graphique sous MATLAB (Graphic User Interface).

Chapitre 01 :

Les Concepts Fondamentaux De La Robotique Mobile

I.1. Introduction :

De manière générale, on regroupe sous l'appellation ***robots mobiles*** l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens. **[02]**

On peut estimer que les robots mobiles à roues constituent la grande partie des robots mobiles. Historiquement, leur étude est venue assez tôt, suivant celle des robots manipulateurs. Leur faible complexité en a fait de bons premiers sujets d'étude pour les roboticiens intéressés par les systèmes autonomes. Cependant, malgré leur simplicité apparente, ces systèmes ont soulevé un grand nombre de problèmes difficiles. De ce fait, les applications industrielles utilisant des robots mobiles sont rares. Cela est dû au fait que, contrairement aux robots manipulateurs qui travaillent exclusivement dans des espaces connus et de manière répétitive, les robots mobiles sont destinés à évoluer de manière autonome dans des environnements dynamiques qui peuvent ne pas être connus.

I.2. Aperçu historique :

En 1920 : Rossum's Universal Robot. Il vient du tchèque "robota" (servitude) et présent une vision des robots comme serviteurs dociles et efficaces pour réaliser les taches pénibles mais qui déjà vont se rebeller contre leurs créateurs. **[04]**

Les premiers travaux sur la robotique remontent a la fin de la deuxième guère mondiale. vers la fin des années quarante, des programmes des recherches commencèrent à Oak Ridge et laboratoire national d'argent pour développer du manipulateur mécanique téléguidés dans le but de transporter les matériaux radioactifs.

Au milieu des années cinquante, les travaux sur les manipulateur téléguidés ont aboute a des systèmes capables de faire des opérations répétitives et autonomes, c'est le cas des manipulateurs programmables commande par un grand ordinateur et qui a été crée par George C.DEVOL. **[06]**

La Tortue construite par GreyWalter dans les années 1950 (Figure1), 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. **[04]**

FIG.I.1 : La Tortue de Grey Walter

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 2) 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 blanc) en utilisant des photodiodes et de s'y recharger.

Les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Figure 2). 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. Ces développements se poursuivent avec le Stanford Cart dans les années 1980, avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement.

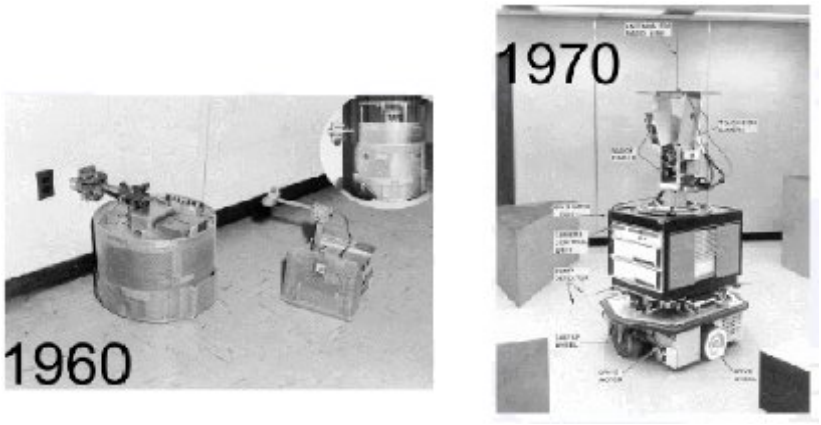

FIG.I.2 : A gauche : Robot "Beast" de l'université John Hopkins dans les années 1960. **A droite :** Le robot Shakey de Stanford en 1969 a été une plateforme de démonstration des recherches en intelligence artificielle.

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 (Figure 3), beaucoup plus réactifs et adaptés à leur environnement.

FIG.I.3 : Genghis, développé par Rodney Brooks au MIT au début des années 1990.

I.3. Définitions :

En général, on peut définir un robot mobile comme étant 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. **[08]**

En particulier, un robot mobile autonome est un système mécanique, électronique et informatique complexe mettant en œuvre :

- Un ensemble de capteurs (extéroceptifs et/ou proprioceptifs) :

Les capteurs extéroceptifs ont pour objectif d'acquérir des informations sur l'environnement proche du véhicule. **Les capteurs proprioceptifs** fournissent des données sur l'état interne du robot (telles que sa vitesse ou sa position).

- Un ensemble d'effecteurs :

L'objectif du robot est d'atteindre un objectif dans son environnement en évitant les obstacles. Le problème que l'on doit résoudre est de déterminer en fonction des données capteurs quelles commandes doivent être envoyées à chaque instant au robot pour atteindre cet objectif. Ces effecteurs ont comme but de permettre au robot d'évoluer dans un monde prévu à l'origine pour l'homme. Les plus courants sont les systèmes à roues, mais il existe aussi des robots à chenilles, à pattes ou se déplaçant par reptation.

Le type de locomotion définit deux types de contraintes :

- les contraintes cinématiques, qui portent sur la géométrie des déplacements possibles du robot.

- Les contraintes dynamiques, liées aux effets du mouvement (accélérations bornées, vitesses bornées, présence de forces d'inertie ou de friction)

Selon sa cinématique, un robot est dit :

- holonome, s'il peut se déplacer instantanément dans toutes les directions.

- non-holonome, si ses déplacements autorisés sont des courbes dont la courbure est bornée. **[01]**

I.4. Les applications des robots mobiles :

Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses", mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées. Parmi les domaines concernés, citons :

- ✓ La robotique de service (hôpital, bureaux)
- ✓ La robotique de loisir (robot 'compagnon')
- ✓ La robotique industrielle ou agricole (entrepôts, récolte de productions agricoles, mines)

- ✓ La robotique en environnement dangereux (spatial, industriel, militaire)

A cela, s'ajoute à l'heure actuelle des nombreuses plates-formes conçues essentiellement pour les laboratoires de recherche. La figure 4 montre quelques exemples de robot réels.

FIG.I.4 : Exemples de robots utilisés dans différentes applications.

I.5. Les avantages des robots mobiles :

Les avantages des robots mobiles sont résumés dans les points suivants :

- ✓ Accroissement de la capacité de production.
- ✓ La qualité de produit et le soin apporté à sa fabrication
- ✓ Le remplacement de l'homme dans les taches pénibles ou dangereuses
- ✓ Manutentions.

I.6. Présentation de robot KHEPERA :

Le robot Khepera est un mini-robot conçu comme outil de recherches et d'enseignement dans le cadre d'un programme suisse de recherches. Il a été développé la première fois en 1992, par une équipe de recherche du laboratoire de microprocesseur et d'interface (LAMI) à l'institut de la technologie fédéral suisse Lausanne (EPFL). Il permet la confrontation au vrai monde des algorithmes développés dans la simulation pour l'exécution de trajectoire, l'action d'éviter d'obstacle, et le prétraitement d'information sensorielle. Le robot Khepera est maintenant largement répandu autour du monde comme plateforme pour différentes expériences et applications de la robotique.

Le Khepera (diamètre 55 mm, hauteur 30 mm, poids 80 g dans sa configuration de base) est montré sur la **Figure 5**, il dispose d'un processeur embarqué performant (M68331, 32 bits, cadencé à 16 MHz), associé à une EEPROM de 128 K octets et une

RAM statique de 256 K octets. Le BIOS (Basic I/Os System) est le système bas niveau embarqué dans le robot. Il offre des capacités multitâches et permet la gestion de plusieurs modules logiciels : acquisition et conversion des données sensorielles, asservissements des moteurs, contrôle de la communication entre différents modules du robot et l'extérieur.

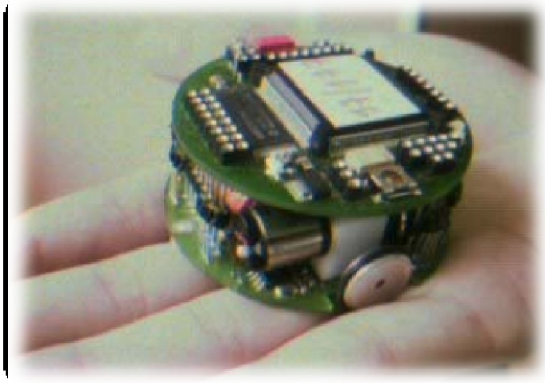

FIG.I. 5 : Le robot Khepera

Les robots Khepera se composent tous de deux modules : le module de bas comportant la partie motrice et la partie d'alimentation d'une part, le module de contrôle possédant le microcontrôleur et la mémoire d'autre part. **[05]**

I.7. modélisation :

I.7.1.Description :

Le robot Khepera est un robot de type unicycle, ce type des robots est très répandu en raison de sa simplicité de construction et de propriétés cinématiques intéressantes.

Ce robot est actionné par deux roues indépendantes et possédant éventuellement une roue folle assure sa stabilité.

I.7.2. Définitions :

On note $= (, \vec{}, \vec{}, \vec{})$ un repère fixe quelconque, dont l'axe $\vec{}$ est vertical, et $= (, \vec{} , \vec{} , \vec{})$ Un repère mobile lié au robot. On choisit généralement pour un point remarquable de la plate-forme, typiquement le centre de l'axe des roues motrices, comme illustré à la figure 6. Qui on y a omis la roue folle, qui n'intervient pas dans la cinématique, dans la mesure où elle a été judicieusement placée.

Par analogie avec la manipulation, on appelle situation [Fourquet 99] ou souvent posture [Campion 96] du robot le vecteur :

$\xi =$,

Où et sont respectivement l'abscisse et l'ordonnée du point dans R et Θ l'angle (⃗, ⃗). La situation du robot est donc définie sur un espace M de dimension m = 3, comparable à l'espace opérationnel d'un manipulateur plan. **[07]**

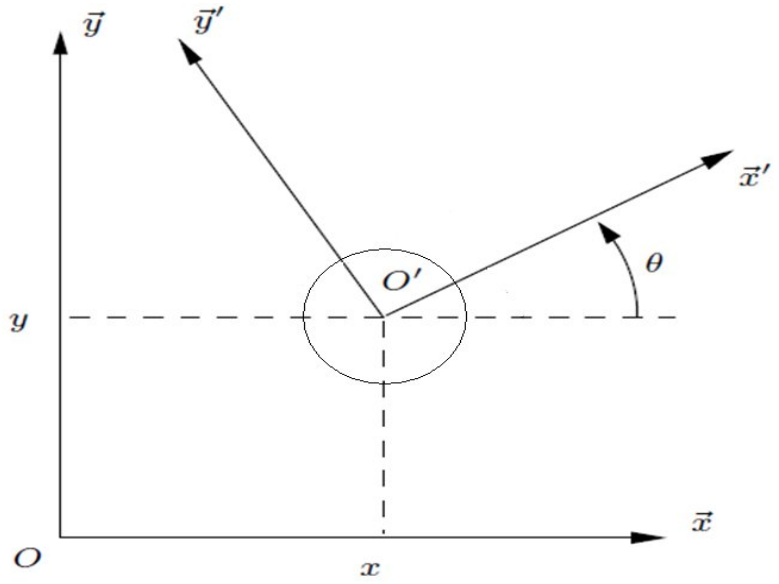

FIG.I.6 – Repérage d'un robot mobile

I.7.3. Roulement sans glissement :

La locomotion à l'aide de roues exploite la friction au contact entre roue et sol. Pour cela, la nature du contact (régularité, matériaux en contact) a une forte influence sur les propriétés du mouvement relatif de la roue par rapport au sol. Dans de bonnes conditions, il y a roulement sans glissement (r.s.g.) de la roue sur le sol, c'est-à-dire que la vitesse relative de la roue par rapport au sol au point de contact est nulle. Théoriquement, pour vérifier cette condition, il faut réunir les hypothèses suivantes :

– le contact entre la roue et le sol est ponctuel ;

– les roues sont indéformables, de rayon *r*.

En pratique le contact se fait sur une surface, ce qui engendre bien évidemment de légers glissements. De même, alors qu'il est raisonnable de dire que des roues pleines sont indéformables, cette hypothèse est largement fausse avec des roues équipées de pneus.

Malgré cela, on supposera toujours qu'il y a r.s.g. et, par ailleurs, que le sol est parfaitement plan.

Mathématiquement, on peut traduire la condition de r.s.g. sur une roue. Soit P le centre de la roue, Q le point de contact de la roue avec le sol, ω l'angle de rotation propre de la roue et θ l'angle entre le plan de la roue et le plan (, ⃗,) ⃗comme indiqué à la figure7. La nullité de la vitesse relative ⃗ roue/sol au point de contact permet d'obtenir une relation vectorielle entre la vitesse ⃗ du centre P de la roue et le vecteur vitesse de rotation ⃗ de la roue :

$$\vec{} = \vec{} + \vec{} \wedge \vec{} = \vec{0}$$

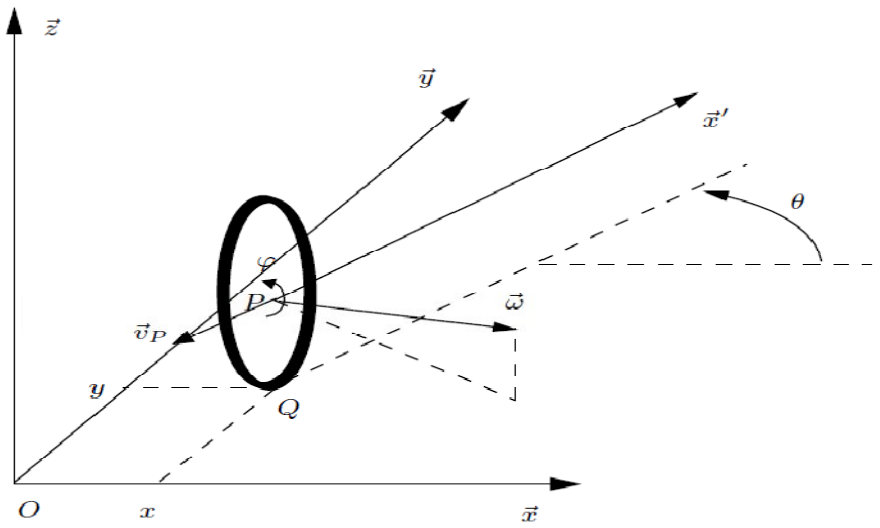

FIG.I. 7 – Caractérisation du roulement sans glissement

Les points P et Q ont pour coordonnées respectives (, ,)et (, ,0). Il vient alors :

$$\cdot \vec{} + \cdot \vec{} + \vec{} \cdot (\quad \theta \vec{} - \quad (\theta \vec{})) \vec{} = \vec{0}$$

$$(\cdot + \quad \cdot) \vec{} \theta (\cdot + \quad \cdot) \vec{} \theta = \vec{0}$$

