Modélisation de l'environnement par grille adaptive et recherche de chemins pour robot mobile

Valerio Boschian-Campaner

THESE

présentée à

L'U.F.R. MATHEMATIQUE, INFORMATIQUE,

MECANIQUE, AUTOMATIQUE

en vue de l'obtention du

DIPLOME DE DOCTEUR DE L'UNIVERSITE DE METZ
option : Production Automatisée

par

Valerio BOSCHIAN-CAMPANER

MODELISATION DE L'ENVIRONNEMENT

PAR GRILLE ADAPTATIVE ET RECHERCHE DE CHEMINS

POUR ROBOT MOBILE

Soutenue le 10 juillet 1990, devant la commission d'examen:

Rapporteurs:	M.	CROWLEY Jim
	M.	HUSSON René
Examinateurs:	M.	LAURENT Claude
	M.	GARDAN Yvon
	M.	PRUSKI Alain

ISGMP — INSTITUT SUPERIEUR DE GENIE MECANIQUE ET PRODUCTIQUE

à Catherine

à Marc-Matthieu

Cette thèse a été réalisée au Laboratoire d'Automatique et d'Electronique Industrielles de Metz dans le cadre du programme robotique, sous la direction scientifique de M. le Professeur Claude LAURENT, directeur du LAEI. Nous exprimons ici à M. le Professeur Claude LAURENT notre sincère gratitude pour sa grande disponibilité, pour ses conseils ainsi que pour avoir accepté d'être membre de notre jury.

Notre gratitude va aussi à M. le Professeur Bernard MUTEL, précédent directeur du LAEI, pour nous avoir accepté dans son laboratoire et encouragé dans nos recherches.

Nous tenons à exprimer notre vive reconnaissance à M. le professeur Jim CROWLEY et à M. René HUSSON pour avoir accepté de juger notre travail en qualité de rapporteur et membre du jury.

Nous sommes trés honoré par la présence dans ce jury de M le professeur Yvon GARDAN.

Nous tenons à remercier vivement M. Alain PRUSKI, Maître de Conférence à L'Université de Metz, pour nous avoir dirigé et conseillé dans nos recherches.

Enfin, tous nos remerciements vont à nos camarades chercheur pour leur aide et leur soutien moral durant toutes ces années, ainsi qu'à Mlle MESSMER pour la gentillesse avec laquelle elle a assuré la frappe de cette thèse.

TABLE DES MATIERES

PAGES

LISTE DES FIGURES

CHAPITRE 1:
INTRODUCTION

I) INTRODUCTION.

1.1) Définition d'un robot mobile autonome.

Depuis longtemps l'homme a été fasciné par les contes et histoires mettant en scène des objets possédant un comportement et un raisonnement comparables à celui de l'homme :

- le célèbre PINOCCHIO qui rendit la joie de vivre à son «père» GUISEPPE
- le joueur de flûte de VAUCANCON, androïde assis de 1,65m qui exécutait les mêmes opérations qu'un joueur de flûte vivant
- le fameux joueur d'échecs du baron Wolfgang de Kempelen qui battait les meilleurs joueurs .

Depuis le milieu du XXème siècle, ces récits où la machine possède une certaine intelligence deviennent réalité. Parmi ces réalisations il y eut

- HOPKINS'BEAST, véhicule mobile, datant des années 50, circulant de manière uniforme le long d'un couloir dans le seul but de rechercher une prise de courant pour recharger, seul, ses batteries [ASI 70],
- SHAKEY, robot mobile datant du milieu des années 60 et considéré par tous les chercheurs comme un archétype dans le domaine des robots mobiles autonomes [NIL 69].

Contrairement aux robots manipulateurs dont la croissance, due à des raisons industrielles, était assez forte, le développement des robots mobiles a stagné au stade expérimental pour ne reprendre qu'au cours des années 80. Déjà SHAKEY était doté d'un mécanisme de prise de décision intégré que l'on pouvait, compte-tenu des faibles moyens informatiques dont on disposait à l'époque, apparenter à de l'Intelligence Artificielle. Depuis, plusieurs générations de robots se sont succédé:

- les robots de première génération :

  ils exécutent uniquement une suite de mouvements pré-enregistrés. Exemple :

  les chariots filoguidés ayant un déplacement matérialisé par un fil de guidage

enterré dans le sol (guidage inductif) ou peint sur le sol (guidage optique).

- les robots de deuxième génération :

  ils sont capables de percevoir leur environnement, de se diriger de manière simple dans celui-ci et de réagir à une petite modification de l'environnement.

- les robots de troisième génération ou robots mobiles autonomes R.M.A.

La littérature spécialisée donne beaucoup de définitions des robots mobiles autonomes mais la meilleure nous semble être celle de Giralt [GIR 90] quand il les définit comme «... des machines dotées de la capacité de raisonner sur la tâche à accomplir et de mettre en oeuvre pour son éxécution des relations intelligentes entre perception et action. Autrement dit, ce sont des machines que l'on peut programmer au niveau tâche.»

En effet, les R.M.A. sont dotés d'une intelligence (artificielle?) qui les rend capables d'évoluer dans un univers hostile, parsemé de pièges, sans commande directe de l'homme.

Ils sont caractérisés par une capacité d'adaptation face à un univers inconnu ou changeant (flexibilité) et une capacité de décision permettant de coordonner au mieux, perception et action.

1.2) Différents domaines d'utilisation.

Les applications actuelles ou potentielles des R.M.A. sont nombreuses. Parmi elles citons :

- Le Nucléaire :

Afin de diminuer l'exposition du personnel (radiations) et pour des raisons de sécurité (risques dus aux vieillissement des installations), l'exploitation et la maintenance des centrales feront de plus en plus appel aux R.M.A. La conception de ces R.M.A. doit prendre en compte un certain nombre de contraintes telles que :

- géométriques (franchissement d'obstacles : escalier, sas, chicanes,...)
- radiologiques (limitation de la capacité de calcul des calculateurs due aux radiations)
- énergétiques (l'autonomie du R.M.A. devra permettre l'alimentation d'outillage de découpe ou de décontamination).

Une expérimentation actuelle est le projet CENTAURE testé dans le cadre d'une collaboration C.E.A.-E.D.F. [FRA 90].

- **L'armée :**

Les robots mobiles intéressent beaucoup l'armée : projets de la DRET en France [FAR 90], projets de la DARPA aux Etats-Unis dans les domaines terre, air, mer ou multi-environnement (projet MERUS) [ROB 89]. La gamme du robot ou pseudo-robot couvre une large plage s'étendant des systèmes téléopérés, c'est-à-dire pré-programmés et guidés à distance par un opérateur, aux engins autonomes entièrement voués à l'intelligence artificielle. Ces derniers vont prendre une place prépondérante sur les futurs champs de bataille.

- **L'agriculture :**

L'agriculture et notamment l'arboriculture ont beaucoup investi dans les robots mobiles. Les robots de récoltes sont déjà parvenus à un stade avancé. Le meilleur exemple en est Magali, robot cueilleur de pommes, dont les principes généraux de fonctionnement sont adaptables à la récolte d'autres fruits [ROB 85] [IND 90].

- **La sécurité civile :**

Le but des robots mobiles avancés pour la sécurité civile est de prolonger l'action de l'homme dans les sites qui lui sont interdits (inaccessibles ou dangereux) et non de se substituer à lui. Ce RMA devra intervenir dans l'enceinte d'un bâtiment (en cas de pollution chimique ou radio-active) ou à l'extérieur (lutte contre l'incendie). Ces contraintes ont donné naissance au programme AMR réalisé par un consortium d'industriels (FRAMATOME, MATRA,...) et de chercheurs (CEA, CNRS,...) français, italiens et espagnols [DEP 90].

- **La surveillance :**

Conçu pour la surveillance d'installations telles que raffineries [THI 90], il ne dispose d'aucun système ou équipement lui permettant d'agir sur l'environnement, tel que bras articulé ou outil quelconque. Il peut néanmoins recueillir des informations venant de son environnement et en fournir une interprétation et un diagnostic.

- **La mer :**

Les opérations sous-marines de l'exploration pétrolière off-shore sont très coûteuses en temps et en personnel. Il est devenu indispensable de développer des robots assumant des tâches fréquentes et routinières (maintenance, inspection). Ces robots sont susceptibles d'apporter une meilleure productivité, un temps d'intervention plus rapide, une fiabilité accrue et des retombées sur l'ensemble des activités océaniques [AUT 88].

- **Autres domaines :**

Dans beaucoup d'autres domaines la robotique mobile suscite des travaux de recherche :

- exploitation forestière et minière : projet PANORAMA [LEM 90]
- le bâtiment et les travaux publics [PRO 87] [PRO 88] [PRO 89]
- la transitique (ou manutention) [DUC 90]
- l'exploitation de planète : projet VAP-RISP (Véhicule Autonome Planétaire - Robot d'Intervention sur Site Planétaire) développé par quatre organismes CEA - CNRS - INRIA - ONERA [GIR 90]

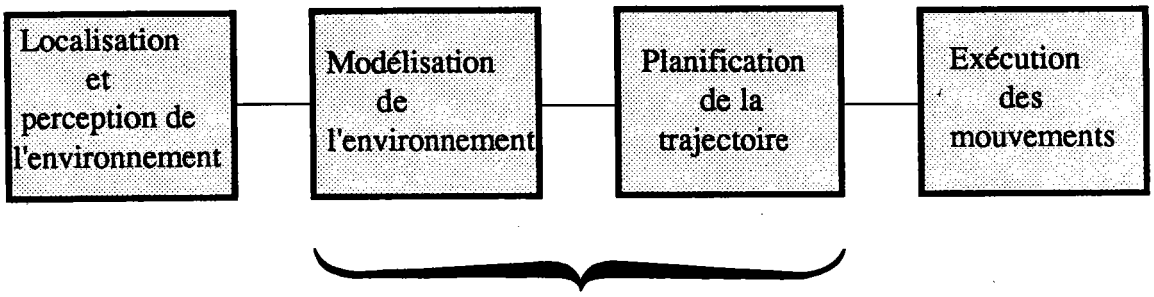
Conclusion :

Nous remarquons que les grands projets, de plus en plus nombreux, sont tous le fruit de collaborations internationales. En effet, les investissements nécessaires à ces projets qui font largement appel à l'Intelligence Artificielle et à l'électronique sont si importants que le groupement de plusieurs intervenants est nécessaire pour réaliser ces projets. Malheureusement, dans beaucoup de domaines où ils seraient très utiles (agriculture, nettoyage, travaux publics) leur rentabilité est loin d'être assurée.

1.3) Présentation de notre travail de recherche.

Ce mémoire présente le travail de recherche effectué au Laboratoire d'Automatique et d'Electronique Industrielles (LAEI) de Metz. Il concerne la modélisation et la planification de trajectoires d'un robot mobile autonome.

La fonction mobilité d'un RMA est composée de plusieurs modules :



Notre travail concerne les modules modélisation et planification.

Ce mémoire comprend deux parties. La première (chap. II) est consacrée à une analyse critique des travaux dans les domaines de la modélisation d'environnement et de la planification de trajectoires pour robots mobiles. Nous y analysons les méthodes globales nécessitant le modèle géométrique complet ainsi que les méthodes locales servant à la navigation «au plus près» des obstacles. La prise en compte de la taille du robot est étudiée pour chacune des modélisations. Nous examinons ensuite les différents algorithmes, exhaustifs et heuristiques, de recherche de trajectoire, et citons les principaux projets dans lesquels ils ont été utilisés.

Dans la deuxième partie (chap. III), nous exposons notre travail de recherche. Ce travail fait partie du projet (V.A.H.M.) du laboratoire concernant la mobilité d'un fauteuil pour handicapés physiques. Nous y proposons une méthode de modélisation d'un environnement par grille adaptative à partir d'un espace déterminé par des objets polyédriques. La méthode consiste à partager l'espace total suivant les sommets parallèlement à un référentiel de base. Les bandes ainsi formées sont découpées selon le critère de l'approche minimale entre le mobile et les obstacles. L'intersection des bandes horizontales et verticales forme des cellules élémentaires auxquelles est associée une variable binéaire selon qu'une cellule fait partie de l'espace libre ou d'un obstacle.

Cette méthode a ensuite été étendue à la modélisation dynamique des objets mobiles et aux environnements $2D^{1/2}$.

La recherche de chemin entre deux points du modèle est réalisée par une association des algorithmes de LEE et A* (chap. 2.3.2) en effectuant des opérations booléennes de masquage. Cette méthode a l'avantage de nécessiter un espace mémoire réduit et de déterminer une trajectoire admissible rapidement.

Dans la conclusion nous analysons les résultats obtenus de manière objective et exposons les recherches futures qui devraient permettre l'implantation du logiciel dans la partie commande du fauteuil VAHM.

En annexe, nous faisons une présentation générale du logiciel MEPT (Modélisation d'Environnement et Planification de Trajectoires) et nous expliquons la fonction de chaque module composant ce logiciel. Ce logiciel de simulation a été conçu dans le but de vérifier la rapidité et l'efficacité des divers algorithmes.

1.4) Le projet V.A.H.M. [CUN 88] [PRU 89 c]

Le projet V.A.H.M. (Véhicule Autonome pour Handicapés Moteur), étudié au L.A.E.I., en collaboration avec des industriels, est un projet de transformation d'un fauteuil roulant en véhicule autonome. Ce véhicule autonome devra permettre à une personne handicapée moteur, en particulier aux personnes atteintes de maladies neuro-musculaires, de

- réaliser des actions que leur état physique leur interdit (manipulation d'objet, par exemple),
- leur éviter une fatigue physique et mentale,
- réaliser des déplacements en toute sécurité à travers des zones sujettes à des perturbations.

C'est pour cette dernière partie que ce travail de recherche a été réalisé. Ce fauteuil sera doté d'un système de télémétrie acoustique et/ou optique lui permettant d'acquérir des informations concernant son environnement d'évolution. Après traitement ces informations serviront à réaliser un modèle géométrique, préambule nécessaire à la planification de la trajectoire à travers les obstacles, vers le but fixé par la personne handicapée. Les différentes pièces d'une maison d'habitation seront l'univers d'évolution du V.A.H.M. et les éléments du mobilier (tables,...) constitueront les obstacles. Une modélisation dynamique est impérative du fait de la mobilité de certains éléments de l'environnement (Ex. : chaise déplacée, présence d'une tierce personne dans la pièce).

Le programme régissant le comportement du fauteuil devra effectuer une modélisation globale de l'environnement puis, grâce aux informations fournies par le système de perception, une modélisation locale sera effectuée.

CHAPITRE II :
MODELISATION ET PLANIFICATION DE TRAJECTOIRE : ETAT DE L'ART

II) MODELISATION ET PLANIFICATION DE TRAJECTOIRE : ETAT DE L'ART.

2.1) Méthodes globales de modélisation.

2.1.1) Modélisation d'un environnement par partitionnement en grille homogène. [THO 83] [THO 84]

2.1.1.1) Principe.

THORPE propose la représentation de l'espace d'évolution d'un robot cylindrique basée sur un graphe de convexité. En partitionnant de manière régulière l'environnement, il détermine des cellules. Un graphe de convexité en est déduit où

- les noeuds représentent les sommets des cellules situés en zone accessible par le robot,
- les arcs représentent le déplacement possible d'un noeud vers ses n plus proches voisins (avec n=4 ou n=8).

2.1.1.2) Taille des cellules.

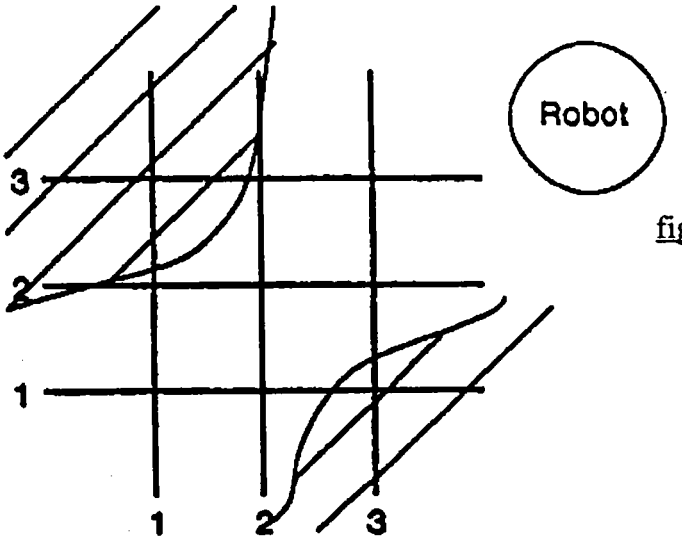


fig. 2.1 : Environnement décomposé en cellules de coté $r\sqrt{2}/2$

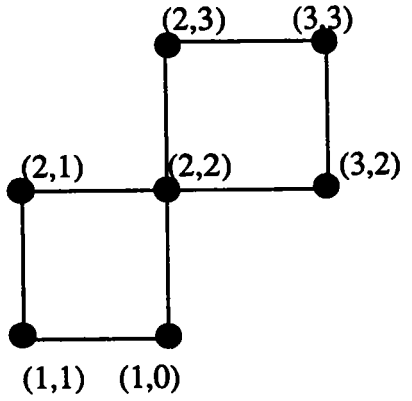


fig. 2.2 : Graphe déduit de la fig. 2.1

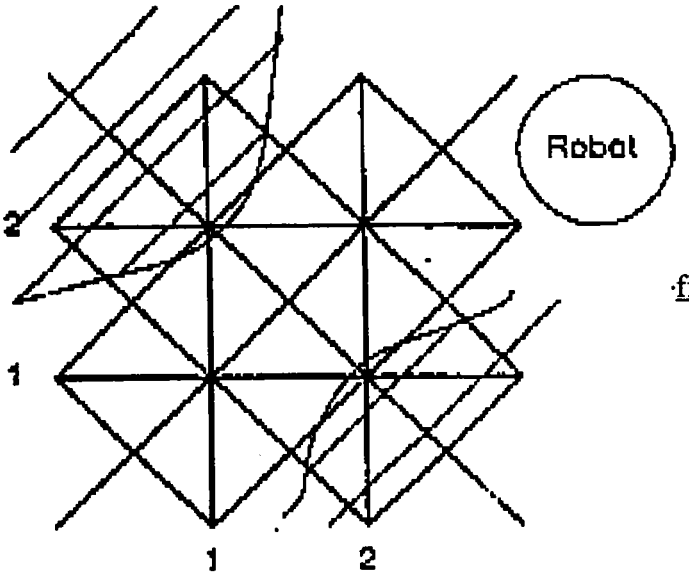


fig. 2.3 : Environnement découpé en cellules de côté r

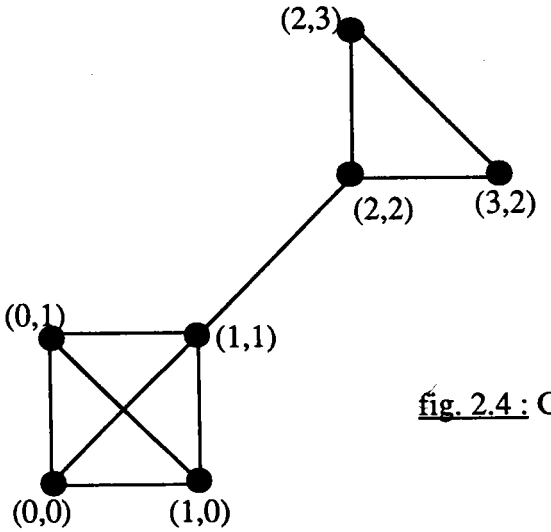


fig. 2.4 : Graphe déduit de la fig 2.3

Thorpe propose deux tailles de cellules :

- cellules de côté $r\sqrt{2}/2$ (avec r = diamètre du robot). Dans ce cas, les noeuds dont la distance relative est égale à $r\sqrt{2}/2$ sont reliés entre eux. Le nombre maximum de liaisons est n=4 (fig. 2.1 et fig. 2.2).
- cellules de côté r. Dans ce cas, les noeuds dont la distance relative est inférieure ou égale à $r\sqrt{2}$ sont reliés entre eux. Le nombre maximum de liaisons est n=8 (fig. 2.3 et fig. 2.4).

Le déplacement est possible sur l'environnement s'il est possible sur le graphe.

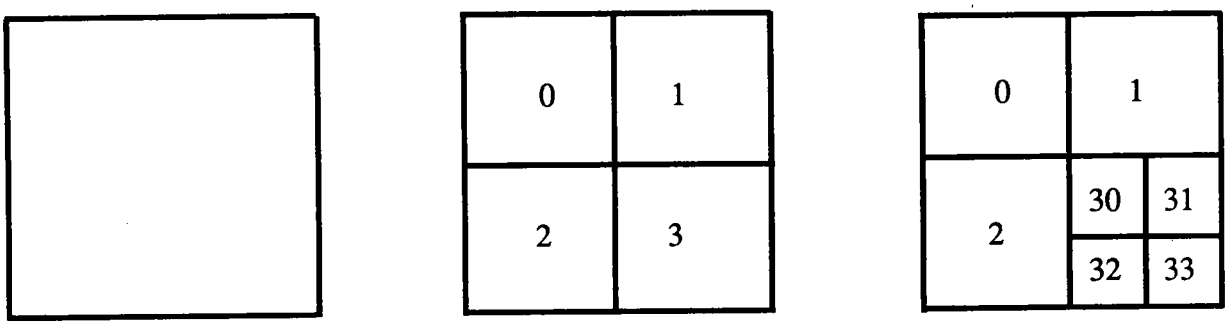
2.1.1.3) Conclusion.

Comme le montrent les figures 2.1 et 2.3, cette modélisation peut être utilisée sur un environnement non polygonal, mais elle nécessite un emplacement mémoire important. L'algorithme A* (chap. 2.3.2) a été utilisé pour déterminer un chemin à travers les obstacles.

2.1.2) Partitionnement par quadtrées. [SHP 85] [KAM 86] [SAM 87]

2.1.2.1) Principe.

Cette méthode consiste à modéliser l'objet en décomposant l'environnement par des divisions successives (fig. 2.5).



(a) (b) (c)

fig. 2.5 : (a) Environnement

(b) Environnement décomposé en 4 quadrants (0,1,2,3) dits de 1 ère génération

(c) Quadrant 3 divisé en 4 sous-quadrants (30,31,32,33) dits de 2 ème génération

Ces divisions successives s'effectuent tant que les trois conditions suivantes ne sont pas satisfaites :

- l'objet à représenter est entièrement recouvert de sous-quadrants,
- la région extérieure (ou complémentaire) à l'objet est entièrement recouverte de sous-quadrants,
- un sous-quadrant appartient soit à l'objet à représenter, soit à la région complémentaire, mais pas aux deux.

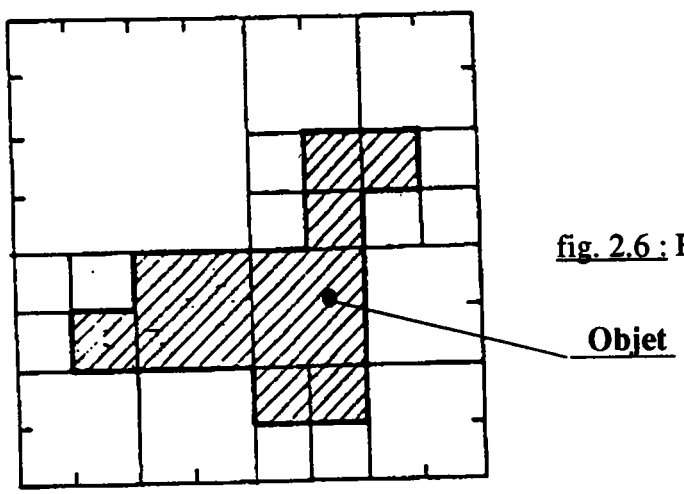


fig. 2.6 : Exemple de décomposition

L'environnement de la figure 2.6 peut être représenté sous forme arborescente à l'aide de noeuds, de feuilles pleines et de feuilles vides (fig. 2.7) où

- les noeuds (○) sont les sous-quadrants appartenant à une région mixte,
- les feuilles pleines (■) sont les sous-quadrants appartenant à l'objet à représenter,
- les feuilles vides (□) sont les sous-quadrants appartenant à la région complémentaire.

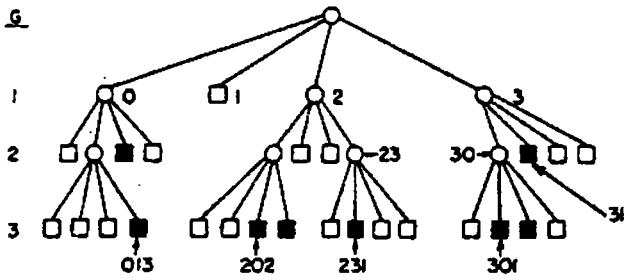


fig. 2.7 : Représentation arborescente de la figure 2.6

2.1.2.2) Mémorisation de cette représentation.

Plusieurs méthodes facilitant la mémorisation de ces informations sont utilisées. Nous citerons simplement celles

- de Breadth First Transversal ou B.T.F. (description par ligne) qui nous donne pour l'exemple représenté par l'arborescence de la figure 2.7

  f = ○,○□○○, □○■□○.....

  f = 2, 2022 , 02102.....

- du Listage des feuilles pleines qui nous donne pour le même exemple que précédemment :

  1ère génération : /

  2ème génération : 02, 31

  3ème génération : 013, 202, 203, 231, 301, 302.

2.1.2.3) Conclusion.

Cette technique, très utilisée, permet un gain mémoire important. Le caractère hiérarchique de cette représentation est essentiellement utilisé pour accélérer la recherche d'un chemin.

Cette technique peut être adaptée à la représentation d'environnement 3D (octree), chaque noeud possédant huit descendants.

2.1.3) Modélisation par expression canonique. [SHP 84] [SHP 85] [PRU 89 b]

2.1.3.1) Principe.

Cette méthode, présentée comme une alternative aux quadtrees, est basée sur la décomposition de l'environnement en 2n x 2m rectangles. Ces rectangles sont obtenus par partitionnement de l'environnement suivant l'axe des x et des y. Chaque bande horizontale et verticale est numérotée avec un code binaire (binaire réfléchi pour SHPITALNI, binaire naturel pour PRUSKI) comportant plusieurs variables (X1 à Xn pour l'axe des x, Y1 à Ym pour l'axe des y). Un objet à modéliser est représenté par la somme de plusieurs rectangles, chaque rectangle étant repéré par les composantes Xi et Yi.

L'objet à modéliser de la figure 2.8 peut être représenté par la fonction canonique

$f = x2\overline{y1}y2 + x1x2\overline{y1}y3 + \overline{x1}x3\overline{y1}y2y3 + x1x2x3y2 + x1x3y1y2y3$

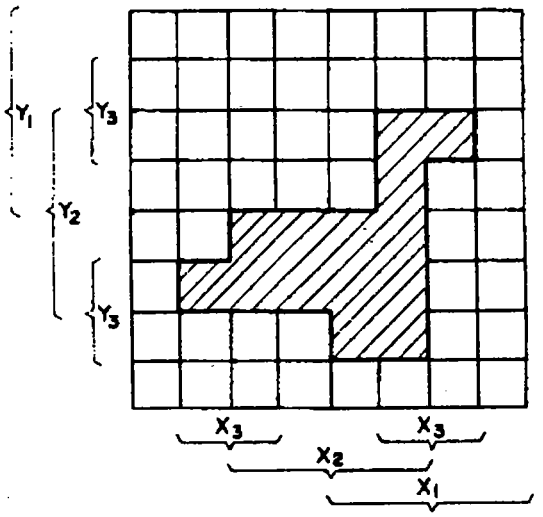


fig. 2.8 : Codage d'une image utilisant le code GRAY

2.1.3.2) Mémorisation de cette représentation.

Chaque terme de la fonction canonique est défini par (n + m) digits associés à (n + m) variables (x1, x2,....xn, y1, y2,...yn). Ces digits seront égaux à

- «0» si la variable est exprimée,
- «1» si la variable est complémentée,
- «2» si la variable est ignorée.

Par exemple, la fonction f vue précédemment sera

f = 212012, 112021,.....

2.1.3.3) Conclusion.

Cette méthode possède de nombreux avantages :

- possibilité de manipuler des rectangles (au lieu de carrés pour les quadtrees),
- le code affecté à chaque rectangle nous renseigne sur sa taille et sa position,
- le codage des images symétriques est facilité.

Cette méthode est aussi applicable à la représentation d'un environnement 3D. On obtient alors un ensemble de 2n x 2m x 2p parallélépipèdes (figure 2.9).

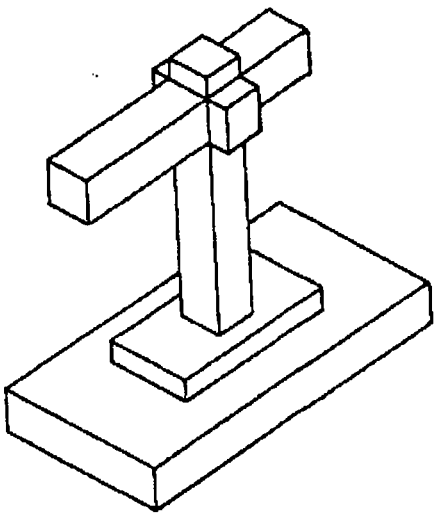


fig. 2.9 : Exemple d'environnement 3D

2.1.4) Modélisation par décomposition en polygones convexes.

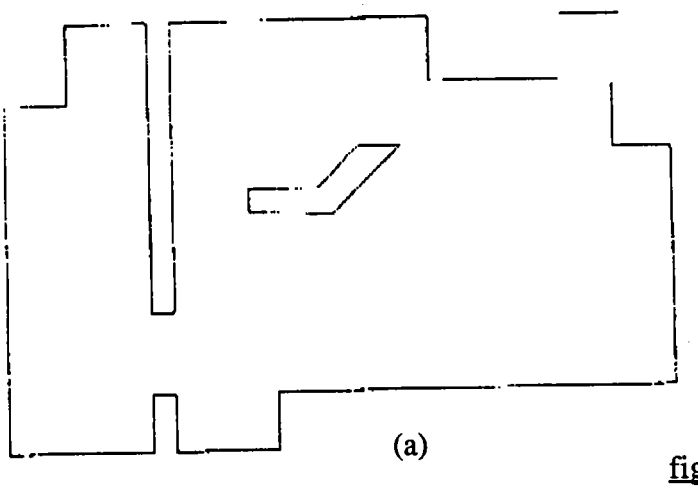
2.1.4.1) Propriétés des polygones convexes (ou surfaces convexes).

Les polygones convexes sont des régions 2D limitées par des segments de droite ayant entre eux des angles inférieurs à 180° (mesurés de l'intérieur du polygone). Cette décomposition repose sur le fait que toute droite joignant deux points quelconques du polygone reste comprise dans ce dernier. Si de tels polygones, représentant l'ensemble de la surface libre, peuvent être trouvés, alors un robot mobile peut se déplacer entre deux points quelconques de cette surface sans heurter un obstacle. Par extension, cette méthode peut aussi s'appliquer aux environnements 3D. Dans ce cas, on détermine des volumes convexes possédant les mêmes propriétés.

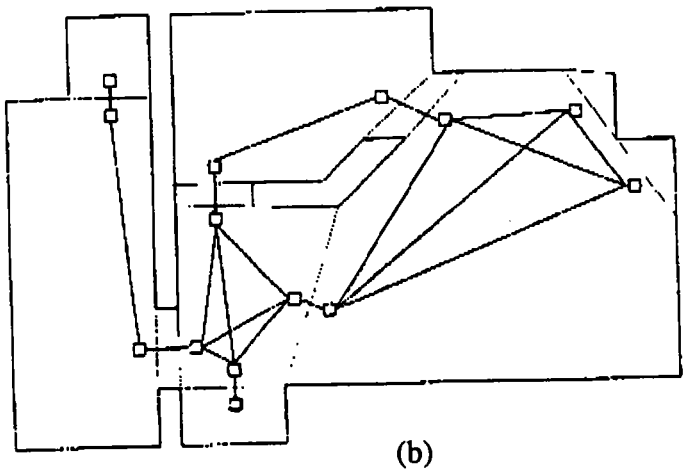
2.1.4.2) Modélisation proposée par CROWLEY. [CRO 84a] [CRO 84b] [CRO 84c] [CRO 85]

La méthode proposée par CROWLEY, dans le cadre de recherches effectuées à l'Institut de Robotique de l'Université de Carnégie Mellon, convient pour la modélisation d'environnements

connus. Des surfaces convexes, les plus grandes possibles, ne se superposant pas et représentant des régions accessibles, sont déduites de l'environnement. A ces surfaces, diminuées d'une valeur équivalente au diamètre du robot, on affecte une zone d'accès sur leurs frontières communes avec d'autres surfaces. On déduit ensuite un graphe de connexité où les noeuds représentent les zones d'accès et où les arcs représentent le déplacement possible entre zones d'accès appartenant à la même surface convexe et entre zones d'accès de surfaces convexes adjacentes.



(a)



(b)

fig. 2.10 : Décomposition

(a) Environnement à décomposer

(b) Décomposition effectuée

Conclusion.

La disposition des zones d'accès permet de changer de surfaces convexes quasiment à angle droit, ce qui évite au robot de se heurter aux obstacles.

2.1.4.3) Modélisation proposée par CHATILA. [CHA 80] [CHA 81] [CHA 82]

Chatila nous propose une méthode de modélisation dynamique d'un environnement 2D que le robot découvre au fur et à mesure de sa progression. Cette modélisation est basée sur des sommets

- de type S : sommets réels, délimitant des côtés entiers de polygone,
- de type P : sommets apparents, projections de sommets de type S sur les parois d'obstacles,

ainsi que des cellules de type

- CS : dont tous les sommets sont de type S,
- CP : dont les sommets sont du type S et du type P,
- C* : représentant des parties inconnues ou non structurées de l'univers.

