L'Algorithme Fil d'Ariane

Juan Manuel Ahuactzin, Emmanuel Mazer, Pierre Bessière

Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle
46 Avenue Felix Vialet 38000 Grenoble, FRANCE

RÉSUMÉ. *Dans ce travail nous présentons une méthode générale de planification en robotique:* l'Algorithme Fil d'Ariane. *Cet algorithme est composé de deux sous-algorithmes SEARCH et EXPLORE. L'algorithme EXPLORE collecte des informations sur l'espace atteignable depuis la position initiale en posant des balises dans l'espace de recherche. SEARCH utilise une méthode locale pour atteindre la position finale à partir des balises posées. Après la présentation de quelques définitions de base nous présentons les fondements théoriques de l'algorithme Fil d'Ariane et nous montrerons que SEARCH et EXPLORE sont exprimés tous les deux comme des algorithmes d'optimisation. Nous montrons que EXPLORE permet d'approcher l'espace atteignable avec une résolution arbitraire en un nombre fini d'itérations. Enfin, nous présentons et discutons un exemple d'application de l'algorithme: la planification de trajectoires pour un robot mobile holonome.*

ABSTRACT. *We present a method for robot motion planning: the Ariadne´s Clew Algorithm. This algorithm is made of two sub-algorithms SEARCH and EXPLORE. The EXPLORE algorithm works by placing landmarks in the search space in such a way that a path from the initial configuration to any landmark is known. The SEARCH algorithm checks, with a local method, if the target may be reached from the last produced landmark. After the presentation of some basic definitions we show that, both the EXPLORE and the SEARCH algorithm can be expressed as optimization problems. We show that the EXPLORE algorithm allows to approach the reacheble space with an arbitrary resolution and after a finite number of iterations. Finally, we present and discuss an example using the Ariadne´s Clew Algorithm: the motion planning problem for an holonomic mobile robot.*

MOTS CLÉS : *raisonnement géométrique, planification de mouvements, robotique.*
KEYWORDS : *geometric reasonning, robot motion planning*

1 Introduction

Le but de cet article est de décrire formellement un algorithme de planification général à l'aide de deux problèmes d'optimisation de $\mathbb{R}^n$ dans R. La démarche que nous avons suivie est la suivante :

Après avoir situé notre travail, nous rappellerons tout d'abord les définitions formelles des concepts fondamentaux utilisés : espace des configurations, chemin dans $\mathbb{R}^n$, ensemble connexe par arc, distance de Hausdorff, billage et pavage d'un espace.

Puis introduirons une première méthode mathématique $EXPLORE_\infty$ qui permet de construire en un nombre fini d'étapes un ensemble discret de points qui approche l'espace libre arbitrairement près au sens de la distance de Hausdorff. Le résultat sur la convergence de cette méthode sera utilisé plus tard pour montrer la convergence d'un algorithme dérivé de la première méthode : $EXPLORE$.

Nous définirons ensuite la notion de *Chemin Manhattan* et la notion d'*espace libre à ε-près* qui représente naturellement les portions de l'espace des configurations qui se trouvent à plus de ε des obstacles.

L'agorithme $EXPLORE$ s'intéresse à des trajectoires entièrement contenues dans cet espace. Il permet d'approcher l'espace atteignable avec un ensemble discret de points. Nous montrerons que cette méthode - $EXPLORE$ - s'exprime sous la forme d'un problème d'optimisation. Le rôle premier de cette méthode est, bien entendu, de fournir une représentation approchée de l'espace atteignable qui peut être utilisée en planification de trajectoire. Cependant son intérêt majeur réside dans la manière dont cette représentation est calculée. En effet, cette méthode fournit une représentation de l'espace qui s'adapte naturellement à la complexité du problème de planification posé. Dans la section 5 nous introduirons deux améliorations possibles à l'algorithme $EXPLORE$: l'algorithme $SEARCH$ et le rebondissement. L'objet de $SEARCH$ est de fournir une méthode permettant d'augmenter l'efficacité d' $EXPLORE$.

L'intérêt de $SEARCH$ par rapport à d'autres méthodes de planification locales qui pourraient jouer un rôle similaire, est d'utiliser les mêmes outils et concepts que ceux employés dans $EXPLORE$. $SEARCH$ constitue donc un complément générique de $EXPLORE$. Enfin nous montrerons comment on peut modifier les fonctions à optimiser en introduisant la notion de rebondissement. Nous présenterons et discuterons un exemple d'application de l'algorithme Fil d'Ariane : la planification de trajectoires pour un robot mobile holonome.

2 Travaux similaires

Durant ces dernières annés, plusieurs méthodes de planification de trajectoires ont été développées. Une analyse des planificateurs de trajectoires existants peut être trouvée dans [LAT 91] et [HWA 92]. Parmi toutes ces méthodes, celle que nous proposons est comparable à celle de Overmars [OVE 92] [OVE 93], Švestka [SVE 92] et à celle du le planificateur SANDROS [CHH 92]. Les trois méthodes utilisent un ensemble de balises pour représenter l'espace libre et un planificateur local pour connecter les balises entre elles. Elles utilisent ce même planificateur pour connecter ces balises aux configurations initiale et finale. Dans le travail de Overmars et Švestka, les balises sont posées aléatoirement dans l'espace de recherche. Lorsqu'une balise est posée dans un *C-obstacle* elle est repositionnée dans la configuration appartenant à l'espace libre la plus proche. Le système SANDROS est composé de deux planificateurs : un planificateur global $\mathcal{G}$ et un planificateur local $\mathcal{L}$. Le planificateur $\mathcal{G}$ engendre une séquence de sous-buts pour guider le robot, et le planificateur $\mathcal{L}$ vérifie si ces sous-buts sont atteignables à partir des balises déjà posées. Les sous-buts sont définis d'une manière hiérarchique avec plusieurs niveaux de résolution.

Notre travail, l'algorithme Fil d'Ariane, montre que le problème de la planification peut, par un changement d'espace approprié, se ramener à la résolution d'un problème d'optimisation et que l'on peut facilement engendrer un planificateur dès lors que l'on sait prévoir la faisabilité d'une commande (ou plan) dans l'espace ou l'on définit les buts. Dans notre approche, nous ne nous intéressons pas à l'espace des configurations qui est, en pratique, difficile à construire et à représenter mais directement à l'espace des trajectoires. L'originalité de notre approche est, en effet, de conduire notre recherche directement dans l'espace des trajectoires. L'avantage principal de notre méthode par rapport aux méthodes précédemment citées est de placer les balises d'une manière plus efficace en permettant notamment l'adaptation automatique à la complexité du problème de planification posé.

3 Définitions

3.1 L'espace des configurations

On dénote par $\mathcal{W}$ l'espace ambiant dans lequel évolue le robot considéré. En général on peut déterminer la position de tous les points du robot dans cet espace par un nombre restreint de paramètres $(x_1, x_2, \ldots, x_n)$. Le choix de ces paramètres définit un espace appelé *espace des configurations* [LAT 91, LOZ 87] que l'on dénote par $\mathcal{C}$. Par définition, la position du robot dans $\mathcal{W}$ est complètement définie par la donnée d'un point dans $\mathcal{C}$.

En général, des contraintes d'ordre mécanique ou physique empêchent le robot de se trouver dans certaines régions de $\mathcal{W}$. Ces régions définissent implicitement des régions interdites de $\mathcal{C}$: on nomme ces régions les *C-obstacles*. Par exemple, pour un robot manipulateur les contraintes d'ordre physique, sont imposées par les obstacles $\mathcal{B}_{i=0,\ldots k} \subset \mathcal{W}$ placés dans l'espace accessible au robot (voir figure 1a). Evidemment, le robot et les obstacles ne peuvent pas occuper simultanément le même sous-espace de $\mathcal{W}$. Ainsi, toutes les configurations positionnant le robot dans l'espace occupé par un obstacle appartiennent, dans l'espace des configurations, aux *C-obstacles*.

Définition 3.1 ***(Emplacement d'un robot)*** *Soient $\mathcal{A}$ un robot, $\mathcal{C}$ l'espace des configurations de $\mathcal{A}$, $q \in \mathcal{C}$ une configuration du robot et $\mathcal{W}$ l'espace de travail de $\mathcal{A}$. L'*emplacement de $\mathcal{A}$ en $\mathcal{W}$ donné par *q est défini comme le sous-espace de $\mathcal{W}$ occupé par $\mathcal{A}$ étant données les valeurs des paramètres de q. Il est noté $\mathcal{A}(q)$.*

Avec la définition précédente, nous pouvons définir les *C-obstacles*. Chacun des obstacles physiques en $\mathcal{W}$ est représenté dans l'espace des configurations $\mathcal{C}$ de $\mathcal{A}$ en un ensemble de régions :

$$\mathcal{CB}_i = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{B}_i \neq \emptyset\}$$

Ainsi, on définit :

Définition 3.2 ***(C-obstacles)*** *L'ensemble des* C-obstacles *noté $\mathcal{CB}$ est défini comme l'union des transformations de tous les obstacles du monde physique dans l'espace des configurations, c'est-à-dire :*

$$\mathcal{CB} = \bigcup_{i=1}^{k} \mathcal{CB}_i$$

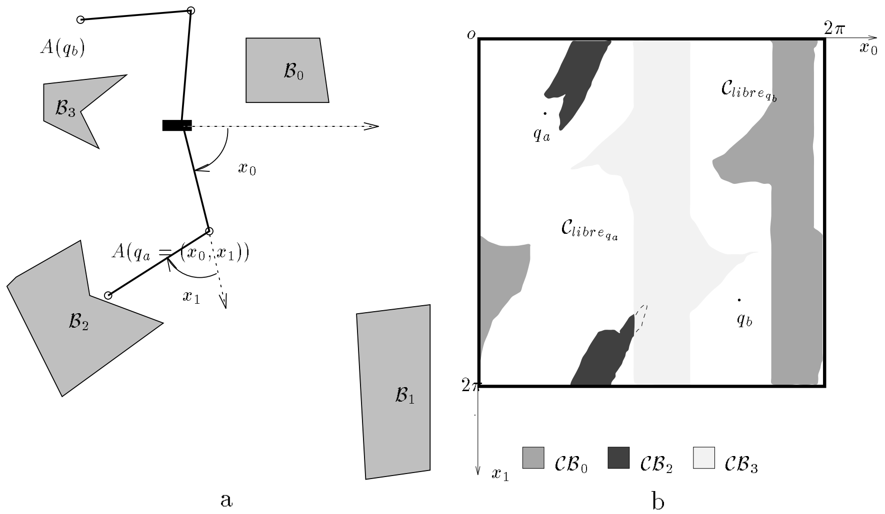


FIG. 1 - *L'espace de travail et l'espace des configurations d'un robot planaire.*

Par exemple, la figure 1a montre un bras manipulateur planaire en deux positions différentes et placé parmi quatre obstacles $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$. Pour définir une configuration pour ce type de robot on utilise deux paramètres x_0 et x_1. Ainsi, l'espace des configurations $\mathcal{C}$ est égale à $[0, 2\pi[^2 \subset \mathbb{R}^2$ (voir figure 1b).