Ceci nous donne le système de contraintes scalaires :

$$\cdot + \quad \cdot \quad \theta = 0 \qquad (1.1)$$

$$\cdot + \quad \cdot \quad \theta = 0 \qquad (1.2)$$

Que l'on peut transformer pour faire apparaître les composantes de vitesse dans le plan de la roue d'une part et perpendiculairement `a la roue d'autre part :

$$- \cdot \quad \theta + \cdot \quad \theta = 0 \qquad (1.3)$$

$$- \cdot \quad \theta + \cdot \quad \theta = - \quad \cdot \qquad (1.4)$$

Ces contraintes traduisent le fait que le vecteur ⃗ soit dans le plan de la roue et ait pour module ˙. **[02]**

I.7.4.Centre instantané de rotation :

Les roues motrices ayant même axe de rotation, le CIR du robot est un point de cet axe. Soit ρ le rayon de courbure de la trajectoire du robot, c'est-à-dire la distance du CIR au point (voir figure 7). Soit L l'entre-axe et ω la vitesse de rotation du robot autour du CIR. Alors les vitesses des roues droite et gauche, respectivement notées et et définies à la figure 8, vérifient :

$$= - \quad \dot{} = (\rho + L)\omega \qquad (1.5)$$

$$= \quad \dot{} = (\rho - L)\omega \qquad (1.6)$$

Ce qui permet de déterminer ρ et ω à partir des vitesses des roues :

$$\rho = L \frac{\dot{} \quad \dot{}}{\dot{}} \qquad (1.7)$$

$$\omega = - \frac{(\quad \dot{})}{} \qquad (1.8)$$

L'équation (1.7) permet de situer le CIR sur l'axe des roues. Par ailleurs ces équations expliquent deux propriétés particulières du mouvement des robots de type unicycle : si $\dot{} = - \dot{}$, le robot se déplace en ligne droite ; si $\dot{} = \dot{}$, alors le robot effectue un rotation sur lui-même. L'utilisation de ces deux seuls modes de locomotion, bien que limitée, permet de découpler les mouvements et de fournir une solution simple pour amener le robot d'une posture à une autre. C'est sans doute là une des raisons du succès de ce type de robots. Pour élaborer une stratégie plus fine de déplacement, il est cependant intéressant de savoir comment la posture du robot est reliée à la commande de ses roues. **[07]**

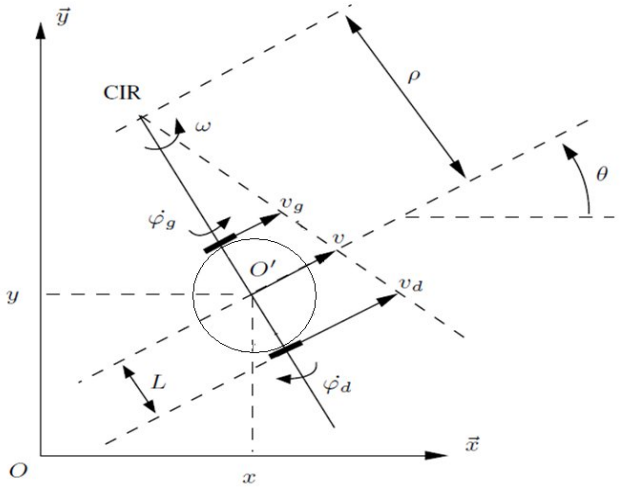

FIG.I.8 – Centre instantané de rotation d'un robot de type unicycle

I.7.5.Choix de la commande :

En ce qui concerne la commande, si l'on se contente de traiter le cas cinématique, on peut considérer que celle-ci est donnée, au plus bas niveau, par les vitesses de rotation des roues. Ceci étant, on préfère généralement exprimer cette commande par la vitesse longitudinale du robot, notée v (en) et sa vitesse de rotation θ̇ (autour de).

Il y a en effet équivalence entre les deux représentations. D'une part, on a :

$$= \frac{}{} = \frac{(\quad \cdot \quad)}{}$$

D'autre part, la vitesse de rotation du robot est égale à la vitesse de rotation autour du CIR [Dudek00]:

$$= \dot{\theta} = - \frac{(\quad \cdot \quad)}{} \qquad (1.9)$$

Conformément à l'équation (1.8). On montre que ces relations sont parfaitement inversibles et qu'il y a ainsi équivalence entre les couples (;) et (v; θ). Désormais, on utilise plutôt ce dernier couple de grandeurs, plus parlantes, quitte à calculer ensuite les angles ou vitesses de consigne des asservissements des roues. **[07]**

I.7.6.Modèle cinématique en posture :

Relier la dérivée de la posture à la commande u=(v ω) est facile. Une simple considération géométrique donne :

$$\dot{} = \cos\theta$$
$$\dot{} = \sin\theta$$
$$\theta =$$

I.8. Système de perception pour robot mobile :

I.8.1. Définition :

La détection d'obstacle est un des problèmes clef de la robotique, tout robot mobile devant évoluer dans un environnement dynamique doit être capable de détecter et éviter les obstacles de manière autonome, c'est un exemple simple et intéressant vu que le sujet sur le quel nous travaillons touche principalement le domaine de la robotique et plus précisément l'évitement d'obstacle ce dernier nécessite une bonne perception, ce qui nous a pousse a détaillé cette notion (perception) dans ce qui suit.

I.8.2. Perception :

La notion de perception en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot mobile pour agir et réagir dans le monde qui l'entoure. Alors que pour des taches de manipulation on peut considérer que l'environnement du robot est relativement structure, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement de sa tache, il est nécessaire que le robot dispose de nombreux captures mesurant aussi bien son état interne que l'environnement dans lequel il évolue. Le choix des captures dépend bien évidemment de l'application envisagée. **[01]**

I.8.3. Les capteurs en robotique mobile :

En robotique mobile, on classe traditionnellement les capteurs en deux catégories selon qu'ils mesurant l'état du robot lui-même ou l'état de son environnement. Dans le premier cas, a l'image de la perception chez les êtres vivants, on parle de proprioception et donc de capteurs proprioceptifs. On trouve par exemple dans cette catégorie les capteurs de position ou de vitesse des roues et les capteurs de charge de la batterie. Les capteurs renseignant sur l'état de l'environnement, donc de ce qui est extérieur eu robot lui-même, sont eux appelles capteurs extéroceptifs. Il s'agit de capteurs donnant la distance du robot à l'environnement, la température, signalant la mise en contact du robot avec l'environnement, etc.

L'étude détaillée des capteurs, qui relève à la fois de la physique, de l'électronique et du traitement du signale, ne sera pas vue ici. Nous nous contenterons d'expliquer les principes de fonctionnement des capteurs présents. On tachera simplement de garder à l'esprit que les défauts inhérents aux différents systèmes de mesure utilises (bruit, erreurs ou échecs de mesures, difficulté de modélisation) influent fortement sur la perception que le robot a de l'environnement.

Nous présentons dans section les capteurs les couramment utilisés en robotique mobile pour les besoins de la navigation.

a. Les capteurs proprioceptifs :

Les capteurs proprioceptifs permettent une mesure de certains paramètres internes au système lui-même. Dans le cas de robotique mobile, ces capteurs sont souvent utilises pour estimer la position courante du robot lors de son déplacement; c'est ce qu'est connu dans la littérature par odométrie.

- **L'odométrie :**

  Elle permet d'estimer le déplacement à partir de la mesure de rotation des roues (ou du déplacement de pattes)/la mesure de rotation est en général effectuée par un codeur optique disposé sur l'axe de la roue, ou sur le système de transmission (par exemple sur la sortie de la boite de vitesse). Le problème majeur de cette mesure est que l'estimation du déplacement fournie dépend très fortement de la qualité du contact entre la roue et le sol. Elle peut être relativement correcte pour une plate-forme à deux roues motrices sur un sol plan de qualité uniforme, mais est en générale quasiment inutilisable seule pour un robot à chenille

- **Le système radar doppler**

  Au lieu de mesurer le déplacement par des mesures sur les roues, il est possible d'utiliser un petit radar pointé vers le sol qui permet de mesurer la vitesse du véhicule par effet doppler. Ce système présente l'avantage d'être beaucoup plus précis que la mesure passant par les roues, et d'être indépendant des dérapages possibles de ces roues, mais est en général plus cher et encombrant. Il est de plus très rare sur les petites plates-formes car il ne peut mesurer de faibles vitesses de déplacement.

- **Les systèmes inertiels**

  La mesure de déplacement potentiellement la plus fiable provient de la mesure des accélérations de la plate-forme par des capteurs inertiels. Cette mesure est potentiellement fiable car elle ne dépend pas de la nature locale de l'environnement.

b. Les capteurs extéroceptifs

Quelle que soit technologie utilisée pour effectuer la mesure télémétrique, le capteur retourne généralement deux informations. La première donne l'angle de gisement, c'est –à-dire la direction dans laquelle a été faite la mesure. La seconde donne

la distance au corps ayant réfléchi l'onde émise. Cette technique de mesure permet donc de positionner les objets présents dans la scène par rapport au robot. Elle se prêté très bien aux environnements d'intérieur; structures, comportant des formes régulières et statiques comme des murs, qui par ailleurs, possèdent généralement de bonnes propriétés de réflexion.

Il existe différents types de télémètre, qui permettent de mesurer la distance à l'environnement, utilisant divers principes physiques.

- **Télémètre à ultrason**

Les télémètres à ultrason sont historiquement les premiers à avoir été utilises .il utilisent la mesure du temps de retour d'une onde sonore réfléchie par les obstacles pour estimer la distance. Ces télémètres sont très simple et peut cher, et sont donc très répandus, mais possèdent de nombreux inconvénients

- **Télémètres à infrarouge :**

Ces télémètres possèdent l'avantage d'avoir un cône de détection beaucoup plus restreint. Ils utilisent un limier infrarouge au lieu d'une onde sonore pour la détection et peuvent être bases sur différents techniques qui permettent de recueillir plus ou moins d'information. Il est possible de mesurer simplement le retour ou non-retour d'une impulsion codée, ce qui permet de détecter la présence ou l'absence d'un obstacle dans une certaine portion de l'espace.

Il est également possible de réaliser une triangulaire sur le faisceau de retour de l'onde lumineuse, ce qui permet d'avoir une mesure de la distance de l'obstacle.

- **Télémètre laser**

Les télémètres les plus utilisés à l'heure actuelle pour des applications de cartographie et de localisation sont les télémètres laser à balayage. Ils utilisent un faisceau laser mis en rotation afin de balayer un plan, en général horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan (figure)

Cette mesure peut être réalisée selon différentes techniques (mesure du temps de retour, interférométrie…) **[01]**

I.9.Conclusion :

Nous avons présenté dans ce chapitre une vision globale sur la robotique mobile et le robot KHEPERA, on a commencé par un historique sur les robots mobiles, et puis faire une définition sur les ensemble de capteurs et d'effecteurs, et on a citée aussi quelques domaine d'applications et quelques avantages, puis on a présenté le robot KHEPERA et la modélisation cinématique de ce robot, et en fin on a citée les différents systèmes de perception pour les robots mobiles.

Après l'étude et la conception de la robotique mobile, il faut présenter la méthode d'optimisation que nous avons choisi (les algorithmes génétiques), le chapitre suivant détaille le principe et les différents techniques de cette méthode.

Chapitre 02 :

Les Algorithmes Génétiques

II.1. Introduction :

Malgré l'évolution permanente des calculateurs et les progrès fulgurants de l'informatique, il existe pour plusieurs problèmes d'optimisation une taille critique de l'espace de solutions admissibles. La méthode permettant d'obtenir une solution optimale est bien évidemment celle de l'énumération complète de l'espace de recherche. Cette dernière est dans la plupart des cas prohibitive. Compte tenu de ces difficultés, certains chercheurs, il y a environ une trentaine d'années, se sont interrogés pour savoir comment faire mieux : il est apparu plusieurs similarités entre le monde biologique et le monde informatique. De ce fait, l'approche évolutive fût utilisée. En particulier, les algorithmes génétiques vu qu'ils présentent des qualités intéressantes pour la résolution de divers problèmes. Ils sont basés sur la théorie de l'évolution des espèces dans leur milieu naturel, soit une transposition artificielle des concepts basiques de la génétique et des lois de survie énoncées par Charles Darwin : les individus les plus adaptés survivent et se reproduisent. Selon Darwin, les mécanismes à l'origine de l'évolution naturelle des êtres vivants reposent sur la compétition qui sélectionne les individus les plus adaptés à l'environnement actuel au détriment des autres. L'hypothèse de la théorie de Darwin, compte tenu des connaissances actuelles de la génétique, montre que ces mécanismes ne sont pas toujours justifiés. Ces mêmes mécanismes seront utilisés dans l'implémentation de l'algorithme génétique.

L'application des algorithmes génétiques aux problèmes d'optimisation a été formalisée par Goldberg en 1989. Ensuite ils se sont vite imposés (comme méthodes d'optimisation globale), en permettant l'optimisation des problèmes très variés. Particulièrement, ces méthodes permettent de traiter des problèmes dont la taille est considérable, ou encore des problèmes non décrits de manière explicite. Leur vaste champ d'action, leur implantation généralement aisée sont certainement à l'origine de leur succès. **[09]**

II.2. Description :

John Holland, ses collègues et ses étudiants ont développé à l'université de Michigan les Algorithmes Génétiques (AGs), métaphores biologiques inspirées des mécanismes de l'évolution darwinienne (sélection naturelle) et de la génétique. Ces

métaphores prennent la forme d'algorithmes de recherche appelés "algorithmes génétiques".

Ces algorithmes font partie de la classe des algorithmes dits stochastiques. En effet une grande partie de leur fonctionnement est basée sur le hasard. Bien qu'utilisant le hasard, les AGs ne sont pas purement aléatoires. Ils exploitent efficacement l'information obtenue précédemment pour spéculer sur la position de nouveaux points à explorer, avec l'espoir d'améliorer la performance.

Les algorithmes génétiques permettent à une population de solutions de converger vers les solutions optimales. Pour ce faire, ils vont utiliser un mécanisme de sélection des individus de la population (les solutions potentielles). Les individus sélectionnés vont être croisés entre eux (exploitation), et certains vont être mutés (exploration). Ces mécanismes d'exploitation et d'exploration vont permettre de converger vers les bonnes solutions en évitant, autant que faire se peut, les optima locaux.

