Le mouvement... une composante des stratégies de navigation

Les stratégies de navigation permettant à un robot mobile de se déplacer pour rejoindre un objectif sont extrêmement diverses, mais elles nécessitent toutes des notions de perception, décision et action à des niveaux divers.

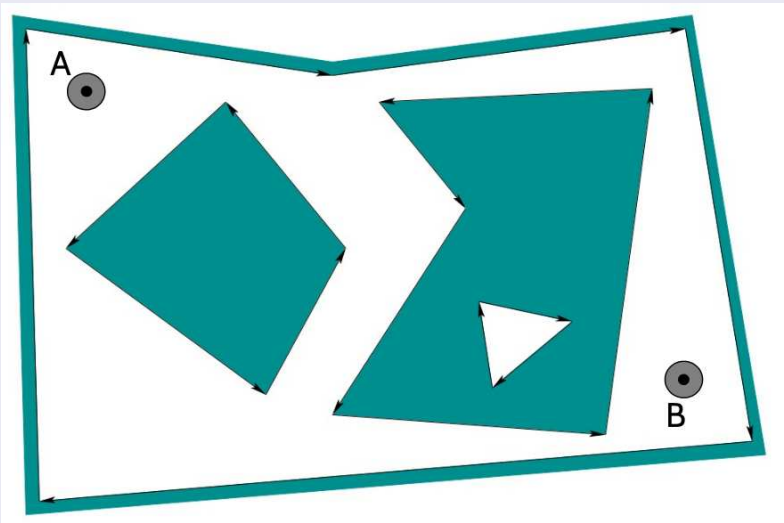
- Navigation réactive :
  - Action réflexe : perception $\Longrightarrow$ action
    - Action liée à un évènement (aller vers un objet car lumière ou bruit)
    - Action liée à une configuration d'amers visibles (équidistance de murs, balises)
    - Action associée à un lieu : carte d'amers-action
  - Pas de modèle global de de l'environnement W, uniquement « locaux »
- Navigation globale :
  - Modèle interne du monde (carte) indépendant du but $\Longrightarrow$ planification
    - Navigation topologique
    - Navigation géométrique

Que faire sans modèle de l'environnement W ?

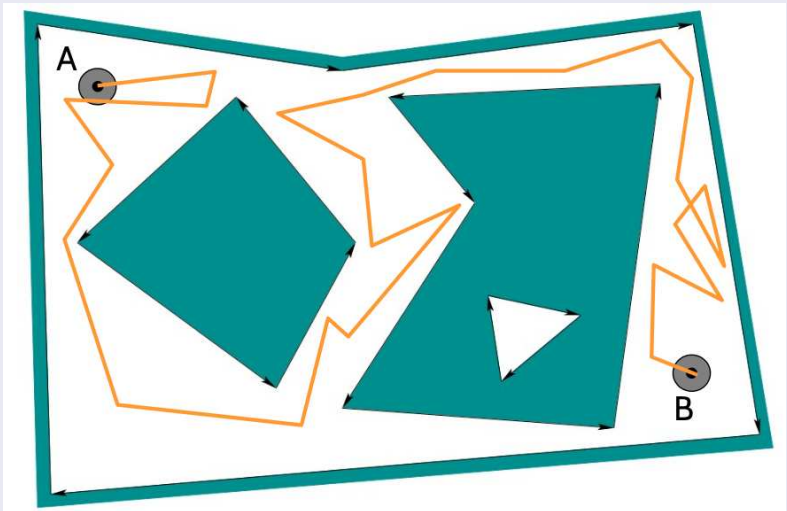
- Mouvements réactifs
- Modélisation incrémentale de W (SLAM)
- Perception pour bouger ou bouger pour percevoir ?

Problématique de la planification de trajectoires

"Existe t'il un chemin sans-collision depuis la position A à la position B ?"

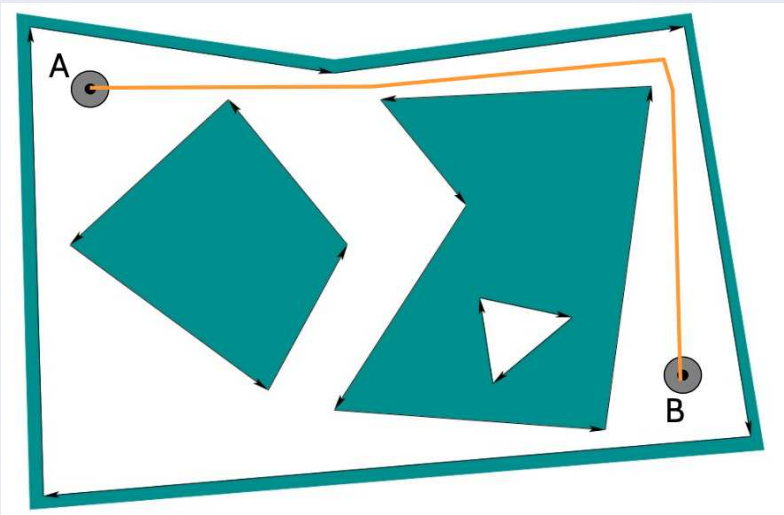


"Quel est un chemin sans-collision depuis la position A à la position B ?"