Au fur et à mesure de l'avancée du robot dans l'environnement, les cellules C* et CP se transforment en cellules CS. Le graphe de connexité (fig. 2.13), déduit du modèle (fig. 2.12), est ré-actualisé en permanence. Les noeuds du graphe représentent le centre des cellules CS, CP et C*, et les arcs représentent le déplacement possible entre cellules adjacentes.

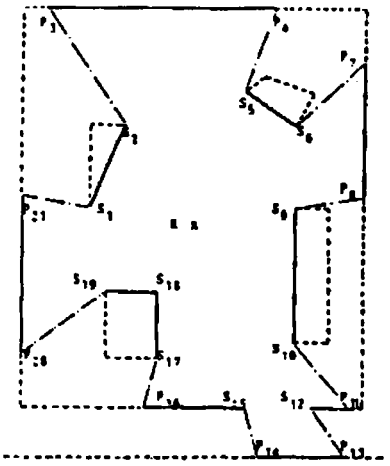


fig. 2.11 : Vue initiale

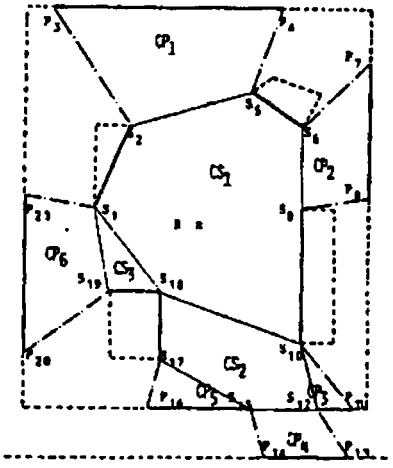


fig. 2.12 : Décomposition effectuée

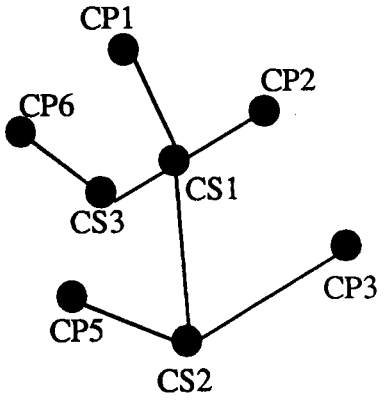


fig. 2.13 : Graphe de connexité

Conclusion.

Cette technique est particulièrement intéressante dans le cas de l'apprentissage. Des cellules, pouvant s'agréger, sont déterminées au fur et à mesure de l'avancée du robot. La prise en compte des dimensions du robot s'effectue par grossissement uniforme des obstacles.

2.1.4.4) Modélisation proposée par LAUMOND. [LAU 83] [LAU 84] [CHA 85]

LAUMOND a repris la décomposition de CHATILA et l'a étendue à la représentation hiérarchique de régions. Exemple : diverses pièces d'une maison (figure 2.14). Le graphe des cellules défini par CHATILA (figure 2.15) a été décomposé en

- sous-graphes représentant la modélisation dans chaque région,
- un graphe de niveau supérieur reliant les différents sous-graphes (fig. 2.16).

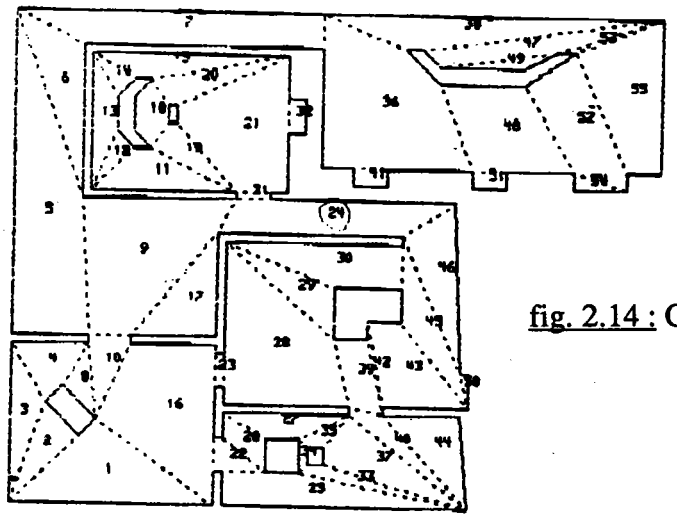


fig. 2.14 : Cellularisation

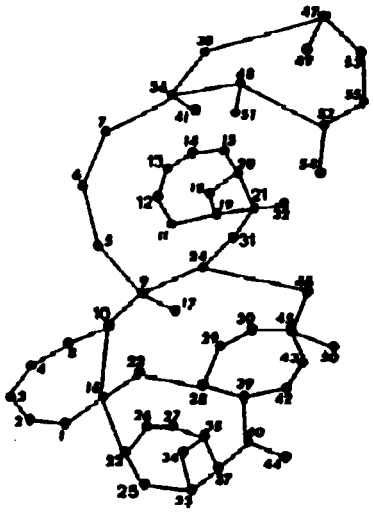


fig. 2.15 : Graphe des cellules

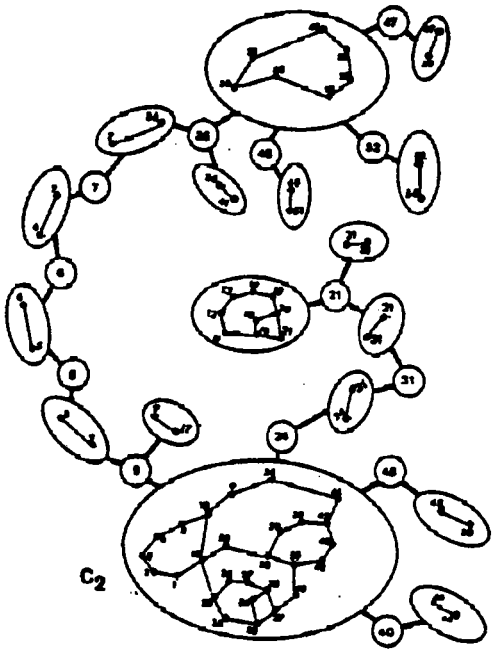


fig. 2.16 : Arbre de décomposition

Conclusion.

L'utilisation de graphes planétaires a permis une meilleure structuration de l'espace d'évolution du robot. La planification de la trajectoire est décomposée en

- une recherche de chemin dans un graphe nous donnant les pièces à traverser (navigation globale),
- une recherche de trajectoire dans les différentes pièces à traverser (navigation locale).

2.1.4.5) Modélisation proposée par SINGH. [SIN 85] [SIN 87]

SINGH propose une méthode de décomposition d'un environnement statique en un maximum d'aires convexes. Aucune aire convexe n'est oubliée.

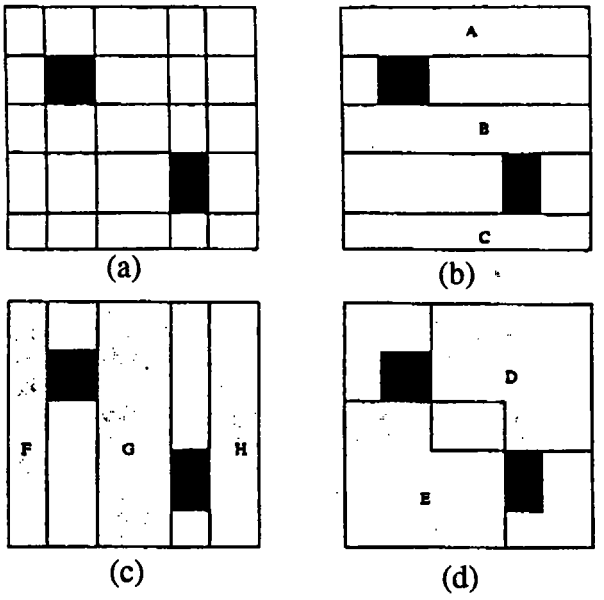


fig. 2.17 : Décomposition en aires convexes

Après une première décomposition (fig. 2.17a), Singh identifie les aires convexes à l'aide de 2 mots de (2n + 1) bits au maximum, avec n nombre de cases représentant un obstacle.

Exemple 1 : la deuxième cellule de la troisième ligne est repérée par

0 1 0 0 0 0 0 1 0 0

Exemple 2 : les mots 1 1 0 0 0 0 0 1 1 0 représentent un ensemble formé des cases suivantes :

1 0 0 0 0 0 0 1 0 0

0 1 0 0 0 0 0 1 0 0

1 0 0 0 0 0 0 0 1 0

0 1 0 0 0 0 0 0 1 0

Définition : Une aire convexe primaire est une aire libre non entièrement incorporée dans une autre aire convexe primaire.

Singh utilise ensuite un algorithme, dérivé des techniques utilisées par QUINE [QUI 52] et Mc CLUSKEY [CLU 56] basé sur des fusions de cellules, visant à obtenir la totalité des aires convexes primaires d'un environnement (fig. 2.17b, c, d).

Enfin un graphe de connexité est déduit. Chaque noeud représente une aire convexe primaire, les noeuds sont joints par des arcs si les surfaces qu'ils représentent ont une partie

commune. La figure 2.18 nous donne le graphe obtenu à partir de l'environnement représenté figure 2.17.

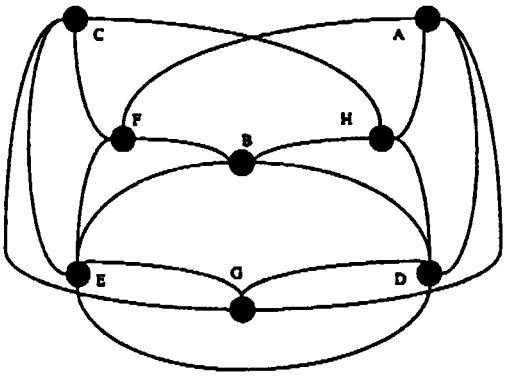


fig. 2.18 : Graphe de connexité déduit de la figure 2.17

Dans un tel graphe, la recherche de trajectoire peut être effectuée à l'aide de nombreux algorithmes.

Conclusion.

Par opposition à THORPE et à CHATILA, cette méthode utilise au maximum les possibilités de convexité d'un environnement dans le but d'optimiser au mieux la trajectoire.

2.1.5) Modélisation par rubans (ou cônes) généralisés. [BRO 83] [BRO 85]

BROOKS propose la modélisation de l'espace d'évolution polyédrique d'un robot à l'aide de «rubans généralisés». Ces «rubans généralisés», insérés entre les obstacles et représentant un passage possible pour le robot, peuvent se superposer (fig. 2.19).

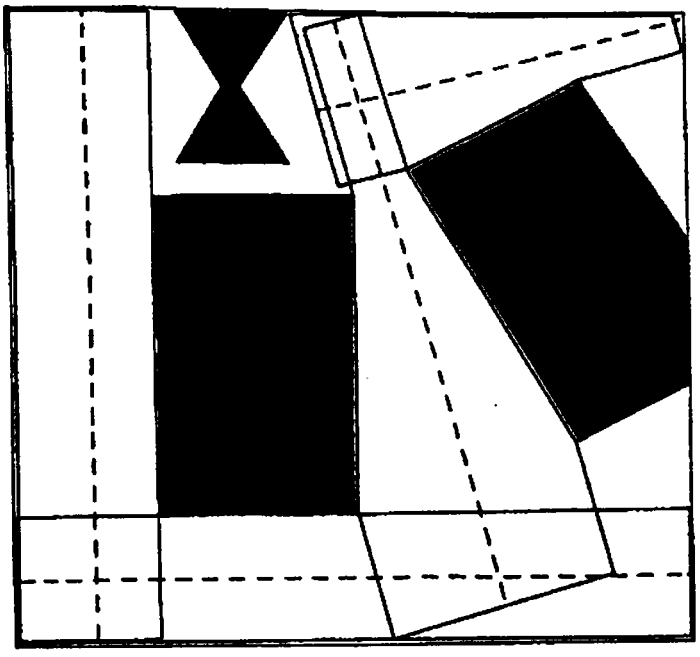


fig. 2.19 : Modélisation par "rubans généralisés"

Un ruban généralisé est défini par un axe et une fonction déterminant la largeur du ruban le long de l'axe. Ses dimensions sont définies par les caractéristiques du robot (taille, orientation) et

celles de l'environnement (fig. 2.20). L'ensemble des rubans ainsi obtenus nous donne la zone navigable à travers laquelle le robot peut se déplacer. Le déplacement du robot est décomposé en translations pures le long des axes et en rotations pures à l'intersection des axes de deux rubans.

On détermine ensuite un graphe de connexité où les noeuds représentent l'intersection des axes des rubans et les arcs le déplacement possible le long de l'axe d'un ruban. La recherche de la trajectoire s'effectue à l'aide de l'algorithme A* [chap. 2.3.2]

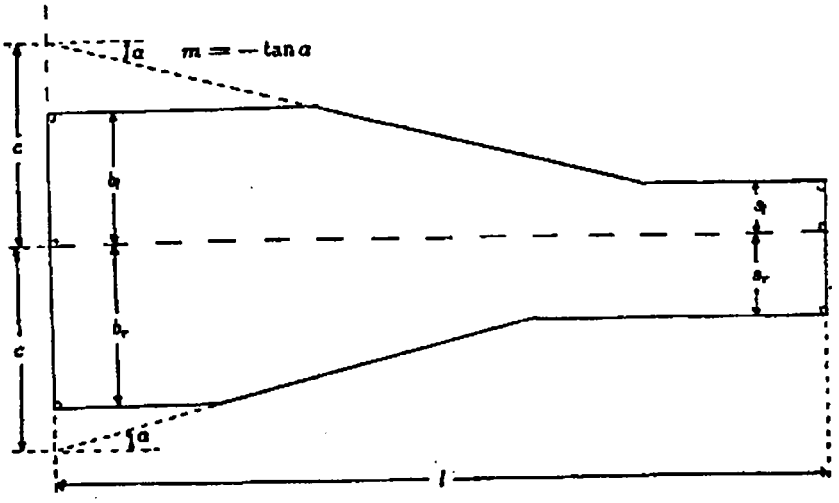


fig. 2.20 : Ruban généralisé définit par ses paramètres.

Conclusion.

Les trajectoires ainsi déterminées sont équidistantes des obstacles. L'inconvénient d'une telle approche est que la discrétisation doit être assez fine vis-à-vis de la géométrie des obstacles, ce qui rend cette représentation très coûteuse.

Cette technique a été étendue à 3 dimensions en «empilant» des plans.

2.1.6) Modélisation par sommets. [LOZ 79] [LOZ 81] [LOZ 83]

LOZANO-PEREZ propose une méthode de modélisation d'un univers connu, composée d'obstacles polygonaux basée sur une extension de ces derniers.

La méthode consiste à transformer les obstacles polygonaux en leur adjoignant une zone interdite dont la taille est fonction de celle du robot. On détermine ainsi les parties libres de l'espace que le robot peut occuper sans heurter d'obstacles.

2.1.6.1) Cas où le robot est assimilé à un point.

Dans ce cas un graphe VG (N, L) est défini avec N représentant l'ensemble des sommets réels et L l'ensemble des arcs joignant les noeuds dont la connection ne passe pas au travers d'un obstacle (fig. 2.21). Le graphe VG est aussi appelé graphe de visibilité.

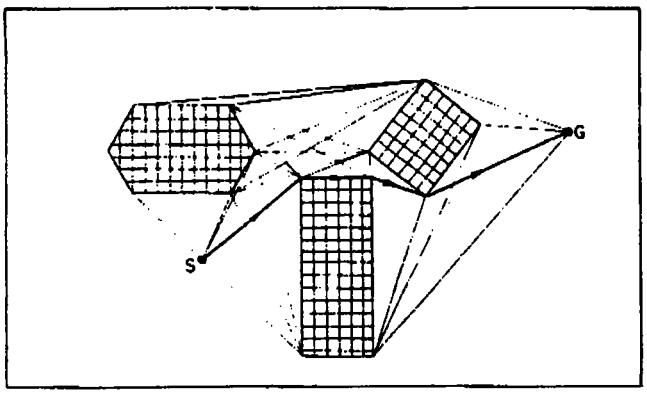


fig. 2.21 : Graphe de visibilité

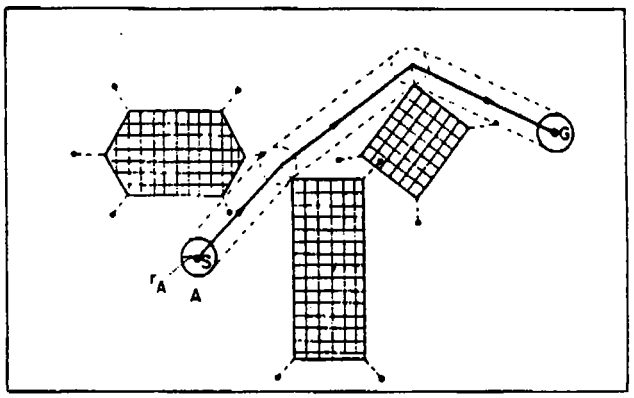


fig. 2.22 : Grossissement uniforme de l'obstacle

2.1.6.2) Cas où le robot n'est pas assimilable à un point.

Deux cas peuvent se présenter :

- le robot est cylindrique de rayon r : la taille des obstacles est uniformément augmentée d'une valeur r (fig. 2.22).
- le robot possède une forme polygonale quelconque : l'opération de «grossissement» dont est sujet l'obstacle dépend de l'orientation du robot. Elle détermine une zone interdite pour un point de référence S du robot et s'effectue grâce aux sommes de MINKOWSKI [KED 86] [POL 88].

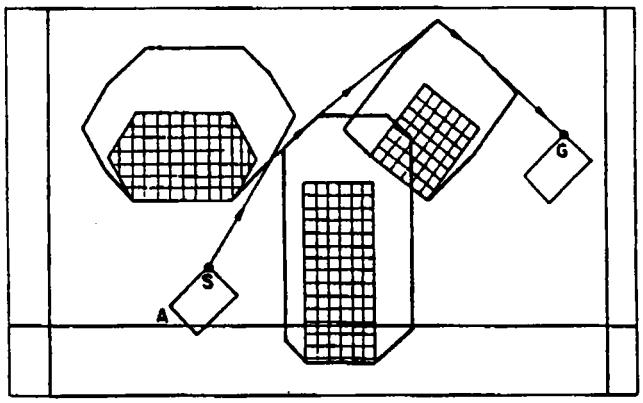


fig. 2.23

fig. 2.23 à 2.25 : Grossissement par somme de MINKOWSKI

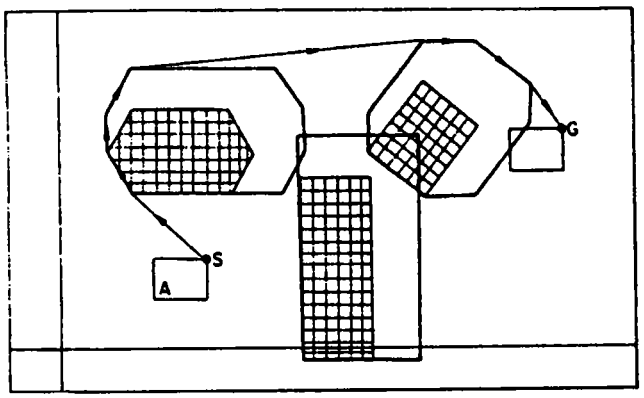


fig. 2.24

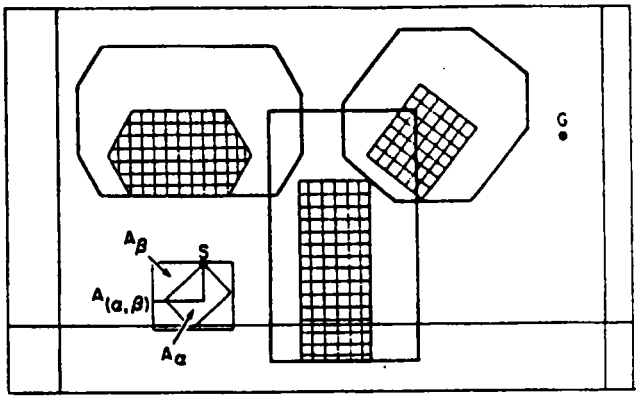


fig. 2.25

Création du graphe :

Dans le cas où le robot n'est pas assimilable à un point matériel, les noeuds du graphe VG (N, L) représentent les sommets fictifs et les arcs le déplacement possible entre noeuds dont la connexion, en ligne droite, ne passe pas sur une zone interdite.

Recherche de la trajectoire :

Elle revient à chercher le plus court chemin de la source au but parmi les obstacles déformés. Elle s'effectue dans le graphe VG (N, L) grâce à l'algorithme A* [chap. 2.3.2].

Conclusion.

Cette technique a notamment été utilisée avec succès dans le projet SHAKEY [NIL 69] pour un des premiers robots mobiles. Cette technique implique que l'orientation du robot ne change pas durant le déplacement. On constate (fig. 2.23 à 2.25) que la trajectoire est fonction de l'orientation du robot.

2.1.7) Diagrammes de VORONOI. [PRE 85]

2.1.7.1.) Présentation.

Définition : Soit un ensemble M = {m1,...mn} de n points dans R^d. Le diagramme de Voronoï est un pavage de R^d constitué des cellules polygonales convexes C1,....Cn définies par

$$Ci = \{ x \in R^d | \forall j=1,..... n, j \neq i, d(x,Ci) <= d(x,Cj)\}$$

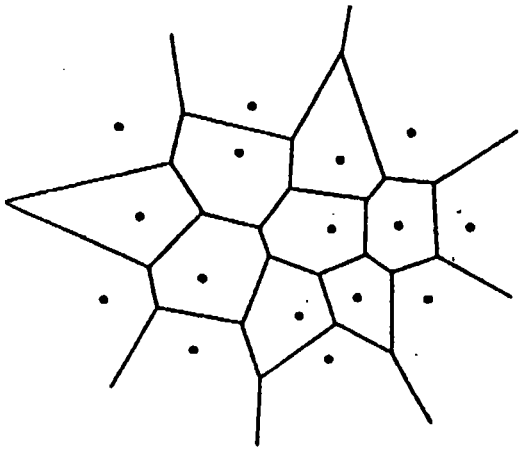


fig. 2.26 : Diagramme de VORONOI de dimension 2

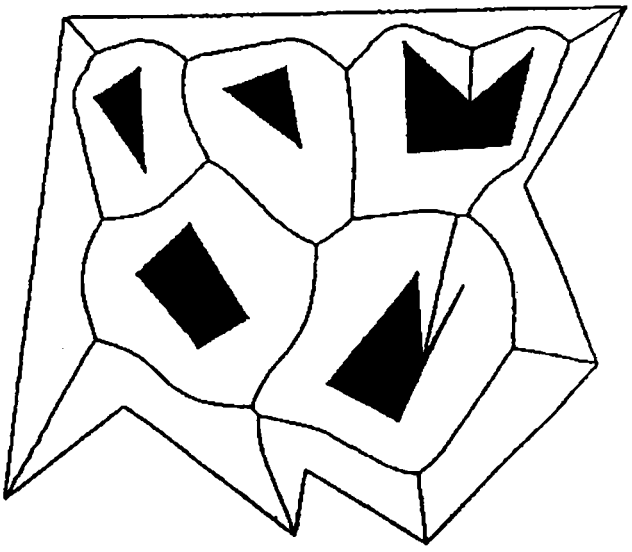


fig. 2.27 : Squelette d'une région polygonale

En dimension 2, les sommets et arêtes des cellules constituent un graphe ayant au plus (2n-5) sommets et (3n-6) arêtes(fig.2.26). Divers algorithmes optimaux [OHY 84] [ASA 85] [FOR 87] permettent d'obtenir le diagramme.

En dimension d, le graphe peut avoir $0(n^{\lceil d/2 \rceil})$sommets (avec $\lceil d/2 \rceil$ partie entière de d/2 +1). Divers algorithmes optimaux [WAT 81] [EDE 86] permettent d'obtenir le diagramme.

2.1.7.2.) Généralisation.

On généralise le diagramme de Voronoï à la construction du squelette d'un polygone (fig. 2.27).

Définition : on appelle squelette l'ensemble des points situés sur la frontière commune des cellules, c'est-à-dire à égale distance d'au moins deux objets.

Les arêtes du diagramme peuvent être, dans ce cas, des paraboles et constituent les points où il est le plus facile, pour un robot circulaire, de passer.

Conclusion.

Les diagrammes de VORONOI sont à la base de plusieurs travaux. Notamment

- pour coordonner les mouvements de 2 ou 3 disques dans un plan encombré de polygones [YAP 84],
- pour coordonner les mouvements de deux manipulateurs à deux degrés de liberté décrits par de simples segments dans le plan, en présence d'obstacles représentés par des polygones [FOR 86].

La triangulation de DELAUNAY, utilisée pour la modélisation [BOI 84] est dérivée des diagrammes de VORONOI.

2.1.8) Autres travaux.

Parmi les autres travaux nous pouvons citer

- **KUAN** [KUA 84] [KUA 85]. Il perfectionne la méthode de BROOKS en utilisant une représentation mixte de l'espace d'évolution. Cette technique utilise des cônes pour représenter les passages étroits entre obstacles et des polygones convexes pour les espaces plus larges. Ces différentes surfaces ne se superposent pas. L'algorithme proposé par KUAN décompose les obstacles concaves en obstacles convexes et détermine un graphe de connexion. Les noeuds représentant les surfaces convexes sont reliés entre eux par des arcs symbolisant les cônes. La recherche de la trajectoire s'effectue grâce à l'algorithme A* (chap. 2.3.2).

- **RUEB** [RUE 87]. La modélisation par hypergraphe est une autre méthode de représentation de l'espace en polygones convexes. La partition est réalisée en prolongeant les arêtes des obstacles. Trois types de régions sont déduites : les obstacles, les régions libres et les zones de superposition des régions libres. Un graphe représente l'espace libre dont les noeuds et les arcs orientés représentent respectivement les limites des obstacles et les adjacences entre les limites. Un partitionnement du graphe est réalisé afin de représenter l'ensemble des arêtes des obstacles qui sont limite d'une région libre. Le graphe et son partitionnement constituent un hypergraphe. La déduction du chemin s'effectue par l'analyse des zones de superposition des régions libres.

2.1.9) Conclusion.

La modèlisation par grille homogène exige des temps de calcul assez longs si l'on demande une représentation précise de l'environnement, donc un grand nombre de cellules. De plus, elle engendre des graphes de connexité importants, d'où la recherche de trajectoire par méthodes heuristiques. On lui préfère, en général, la modélisation polygonale ou par quadtree, plus récentes et nécessitant un espace mémoire plus réduit.

2.2) Méthodes locales de modélisation.

Par opposition aux méthodes globales, les méthodes locales n'utilisent pas le modèle complet de l'environnement. Elles ne prennent en compte que l'environnement immédiat du mobile, c'est-à-dire les obstacles visibles. Deux méthodes sont principalement utilisées.

2.2.1) Méthode des potentiels.

L'utilisation de champs potentiels a été développée principalement et indépendamment par KHATIB et KROGH.

2.2.1.1) KHATIB [KHA 78] [KHA 86].

Il considère que le modèle se trouve dans un champ de potentiel fictif

$$U(x) = Uxb(x) + \sum_{k=1}^{m} Uk(x)$$

avec Uxb(x) : potentiel attractif du but,

Uk(x) : potentiel répulsif de l'obstacle k.

Ce champ de potentiel fictif engendre une force fictive

$$F(x) = Fxb(x) + \sum_{k=1}^{m} Fk(x)$$

avec $Fxb(x) = -$ grad $Uxb(x)$: force attractive exercée sur le point courant x par le point but xb

$Fk(x) =$ - grad $Uk(x)$: force répulsive exercée sur le point courant x par les arêtes de l'obstacle k.

La force Fxb(x) peut être soit

- proportionnelle à la distance point courant/point but : $Fxb(x) = - kp(x-xb)$ où kp représente un gain en position. Pour avoir une stabilité du système, on rajoute un terme d'amortissement, ce qui donne

$$Fxb(x) = - kp(x-xb) - kv \dot{x}$$

- de module constant et dirigé vers le but. Dans ce cas le potentiel générant cette force est de la forme

$$Uxb(x) = kp\|x - xb\| - kp/2 \quad si \ \|x - xb\| > 1$$

$$Uxb(x) = \frac{1}{2} kp\|x - xb\|^2 \quad si \ \|x - xb\| <= 1$$

La force Fk(x) a un champ d'activité limité à une zone d'influence σ . Elle est de la forme

$$Fk(x) = \eta.(1/d_k - 1/\sigma).\frac{1}{d_k^2}.grad\ d_k \qquad si\ d_k <= \sigma$$

$$Fk(x) = 0 \qquad si\ d_k > \sigma$$

avec d_k : distance mini. entre le point courant et l'obstacle k

η : gain constant (ou constante).

2.2.1.2) KROGH [KRO 84] [KRO 85a] [KRO 85b] [KRO 86].

Il procède d'une manière similaire en soumettant le mobile à un ensemble de forces

$$F(x) = Fxb(x) + \sum_{k=1}^{k=m} Fk(x)$$

avec Fxb(x) : force attractive exercée par le but xb.

Fk(x) : force répulsive (résultant d'un potentiel répulsif) affectée aux obstacles.

Force attractive : l'intensité de cette force est fonction de T (vg, dg) : temps minimum mis par le mobile pour parcourir la distance dg le séparant du but. Le parcours s'effectue selon le diagramme suivant:

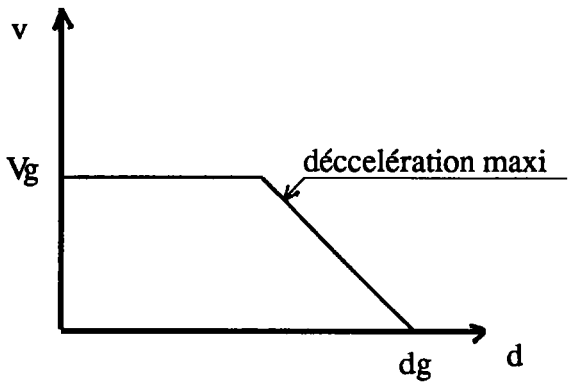


fig. 2.28 : Diagramme (espace,vitesse) du mobile

$$T(Vg,dg) = ((2Vg^2 + 4\alpha dg)^{1/2} - Vg)/\alpha \quad \text{si condition I}$$

$$T(Vg,dg) = ((2Vg^2 - 4\alpha dg)^{1/2} + Vg)/\alpha \quad \text{si conditionII}$$

avec condition I : ($Vg < 0$ et $2\alpha dg >= -Vg^2$) ou ($Vg >= 0$ et $2\alpha dg > -Vg^2$)

condition II : ($Vg <= 0$ et $2\alpha dg < -Vg^2$) ou ($Vg > 0$ et $2\alpha dg <= -Vg^2$)

La force d'attraction est définie comme la somme géométrique de 2 composantes orthogonales.

$$F(Vg,dg) = \omega(Vg,dg)ng + \omega(Vo,0)og$$

$$\text{avec } w(y,x) = \begin{cases} T(y,x) \text{ si condition I} \\ -T(y,x) \text{ si condition II} \end{cases}$$

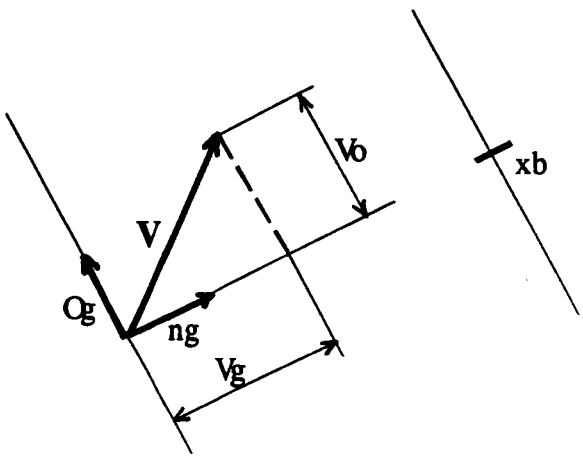


fig. 2.29 : Paramètres utilisés pour définir la force d'attraction.

Force répulsive : l'intensité de cette force est fonction de la «réserve» de temps pour l'évitement de l'obstacle.

Soit ti le temps minimum nécessaire pour arrêter un mobile animé d'une vitesse vi à l'instant to et animé d'une déccélération maximum αi .

$$ti = vi(to)/\alpha i$$

Soit Ti le temps mis par le mobile animé à l'instant to d'une vitesse vi pour s'arrêter sur la distance di qui le sépare de l'obstacle (la déccélération n'étant pas maximale).

$$Ti = 2di/vi(t0)$$

De la fonction potentielle, égale à l'inverse de la réserve de temps TI-ti, définie par

$Uk(x) = 0$ si $vi <= 0$

$Uk(x) = \alpha vi/(2di\alpha - vi^2)s$ si $vi > 0$

on déduit les forces fictives répulsives dirigées de l'obstacle vers le point courant x

$Fk(x) = -\ grad\ Uk(x) = 0$ si $vi <= 0$

$Fk(x) = 2(Uk(x))^2/vi$ si $vi > 0$

2.2.1.3) Autres travaux.

Brièvement nous pouvons citer

- **TAKEGAKI** et **ARIMOTO** [TAK 81]. Ils décrivent les effets des champs potentiels grâce aux formules de HAMILTON et montrent que leur utilisation dans des zones où il existe à la fois des forces d'attraction et de répulsion, est globalement stable.

- **NEWMAN** et **HOGAN** [NEW 87] se basent sur l'énergie cinétique développée par l'objet en mouvement. Ils en déduisent des règles pour la création de champs potentiels répulsifs affectés aux obstacles.

2.2.1.4) Conclusion.

La méthode des potentiels est d'une utilisation simple pour calculer la trajectoire des robots mobiles. En pratique, il est délicat d'ajuster les coefficients utilisés pour pondérer le terme attractif et les termes répulsifs du potentiel : ainsi on maitrise mal la distance critique au-delà de laquelle il est interdit au robot de s'approcher d'un obstacle.