II.3. Terminologie et éléments de base

Un algorithme génétique recherche les extrêmes d'une fonction définie sur un espace de données appelé *population*. Par analogie avec la génétique, chaque *individu* de cette population est un chromosome et chaque *caractéristique* de l'individu est un gène. Dans un cas simple, un gène sera représenté par un bit (0 ou 1), un chromosome par une chaîne de bits. Chaque gène représente une partie élémentaire du problème, il peut être assimilé à une variable et peut prendre des valeurs différentes appelées *allèles***,** La position du gène dans le chromosome se nomme *locus.*

On parle également de *génotype* et de *phénotype*. Le génotype représente l'ensemble des valeurs des gènes du chromosome alors que le phénotype représente la solution réelle après transformation du chromosome. Lors de la génération d'une nouvelle population, des opérateurs génétiques tels que la sélection, le croisement et la mutation sont nécessaires pour la manipulation des chromosomes.

Le tableau 1 présente une récapitulation de la terminologie naturelle et celle utilisée par les algorithmes génétiques.

Nature	Algorithme génétique
Chromosome	Chaîne
Gène	Trait, caractéristique
Allèle	Valeur de la caractéristique
Locus	Position dans la chaîne
Génotype	Structure Ensemble des valeurs des gènes
Phénotype	Ensemble de paramètres, structure décodée Evaluation d'un génotype

TAB.II.1.Comparaison de la terminologie naturelle et celle des algorithmes génétiques

Les AGs utilisent donc un vocabulaire similaire à celui de la génétique. On parlera ainsi d'individus ou chromosomes dans une population. Chaque individu ou chromosome est constitué d'un ensemble d'éléments appelés gènes contenant les caractères héréditaires de l'individu. Ils utilisent un mécanisme de sélection naturelle, basée essentiellement sur la reproduction et sur le codage génétique qui stocke les informations décrivant l'individu sous forme de gènes imitant les systèmes naturels de l'évolution des espèces.

II.4. Évolution des espèces :

Dans un environnement quelconque dans lequel vit une population primitive, i.e. peu adaptée à cet environnement. Bien sûr, quoique globalement inadaptée, cette population n'est pas uniforme : certains individus sont mieux armés que d'autres pour profiter des ressources offertes par l'environnement (nourritures, abris, etc.) et pour faire face aux dangers qui y rôdent (prédateurs, intempéries, etc.). Ces individus mieux équipés ont par conséquent une probabilité de survie plus grande que leurs congénères et auront de fait d'autant plus de chances de pouvoir se reproduire. En se reproduisant entre individus bien adaptés, ils vont transmettre à leurs enfants ces caractéristiques qui faisaient leur excellence. La population qui résultera de cette reproduction sera donc globalement mieux adaptée à l'environnement que la précédente puisque la plupart des

individus auront hérité de plusieurs (puisque chacun hérite à la fois de sa mère et de son père) des caractéristiques de l' "élite" de la génération précédente. Et c'est ainsi, en recombinant à chaque génération les caractéristiques élémentaires de bonne adaptation et en saupoudrant-le tout d'un peu de hasard, que la population va évoluer vers une adéquation toujours meilleure avec l'environnement. Par analogie, les AGs joignent le même principe, ils sont basés sur le principe d'" évolution" d'une population d'individus. Dans celle-ci, ce sont en général les plus forts, c'est-à-dire les mieux adaptés au milieu, qui survivent et engendrent des progénitures. À partir des données du problème, on crée (généralement aléatoirement) une " population" de solutions admissibles. Puis on évalue chacune des solutions. On élimine une partie infime de celles qui se sont montrées inutiles, et on recombine les gènes des autres afin d'obtenir de nouveaux individus-solutions. Ainsi, à chaque génération un nouvel ensemble de créatures artificielles (des chaînes de caractères) est crée en utilisant des parties des meilleurs individus de la génération précédente ainsi que des parties innovatrices. Selon la théorie évolutionniste, cette nouvelle génération sera globalement plus adaptée au problème que la précédente. Ce procédé est alors répété jusqu'à la naissance d'une solution que l'on jugera satisfaisante.

II.5. A quoi sert l'algorithme génétique ?

L'algorithme génétique *résout des problèmes* n'ayant pas de méthode de résolution décrite précisément ou dont la solution exacte, si elle est connue, est trop compliquée pour être calculée en un temps raisonnable. Ceci dit, face à un problème pour lequel il existe pour ainsi dire une infinité de solutions, plutôt que d'essayer naïvement toutes les solutions une à une pour trouver la meilleure, on va explorer l'espace des solutions en se laissant guider par les principes des algorithmes génétiques.

II.6. Conception d'un algorithme génétique

La simplicité de mise en œuvre et l'efficacité constituent deux des caractéristiques les plus attrayantes de l'approche proposée par les AGs. La mise en œuvre d'un algorithme génétique sollicite la disponibilité :

- d'une *représentation génétique* du problème, c'est-à-dire un codage approprié des solutions sous la forme de chromosomes. Cette étape associe à chacun des points de l'espace de recherche une structure de données. Elle se place généralement après une

phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques ;

- d'un *mécanisme de génération* de la population initiale. Ce mécanisme doit être capable de produire une population non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut prendre plus ou moins rapidement la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche ;
- d'une *fonction d'évaluation* pour mesurer la force de chaque chromosome ;
- d'un *mode de sélection* des chromosomes à reproduire ;
- des *opérateurs* permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace de recherche ;
- des *valeurs* pour les *paramètres* qu'utilise l'algorithme : taille de la population, nombre total de générations ou critère d'arrêt, probabilités de croisement et de mutation.

II.7. Comment fonctionne l'algorithme génétique ?

Un algorithme génétique fonctionne typiquement à travers un cycle simple de quatre étapes :

1. création d'une population de chromosomes ;
2. évaluation de chaque chromosome ;
3. sélection des meilleurs chromosomes ;
4. manipulation génétique, pour créer une nouvelle population de chromosomes.

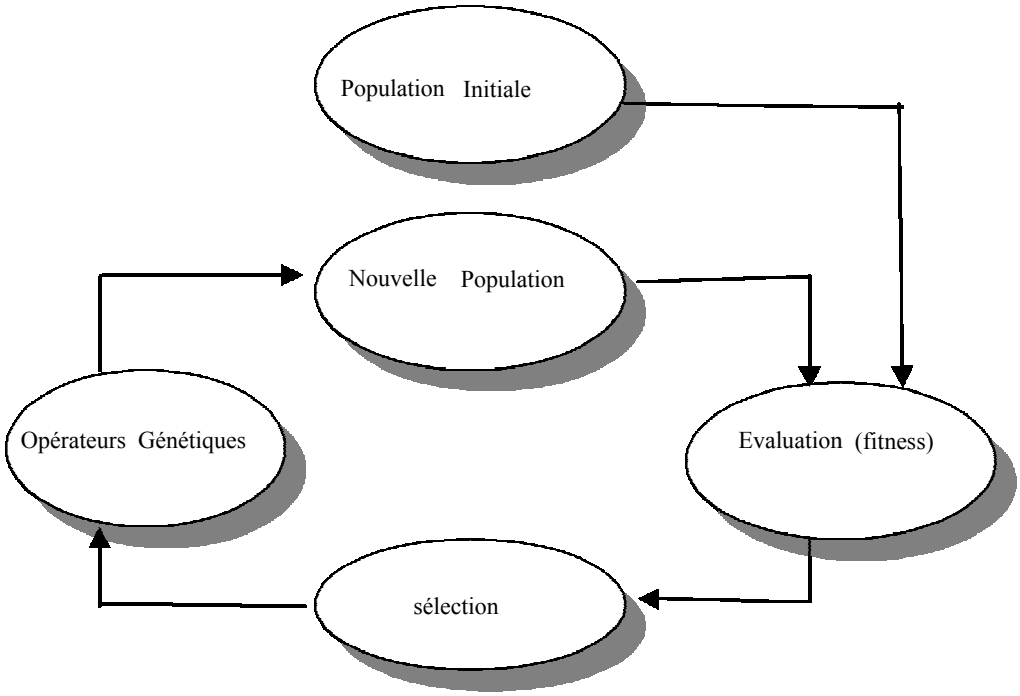

FIG.II.1- Cycle génétique

Le cycle décrit par la Figure.1 est inspiré par la terminologie génétique. Lors de chaque cycle, une nouvelle génération de solutions du problème est obtenue. Initialement, une population initiale est générée où chaque individu-solution de la population est codé sous forme d'une chaîne de caractères (chromosomes). Ensuite, une évaluation de chaque chromosome sera établie. Cette évaluation consiste à évaluer la qualité des chromosomes à l'aide de la fonction d'évaluation : fitness. Ce qui permet de sélectionner les chromosomes les plus adaptés et par conséquent leur appliquer les opérateurs génétiques (croisement et mutation) ce qui crée une nouvelle génération.

A la fin du cycle, une nouvelle population est acquise ouvrant ainsi la voie pour une nouvelle génération et par conséquent un nouveau cycle.

II.8. Variantes :

En fait, les algorithmes génétiques sont une famille d'algorithmes, basés autour des mêmes idées. Cependant il existe beaucoup de variantes possibles suivant la représentation choisie, les opérateurs de croisement, de mutation et de sélection. La section suivante présente les choix les plus courants qui définissent les variantes.

II.8.1. Codage :

Le codage est une modélisation d'une solution d'un problème donné sous forme d'une séquence de caractères appelée chromosome où chaque caractère, dit aussi gène, représente une variable ou une partie du problème. La tâche principale consiste à choisir le contenu des gènes qui facilite la description du problème et respecte ses contraintes. La littérature définit deux types de codage : binaire et réel. **[09]**

II.8.1.a. Codage binaire

Le codage classique utilise l'alphabet binaire : 0,1. Dans ce cas le chromosome représente simplement une suite de 0 et de 1. Le codage binaire est également indépendant des opérateurs génétiques (croisement et mutation) du moment où ces derniers ne nécessitent aucune spécification. En effet, toute manipulation d'un chromosome donne naissance à un nouveau chromosome valide. Dans la pratique, le codage binaire peut présenter des difficultés. En effet, il est parfois très difficile ou très lourd de coder des solutions de cette manière. En outre, dans certain cas la taille mémoire requise peut devenir prohibitive.

II.8.1.b .Codage réel

Pour certain problème d'optimisation, il est plus pratique d'utiliser un codage réel des chromosomes. Un gène est ainsi représenté par un nombre réel au lieu d'avoir à coder les réels en binaire puis de les décoder pour les transformer en solutions effectives. Le codage réel permet d'augmenter l'efficacité de l'algorithme génétique et d'éviter des opérations de décodage supplémentaires. En effet, un chromosome codé en réels est plus court que celui codé en binaire.

II.8.2. Évaluation : fitness

L'opérateur d'évaluation n'est pas anodin. Il est utilisé par l'opérateur de sélection pour faire son choix des individus à conserver. Ainsi, pour mesurer les performances de chaque individu qui correspond à une solution donnée du problème à résoudre, on introduit une fonction d'évaluation. Elle permet de quantifier la capacité d'un individu à survivre en lui affectant un poids couramment appelé *fitness*. La force de chaque chromosome de la population est calculée afin que les plus forts soient retenus (étape de sélection) puis modifiés (croisement et mutation). La complexité de la fonction d'évaluation dépend essentiellement du problème et de ses contraintes.

Ces deux derniers éléments, codage et évaluation, sont les seuls éléments spécifiques au problème à résoudre. Une fois qu'ils sont fixés, l'algorithme génétique que l'on appliquera sera toujours le même.

II.8.3. Population initiale

Une fois le codage choisi, une population initiale formée de solutions admissibles du problème doit être déterminée. Plusieurs mécanismes de génération de la population initiale sont utilisés dans la littérature. Le choix de l'initialisation se fera en fonction des connaissances que l'utilisateur a sur le problème. S'il n'a pas d'informations particulières, alors une initialisation aléatoire, la plus uniforme possible afin de favoriser une exploration de l'espace de recherche maximum, sera la plus adaptée. Mais dans d'autres cas, il est possible d'utiliser d'autres mécanismes. Par ailleurs, cette étape présente un problème principal qui est celui du choix de la taille de la population. En effet une population trop grande augmente le temps de calcul et demande un espace mémoire considérable, alors qu'une population trop petite conduit à l'obtention d'un optimum local.

II.8.4. Critère d'arrêt

Déterminer l'arrêt d'un processus génétique est l'une des difficultés majeures de l'approche génétique. En effet, si l'on excepte le cas des problèmes artificiels, on ne sait jamais si l'on a trouvé l'optimum. Dans la pratique, l'utilisateur déclare un nombre de générations maximum. La recherche peut également être stoppée lorsque tous les individus d'une même population sont des copies d'un même individu. On dit alors qu'il y a " perte de diversité génétique " .

Les critères d'arrêt se résument alors en :

1. Arrêt après un nombre de générations fixé à priori.
2. Arrêt lorsque la population cesse d'évoluer ou en présence d'une population homogène.

II.8.5. Sélection

L'opérateur de sélection est chargé de " favoriser" les meilleurs individus. Plus formellement, l'opérateur de sélection va générer à partir de la population courante une nouvelle population par copie des individus choisis de la population courante. La copie des chaînes s'effectue en fonction des valeurs de la fonction d'adaptation.

Ce procédé permet de donner aux meilleures chaînes, une probabilité élevée de contribuer à la génération suivante. Cet opérateur est bien entendu une version artificielle de la sélection naturelle, la survie darwinienne des chaînes les plus adaptées .

Il existe de nombreuses techniques de sélection, les plus courantes seront évoquées dans la section suivante.

- **La sélection par classement** : elle consiste à ranger les individus de la population dans un ordre croissant (ou décroissant selon l'objectif) et à retenir un nombre fixé de génotypes. Ainsi, seuls les individus les plus forts sont conservés. L'inconvénient majeur de cette méthode est la convergence prématurée de l'algorithme génétique. Il est parfois nécessaire de garder quelques individus jugés faibles pour créer la diversité au niveau de la population. Une autre difficulté consiste à fixer une limite à la sélection ce qui empêche parfois de garder des bons candidats pour les futures générations.

- **La sélection par la roulette :** elle consiste à créer une roue de loterie biaisée pour laquelle chaque individu de la population occupe une section de la roue proportionnelle à sa valeur d'évaluation. Ainsi, même les individus les plus faibles ont une chance de survivre.