Le complémentaire de $\mathcal{CB}$ définit un sous-espace de $\mathcal{C}$ où le robot n'est pas en collision avec l'environnement. Cet espace est appelé *l'espace libre*:

Définition 3.3 ***(Espace libre)*** L'espace libre*, noté $\mathcal{C}_{libre}$, d'un robot $\mathcal{A}$ est défini comme l'ensemble des configurations libres de collision de $\mathcal{A}$:*

$$\mathcal{C}_{libre} = \mathcal{C} \backslash \mathcal{CB} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{B}_{i=1,2,\ldots,k} = \emptyset\}$$

3.2 Chemins et ensembles connexes par arc

Définition 3.4 ***(Chemin de R^n)*** *Soit I l'intervalle $[0,1] \subset \mathbb{R}$. Un* chemin *$\hat{\varphi}$ de R^n d'un point initial $q_\circ$ à un point final $q_\bullet$ est défini comme un n-uplet de n fonctions continues $(\varphi_1(\alpha), \varphi_2(\alpha), \ldots, \varphi_n(\alpha))$ de $I \to \mathbb{R}$ avec :*

$$q_\circ = (\varphi_1(0), \varphi_2(0), \ldots, \varphi_n(0)) \quad et \quad q_\bullet = (\varphi_1(1), \varphi_2(1), \ldots, \varphi_n(1))$$

Par conséquent, pour un robot à n degrés de liberté, toute trajectoire est représentée par un ensemble de fonctions continues dans l'espace des configurations du robot. De plus, une trajectoire sans collision $\hat{\varphi}$ ne doit pas contenir une intersection avec les *C-obstacles*, soit :$\forall \alpha \in I$, $(\varphi_1(\alpha), \varphi_2(\alpha), \ldots, \varphi_n(\alpha)) \notin \mathcal{CB}$.

Définition 3.5 ***(Connexes par arc)*** *On dit que deux ensembles $\mathcal{C}_1$ et $\mathcal{C}_2 \subset C$ sont* connexes par arc *si et seulement si $\forall q_1, q_2 \in \mathcal{C}_1, \mathcal{C}_2$ il existe un chemin de q_1 à q_2 dans $\mathcal{C}$.*

On peut noter qu'un ensemble connexe par arc est forcément connexe mais que la réciproque n'est pas vraie. Par exemple, l'ensemble $[0,1[\ \cup\ ]1,2]$ est connexe, au sens général de la topologie, mais n'est pas connexe par arc [DUG 76].

Définition 3.6 ***(Espace atteignable)*** *Soit q une configuration en $\mathcal{C}_{libre}$, on définit* l'espace atteignable *de q : $\mathcal{C}_{libre_q} \subset \mathcal{C}_{libre}$ comme l'union de tous les sous-ensembles connexes par arc de $\mathcal{C}_{libre}$ contenant q. On appelle aussi à $\mathcal{C}_{libre_q}$* la composante connexe par arc *de q.*

Par exemple, dans la figure 1b, nous avons deux régions non connexes et donc non connexes par arc de l'espace libre : $\mathcal{C}_{libre_{q_a}}$ et $\mathcal{C}_{libre_{q_b}}$. Autrement dit, il n'existe pas de chemin en $\mathcal{C}_{libre}$ allant de q_a à q_b ou vice versa.

3.3 La distance de Hausdorff

La distance de Hausdorff [DUG 76, PRS 85] permet de mesurer la proximité entre deux sous ensembles. Elle nous permettra de parler d'approximation de C_{libre_q} par un ensemble discret de points.

Définition 3.7 ***(Distances en R^n)*** *Soit $Y \subset I\!R^n$ un ensemble non vide nous définissons :*

1. *la* distance entre deux points $x, y \in I\!R^n$: $d(x,y) = ||x - y||$,
2. *la* distance d'un point $x \in I\!R^n$ *à l'ensemble Y :*
   $d(x,Y) = d(Y,x) = \min\{d(x,y) | y \in Y\}$ *pour Y*

Définition 3.8 ***(Fonction et distance d'Hausdorff)*** *Soient $\mathcal{F}(I\!R^n)$ l'ensemble des ensembles non vides bornés de $I\!R^n$ et $(X,Y) \in \mathcal{F}(I\!R^n) \times \mathcal{F}(I\!R^n)$, on définit la* fonction d'Hausdorff ***de*** *X* ***à*** *Y comme :*

$$h(X,Y) = \sup\{d(x,Y) | x \in X\} = \sup_{x \in X} \inf_{y \in Y} d(x,y)$$

et la distance d'Hausdorff ***entre*** *X* ***et*** *Y comme :*

$$\rho(X,Y) = \max[h(X,Y), h(Y,X)]$$

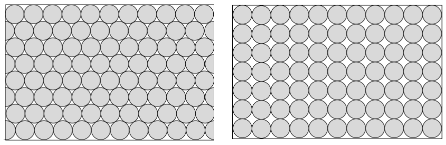
Nous utilisons cette distance pour estimer l'approximation de tout ensemble X borné par un ensemble énumérable de points $EL \subset X$. On peut remarquer que si $EL \subset X$ alors nous avons : $\forall q_i \in EL,\ d(q_i, X) = 0$ (voir définition 3.7). Ainsi, $h(EL, X) = 0$ et par conséquent :

$$\rho(EL, X) = h(X, EL)$$

3.4 Billage et pavage d'un espace

Si nous sommes capables de construire un ensemble de points EL tel que $\rho(EL, \mathcal{C}_{libre_{q_0}}) < \varepsilon$ alors : pour toute configuration $q \in \mathcal{C}_{libre_{q_0}}$ il existe au moins un point $p \in EL$ tel que la distance entre q et p est inférieure à ε. Autrement dit, toute configuration $q \in \mathcal{C}_{libre_{q_0}}$ a au moins un représentant en EL a $\varepsilon-$près.

De cette façon au lieu de calculer tout l'espace des configurations on peut l'approcher par un ensemble fini de points. Chacun de ces points représente les



a b

FIG. 2 - *Deux billages de densité différente sur* $\mathbb{R}^2$

configurations voisines à une distance inférieure ou égale à ε. Ceci revient à réaliser un *pavage* à $\varepsilon-$près de l'espace atteignable de q_0.

Bien entendu, nous sommes intéressés par des pavages "efficaces", c'est-à-dire que nous cherchons à approximer $C_{libre_{q_0}}$ avec le plus petit nombre de points. Ainsi, il faut établir un éloignement minimal entre les représentants. Cet éloignement définit implicitement un *billage* de l'espace approché.

Avant de définir les concepts de billage et de pavage [SAA 70] nous faisons un rappel de quelques concepts fondamentaux. A partir de ces concepts nous définirons la borne supérieure du cardinal du billage de rayon r d'un espace, autrement dit la borne supérieure du nombre de boules de rayon r pouvant occuper un sous espace borné de $\mathbb{R}^n$.

Définition 3.9 ***(Simplexe)*** *Un* simplex S *est défini comme l'enveloppe convexe d'un ensemble de* $(n+1)$ *points affinement indépendants de* $\mathbb{E}^n$. *On dit que* S *est* régulier *si tous ses côtés sont égaux .*

Définition 3.10 ***(Boule de*** $\mathbb{E}^n$***)*** *L'ensemble* $Bl(x,r) \subset \mathbb{E}^n$ *est appelé* "boule de centre x et de rayon r". *Il est défini comme* $Bl(x,r) = \{y \in \mathbb{R}^n \quad | \quad ||x-y|| \leq r\}$.

Ainsi, soit EL un ensemble de points de $\mathbb{E}^n$. L'*ensemble de boules de rayon* r *produit par* EL est noté $U(EL,r)$ c'est-à-dire : $U(EL,r) = \{Bl(p,r) \quad | \quad p \in EL\}$

Définition 3.11 ***(Billage)*** *Soit* $EL = (p_i)_{i \in \mathbb{N}^*}$ *une séquence de points en* $\mathbb{E}^n$ *et* $r \in \mathbb{R}$. $\mathcal{K} = U(EL,r)$ *est appelé* un billage *si pour tout couple de points* $(p_i,p_j) \in EL \times EL$ *où* $i \neq j$ $d(p_i,p_j) \geq 2r$. *Nous appelons* EL *l'ensemble des balises et* r *le rayon du billage.*

Nous notons C_s **le cube** $\{x \in \mathbb{R}^n | -\frac{s}{2} \leq x_i \leq \frac{s}{2}, 1 \leq i \leq n\}$ et $\nu(Y)$ **le volume** d'un sous-ensemble compact [1] Y de $\mathbb{R}^n$.

Ainsi, nous définissons :

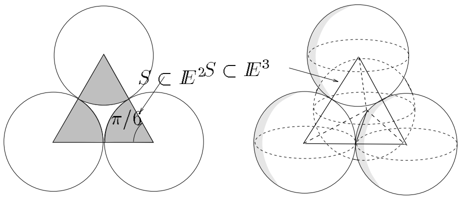
Définition 3.12 ***(Densité d'un billage)*** *Soit* $\mathcal{K}$ *un billage de* $\mathbb{R}^n$, *nous définissons la* densité du billage $\mathcal{K}$ *comme :*

$$\Delta_n(\mathcal{K}) = \lim_{s \to \infty} \nu(C_s)^{-1} \sum_{p_i : Bl(p_i,r) \subset C_s} \nu(Bl(p_i,r)) \tag{1}$$

1. fermé et borné.

La figure 2 montre deux billages en $\mathbb{E}^2$ avec des densités différentes.

La meilleure estimation de la *borne supérieure* pour un billage dans $\mathbb{E}^n$ notée σ_n a été établie par Rogers [ROG 64]. Cette borne est définie comme la proportion du volume d'un simplexe régulier S de côté 2 dans $\mathbb{E}^n$, lequel est couvert par les $(n+1)$ boules de rayon 1, centrées aux sommets de S. On note σ_n la densité de Rogers. La figure 3 représente les simplexes réguliers pour $\mathbb{E}^2$ et $\mathbb{E}^3$. Par exemple, dans le cas du simplexe régulier en $\mathbb{E}^2$ il est très facile de calculer la densité de Rogers. L'aire du triangle est $\sqrt{3}$. L'aire de chacun des secteurs (marqués en gris dans la figure 3) est $\pi/6$. Ainsi, l'aire des trois secteurs est $3\pi/6 = \pi/2$. La densité de Rogers pour $\mathbb{E}^2$ est alors $\pi/\sqrt{12} \approx 0.9069$ [SAA 70].



a b

FIG. 3 - *Un simplex de $\mathbb{R}^2$ et de $\mathbb{R}^3$.*

A partir de la borne supérieure de la densité de Rogers nous pouvons introduire la borne supérieure de la *densité centrale* de Rogers [SCH 72] notée σ'_n et définie comme :

$$\sigma'_n = \sigma_n / J_n \tag{2}$$

où J_n est le volume d'une boule de rayon 1 dans $\mathbb{E}^n$. Le volume J_n est égal à :

$$J_n = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}$$

où la fonction Γ est définie comme suit :

$$\Gamma(a) = \int_0^{\infty} e^{-x} x^{a-1} dx$$

Ainsi, pour une boule de rayon r en $\mathbb{E}^n$ nous avons :

$$\nu(Bl(x,r)) = r^n J_n$$

Pour un ensemble $X \subset \mathbb{R}^n$ nous notons $s(X) = \min\{s \in \mathbb{R} | X \subset C_s\}$, c'est-à-dire que $S(X)$ est la taille du côté du plus petit cube de côtés parallèles aux axes contenant X.