2.2.2) Méthode des contraintes. [TOU 86a] [TOU 86b] [FAV 87] [TOU 88]

L'essentiel de cette méthode consiste à traduire les interactions entre solides susceptibles d'entrer en collision en des contraintes sur les déplacements (vitesse, orientation) exprimées dans l'espace de configuration du système.

La distance entre les solides ne doit pas décroître trop rapidement lorsque celle-ci est inférieure à la somme des distances d'influence de chacun des solides.

La collision entre deux solides convexes sera évitée si on respecte l'inégalité suivante

$$\dot{d} >= -\delta \frac{d - (\varepsilon 1 + \varepsilon 2)}{\sigma 1 + \sigma 2 - (\varepsilon 1 + \varepsilon 2)}$$

avec d : distance euclidienne entre les deux solides.

$\sigma 1$, $\sigma 2$: distance d'influence des deux solides.

$\varepsilon 1$,$\varepsilon 2$: distance de sécurité des deux solides.

δ : coefficient permettant d'ajuster la vitesse de convergence vers la limite $d=\varepsilon 1 + \varepsilon 2$.

Conclusion.

A l'approche d'un obstacle, le champ potentiel repoussera le robot, tandis qu'un robot commandé par la méthode des contraintes reste passif. La méthode des contraintes permet d'éviter les phénomènes de rebond sur les obstacles successifs observés dans un champ potentiel.

2.2.3) Conclusion.

Ces méthodes visent à éviter une collision en créant des espaces de protection autour des obstacles. Elles sont d'une utilisation simple, mais ne nous offrent pas la garantie d'atteindre le but si certains obstacles de formes concaves existent. Elles sont le plus souvent couplées à des méthodes globales.

2.3) Recherche de trajectoire.

Celle-ci se fait en deux temps :

- symbolisation de l'univers dans lequel se déplace le robot mobile par un graphe,

- utilisation d'un algorithme de recherche du chemin.

2.3.1) Eléments de la théorie des graphes. [AZO 76] [GON 79]

2.3.1.1) Terminologie.

Un graphe G= [X, U] est défini par

- X = {1, 2, 3, N} ensemble des N noeuds constituant le graphe,

- U = {(1, 2), (1, 3),.....(N-1, N)} ensemble des M arcs reliant les noeuds.

Ces arcs peuvent être orientés (fig. 2.30) ou non orientés (fig. 2.31).

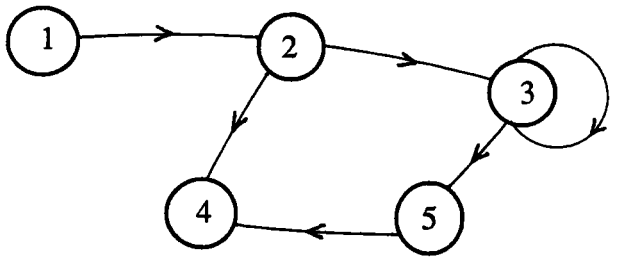


fig. 2.30 : Exemple de graphe orienté

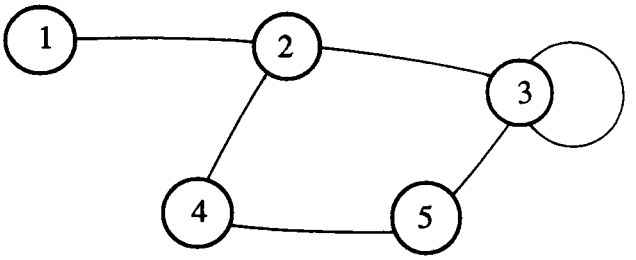


fig. 2.31 :Exemple de graphe non orienté

On désigne par lij la longueur de l'arc reliant le noeud i au noeud j. Dans le cas d'un graphe dense, nous avons au maximum $M = N^2$

On note : Γ_i l'ensemble des successeurs du noeud i (ex. fig.2.30:Γ_2={3,4})

Γ_i^{-1} l'ensemble des prédécesseurs du noeud i (ex. fig.2.30:Γ_2^{-1}={1})

On appelle chemin une succession d'arcs adjacents permettant de passer d'un noeud à un autre.

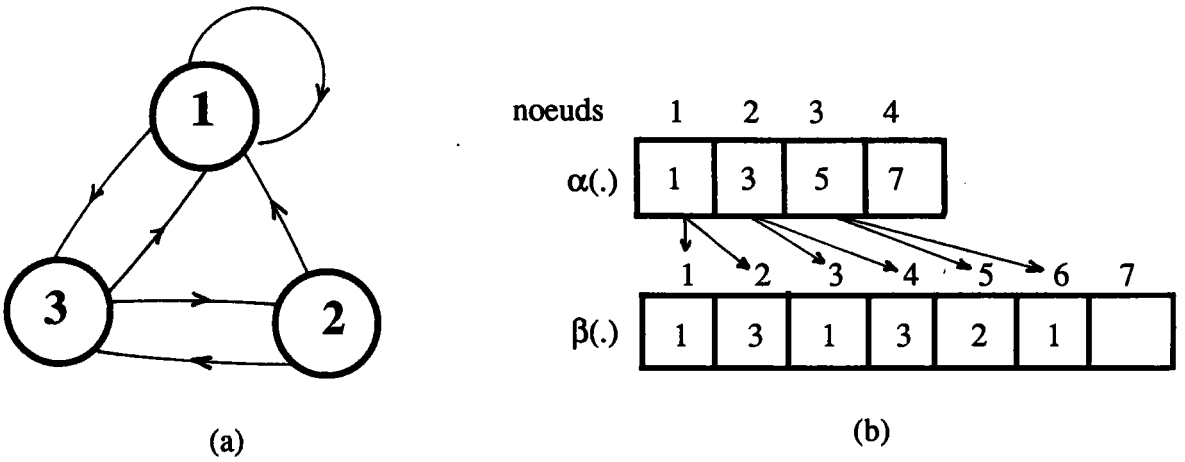
2.3.1.2) Représentation d'un graphe.

Plusieurs représentations peuvent être utilisées pour représenter un graphe G. Chaque représentation n'est pas équivalente du point de vue rapidité des algorithmes de recherche.

a) Représentation à partir de la matrice d'adjacence.

La matrice d'adjacence, forme condensée, utilise deux tableaux $\alpha(.)$ de dimension N+1 et $\beta(.)$ de dimension M (arcs orientés) ou 2M (arcs non orientés).

Pour chaque noeud i, la liste des successeurs de i est contenue dans le tableau $\beta(.)$ à partir de la case numéro $\alpha(i)$ .On peut en conclure que l'ensemble des informations relatives au noeud i est contenu entre les cases $\alpha(i)$ et $\alpha(i+1) - 1$.



(a) (b)

fig. 2.32 : (a) Graphe

(b) Matrice d'adjacence, forme condensée

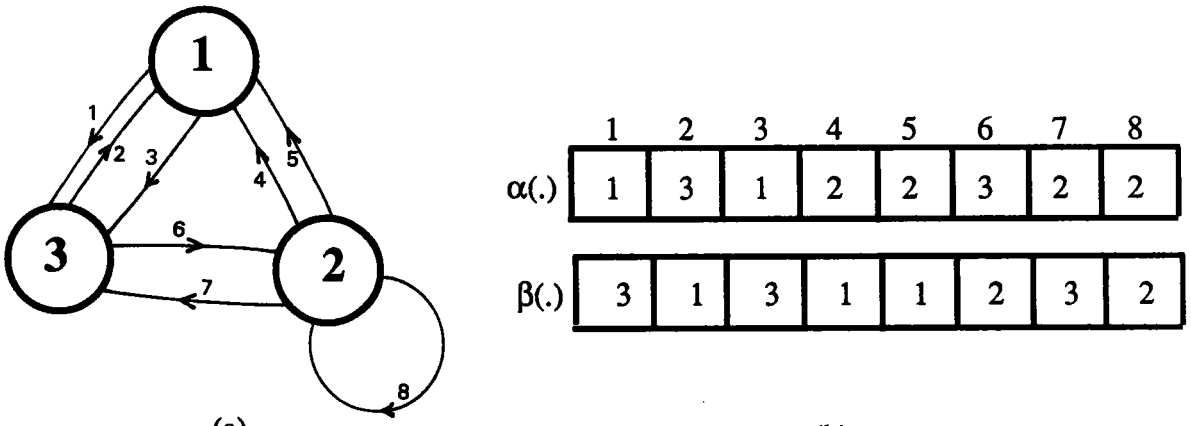
b) Représentation à partir de la matrice d'incidence sommets-arcs ou arcs-sommets.

Deux méthodes sont possibles :

b1) Liste des arêtes.

La matrice d'incidence, forme condensée, utilise deux tableaux $\alpha(.)$ et $\beta(.)$ de dimension M donnant pour chaque arc i = 1, 2,....M, le numéro $\alpha(i)$ du noeud d'origine et $\beta(i)$ du noeud d'extrémité.

Exemple : (fig. 2.33)



(a) (b)

fig. 2.33 : (a) Graphe

(b) Matrice d'incidence, forme condensée

b2) Liste des noeuds.

La matrice d'incidence, forme condensée, utilise ici trois tableaux $\alpha(.)$ de longueur N+1, $\beta(.)$ et $\delta(.)$ de longueur M. Pour chaque noeud i, la liste des arcs reliés au noeud i est contenue dans le tableau $\beta(.)$ à partir de la case $\alpha(i)$ et $\alpha(i+1) - 1$. L'autre extrémité de l'arc est précisée dans le tableau $\delta(.)$.

Exemple : considérons l'exemple précédent, la représentation est alors donnée par la figure 2.34.

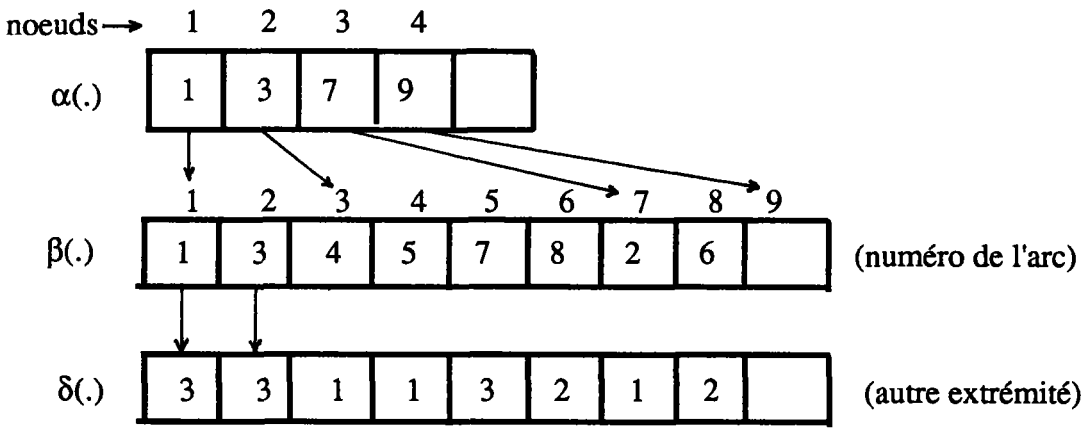


fig. 2.34 : Matrice d'incidence sous forme condensée

Remarque : Dans tous les cas, si le graphe est pondéré, il faudra utiliser un tableau supplémentaire où sera consignée la valeur du poids de chaque arc.

2.3.1.3) Temps de calcul. [AHO 74] [SED 83]

La complexité d'un algorithme, fonction du nombre d'opérations à exécuter, donne une approximation du temps de calcul. On suppose que toutes les opérations (telles que comparaisons, branchements, fonctions analytiques, fonctions trigonométriques, partie entière,...) s'effectuent en une même unité de temps. La complexité f(n) d'un algorithme pourra être caractérisée par une borne supérieure = $0(g(n))$ et par une borne inférieure = $\Omega(h(n))$.

Exemple : Si un algorithme possède une complexité en $0(M.N^2)$, avec M et N le nombre d'arêtes et de noeuds du graphe, le temps de calcul sera

$$T <= C1.(M.N^2) + C2$$

où C1 et C2 sont deux constantes indépendantes de M et N.

2.3.2) Algorithmes.

Parmi les algorithmes les plus utilisés, nous citerons

2.3.2.1) Algorithme de MOORE-DIJKSTRA. [MOO 57] [DIJ 59]

Cet algorithme nous donne le plus court chemin d'un sommet à tous les autres dans un graphe. Le principe repose sur le fait que si un noeud p quelconque de l'arbre de recherche est situé sur le chemin minimal du noeud source S au noeud but B, alors le chemin minimal de S à P fait partie du chemin minimal de S à B.

Définition.

- Soit $\pi^*(i)$ la longueur minimale du chemin du noeud 1 au noeud i. Nous avons en particulier $\pi^*(1) = 0$.
- L'ensemble des sommets X est partitionné en deux sous-ensembles S et $\overline{S}$. Initialement nous avons $S = \{1\}$ et $\overline{S} = X\text{-}S$.
- a chaque sommet i de X on affecte une étiquette $\pi(i)$ vérifiant la propriété suivante

- si $i \in S$ $\pi(i) = \pi^*(i)$
- si $i \in \overline{S}$ $\pi(i) = \min (\pi(k) + lki)$ avec $k \in S$ et $k \in \Gamma_i^{-1}$

Algorithme.

a)Phase d'initialisation.

$S=\{1\}$ $\overline{S} = \{2,3,\ldots\ldots,N\}$

$$\pi(1) = O \ , \ \pi(i) = \begin{cases} lij \text{ si } i \in \Gamma_i \\ +\infty \text{ sinon} \end{cases}$$

b)Sélectionner $j \in \overline{S}$ tel que $\pi(i) = \min \pi(j)$ avec $j \in \overline{S}$

faire $\overline{S} - \{j\} \longrightarrow \overline{S}$

si $|\overline{S}| = 0$ FIN

sinon aller en (C).

c)Faire pour tout $i \in \Gamma_j$ et $i \in \overline{S}$ $\min (\pi(i), \pi(j) + lij) \longrightarrow \pi(i)$

Retourner en (b).

Complexité.

D'après [GON 79] la complexité de l'algorithme de MOORE-DIJKSTRA est en $0(N^2)$. Si le graphe est peu dense et s'il est défini par ses successeurs de chaque sommet (matrice des adjacences), alors la complexité de l'algorithme est en 0(MlogN).

Applications.

Cet algorithme est notamment utilisé sur les projets suivants

- CART, Université de STANDFORD, [MOR 80],
- robot japonais, Faculté de Technologie d'OSAKA [FUJ 71].

2.3.2.2) Algorithme de LEE. [LEE 61] [PRU 86] [SIA 86]

L'algorithme de LEE est bien adapté à la recherche d'un chemin optimal à travers un environnement modélisé par une grille.

Définition.

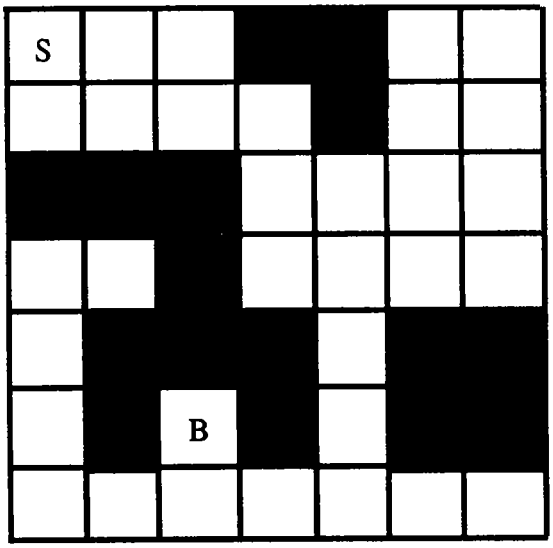


fig. 2.35 : Exemple d'environnement

case obstacle

case source

case but

Une case accessible (☐) est appelée

- case vide si aucune valeur ne lui est affectée,
- case pleine si une valeur lui a été affectée.

On désigne par i le numéro d'expansion.

Algorithme.

1) Sélectionner la case départ, lui affecter la valeur 0. Initialiser i à 1.

2) Affecter la valeur i à chaque case adjacente à la case comportant la valeur i-1.

3) Si la case but est pleine saut en 4

Sinon i=i+1

Saut en 2.

4) Fin.

Après la ième itération de l'algorithme, la case but est pleine.

Recherche de chemin.

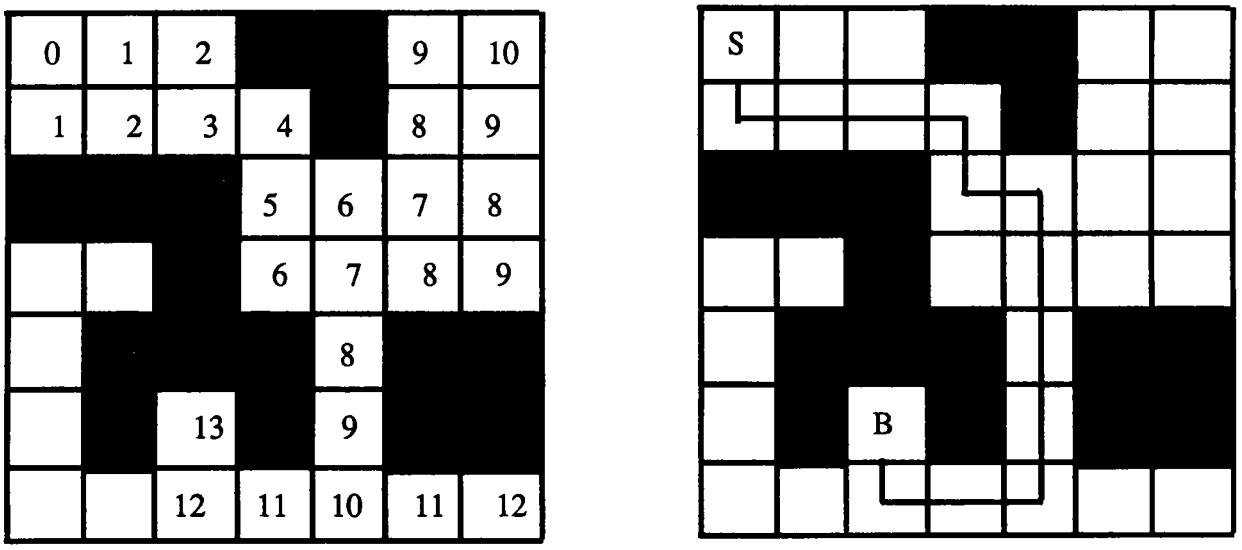


fig. 2.36 : Détermination de la trajectoire

La dernière case faisant partie de notre trajectoire est affectée de la valeur i, l'avant-dernière case est affectée de la valeur i-1, etc... Il nous est facile de retrouver ainsi l'ensemble des cellules faisant partie du chemin optimal.

Application.

Cet algorithme est notamment utilisé

- au LAAS, Université Paul Sabatier à Toulouse, [LAP 80] [PRA 80]
- sur le robot de Leningrad [IGN 73].

Cet algorithme de LEE peut être utilisé avec un environnement 3D. Dans ce cas, le modèle est représenté par plusieurs grilles juxtaposées(fig.2.37).

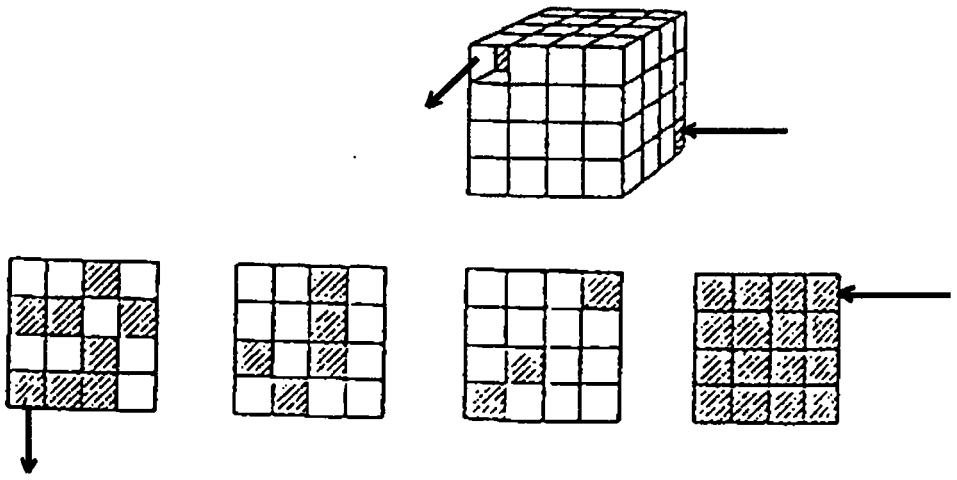


fig. 2.37 : Exemple d'environnement 3D

2.3.2.3) Algorithme de BELLMAN. [BEL 58]

L'algorithme de BELLMAN est une variante de celui de MOORE-DIJKSTRA. Il détermine le chemin de coût minimum entre un noeud et tous les autres.

Définition.

Soit $\pi^1(i)$ la longueur minimum du noeud 1 au noeud i et ne comprenant pas plus d'un arc.

Soit $\pi^k(i)$ la longueur minimum du noeud 1 au noeud i et ne comprenant pas plus de k arcs.

Principe.

1) Poser $\pi^0(1) = 0$ et $\pi^0(i) = +\infty$ pour tous les autre sommets i

2) à l'itération k, faire $\pi^k(1) = 0$ et pour tous les autres sommets i, poser

$$\pi^k(i) = \min_{j \in \Gamma_i^{k-1}} (\pi^{k-1}(j) + lij)$$

3) si $\pi^k(i) = \pi^{k-1}(i)$ pour tout i alors aller en (4)

si k <= N-1 aller en (2) avec k=k+1

si k = N il existe un circuit de longueur négative

4) FIN

Complexité.

S'il n'existe pas de circuit de longueur négative, l'algorithme de BELLMAN nécessite (N-1) itérations. L'étape 2 de l'algorithme nécessite M additions et M comparaisons. On peut en

conclure, d'après [GON 79] que l'algorithme de BELLMAN possède une complexité maximum en 0(M.N).

Application.

Cet algorithme a notamment été employé

- dans le projet JPL (Jet Propulsion Laboratory), [FLO 69]
- sur un robot japonais, Laboratoire d'Electronique de TOKYO, [KAK 72]

2.3.2.4) Algorithme A et A*. [HAR 68] [NIL 80]

Principe.

Cet algorithme est basé sur l'utilisation d'une fonction d'évaluation $\hat{f}(n)$. Cette fonction $\hat{f}(n)$ est l'estimation du coût du chemin optimal (noté f*(n)) reliant le noeud source au noeud but et passant par le noeud n.

Nous avons $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$

avec

-$\hat{g}(n)$: estimation du coût du chemin optimal g*(n) reliant le noeud source au noeud n.

-$\hat{h}(n)$: estimation du coût du chemin optimal h*(n) reliant le noeud n au noeud but.

D'une manière générale nous avons $\hat{g}(n) >= g^*(n)$. L'estimation de la distance du noeud n au noeud but s'effectue par l'intermédiaire de l'heuristique $\hat{h}(n)$.

La procédure ainsi décrite est appelée algorithme A. Si on n'utilise pas d'heuristique ($\hat{h}(n) = 0$), on obtient l'algorithme de recherche du plus court chemin vu par DIJKSTRA [chap. 2.3.2.1]. Si $\hat{h}(n) <= h^*(n), \forall n$ on obtient l'algorithme A*, grâce auquel nous sommes assurés de trouver un chemin s'il existe.

Algorithme A*.

Soit T un ensemble non vide de noeuds but.

1) Ouvrir le noeud source ns et calculer la fonction d'évaluation $\hat{f}(ns)$.
2) Sélectionner le noeud ouvert n pour lequel $\hat{f}(n)$ est minimum. Résoudre arbitrairement mais toujours en faveur d'un noeud $n \in T$.
3) Si $n \in T$, fermer le noeud n et aller en S.
4) Autrement, fermer le noeud n et développer le noeud n (c'est-à-dire déterminer ses

succeseurs).

Calculer $\hat{f}(n)$ pour chaque successeur de n et ouvrir chaque noeud non encore fermé.

Réouvrir chaque noeud ni, successeur de n, pour lesquels $\hat{f}(n_i)$ est plus petit que lorsque le noeud ni était considéré comme fermé.

Retourner en 2.

5) Fin.

Temps de parcours.

D'après [TOU 88], la complexité de l'algorithme A* est en $0(N^2)$. Si les noeuds sont stockés dans les listes triées (liste des noeuds), alors la complexité maxi requise par l'algorithme est en $0(\min (N^2, M \log N)$. .

Conclusion.

L'algorithme A*, appelé algorithme heuristique, diminue le temps de recherche et la capacité mémoire nécessaire, par contre il ne garantit pas une solution optimale. L'algorithme A a été utilisé au LERM, Université de Clermont-Ferrand, [RIC 81]. L'algorithme A* a été utilisé

- dans le projet SHAKEY, [RAP 76a] [RAP 76b]
- dans le projet ARGOS, Université Paul Sabatier [CAY 76] [CAY 78].

2.3.2.5) Cas particulier.

Dans le cas d'un mobile en translation simple dans un plan, la prise en compte de la taille d'un mobile peut se faire en «grossissant» l'obstacle et en réduisant le mobile à un point. Grossissement uniforme si le robot est circulaire, grossissement par somme de MINKOWSKI dans le cas d'un robot de forme polyédrique convexe.

Dans le cas d'un mobile en translation et rotation simultanées, de nombreuses méthodes, s'appuyant sur les dimensions réelles des obstacles ont été développées. Ces méthodes nécessitent une description détaillée du robot et de l'environnement. Partant d'une configuration initiale CI (position et orientation), elles recherchent s'il existe un mouvement continu pour parvenir à une configuration finale CF (fig. 2.38) D'après BOISSONAT [TOU 88] ces méthodes peuvent se classer en trois catégories :

- les méthodes directes, développées initialement par SCHWARTZ et SHARIR [SCH 83],
- les méthodes de projection, utilisées par [LEV 87] [SCH 84] [AVN 87],

- les méthodes de rétraction, employées par [ODU 85] [AVN 88a] [YAP 84].

L'avantage principal de ces méthodes est de ne pas grossir l'obstacle dans le but de tenir compte de la taille du mobile. En effet, un grossissement de l'obstacle pourrait, selon la forme du mobile et de l'environnement, provoquer l'obstruction de chemins possibles.
L'inconvénient est le temps très long nécessaire au calcul de la trajectoire.

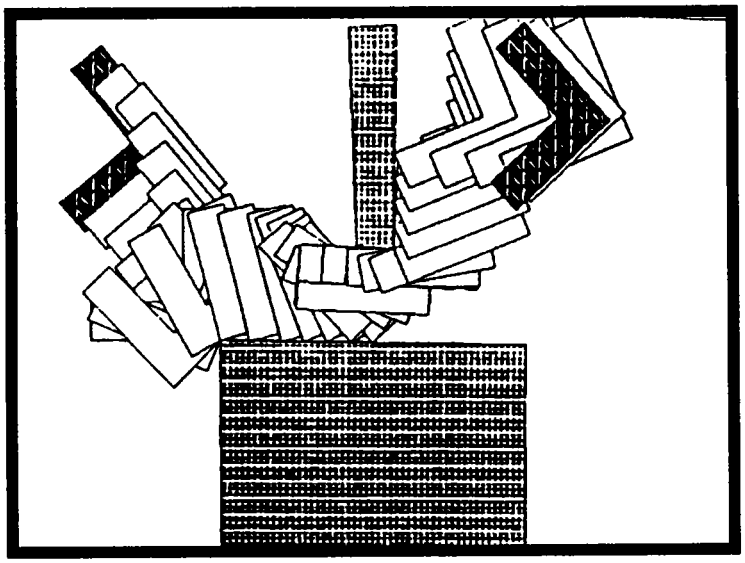


fig. 2.38 : Mobile en translation et en rotation dans le plan [AVN 88b]

2.3.3) Conclusion.

Les algorithmes dits exhaustifs (BELLMAN, MOORE-DIJKSTRA) présentent l'avantage de déterminer de manière certaine la solution dans un graphe fini, mais deviennent rapidement impraticables avec un nombre de noeuds et d'arcs important.

L'algorithme heuristique A* ne permet pas une exploitation complète de l'ensemble des solutions possibles, mais nous donne uns solution admissible tout en diminuant la place mémoire et le temps nécessaire.

CHAPITRE III :
MODELISATION PAR GRILLE NON-HOMOGENE RECHERCHE DE TRAJECTOIRE

III) MODELISATION PAR GRILLE NON-HOMOGENE ET RECHERCHE DE TRAJECTOIRE.

3.1) Introduction.

Nos travaux, développés dans ce chapitre, consistent :

- En une méthode de modélisation de l'espace d'évolution d'un robot mobile basée sur une cellularisation en grille non-homogène. Cette solution que nous utilisons en 2D et $2D^{1/2}$ présente l'avantage de n'occuper qu'un espace mémoire réduit, mais aussi d'être modifiable dynamiquement, (ce qui est indispensable pour l'apprentissage et dans le cas d'obstacles mobiles).
- En une méthode de recherche de trajectoire 2D et $2D^{1/2}$ par une association des algorithmes de LEE et A* en effectuant des opérations de masquage.

Les temps de modélisation et de planification, donnés à titre indicatif, sont obtenus en simulation sur un IBM PS/2 H21 équipé d'un processeur 80286 à 10 Mhz et d'un coprocesseur arithmétique 80287.

3.2) Modélisation statique d'un environnement 2D

3.2.1) Modélisation par grille [PRU 89 a]

3.2.1.1) Définition.

La modélisation par grille consiste à partager un espace en bandes horizontales et verticales dont les intersections forment des cellules élémentaires. A chaque cellule est associée une fonction déterminant les conditions d'accès. Soit une fonction binaire (accès autorisé, accès interdit), soit une fonction plus complexe prenant en compte d'autres éléments (tâche, géométrie...). La précision du modèle est liée à l'écartement de la maille. Le plus habituellement la grille est tracée de manière homogène en fonction de la précision désirée. Cette façon de procéder nécessite une largeur de maille faible si la précision désirée est élevée, ce qui pose un problème d'encombrement en mémoire. D'autre part, la recherche de chemin entre deux cellules ne pourra être effectuée qu'à l'aide d'un algorithme du type A* parce que l'arbre de recherche serait trop important. Un algorithme de type LEE est plus adapté, mais nécessite un espace mémoire et un temps de calcul trop importants.

3.2.1.2) Principe de la modélisation par grille non-homogène.

Le principe de la modélisation que nous proposons repose sur le partitionnement de l'espace complet par une grille non-homogène dont les mailles ont une largeur et une longueur qui sont fonctions de l'environnement. Les obstacles sont représentés, pour le modèle 2D, par des polygones convexes ou concaves. L'intersection des bandes horizontales et verticales forme les cellules élémentaires auxquelles est associée une variable binaire selon qu'une cellule fait partie de l'espace libre ou d'un obstacle.

3.2.1.3) Erreur de modélisation.

La forme de l'objet à modéliser influence la dimension des mailles. Nous définissons l'erreur de modélisation comme la plus grande distance à l'intérieur d'une cellule entre l'objet à modéliser et la limite de la cellule.

Dans le cas de la modélisation (2D) d'un carré de côté c, la grille qui modélise cet objet sans erreur a une largeur de maille de valeur c et coïncide avec les côtés du carré. Dans le cas d'un objet quelconque, l'erreur maximale sera de $(l^2 + L^2)^{1/2}$ avec L et l les largeurs horizontale et verticale des mailles. Dans le contexte de la robotique mobile il n'est pas nécessaire, comme en CAO, de modéliser un environnement avec une précision élevée. Le critère de précision constitue la navigation «au plus près» d'un obstacle. Dans le cas d'une évolution très proche d'un obstacle, des capteurs de proximétrie tels que les ultra-sons viendront pallier le manque de précision.

3.2.1.4) Calcul de la distance minimale et maximale d'approche.

Considérons deux distances particulières : les distances d'approche minimale et maximale entre le centre d'une cellule et la droite à modéliser (fig. 3.1). Le modèle de l'environnement est le support qui assurera l'établissement de la trajectoire qui, par commodité, s'effectue par rapport aux centres des cellules (fig. 3.2). Nous nous apercevons que l'erreur de modélisation est fonction du paramètre α, angle entre la droite à modéliser et la grille.

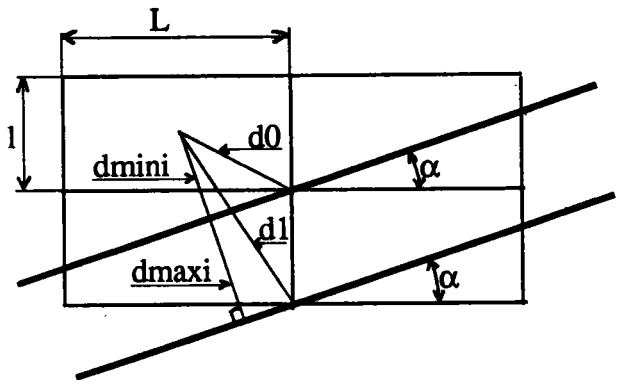


fig. 3.1 : Détermination des distances minimale et maximale d'approche

Nous définissons dmini comme la distance minimale d'approche de l'obstacle. Nous avons **dmini =d0.sin(α + β)**

avec $d0^2 = L^2/4 + l^2/4$

α : angle entre la droite à modéliser et la grille.

ß: : angle tel que tg β = l/L

Nous définissons dmaxi comme la distance maximale entre le mobile et l'obstacle lors de la détermination d'une trajectoire d'évolution «au plus près» de l'environnement.

dmaxi = d1.sin(α + δ)

avec $d1^2 = L^2/4 + (3l/L)^2$

δ : angle tel que tg δ = 3l/L

Dans le cas du déplacement du mobile, il est intéressant de se fixer une distance minimale d'approche de l'obstacle, notamment pour des raisons de sécurité.