Si la population d'individus est de taille égale à N, alors la probabilité de sélection d'un individu x_i notée $p(x_i)$ est égale à :

$$p(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)} \qquad \text{(II.1)}$$

En pratique, on calcule pour chaque individu x_i sa probabilité cumulée $q_i = \sum_{j=1}^{i} p(x_j)$ et on choisir aléatoirement un nombre r compris entre 0 et 1. **[03]**

L'individu retenu est x_1 si $q_1 \geq r$ ou x_i $(2 \leq i \leq N)$ si $q_{i-1} < r < q_i$ .ce processus est répété N fois. Avec une telle sélection, un individu fort peut être choisi plusieurs fois .par contre, un individu faible a moins de chance d'être sélectionné. C'est cette sélection qui a été exclusivement utilisée dans ce mémoire.

♦ **La sélection par tournoi** : elle consiste à choisir aléatoirement deux ou plusieurs individus et à sélectionner le plus fort. Ce processus est répété plusieurs fois jusqu'à l'obtention de N individus. L'avantage d'une telle sélection est d'éviter qu'un individu très fort soit sélectionné plusieurs fois.

II.8.6. Croisement

La naissance d'un nouvel individu, nécessite la prise aléatoire d'une partie des gènes de chacun des deux parents. Ce phénomène, issu de la nature est appelé croisement (crossover). Il s'agit d'un processus essentiel pour explorer l'espace des solutions possibles. Une fois la sélection terminée, les individus sont aléatoirement répartis en couples. Les chromosomes parents sont alors copiés et recombinés afin de produire chacun deux descendants ayant des caractéristiques issues des deux parents. Dans le but de garder quelques individus parents dans la prochaine population, on associe à l'algorithme génétique une probabilité de croisement, qui permet de décider si les parents seront croisés entre eux ou s'ils seront tout simplement recopiés dans la population suivante . **[03]**

La littérature définit plusieurs opérateurs de croisement. Ils diffèrent selon le type de codage adapté et la nature du problème traité.

II.8.6.a. Croisement binaire

Ce croisement peut avoir recourt à plusieurs types en occurrence:

♦ **Croisement en 1-point :** c'est le croisement le plus simple et le plus connu dans la littérature. Il consiste à choisir au hasard un point de croisement pour chaque couple de chromosomes. Les sous-chaînes situées après ce point sont par la suite inter-changées pour former les deux fils (Figure 2). **[03]**

Parent1 :	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
Parent2 :	1	1	0	0	1	0	0	1
Fils 1 :	**0**	**1**	**1**	**0**	**1**	0	0	1
Fils 2 :	1	1	0	0	1	**1**	**0**	**1**

FIG.II.2- Croisement en un point de deux chromosomes

♦ **Croisement en n-points :** Ce type de croisement s'énonce par un choix aléatoirement de n-points de coupure pour dissocier chaque parent en n+1 fragments. Pour former un fils, il suffit de concaténer alternativement n+1 sous chaînes à partir des deux parents. Ce croisement cherche à explorer tout l'espace de solutions possibles en créant des descendants ayant des caractéristiques très loin des parents.

♦ **Croisement en 2-points :** c'est un cas particulier du croisement en n-points. On choisit aléatoirement deux points de coupure pour créer les descendants.

♦ **Croisement uniforme :** cette technique génère des progénitures gène par gèneà partir des deux parents. Il existe des versions distinctes de ce croisement. La plus connue est celle qui utilise un masque. S'il est égal à 1, l'enfant 1 reçoit l'allèle correspondant du parent 1 et l'enfant 2 reçoit celui du parent 2. Sinon, l'échange se fait dans l'autre sens (Figure 3).

Parent1 :		0	1	1	0	1	1	0	1
Parent2 :		1	0	0	0	1	0	1	1
Masque		0	1	0	1	0	0	1	1
Fils1	:	1	1	0	0	1	0	0	1
Fils2	:	0	0	1	0	1	1	1	1

FIG.II.3- Croisement uniforme

II.8.6.b. Croisement réel

Le codage réel requiert des opérateurs génétiques spécifiques pour la manipulation des chromosomes. Il est de plusieurs types :

♦ **Ordre de base cyclique** : pour créer un fils, il suffit de copier une sous-chaîne d'un parent et de compléter les gènes manquants à partir de l'autre parent, en maintenant l'ordre des gènes. Généralement, une fois deux chromosomes parents sélectionnés pour le croisement, deux points de coupures sont choisis aléatoirement sur chaque parent. Ensuite on place les sous-chaînes entre les points de coupure sur les deux fils dans la même position que les parents. Pour compléter les gènes manquants du fils 1, on commence par insérer les gènes situés à droite du

deuxième point de coupure du parent 2 tout en gardant l'ordre des gènes et en ignorant les gènes déjà pris. Le deuxième fils est complété à partir du parent 1 de la même manière que le fils 1. La Figure 4 montre sur un exemple des étapes de ce type de croisement.

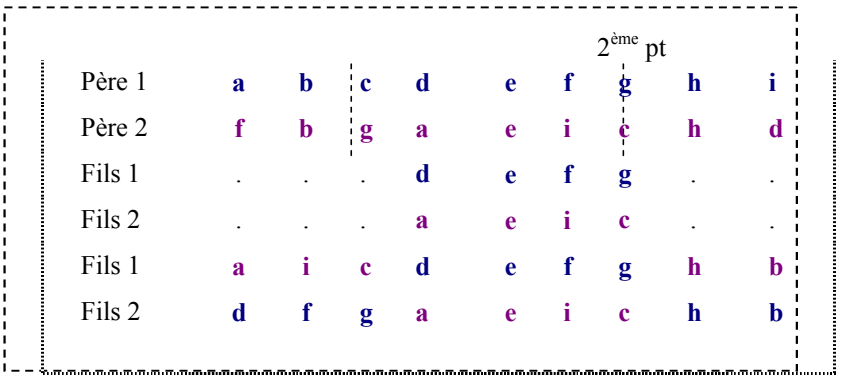

FIG.II.4- Croisement d'ordre de base cyclique

♦ **Croisement d'ordre maximal** : ce type de croisement a pour objectif de garder le maximum possible les positions et l'ordre des gènes. On commence par choisir aléatoirement deux points de coupure. Les sous-chaînes situées au milieu sont inter-changées. Les gènes manquants sont par la suite complétés à partir de chaque père en allant de gauche à droite et en choisissant le premier caractère disponible. A la différence du croisement de base cyclique, le fils 1 est complété à partir du parent 1 et le fils 2 à partir du parent 2. La Figure 5 illustre un exemple de croisement d'ordre maximal.

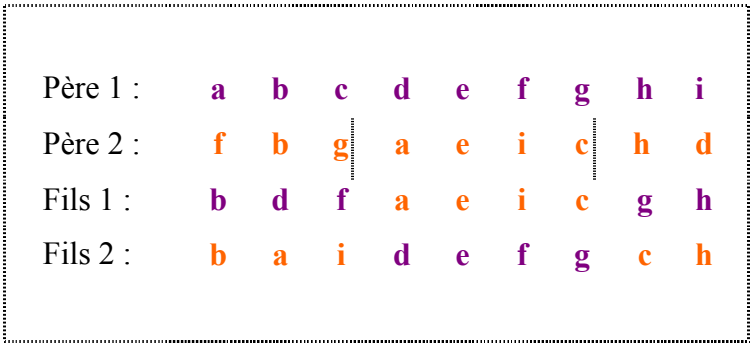

FIG.II.5- Croisement d'ordre maximal

II.8.7. Mutation

La mutation est définie étant la modification aléatoire d'une partie d'un chromosome. Elle constitue une exploration aléatoire de l'espace des chaînes [Gol94]. C'est un phénomène qui a un rôle théoriquement plus marginal : il est là pour éviter une perte irréparable de la diversité. Différentes manières de mutation d'un chromosome sont aussi définies dans la littérature. **[03]**

II.8.7.a. Mutation en codage binaire

Dans un algorithme génétique simple, la mutation en codage binaire est la modification aléatoire occasionnelle (de faible probabilité) de la valeur d'un caractère de la chaîne.

II.8.7.b .Mutation en codage réel

Pour le codage réel, les opérateurs de mutation les plus connus et les plus utilisés sont les suivants :

- L'opérateur **d'inversion simple** : consiste à choisir aléatoirement deux points de coupure et inverser les positions des bits situés au milieu.

- L'opérateur **d'insertion** : consiste à sélectionner au hasard un bit et une position dans le chromosome à muter, puis à insérer le bit en question dans la position choisie.

- L'opérateur **d'échange réciproque** : cet opérateur permet la sélection de deux bits et les inter changés.

L'utilisation de probabilités ne signifie pas que la méthode n'est qu'une exploration aléatoire. Les AGs utilisent des choix aléatoires comme des outils pour guider l'exploration à travers les régions de l'espace de recherche, avec une amélioration probable.

II.9.Valeurs des paramètres

Les paramètres qui conditionnent la convergence d'un algorithme génétique sont :

- ✓ la taille de la population d'individus ;
- ✓ le nombre maximal de générations ;
- ✓ la probabilité de croisement ;
- ✓ la probabilité de mutation.

Les valeurs de tels paramètres dépendent fortement de la problématique étudiée. Ainsi il n'existe pas de paramètres qui soient adaptés à la résolution de tous les problèmes qui peuvent être posés à un algorithme génétique. Cependant, certaines valeurs sont souvent utilisées (définies dans la littérature) et peuvent être de bons points de départ pour démarrer une recherche de solutions à l'aide d'un AG.

- la probabilité de croisement est choisie dans l'intervalle [0.7, 0.99] ;
- la probabilité de mutation est choisie dans l'intervalle [0.001,0.01].

Trouver de bonnes valeurs à ces paramètres est donc un problème parfois délicat.

II.10. Applications

Ayant été reconnue comme une approche valide des problèmes nécessitant une exploration performante et économique du point de vue calcul, les algorithmes génétiques sont maintenant appliqués plus largement, aux domaines des affaires, à la recherche scientifique en général, ainsi que pour l'industrie. Les raisons de ce nombre grandissant d'applications sont claires. Ces algorithmes sont simples d'un point de vue de calcul, cependant très performants dans leur recherche d'amélioration.

II.11. Conclusion

Dans ce chapitre, nous avons introduit les algorithmes génétiques, métaphores biologiques inspirées des mécanismes de l'évolution darwinienne et de la génétique, et utilisés comme outils d'optimisation ou de recherche combinatoire. Nous avons examiné leur terminologie de base, leurs principes, les opérateurs participants à l'exploration de l'espace de recherche tout en mettant l'accent sur leur utilité, leur conception et leur mode de fonctionnement. Ces algorithmes sont souvent associés à un processus stochastique, pour éviter qu'un mauvais choix de départ ou des biais systématiques ne les pénalise. Aussi, nous avons exposé d'une manière générale les méthodes les plus utilisées pour chaque opérateur génétique à savoir sélection, mutation et croisement.

Chapitre 03 :

Présentation Du Simulateur

III.1. Introduction :

Nous présentons dans ce chapitre la description détaillée des méthodes que nous avons utilisé pour l'optimisation par les algorithmes génétiques et le champ de potentiel artificiel pour la planification de la trajectoire le plus court chemin du robot Khepera.

Nous présentons aussi les différentes composantes de l'application, ainsi que les divers techniques implémentées.

III.2.Présentation des environnements :

Premièrement nous avons choisi un environnement libre qui ne contient aucun obstacle.

Et pour l'optimisation par les algorithmes génétiques nous avons choisi deux environnements contraignants, le premier c'est un environnement à deux obstacles, et le deuxième a quatre obstacles.

Chaque obstacle est représenté par un rectangle, cette représentation permet de choisir les coins des rectangles comme des positions nécessaires pour le déplacement du robot et pour l'évitement des obstacles.

Dans la partie du champ de potentiel artificiel nous avons choisi un environnement plein d'obstacle (contraignant) qui contient six obstacles qui sont représentés par des cercles.

Dans cet environnement nous avons planifié la trajectoire du robot Khépéra, puis nous avons planifié le trajectoire du trois robots sur le même environnement.

III.3.Optimisation par les algorithmes génétiques :

III.3.1. Le codage :

Le codage que nous avons retenu est simple, c'est le codage binaire. La longueur de chaque gène est le nombre minimal de bits que nous pouvons coder tous les coins qui se trouvent dans l'environnement.

Et chaque individu contient un nombre minimal de bit de gène multiplié par le nombre minimal des coins que le robot peut les passe.

III.3.2. La fonction principal d'algorithme génétique :

La réalisation de la fonction d'algorithme génétique consiste à développer tous les fonctions utilisées dans notre fonction principale suivante :

```
Algorithme génétique (N=taille_population, K=taille_d'individu, nb_génération,pc,pm)
    Début
        Initialisation de N individus de K bit ;
        Pour g=1 à nb_génération
            Evaluation de la population ;
            La sélection ;
            Croisement suivant pc ;
            Mutation suivant pm ;
            Mise à jour de la population courante ;
        Fin pour.
    Fin.
```

Pour la première génération : la population initiale, les individus sont généralement créés au hasard. Une fois que les individus générés, il est nécessaire d'évaluer chacune d'elles. Une fonction d'évaluation (FE ou fitness function) doit donc être définie.

Ainsi, plus la valeur de la FE d'un individu est élevée, plus ses allèles (valeur de ses gènes) sont pertinents. Cette évaluation se fait le plus souvent à l'aide de plusieurs critères. Dans un deuxième temps, les couples d'individus (parents) qui pourront se reproduire sont sélectionnés.

Les principes généraux de croisement et de mutation sont les suivants : pour le croisement deux individus sont sélectionnés suivant la probabilité $\mathbf{p_c}$ .ces deux parents seront donc compilés pour générer deux enfants. Ils peuvent être combinés par des croisements mono ou multipoints. Dans tous les cas, les enfants posséderont une partie des gènes d'un parent et une partie de l'autre. Pour la mutation, un individu est sélectionné, un ou plusieurs de ses gènes est transformé suivant la probabilité $\mathbf{p_m}$.

III.3.2.1. L'initialisation :

La population initiale sera composée des N individus générées aléatoirement. Chaque individu contient K bits.

```
Fonction initialisation (N=taille_population, K=taille_d'individu)
    Début
      Pour i=1 a N faire
          Pour j=1 a K faire
              Parent_i [i, j]=Générer  aléatoirement 1 ou 0 ;
          Fin pour
      Fin pour
    Fin
```

III.3.2.2. La procédure d'évaluation :

Pour l'évaluation de chaque individu on utilise une fonction qui calcul la somme de nombre d'obstacle qui se trouve entre chaque deux positions et la distance total entre la position initiale et finale.

Et pour l'adaptation de la valeur de nombre d'obstacle et la distance, on utilise la fonction suivante :

F = (adaptobs+adaptdis)

Adaptobs :(nombre max des obstacle qui peuvent trouve dans le trajectoire) –(nombre des obstacles qui se trouvent entre chaque deux position successive).