On remarque ainsi que si nous avons un billage $\mathcal{K} = U(EL, r)$ d'un espace $X \subset \mathbb{R}^n$ avec la propriété $EL \subset X$ alors $\mathcal{K} \subset C_{(s(X)+2r)}$(voir figure 4). Nous pouvons alors montrer la proposition suivante :

Proposition 3.1 ***(Borne supérieure du cardinal d'un billage)*** *Soit $X \subset \mathbb{R}^n$ et $\mathcal{K} = U(EL, r)$ un billage sphérique tel que $EL \subset X$ alors la borne supérieure de la*

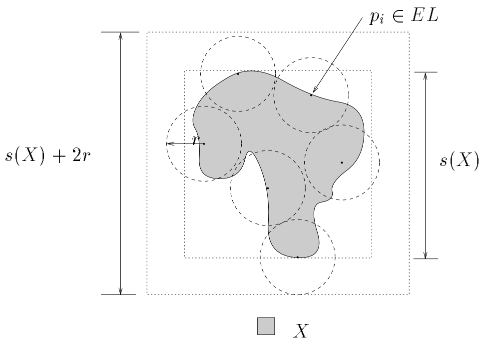


FIG. 4 - *Billage d'un espace X et le cube $C_{s(X)}$.*

cardinalité de EL est donnée par :

$$Card(EL) \leq \left\lfloor \frac{\sigma'_n * (s(X) + 2r)^n}{r^n} \right\rfloor \tag{3}$$

Démonstration :

Soit $\mathcal{K}' = U(EL', r)$ le billage de cardinal maximal de $C_{s(X)}$ tel que $EL' \subset C_{s(X)}$. La densité Δ de ce billage est :

$$\Delta = \frac{Card(EL') * J_n * r^n}{(s(X) + 2r)^n} \equiv \frac{volume\ des\ boules}{volume\ du\ cube}$$

Etant donné que $X \subset C_{s(X)}$ nous avons : $Card(EL) \leq Card(EL')$.

Définition 3.13 ***(Pavage)*** *Soient $EL = (p_i)_{i \in \mathbb{N}^*}$ une séquence de points de $\mathbb{R}^n$ et $r \in \mathbb{R}$ alors $P = U(EL, r)$ est appelé un* pavage *de X si et seulement si $X \subset P$. Nous appelons EL ensemble de balises et r rayon du pavage.*

Une conséquence immédiate de la définition d'un pavage est que $\rho(EL, X) \leq r$ et que par conséquent EL est une approximation de X à r-près au sens de la distance de Hausdorff.

Dans la section qui suit, nous présentons une méthode de construction d'un ensemble de points $EL \subset \mathcal{C}_{libre}$ qui réalise à la fois un billage $\mathcal{K} = U(EL, \varepsilon/2)$ et un pavage $P = U(EL, \varepsilon)$ de l'espace libre. Le pavage P de $\mathcal{C}_{libre}$ nous permet de montrer que EL est une approximation de $\mathcal{C}_{libre}$ et le billage K nous permet d'affirmer que cette approximation peut être obtenue en un nombre fini d'étapes.

4 L'algorithme $EXPLORE$

4.1 L'algorithme $EXPLORE_\infty$

L'algorithme $EXPLORE_\infty$ est utilisé afin de construire incrémentalement un pavage $\mathcal{K}$ pour tout espace compact $X \subset \mathbb{R}^n$. Le principe de la méthode consiste à placer des *balises* dans l'espace X en commençant par un point initial donné $x_\circ \in X$. Ainsi, la première balise L_1 de l'ensemble de balises EL est le point $x_\circ$. Puis, $EXPLORE_\infty$ sélectionne le point $p \in X$ le plus éloigné et qui devient alors une nouvelle balise de EL. On continue cette procédure en sélectionnant à chaque itération le point de X le plus éloigné possible des balises précédemment posées. On peut remarquer que cela conduit à diminuer la distance d'Hausdorff entre l'espace X et l'ensemble de points EL. De cette façon, si une balise est posée à une distance r des balises précédemment posées on est certain de réaliser un billage de l'espace libre avec des boules de rayon $r/2$. En utilisant la densité de Rogers on montre que le nombre de balises espacées de r est borné et que, dans un espace borné, si le nombre de balises augmente indéfiniment le rayon r ne peut que diminuer ce qui conduit nécessairement à une approximation de plus en plus fine de l'espace libre.

L'algorithme $EXPLORE_\infty$ est présenté ci-dessous. Il a pour paramètres d'entrée la valeur de résolution souhaitée $\varepsilon \in \mathbb{R}$, et $x_\circ \in X$ le point de départ. Ainsi, l'algorithme $EXPLORE_\infty$ nous donne EL^ε, l'ensemble des balises du pavage à $\varepsilon-$près de X.

L'algorithme est le suivant :

$ALGORITHME\_EXPLORE_\infty(x_\circ, \varepsilon)$
begin
$L_1 = x_\circ;$
$EL_1 = \{L_1\}$
$t = 1;$
do
  $p_t = p : \max_{p \in X} d(EL_t, p);$
  $L_{t+1} = p_t;$ /* on choisi la nouvelle balise */
  $EL_{t+1} = EL_t \cup \{L_{t+1}\};$
  $\varepsilon_t = d(EL_t, p_t);$
  $t = t + 1$
while($\varepsilon_{t-1} > \varepsilon$);
$EL^\varepsilon = EL_{t-1}$;
return(EL^ε);
end

L'algorithme commence de la manière suivante : soit $x_\circ \in X$ le point initial donné nous initialisons $EL_1 = \{L_1\}$ où $L_1 = x_\circ$, et nous appelons p_1 le point le plus éloigné de $x_\circ$.

$$p_1 : \max_{p \in X} d(L_1, p)$$

En accord avec notre définition $L_2 = p_1$ est le point le plus éloigné dans X de L_1. On pose $EL_2 = EL_1 \cup \{L_2\}$ et $\varepsilon_1 = \max_{p \in X} d(L_1, p)$ pour finir la première itération. Il est clair que $U(EL_2, \varepsilon_1/2)$ est un billage de X et que $U(EL_2, \varepsilon_1)$ est un pavage de X, c'est-à-dire $X \subset U(EL_2, \varepsilon_1)$. Nous continuons avec la deuxième itération de l'algorithme : étant donné un point $p \in X$ nous considérons le minimum entre $d(L_1, p)$ et $d(L_2, p)$ c'est-à-dire $d(EL_2, p)$ et nous essayons de maximiser cette valeur entre tous les points de X :

$$p_2 : \max_{p \in X} d(EL_2, p) = \max_{p \in X} \min_{L_i \in EL_2} d(L_i, p)$$

La troisième balise est alors $L_3 = p_2$ et $\varepsilon_2 = \max_{p \in X} d(EL_2, p)$. De même qu'à la première itération, $U(EL_3, \varepsilon/2)$ et $U(EL_3, \varepsilon_2)$ représente respectivement un billage et un pavage de X. D'une manière générale, pour un ensemble EL_t avec t balises nous cherchons à optimiser $\max_{p \in X} d(EL_t, p)$. Dès lors nous notons pour $t \in \mathbb{N}^*$:

$$p_t : \max_{p \in X} d(EL_t, p) = \max_{p \in X} \min_{L_i \in EL_t} d(L_i, p)$$

En posant $L_{t+1} = p_t$ nous obtenons notre $(t+1)^{ième}$ balise. Les ensembles $U(EL_{t+1}, \varepsilon_t/2)$ et $U(EL_{t+1}, \varepsilon_t)$ sont respectivement un billage et un pavage de l'ensemble X

Ainsi, chaque itération t se ramène à un problème d'optimisation. Etant donné un instant $t \in \mathbb{N}^*$, $EXPLORE_\infty$ est une fonction de la forme[2] :

$$EXPLORE_\infty(t) = \begin{cases} \text{Maximiser } d(EL_t, p) \\ \text{sous la contrainte :} \\ p \in X \end{cases}$$

Etant donné que X est un espace borné, nous avons le théorème suivant :

Théorème 4.1 ***(Convergence)*** *Soit X un espace compact, pour l'application de $EXPLORE_\infty$ en X :*

$$\lim_{t \to \infty} EXPLORE_\infty(t) = 0 \tag{4}$$

Démonstration :

Soit $t \in \mathbb{N}^*$ et $EXPLORE_\infty(t) = \varepsilon_t$, il est évident que $\mathcal{K}_{t+1} = U(EL_{t+1}, \varepsilon_t/2)$ est un billage de X. De cette manière, si $s = s(X)$ nous obtenons :

$$\sigma_n \geq \Delta_n(\mathcal{K}_{t+1}) = \frac{(t+1) * J_n * (\varepsilon_t/2)^n}{(s+\varepsilon_t)^n} \equiv \frac{\textit{volume des boules}}{\textit{volume du cube}}$$

ainsi pour $(1 - 2 * (\sigma'/(t+1))^{\frac{1}{n}}) > 0$, c'est-à-dire pour $t > \lceil 2^n \sigma'_n - 1 \rceil$, nous avons le résultat suivant :

$$\varepsilon_t \leq \frac{2s * (\frac{\sigma'_n}{t+1})^{\frac{1}{n}}}{1 - 2 * (\frac{\sigma'_n}{t+1})^{\frac{1}{n}}} \tag{5}$$

Par conséquent :

$$\lim_{t \to \infty} EXPLORE_\infty(t) \leq \lim_{t \to \infty} \frac{2s * (\frac{\sigma'_n}{t+1})^{\frac{1}{n}}}{1 - 2 * (\frac{\sigma'_n}{t+1})^{\frac{1}{n}}} = 0$$

Ce qui nous donne le résultat cherché.

---

2. Attention, il faut différencier la procédure $ALGORITHME\_EXPLORE_\infty(x_0, \varepsilon)$ de la fonction $EXPLORE_\infty(t)$

En utilisant le théorème 3.1 on peut donner le nombre d'itérations T qui rend $EXPLORE_\infty(T) < \varepsilon$.

Théorème 4.2 ***(Existance)***

$$Si \quad T > \left\lfloor \frac{\sigma_n' * (s(X) + \varepsilon)^n}{(\frac{\varepsilon}{2})^n} \right\rfloor - 1 \quad alors \quad EXPLORE_\infty(T) < \varepsilon. \tag{6}$$

De ce fait, nous pouvons réaliser un pavage de X dont le rayon tend vers 0 et donc approcher tout point de X à une distance ε arbitraire. Dans la suite de l'exposé, nous proposons un algorithme similaire pour la construction d'un pavage d'un sous-ensemble particulier de l'espace $\mathcal{C}_{libre}$: *l'espace libre à* $\varepsilon-$*près* noté $\mathcal{C}^{\varepsilon}_{libre}$. Cette construction est effectuée à partir d'une configuration initiale q_0 donnée en utilisant des trajectoires d'un type particulier : les chemins Manhattan.

4.2 Espace libre à ε-près et chemins Manhattan

Dans cette partie nous définissons la notion d'espace libre à $\varepsilon-$près. Intuitivement, on peut dire qu'une trajectoire dans cet espace passe toujours à plus de ε des $\mathcal{C}$-obstacles. Nous introduisons ensuite un type particulier de trajectoire : les chemins Manhattan d'ordre k. Sur un tel chemin on ne déplace qu'un degré de liberté à la fois. L'intérêt de ces chemins est double : ils sont facilement paramétrables et ils facilitent les calculs géométriques. Puis nous présentons l'algorithme $EXPLORE$ ainsi que les résultats principaux qui forment le cœur de cette section.

4.2.1 L'espace libre à $\varepsilon-$près

Définition 4.1 ***(Espace libre à*** $\varepsilon-$***près)*** *Etant donnés* $\varepsilon > 0$ *et la métrique* $d(x,y)$, l'espace libre à $\varepsilon-$près *de* $\mathcal{C}_{libre}$ *est défini par :*

$$\mathcal{C}^{\varepsilon}_{libre} = \{q \in \mathcal{C}_{libre} | d(q, \mathcal{CB}) \geq \varepsilon\}.$$

Un exemple de cet espace dans $\mathbb{R}^2$ est montré sur la figure 5. Evidemment $\mathcal{C}^{\varepsilon}_{libre}$ peut être composé d'une ou plusieurs composantes connexes bornées.