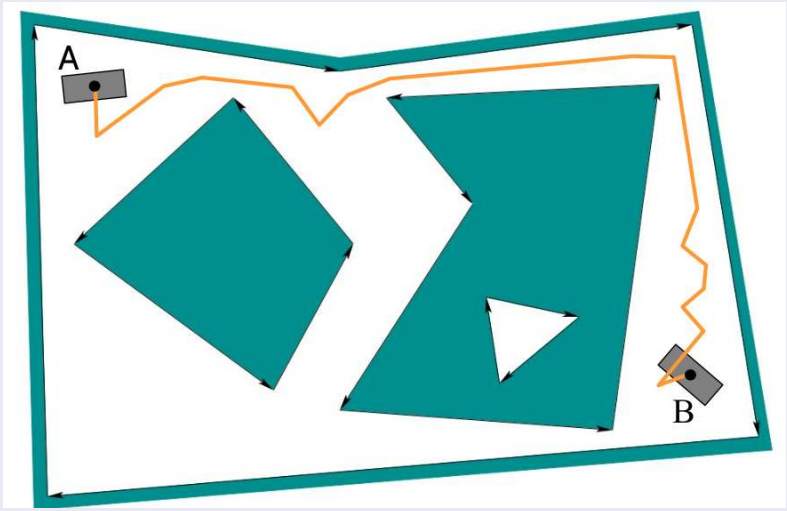


Problématique de la planification de trajectoires

"Quel est le chemin sans-collision le plus court depuis la position A à la position B ?"

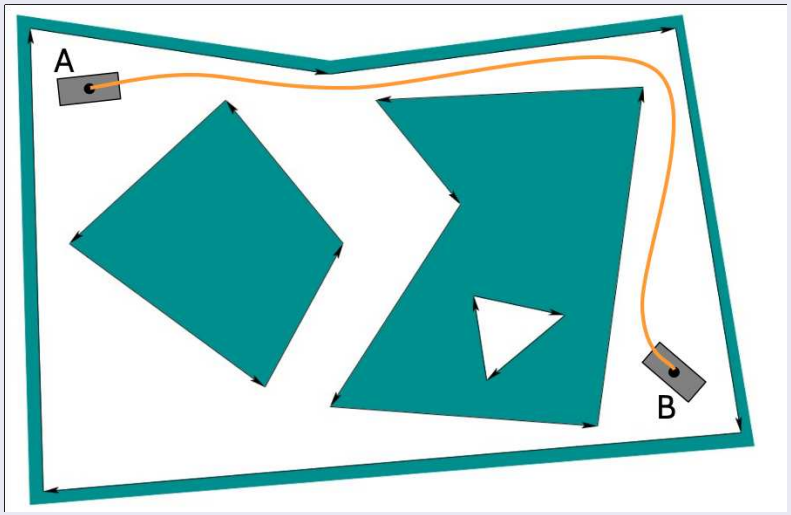


"Quel est un chemin sans-collision depuis la position A à la position B pour un robot polygonal ?"

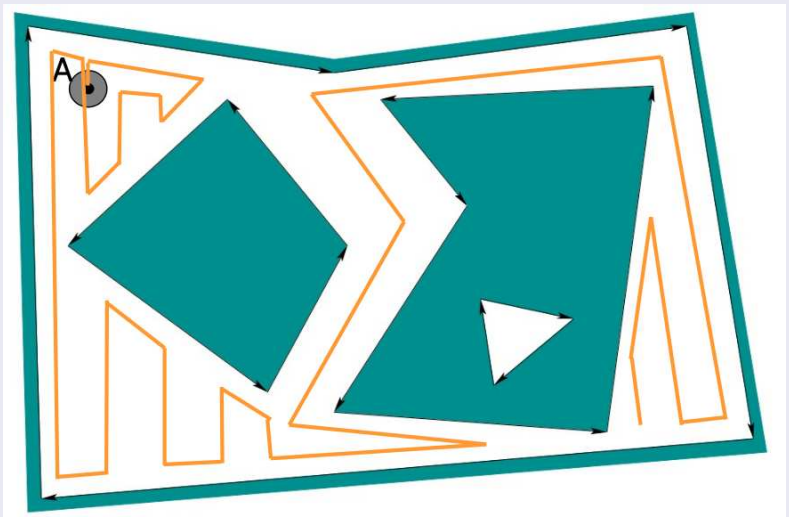


Problématique de la planification de trajectoires

"Quel est un chemin sans-collision depuis la position A à la position B pour un robot polygonal non-holonome ?"