Adaptdis :(nombre max de distance – distance entre position initiale et finale)/100

III.3.2.3. La procédure de sélection :

La procédure de sélection que nous avons utilisée est la roulette biaisée de Goldberg « roulette Wheel » cette dernière fonctionne sur le principe suivant :

Chaque individu i est représenté par sa fitness tel que l'équation (II.1).

On procède alors à la sélection de N individus survivants en effectuant N lignes aléatoire dans le domaine [0,1]. Soit R le nombre aléatoire tiré, l'individu j sélectionne sera :

$$\sum \geq R$$

(Selon cette procédure de sélection, plus un individu est fort, plus il y aura des chances de survivre)

III.3.2.4. Le croisement :

Les individus survivants à la phase de sélection sont appariés aléatoirement et chaque paire formée va subir le croisement avec une probabilité pc, de nombreux types de croisement différents existent dans la littérature ,préservant plus ou moins l'identité génétique des parents et permettent de se déplacer dans tout l'espace des grande, plus la convergence de l'algorithme est rapide (mais plus on risque de convergence vers un optimum local),nous avons utilisé le croisement à 1 point , pour se faire ça on génère un point entier (cpoint) entre (1 et le longueur d'individus -1) à l'aide des fonctions round et rand (sous matlab) puis nous avons fait le croisement à partir de ce point.

```
Fonction croisement (parent1,parent2,pc)
    Début
        Si (random<pc)
            Générer un point (cpoint) entre (1 et K-1) ;
            Fils_1=parent_1 [1, cpoint] U parent_2 [cpoint, K] ;
            Fils_2= parent_2 [1, cpoint] U parent_1 [cpoint, K] ;
            Evaluer les fils ;
        Sinon
            Fils_1=parent_1 ;
            Fils_2=parent_2 ;
        Fin si
    Fin
```

III.3.2.5. La mutation :

Les individus de la population issus du croisement vont ensuite subir à un processus de mutation avec une probabilité $\mathbf{p_m}$. Nous présentons ici la plus simple qui consiste à modifier un bit aléatoirement d'un individu, cette méthode est expliquée sur la fonction suivante :

```
Fonction mutation (parent, pm)
    Début
        Si (random<pm)
            Générer un point (cpoint) entre (1 et K) ;
            Fils [mpoint]=valeur absolu (parent [mpoint]-1) ;
            Evaluer le fils ;
        Sinon
            Fils=parent ;
        Fin si
    Fin
```

La mutation évite la dégénérescence de la population (en d'autre terme, elle permet de quitter les extrêmes locaux)

III.4. Environnement du travail :

Ce logiciel a été développé sur une machine de type Intel pentium IV, dans l'environnement WINDOWS XP, avec l'utilisation de MATLAB 7.1 comme un langage de programmation, facile et simple.

III.5. Présentation du logiciel :

Dans la figure ci-dessous, on trouve la fenêtre principale de notre simulateur.

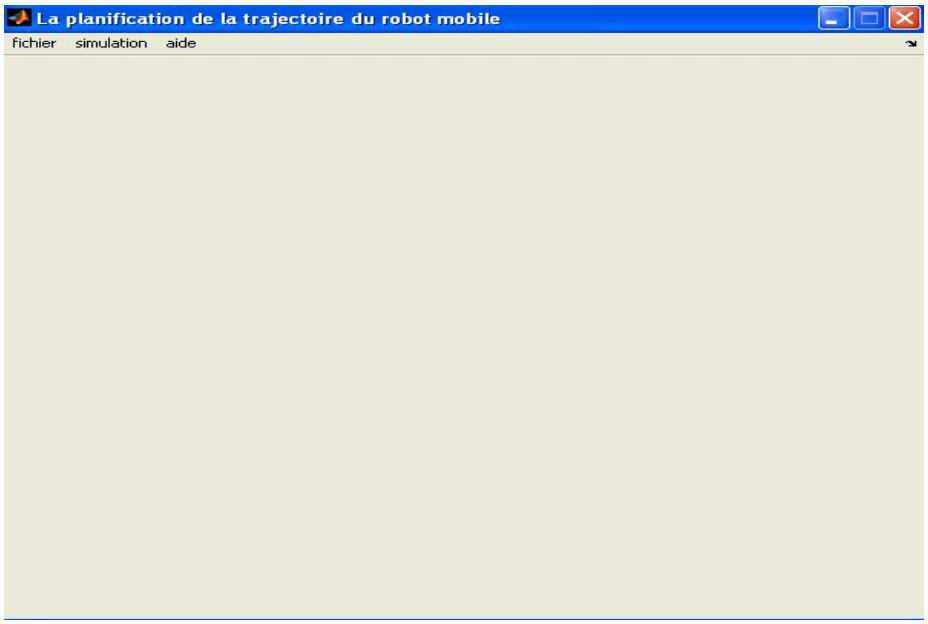

FIG.III.1 la fenêtre principale du notre logiciel

III.5.1 .barre de menu:

On trouve dans la barre de menu trois boutons principaux : fichier, simulation, aide.

Chaque bouton contient des boutons secondaires illustrés par la figure suivante :

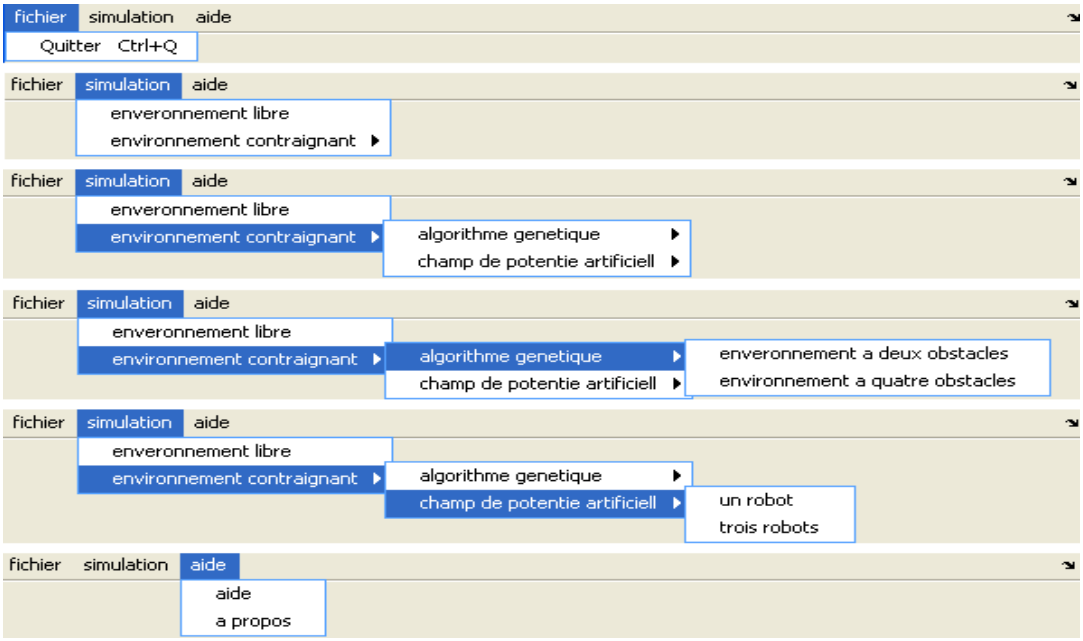

FIG.III.2. les boutons de la barre de menu.

III.5.2.La description détaillée de chaque bouton :

✓ Le bouton fichier :

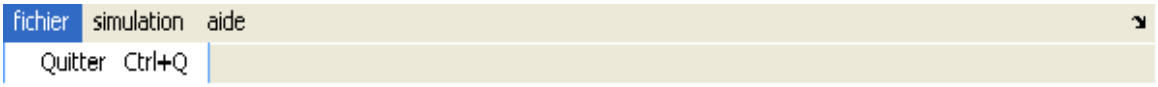

FIG.III.3. le bouton Quitter.

Ce bouton (Quitter) pour quitte le simulateur.

✓ Le bouton environnement libre :

FIG.III.4. le bouton environnement libre.

Ce bouton permet le passage à la figure de l'environnement libre, illustré ci-dessous :

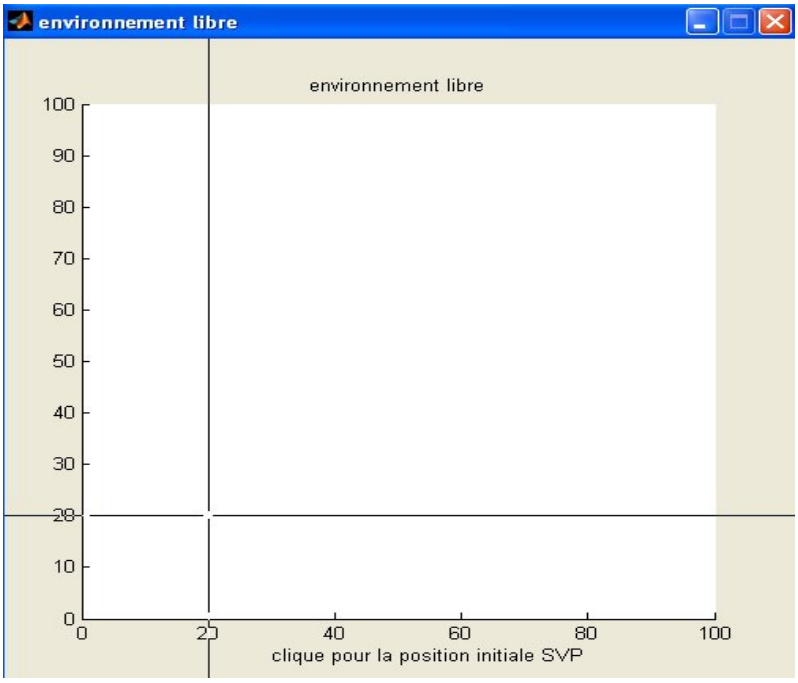

FIG.III.5. L'environnement libre (position initiale).

Ce repaire qu'apparu dans l'environnement de simulation permet de définir la position initiale du robot après la clique de la souris.

Puis on choisi la position finale de même manière. Comme la figure suivante :

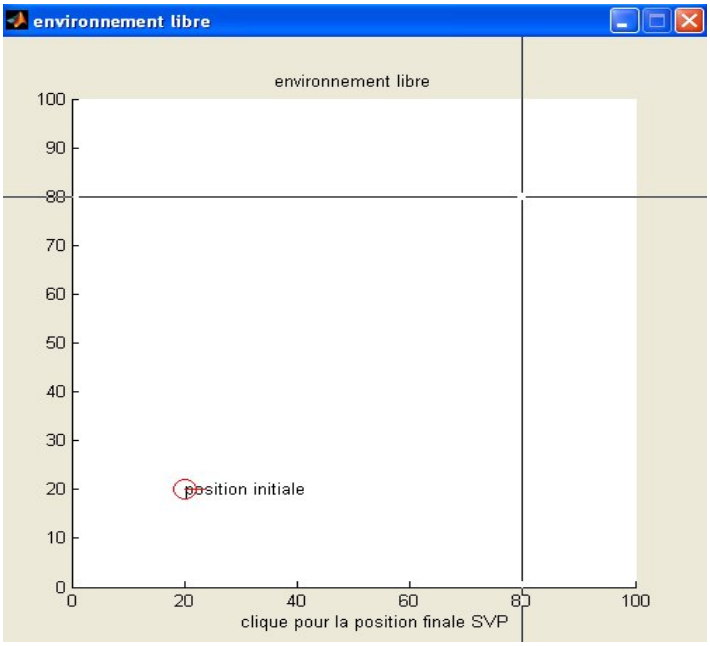

FIG.III.6. L'environnement libre (position finale).

Après la définition de la position initiale et la position finale.

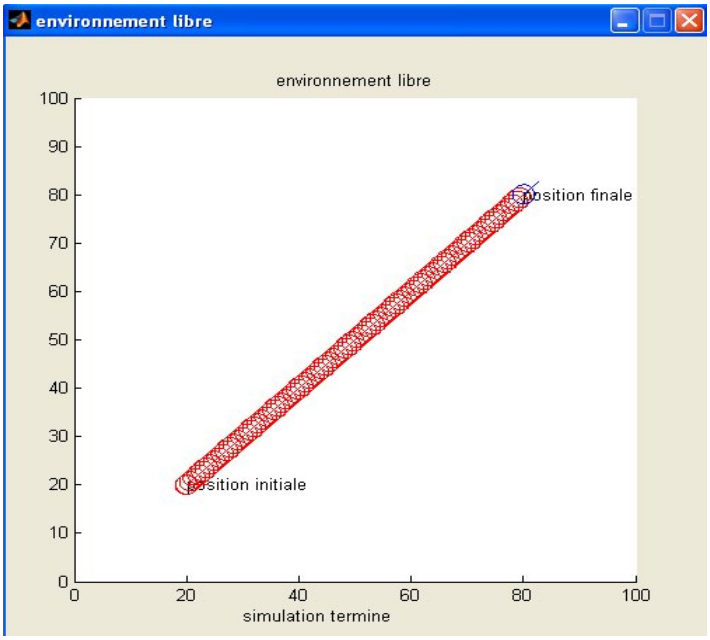

FIG.III.7. L'environnement libre (trajectoire du robot).

On remarque ici que le robot suit une ligne droite, puisque on n'a aucun obstacle entre la position initiale et la position finale.

Dans l'environnement contraignant on applique deux méthodes qui sont algorithme génétique et le champ de potentiel artificiel.

✓ Le bouton environnement a deux obstacles (AG):

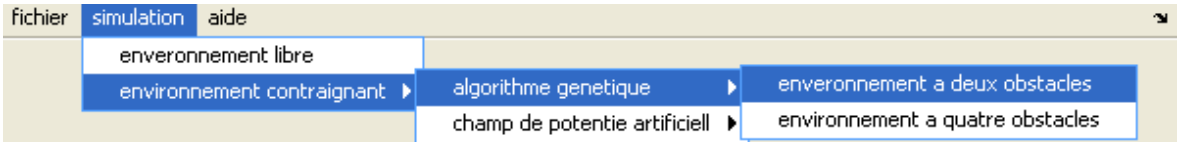

FIG.III.8. le bouton environnement a deux obstacles (AG).