4.2.2 Les Chemins Manhattan

Soit $n \in N^*$. On note $\wp(n)$ l'ensemble des fonctions continues de I à valeurs dans $\mathbb{R}^n$. Autrement dit :

$$\hat{\varphi} \in \wp(n) \Longleftrightarrow \left\{ \begin{array}{l} \hat{\varphi} : [0,1] \to \mathbb{R}^n \\ \alpha \to [\varphi_1(\alpha), \ldots, \varphi_n(\alpha)] \end{array} \right.$$

où chaque fonction φ_i, $i = 1, 2, \ldots n$ est continue de $[0,1]$ dans $\mathbb{R}$. $\wp(n)$ est clairement un espace vectoriel réel.

Un *chemin* de $\mathbb{R}^n$ est, d'un "point de vue géométrique", l'ensemble image $\hat{\varphi}([0,1]) = \{\hat{\varphi}(\alpha), \alpha \in [0,1]\}$ pour $\hat{\varphi} \in \wp(n)$. Il est clair qu'un tel chemin $\hat{\varphi}$ en

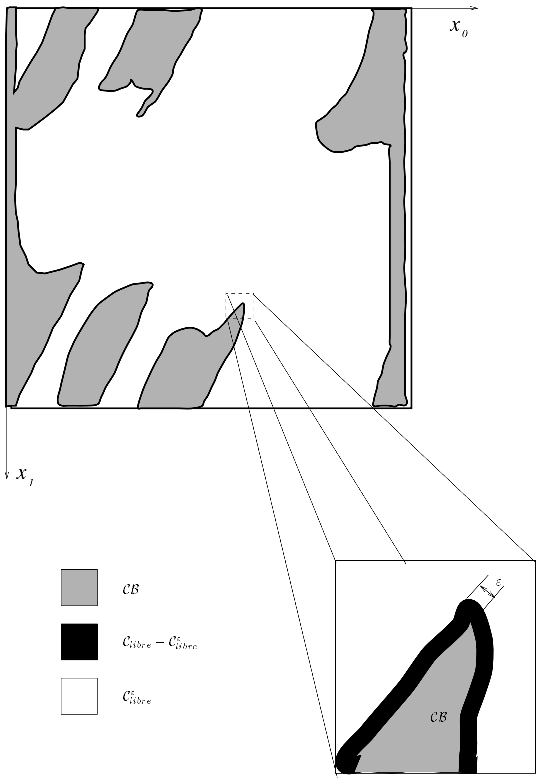


FIG. 5 - *L'espace* $\mathcal{C}^{\varepsilon}_{libre}$.

$I\!R^n$ admet plusieurs paramétrisations, par exemple $sin(\alpha * 2\pi) = cos(\alpha * 2\pi + \frac{\pi}{2})$ pour $\alpha \in I$. De façon plus précise, on définit :

Définition 4.2 ***(Relation d'équivalence, classe d'équivalence)***

1. *Soit* $\hat{\varphi}$ *et* $\hat{\tau} \in \wp(n)$. *Alors* $\hat{\varphi} \sim \hat{\tau}$ *si et seulement si :*
   (a) $\hat{\varphi}(0) = \hat{\tau}(0)$ *et* $\hat{\varphi}(1) = \hat{\tau}(1)$
   (b) $\exists f : [0,1] \to [0,1]$ *tel que* f *est continue et* $\forall \alpha \in [0,1]$, $\hat{\varphi}(\alpha) = \hat{\tau}(f(\alpha))$.

   $\hat{\varphi} \sim \hat{\tau}$ *définit une* relation d'équivalence *dans* $\wp(n)$.
2. *Un* chemin *de* $I\!R^n$ *est une* classe d'équivalence *dans* $\wp(n)$ *modulo la relation d'équivalence* $\hat{\varphi} \sim \hat{\tau}$, *c'est-à-dire un élément de* $\wp(n)/\sim$.

En pratique, nous identifions les chemins de $I\!R^n$ avec les fonctions de $\wp(n)$, c'est-à-dire avec leurs représentants.

Introduisons une norme sur l'espace vectoriel $\wp(n)$. Pour $\hat{\varphi} \in \wp(n)$, on pose :

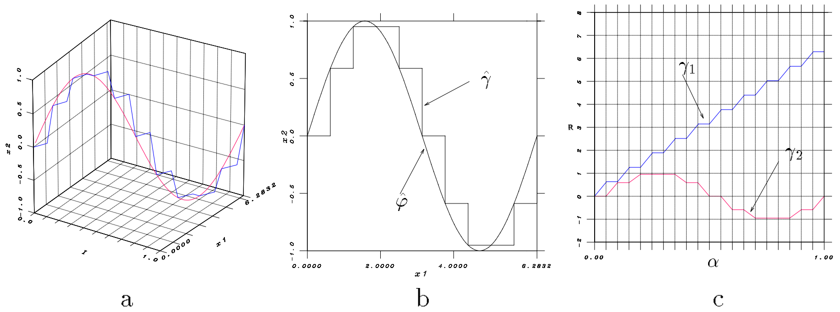


FIG. 6 - *Une fonction $\hat{\varphi} \in \wp(2)$, une autre $\hat{\gamma} \in \mathcal{M}(2)$ et ses images en $x_1 \times x_2$*

$$||\hat{\varphi}||_\infty = \max\{||\varphi_i||_\infty\} \quad \text{pour} \quad i \in \{1, 2, \ldots, n\}$$

où

$$||\varphi_i||_\infty = \sup\{|\varphi_i(\alpha)|_\infty\} \quad t \in [0, 1]$$

Ainsi, $(\wp(n), ||\ \ ||_\infty)$ devient un espace vectoriel normé.

Définition 4.3 ***(Fonction Manhattan)*** *On considère le sous-ensemble $\mathcal{M}(n)$ de $\wp(n)$ des fonctions*
$\hat{\gamma} : [0, 1] \to I\!R^n$, $\alpha \to (\gamma_1(\alpha), \ldots \gamma_n(\alpha))$ défini de la manière suivante :
$\hat{\gamma} \in \mathcal{M}(n)$ si et seulement s'il existe une subdivision

$$(c_j)_{j=0,1,\ldots,N} \quad \textit{de} \quad [0, 1] : 0 = c_0 < c_1 < \ldots < c_N = 1$$

telle que : $\forall j \in \{0, 1, \ldots, N-1\}, \exists k_j \in \{1, 2, \ldots n\}$ tel que la fonction numérique γ_{k_j} soit affine sur $[c_j, c_{j+1}]$ et chaque γ_i, $i \neq k_j$ soit constante sur $[c_j, c_{j+1}]$

Autrement dit sur $[c_j, c_{j+1}]$ seule la fonction γ_{k_j} peut être non constante (et affine). Une telle fonction $\hat{\gamma}$ est appelée fonction Manhattan*.*

Un exemple d'une fonction $\hat{\varphi} \in \wp(2)$ et une fonction Manhattan $\hat{\gamma} \in \mathcal{M}(2)$ est montré dans la figure 6a. Les images de deux fonctions, $\hat{\varphi}([0, 1])$ et $\hat{\gamma}([0, 1])$ sont montrées dans la figure 6b. comme nous l'avons défini précédemment la fonction $\hat{\varphi}$ est composée de deux fonctions continues $\gamma_1 : I \to I\!R$ et $\gamma_2 : I \to I\!R$, en plus seul un de deux fonctions peut être non constant dans chacune des intervalles formés par les subdivisions $(c_j)_{j=0,1,\ldots N}$, la figure 6c montre cette situation.

Comme précédemment, on définit géométriquement un *chemin Manhattan* comme l'ensemble image $\hat{\gamma}([0, 1])$ ou comme une classe d'équivalence[3] modulo

3. On peut montrer que $\mathcal{M}(n)$ est dense dans $\wp(n)$. Autrement dit, on peut approcher toute courbe paramétrée continue, c'est-à-dire tout chemin de $I\!R^n$, d'aussi près que l'on veut par un chemin Manhattan. Soit :

$$\forall \hat{\varphi} \in \wp(n), \forall \varepsilon > 0, \exists \hat{\gamma} \in \mathcal{M}(n) : ||\hat{\varphi} - \hat{\gamma}||_\infty < \varepsilon$$

la relation $\sim$.

Définition 4.4 ***(Chemin Manhattan d'ordre 1)*** *Soit un système à n degrés de liberté et $q_\circ = (x_1, \ldots, x_n)$ une configuration donnée de ce système, on définit un* chemin Manhattan d'ordre 1 *et partant de $q_\circ$ comme l'application $\hat{\gamma}^1 : \alpha \to I\!R^n$ où :*

$$\gamma_i(\alpha) = \begin{cases} x_i & pour \quad 0 \leq \alpha \leq \frac{(i-1)}{n} \\ x_i + \Delta_i * (n * \alpha - i) & pour \quad \frac{(i-1)}{n} \leq \alpha \leq \frac{i}{n} \\ x_i + \Delta_i & pour \quad \frac{i}{n} \leq \alpha \leq 1 \end{cases} \tag{7}$$

avec $\Delta_i \in I\!R$.

Ce type de trajectoire consiste à déplacer successivement un degré de liberté à la fois, chacun à son tour. De cette manière $\hat{\gamma}^1$ est noté comme :

$$\hat{\gamma}^1 = (\Delta_1^1, \Delta_2^1, \ldots, \Delta_i^1, \ldots, \Delta_n^1) \tag{8}$$

Définition 4.5 ***(Chemin Manhattan d'ordre k)*** *Soit un système à n degrés de liberté et $q_\circ = (x_1, \ldots, x_n)$ une configuration donnée de ce système. On définit* un chemin Manhattan d'ordre *k partant de $q_\circ$ comme la concaténation de k chemins Manhattan d'ordre* 1*. On note un tel chemin :*

$$\hat{\gamma}^k = \hat{\gamma}_1^1 * \hat{\gamma}_2^1 * \ldots * \hat{\gamma}_k^1 = (\Delta_1^1, \Delta_2^1, \ldots, \Delta_n^1, \ldots, \Delta_i^j, \ldots, \Delta_n^k) \tag{9}$$

Où Δ_i^j représente le mouvement relatif du degré de liberté i dans $\hat{\gamma}_j^1$. Ainsi, $\hat{\gamma}_0^1(0) = q_\circ, \hat{\gamma}_k^1(1) = q_\bullet$ et $\hat{\gamma}_{j-1}^1(0) = \hat{\gamma}_j^1(1)$ pour $j = 2, \ldots, k$. Par exemple, la figure 7 représente un chemin Manhattan d'ordre 5 pour un robot à deux degrés de liberté.

4.3 L'algorithme $EXPLORE$

Avant de définir l'algorithme $EXPLORE$ en utilisant des chemins Manhattan nous définissons la post-image produite par ce type de chemins.