"Quel est le chemin sans-collision le plus court depuis la position A qui recouvre la surface navigable ?"



Planification de chemins/trajectoires

Objectif

Produire un chemin (ou trajectoire) sans collision

Problème complexe car nombreuses contraintes :

- Géométrie et modèle du robot (ex : contraintes différentielles)
- Prise en compte de la dynamique du robot ?
- Modèle de l'environnement
- Connaissance partielle ou complète
- Incertitudes sur le robot et l'environnement (modèle, capteurs, ...)
- Obstacles fixes ou mobiles
- Mono ou multi-robots
- Etc...

$\Longrightarrow$ Formulation mathématique générale complexe.

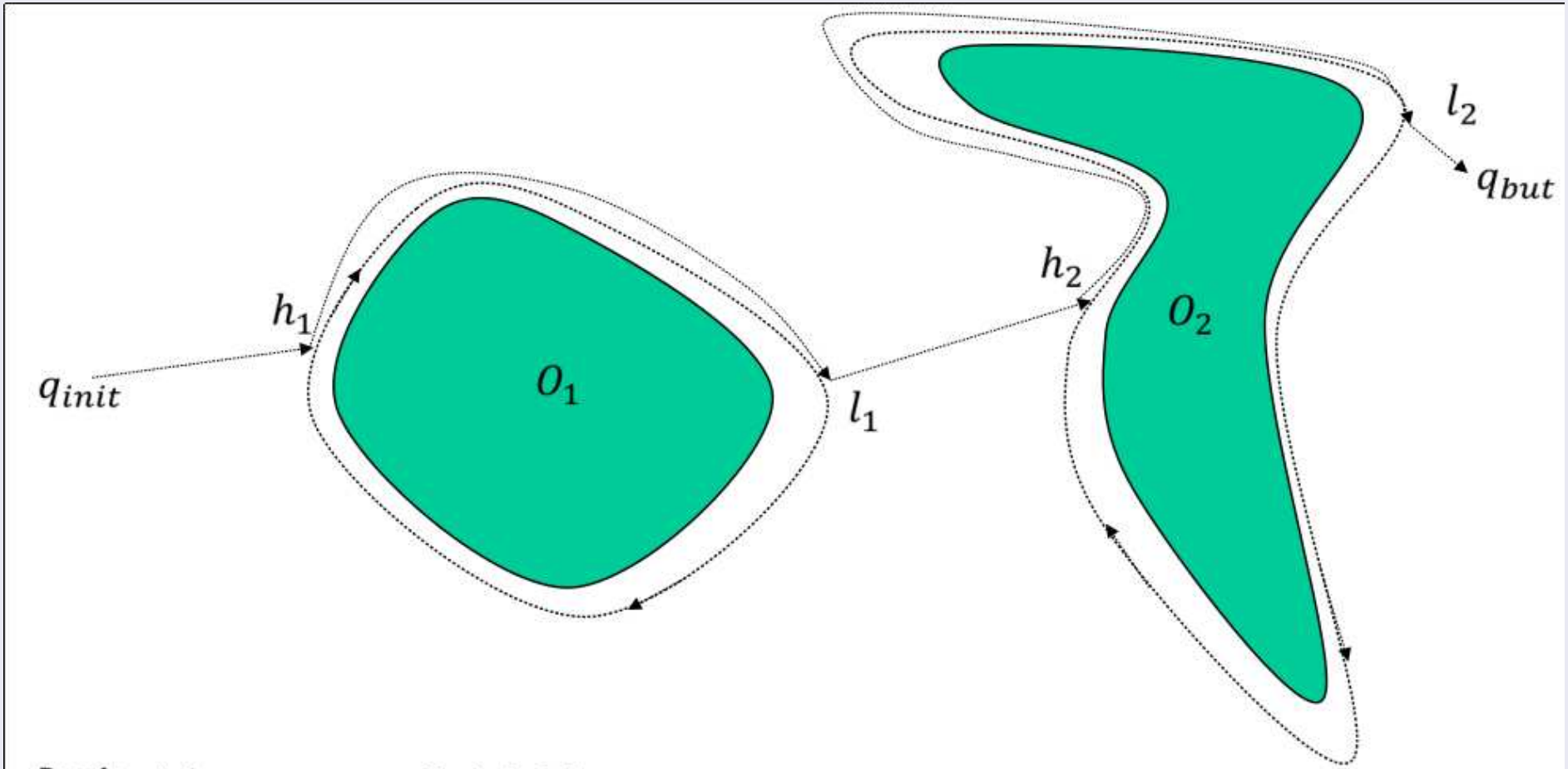
On se ramène souvent à un problème **géométrique**.