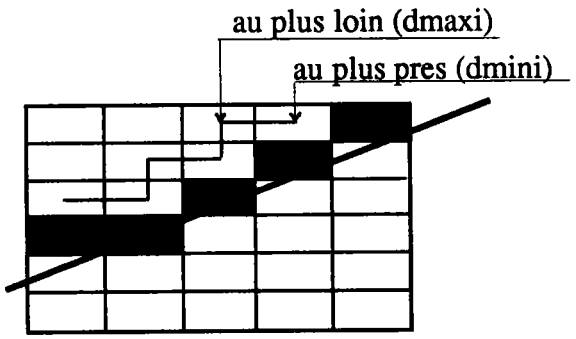


fig. 3.2 : Représentation des distances minimales et maximales d'approche.

La constante dmini peut être reliée aux dimensions du robot, par exemple **2dmini = k.(dimension du robot)** où k est un coefficient de sécurité.

3.2.1.5) Détermination de la taille des cellules.

Afin de déterminer la longueur et la largeur des cellules, nous allons alternativement faire tendre ces valeurs vers zéro (fig. 3.3) :

a) Plaçons-nous dans le cas où L est finie et l tend vers zéro (le maillage devient

infiniment petit).

$$\lim_{l \to 0} dmini = \lim_{l \to 0} d0.\sin(\alpha + \beta) = (L/2).\sin\alpha \qquad \text{(cas a)}$$

b) Plaçons-nous dans le cas où l est finie et L tend vers zéro (le maillage devient infiniment petit).

$$\lim_{L \to 0} dmini = \lim_{L \to 0} d0.\sin(\alpha + \beta) = (l/2).\cos\alpha \qquad \text{(cas b)}$$

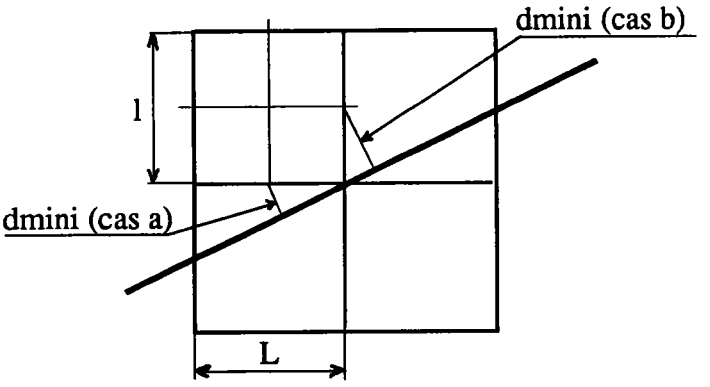


fig. 3.3 : Représentation de la distance dmini

c) Résultats:

nous obtenons

$$\text{(cas a)} \longrightarrow L = \frac{2.dmini}{\sin\alpha} \qquad (1)$$

$$\text{(cas b)} \longrightarrow l = \frac{2.dmini}{\cos\alpha} \qquad (2)$$

3.2.1.6) Détermination du nombre de partition.

Connaissant les dimensions d'une cellule (L et l), prenons le cas d'un obstacle réel décrit par une droite de longueur finie LG et désignons par m et n respectivement le nombre de partitions verticale et horizontale (fig.3.4).

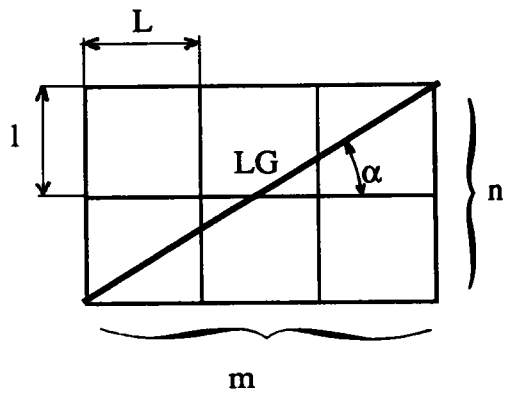


fig. 3.4 : Nombre de partition

D'après la figure 3.4, nous avons

$$m = \frac{LG.\cos\alpha}{L} \qquad n = \frac{LG.\sin\alpha}{l}$$

avec **(1)** et **(2)**

nous obtenons

$$n = \frac{LG.\sin\alpha.\cos\alpha}{2.dmini} \qquad m = \frac{LG.\cos\alpha.\sin\alpha}{2.dmini}$$

soit

$$\mathbf{m = n = \frac{LG.\sin(2\alpha)}{4.dmini}}$$

Les grandeurs de m et n sont des valeurs entières par excès. Nous remarquons que le nombre de partition maximal

$$\mathbf{m_{maxi} = n_{maxi} = \frac{LG}{4.dmini}}$$

est obtenu pour $\alpha = \pi/4$

3.2.2) Procédure de détermination du modèle.

Considérons un environnement dont les obstacles polygonaux sont représentés par des ensembles de droites et un référentiel orthogonal. La détermination du modèle est effectuée en 3 phases :

- création de la grille non-homogène,
- transformation en grille homogène,
- création du modèle.

3.2.2.1) Création de la grille.

a) La première étape de modélisation consiste à partager l'espace par des droites parallèles à un axe du référentiel 2D et passant par tous les sommets. Puis nous réitérons l'opération par rapport à l'autre axe du référentiel. La grille obtenue comporte n+1 bandes horizontales et m+1 bandes verticales avec n et m le nombre de sommets indépendants, c'est-à-dire qui ne se trouvent pas sur une même verticale ou horizontale (fig.3.5).

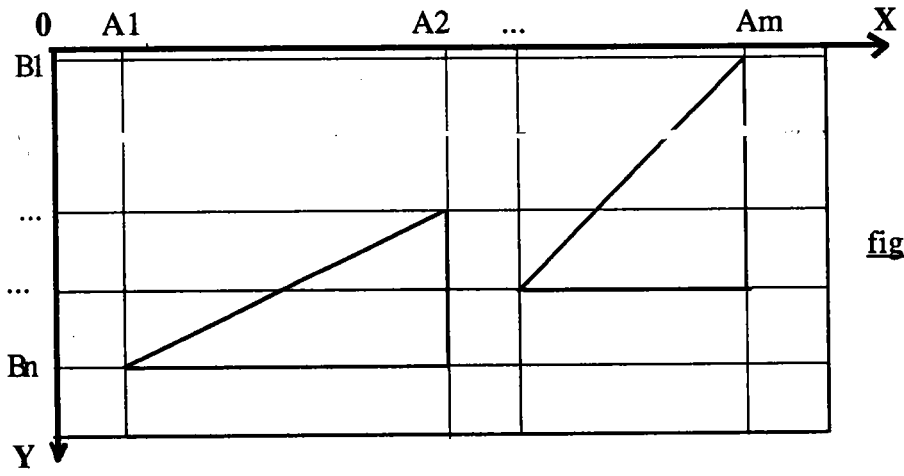


fig. 3.5 : Partitionnement par droites parallèles

b) Dans une deuxième étape nous créons 2 tableaux respectivement avec les coordonnées des droites parallèles selon chaque axe.

A = {A1, A2,..., Am}

B = {B1, B2,..., Bn}

c) Dans une troisième étape nous cherchons pour toutes les bandes verticales et horizontales ainsi formées, la pente la plus proche de 1, c'est-à-dire pour α le plus proche de $\pi/4$.

d) Ensuite nous calculons les partitions maximales associées à chaque bande.

Pa = {mO, m1,..., mm}

Pb = {n0, n1,..., nn}

Dans l'exemple de la figure 3.6 nous trouvons

Pa = {1, 7, 1, 8, 1}

Pb = {1, 5, 3, 2, 1}

e) Dans une dernière étape nous effectuons la partition selon mi et ni.

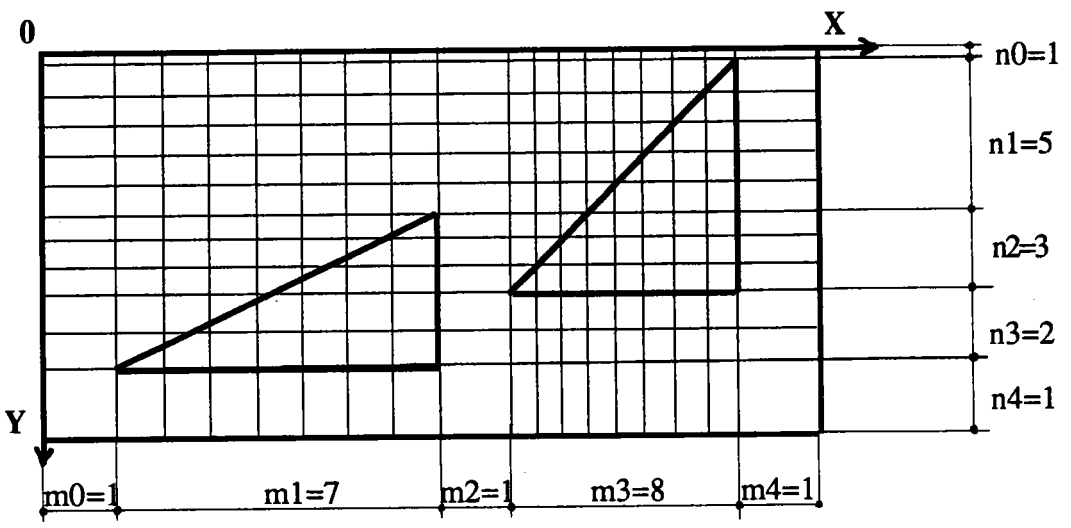


fig.3.6 : Partitionnement en bandes et colonnes

3.2.2.2) Transformation en grille homogène et transformation de coordonnées.

L'étape suivante consiste à transformer la grille de manière à la rendre homogène. Chaque bande verticale ou horizontale devant être de la même dimension, cette opération entraîne que tous les sommets voient leurs coordonnées modifiées (fig. 3.7).

Soient Si = {Ai, Bi} les coordonnées d'un sommet et nj, mj la partition réalisée dans la jième bande (horizontale ou verticale), les nouvelles coordonnées du sommet Si sont :

$$S^*i = \{ \sum_{j=0}^{j=i} mj , \sum_{j=0}^{j=i} nj \} = \{ \beta i , \gamma i \}$$

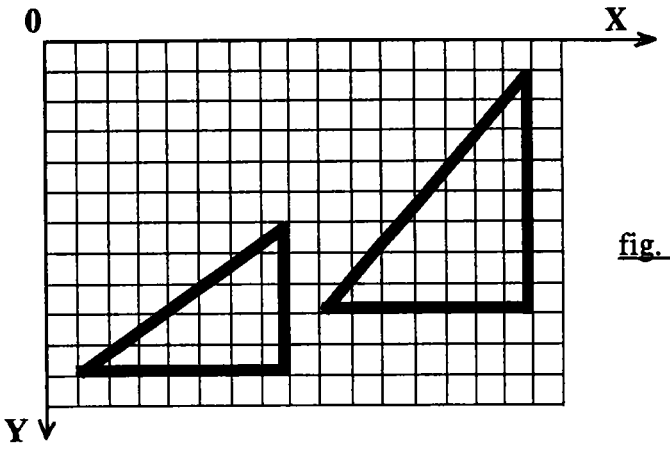


fig. 3.7 : Transformation en grille homogène

Propriété : l'intérêt de cette transformation réside dans la simplification d'un traitement ultérieur des informations pour l'obtention du modèle final.

3.2.2.3) Création du modèle.

Le modèle final M(p, q) est constitué de p mots de q bits représentant l'état des différentes cellules définies par la grille. Les mots représentent les bandes de cellules horizontales et chaque bit l'état des cellules verticales dans la bande considérée. Si l'accès d'une cellule est autorisée celle-ci est mise à «1», dans le cas contraire elle est forcée à «0».

Nous utilisons, pour générer le modèle, l'algorithme de BRESENHAM (paragraphe suivant) et nous l'avons retenu parmi tant d'autres (STOCKTOM, LUCAS, EARNSHAW , etc... [ROG 82]) pour des raisons de simplicité, de précision et de rapidité.

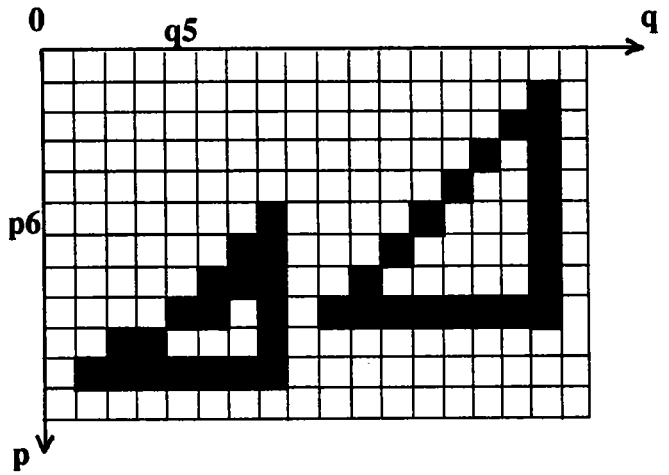


fig. 3.8 : Modèle servant au calcul de M(p,q)

Dans l'exemple de la fig. 3.8 :

- le mot p6 vaut : 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1
- le mot q5 vaut : 1 1 1 1 1 1 1 1 1 0 1 0 1
- le bit M(6, 5) vaut 1

Nous remarquons que certaines cases, situées à l'intérieur des limites d'un obstacle, sont d'accès autorisé (valeur 1) mais qu'il est impossible d'y accéder du fait de leur position.

3.2.2.4) Algorithme de BRESENHAM [ROG 82]

Cet algorithme utilisé en infographie nous permet de faire la meilleure approximation possible d'une droite à l'aide de points ou de cellules.

Soient Dk un segment de droite appartenant à un obstacle, $S^*k = \{ \beta k , \gamma k \}$ et $S^*k+1 = \{\beta k+1 , \gamma k+1 \}$ les sommets du modèle réduit de Dk. Nous nous trouvons dans un modèle discret où il y a correspondance entre les coordonnées des cellules et la position des bits dans le mot.

Soit la fonction Signe (z)= 1 si $z > 0$

= O si $z = 0$

= -1 si $z < 0$

Algorithme de BRESENHAM

$x = \beta k$; $y = \gamma k$;

$\Delta x = abs(\beta k+1 - \beta k)$; $\Delta y = abs(\gamma k+1 - \gamma k)$;

$s1 = Signe(\beta k+1 - \beta k)$; $s2 = Signe(\gamma k+1 - \gamma k)$;

Si $\Delta y > \Delta x$

alors | temp = Δx

| $\Delta x = \Delta y$

| Δy = temp

| interchange = 1

sinon | interchange = 0

$e = \Delta x/\Delta y - 0,5$

Faire de i=1 à Δx

M(x,y) = 0

Tant que e >= 0

Si interchange = 1

alors x = x + s1

sinon y = y + s2

```
        | e = e - 1
Si interchange= 1
        alors y = y + s2
        sinon x = x + s1
e = e + Δy/Δx
```

Exemple.

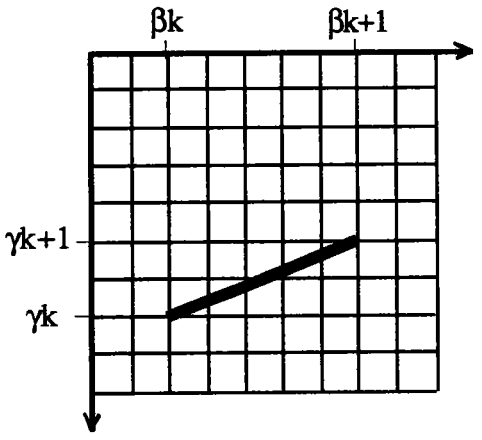


fig. 3.9 : Droite sur grille homogène

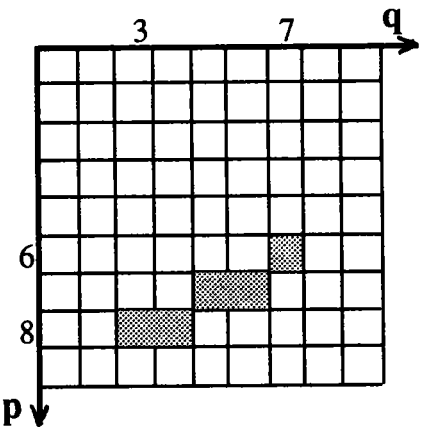


fig. 3.10 : Modèle généré

Soient S*k = {3, 8} et S*k+1 = {8, 6} les points extrêmes du segment de droite Dk.

En parcourant l'algorithme nous obtenons

x=3 ; y=8;

Δx=5;Δy=2;

s1=1;s2=-1;

interchange=0;

e=2/5 -1/2= -1/10;

1ère itération i=1;M(3,8)=0;
x=3+1=4;
e=-1/10 +2/5 =3/10;

2ème itération i=2;M(4,8)=0;
y=8-1=7;e=3/10 - 1=-7/10;
x=4+1=5;
e=-7/10 +2/5=-3/10;

3ème itération i=3;M(5,7)=0;
x=5+1=6;
e=-3/10 + 2/5=1/10;

4ème itération i=4;M(6,7)=0;
y=7-1=6;e=1/10 - 1=-9/10;
x=6+1=7;
e=-9/10 +2/5=-5/10;

5ème itération i=5;M(7,6)=0;
x=7+1=8;
e=-5/10 + 2/5=-1/10;

fin

3.2.2.5) Prise en compte des dimensions du robot.

Elle s'effectue suivant le type d'arête délimitant les obstacles que l'on va rencontrer :

- arêtes non parallèles aux axes du référentiel .

La modélisation décrite précédemment tient compte des dimensions du robot. Compte tenu de la distance minimale d'approche dmini calculée au chapitre 3.2.1.4, nous allons déterminer la distance entre deux droites parallèles qui permettra le passage du robot.

a) Soient deux segments, l'un en face de l'autre, la distance minimale qui permet de manière certaine le passage, est égale à 4L sur l'horizontale et égale à 4l sur la verticale (fig. 3.11). Cette disposition évite que les cellules intégrant un élément d'obstacle ne soient contigues.

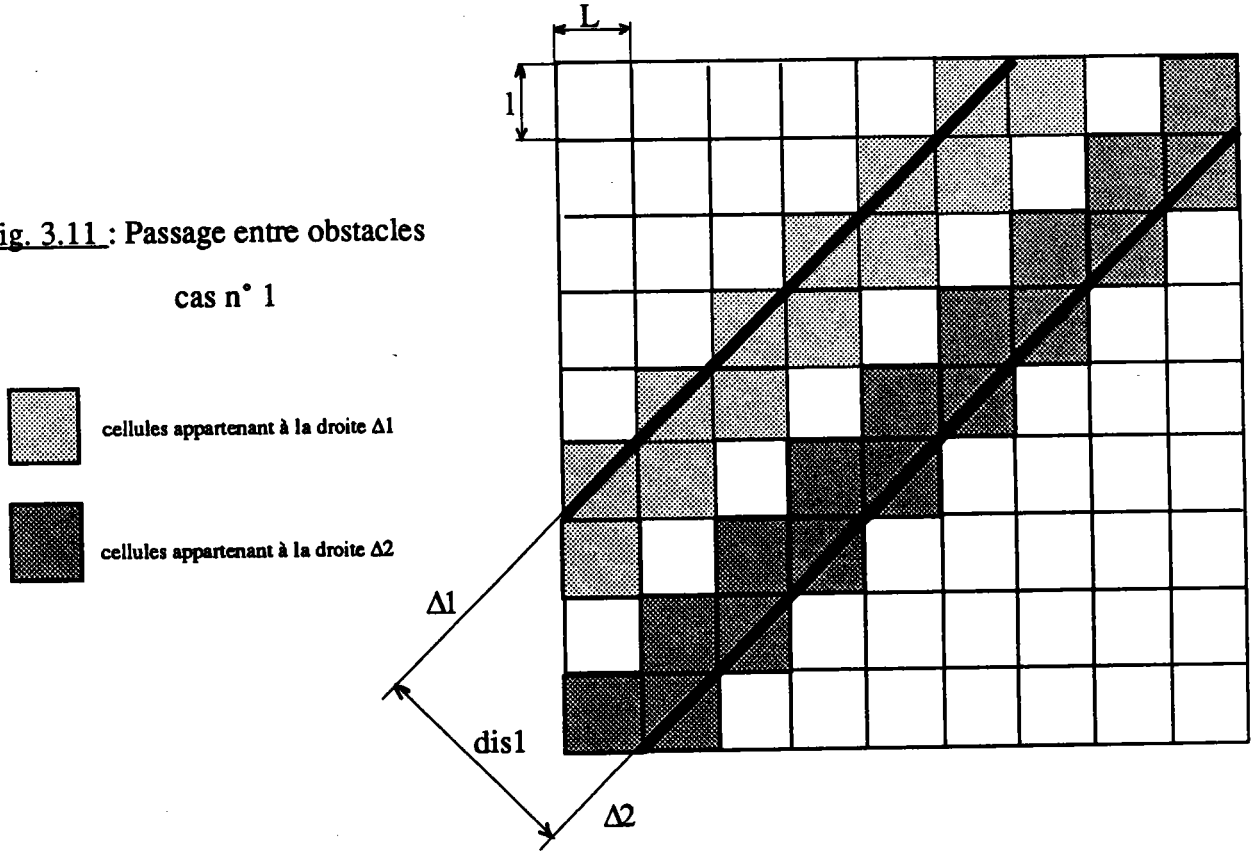


fig. 3.11 : Passage entre obstacles

cas n° 1

Pour permettre le passage d'un robot entre deux droites parallèles, nous devons satisfaire la relation suivante :

dis1>4dmini

b) Pour que le contact entre segments soit réalisé, il faut que les distances sur l'horizontale soient inférieures à 2L et sur la verticale à 2l (fig. 3.12).

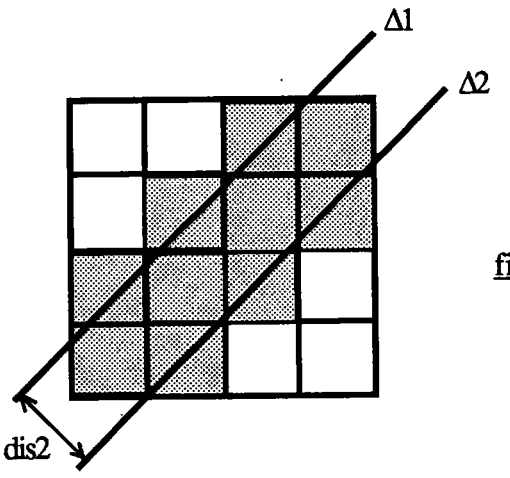


fig. 3.12 : Passage entre obstacles, cas n°2

Si dis2<2dmini, nous constatons que certaines cellules appartiennent aux deux droites, donc le passage est impossible. Pour que le passage entre deux droites parallèles ne soit pas possible, il faut que

dis2<2dmini

c) Entre ces deux distances, le contact entre les cellules du mobile est une fonction de la longueur du segment et de l'angle α(fig.3.13).

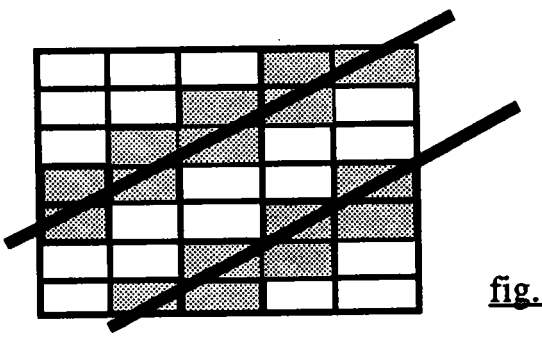


fig. 3.13 : Passage entre obstacles, cas n°3

d) Récapitulatif.

Soit dis la distance normale entre deux segments de droite parallèles . Si nous prenons **2dmini = k.(dimension du robot)** (cf chap. 3.2.1.4), nous obtenons les relations suivantes

dis>2.k.(dimension du robot) le passage est possible entre ces deux segments de droite.

dis>k.(dimension du robot) le passage est probable.

dis<k.(dimension du robot) le passage n'est pas possible.

- Arêtes parallèles aux axes de notre référentiel.

Dans ce cas la prise en compte des dimensions du robot se fera en agrandissant partiellement les obstacles :

- par les sommes de MINKOWSKI [POL 88] si le robot possède une forme polygonale,

- d'une valeur constante r, équivalente au rayon du robot, si la forme de celui-ci est assimilée à un cylindre (cas du véhicule VAHM).

Cet agrandissement se fera sur les arêtes parallèles aux axes du référentiel (fig. 3.14).

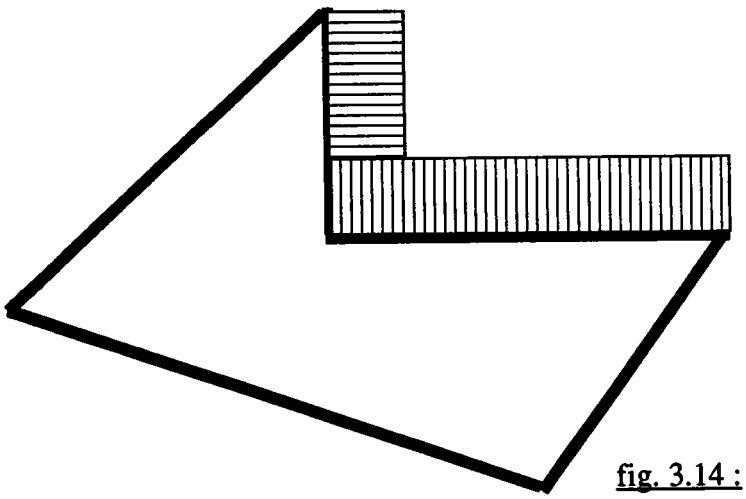


fig. 3.14 : Agrandissement partiel d'un obstacle

3.2.3) Caractéristiques du modèle.

3.2.3.1) Complexité

La complexité de la détermination du modèle est égale à la complexité maximale des diverses opérations prises indépendamment.

Pour expliciter ceci, prenons pour exemple la création du tableau A contenant les abscisses des partitionnements (chap. 3.2.2.1). Soient m le nombre de sommets indépendants suivant x et p

le nombre d'arêtes. Le classement d'un nombre dans un tableau de i éléments nécessite $\lceil \log_2 i \rceil$ opérations avec $\lceil \log_2 i \rceil$ valeur entière par excès de $\log_2 i$ [PAP 82] .

Le nombre total d'opérations pour le classement des m sommets est

$$\sum_{i=1}^{i=m} \log_2 i = \sum_{i=1}^{i=m} \frac{\text{Log } i}{\text{Log2}} = \frac{1}{\text{Log2}} [i \text{ Logi} - i]$$

d'où une complexité de l'ordre de 0(mLogm).

Nous procédons d'une manière similaire pour les autres opérations nécessaires à la création du modèle et nous obtenons les résultats du tableau suivant :

Opération	**Nombre d'opérations**	**Complexité**
Création du tableau A	Σ log i	0(m.Log m)
Création du tableau B	Σ log i	0(n.Log n)
Détermination de la pente la plus proche de 1 dans chaque bande verticale	(m+1).p	0(m.p)
Détermination de la pente la plus proche de 1 dans chaque bande horizontale	(n+1).p	0(n.p)
Calcul du nombre de partitions dans chaque bande verticale	(m+1).constante	0(m)
Détermination du nombre de partitions dans chaque bande horizontale	(n+1).constante	0(n)

avec n nombre de sommets indépendants suivant y.

La complexité maximale est en 0(max(mp, np)), c'est-à-dire $0(p^2)$ sachant que max(m, n)=p.

3.2.3.2) Place mémoire

L'utilisation d'une grille adaptative (ou grille non-homogène) nécessite de la place mémoire pour:

- le modèle de calcul M(p,q)
- les tableaux contenant les coordonnées des droites réalisant le découpage de l'environnement suivant les deux axes et passant par les sommets des obstacles.
- les tableaux contenant le nombre de partitions associés à chaque bande.

Nous avons modélisé l'environnement représenté par la figure 3.18 (a) par trois méthodes (polygones convexes, quadtrée et grille adaptative) et nous avons comparé la place mémoire requise.

Modélisation par polygones convexes

La décomposition en polygones convexes nous donne, au minimun, 13 polygones et nécessite 110 entiers (mots de 16 bits) pour sa mémorisation.

Modélisation par quadtrée.

Le partitionnement de l'environnement par quadtrée nous donne une arborescence de 120 noeuds et 380 feuilles(divisions jusqu'à la 5ème génération). La recherche de trajectoires dans cette arborescence nécessite 7 pointeurs par noeuds[FRY 87] soit un total de 840 entiers.

Modélisation par grille adaptative.

Ce modèle nécessite 28 entiers longs (32 bits) (fig.3.18 d), les tableaux contenant les coordonnées des droites réalisant le découpage possèdent 23 entiers et les tableaux précisant le nombre de partitions seront composés de 23 entiers. Cette méthode modélise l'environnement grâce à 102 entiers.

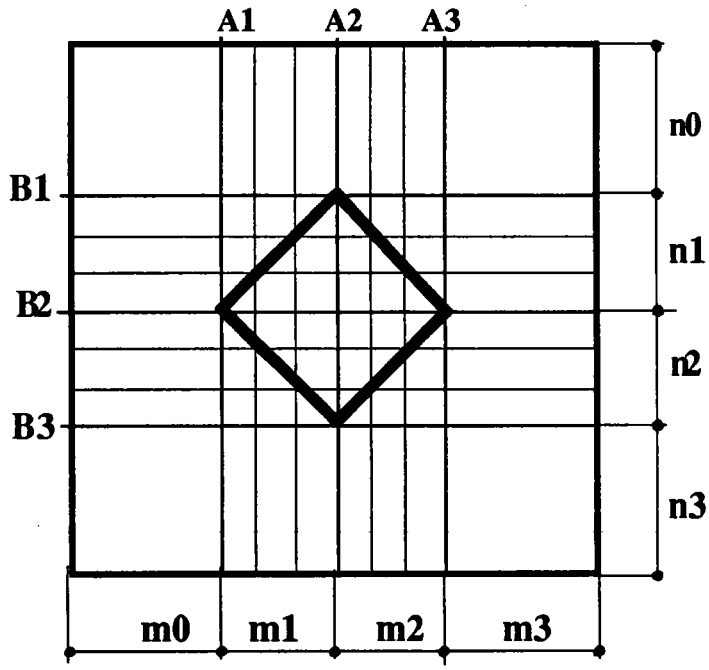
Nous constatons que la modélisation par grille adaptative nécessite une mémoire plus réduite. Outre cet avantage, le modèle généré est bien adapté à la recherche de trajectoire par l'algorithme heuristique que nous développerons paragraphe 3.5.

3.2.4) Réduction du modèle.

Il s'avère que dans une pièce les meubles et accessoires sont souvent alignés et parallèles aux limites. D'autre part, d'après la formule

$$m=n=\frac{LG.\sin(2\alpha)}{4.dmini}$$

vue au chapitre 3.2.1.6, nous remarquons que le nombre de partitions est fonction de l'angle α (entre le segment à modéliser et un axe du référentiel). Ce nombre de partitions tend vers zéro lorsque α tend vers zéro. Si nous prenons une autre base, non obligatoirement orthogonale, telle que α soit minimal, nous réduisons le nombre d'informations à mémoriser (fig. 3.15 et 3.16).



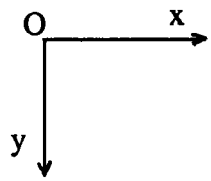
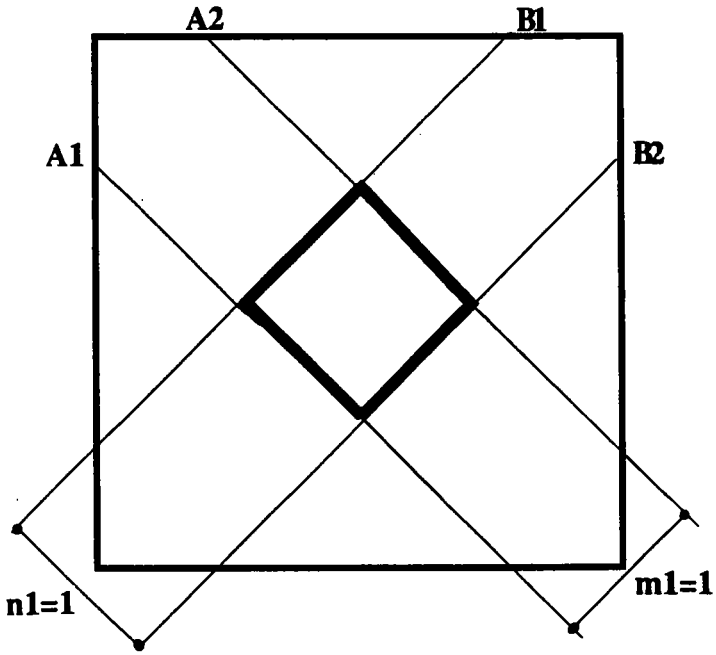


fig. 3.15 : Partitionnement avec le repère (Ox,Oy)



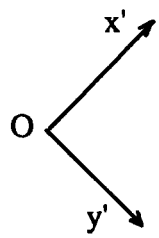


fig. 3.16 : Partitionnement avec le repère (Ox',Oy')

3.2.4.1) Procédure de réduction de la grille.

a) Recherche de l'angle optimal.

Cette étape consiste à trouver les angles $\alpha 1$ et $\alpha 2$ de rotation des axes orthogonaux afin de trouver le nombre minimal de partitions.

a1) Changement de base. Il s'agit de calculer les positions des sommets dans la nouvelle base. Soient $\theta 1$ et $\theta 2$ les rotations des axes x et y qui deviennent x' et y' après rotation (fig. 3.17).