Définition 4.6 ***(Ensemble de chemins Manhattan)*** *Soient $\mathcal{C}_{libre}$ l'espace des configurations d'un système à n degrés de liberté et $q \in \mathcal{C}_{libre}$ une configuration de cet espace. On note $\mathcal{M}(\mathcal{C}_{libre}; \ell, q)$* l'ensemble de chemins Manhattan *d'ordre $k \leq \ell$ où pour tout $\hat{\gamma}^k \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q)$, $\hat{\gamma}(0) = q$ et $\hat{\gamma}^k([0,1]) \in \mathcal{C}_{libre}$ et $\mathcal{M}(\mathcal{C}_{libre}; \ell, EL) = \bigcup_{q \in EL} \mathcal{M}(\mathcal{C}_{libre}; \ell, q)$*

Définition 4.7 ***(Post-image produite par les chemin Manhattan)*** *Etant donné un ensemble de balises EL de l'espace libre d'un système à n degrés de liberté nous définissons la* "post-image" *de EL produite par $\mathcal{M}(\mathcal{C}_{libre}; \ell, EL)$ comme :*

$$\mathcal{P}st_{\mathcal{M}}(\mathcal{C}_{libre}; \ell, EL) = \{q = \hat{\gamma}(1) | \hat{\gamma} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, EL)\}$$

c'est-à-dire l'ensemble de toutes les configurations en $\mathcal{C}_{libre}$ pour lesquelles il existe un chemin Manhattan d'ordre inférieur ou égal à ℓ et ayant comme point de départ une balise $L_i \in EL$.

Dans la suite nous omettrons l'indice "k" pour tout chemin Manhattan d'ordre $k \leq \ell$ en $\mathcal{M}(\mathcal{C}_{libre}; \ell, q)$. C'est-à-dire que nous l'écrirons uniquement $\hat{\gamma}$.

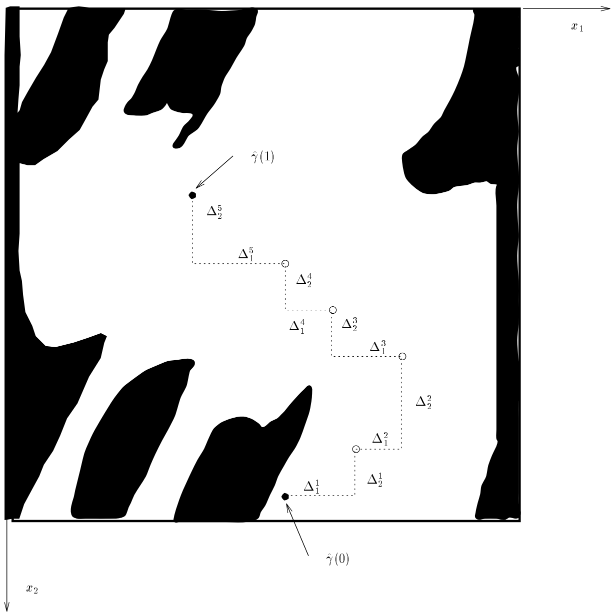


FIG. 7 - *Un chemin Manhattan dans l'espace des configurations*

4.3.1 Description de l'algorithme

Nous définissons l'algorithme $EXPLORE$ comme une variante de l'algorithme $EXPLORE_\infty$ La différence réside dans l'espace de recherche utilisé par la fonction d'optimisation de $EXPLORE$. Cet espace est la post-image engendrée par l'ensemble de trajectoires Manhattan d'ordre inférieur ou égale à une valeur ℓ donnée.

Si nous avons t balises et que nous appelons EL_t l'ensemble de ces balises, nous notons :

$$q_t : \max_{q \in \mathcal{P}st_{\mathcal{M}}(\mathcal{C}_{libre};\ell,EL_t)} d(q, EL_t) = \max_{q \in \mathcal{P}st_{\mathcal{M}}(\mathcal{C}_{libre};\ell,EL_t)} \min_{L_i \in EL_t} d(L_i, q)$$

la configuration q_t définit la nouvelle balise $L_{t+1} = q_t$. Nous arrivons alors à la définition de l'algorithme $EXPLORE$ sous la forme d'un problème d'optimisation :

$$EXPLORE(t) = \begin{cases} \text{Maximiser } d(EL_t, q) \\ \text{sous la contrainte :} \\ q \in \mathcal{P}st_{\mathcal{M}}(\mathcal{C}_{libre}; \ell, EL_t) \end{cases}$$

Théorème 4.3 ***(Exploration)*** *Soit $q_\bullet, q_\circ \in \mathcal{C}^\varepsilon_{libre}$ nous avons la propriété suivante pour l'application de $EXPLORE$: Si $EXPLORE(t) < \varepsilon \quad \wedge \quad q_\bullet \notin U(EL_t, \varepsilon)$ alors $\mathcal{C}^\varepsilon_{libre_{q_\circ}} \neq \mathcal{C}^\varepsilon_{libre_{q_\bullet}}$.*

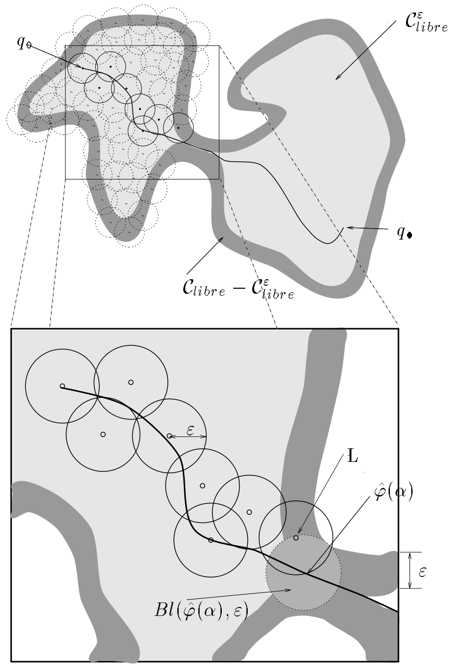


FIG. 8 - *Pavage de $\mathcal{C}_{libre}$*

Démonstration par l'absurde :

Prémisses :

$$EXPLORE(t) < \varepsilon \equiv \forall q \in \mathcal{P}st_{\mathcal{M}}(\mathcal{C}_{libre}; \ell, EL_t), \quad d(q, EL_t) < \varepsilon \tag{10}$$

$$q_\bullet \notin U(EL_t, \varepsilon) \equiv d(q_\bullet, EL_t) > \varepsilon \tag{11}$$

Hypothèse :

$$\mathcal{C}^{\varepsilon}_{libre_{q_\circ}} = \mathcal{C}^{\varepsilon}_{libre_{q_\bullet}} \tag{12}$$

Nous déduisons de 12 que :

$$\exists \hat{\varphi} : I \to \mathcal{C}^{\varepsilon}_{libre_{q_\circ}} \quad \textit{tel que} \quad \hat{\varphi}(0) = q_\circ, \hat{\varphi}(1) = q_\bullet \tag{13}$$

Soit $\Lambda = \{\alpha \in I | d(EL_t, \hat{\varphi}(\alpha)) = \varepsilon\}$, nous notons par α_t la valeur maximale de Λ, c'est-à-dire :

$$\alpha_t = \max \alpha \in \Lambda$$

Notons maintenant $L = (x_1, x_2, \ldots, x_n)$ la balise la plus proche de $\hat{\varphi}(\alpha_t) = (x'_1, x'_2, \ldots, x'_n)$. Il est facile de voir que $d(L, \hat{\varphi}(\alpha_t)) = \varepsilon$ car $d(q_\bullet, EL_t) > \varepsilon$. (La trajectoire traverse un ou plusieurs fois la boule $Bl(L, \varepsilon)$, voir figure 8)

D'autre part nous avons de 13 que $Bl(\hat{\varphi}(\alpha_t), \varepsilon) \in \mathcal{C}_{libre}$. Ainsi, il existe un chemin Manhattan $\hat{\gamma}_t$ en $\mathcal{C}_{libre}$ d'ordre 1 de L à $\hat{\varphi}(\alpha_t)$ tel que :

$$\hat{\gamma}_t = (x'_1 - x_1, x'_2 - x_2, \ldots, x'_n - x_n)$$

De ce fait, $\exists q_\alpha = \hat{\varphi}(\alpha) \in \mathcal{P}st_{\mathcal{M}}(\mathcal{C}_{libre}; \ell, EL_t)$ tel que $d(q_\alpha, EL_t) = \varepsilon$, ce qui est en contradiction avec 10.

La valeur de $EXPLORE(t)$ nous renseigne en fait sur la topologie de l'espace libre. En effet, si à l'instant t on n'a pas placé une balise à moins de $EXPLORE(t)$ de la position but cela veut dire qu'il faudra passer par un "couloir" de l'espace des configurations de diamètre inférieur ou égal à $EXPLORE(t)$ pour atteindre le but.

Nous pouvons conclure du théorème précédent que la difficulté de construire un chemin entre deux configurations $q_\circ$,$q_\bullet$ appartenant à la même composante connexe de C_{libre} avec l'algorithme $EXPLORE$ est liée à la valeur ε^* où :

$$\varepsilon^* = max\{\varepsilon \in I\!R | \mathcal{C}^{\varepsilon}_{libre_{q_\circ}} = \mathcal{C}^{\varepsilon}_{libre_{q_\bullet}}\}$$

Ainsi, pour $\varepsilon \leq \varepsilon^*$, si $EXPLORE(t) \leq \varepsilon$ alors $d(EL_t, q_\bullet) \leq \varepsilon$. Autrement dit, plus ε^* est grand plus $EXPLORE$ trouvera rapidement une solution. Enfin, en utilisant le résultat de la convergence sur $EXPLORE_\infty$ on montre facilement que $\lim_{t\to\infty} EXPLORE(t) = 0$ et donc que l'on peut approcher tout point de $C^{\varepsilon}_{libre_{q_\bullet}}$ en un nombre fini d'itérations.

Théorème 4.4 *(Convergence)*

$$\lim_{t\to\infty} EXPLORE(t) = 0 \tag{14}$$

Démonstration :

Par définition $\forall t$ $EXPLORE(t) \leq EXPLORE_\infty(t)$ ce qui d'après le théorème 4.1 nous donne le résultat.

5 Améliorations à l'algorithme $EXPLORE$

5.1 L'algorithme $SEARCH$

5.1.1 Définition

Dans le chapitre précédent nous avons défini $\mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ comme l'ensemble des chemins Manhattan dans l'espace libre partant de $q_\circ$. Ainsi, nous pouvons maintenant définir une fonction $F : \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ) \to I\!R$ prenant pour valeur la distance Euclidienne de l'extrémité d'une trajectoire $\hat{\gamma}_{q_\circ} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ à une configuration $q_\bullet \in \mathcal{C}_{libre}$:

$$F(\hat{\gamma}_{q_\circ}, q_\bullet) = \|\hat{\gamma}_{q_\circ}(1) - q_\bullet\|$$

Nous pouvons donc exprimer le problème de la recherche d'un chemin Manhattan sans collision d'une configuration $q_\circ$ à une autre $q_\bullet$ comme un problème d'optimisation [AMB 93]. Ce problème s'exprime comme :

$$SEARCH(q_\circ, q_\bullet) = \begin{cases} \text{Minimiser } F(\hat{\gamma}_{q_\circ}, q_\bullet) \\ \text{sous la contrainte:} \\ \hat{\gamma}_{q_\circ} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ) \end{cases}$$

S'il existe un chemin Manhattan $\hat{\gamma}_{q_\circ} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ reliant la configuration initiale à la configuration finale alors :

$$SEARCH(q_\circ, q_\bullet) = 0$$

Bien qu'il n'existe pas de forme analytique de F, nous pouvons calculer $F(\hat{\gamma}_{q_\circ}, q_\bullet)$ pour tout $\hat{\gamma}_{q_\circ} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ et utiliser une méthode d'optimisation pour trouver un optimum (global ou local) à cette fonction. Etant donnée une méthode d'optimisation, on peut alors définir implicitement la pré-image de $q_\bullet$ produite par notre planificateur comme :

$$\mathcal{P}_{\mathcal{L}}(q_\bullet) = \{q \in \mathcal{C}_{libre} | SEARCH(q, q_\bullet) = 0\}$$

La dimension de cette pré-image dépend à la fois du type d'algorithme d'optimisation utilisé et de la position du but $q_\bullet$. Dans le paragraphe suivant nous proposons une amélioration de cette fonction. Elle permet d'agrandir cette pré-image et donc d'augmenter encore l'efficacité de EXPLORE.