Pourtant le problème ponctuel est simple....

- Robot ponctuel
- Localisation parfaite (pour se repérer / but)
- Mouvement dans un plan
- Détection de contact (ou de distance robot- obstacle)
- Environnement borné et inconnu

Stratégie simple (algorithmes « Bug ») :

- Aller en ligne droite vers le but
- Si obstacle, le contourner (suivre le contour de l'obstacle)



Bug1: entrée: q_{init}, q_{but} ; sortie: trajectoire

Tant que q_{but} non atteint, faire

1. Avancer en ligne droite vers q_{but}
2. Si un obstacle est détecté, le contourner entièrement
3. Revenir à la position autour de l'obstacle la plus proche de q_{but}

- Quelle est la longueur maximum de la trajectoire calculée $L(Bug1)$?
- Donner un exemple ou cette longueur est atteinte ?
- Comment améliorer $Bug1$ pour générer des trajectoires plus courtes ?
- Quelle est la longueur maximum de la nouvelle trajectoire calculée $L(BugX)$?
- Donner un exemple.

Mais un robot n'est pas un point !

Mouvement

- calculer une trajectoire/chemin pour réaliser une tâche...
- ... en respectant des contraintes (non-collision, limite de vitesse,...)
- envoyer les consignes de commande au système

Planification de trajectoires

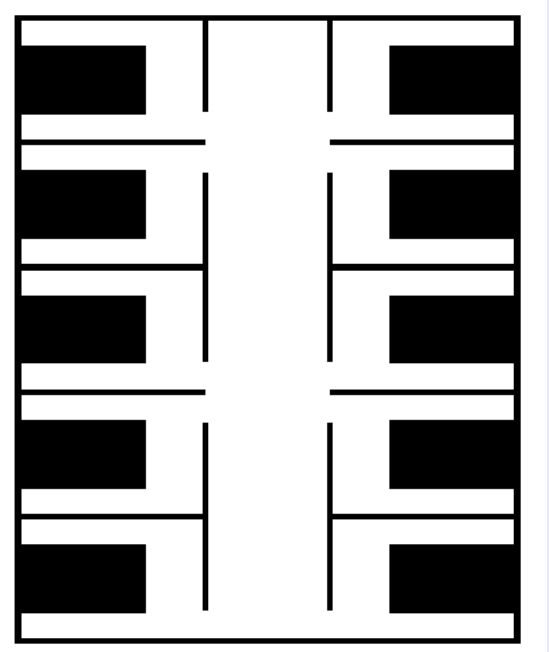
- Modéliser l'espace de travail (environnement)
- Définir l'espace des configurations
- Mettre en œuvre un algorithme de Planification de trajectoires (en général dans l'espace des configurations mais parfois dans W).

Modélisation de W

1

Carte métrique

- Représentation des obstacles : primitives géométriques
  - Polygones/segments en 2D
  - Triangles ou polyèdres en 3D
- Carte métrique dense
- Difficile à construire à partir de données capteurs
- Planification facile
- Passage à l'échelle : nombre et complexité des obstacles