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \frac{1}{\cos\varepsilon} \begin{vmatrix} \cos\theta 2 & \sin\theta 2 \\ -\sin\theta 1 & \cos\theta 1 \end{vmatrix}$$

avec $\varepsilon = \theta 2 - \theta 1$

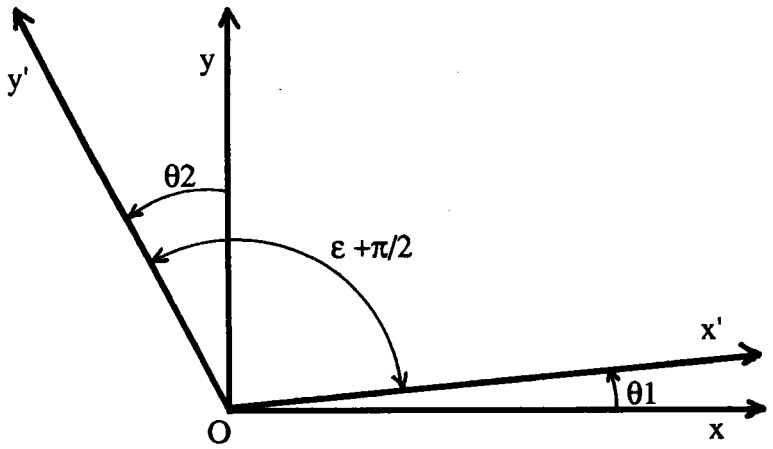


fig. 3.17 : Changement de base

a2) Calcul des partitions minimales.

Soient $Si^* = \{Ai, Bi\} = \{ \sum_{j=0}^{j=i} m'j , \sum_{j=0}^{j=i} n'j \}$ les coordonnées d'un point dans la base [X']. Nous recherchons les angles $\theta 1$ et $\theta 2$ afin de minimiser les termes Σ m'j et Σ n'j.

3.2.4.2) Conclusion.

Actuellement la recherche des angles $\theta 1$ et $\theta 2$ s'effectue de manière exhaustive, ce qui pénalise le temps de recherche du modèle. Si l'environnement n'a pas d'orientation privilégiée, il est possible que le processus ne délivre pas de résultats substantiellement meilleurs. Une disposition aléatoire d'un grand nombre de segments donne une forte probabilité de rencontrer une pente proche de 1.

3.2.5) Exemple.

Cet exemple récapitule les différentes étapes de la modélisation.

L'environnement composé d'obstacles polygonaux est découpé en bandes horizontales et verticales par des droites parallèles aux axes 0x, 0y (fig. 3.18a). Chaque bande est ensuite découpée en un certain nombre de partitions (fig. 3.18b).

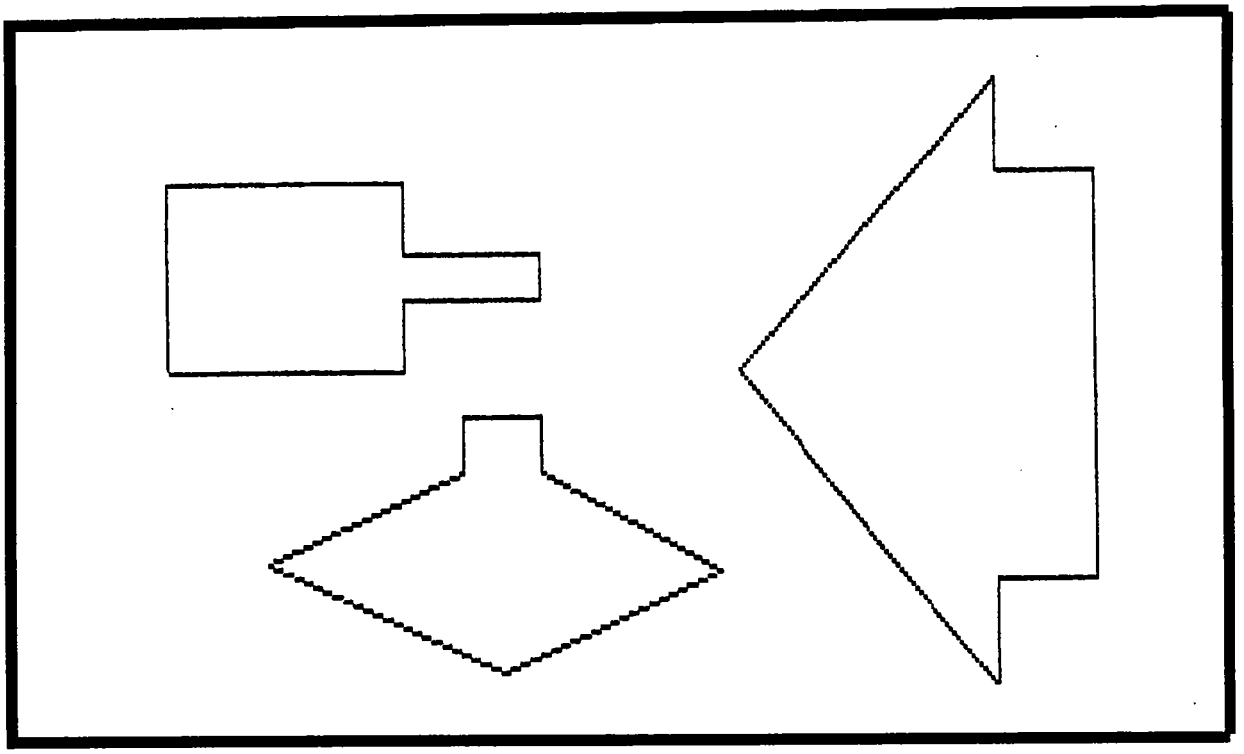


fig. 3.18 a : Modèle géométrique de départ

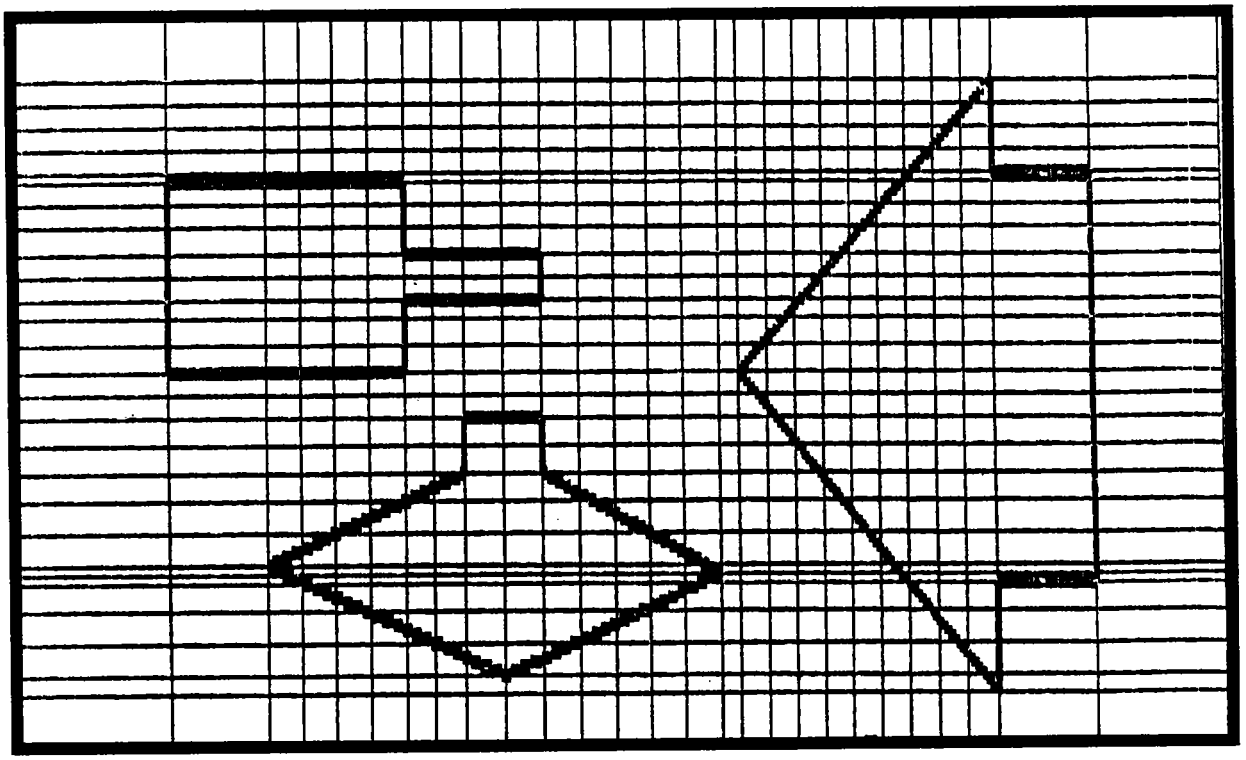


fig. 3.18 b : Partitionnement en bandes et colonnes

On effectue ensuite la transformation en une grille homogène en modifiant les coordonnées de chaque sommet (fig. 3.18c). Le modèle final est alors constitué de p mots de q bits où p représente la ligne et q représente la colonne (fig. 3.18d).

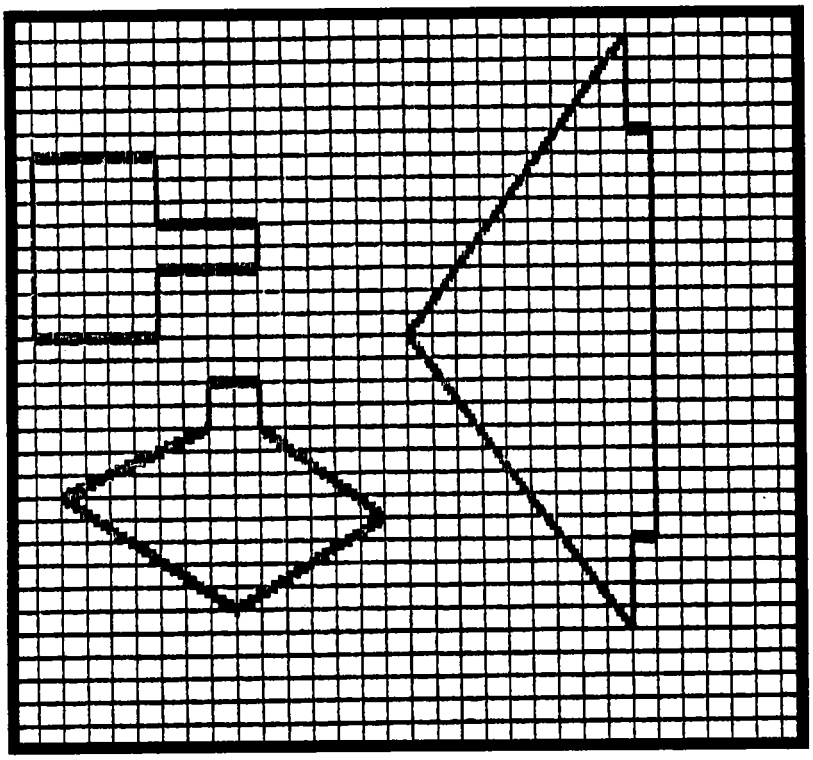


fig. 3.18 c : Transformation en grille homogène

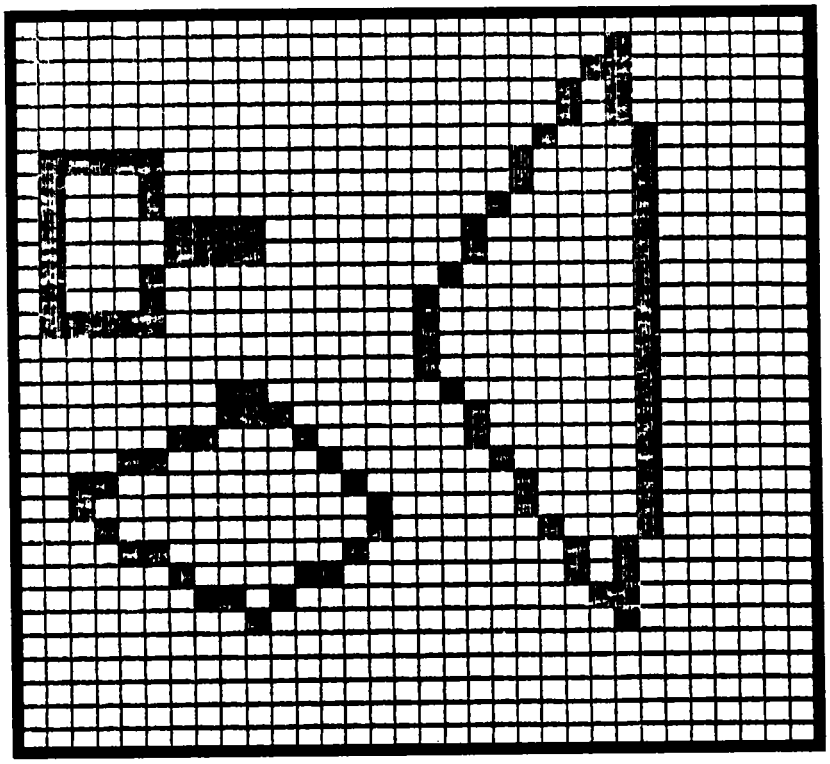


fig. 3.18 d :Obtention du modèle de calcul

L'ensemble des opérations de modélisation a duré 0,33 seconde. Le modèle de calcul ainsi obtenu sera utilisé tel quel dans le chapitre recherche de trajectoire.

3.3) Adaptation du modèle dans le cas d'un environnement dynamique 2D.

3.3.1) Introduction.

Le modèle statique étant construit, il n'est pas envisageable de le recalculer au fur et à mesure de sa modification dans le cas où l'environnement est constitué d'objets mobiles. En effet, le temps de calcul du nouveau modèle retarde d'autant la recherche de trajectoire du robot mobile. Il est par conséquent indispensable de minimiser la phase de remise à jour du modèle et donc d'avoir une modification dynamique. Cette minimisation sera obtenue par une modification localisée à la zone concernée par les déplacements. Nous allons décrire cette méthodologie au travers d'un exemple.

3.3.2) Principe.

Considérons l'environnement représenté à l'instant t1 par la figure 3.19a. Il est composé de 5 obstacles (01 à 05).

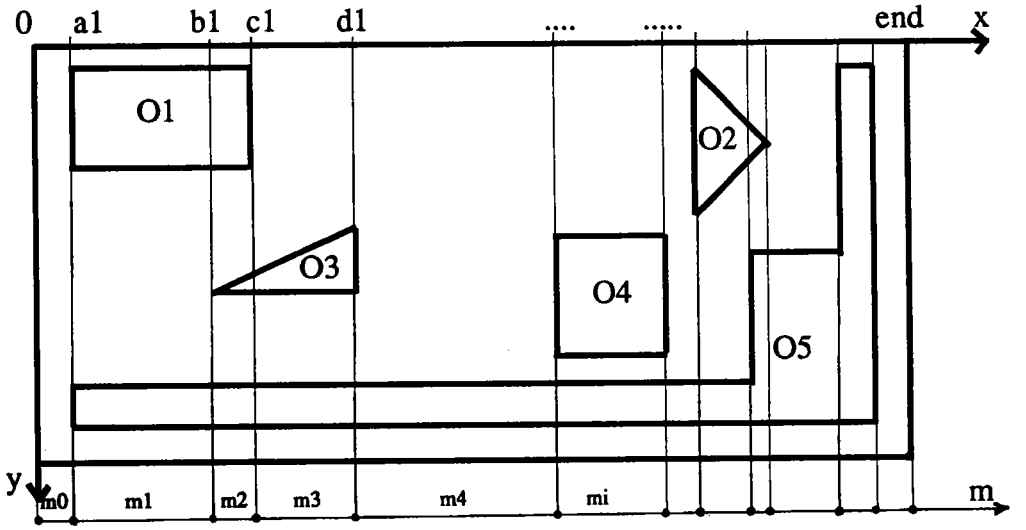


fig. 3.19 a : Environnement avant mouvement.

Après une translation suivant Ox de l'obstacle 3 la nouvelle configuration est représentée par la figure 3.19 b.

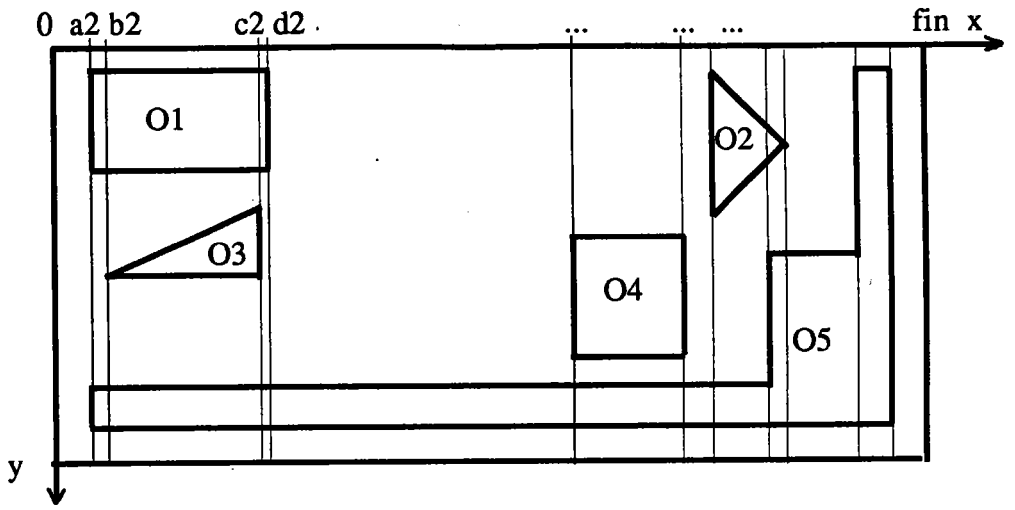
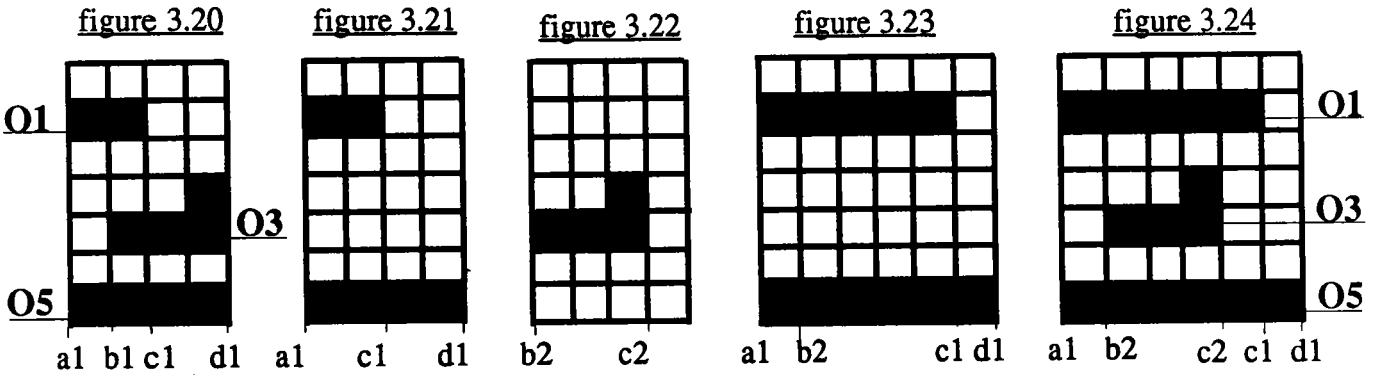


fig. 3.19 b : Environnement après mouvement

En considérant les abscisses extrêmes des objets ayant évolué, nous délimitons la zone concernée par le changement (dont le modèle est donné figure 3.20), zone verticale comprise entre les abscisses a1 et d1.

La modification dynamique du modèle statique se fait selon 4 étapes :

- suppression de l'objet mobile de la grille (fig. 3.21),
- modélisation séparée de l'objet mobile (fig. 3.22),
- création ou suppression de colonnes (fig. 3.23),
- insertion de l'objet mobile dans la grille (fig. 3.24).



figures 3.20 à 3.24 : Codage de la zone concernée par la modification

3.3.3) Méthodologie adoptée.

La méthodologie utilisée pour la modification dynamique du nombre de colonnes repose sur 4 opérations.

a) Création d'une liste de fusion avant mouvement.

Considérons l'environnement représenté par la figure 3.18. Créons une liste, LISTE1_X, associée à l'axe 0x de notre plan et comportant la liste ordonnée des sommets.

LISTE1_X={0, a1, b1, c1, ..., fin}.

Calculons pour chaque bande verticale le partitionnement nécessaire. Soit PARTITION1_X la liste contenant ces valeurs.

PARTITION1_X={m0, m1, m2, ..., mi, ..., mm} avec mi la partition calculée entre la cote LISTE1_X[i] et LISTE1_X[i+1].

Nous réitérons l'opération précédente pour l'axe 0y et nous obtenons LISTE1_Y et PARTITION1_Y.

b) Création d'une liste de fusions après mouvement.

Considérons maintenant l'environnement représenté par la figure 3.19.

Nous établissons de nouveau la liste ordonnée des sommets (LISTE2_X) et la liste des partitions correspondantes (PARTITION2_X). Cette dernière est obtenue par modification de la liste PARTITION1_X dans la zone concernée par les changements.

Nous réitérons les opérations précédentes pour l'axe 0y et nous obtenons LISTE2_Y et PARTITION2_Y.

c) Procédure création-suppression.

Cette procédure permet de déterminer dans la zone concernée par le changement si des créations ou des suppressions de colonnes sont nécessaires. Cette procédure s'appuie sur la comparaison successive des abscisses des listes LISTE1_X et LISTE2_X et des partitionnements correspondants.

Algorithme.

1)Initialisation i=j=1

2)Si (LISTE1_X[i]>LISTE2_X[j])

{Création de PARTITION2_X[j] colonnes à la cote LISTE2_X[j];

j++;

saut en 7;}

3)Si (LISTE1_X[i]<LISTE2_X[j])

{Suppression de PARTITION2_X[i] colonnes à la cote LISTE1_X[i];

i++;

saut en 7.}

4)Si (PARTITION1_X[i]>PARTITION2_X[j])

{Suppression de (PARTITION1_X[i]-PARTITION2_X[j]) colonnes à la cote

LISTE1_X[i]

saut en 6.}

5) Si (PARTITION1_X[i]<PARTITION2_X[j])

{Création de (PARTITION2_X[j]-PARTITION1_X[i]) colonnes à la cote

LISTE1_X[i].

6)i++; j++;

7)Si ((LISTE1_X=fin) & (LISTE2_X=fin)) saut en 8

saut en 2

8)FIN

Cette procédure génère 2 listes

- Création [.] contenant les colonnes à créer,
- Suppression [.] contenant les colonnes à supprimer.

d) Simplification.

Sur le modèle statique la procédure de simplification a pour but de réduire, voire d'annuler, les créations et suppressions de colonnes se produisant dans le même intervalle.Bien entendu, cette opération reste limitée à la zone perturbée par le déplacement du mobile.

Algorithme :

1)i=0;

2)j=0; k=0;

3)Si (LISTE1_X[i]_Création[j]_LISTE1_X[i+1]) aller en 8

4)Si (Création[j+1]=fin) aller en 6

5)j++; saut en 3;

6)i++;

Si (LISTE1_X[i+1]=fin) FIN

7)Saut en 2;

8)Si (LISTE1_X[i]_Suppression[k]LISTE1_X[i+1] aller en 11

9)Si (Suppression[k]=fin) saut en 4

10)k++; saut en 8;

11)Création[j]=0; Suppression[k]=0; saut en 4

Le raisonnement sera identique pour un modèle subissant une variation du maillage horizontal.

3.3.4) Apprentissage de l'environnement.

3.3.4.1) Principe.

Cette procédure de modélisation dynamique peut s'utiliser pour l'apprentissage d'un environnement. La fonction apprentissage est basée sur la constitution d'une base de connaissance de l'environnement. Chaque obstacle y est repéré par ses sommets dont les coordonnées cartésiennes se suivent dans le sens du balayage effectué par le système de perception (télémètre, par exemple). Cette base de connaissance de l'environnement va être complétée au fur et à mesure de l'avancée du robot dans son univers inconnu. Au cours de son évolution, le robot mobile va rencontrer deux types d'obstacles :

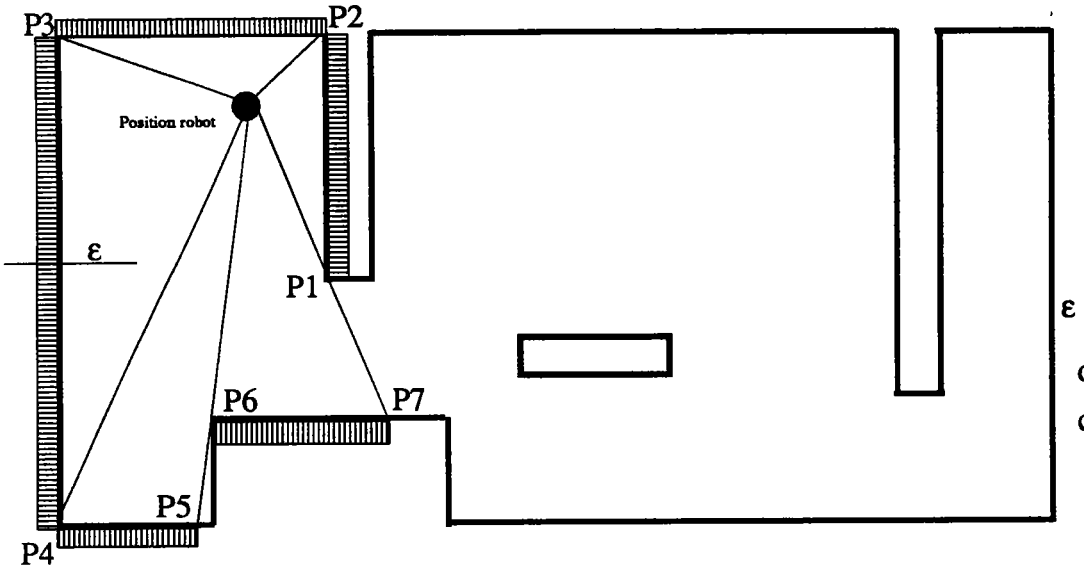
- les obstacles entièrement définis (dont tous les sommets sont connus) : type 1,
- les obstacles partiellement définis : type 2.

Cette base de connaissance partielle va servir à l'élaboration du modèle géométrique partiel indispensable au calcul de la trajectoire.

3.3.4.2) Application.

Nous allons appliquer la méthode d'apprentissage à un robot mobile découvrant et mémorisant l'environnement grâce à plusieurs scrutations.

1ère scrutation :



ε : épaisseur affectée aux obstacles partiellement connus.

fig. 3.25 : Première scrutation.

Exemple d'environnement inspiré de [CRO 83]

La base de connaissance correspondant à la figure 3.25 est :

obstacles type 1 :

obstacles type 2 : P1, P2, P3, P4, P5

P6,P7

2ème scrutation:

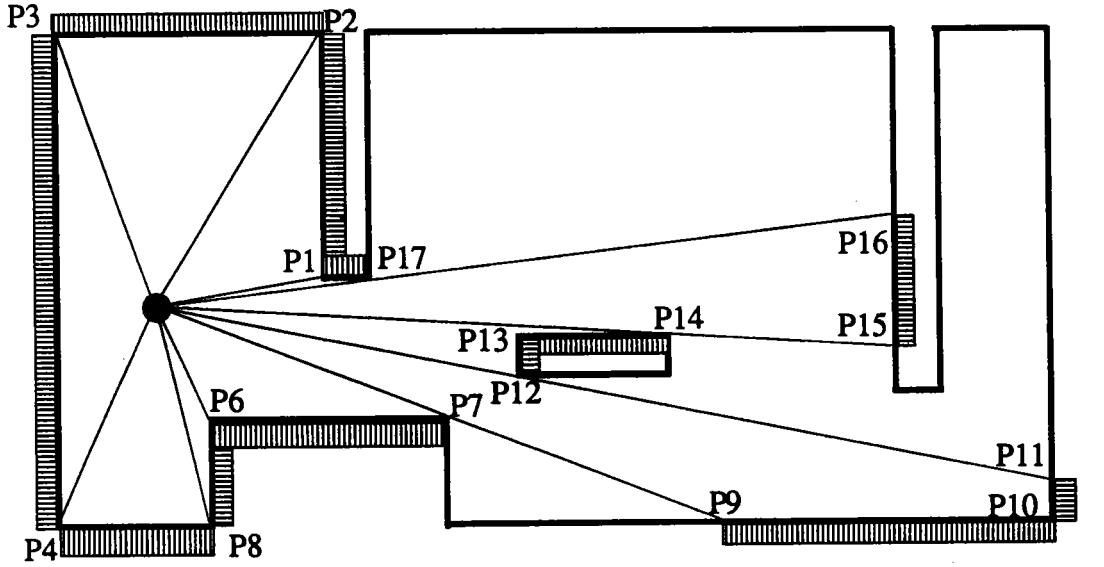


fig. 3.26 : Deuxième scrutation.

La base de connaissance correspondant à la figure 3.26 est :

obstacles type 1 :

obstacles type 2 : P17, P1, P2, P3, P4, P8, P6, P7

P9, P10, P11

P12, P13, P14

P15, P16

3ème scrutation :

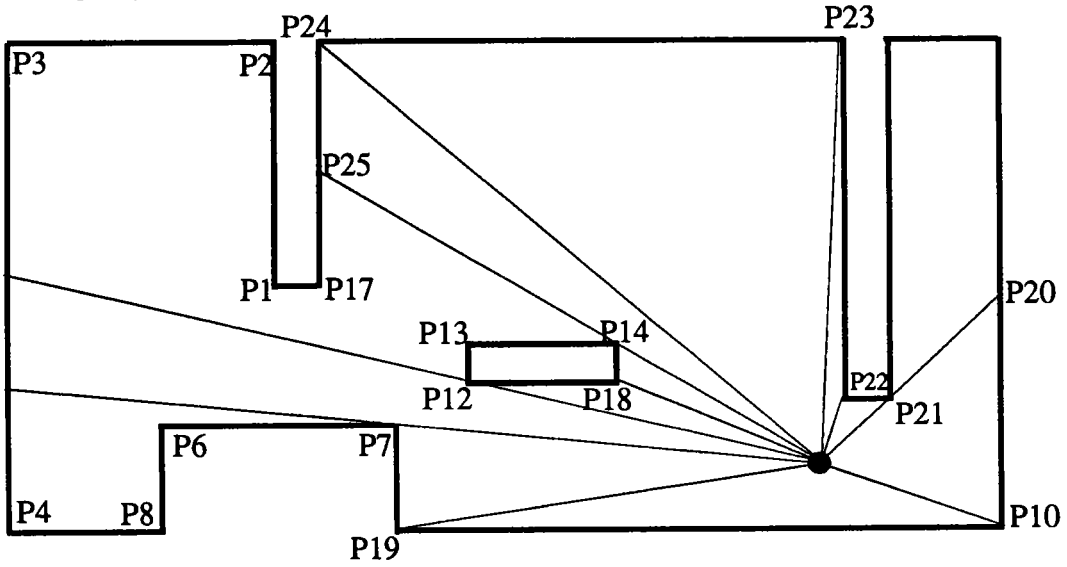


fig. 3.27 : Troisième scrutation

La base de connaissance correspondant à la figure 3.27

obstacles type 1 : P12, P13, P14, P18, P12

obstacles type 2 :P17, P1, P2, P3, P4, P8, P6, P7,P19, P10, P20

P21, P22, P23, P24, P25

Au cours de son évolution la connaissance de nouveaux points servira à enrichir la base de connaissance et aussi à transformer les obstacles de type 2 en obstacles de type 1. L'élaboration et la structuration de cette base de connaissance devra s'effectuer de telle manière qu'il n'y ait ni redondance, ni incompatibilité.

Modélisation par grille.

Une première modélisation, correspondant à la première scrutation, est effectuée (fig. 3.28). Les obstacles de type 1 (donc entièrement connus) sont modélisés par la procédure vue au paragraphe 3.22. Les obstacles de type 2 sont modélisés après avoir doté les droites, contenues dans la base de connaissance, d'une épaisseur ε en vue de les transformer en surface (voir figure 3.25).

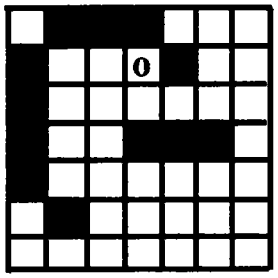


fig. 3.28 : Modélisation correspondant à la fig. 3.25

o : position du robot

Modification de la modélisation.

Après un déplacement, le système de perception effectue un nouvel enregistrement sur son environnement et complète la base de connaissance en conséquence. Une remodélisation de l'environnement entier serait possible et très rapide dans le cas d'un univers simple, mais nécessiterait trop de temps dans le cas d'un environnement complexe.

Dans ce cas, la modélisation dynamique est donc recommandée car les modifications de maillage restent localisées à la zone associée à l'évolution.

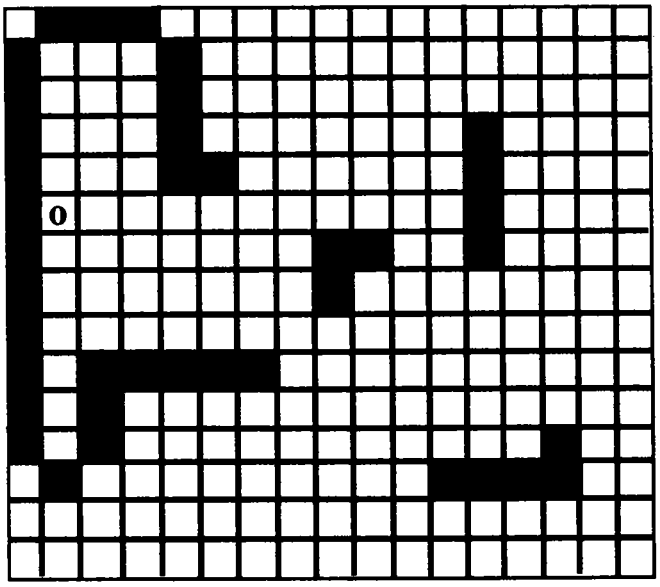


fig. 3.29 : Modélisation de la figure 3.26

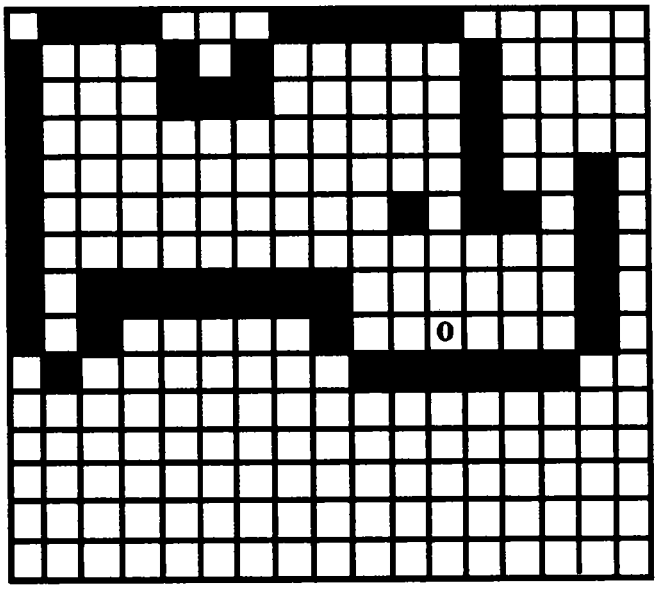


fig. 3.30 : Modélisation de la figure 3.27

3.3.5) Conclusion.