5.1.2 La "pré-image Manhattan" d'ordre 1

Soit un point final $q_\bullet$ dans l'espace des configurations d'un système à n degrés de liberté. Il est possible de définir une région où cette configuration peut être atteinte avec des déplacements "simples" [ERD 86, LMT 84], c'est-à-dire une région où $q_\bullet$ est "visible". Il existe plusieurs manières de définir les pré-images. Nous définissons un type particulier de pré-image: *la "pré-image Manhattan"* d'ordre 1. Cette pré-image est définie grâce au prédicat MP. Soit $q_a, q_b \in \mathcal{C}_{libre}$ deux configurations d'un système à n degrés de liberté; *le prédicat* $MP(q_a, q_b)$ est vrai si et seulement s'il existe un chemin $\hat{\gamma}_{q_a} \in \mathcal{M}(\mathcal{C}_{libre}; 1, q_a)$ tel que $\hat{\gamma}_{q_a}(1) = q_b$.

Définition 5.1 ***(Prè-image produite par les chemins Manhattan)*** *Pour une configuration* $q_a \in \mathcal{C}_{libre}$ *la* pré-image Manhattan *de* q_a *est définie par:*

$$\mathcal{P}_{\mathcal{M}}(q_a) = \{q \in \mathcal{C}_{libre} | MP(q, q_a) = vrai\} \tag{15}$$

La figure 9 montre la pré-image pour deux configurations $q_a, q_b \in C_{libre} \subset I\!R^2$. Notons que la pré-image de q_a est beaucoup plus petite que celle de q_b.

Evidemment, une trajectoire de $q_\circ$ à $q_\bullet$ existe s'il existe une trajectoire $\hat{\gamma}_{q_\circ} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ telle que $d(\hat{\gamma}_{q_\circ}(1), \mathcal{P}_{\mathcal{M}}(q_\bullet)) = 0$. Nous pouvons alors définir la fonction F comme suit :

$$F(\hat{\gamma}_{q_\circ}, q_\bullet) = \begin{cases} 0 & \text{si } \hat{\gamma}_{q_\circ}(1) \in \mathcal{P}_{\mathcal{M}}(q_\bullet) \\ d(\hat{\gamma}_{q_\circ}(1), q_\bullet) & \text{autrement} \end{cases}$$

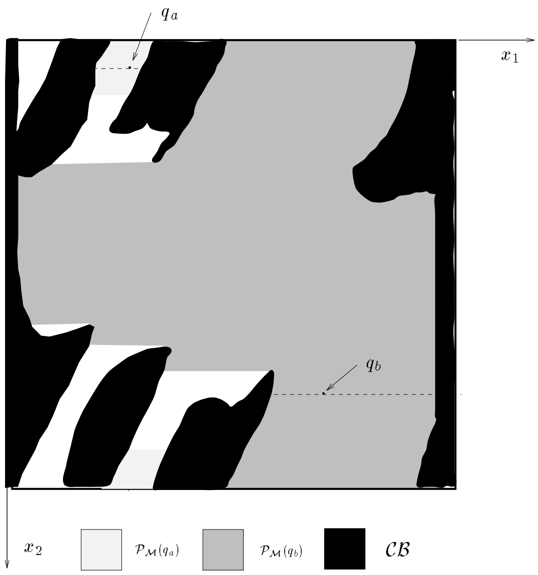


FIG. 9 - *La pré-image de deux points dans $\mathcal{C}_{libre}$*

5.2 Génération efficace de chemins Manhattan de $\mathcal{C}_{libre}$

Si nous voulons appliquer les algorithmes $SEARCH$ et $EXPLORE$, nous devons être capables d'engendrer des trajectoires Manhattan d'ordre $k \leq \ell$ dans l'espace libre d'un système à n degrés de liberté. Dans la suite de cette section nous décrivons une technique pour engendrer de telles trajectoires à partir d'un vecteur $x \in I\!R^{n*\ell}$.

5.2.1 L'ensemble propre d'une trajectoire Manhattan

Nous avons vu dans la section précédente que tout chemin Manhattan $\hat{\gamma} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ d'ordre $k \leq \ell$ pour un robot à n degrés de liberté est codé de la manière suivante :

$$\hat{\gamma} = (\Delta_1^1, \Delta_2^1, \ldots, \Delta_n^1, \ldots, \Delta_i^j, \ldots, \Delta_n^k) \tag{16}$$

c'est-à-dire que nous avons $k*\ell$ configurations intermédiaires $\{q_1, q_2, \ldots, q_{k*\ell}\}$. Nous définissons de cette façon :

Définition 5.2 ***(Ensemble propre d'un chemin Manhattan)*** *Soit $\hat{\gamma}$ un chemin Manhattan d'ordre $k \leq \ell$ de $\mathcal{M}(\mathcal{C}_{libre}; \ell, q_0)$. On définit* l'ensemble propre *de $\hat{\gamma}$, noté $Q(\hat{\gamma}) = \{q_0, q_1, \ldots, q_{k*\ell}\}$, comme l'ensemble des configurations intermédiaires de $\hat{\gamma}$ où*

$$q_{m-1} = (x_1, x_2, \ldots, x_i, \ldots, x_n), \quad q_m = (x_1, x_2, \ldots, x_i + \Delta_i^j, \ldots, x_n)$$
$$\textit{avec} \quad m = n*(j-1)+i$$

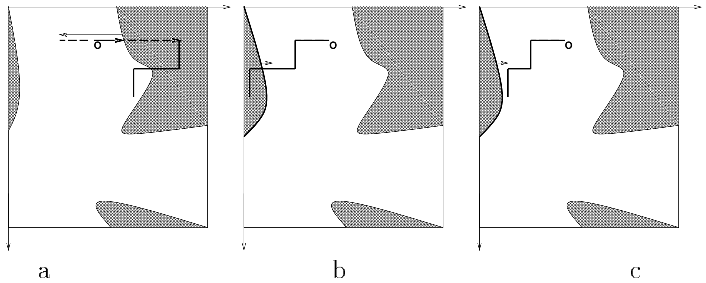


Fig. 10 - *Le concept de rebondissement*

5.2.2 Le "rebondissement" des trajectoires

Nous pouvons remarquer que toute trajectoire $\hat{\gamma} \in \mathcal{M}(\mathcal{C}_{libre};, \ell, q_\circ)$ est un vecteur en $I\!R^{n*\ell}$ mais que le cas contraire n'est pas forcément vrai.

Par conséquent, $\mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ est un sous-ensemble de $I\!R^{n*\ell}$ et évidemment l'ensemble $I\!R^{n*\ell} - \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$ code des trajectoires avec collision. Dans la suite nous proposons une sémantique capable d'engendrer à partir de tout vecteur $\hat{x} \in I\!R^{n*\ell}$ une trajectoire Manhattan $\hat{\gamma} \in \mathcal{M}(\mathcal{C}_{libre}; \ell, q_\circ)$. Pour cela nous considérons que la trajectoire "rebondit" sur les *C-obstacles*. Ainsi, la trajectoire originale de la figure 10a se transforme-t-elle en la trajectoire de la figure 10b lors de la première collision et en la trajectoire 10c lors de la deuxième collision.

Notons $\hat{x} = (x_1^1, x_2^1, \ldots x_n^1, \ldots, x_n^k) \in I\!R^{n*k}$. La procédure de transformation d'un vecteur $\hat{x}$ en une trajectoire Manhattan est la suivante :

On calcule à partir de q_0 le débattement $A_0 = [x_1^{min}, x_1^{max}] \in \mathcal{C}_{libre}$ parallèle à l'axe x_1 et à travers q_0. La valeur Δ_1^1 est alors obtenue en fonction de q_0 et x_1^1; $\Delta_1^1 = rebond(q_0, x_1^1)$. On obtient alors $q_1 \in A_0$ tel que $q_1 = q_0 + (\Delta_1^1, 0, 0, \ldots .0)$. On exécute la même procédure pour q_1 afin d'obtenir q_2, et ainsi de suite jusqu'à la complète obtention de l'ensemble propre de $\hat{\gamma}$.

L'idée générale de la fonction *rebond* est de trouver la configuration q_m, atteinte en partant d'une configuration q_{m-1} et en se déplaçant d'une distance x_i^j dans un intervalle A_m (voir figure 11). Si au moment de parcourir une distance $\delta < x_i^j$ on arrive à x_i^{max} ou x_i^{min}, la trajectoire "rebondit" et continue dans le sens inverse. On exécute alors cette même procédure pour le reste de la valeur x_i^j, c'est-à-dire pour $(x_i^j - \delta)$.

Avec cette technique tout vecteur $x \in I\!R^{n*\ell}$ code une trajectoire Manhattan de longueur $k \leq \ell$ dans l'espace libre de $\mathcal{C}$. Ainsi, nous pouvons coder toutes les trajectoires Manhattan de longueur $k \leq \ell$ avec $I\!R^{n*\ell}$. D'une manière similaire il est possible avec l'espace $[1, 2, \ldots, t] \times R^{n*\ell}$ de coder toutes les trajectoires de $\mathcal{M}(\mathcal{C}_{libre}; \ell, EL_t)$. Dans ce cas, la valeur $k \in [1, 2, \ldots, t]$ indique la balise L_k de départ et $x \in I\!R^{n*\ell}$ la trajectoire Manhattan.