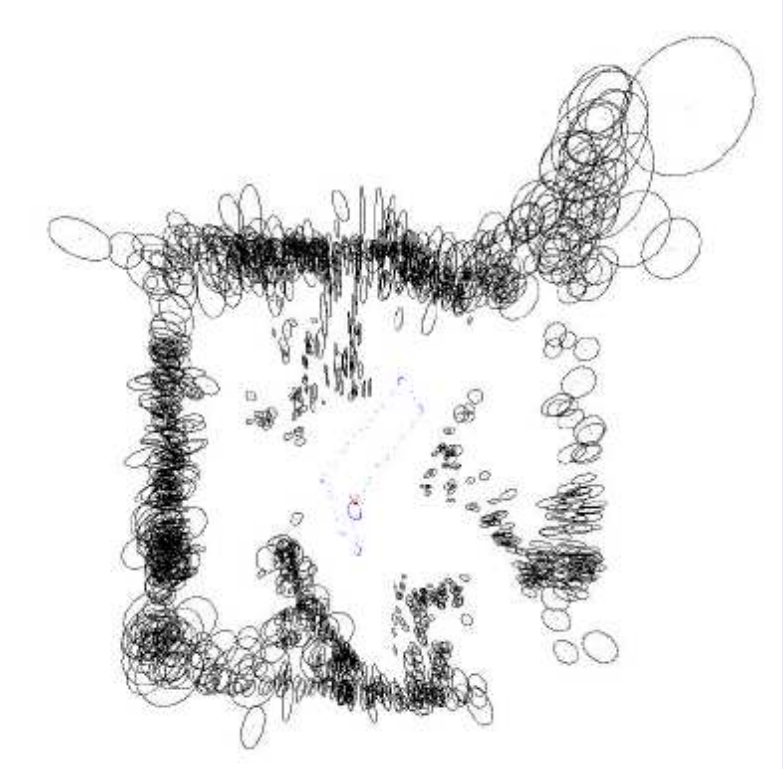


1. (Cartes issues de F. Colas, Inria)

Modélisation de W

Carte d'amers

- liste de points de référence :
  - descripteur
  - position
- représentation des éléments saillants
- carte métrique creuse
- facile à construire à partir des données capteurs
- conçue pour la localisation
- planification très difficile
- passage à l'échelle : nombre des points de référence



Modélisation de W

Carte grille d'occupation

- représentation dense de l'espace :
  - segmentation de l'espace
  - probabilité d'occupation
- espace libre, occupé ou inconnu
- carte métrique dense
- bonne construction à partir de données de distance
- planification facile
- passage à l'échelle : surface (ou volume) et résolution.

Modélisation de W

Carte d'élévation MNT

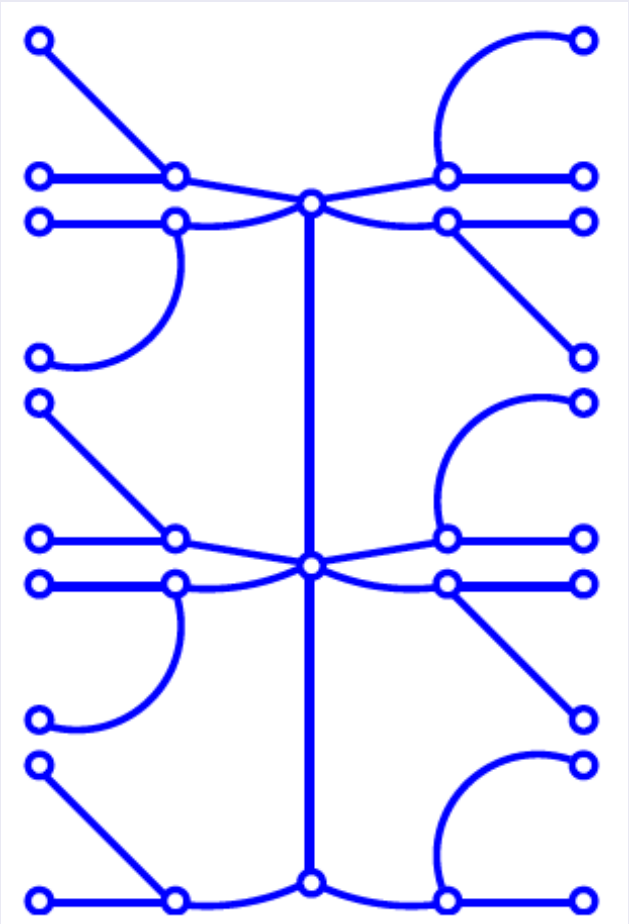
- représentation de la hauteur du sol
- carte en 2,5D
- carte métrique dense
- bonne construction à partir de données de distance
- planification facile
- passage à l'échelle : surface et résolution

(vidéo)