Ce bouton permet le passage à l'environnement a deux obstacles, illustré par la figure suivante (Fig.III.9) :

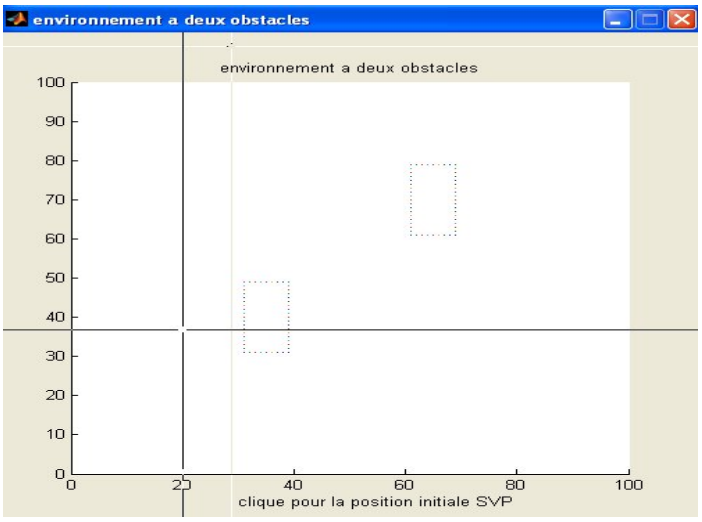

FIG.III.9. L'environnement a deux obstacles (position initiale).

Après la définition de la position initiale et la position finale, la simulation est commence et désigné la trajectoire du robot, et nous avons présenté aussi l'évolution du meilleur individu de chaque population les figures (fig.III.10 et fig.III.11) illustrent ça.

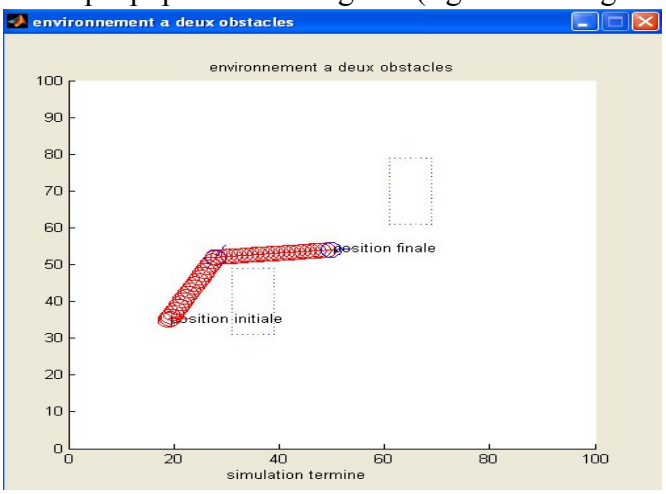

FIG.III.10. L'environnement a deux obstacles (la trajectoire).

Dans cette figure on observe que le robot suit le plus court chemin en évitant l'obstacle.

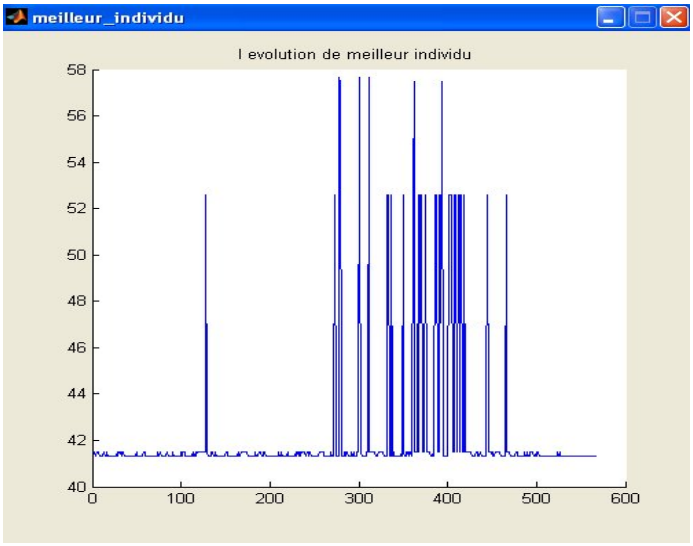

FIG.III.11. L'environnement a deux obstacles (meilleur individu).

Dans cette dernière figure, nous avons représenté l'évolution des meilleurs individus, on observe que la valeur d'individu est stabilisé a 40.96, cette valeur représente la longueur de trajectoire le plus court chemin qu'illustré par la figure (fig.III.10).

✓ Le bouton environnement à quatre obstacles :

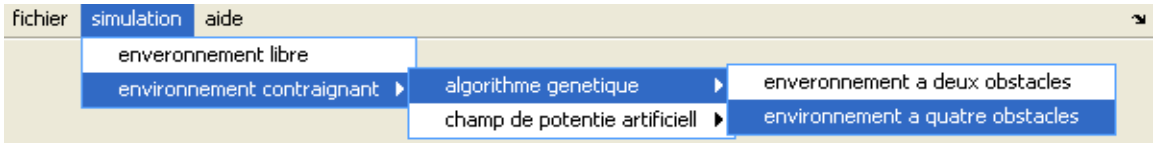

FIG.III.12. le bouton environnement a quatre obstacles.

Ce bouton permet le passage à l'environnement à quatre obstacles, illustré par la figure suivante (fig.III.9) :

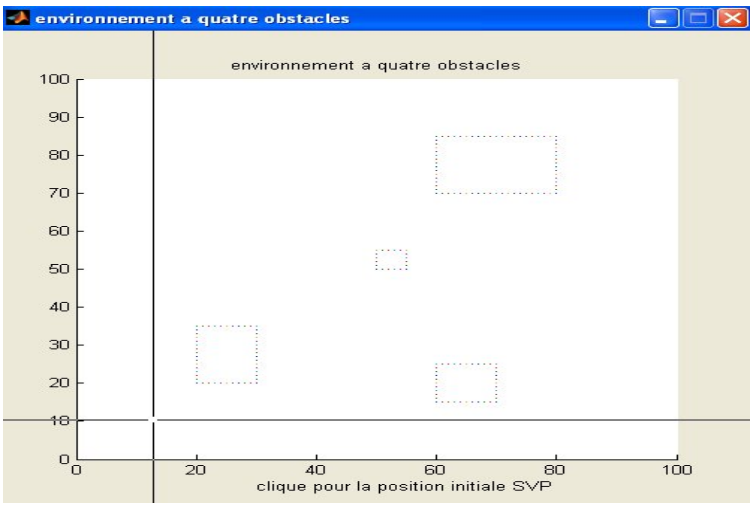

FIG.III.13. L'environnement a quatre obstacles.

Dans cet environnement, on peut choisir la position initiale et la position finale, et après la simulation, en observe la trajectoire du robot et l'évolution des meilleurs individus illustré par les figures suivantes :

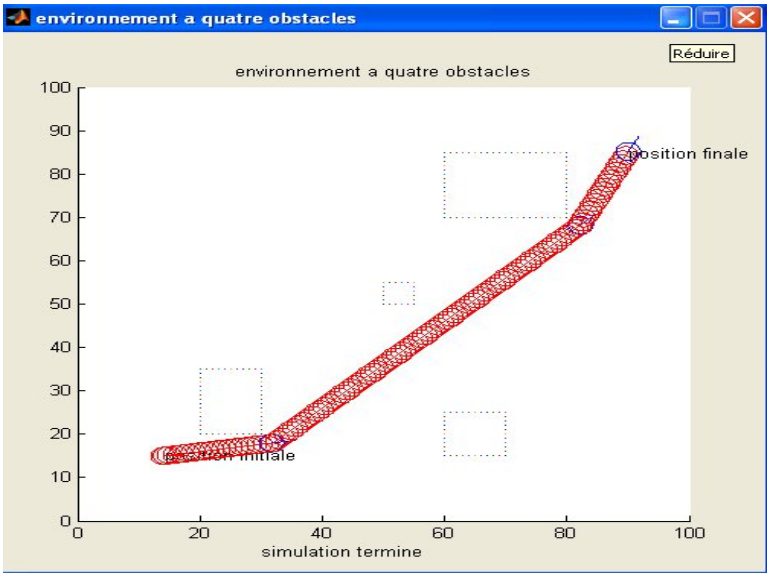

FIG.III.14. L'environnement a quatre obstacles (la trajectoire).

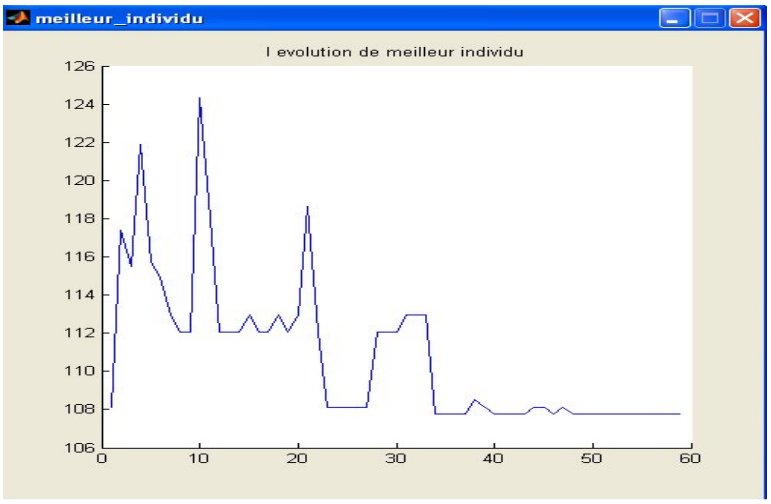

FIG.III.15. L'environnement a quatre obstacles (meilleur individu).

✓ Le bouton champ de potentiel artificiel:

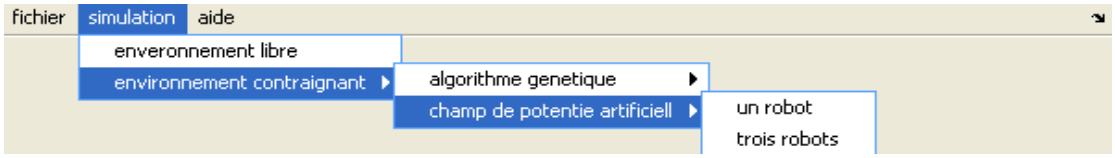

FIG.III.16. Le bouton champ de potentiel artificiel.

Dans notre logiciel nous avons simulé aussi une autre méthode d'établir la trajectoire le plus court chemin à l'aide de champ de potentiel artificiel placé à la position finale « le but » (champ attractif) et à chaque obstacle (champs répulsifs).

Lorsqu'on commence la simulation, par clique a un bouton de champ potentiel (un robot ou trois robots), l'environnement de simulation est apparu, on peut définir la position initiale et la position finale, la figure suivante montre ça.

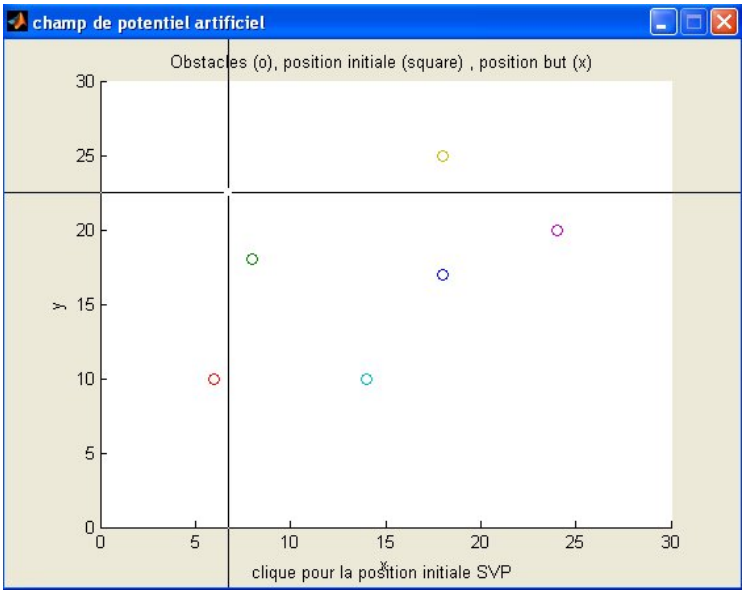

FIG.III.17. L'environnement « champ potentiel » (position initiale).

Cet environnement est apparu après le clique au bouton « un robot » ou bouton « trois robot », il permet de définir la position initiale (ou les positions initiales de chaque robot si on a trois robots) et la position finale.

Après la définition des positions initiales et la position finale, les figures suivantes (fig.III.16; fig.III.17 ; fig.III.18 ; fig.III.19) sont apparu, la figure (fig.III.16) représente les champs potentiels répulsifs de chaque obstacles dans l'environnement.

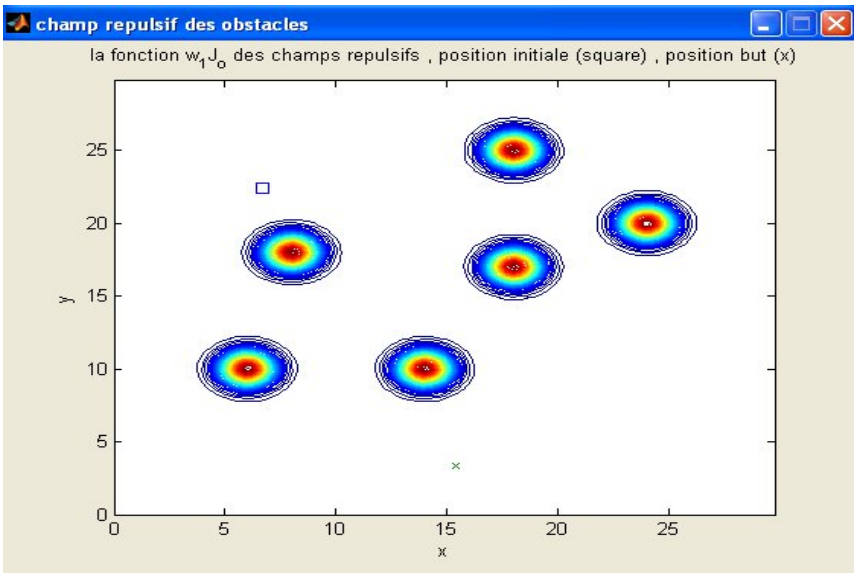

FIG.III.18. Le champ potentiel répulsif de chaque obstacle.