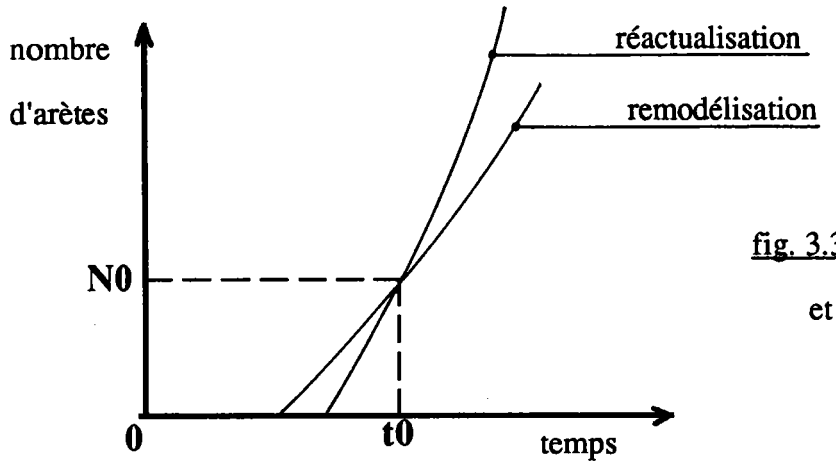


fig. 3.31 : Temps de remise à jour et de réactualisation.

On constate, par expérience, qu'il existe un nombre critique N0 d'arêtes au-dessus duquel une réactualisation est plus rapide qu'une modélisation complète mais qu'inversement en-dessous duquel une modélisation est préférable (parce que plus rapide).

Ce principe de réactualisation d'un modèle pourra s'adapter à l'évolution d'un robot dans un univers dont il découvrira et mémorisera la géométrie.

3.4) Modélisation statique d'un environnement $2D^{1/2}$

3.4.1) Principe.

Le déplacement d'un robot dans un univers complexe, constitué de surfaces planes, horizontales et inclinées, nécessite une représentation adaptée. Nous nous plaçons ici dans un univers restreint à $2D^{1/2}$ composé uniquement de formes polyédrales. Le principe de cette méthode consiste à décomposer le modèle en surfaces élémentaires, à calculer la représentation de chacune d'elle séparément, puis à rassembler l'ensemble en un minimum de grilles finales.

3.4.2) Obtention du modèle.

Soit l'exemple de la figure 3.32

3.4.2.1) Décomposition du modèle en surface.

La première opération consiste à décomposer notre univers en 2 types de surfaces :

- les plans (surfaces planes ayant une altitude constante).
- les transitions (surfaces planes ayant une altitude variable).

Nous obtenons, pour l'exemple de la figure 3.32

- 2 plans P1 et P2
- 2 transitions T1 et T2.

3.4.2.2) Détermination des accessibilités des surfaces.

Le passage éventuel entre plan et transition, ou entre transitions, dépend de l'inclinaison relative et absolue de ces surfaces.

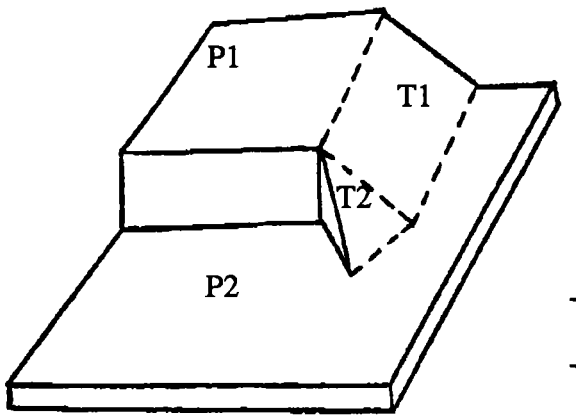


fig. 3.32 : Exemple de décomposition

– – – – frontières franchissables.

——— frontières non franchissables.

Convention.

$\sigma i, j = 1$ s'il y a possibilité d'évoluer entre le plan i et la transition j ($\sigma i, j = 0$ dans le cas contraire).

$\sigma ij = 1$ s'il y a possibilité d'évoluer entre la transition i et la transition j ($\sigma ij = 0$ dans le cas contraire) avec i<>j et $\sigma ij = \sigma ji$

Dans le cas de la figure 3.32 nous avons

$\sigma 1, 1 = 1 \quad \sigma 1, 2 = 0 \quad \sigma 2, 1 = 1 \quad \sigma 2, 2 = 1$

et $\sigma 12 = \sigma 21 = 1$

3.4.2.3) Modélisation des surfaces.

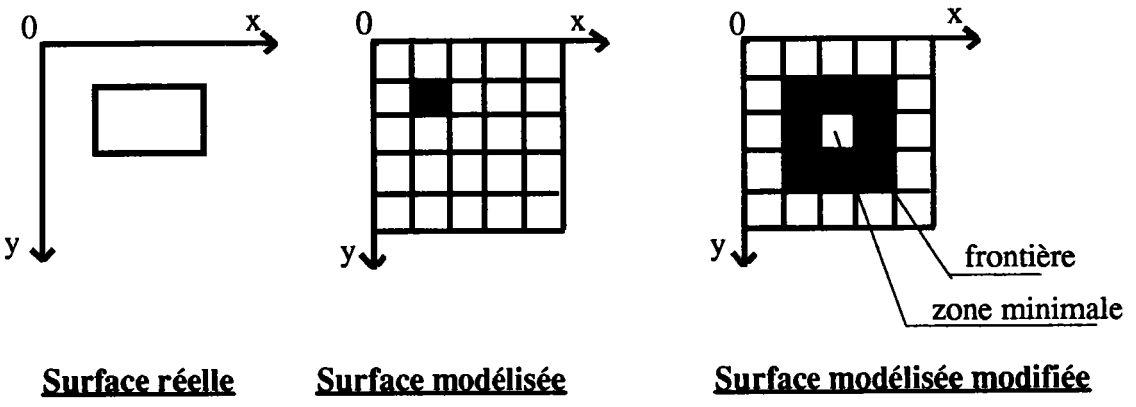
Chaque surface décomposée est modélisée par la méthode vue au paragraphe 3.2.2. Néanmoins les modèles obtenus doivent être conformes à certaines règles :

a) Règle n°1

Chaque surface doit être représentée par une zone minimale. Les frontières représentent les limites de notre surface que l'on pourra franchir ou non suivant les caractéristiques de la surface adjacente.

Dans les deux exemples ci-dessous, nous agrandissons notre surface afin que compte tenu de «l'épaisseur» de la frontière, il reste une zone minimale pour représenter la surface réelle

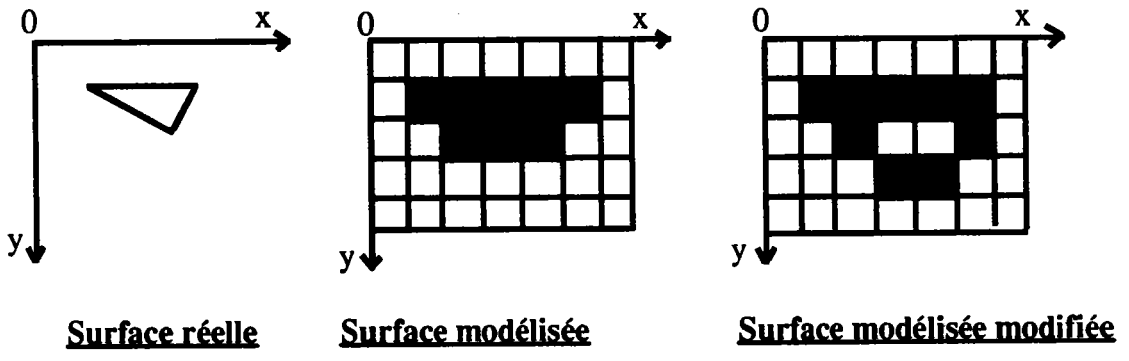
Exemple n° 1 (figure 3.33)



Surface réelle **Surface modélisée** **Surface modélisée modifiée**

fig. 3.33 : Exemple n°1

Exemple n° 2 (figure 3.34)



Surface réelle **Surface modélisée** **Surface modélisée modifiée**

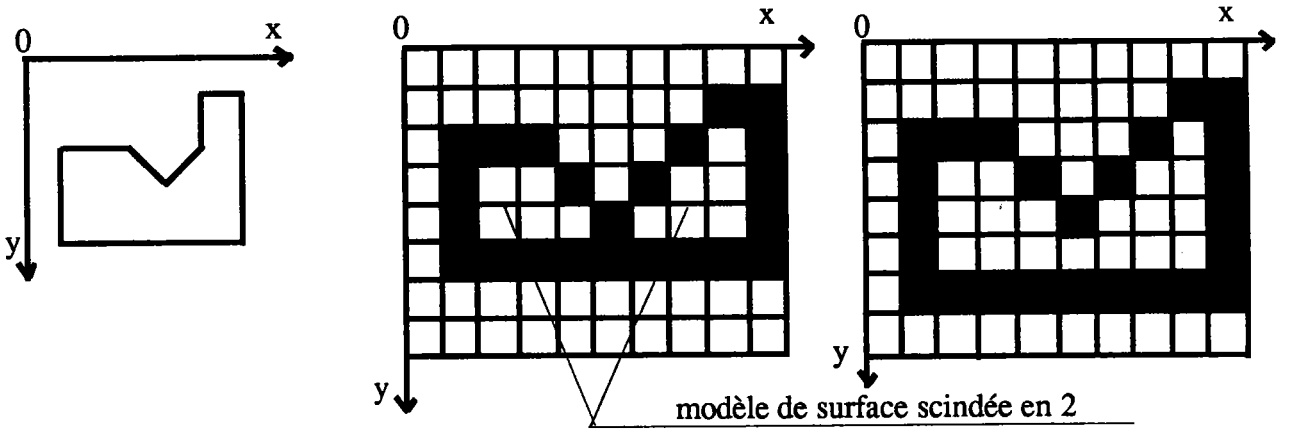
fig. 3.34 : Exemple n°2

b) Règle n°2

La zone minimale de représentation de la surface ne doit pas présenter de discontinuités.

L'exemple de la figure 3.35 représente le plan P2 de la figure 3.32. Nous devons nous assurer que le robot peut librement accéder à toutes les cellules constituant le plan.

Exemple n° 3 (figure 3.35)



Surface réelle **Surface modélisée** **Surface modélisée modifiée**

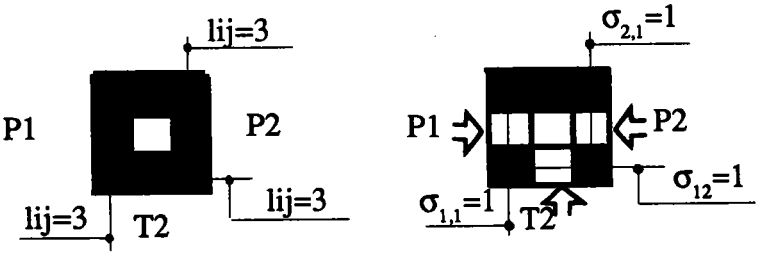
3.4.2.4) Représentation des accessibilités.

Il s'agit «d'ouvrir» les frontières qui sont franchissables.

Soit lij le nombre de cellules représentant la frontière formée par l'intersection de la transition j et du plan i (ou transition i).

Règle : on montre l'accessibilité d'une surface en rendant franchissables les (li, j - 2) cellules centrales constituant la frontière.

Dans l'exemple de la figure 3.32 la transition T1 devient



Modélisation initiale **Modélisation finale**

fig. 3.36 : Représentation de la transition T1

On peut ainsi accéder à la transition T1 depuis les plans P1, P2 et la transition T2 par l'ouverture de la cellule centrale (lij - 2 = 3 - 2 = 1) correspondante.

Le raisonnement est identique pour une arête non parallèle à notre référentiel.

3.4.2.5) Superposition.

L'opération finale consiste à ranger les différentes surfaces modélisées dans un certain nombre de grilles afin de réduire le modèle final. Cette superposition s'effectue par des opérations logiques sur les p mots de q bits représentant l'état des différentes cellules des grilles, à condition qu'il n'y ait pas enchevêtrement des surfaces. Cette opération de superposition nous permet :

- un gain de place mémoire,
- d'être plus rapide au niveau recherche de trajectoires.

Plusieurs méthodes de rangement sont envisageables suivant le but recherché

- classement par type de surface (plan ou transition),
- classement mixte. Celui-ci nous donne pour la figure 3.32 le modèle de la figure 3.37.

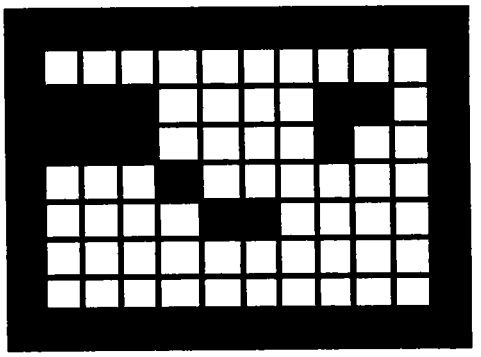


fig. 3.37 : Classement mixte pour l'exemple de la figure 3.32

3.4.3) Conclusion.

Si l'environnement ne comporte pas plusieurs plans ou transitions superposés (fig. 3.38a), il peut être modélisé sur une seule grille. Dans le cas contraire (exemple fig. 3.38b), plusieurs grilles sont nécessaires. Cette modélisation peut convenir, par exemple, pour modéliser un magasin (ou un atelier) sur plusieurs niveaux comportant des rampes d'accès utilisables par des robots mobiles.

Sur les figures 3.38 (a), (b) et (c), des modèles géométriques d'environnement $2D^{1/2}$ sont donnés ainsi que les modèles de calcul correspondant (fig. 3.38 (d), (e) et (f)) et les temps de modélisation.

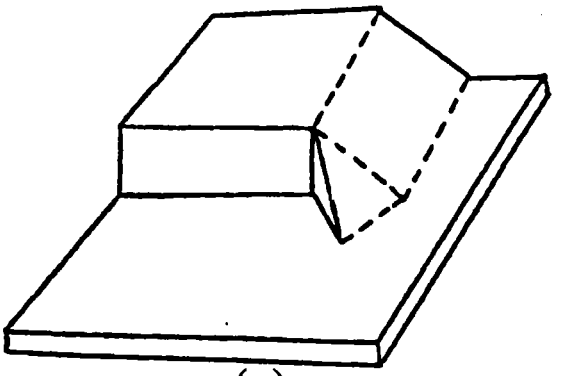
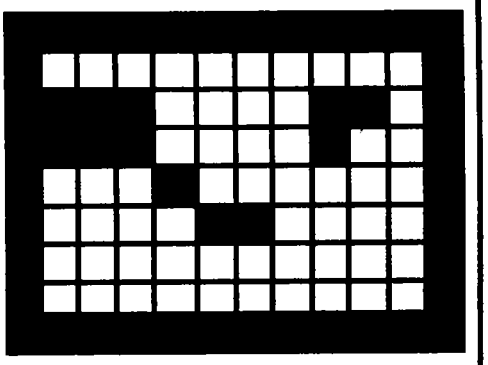
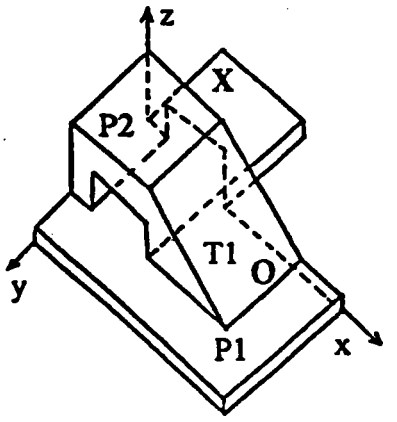
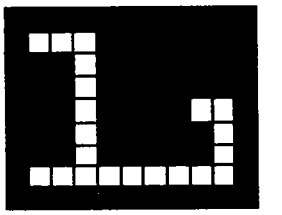
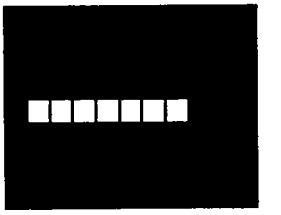
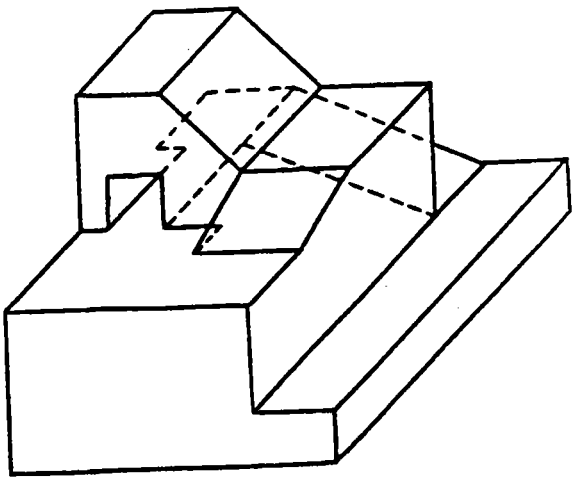
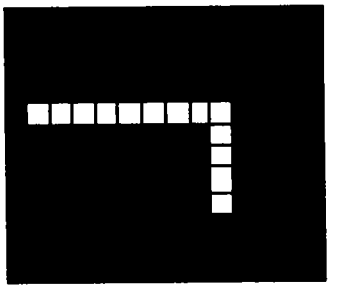
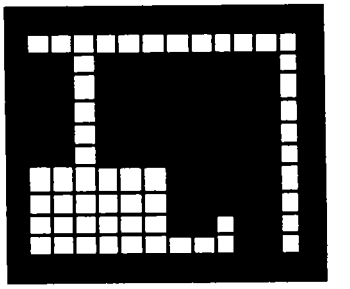
Modèle géométrique	Modèle de calcul	Temps
 (a)	 (d)	0,13s
 (b)	  (e)	0,19s
 (c)	  (f)	0,26s

fig. 3.38 : Exemples de modélisation

3.5) Recherche de trajectoire 2D.

3.5.1) Principes.

3.5.1.1) Eléments de base.

La recherche de chemin constitue l'étape de planification du déplacement du mobile. La méthode se base sur 4 éléments : le modèle M(p, q), une liste $\pi(n)$ des cellules successives que composent le chemin, un code C(n) d'autorisation de déplacement et une fonction F qui assure l'évaluation des priorités de déplacement. Le modèle de grille décrit par le vecteur M(p, q) représente la carte d'admissibilité de l'environnement. Chaque bit q d'une composante p du vecteur représente une cellule. Toutes les cellules ont une importance dimensionnelle et fonctionnelle égale. La liste $\pi(n)$ définit l'ensemble des cellules successives de la cellule source $\pi(1)$ dans laquelle se situe le mobile, à la cellule but dans laquelle se trouve le point destination. A chaque élément de la liste $\pi(n)$ est associé un code C(n) qui précise les possibilités de mouvement d'une cellule vers les cellules voisines. Huit mouvements sont codés en considérant toujours la grille à maillage orthogonal:

- les déplacements de base : haut, bas, droite, gauche (fig. 3.39a).
- les associations de déplacements de base (fig. 3.39b).

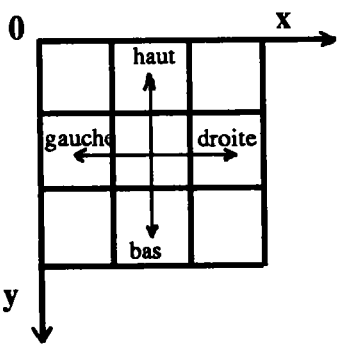


fig. 3.39 a : Déplacements de base

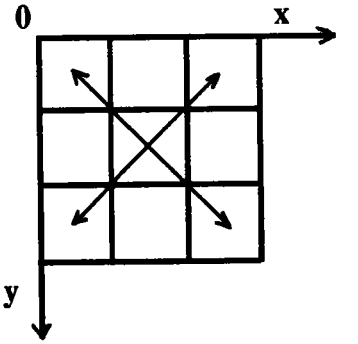


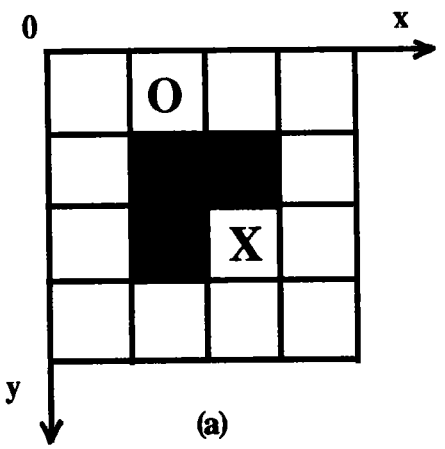
fig. 3.39 b : Association des déplacements de base

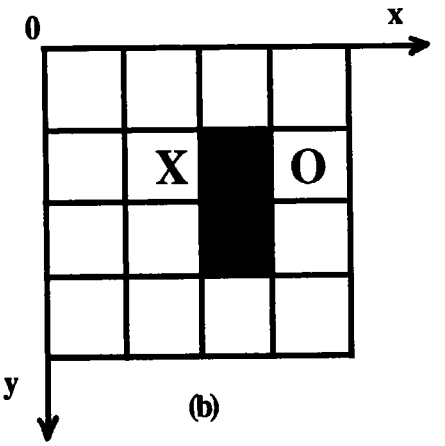
Le code C(n) établit la carte de déplacements autorisés de la cellule n faisant partie du chemin. La fonction d'évaluation des priorités précise, en cas de litiges, le choix pour lequel il faut opter. La fonction est à définir par le programmeur en vertu des objectifs et du type de problèmes à résoudre.

3.5.1.2) Critères de rapprochements.

Soit M(i, j) la cellule dans laquelle est situé le mobile et soit M(v, w) la cellule but. La cellule qui suit M(i, j) est définie selon le critère de rapprochement le plus rapide du but ou selon la différence entre les indices i, v et j, w. D'où le tableau ci-dessous :

cas	(t0,t1)	(i,j)	Exemple
v>i	t0=1		fig. 3.40 a
v<i	t0=-1	i=i+t0	fig.3.40 b
v=i	t0=0		fig. 3.40 c
w>j	t1=1		fig.3.40 a
w<j	t1=-1	j=j+t1	fig.3.40 c
w=j	t1=0		fig.3.40 b





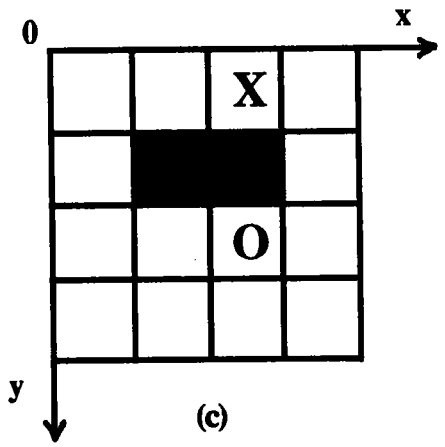


fig. 3.40 : Exemples pour le calcul de t0 et t1

O: Source

X: But

: Cellule accessible

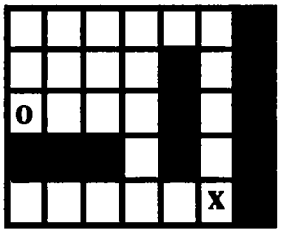
: Cellule obstacle

3.5.1.3) Exemple de calcul.

La recherche des cellules faisant partie du chemin s'effectue par des opérations booléennes sur les composantes du vecteur M(p, q). Prenons un cas concret pour expliquer la méthode. Soit l'environnement ci-dessous :

o : Pr

x : Pb



1 1 1 1 1 1 0
1 1 1 1 0 1 0
1 1 1 1 0 1 0
0 0 0 1 0 1 0
1 1 1 1 1 1 0

Notons la position actuelle du robot (Pr) dans la ligne 3, colonne 1 et le but (Pb) dans la colonne 6, ligne 5 :

Pr : 1 0 0 0 0 0 0 ligne 3

Pb : 0 0 0 0 0 1 0 ligne 5

Calculons un masque à partir de la position du robot et du calcul de t0 :

t0 = 1: masque = Pr OR (Pr /2)

t0 = -1: masque = Pr OR (Pr x 2)

La multiplication et la division par deux représentent un décalage respectivement à gauche et à droite du mot binaire position robot. Dans le cas de l'exemple:

masque = 1 1 0 0 0 0 0

L'étape suivante consiste à effectuer les possiblités de mouvements sur la ligne actuelle v et sur la ligne v+t1. D'où l'opération suivante de masquage :

A = masque AND M(v) = 1 1 0 0 0 0 0

et

B = masque AND M(v+t1) = 0 0 0 0 0 0 0

Dans notre cas le choix de la cellule suivante est donné par A, la seule possible puisque B=0. Dans le cas général, plusieurs cas se présentent.

Calculons les trois états suivants :

S0 = A AND NOT(Pr)

S1 = A AND B

S2 = S1 AND (S1>>1) si t0=1 (décalage à droite de un bit)

ou S2 = S1 AND (S1<<1) si t0=-1 (décalage à gauche de un bit)

puis

```
if (S2 !=0)  Pr = S2
else
    if (S1 !=0)  Pr = S1
    else
        if (S0 !=0)  Pr = S0
        else déplacement impossible
```

L'ordre de test de S0 et S1 peut être imposé. Il détermine la priorité du mouvement horizontal sur le mouvement vertical ou inversement. Dès que le choix est validé, la cellule contenant le robot est interdite afin de ne plus pouvoir être atteinte une seconde fois.

Cette interdiction est notifiée en rendant inaccessible sur le modèle la cellule contenant le robot.

3.5.2) L'algorithme SEARCHPATH.

L'algorithme SEARCHPATH ($\pi 0$, πn) déterminant la liste des cellules faisant partie de la trajectoire de la cellule source $\pi 0$ à la cellule but πn est donné ci-dessous.

3.5.2.1) Principes de l'algorithme.

1. Créer une liste des cellules formant les chemins ayant un élément $\pi(1)$ qui est la position actuelle du robot. Le code C(1) associé autorise tout déplacement autour de $\pi(1)$. Initialiser i à 1.
2. Faire M(p, q) = 0 avec p et q désignant le bit du vecteur position actuelle du robot.
3. Calculer les valeurs t0 et t1 déterminant le sens de progression possible en $\pi(i)$ selon les possibilités définies par C(i) et selon la fonction de priorité F.

Si aucune possibilité, alors saut en 10.

4.Modifier le code C(i) en fonction du choix de t0 et t1.

5.Déterminer le masque.

6.Calculer la nouvelle position du robot. Si la nouvelle position n'existe pas, alors saut en 3.

7.i=i+1. Ajouter à la liste π(i) la nouvelle position du robot et le code associé C(i) autorisant tout déplacement autour de π(i).

8.Si position robot est égale au but, alors FIN.

9.Saut en 2.

10.i=i-1; Si i=-1, alors FIN : pas de déplacement possible.

11.Saut en 3.

3.5.2.2) Opération de backtracking.

Le mobile, ayant comme stratégie conductrice de se rapprocher constamment du but, peut, dans certains cas, se retrouver bloqué. C'est le cas, par exemple, de la figure 3.41.

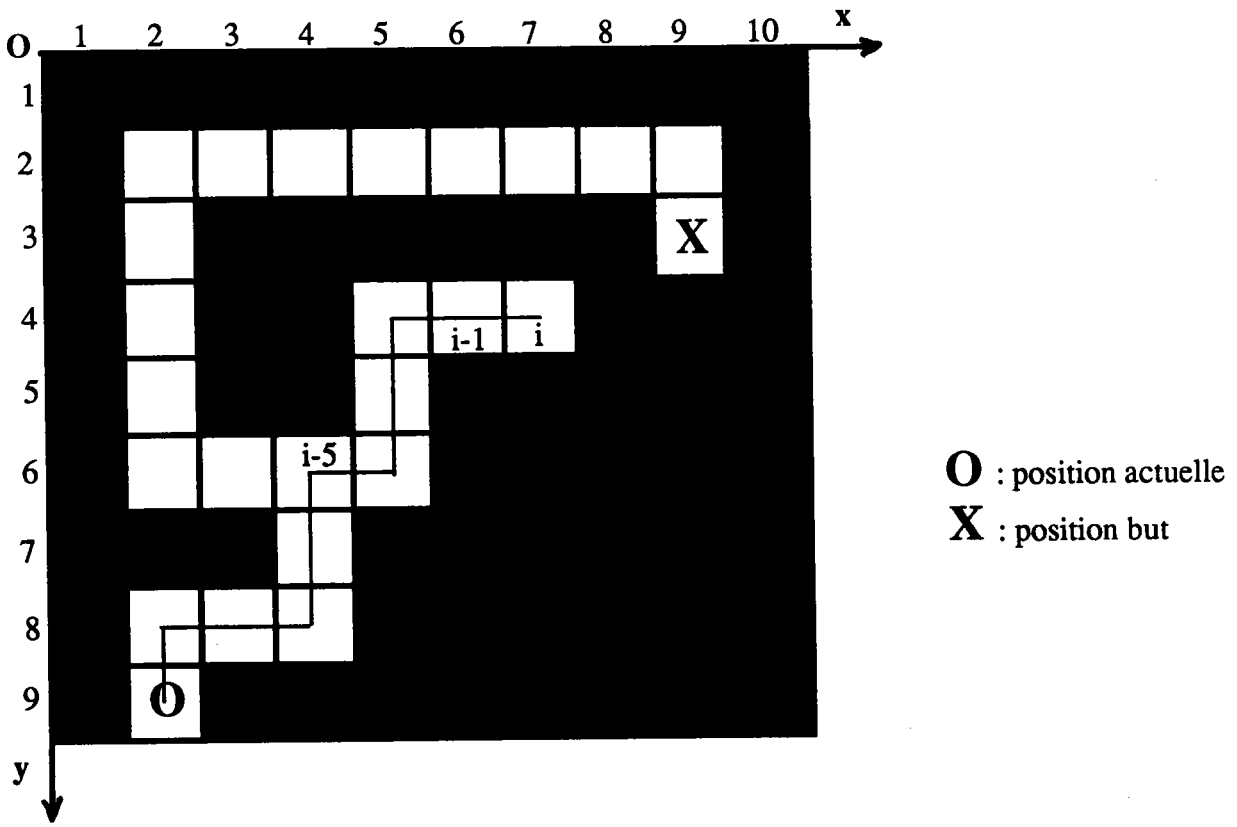


fig. 3.41 : Opération de backtracking

Pour remédier à cette situation, l'algorithme utilise une procédure de backtracking visant à exploiter les mouvements encore exploitables affectés aux cellules traversées.

Voyant qu'aucun mouvement n'est plus possible à partir de la cellule i (C(i)=0), SEARCHPATH reviendra sur les cellules traversées (successivement i-1, i-2, i-3,...) jusqu'au moment où de nouvelles possibilités de mouvement lui sont offertes.

Dans l'exemple de la figure 3.41, la case (i-5) offrira une nouvelle possibilité de mouvement dans la direction (x-).

Tous les mouvements encore possibles à partir des cellules utilisées sont inscrits dans le tableau C, à l'emplacement C(i).

La recherche de trajectoire dans l'exemple de la figure 3.41 peut se représenter sur la figure 3.42.