Modélisation de W

Carte topologique

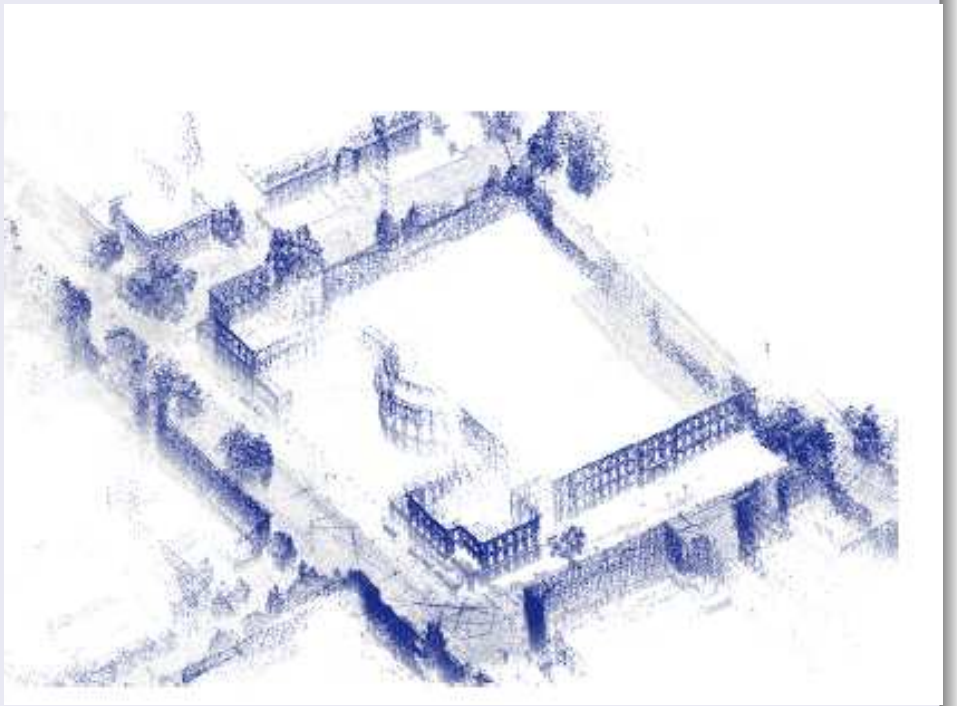
- liste de points de référence :
  - descripteur
  - position
- représentation des éléments saillants
- carte métrique creuse
- facile à construire à partir des données capteurs
- conçue pour la localisation
- planification très difficile
- passage à l'échelle : nombre des points de référence



Modélisation de W

Conclusion sur les cartes

- Différents usages
  - localisation
  - planification de chemin
  - planification de tâche
  - visualisation
- Différentes caractéristiques
  - métrique ou non
  - dense ou non
  - facilité de construction
  - facilité de planification

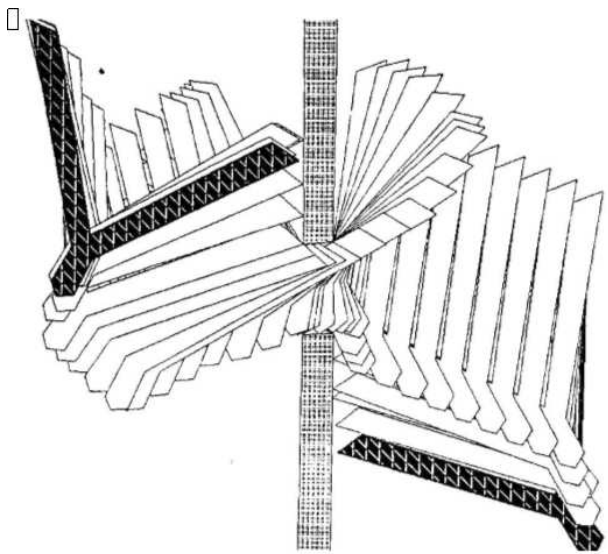


Espace des configurations *EC*

W : espace de travail de dimension N (2 ou 3)
A : robot, objet rigide
Oi : obstacle rigide, fixe de W (i=1,...,p)
Géométrie et placement des objets connus

Étant donné une position et une orientation initiale q_{init} et finale q_{goal} de A dans W, peut-on trouver pour A une trajectoire sans collision avec les O_i entre q_{init} et q_{goal} ?

Simple pour un robot ponctuel mais le problème est difficile dans le cas général...