La figure (fig.III.17) représente le champ potentiel attractif de la position finale (le but)

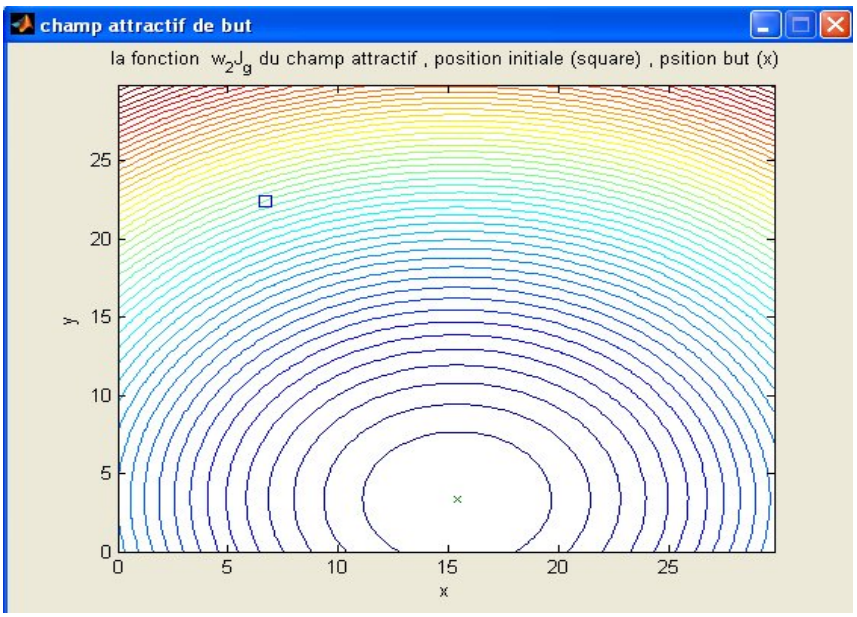

FIG.III.19. Le champ potentiel attractif du but.

La figure (fig.III.18) représente l'intersection entre les champs répulsifs des obstacles el le champ attractif du but.

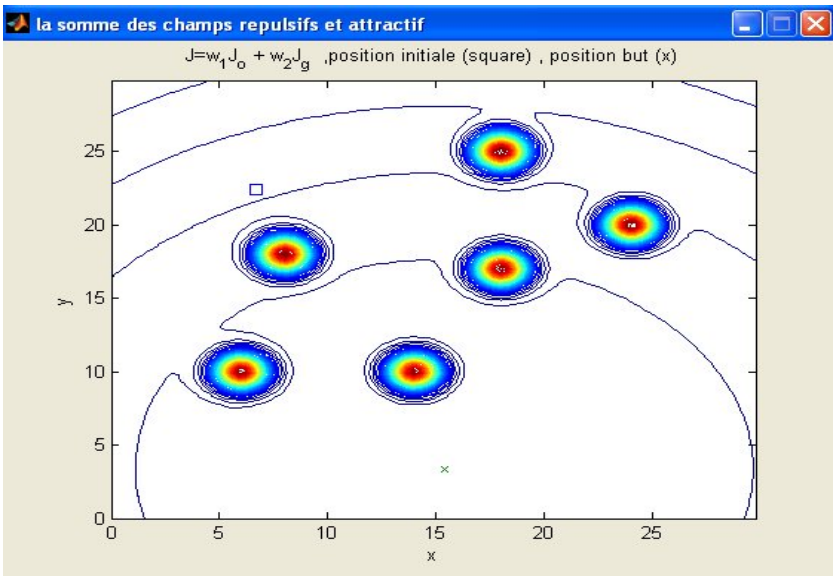

FIG.III.20. L'intersection du champ répulsif et attractif.

La figure (fig.III.19) représente la trajectoire du robot commence a partir d'une position initiale à la position finale.

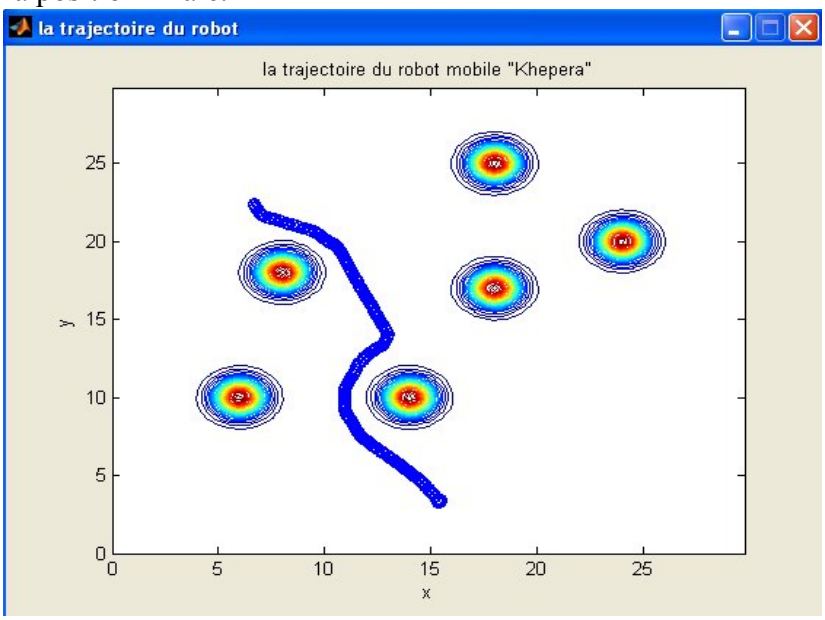

FIG.III.21. La trajectoire du robot.

Dans le cas des trois robots la figure suivante (fig.III.20) illustré exemple de trajectoire des trois robots.

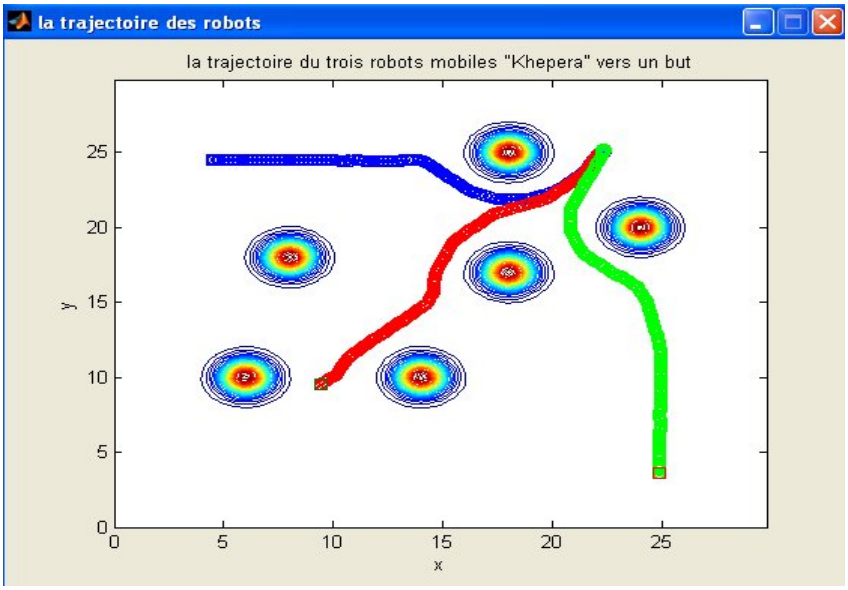

FIG.III.22. La trajectoire du trois robots.

✓ Le bouton aide: permet d'afficher des informations générales sur l'application

FIG.III.23. Le bouton aide.

✓ Le bouton a propos: Affiche des informations sur le projet

FIG.III.24. Le bouton a propos.

III.6. Conclusion :

On essayant dans ce chapitre de présenter le simulateur que l'on a réalisé, notre logiciel est simple et flexible pour l'utilisateur, il comporte les méthodes et les fondements que nous avons étudiés et présentés dans notre projet, dont principalement la modélisation cinématique et les algorithmes génétiques, et sans oublier aussi la méthode du champ de potentiel artificiel pour établir la trajectoire.

Conclusion Générale

CONCLUSION GENERALE

L'objectif principal de notre travail était de faire la planification d'une trajectoire d'un robot mobile unicycle (Khepera) optimisé par les algorithmes génétique et le champ de potentiel artificiel qui ont été réalisé avec succès.

A cet effet, nous avons commencé par la présentation de la robotique mobile, et les algorithmes génétiques qu'est un outil efficace pour une classe de problème très large.

Ensuite, nous avons présenté en détails le modèle cinématique du robot mobile Khepera et les méthodes de perception, et comment faire l'étape d'optimisation par les algorithmes génétiques : le codage, les operateurs génétiques comme la sélection, croisement, et la mutation.

Nous avons décomposé notre application en deux parties : dans la première, nous avons utilisé comme un outil d'optimisation les algorithmes génétiques. Ce type d'optimisation nécessite de maximiser la taille de la population initiale et le nombre d'itérations pour atteindre le bon résultat, ça conduite de prendre un temps considérable pour l'exécution.

Dans la deuxième nous avons appliqué une autre méthode pour la planification de trajectoire du robot mobile qu'est la planification par champ de potentiel artificiel.

Notre outil est caractérisé par une interface graphique claire, et simple à utiliser, et qui permet aux utilisateurs un grand degré d'interaction à l'aide des animations et des messages explicatifs.

Ce travail nous a permis de :

- ✓ De bien comprendre les algorithmes génétiques et le champ potentiel.
- ✓ Bien comprendre quelques notions de bases qui concerne la modélisation cinématique et les systèmes de perception pour les robots mobiles.
- ✓ Avoir une bonne maitrise de logiciel de programmation (MATLAB 7.1 et sa GUI).
- ✓ D'avoir une bonne expérience dans le domaine de recherche.

A ce point proposons les extensions futures suivantes :

- ✓ Proposer l'étude d'autre algorithmes de planification de la trajectoire de robot mobile comme :
  - Algorithme A/A*.
  - Algorithme de Dijkstra.
  - Algorithme de Bellman-Ford-Moore.
  - Algorithme de Floyd-Warshall.

Annexe A :

La Méthode de Champ de Potentiel Artificiel

Méthode du champ de potentiel Artificiel:

Depuis qu'un siècle, Lord Kelvin a remarqué qu'un système mécanique dissipatif perd son énergie potentielle le long des ses trajectoire et qu'en conséquent les trajectoires passant au voisinage d'un minimum d'énergie potentielle convergent asymptotiquement vers ce minimum. Dans le contexte de la robotique, ce type à inspiré O.Khatib, qui a proposé une méthode d'évitement d'obstacle en définissant dans l'espace d'évolution du robot un champ de potentiel artificiel analogue au champ de gravité [Kha80]. Dans cette méthode devenue classique, le but à atteindre joue un rôle d'un pôle attractif, alors que les obstacles génèrent un champ répulsif. Le robot suit alors le gradient de potentiel ainsi défini, jusqu'à atteindre le minimum.

Cette méthode est bien adaptée à la problématique du contrôle du mouvement des robots mobiles : la force dérivée du potentiel permet de calculer une direction de déplacement et une accélération proportionnelle à son intensité, et est donc facilement transformable en commande dynamique (accélération) ou cinématique (vitesse ou position).

Les avantages de la méthode de champ de potentiel :

- Simplicité algorithmique de la mise en œuvre (la méthode est analytique) ;
- Adaptabilité à de nombreux types de capteurs extéroceptifs ;
- Possibilité de traiter aussi bien les obstacles fixes que les obstacles mobiles.

Formalisation de la méthode :

Considérons un obstacle O_i, que nous représentons par une fonction analytique dans le plan (O, *x, y*). Si X_g le vecteur qui représente la position du but et X celle du robot (X est le vecteur d'état du robot), nous pouvons écrire la formule du potentiel artificiel *U(X)* applique au robot :

$$U(X) = U_g(X) + U_{O_i}(X) \qquad \text{(A.1)}$$

où

$U_g(X)$: Le potentiel attractif produit par le but en X

$U_{O_i}(X)$: le potentielle répulsif induit par l'obstacle en X.

Nous pouvons alors calculer la force résultante F par :

$$F = F_g + F_{O_I} \tag{A.2}$$

$$\begin{aligned} F_g(X) &= -\overrightarrow{grad}[U_g(X)] \\ F_{O_i}(X) &= -\overrightarrow{grad}[U_{O_i}(X)] \end{aligned} \tag{A.3}$$

La force F_g est une force attractive qui permet au point de contrôle du robot d'arrive au but, F_{O_i} est une force induisant une répulsion artificielle de la surface de l'obstacle produite par le potentiel $U_{O_i}(X)$.

Le potentiel attractif $U_g(X)$ est une fonction positive dont la dérivée de premier ordre est continue, et dont le seul minimum est de valeur nulle à $X = X_g$. Par conséquent, le potentiel $U_{O_i}(X)$ sera défini de telle sorte que la fonction de potentiel artificiel *U(X)* soit non négative, continu dérivable et qui atteigne son minimum de valeur nulle à $X = X_g$. La fonction $U_{O_i}(X)$ doit elle-même être continu, non négative, dérivable, sa valeur devant tendre vers l'infini sur la surface de l'obstacle.

Dans le cas général où *n* obstacle sont présents dans l'environnement, le potentiel répulsif $U_O(X)$ est la somme des potentiels répulsifs $U_{O_i}(X)$ générés par chacun des obstacles, et la force résultante totale s'écrit :

$$F = F_g + F_O = F_g + \sum_{i=1}^{n} F_{O_i} \tag{A.4}$$

Cette force résultante F est alors transformée en commende du robot.

La difficulté réside maintenant dans la définition des fonctions de potentiel U_{O_i} et U_g.

Si potentiel attractif $U_g(X)$ et répulsif $U_{O_i}(X)$ vérifie les conditions précédents alors, la fonction $U_O(X)$ doit donc elle-même être continue, dérivable, sa valeur devant tendre vers l'infini sur la surface de l'obstacle. Les fonctions initialement proposées dans [Kha80] sont les suivantes :

- Le potentiel attractif est défini par :

$$U_g = \frac{1}{2} K_g (X - X_g)^T (X - X_g) \quad \text{(A.5)}$$

où K_g est un gain à déterminer. Ce potentiel génère donc une force :

$$F_g = -K_g (X - X_g) \quad \text{(A.6)}$$

- Afin d'éviter que le minimum global du potentiel total diffère de celui du potentiel attractif, la fonction de potentiel répulsif ne doit avoir une influence que dans une région délimité autour de l'obstacle, définir une distance d_0 (distance euclidienne) et la distance d_0 doit donc toujours être inférieur ou égale à la plus courte distance le but et l'obstacle.

Dans [Kha80], ce potentiel est donné par l'équation suivante

$$U_O(X) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d(X)} - \frac{1}{d_0}\right)^2 & \text{Si } d < d_0 \\ 0 & \text{sinon} \end{cases} \quad \text{(A.7)}$$

où d la plus courte distance de l'obstacle O au robot, et η le gain de cette fonction.

Ce potentiel génère la force :

$$F_O(X) = \begin{cases} \eta\left(\frac{1}{d(X)} - \frac{1}{d_0}\right)\frac{1}{d(X)^2}\frac{\partial d}{\partial X} & \text{Si } d < d_0 \\ 0 & \text{sinon} \end{cases} \quad \text{(A.8)}$$

Dans cette équation, le terme $\frac{\partial d}{\partial X}$ représente le vecteur unitaire de la dérivée partielle de la distance d entre le robot et l'obstacle, il détermine la direction de la force à appliquer.