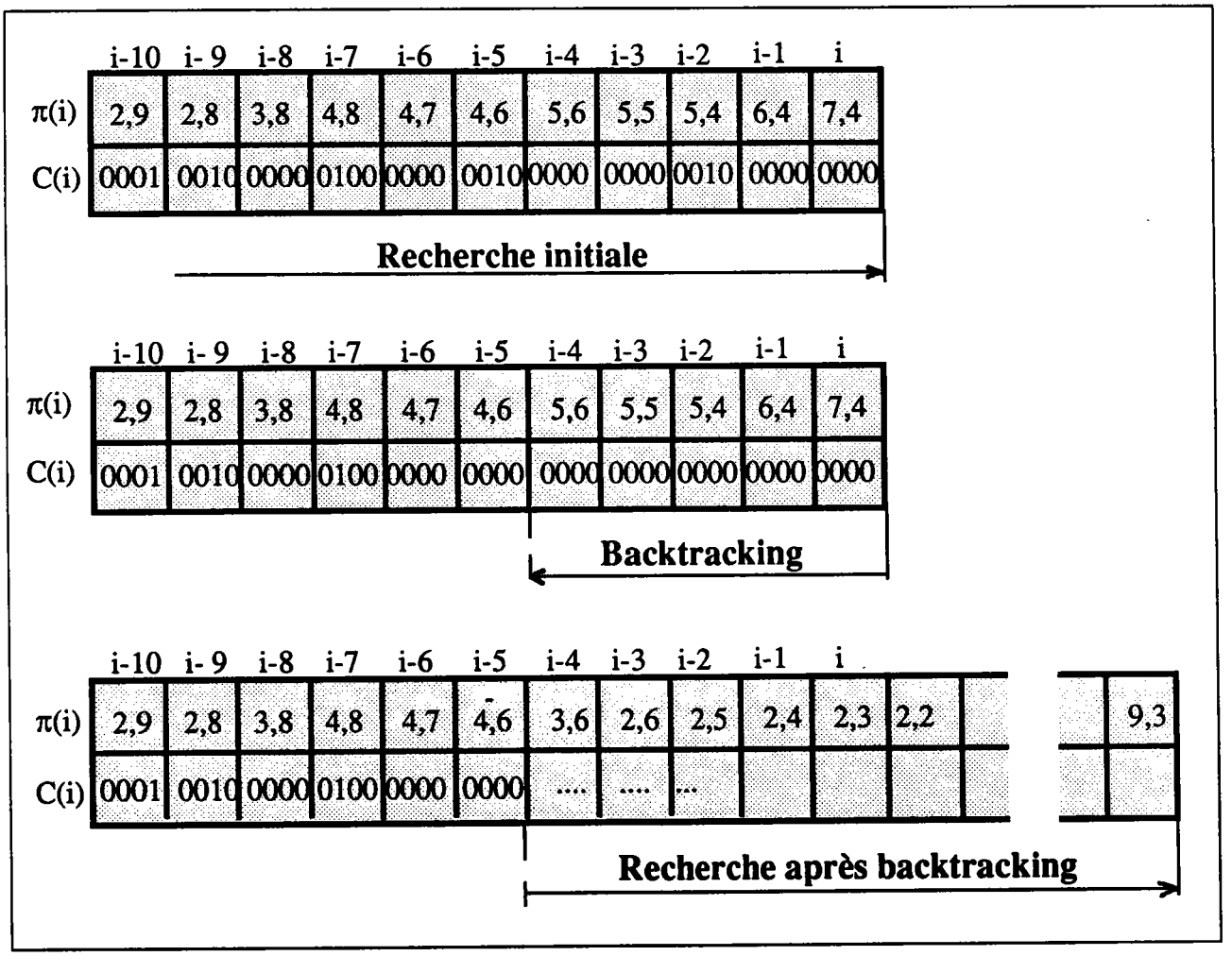


fig. 3.42 : Recherche de trajectoire

avec C(i) = (bit1, bit2, bit3, bit4) :

bit1 = 1 si mouvement suivant (x+) non encore exploité.

bit1 = 0 dans le cas contraire.

bit2 = 1 si mouvement suivant (y+) non encore exploité.

bit3 = 1 si mouvement suivant (x-) non encore exploité.

bit4 = 1 si mouvement suivant (y-) non encore exploité.

avec $\pi(i)$ = (n° de la colonne, n° du mot).

3.5.3) Optimisation de la trajectoire.

3.5.3.1) Algorithme d'optimisation.

Le résultat de SEARCHPATH ($\pi 0$, πn) détermine une trajectoire qui n'est pas toujours intéressante au point de vue de la rapidité de progression vers le but. Le critère de rapprochement le plus rapide utilisé par SEARCHPATH induit dans certains cas un allongement de la trajectoire comme le montre la figure 3.43. Nous proposons une méthode dérivée de l'algorithme A* qui permet de déterminer une trajectoire intégrant la prévision de la présence d'obstacles. La première étape consiste à trouver le chemin par SEARCHPATH ($\pi 0$, πn). Ouvrons, à partir de la liste π, deux autres listes, l'une Gi regroupant les cellules de $\pi 0$ à πi et l'autre Hi regroupant les cellules de $\pi i+1$ à πn. Déterminons deux fonctions de coût Cg et Ch représentant respectivement le nombre de cellules Gi et Hi. L'algorithme d'optimisation consiste à calculer pour tout i avec $0<i<n$, le coût minimum $C^* = Cg + Ch$.

Nous obtenons l'algorithme ci-dessous :

Principes de l'algorithme :

```
i=s;
Gi=(π0,...., πi);Hi=(πi+1,.....,πn); coût=COST(G);
WHILE i<n
{
        T=SEARCHPATH(π0, πi);
        if (COST(Gi)<=coût)  Gi+s=(πi+1,....,πi+s);

                else   Gi+s=(Gi, πi+1, ....., πi+s);
```

Hi+s=(pi+s+1, ..., pn);

i=i+s; coût=COST(Gi);

}

avec COST(Gi)= Cg + Ch

s est un paramètre variant de 1 à n qui permet d'obtenir un compromis entre la rapidité de recherche (s grand) et la finesse (s petit) de la trajectoire. Dans une seconde étape recommençons le même processus en inversant le but et la source. Le choix de la trajectoire finale est réalisée en calculant le coût de la distance sur site réel.

3.5.3.2) Technique d'optimisation.

Observons sur un exemple simple le fonctionnement de cette optimisation. Connaissant la technique de rapprochement adoptée par SEARCHPATH (qui consiste à se diriger constamment vers le but), nous avons élaboré un environnement et un emplacement source-but qui allonge volontairement la trajectoire (fig. 3.43).

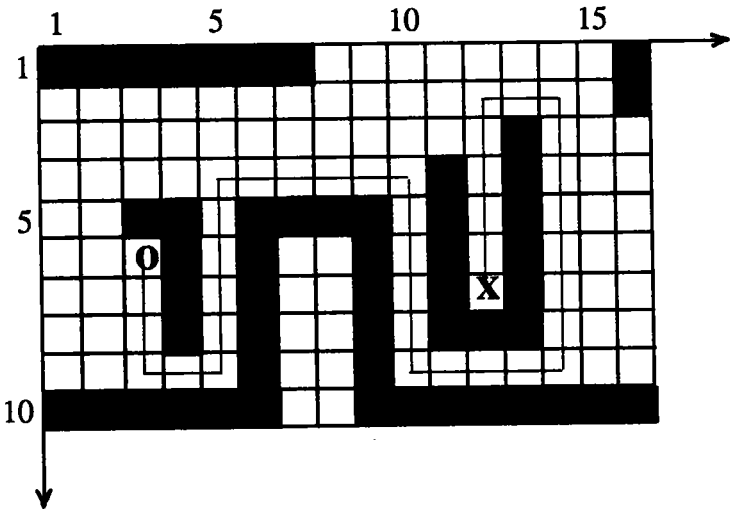


fig. 3.43 : Trajectoire initiale

O : position source

X : position but

L'optimisation consiste à se donner des sous-buts, distants de s cellules dans la liste π, que le mobile essaye de rallier grâce à l'algorithme SEARCHPATH. Si la nouvelle trajectoire de la source au sous-but est plus courte que l'ancienne, nous modifions la liste π.

Exemple :

Soit π = ((3, 6), (3, 7),(3, 8), (3, 9), (4, 9), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (6,4),....., (12, 7)) la liste des cellules faisant partie de la trajectoire.

Nous fixons arbitrairement le paramètre de saut s à 5.

a) le premier sous-but (π(s)) est la case (5, 9).

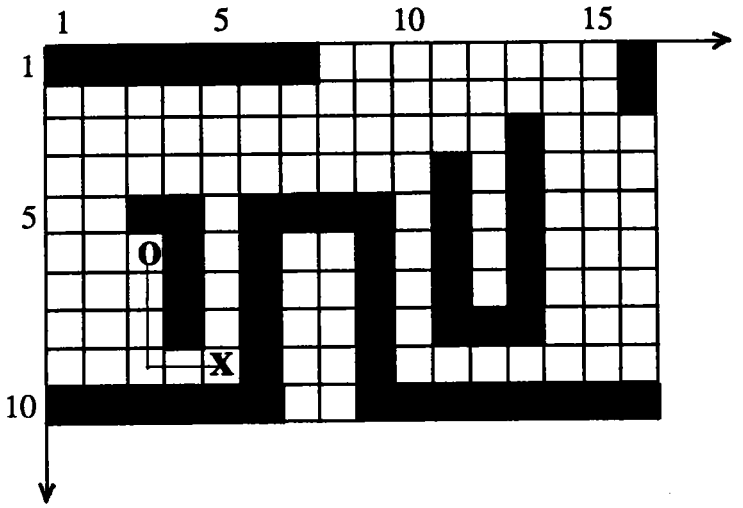


fig. 3.44 : Trajectoire partielle n°1

Nous appliquons SEARCHPATH entre les points π(0) et π(s) et nous constatons que la nouvelle trajectoire partielle obtenue (fig. 3.44) fait intégralement partie de l'ancienne : pas de modification de la trajectoire initiale.

b) le deuxième sous-but π(2s) est la case (5, 4).

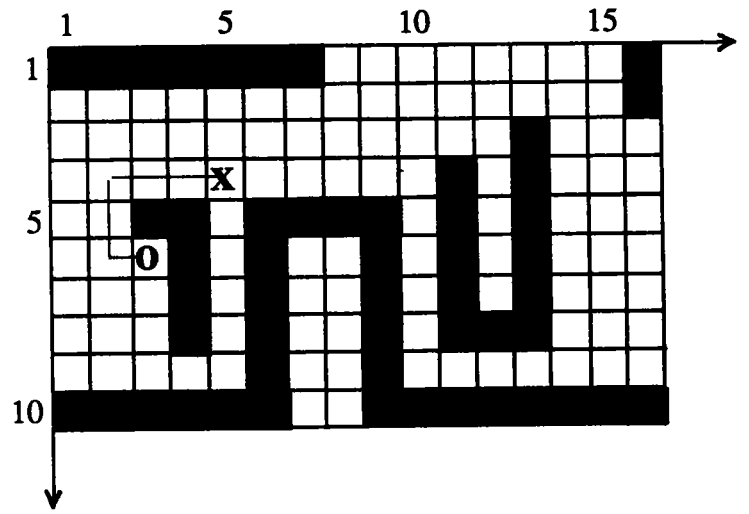


fig. 3.45 : Trajectoire partielle n°2

Nous appliquons SEARCHPATH entre les points $\pi(0)$ et $\pi(2s)$ et nous constatons que la nouvelle trajectoire obtenue (fig. 3.45) est plus courte que l'ancienne. La liste π, modifiée, devient π=((3, 6), (2, 6), (2, 5), (2, 4), (3, 4), (4, 4), (5, 4), 6, 4), ...(12, 7)).

c) La procédure d'optimisation terminée, nous obtenons la figure 3.46.

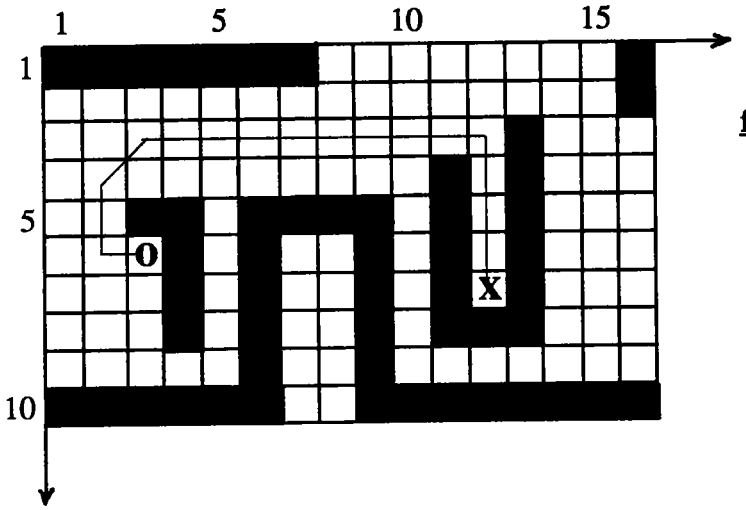


fig. 3.46 : Trajectoire après optimisation

O : position source

X : position but

Nous constatons que la liste p ne comporte plus que 17 cellules contre 38 auparavant.

3.5.3.3) Inversion source-but.

Lors de la procédure SEARCHPATH, il est possible que, vu la disposition des obstacles, l'inversion de la source et du but nous détermine une trajectoire plus rapide.

Cette possibilité ne peut s'effectuer que lorsqu'une modélisation globale de l'environnement a été réalisée. Le modèle de la figure 3.47 a été choisi dans le but d'illustrer l'utilité d'une telle procédure.

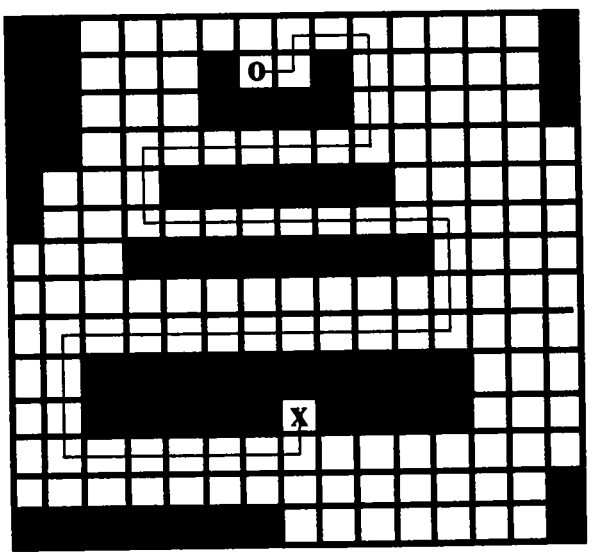


fig. 3.47 : Trajectoire initiale

O : position source

X : position but

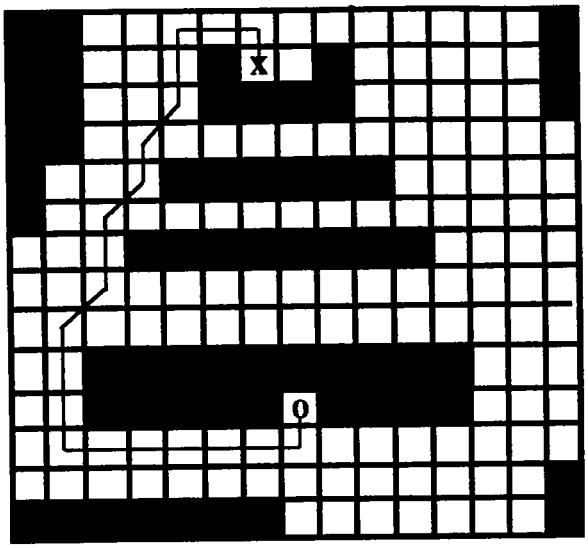


fig. 3.48 : Trajectoire après inversion

0 : position source

X : position but

La disposition des obstacles favorise cette inversion et nous donne des résultats meilleurs. Dans le cas de la figure 3.47, SEARCHPATH nous détermine une trajectoire comportant 46 cellules. En inversant source et but 21 cellules sont nécessaires pour rallier ces deux points(fig.3.48).

3.5.4) Complexité.

3.5.4.1) De l'algorithme SEARCHPATH.

Le nombre maximal d'opérations n1 nécessaire pour une itération de l'algorithme (avancée d'une cellule du robot) est indépendant de la taille de la grille. L'algorithme est parcouru tant que le but n'est pas atteint, soit, au maximum, m fois (avec m nombre de cellules accessibles). Le nombre d'opérations est donc, au plus, égal à n1xm. On en déduit la complexité maximale en 0(m).

3.5.4.2) De la procédure d'optimisation.

Soient m le nombre de cellules accessibles et s le paramètre de saut.

Soit k la partie entière de m/s. L'algorithme d'optimisation est itéré au maximum N=k+(m-k.s) fois. Si s vaut 1, N vaut m. A chacune des m itérations, l'algorithme SEARCHPATH, d'une complexité en 0(m), est exécuté. Nous avons donc une complexité maximale en $0(m^2)$.

3.5.5) Trajectoire sur site réel.

Considérons la trajectoire sur le modèle de calcul représenté par la figure 3.49. La trajectoire sur site réel est obtenue en reliant en ligne droite le centre des cellules figurant dans la liste π (fig. 3.50). Vu la stratégie adoptée par l'algorithme SEARCHPATH, nous avons la certitude que cette trajectoire est entièrement comprise dans l'espace libre.

Nous pouvons aussi «lisser» la trajectoire tout en l'obligeant à rester à l'intérieur des cellules. Cette technique de lissage consiste à anticiper la trajectoire en joignant le centre de la cellule $\pi i+1$ dès que l'on entre dans la cellule πi (fig. 3.51).

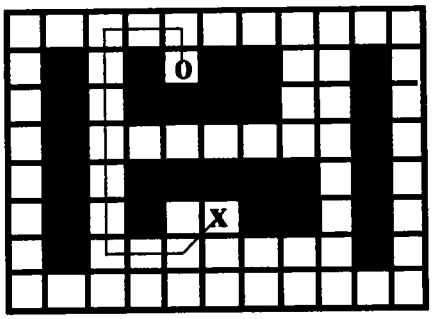


fig. 3.49 : Trajectoire sur le modèle de calcul

O : source

X : but

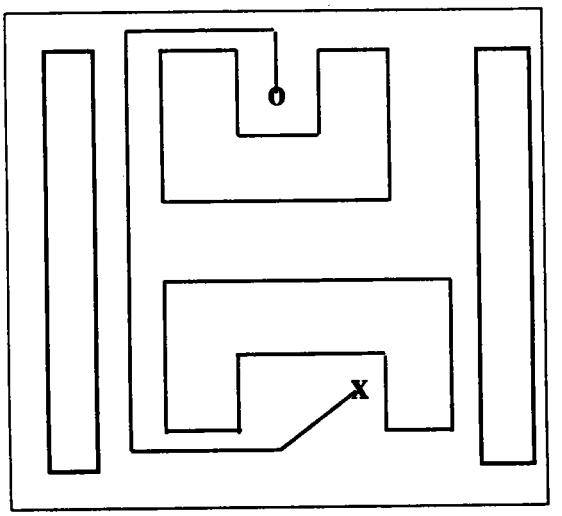


fig. 3.50 : Trajectoire simple sur site réel

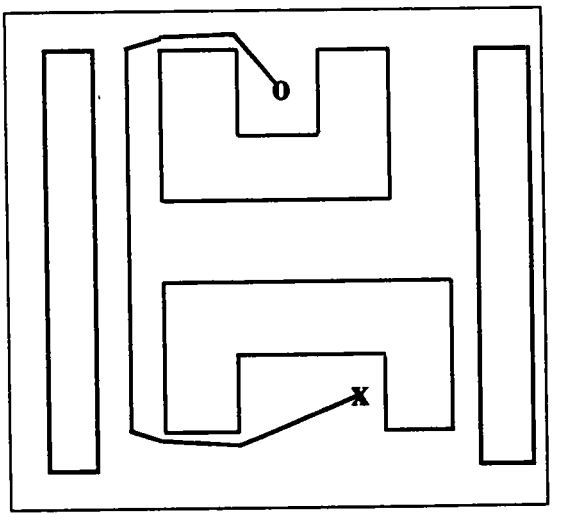


fig. 3.51 : Trajectoire lissée sur site réel

3.6) Recherche de trajectoire $2D^{1/2}$

3.6.1) Introduction.

Comme nous l'avons vu au chapitre modélisation, la majorité des univers $2D^{1/2}$ peuvent être représentés sur une grille unique. Dans ce cas, l'algorithme SEARCHPATH (chap. 2.3.5), suivi de la procédure d'optimisation, peut être appliqué. Dans le cas contraire il est nécessaire d'adopter une nouvelle méthodologie basée sur l'algorithme SEARCHPATH1.

3.6.2) Principes.

3.6.2.1) Eléments de base.

La méthode de planification de la trajectoire repose sur cinq éléments : le modèle M(k, p, q) (k grille(s), p mot(s) de q bits), une liste $\pi(n)$ des cellules successives occupées par le robot lors de son déplacement, un code C1(n) d'autorisation de se déplacer dans la même grille, un code C2(n) d'autorisation de changer de grille et une fonction F1(n) qui assure l'évaluation des diverses priorités de déplacement. Les k x p x q cellules du modèle ont des dimensions et un poids identiques. La détermination des cellules $\pi(n)$ faisant partie du chemin s'effectue grâce à des opérations booléennes sur les composantes du vecteur M(k, p, q). Le code C1(n), fonction de la nature du déplacement précédent et de la position de la cellule, nous donne le déplacement autorisé à partir de la cellule n. Le code C2(n), fonction du déplacement précédent et de la position relative entre la grille contenant la position actuelle et la grille but, nous donne le changement de grille autorisé à partir de la cellule n.

3.6.2.2) Critères de rapprochement.

Des critères de rapprochement Rx, Ry et Rz, fonction de la position relative cellule actuelle/cellule but, nous donne les grilles, lignes et colonnes de notre modèle sur lesquels doit s'effectuer l'opération de masquage.

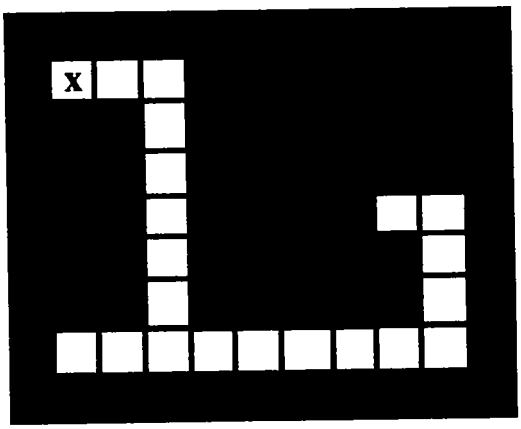
Soient $\pi(i, j, k)$ la cellule dans laquelle se trouve le mobile et $\pi(u, v, w)$ la cellule but. La cellule qui suit $\pi(i, j, k)$ est définie par le critère de rapprochement le plus rapide du but.

Cas	Rapprochement	
u>i	Rx = 1	i=i+Rx
u<i	Rx = -1	
u=i	Rx = 0	
v>j	Ry = 1	j=j+Ry
v<j	Ry = -1	
v=j	Ry = 0	
w>k	Rz = 1	k=k+Rz
w<k	Rz = -1	
w=k	Rz= 0	

3.6.2.3) Exemple.

x : Pb

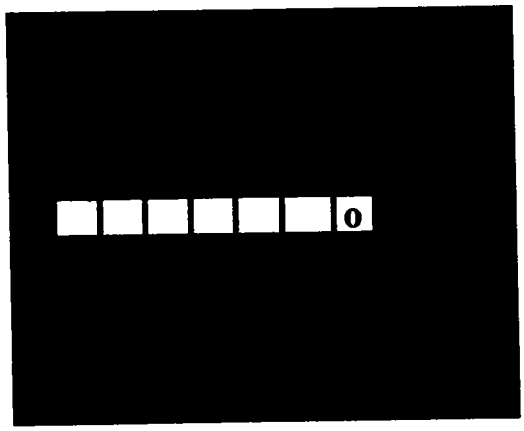
grille 0



0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1 1 0
0 0 0 1 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0

o : Pr

grille 1



0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

fig. 3.52 : Grilles représentant le modèle

Considérons l'exemple représenté par la figure 3.38b. La procédure de modélisation nous donne les 2 grilles de la figure 3.52. Le plan P1 est représenté sur la grille 0, le plan P2 et la transition T1 sont représentés sur la grille 1.

Nous trouvons la position du robot (Pr) sur la colonne 8, ligne 5 de la grille 1 et la position du but (Pb) sur la colonne 2, ligne 2 de la grille 0.

Soit

Pr = 00000001000 000010000 01

Pb = 01000000000 010000000 00

position colonne position ligne grille

Nous allons maintenant effectuer des opération booléennes (masque) afin de déterminer la (ou les différentes) possibilité(s) d'évolution du robot mobile.

Calcul d'un masque.

Le calcul du masque en fonction de la position relative robot/but nous donne :

- **masque = Pr OR (Pr / 2) si Rx = 1** sachant que la division par 2 nous donne un décalage vers la droite.
- **masque = Pr OR (Pr x 2) si Rx = -1** sachant que la multiplication par 2 nous donne un décalage vers la gauche.

Calcul des possibilités d'évolution.

Soient k et j la grille et la ligne actuelle.

Soient w et v la grille et la ligne but.

Soient Rx=1, Ry=-1 et Rz=-1 les critères de rapprochement définis, nous avons

A1=masque AND M(k, j) = 00000001000

B1=masque AND M(k, j+Ry) = 00000000000

A2=masque AND M(k+Rz, j) = 00000000100

B2=masque AND M(k+Rz, j+Ry)= 00000000000

a) **Cas particulier.**

Excluons des possibilités d'évolution la position actuelle du robot

A1=A1 AND NOT(Pr) = 0 0 0 0 0 0 0 0 0 0 0 0

Dans ce cas, la seule possibilité d'évolution est A2.

b) **Cas général.**

Dans le cas général, les différentes possibilités d'évolution sont données par diverses opérations logiques sur les mots binaires A1, B1, A2 et B2. La priorité entre ces différentes solutions est donnée par la fonction F1(n). Cette fonction F1(n) peut privilégier soit

- la recherche dans le même plan par un algorithme vu au chapitre 3.5.1.3.

- le changement de plan. Dans ce cas, la position relative des cases accessibles dans les différentes grilles nous indique si le changement de plan est possible.

Les différentes possibilités d'évolution sont résumées sur le tableau ci-dessous.

Mouvement	Condition
Evolution dans le meme plan	if NOT(((A1 AND A2)==0)&((B1 AND B2)==0))) changement de plan non possible else if (A2!=0) Pr = A2 else if (S3!=0) Pr = S3 else changement de plan non possible N.B. : les tests (A2!=0) et (S3!=0) peuvent être inversés
Evolution vers un plan différent	if (S2!=0) pas robot = S2 else if (S1!=0) Pr = S1 else if (S0!=0) Pr = S0 else déplacement impossible N.B. :les tests (S0!=0) et (S1!=0) peuvent être inversés

avec

S0 = A1 AND NOT(Pr)
S1 = A1 AND B1
S2 = S1 AND (S1>>1) si Rx = 1
S2 = S1 AND (S1<<1) si Rx = -1
S3 = B2 AND (Pr)

3.6.3) Algorithme SEARCHPATH1

L'algorithme SEARCHPATH1 ($\pi 0$, πn) nous donne la liste des cellules faisant partie de la trajectoire de la cellule source $\pi(0)$ à la cellule but $\pi(n)$.

3.6.3.1) Principes de l'algorithme.

1)Créer une liste de cellules formant les chemins comprenant un élément $\pi(1)$ qui est la position actuelle du robot. Les codes C1(n) et C2(n) associés autorisent tout déplacement à partir de $\pi(1)$. Initialiser i à 1.

2)Faire M(k, p, q)=0 avec p et q désignant le bit du vecteur position actuelle du robot dans la grille k.

3)Calculer les valeurs Rx, Ry et Rz déterminant le sens de progression possible en $\pi(i)$ selon les possibilités définies par C1(n) et C2(n) et selon la fonction de priorité F1. S'il n'y a aucune possibilité, alors saut en 10.

4)Modifier les codes C1(n) et C2(n) en fonction du choix Rx, Ry et Rz.

5)Déterminer le masque.

6)Calculer la nouvelle position du robot. Si la nouvelle position du robot n'existe pas, alors saut en 3.

7)i=i+1. Ajouter à la liste $\pi(i)$ la nouvelle position du robot et les codes associés C1(n) et C2(n) autorisant tout déplacement autour de $\pi(i)$.

8)Si position robot = position but, alors FIN

9)Saut en 2

10)i=i-1. Si i=0, alors FIN : pas de déplacement possible.

11)Saut en 3.

3.6.3.2) Opération de backtracking.

L'opération de backtracking, évitant les échecs dans le cas d'une planification parmi des obstacles concaves et abordée au chapitre 3.5.2.2, est de nouveau utilisée dans SEARCHPATH1.

3.6.3.3.) Opération d'optimisation.

La trajectoire obtenue est optimisée par un algorithme analogue à celui vu au chapitre 3.5.3. Cette opération d'optimisation consiste à prendre des sous-buts, situés sur la trajectoire initiale, que le mobile essaye de rallier grâce à l'algorithme SEARCHPATH1.

3.7) Modélisation et planification pour le projet VAHM.

3.7.1) Modélisation.

L'univers dans lequel va évoluer le fauteuil roulant est une habitation que l'on peut représenter comme un univers $2D^{1/2}$. Ce $2D^{1/2}$ peut se modéliser par n environnement(s) bi-dimensionnel(s), représentant les n niveaux d'une maison et communiquant entre eux par ascenseur ou rampe d'accès spéciale pour fauteuils roulants.

La procédure de modélisation, appliquée à la figure 3.53, nous donne la figure 3.55.

En cas de niveau fortement encombré, il est possible de le décomposer en plusieurs grilles représentant une ou plusieurs pièces. Ces grilles seront reliées par un graphe de connexité. La recherche de la trajectoire se fera par

- une recherche sélective des grilles à traverser (navigation globale),
- une recherche dans chaque grille sélectionnée (navigation locale).

3.7.2) Planification.

La modélisation de l'environnement nous donne plusieurs grilles, représentant plusieurs niveaux, reliées les unes aux autres par un ou plusieurs points de passage correspondant aux ascenseurs ou rampes d'accès. La recherche de la trajectoire s'effectue en utilisant l'algorithme SEARCHPATH. Plusieurs cas peuvent se présenter :

a)Si la position instantannée du fauteuil et la position but se trouvent sur la même grille, SEARCHPATH est appliqué entre ces deux points.

b)Si la position instantannée du fauteuil et la position but se trouvent sur des grilles différentes mais directement reliées par un point de passage commun, SEARCHPATH est appliqué

- entre la position instantannée et le point de passage, puis

- entre le point de passage et la position but.

La trajectoire totale sera égale à la somme des deux trajectoires obtenues.

c)Si la position instantanée du fauteuil et la position but se trouvent sur des grilles différentes non directement reliées par un point de passage, la trajectoire totale sera obtenue en appliquant SEARCHPATH à travers les différentes grilles.

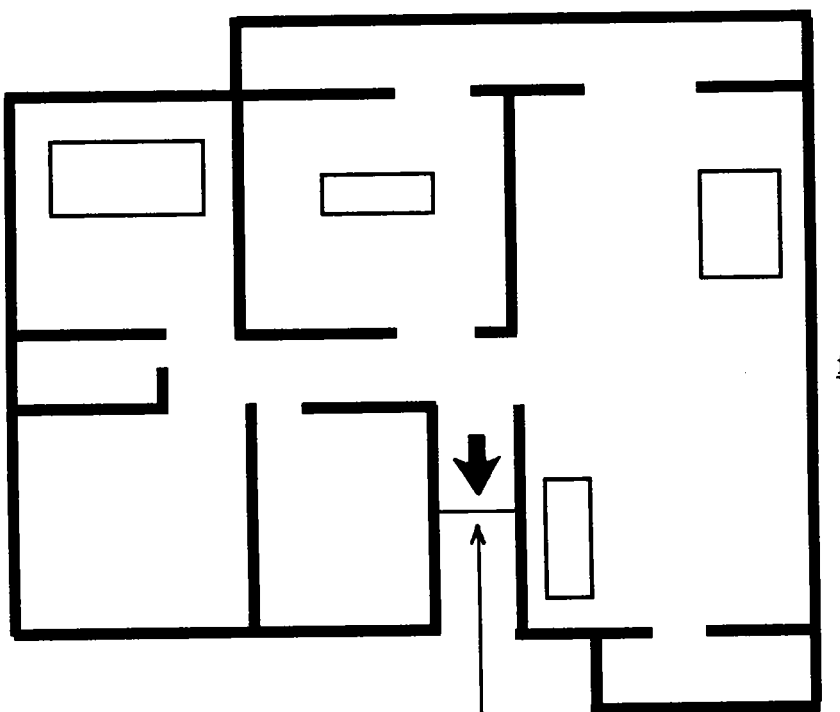


fig. 3.53 : Habitation niveau 1

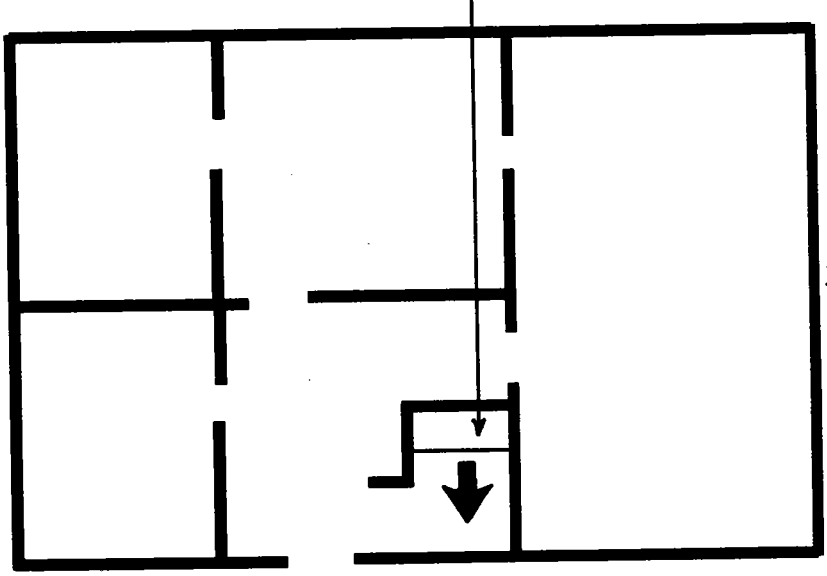
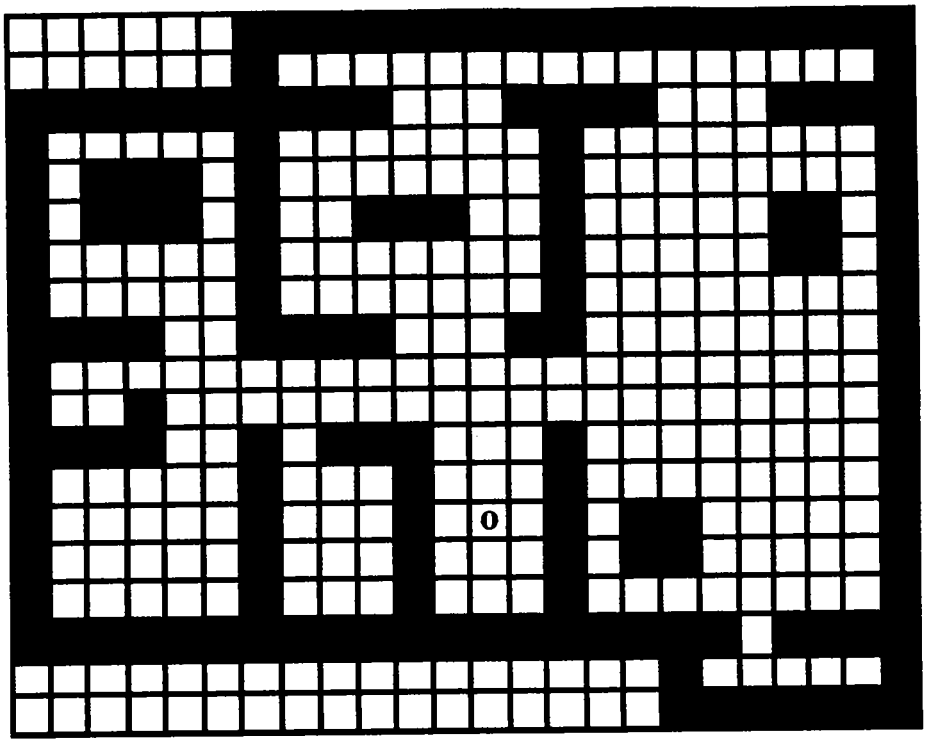


fig. 3.54 : Habitation niveau 0



0 : point de passage

niveau 1/niveau 2

fig. 3.55 : Modèle correspondant à la figure 3.53

CHAPITRE IV :
CONCLUSION

IV) CONCLUSION

Les robots mobiles autonomes suscitent actuellement des projets ambitieux et futuristes. Ceux-ci vont du domaine agricole, avec Magali, au domaine spatial avec le projet VAP-RISP : robot d'exploration pour une mission sur Mars. Pourtant s'il existe un domaine où la robotique mobile peut rendre encore plus de services quotidiens, c'est bien celui de l'aide aux handicapés. C'est avec cet objectif que le L.A.E.I. travaille sur le projet V.A.H.M.