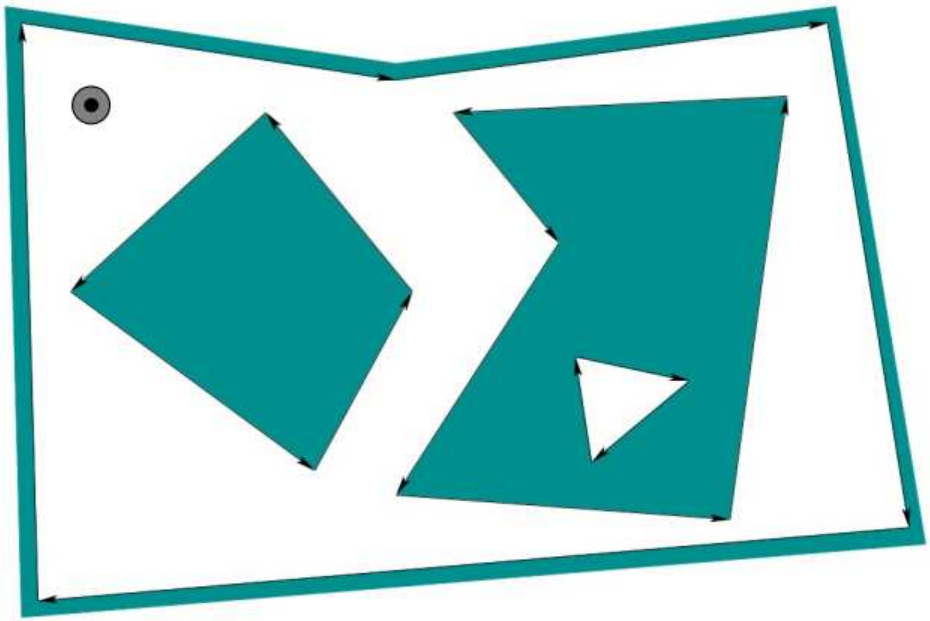
Espace des configurations EC

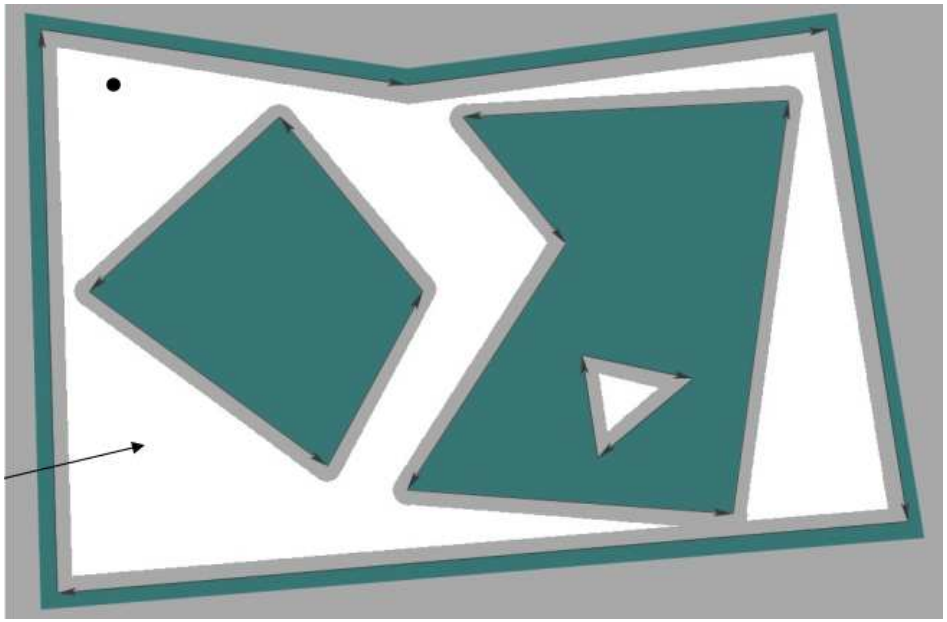
- Problème complexe $\Longrightarrow$ transformer le problème de déplacement d'un corps rigide en un problème de déplacement d'un point dans un certain espace qui caractérise tous les placements possibles de A : l'espace des configurations EC.
- On appelle configuration q de A, la donnée de m paramètres $(q_1 q_2, ..., q_m)$ qui caractérisent la position et l'orientation de A dans W.
- les paramètres $(q_1 q_2, ..., q_m)$ sont appelés degrés de libertés du robot
  La dimension de l'espace des configurations $EC = m$

- Comment définit on entièrement et de façon unique une posture (configuration) du robot ?
- Degrés de libertés ?
- Dimension de l'espace des configurations ?

De l'Espace de travail W...à l'Espace des Configurations EC

Dans l'espace des configurations, une configuration = un « point » (dimension 0)=> Quelque soit le robot, on cherche un chemin pour un point dans l'espace des configurations sans collisions, dit espace des configurations libre et généralement noté EC_{free}