Annexe B :

Les Interfaces Graphiques Sous MATLAB

I. Créer des interfaces avec Matlab :

Un GUI (Graphic User Interface) est constitué d'objets d'interface (unicontrols) dotés de méthodes et de propriétés programmables dans des scripts.

Pour décrire un GUI, Matlab utilise deux fichiers :

- Un fichier figure (d'extension .fig), qui contient le layout du GUI, ou disposition des objets d'interface.
- Un fichier script (d'extension .m), qui contient les comportements de l'interface.

Par exemple, le GUI two_axes ci-dessous est décrit par les deux fichiers two_axes.m et two_axes.fig :

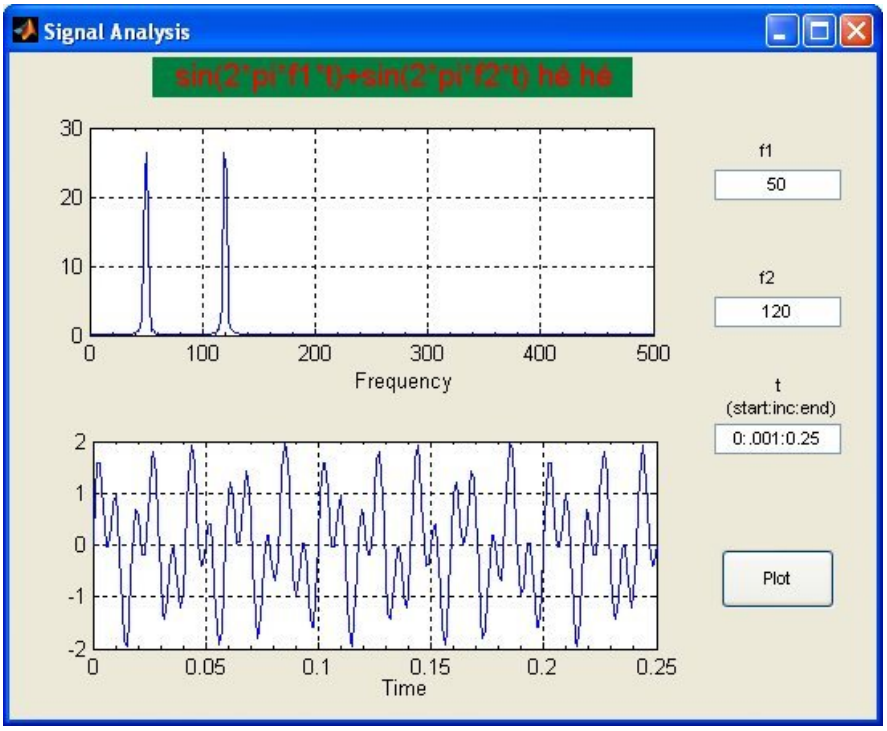

FIG.B.1. Exemple, le GUI deux axes.

II. L'éditeur de GUI :

GUIDE (pour Graphic User Interface Development Editor) contient les outils pour :

1. disposer les objets d'interface (layout)
2. adapter les valeurs de leurs propriétés
3. programmer les comportements (Callbacks)

On exécute GUIDE depuis la fenêtre Matlab :

1. avec la commande : >> guide

2. à l'aide du raccourci dans la barre d'outils

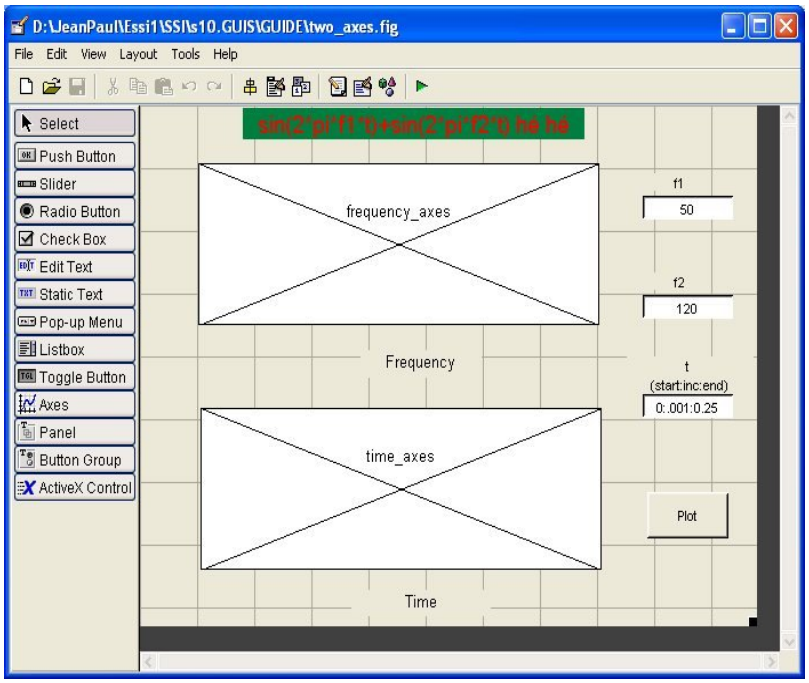

FIG.B.2.Graphic User Interface .

La liste de propriétés (non exhaustive) :

Tous les objets d'interface sont dotés de propriétés accessibles et modifiables.

- ✓ Tag, est le nom de l'objet
- ✓ Style, son type, bouton, texte, panel …
- ✓ Name, Pointer, WindowStyle (figure)
- ✓ String, le texte qui apparaît (text, edit, …)
- ✓ ForegroundColor, …
- ✓ Max, Min, Step (slider)
- ✓ Value
- ✓ Callback, pour coder les comportements
- ✓ ButtonDownFcn, ResizeFcn, CreateFcn …
- ✓ FontSize, FontWeight, FontName, …

L'inspecteur de propriétés est appelé par GUIDE pour modifier les propriétés des objets d'interface

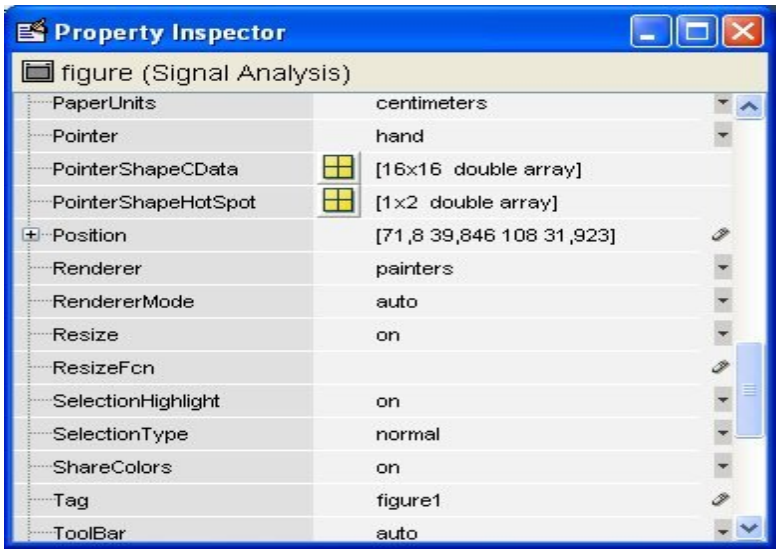

FIG.B.3. L'inspecteur de propriétés.

III. Les propriétés des objets d'interface :

Pour atteindre les propriétés des objets d'interface, on utilise des 'handler'.

À tout objet d'interface, Matlab peut associer des pointeurs, ou **handler,** pour l'accès aux propriétés.

Certains handlers sont définis par défaut :

- ✓ gcf : handler de la figure courante (get current figure)
- ✓ gca : axes courants de tracé (get current axes)
- ✓ gcbf : la figure cliquée (get callback figure)
- ✓ gcbo : l'objet qui appelle (celui sur lequel on a cliqué)
- ✓ Le handler 0 est la fenêtre interpréteur Matlab
- ✓ Le handler 1 est la figure 1, …

Voici des instructions qui utilisent un handler :

- ✓ get et set, respectivement pour lire et pour changer les valeurs des propriétés
- ✓ findobj, pour retrouver le handler d'un objet
- ✓ propedit, pour éditer les propriétés d'un objet.
- ✓ delete, pour effacer un objet.

IV. Comment programmer un GUI :

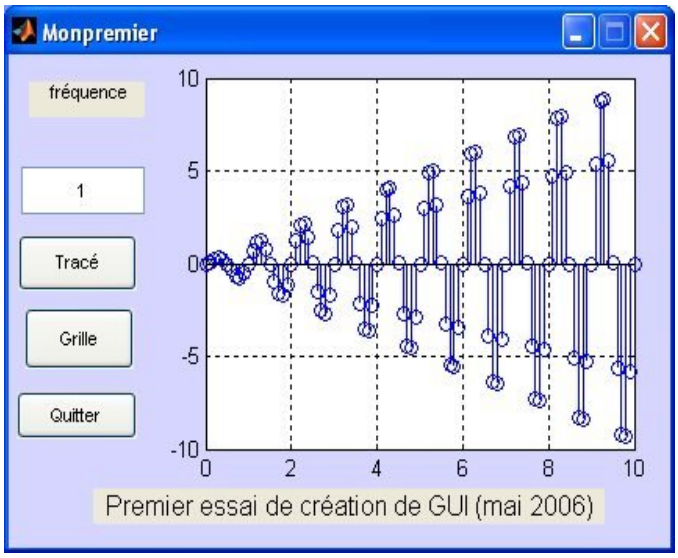

FIG.B.4. Exemple « monpremier »

- dans la propriété ‘Callback’, on place :
  - ✓ Soit une instruction : telle ‘grid’ ou ‘close(gcf)’
  - ✓ Soit un script Matlab placé dans le path, tel 'script.m' pour le bouton 'Tracé'
- pour échanger des informations entre les objets d’interface, on utilise Workspace .
- Pour adresser un objet d’interface.

V. Une autre façon de programmer un GUI en utilisant le fichier script

GUIDE génère automatiquement des méthodes vides pour les callbacks des objets d’interface, il les appelle dans la propriété Callback et leur transmet les deux arguments suivants :

1. hObject, le handler de l'objet appelant
2. Handles, une structure qui contient les handlers de tous les objets d'interface, et permet ainsi l'accès à toutes leurs propriétés.

Pour programmer le GUI, il suffit de compléter ces méthodes créées par GUIDE dans le fichier script du GUI, ce qui évite de multiplier les scripts

Pour les échanges de variables entre les méthodes du GUI, il y a deux possibilités :

- Soit on utilise la structure handles :

```
% callback émetteur
handles.mydata=x % pour transmettre x
guidata(hObject,handles) % sauvegarde
% callback récepteur
y=handles.mydata % récupérer x dans y
```

- Soit on utilise la déclaration global dans toutes les méthodes qui doivent partager une variable

```
global x
x= . .
```

VI. Une autre façon de programmer un GUI en utilisant le fichier script

Voici le script MonSecondGUI.m :

- l'outil f donne la liste des fonctions du script
- l'éditeur donne des diagnostics sur le script
- Dock/Undock, flêches en haut à droite

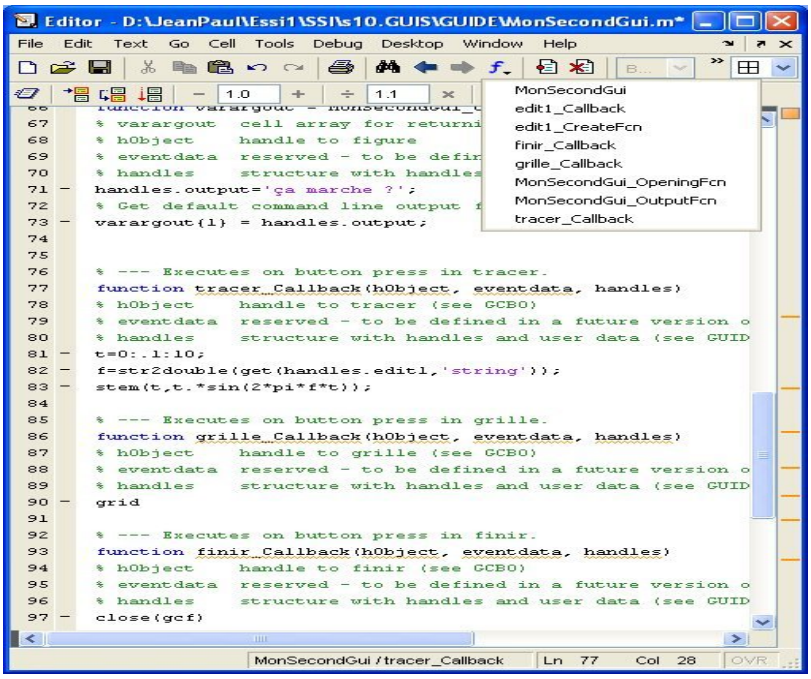

FIG.B.3. Exemple du fichier script

Bibliographie :

[01] : Benmakhlouf Abdeslam ; « Controleur flou pour la navigation d'un robot mobile d'intérieur »;En vue de l'obtention du diplôme de : magister en robotique laboratoire électronique avancée (LEA) Batna.

[02] : Bernard Bayle ; « Robtique mobile » ; Ecole Nationale Supérieure de Physique de Strasbourg; Université Louis Pasteur ; année 2008–2009 ; bernard@eavr.u-strasbg.fr.

[03] : David E.Goldberg ; « Algorithme génétiques » ; traduit de l'anglais par Vincent Corruble; Editions Addition-Wesley ; France

[04] : David Filliat ; « La robotique mobile » cours C10-2 ENSTA ; 2 Octobre 2004.

[05] : Jean Paul Laumond ; « La robotique mobile » ; Paris hermès science publication ; 2001.

[06] : Khiat Abdeaziz & Ouzaid Said ; « Etude de la manœuvrabilité d'un robot mobile autonome non holonome évoluant dans un espace restreint un espace » ; Mémoire d'ingénieur d'état en informatique (INI) ; 1997.

[07] : Lagriffoul Fabien ; Localisation through Supervised Learning ; Département d'informatique ; Université Umea ; Suède; 11Janvier 2005 ; ens03fll@cs.umu.se

[08] : Nassoy François et Ngon Yily ; « Navigation d'un robot autonome » ; Projet informatique Université Cergy-Pontoise - site de St Martin ; Année 2006.

[09] : Sophie Voisin ; « Application des Algorithmes Génétiques » ; Rapport de stage ; Laboratoire Electronique Informatique et Image ; Mars - Juin 2004.