Le projet V.A.H.M. propose la transformation d'un fauteuil roulant en véhicule autonome. Ce véhicule devra permettre aux personnes atteintes de maladie neuro-musculaires, de manipuler des objets à l'aide d'un bras articulé actuellement à l'étude, et de pouvoir se déplacer de manière automatique ou manuelle dans une habitation. La planification de la trajectoire à travers l'habitation, gérée par un système décisionnel intégré au V.A.H.M. prendra en compte l'environnement perçu par des capteurs appropriés.

Afin de situer le travail qui a été réalisé, nous avons exposé dans le chapitre II les diverses approches utilisées par des chercheurs ou équipes de chercheurs qui traitent du problème de la planification de trajectoire. Dans le chapitre III, nous avons exposé la contribution que nous avons apportée au projet V.A.H.M., laquelle se situe au niveau de la planification de trajectoire. En effet, nous avons proposé une méthode de modélisation de l'espace d'évolution d'un robot mobile basée sur la cellularisation en grille non homogène. La solution que nous avons examinée en 2D puis en $2D^{1/2}$ possède les avantages de n'occuper qu'une place mémoire réduite ainsi que de pouvoir se modifier de manière dynamique. Enfin nous avons examiné une méthode de recherche de trajectoire 2D puis en $2D^{1/2}$, dérivée des algorithmes de LEE et A*, adaptée à cette modélisation cellulaire.

La méthode de modélisation que nous avons proposée

- nous apporte un modèle de calcul rapide (les temps donnés à titre indicatif au paragraphe 3.25 le prouvent) :
- n'occupe qu'une place mémoire nettement réduite par rapport aux modélisations par grilles homogènes,
- détermine un modèle de calcul

-pouvant se modifier dynamiquement d'une manière rapide (voir les pro cédures de modification du maillage),

-dont les caractéristiques sont faciles à gérer et à stocker (mots de 32 bits).

En outre, la grille qui représente l'environnement est rendue homogène, ce qui facilite la recherche de trajectoire grâce à l'algorithme SEARCHPATH. La prise en compte des dimensions du robot s'effectue, dans le cas des arêtes non parallèles aux axes, au moment de la modélisation. La taille du robot intervient directement dans les dimensions de la grille.

La recherche de trajectoire que nous avons proposée utilise un algorithme heuristique SEARCHPATH. Grâce à une procédure de backtracking, cet algorithme trouve toujours, si elle existe, une évolution admissible : la trajectoire déterminée ne sera peut-être pas optimale, mais elle sera trouvée très rapidement et nécessitera une place mémoire plus faible. SEARCHPATH est basé sur des opérations binaires élémentaires (OU, ET, NON, ...) et les calculs sont facilités par une structuration du modèle en mots de 32 bits.

Par contre, comme pour toute heuristique, il est toujours posssible de trouver des configurations géométriques annulant les effets de rapidité dont fait preuve cet algorithme.

Puisqu'une heuristique est adoptée pour résoudre le problème de la planification de la trajectoire, il importe peu, dans la majorité des cas, que des cellules identiques sur le modèle de calcul représentent sur l'environnement des surfaces de tailles différentes.

Le logiciel MEPT, présenté en annexe, sera implanté très prochainement sur le V.A.H.M. et sera testé. Il nécessitera la connexion des différents modules : perception, localisation et commande des mouvements. Ensuite pour le V.A.H.M., la phase d'apprentissage commencera avec l'exploration de son espace d'évolution. Les données acquises progressivement par le système de perception serviront à construire le modèle géométrique, prélude à la modélisation et à la planification.

Actuellement le déplacement entre les cellules faisant partie de la trajectoire s'effectue par segments de droite entre centres de cellules et lissage éventuel (paragraphe 3.5.5). Un objectif serait d'obtenir sur le site réel un changement d'orientation progressif.

L'assemblage des différents modules composant la partie mobilité va permettre une mise en service prochaine du V.A.H.M. Cette étude, que nous avons réalisée pour le projet V.A.H.M., peut sans problème s'adapter à un robot mobile naviguant dans un univers industriel.

REFERENCES BIBLIOGRAPHIQUES

[AHO 74] : AHO A.V., HOPCROFT J.E., ULLMANN J.D.,
«The Design and Analysis of Computer Algorithms» Addison Weslay, 1974.

[ASA 85] : ASANO T., EDAHIRO M., IMAI H., IRI M., MURATO K.
«Practical use of bucketing techniques in computational geometry» In computational Geometry, GT Toussaint éditeur, North Holland,1985, pp. 153-195.

[ASI 70] : ASIMOV I.
«Robot»
Fawcett Crest Publication, Greenwich, Conn. 1970.

[AUT 88] : «Recherche et développement sur les robots pour milieu hostile»
Automates & Robots, n°1, janvier 1988, pp. 12-13.

[AVN 87] : AVNAIM F., BOISSONNAT J.D.,
«The polygon containment problem : 1. simultaneous containment under translation»
INRIA rapport de recherche n° 689 (juin 1987).

[AVN 88a] : AVNAIM F., BOISSONNAT J.D.
«Polygon placement under translation and rotation» Symp. on Theoretical Aspects of Computer Science ,88, Lectures Notes in Computer Science 294, Springer Verlag.

[AVN 88b] : AVNAIM F., BOISSONNAT J.D., FAVERJON B.
«A pratical exact motion planning algorithm for polygonal objects amidst polygonal obstacles» CH 2555-1/88/ IEEE 1988 pp. 1656-1661.

[AZO 76] : AZOULAY P. DASSONVILLE P.
<<Recherche opérationnelle de gestion.>> Presses universitaires de France, Tome 1, Paris, 1976.

[BEL 58] : BELLMAN S.E. 1965
<<La programmation dynamique et ses applications.>> Editions DUNOD, PARIS, 1965.

[BOI 84] : BOISSONNAT J.D. FAUGERAS O.D., FAVERJON B., KOFAKIS P., LUSTMAN F., TOSCANI G.
«Modélisation des objets tri-dimensionnels» 3ème journées annuelles du programme ARA Toulouse, 26-28 Septembre 1984.

[BRO 83] : BROOKS R.A.
«Solving the Find-Path Problem by Good Representation of Free Space».
IEEE Transaction on SMC, Mars 83, volume 13, n°3, pp. 190-197.

[BRO 85] : BROOKS R.A., LOZANO-PEREZ T.
«A Subdivision Algorithm in Configuration Space for Findpath with Rotation».
IEEE Transaction on SMC, Mars)Avril 85, volume 15, n°2, pp. 224-233.

[CAY 76] : M. CAYROL, B. FADE, H. FARRENY
<<Etude en simulation des stratégies de contrôle d'un robot : le projet ARGOS.>>
Automatisme, mai 1976, 21, 5, pp. 162-172.

[CAY 78] : M. CAYROL
<<Conception de la simulation d'un robot prototype minimal.>> Thèse de 3ème cycle. U.P.S. Toulouse, 28 avril 1978.

[CHA 80] : CHATILA R., GIRALT G.
«A task-oriented navigation system for the mobile robot HILARE»
Proc. 2nd International Meeting on Artificial Intelligence, Leningrad, URSS, october 1980.

[CHA 81] : CHATILA R.
«Système de Navigation pour un Robot Mobile Autonome : Modélisation et Processus Décisionnels» Thèse de Docteur Ingénieur, Université Paul Sabatier, Toulouse, juillet 81.

[CHA 82] : CHATILA R.
«Path Planning and Environment Learning in a Mobile Robot System» European Conference on Artificial Intelligence, 12-14 juillet 82, ORSAY, FRANCE.

[CHA 85] : CHATILA R. LAUMOND J.P.
«Position Referencing and Consistent Word Modeling for Mobile Robot» IEEE Conf. on Robotic and Automation, St Louis, 1985.

[CLU 56] : Mc CLUSEY E.J.
«Minimisation of Booleans Functions» Bell Syst. Tech. J., volume 35, pp. 1417-1444, Novembre 1956.

[CRO 84a] : CROWLEY J.L.(a)
«Navigation for an intelligent mobile robot» Carnegie-Mellon University, Pittsburg Rapport technique, CMU-RI-TR-84-18.

[CRO 84b] : CROWLEY J.L., REDPATH R.J.
«An Algorithm for Maximum Area Convex Decomposition». Technical Report , CMU Robotics Institute, 1984.

[CRO 84c] : CROWLEY J.L.
«Word Modelling and Position Estimation for an Intelligent Mobile Robot» In Seventh. Internatioal Conference on Pattern Recognition August, 1984.

[CRO 85] : CROWLEY J.L.
«Navigation for an Intelligent Mobile Robot» IEEE Journal of Robotics and Automation Vol. RA-1, n°1, March 1985.

[CUN 88] : CUNIN J.C.
«Projet Robotique» Rapport interne AFM (Association Française contre les Myopa thies), octobre 88.

[DEP 90] : DEPLANTE M., MORNAS D., OLLIER J.L.
«Le programme AMR de robots mobiles avancés pour la sécurité civile» Séminaire Les Robots Mobiles, Paris-La Défense, 14-15 mars 1990.

[DIJ 59] : DIJKSTRA E.W.
<<A note on two problems in connection with graphs>>. Numerische Mathematik, 1959, pp. 269-271.

[DUC 90] : DUCLOS J.C.
«Champ d'application de la robotique mobile transitique/manutention» Séminaire Les Robots Mobiles, Paris-La Défense, 14-15 mars 1990.

[EDE 86] : EDELSBRUNNER H., SEIDEL R.
«Arrangements of planes and Voronoï diagrams» Discrete and Computational Geometry, Volume 1, n°1, 1986.

[FAR 90] : FARGEON C., DELETOMBE E.
«Robotique Militaire et Essais» Séminaire Les Robots Mobiles, Paris-La Défense, 14-15 mars, 1990.

[FAV 87] : FAVERJON B., TOURNASSOUD P.
«A local based approach for path planning of munipulators with a high number of degrees of freedom» Rapport de recherche INRIA 621, février 1987.

[FLO 69] : M.H. FLOOM
<<The combination of a global routing algorithm and a path finding algorithm for an unmanned roving vehicle>>. Thesis of Master of Science, oct. 1969. Monterey (Cal) USA.

[FOR 86] : FORTUNE S., WILFONG G., YAP C.K.
«Coordinated motion of two robot arms» In Proceedings of IEEE Int. Conference on Robotics and Automation, pp. 1216-1223, San Francisco, avril 1986.

[FOR 87] : FORTUNE S.J.
«A sweepline algorithm for Voronoï diagrams» Algorithmica, volume 2, n°2, 1987.

[FRA 90] : FRAIZE G.
«La robotique mobile dans l'industrie nucléaire : état de l'art et application» Séminaire Les Robots Mobiles, Paris-La Défense, 14-15 mars 1990.

[FRY] : FRYXELL R.C.
<<Navigation planning using Quadtrees>> SPIE volume 852 Mobile Robots II.

[FUJ 71] : K. FUJII, T. MORITA, M. YACHIDA
<<Study on route searching algorithm of the intelligent robot>>. Shisuternu to Seigo (Système et Contrôle) 1971, 15, 11, pp. 68-75.

[GIR 90] : GIRALT G.
«Les robots mobiles : principales réalisations et perspectives» Développements futurs en Robotique, Paris, 8 mars 90.

[GON 79] : GONDRAN Michel, MINOUX Michel
<<Graphes et Algorithmes>>
Editions Eyrolles, 61 boulevard Saint Germain, Paris 5ème, 1979, pp. 36.

[HAR 68] : HART P.E., NILSSON N.J., RAPHAEL B.
<<A formal Basis for the Heuristic Determination of Minimum Cost Paths>>
IEEE Transaction on SMC, Juillet 68, volume 4, n°2, pp. 100-107.

[IGN 73] : M.B. IGNATEV, F.M. KULAKOV, A.M. POKROVSKIY
<<Robot-manipulator control algorithms.>> Joint publications research service.Arlington (Virginia), aug. 6 1973, pp. 231-244.

[IND 90] : «Machinisme agricole : une année très fertile»
Industrie mécanique, n°1548, 8 mars 1990, pp. 3.6.

[KAK 72] : M. KAKIKURA , T. MISHIMA
<<On the route-finding for an intelligent robot.>> Bull. Electrotech. Lab. (Japan), 1972, 36, 5, pp. 33-45.

[KAM 86] : KAMBHAMPATI S. and DAVIS L.S.
<<Multiresolution Path Planning for Mobile Robots.>> IEEE Journal of Robotics and Automation Vol. RA-2, n°3, Septembre 1986.

[KED 86] : KEDEM K., LIVNE R. PACH J., SHARIR M.
«On the union of Jordan Regions and collision free translational motion amidst polygonal obstacles». Discrete and Computational Geometry 1, GT Toussaint éditeur, North Holland, 1986, pp. 59-71.

[KHA 78] : KHATIB O., LE MAITRE J.F.
«Dynamic control of manipulators operating in a complex environment». In Proceeding of the 3rd CISM-IF to MM. Udine, Italie, Septembre 1978.

[KHA 86] : KHATIB O.
«Real-Time Obstacle Auvidance for Manipulators and Mobile Robots». The International Journal of Robotics Research. Vol. 5, n°1, pp. 90-99, Printemps 1986.

[KRO 84] : KROGH B.H.
«A Generalized Potentiel Field Approach to Obstacle Avoidance Control». Robotics Research : The Next Five Years and Beyond.
14-16 Août 1984, BETHLEHEM, Pennsylvania.

[KRO 85a] : KROGH B.H.
«Guaranteed Steering Control» America Control Conference, 19-21 juin 1985, Boston, pp. 950-955.

[KRO 85b] : KROGH B.H., GRAETTINGER T.
«Maneuverability Constraints for Supervisory Steering» 24th Conference on Decision and Control, Fort Lauderdale, FL, Décembre 1985, pp. 279-284.

[KRO 86] : KROGH B.H., THORPE C.E.
«Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles». CH 2282-2/86-IEEE, pp. 1664-1669.

[KUA 84] : KUAN T.D., BROOKS R.A., ZAMISKA J.C., DAS M.
«Automatic Path Planning for a mobile robot using a mixed representation of free space». IEEE First Conference on Artificial Intelligence Application, Denver, Colorado, Décembre 1984.

[KUA 85] : KUAN T.D., ZAMISKA J.C., BROOKS R.A.
«Natural decomposition of free space for path planning» IEEE conference, Saint-Louis, mars 1985.

[LAP 80] : LAPORTE A.
<<Un robot expérimental d'assemblage automatique opérant dans un univers de blocs.>> Thèse de Docteur-Ingénieur. U.P.S. Toulouse, 9 juillet 1980.

[LAU 83] : LAUMOND J.
«Model Structuring and Concept Recognition : Two Aspects of Learning for a Mobile Robot». Proc. 8th Conf. on Artificial Intelligence, Karlsruhe, West Germany, p. 839.

[LAU 84] : LAUMOND J.P.
«Utilisation de Graphes Planétaires pour l'Apprentissage de la Structure de l'Espace d'Evolution d'un Robot Mobile». Thèse de Docteur de 3ème cycle, Université Paul Sabatier, Toulouse 1984.

[LEE 61] : LEE C.Y.
«An algorithm for path connections and its application». IRE Trans. on Electronic Computers, Sept. 1961, pp. 346-365.

[LEM 90] : LEMARQUAND P.
«Le projet PANORAMA et ses applications forestières et minières» Séminaire Les Robots Mobiles. Paris-La Défense, 14-15 mars 1990.

[LEV 87] : LEVEN D., SHARIR M.
«An efficient and simple motion planning algorithm for a ladder moving in a two dimensional space amidst polygonal barners» Journal of Algorithms, vol. 8, n°7 (1987).

[LOZ 79] : LOZANO-PEREZ T., WESLEY M.A.
«An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles» Communications of the ACM, Octobre 1979, volume 22, n°10, pp. 560-570.

[LOZ 81] : LOZANO-PEREZ T.
«Automatic Planning of Manipulator Transfer Movements» IEEE Transactions on S.M.C., volume SMC-11, n°10, octobre 81.

[LOZ 83] : LOZANO-PEREZ T.
«Spatial Planning : a configuration space approach» IEEE Transaction on comp.,volume 32, n°2, 1983.

[MOO 57] : MOORE E.F. (1957)
<<The shortest path throught a maze>> Annale du Computation Laboratory de l'université de Hardward, 1957, pp. 285-292.

[MOR 80] : H.P. MORAVEC
<<Obstacle avoidance and navigation in the real world by a seeing robot rover.>> Ph. D. Thesis, Standford University, sept. 2, 1980.

[NEW 87] : NEWMAN W.S., HOGAN N.
«Hight Speed Robot Control and Obstacle Avoidance Using Dynamic Potentiel Functions.» CH 2413-3/87 IEEE, pp. 14-24.

[NIL 69] : N.J. NILSSON
<<A mobile automaton : an application of artificial intelligence techniques.>> Proc. of the 1st I.J.C.A.I., 7-9 mai 1969, pp. 509-520.

[NIL 80] : NILSSON N.J. (1980)
<<Principles of artificial intelligence.>> Tiogo Publishing Co., 1980 I.S.B.N. 0-935382-01-1

[ODU 85] : O'DUNLAING, YAP C.
«A retraction method for planning the motion of a disk». Journal of algorithms 6 (1985) pp. 104-111.

[OHY 84] : OHYA T., IRI M., MURATO K.
«Improvements of the Incremental Method for the Voronoï diagram with computational comparison of various algorithms». Journal of the Operations Research Society of Japan, volume 27 (4), 1984, pp. 306-336.

[PAP 82] : PAPADIMITRIOU C.H.
«Combinational optimisation» Editeur PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632, pp. 170-173.

[PRA 80] : PRAJOUX R., SOBEK R.P., LAPORTE A., CHATILA R.
<<A robot system utilizing task specific planning in a blocks-world assembly experiment.>> Proc. of the 10th I.S.I.R. Milan, mars 1980, pp. 281-292.

[POL 88] : POLLACK R., SHARIR M., SIFRONY S.,
«Separating two simple polygons by a sequence of translations» Discrete and Computational Geometry 3, GT Toussaint éditeur, North Holland, 1988, pp. 123-136.

[PRE 85] : PREPARATA F.P., SHAMOS M.I.,
«Computational Geometry : an Introduction». Springer Verlag, 1985.

[PRO 87] : Proceedings of the Fourth International Symposium on Robotics and Artificial Intelligence in Building Construction, TECHNION, Haïfa, 22-25 june 1987.

[PRO 88] : Proceedings of the Fith International Symposium on Robotics in Construction, JIRA, Tokyo, 6-8 june 1988.

[PRO 89] : Proceedings of the Sixth International Symposium on Automation and Robotics in Construction, CII, San Francisco, 6-8 juin 1989.

[PRU 86] : PRUSKI A.,
«Les robots mobiles. Localisation et navigation.» Rapport interne L.A.E.I., Université de Metz, Déc. 1986.

[PRU 89 a] : PRUSKI A., BOSCHIAN V.,
«Grid Modelisation for Autonomous Robot.» 6th Symposium on Information Control Problems in Manufacturing Technology, Madrid, 26-29 Septembre 1989.

[PRU 89 b] : PRUSKI A.,
«Codes multivaleurs.» Rapport Interne L.A.E.I., 1989.

[PRU 89 c] : PRUSKI A.,
«V.A.H.M. : Véhicule Autonome pour Handicapés Moteur.» Rapport Interne L.A.E.I., Juillet 89.

[QUI 52] : QUINE W.V.,
«The problem of simplifying truth functions.» Amer. Math. Monthly, volume 59, pp. 521-531, oct. 1952.

[RAP 76a] : RAPHAEL B.
<<The thinking computer mind inside matter.>> Freeman and company, San Francisco, 1976, pp. 275-288.

[RAP 76b] : RAPHAEL B.
<<The robot «SHAKEY» and «His» successors.>> Computers and People, oct. 1976, pp. 7 et 21.

[RIC 81] : RICHETIN M., NARANJO M., RIVES G.
<<Commande d'un automate à partir d'informations de reconnaissance des formes.>> 3ème Congrès de Reconnaissance des Formes et d'Intelligence Artificielle. AFCET, Nancy, sept. 1981, pp. 461-472.

[ROB 89] : «Un robot ne parle pas sous la torture.» Le journal de la robotique, n°55, sept. 89, pp. 32-33.

[ROB 85] : «Un modèle probatoire de robot cueilleur de fruits.» Le journal de la robotique, n°11, sept. 85, pp. 8-9.

[ROG 82] : ROGERS D.F.
«Algorithmes pour infographie.» Mc GRAW-HILL, 1983.

[RUE 87] : RUEB K.D., WONG A.K.C.,
«Structuring Free Space as a Hypergraph for Roving Robot Path Planning and Navigation.» IEEE Transaction on pattern analysis and machine intelligence, vol. PAMI-9, n°2, mars 1987.

[SAM 87] : SAMET H., SHAFFER C.A., NELSON R.C.,
<<Recent Developments in linear quadtree.>> Image and vision computing, vol. 5, n°3, Août 1987, pp. 187-197.

[SCH 83] : SCHWARTZ J.T., SHARIR M.,
«On the piano mover's problem II. General techniques for computing topological properties of real algebraic manifolds.» Adv. Appl. Math, 4 (1983), pp. 298-351.

[SCH 84] : SCHWARTZ J.T., SHARIR M.,
«On the piano mover's problem : V. The case of a rod moving in three dimensional space amidst polyhedral obstacles.» Communication Pure Appl. Math 37 (1984), pp. 815-848.

[SED 83] : SEDGEWICK R.
«Algorithms.» Addison Wesley, 1983.

[SHP 84] : SHPITALNI M. (1984),
<<Switching Functions and Solid Geometrical Modeler.>> Proccedings of the I.E.E.E., vol. 72, n°1, janvier 84.

[SHP 85] : SHPITALNI M.
<<Switching Function Based Representation.>> Annals of the CIRP, vol. 34/1/1985, pp. 163-167.

[SIA 86] : SIARRY P.
<<La méthode du recuit-simulé.>> Laboratoire d'électronique, ESPCI, Nov. 1986.

[SIN 85] : SINGH J.S.
«Path planning and navigation for a mobile robot.» Master's thesis, Lehigh University, Août 1985.

[SIN 87] : SINGH J.S., WAGH M.,
<<Robot Path Planning using Intersection Convex Shapes Analysis and Simulation.>> I.E.E.E. Journal of Robotics and Automation, Vol. RA-3, n°2, avril 1987.

[TAK 81] : TAKEGAKI M., ARIMOTO S.,
«A New Feedback Method for Dynamic Control of Manipulators.» ASME J. Dynamic Systems Measurements and Control, Vol. 102, Juin 1981, pp. 119-125.

[THI 90] : THIEBAULT E.
«Le Robot Rondier Elf.» Séminaire Les Robots Mobiles, Paris-La Défense, 14-15 mars 1990.

[THO 83] : THORPE C.F.
«An Analysis of Interest Operator for FIDO.» Carnegie Mellon University, Pittsburg Rapport Technique CMU-RI-TR-83-19.

[THO 84] : THORPE C.F.
«Path Relaxation : Path Planning for a Mobile Robot.» Carnegie Mellon University, Pittsburg, Rapport Technique CMU-RI-TR-84-5.

[TOU 86a] : TOURNASSOUD P.
«On Motion Coordination.» Rapport de Recherche 549, INRIA, juillet 1986.

[TOU 86b] : TOURNASSOUD P.
«A Stategy for obstacle avoidance and its application to multi-robot systems.» In Proceedings of IEEE Int. Conference on Robotics and Automation, pp. 1224-1229, San-Francisco, avril 1986.

[TOU 88] : TOURNASSOUD P.
<<Géométrie et Intelligence Artificielle pour les robots.>> Edition HERMES, 51, Rue Hennequin 75017 Paris, 1988.

[WAT 81] : WATSON P.P.
«Computing the n-dimensional Delaunay triangulation with application to Voronoï polytapes.» Computer Journal, vol. 2, n°2, 1981, pp. 167-172.

[YAP 84] : YAP C.
«Coordination the moving of several discs.» Tech. Rent. n°105, Computer Science Dept., Courant Institute, New-York University, Février 1984.

ANNEXE
LOGICIEL MEPT

LOGICIEL M.E.P.T.

Afin de vérifier l'efficacité et la rapidité des algorithmes décrits au chapitre III, nous avons développé, pour la simulation, un logiciel de Modélisation de l'Environnement et de la Planification de Trajectoire (M.E.P.T.) pour robot mobile autonome.

1) Présentation.

M.E.P.T. est un logiciel, écrit en langage C, développé sur compatible et composé de 4 modules indépendants :

- module d'acquisition de données,
- module de modélisation,
- module d'animation dynamique,
- module de recherche de trajectoire et d'optimisation.

La structure modulaire du logiciel nous permet une exploitation séparée des différents modules.

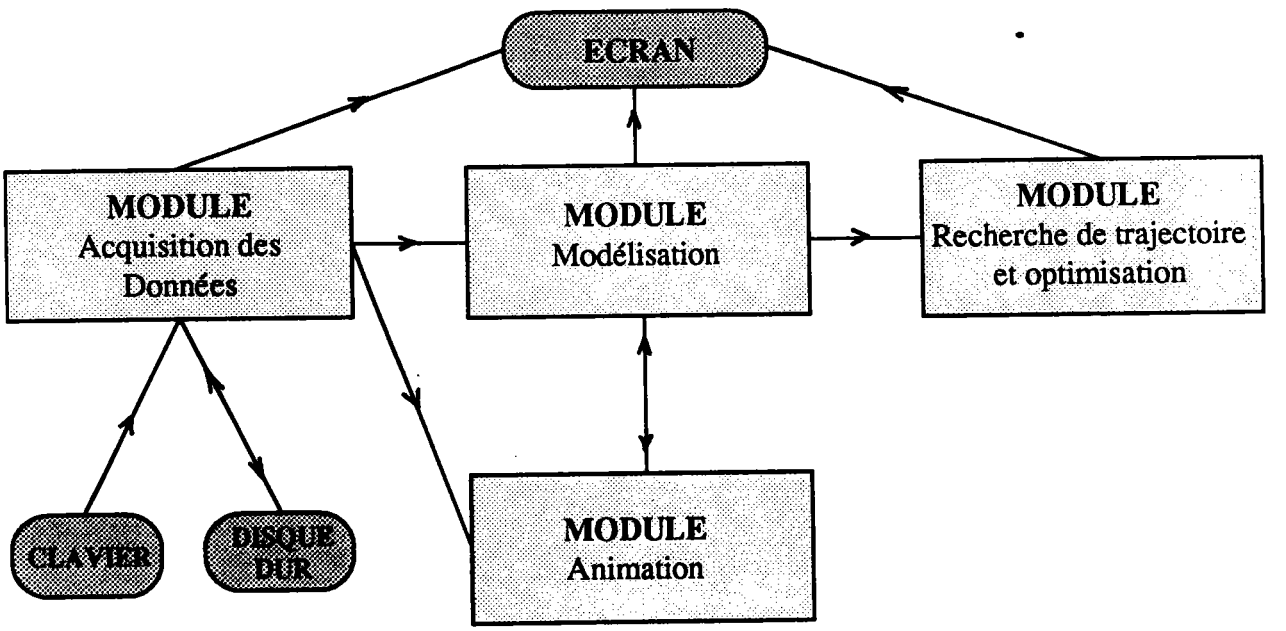


fig. 4.1 : Architecture générale du logiciel

2) Module d'acquisition des données.

Ce module permet de saisir et de modifier partiellement un environnement. L'environnement peut être défini de diverses manières :

- à l'aide de segments de droites définis graphiquement par le concepteur (cas du 2D),
- à l'aide de segments de droites dont l'utilisateur définit les coordonnées extrêmes par clavier (cas du 2D 1/2),
- à l'aide d'un ensemble de points (partie à terminer).

Les informations ainsi recueillies pourront être mémorisées sur disque dur.

3) Module modélisation.

Ce module de modélisation nous permet de déterminer un modèle de calcul à partir du modèle géométrique obtenu dans le modèle d'acquisition. Le modèle de calcul sera un tableau de

- 32 entiers longs, représentant un environnement de 32x32 cellules (version expérimentale),
- 256x8 entiers longs, représentant un environnement de 256x256 cellules.

Ce module effectue les opérations de

- partitionnement de l'environnement en bandes parallèles aux axes,
- création de tableaux ordonnés contenant les coordonnées des différentes bandes horizontales et verticales,
- recherche dans chaque bande obtenue de la pente la plus proche de 1,
- calcul de la partition maxi dans chaque bande,
- transformation en une grille homogène,
- création du modèle.

En outre il permet une visualisation, totale ou partielle, (cas de l'environnement 256x256), du modèle obtenu à l'écran.

4) Module animation dynamique.

Ce module permet la saisie et le déplacement d'obstacles en translation sur le modèle géométrique. La visualisation du modèle de calcul réactualisé s'effectue en même temps que le déplacement de l'obstacle.

Ce module nous a permis de vérifier la validité des algorithmes donnés dans le chapitre 3.3.3 concernant l'adaptation du modèle statique dans le cas d'un environnement dynamique.

5) Module recherche de trajectoire et optimisation.

Dans ce module recherche de trajectoire, nous effectuons des opérations de masquage sur le modèle de calcul, obtenant ainsi une trajectoire, à travers les obstacles, de la position source à la position but.

Si l'opérateur décide d'optimiser la trajectoire obtenue, il fournit au module la valeur de s (: saut, voir chapitre 3.5.3.), paramètre variant de 1 à n permettant d'obtenir un compromis entre la rapidité (s grand) et la finesse (s petit) de la trajectoire.

L'inversion source-but, demandée par l'opérateur, peut, lorsque le modèle s'y prête, permettre un raccourcissement de la trajectoire.

Une visualisation de la trajectoire, instantanée ou finale, sur le modèle de calcul ou sur le modèle géométrique, est fournie à l'écran.

6) Conclusion.

La réalisation de ce logiciel nous a permis de simuler le comportement d'un robot mobile dans un environnement conçu par l'opérateur.

Il est possible que l'implantation future de ce logiciel sur le fauteuil V.A.H.M. fasse apparaître quelques aléas de fonctionnement non prévisibles, mais que la conception modulaire de ce logiciel devrait aider à les résoudre.

RESUME

Les robots mobiles suscitent actuellement des projets ambitieux et futuristes. Ceux-ci vont du domaine agricole, avec Magali, au domaine spatial avec le projet VAP-RISP : robot d'exploration pour une mission sur Mars. Pourtant s'il existe un domaine ou la robotique mobile peut rendre encore plus de services quotidiens, c'est celui de l'aide aux handicapés. C'est avec cet objectif que le LAEI travaille sur le projet VAHM (Véhicule Autonomes pour Handicapés Moteur) proposant de rendre autonome un fauteuil électrique. Notre travail concerne la modélisation de l'environnement et la recherche de chemins.

Après une étude bibliographique critique, nous présentons :

a) une méthode de modélisation d'un environnement par grille adaptative à partir d'un espace déterminé par des objets polyédriques. La méthode consiste à partager l'espace en bandes parallèles à un référentiel. Ces bandes qui s'appuient sur les sommets des objets sont découpées selon le critère d'approche minimale des obstacles par le mobile. L'intersection des bandes horizontales et verticales forme des cellules élémentaires auxquelles est associé une variable binaire selon que la cellule fait partie de l'espace libre ou d'un obstacle. Cette méthode a ensuite été étendue à la modélisation

   -dynamique (indispensable pour l'apprentissage et dans le cas d'obstacles mobiles).
   -d'environnements 2D $^{1/2}$ (représentant, par exemple une maison mais aussi un univers industriel).

b) une méthode de recherche de chemins entre deux points du modèle. Celle-ci a été conçue à partir des algorithmes de LEE et A*, en effectuant des opérations de masquage. Cette méthode a l'avantage de nécessiter un espace mémoire réduit et de déterminer une trajectoire admissible rapidement.

La modélisation par grille adaptative détermine un modèle de calcul homogène directement utilisable par l'algorithme de recherche de trajectoire SEARCHPATH. Ces méthodes (modélisation et recherche de chemins) nous ont conduit à élaborer un logiciel de simulation (MEPT) qui nous permet de vérifier la rapidité et l'éfficacité des différents algorithmes.