Ainsi, les problèmes d'optimisation posés par les algorithmes *SEARCH* et *EXPLORE* peuvent être reécrits comme :

$$SEARCH(q_\circ, q_\bullet) = \begin{cases} \text{Minimiser } F(\hat{\gamma}_{q_\circ}, q_\bullet) \\ \hat{\gamma}_{q_\circ} \in I\!R^{n*\ell} \end{cases}$$

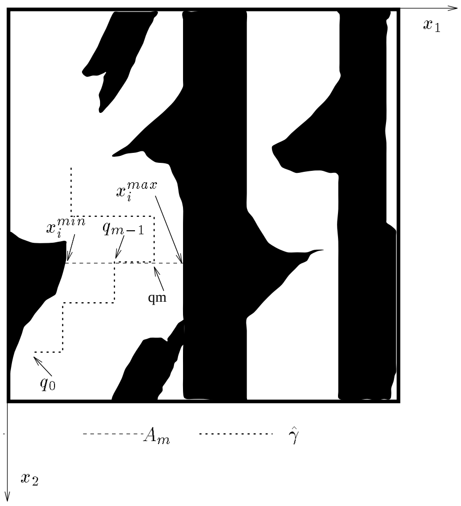


FIG. 11 - *Le segment A_m calculé a partir d'une configuration q_{m-1}*

$$EXPLORE(t) = \begin{cases} \text{Maximiser } d(EL_t, \hat{\gamma}(1)) \\ \hat{\gamma} \in [1, 2, \ldots, t] \times I\!R^{n*\ell} \end{cases}$$

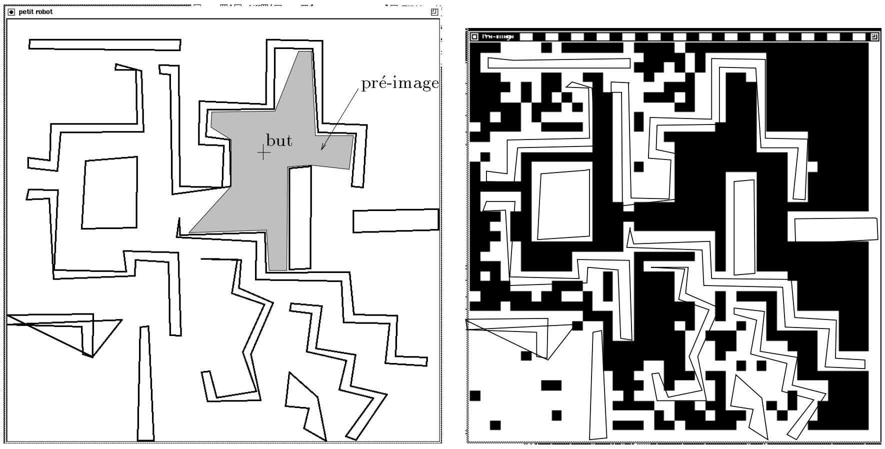
Nous obtenons finalement l'Algorithme Fil d'Ariane comme la combinaison de deux problèmes d'optimisation :

DEBUT : $L_1 \leftarrow q_0, EL \leftarrow \{L_1 = q_0\}, t \leftarrow 1$

1. EXECUTION DE SEARCH : Si $SEARCH(L_t, q_\bullet) = 0$ alors un chemin a été trouvé, sinon prendre la balise L_{t+1} comme nouvelle configuration de départ.
2. EXECUTION D'EXPLORE : Placer une nouvelle balise L_{t+1} avec la fonction $EXPLORE(t)$, $EL \leftarrow EL \cup \{L_{t+1}\}$, $y \leftarrow t+1$.

6 Exemple d'utilisation pour un robot mobile holonome

L'application principale que nous avons développée de l'algorithme Fil d'Ariane est un système parallèle pour la planification de trajectoires en "temps réel" de deux robots à six degrés de liberté [BES 93, ATC 93, MAT 93, MAT 93]. Néanmoins, dans cet article nous présentons l'utilisation de l'algorithme Fil d'Ariane pour la planification de trajectoires d'un robot mobile holonome étant donné que cet exemple est plus facile à comprendre et plus didactique. Rappelons que dans ce problème il s'agit de trouver un chemin sans collision pour un véhicule pouvant tourner sur place. Dans la maquette réalisée, l'utilisateur peut modifier les positions des obstacles au cours du déplacement du véhicule. Notre objectif est alors de replanifier immédiatement une nouvelle trajectoire vers le but sans interrompre le mouvement du robot. L'objectif visé est de montrer la rapidité du planificateur proposé et la possibilité de construire un planificateur global réactif.



a b

FIG. 12 - *La pré-image des mouvements "directs" au but et la pré-image $\mathcal{P}_{\mathcal{L}}$.*

6.1 Description et résultats expérimentaux

6.1.1 L'implantation de l'algorithme *SEARCH*

Dans cette application nous utilisons une version modifiée des chemins Manhattan précédemment décrits. Nous considérons un sous-ensemble discret de tous les chemins partant d'une configuration donnée $q_\circ$ et d'une longueur inférieure ou égale à d. Un chemin $\hat{\gamma}$ est codé grâce à k entiers positifs $\{\theta_i\}_{i=0,k-1}$ qui codent l'angle entre deux segments linéaires de longueur égale à $\frac{d}{k}$. Dans un espace complètement libre, ce type de trajectoire est seulement un ensemble de segments connectés de longueur $\frac{d}{k}$. Notons q_i le point initial de chacun des segments. Selon notre définition le robot tourne en θ_i au point q_i; ensuite, il exécute un mouvement en ligne droite vers q_{i+1}.

Dans un espace encombré d'obstacles ce codage peut aussi représenter une trajectoire sans collision à condition d'utiliser la notion de "rebond". Lorsqu'un segment $\{q_i, q_{i+1}\}$ rentre en collision avec un obstacle, la valeur de θ_i est incrémentée ou décrémentée jusqu'à trouver une valeur θ_i' pour laquelle cette collision disparaît. On dit que la trajectoire "rebondit" sur l'obstacle. En utilisant cette technique, un vecteur d'entiers de dimension k représente une trajectoire d'ordre k et de longueur d. On peut remarquer que cette technique de codage permet une recherche dans l'espace de chemins réalisables formés d'une séquence de rotations et de déplacements de longueur fini.

Pour tout chemin $\hat{\gamma}$ et donc pour tout vecteur de $\mathbb{N}^k$ nous définissons la fonction $F(\hat{\gamma}, q_\bullet)$ comme suit : $F(\hat{\gamma}, q_\bullet) = 0$ s'il existe un mouvement "direct" d'une des configurations $\{q_i\}_{i=0,k-1}$ au but et $F(\hat{\gamma}, q_\bullet) = \min_{i=0,k-1} ||q_i - q_\bullet||$ sinon. Bien qu'il n'existe pas de forme analytique pour $F(\hat{\gamma}, q_\bullet)$, un algorithme génétique [GOL 89, AMB 93] peut être utilisé pour minimiser cette fonction de $\mathbb{N}^k$ dans $\mathbb{R}^+$.

L'ensemble des configurations pour lesquelles un mouvement direct au but est possible définit implicitement une pré-image (voir figure 12a). Une autre pré-image,

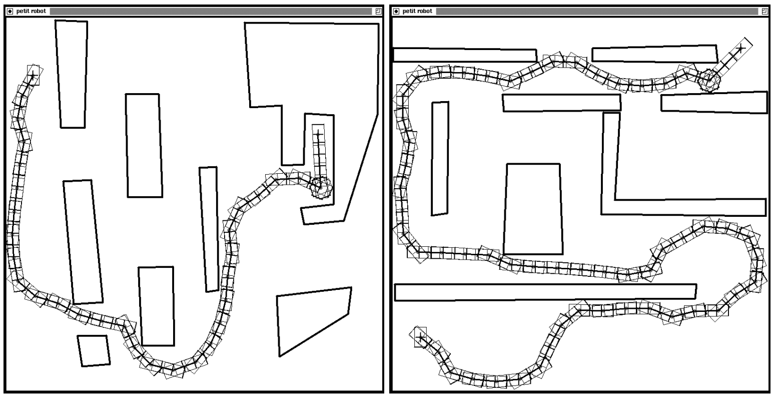


FIG. 13 - *Deux chemins trouvés par $SEARCH$.*

de taille plus grande, est définie implicitement par la fonction $F(\hat{\gamma}, q_\bullet)$ et les algorithmes génétiques. Cette pré-image est définie comme l'ensemble des configurations $\mathcal{P}_\mathcal{L}$ où pour tout $q \in \mathcal{P}_\mathcal{L}$ l'algorithme génétique est capable de trouver un chemin $\hat{\gamma}$ ("rotation" et "déplacement") partant de q et tel que $F(\hat{\gamma}, q_\bullet) = 0$.

A titre d'exemple, et pour montrer que cette pré-image peut couvrir une partie non négligeable de l'espace libre, nous présentons dans la figure 12b la pré-image engendrée par $SEARCH$ en utilisant des trajectoires d'ordre 10 avec une longueur de segment égale à trois fois la longueur du robot. La région noire représente toutes les configurations de départ pour lesquelles une trajectoire a été trouvée vers la configuration finale montrée dans la figure 12a. Cette région correspond donc à la pré-image $\mathcal{P}_\mathcal{L}$ précédemment définie. On peut remarquer en particulier que les obstacles n'appartiennent pas à cette pré-image.

Ainsi, partant d'une configuration initiale, il suffit de placer une balise dans cette pré-image pour trouver une trajectoire. La dimension de la pré-image est une bonne indication du nombre de balises que devra déposer l'algorithme $EXPLORE$ pour trouver une solution. Plus elle est grande plus les chances d'y placer rapidement une balise avec $EXPLORE$ est élevée.

Dans l'implantation de $SEARCH$ nous utilisons des trajectoires formées de six rotations et six déplacements. Les déplacements représentent toujours trois fois la longueur du véhicule. Ces paramètres sont définis expérimentalement. Ainsi, une trajectoire pour l'algorithme $SEARCH$ est représentée de la manière suivante :

$$(\theta_1, \ldots, \theta_i, \ldots, \theta_6)$$

où chacun des θ_i est codé par un vecteur S_i de 7 bits. Un chemin, comme ceux montrés dans la figure 13, est trouvé en 7 générations pour une population de l'algorithme génétique de 25 individus et dans un temps inférieur à 0.05 secondes pour une station de travail Sparc 2.

Lorsque le robot mobile est en train d'exécuter une trajectoire, l'utilisateur peut, en utilisant le simulateur, changer la position des obstacles. Le planificateur réagit alors immédiatement et recalcule une nouvelle trajectoire. Par exemple, si la porte représentée dans la figure 14 est fermée avant que le robot ne la franchisse, le planificateur est capable de reconstruire instantanément une nouvelle trajectoire.

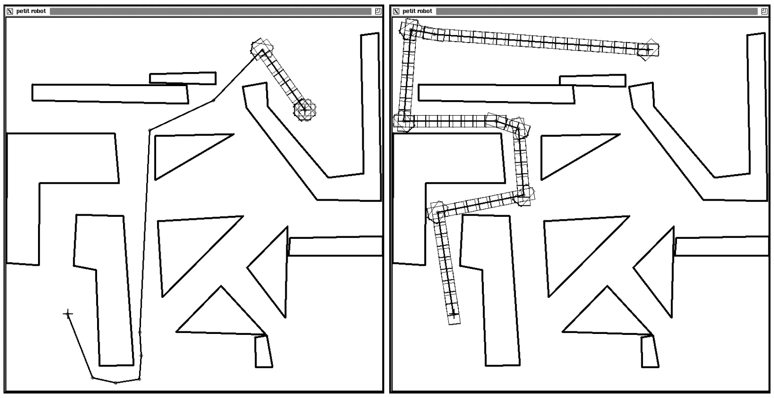
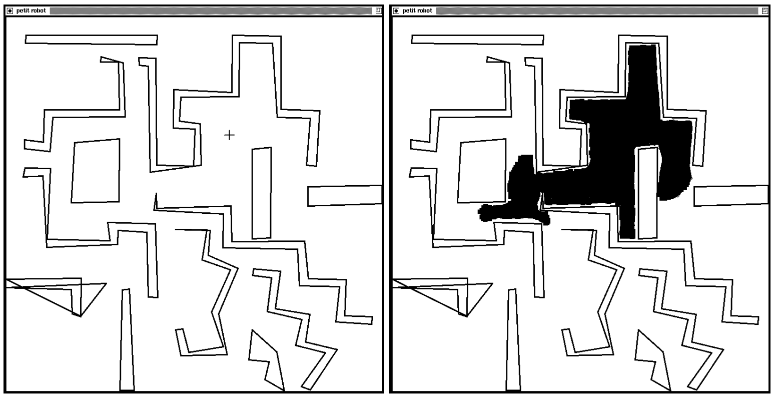


FIG. 14 - *Planification de trajectoires dans un environnement dynamique.*



a b

FIG. 15 - *La post-image produite par une balise.*

6.1.2 Implantation de l'algorithme $EXPLORE$

L'espace de recherche de $EXPLORE$ est directement inspiré de l'espace des trajectoires précédemment décrit pour $SEARCH$. Nous utilisons aussi un ensemble de six entiers pour représenter le chemin et nous suivons la même technique de rebondissement pour engendrer des chemins dans l'espace libre. Nous ajoutons un nouvel entier I_t à cet ensemble pour coder la position de départ du chemin. La valeur de I_t code l'indice de la balise de laquelle partira le chemin.

Un algorithme génétique [BES 93, ATC 93, MAT 93, MAT 93, MAT 93] est utilisé pour maximiser la fonction $EXPLORE(t)$. I_t et les θ_i sont codés par un vecteur S_i de 7 bits. Une trajectoire pour l'algorithme $EXPLORE$ est représentée de la manière suivante :

$$(I_t, \theta_1, \ldots, \theta_i, \ldots, \theta_6)$$

Par conséquent, nous définissons implicitement une post-image engendrée par l'ensemble des balises et l'ensemble des trajectoires utilisées. La figure 15b est la post-image de la balise montrée dans la figure 15a. La région noire représente la zone qui peut être atteinte par des chemins d'ordre six et partant de cette balise.

L'algorithme $EXPLORE$ consiste à maximiser la distance aux balises déjà placées en utilisant des trajectoires d'ordre six. La figure 16 montre comment l'algorithme $EXPLORE$ place d'une manière successive les balises dans l'espace libre du robot. On peut remarquer la distribution uniforme des balises dans l'espace exploré.

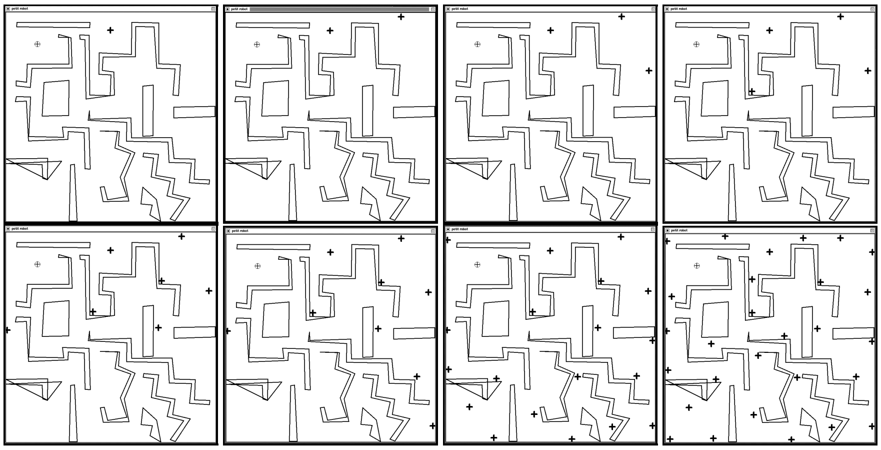


FIG. 16 - *L'évolution des balises dans l'espace libre.*

7 Conclusion

Nous avons développé une technique pour la planification automatique de trajectoire en robotique : l'algorithme Fil d'Ariane. Cette technique est composée de deux sous-algorithmes, $EXPLORE$ et $SEARCH$. L'algorithme $EXPLORE$ permet d'approcher à ε-près tout espace S connexe par arc. Nous avons défini une approximation à ε-près comme un ensemble EL de points de S tel que, pour toute configuration q

appartenant à S, il existe un point p de EL à une distance inférieure ou égale à ε. Nous avons appelé EL l'ensemble des balises. L'algorithme SEARCH est un algorithme de planification local qui exécute la recherche d'un chemin d'une configuration initiale à une configuration finale.

Les algorithmes $EXPLORE$ et $SEARCH$ ont été exprimés comme des problèmes d'optimisation sans contrainte sur $\mathbb{R}^{n*\ell}$ où n est la dimension de l'espace des commandes utilisées pour parcourir S et ℓ l'ordre des trajectoires que l'on désire utiliser. Grâce à une méthode de décodage : " le rebondissement ", il est possible de transformer tout vecteur $\hat{x} \in \mathbb{R}^{n*\ell}$ dans l'ensemble image d'un chemin $\hat{\gamma}$ de S. On dit alors que $\hat{x}$ code $\hat{\gamma}$.

Le but de l'algorithme Fil d'Ariane a été défini comme suit : à partir d'une configuration initiale $q_\circ$ construire une approximation EL de l'espace de recherche S, c'est le rôle de la fonction $EXPLORE$. Le but de l'algorithme $SEARCH$ est de vérifier si la nouvelle balise placée par $EXPLORE$ appartient à la pré-image du but.

Nous avons montré que l'algorithme Fil d'Ariane est complet pour une résolution donnée[4]. Cette dernière propriété vient du fait que l'algorithme $EXPLORE$ assure que : s'il existe un chemin d'une configuration initiale $q_\circ$ à une configuration finale $q_\bullet$ et que ce chemin est inclus dans un "tube" de rayon ε appartenant entièrement à l'espace libre, alors il existe un temps fini T tel que $EXPLORE$ est capable de construire un chemin de $q_\circ$ à $q_\bullet$. L'algorithme $SEARCH$ est alors une amélioration de l'algorithme $EXPLORE$ qui permet d'accélérer la construction d'un chemin de $q_\circ$ à une configuration particulière $q_\bullet$.

Nous avons montré la validité de l'Algorithme Fil d'Ariane en développant plusieurs planificateurs. Celui décrit dans cet article est un planificateur de trajectoires pour un robot mobile holonome. Nous avons par ailleurs développé un planificateur de trajectoires pour un bras manipulateur à six degrés de liberté placé dans un environnement dynamique. L'algorithme a montré sa capacité à réagir à des changements rapides et imprévisibles de l'environnement, c'est-à-dire que l'algorithme peut-être utilisé comme un planificateur réactif pour certains problèmes. Lorsqu'une trajectoire est calculée et qu'un objet change de position, il est possible de re-planifier, avec un temps de réaction faible, une nouvelle trajectoire.La méthode de planification que nous avons proposée, réunit les avantages suivants :

1. elle ne fait pas appel au calcul de l'espace des configurations, mais elle peut en fournir une approximation,

2. elle s'adapte naturellement à la difficulté du problème en trouvant la "bonne" approximation de l'espace des configurations,

3. elle est facile à implémenter même sur des machines massivement parallèles,

4. et elle peut être utilisée dans d'autres problèmes de planification que celui de la planification de trajectoires.

4. "resolution complet"

Références

[BES 93] Pierre Bessière, Juan Manuel Ahuactzin, El-Ghazali Talbi, Emmanuel Mazer. *The "Ariadne's clew algorithm": Global Planning with local methods"*, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems'93 (IROS'93), Yokohama, Japan July 26-30, 1993.

[CGK 93] Daniel J. Challou, Maria Gini and Vipin Kummar: *Parallel Search Algorithms for Robot Motion Planning*, Proceedings of the 1993 IEEE International Conference on Robotic and Automation, Atlanta, Georgia, USA - May 1993.

[WAR 93] Charles W. Warren*Fast Path Planning Using Modified * Method*, Proceedings of the 1993 IEEE International Conference on Robotic and Automation, Atlanta, Georgia, USA - May 1993.

[CHH 92] C. Chen, Yong K. Hwang : *SANDROS: A Motion Planner with Performance Proportional to Task Difficulty*, Proceedings of the 1992 IEEE International Conference on Robotic and Automation,Nice, France - Mai 1992.

[DUG 76] James Dugundji: *TOPOLOGY*, Allyn and Bacon, inc., Boston, Lonon, Sydney, Toronto, July 1976.

[ERD 86] Michel Erdman: *Using Backprojections for Fine Motion Planning with Uncertainty*, The international Journal of Robotics Research, Vol. 5 No. 1, Spring 1986, Massachusetts Institute of Technology.

[GOL 89] David E. Goldberg: *Genetic algorithms in search, optimization and machine learning*, The University of Alabama, Addison-Wesley publishing company, inc. 1989.

[HWA 92] Yong K. Hwang and Narendra Ahuja: *Gross Motion Planning A Survey* Advanced Robotics Redundancy and Optimization, ACM Computing Surveys, Addison-Wesley Publishing Company Vol 24, No. 3, Septembre 1992.

[AMB 93] Juan Manuel Ahuactzin, E. Mazer, P. Bessiere, E.G. Talbi : *Using Genetic Algorithms for Robot Motion Planning,* en,*Lecture Notes in Computer Science 708, Geometric Reasoning for perception an Action, pp.84-93* Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest, 1993.

[ATC 93] Juan Manuel Ahuactzin, El-Gazali Talbi, Thierry Chatroux, Pierre Bessiere, Emmanuel Mazer: *A Massively Parallel Implementation of the Ariadnés Clew Algorithm*, Intelligent Robotic System9́3, Zakopane, Poland, 20-24 July 1993.

[LAT 91] Jean-Claude Latombe: *Robot Motion Planning*, Ed. Kluwer Academic Publisher, 1991.

[LMT 84] Tomás Lozano-Pérez, Matthew T. Mason and Russell H. Taylor: *Automatic Systhesis of Fine-Motion Strategies for Robots*, The international Journal of Robotics Research, Vol. 3 No. 1, Spring 1984, Massachusetts Institute of Technology.

[LOZ 87] Tomás Lozano-Pérez: *A simple motion-planning algorithm for general robot manipulators*, IEEE Robotics and Automation, Juin 1987.

[MAT 93] Emmanuel Mazer, Juan Manuel Ahuactzin, El-Ghazali Talbi, Pierre. Bessiere: *The Ariadnés Clew Algorithm*, From Animals to Animats: The Second International Conference on Simulation of Adaptive Behavior (SAB92). Honolulu, Hawaii, USA December 7-11, 1992.

[MAT 93] Emmanuel Mazer, Juan Manuel Ahuactzin, El-Ghazali Talbi, Pierre. Bessiere: *Robot Motion Planning with the Ariadnés Clew Algorithm*, International Conference on Intelligent Autonomous Systems, Pittsburgh PA, USA February 15-19, 1993.

[MAT 93] Emmanuel Mazer, Juan Manuel Ahuactzin, El-Ghazali Talbi, Pierre Bessière, Thierry Chatroux: *Parallel Motion Planning with The Ariadnés clew algorithm*, Third International Symposium On Experimental Robotics, Kyoto, Japan, 28-30 October 1993.

[OVE 92] Mark H. Overmars: *A random approach to motion planning*, Utrecht University, Departament of Computer Science, The Netherlands 1992.

[OVE 93] Mark H. Overmars: *A random approach to motion planning*, Spring School on Robot Motion Planning, Rodez, France, March-April, 1993.

[PRJ 93] Prabir K. Pal and K. Jayarajan:, Fast Path Planning for Robot Manipulators Using Spatial Relation in the Configuration Space,Proceedings of the 1991 IEEE International Conference on Robotic and Automation, Atlanta, Georgia, USA - May 1993.

[PRS 85] Franco P. Preparata, Michael Ian Shamos: *Computational Geometry: an intriduction*, Springer-Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest, 1985.

[ROG 64] C.A. Rogers: *"Packing and Covering"*, Cambridge University Press, New York, 1964.

[SAA 70] Thomas L. Saaty: *Optimization in Integers and Related Extremal Problems*, Mc-Graw Hill, USA, 1970.

[SCH 72] A. Schijver (Editor): *Pakink and Covering in Conbinatorics*, Mathematical Centre Tracts 106 p.141-177,.

[SVE 92] Petr Švestka:, *A Probabilistic Approach to Motion Planning for Car-Like Robots*, Utrecht University, Departament of Computer Science, The Netherlands 1992.