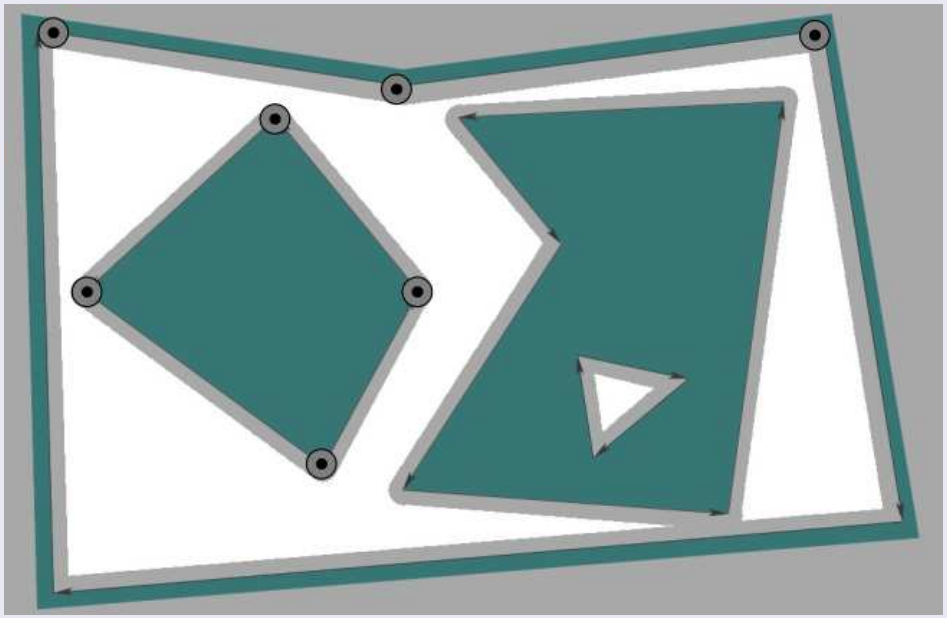




De l'Espace de travail W...à l'Espace des Configurations EC

Dans le cas d'un robot holonome et de forme circulaire, on construit l'espace des configurations en « érodant » l'espace de travail par la géométrie du robot (somme / différence de Minkowski)

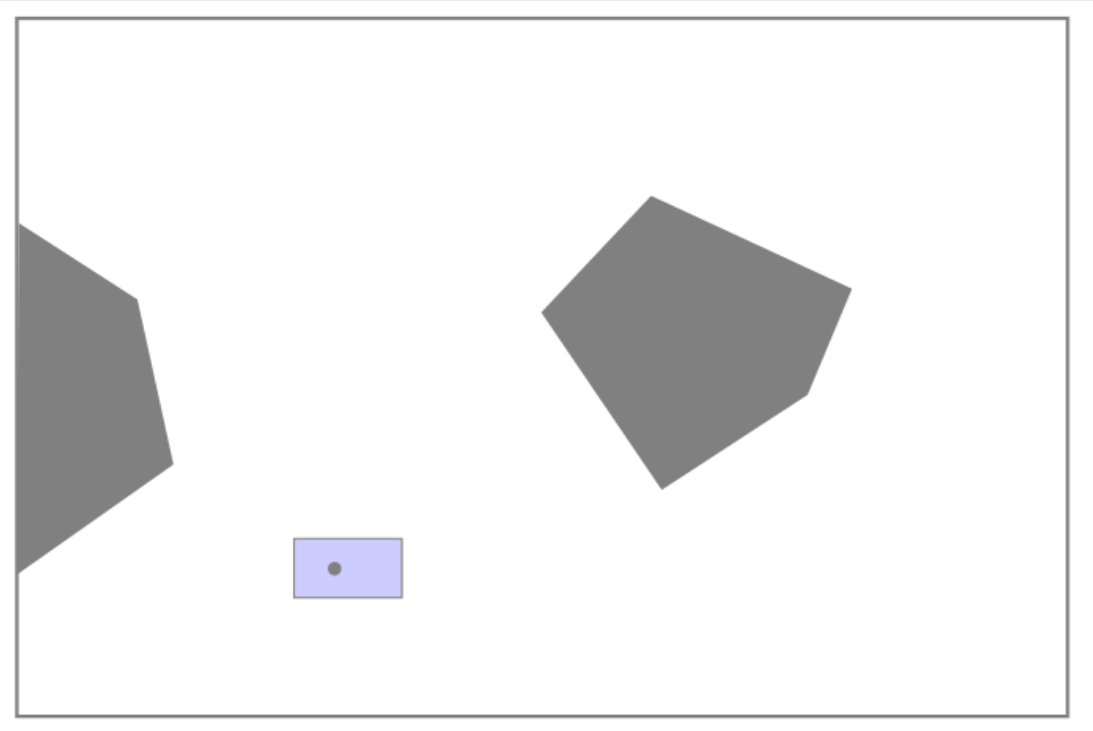
- On fait glisser le centre du robot le long des obstacles
- Dépend du choix du repère du robot
- Pour des géométries plus générales, on applique cette opération pour chaque degré de liberté



De l'Espace de travail W...à l'Espace des Configurations EC

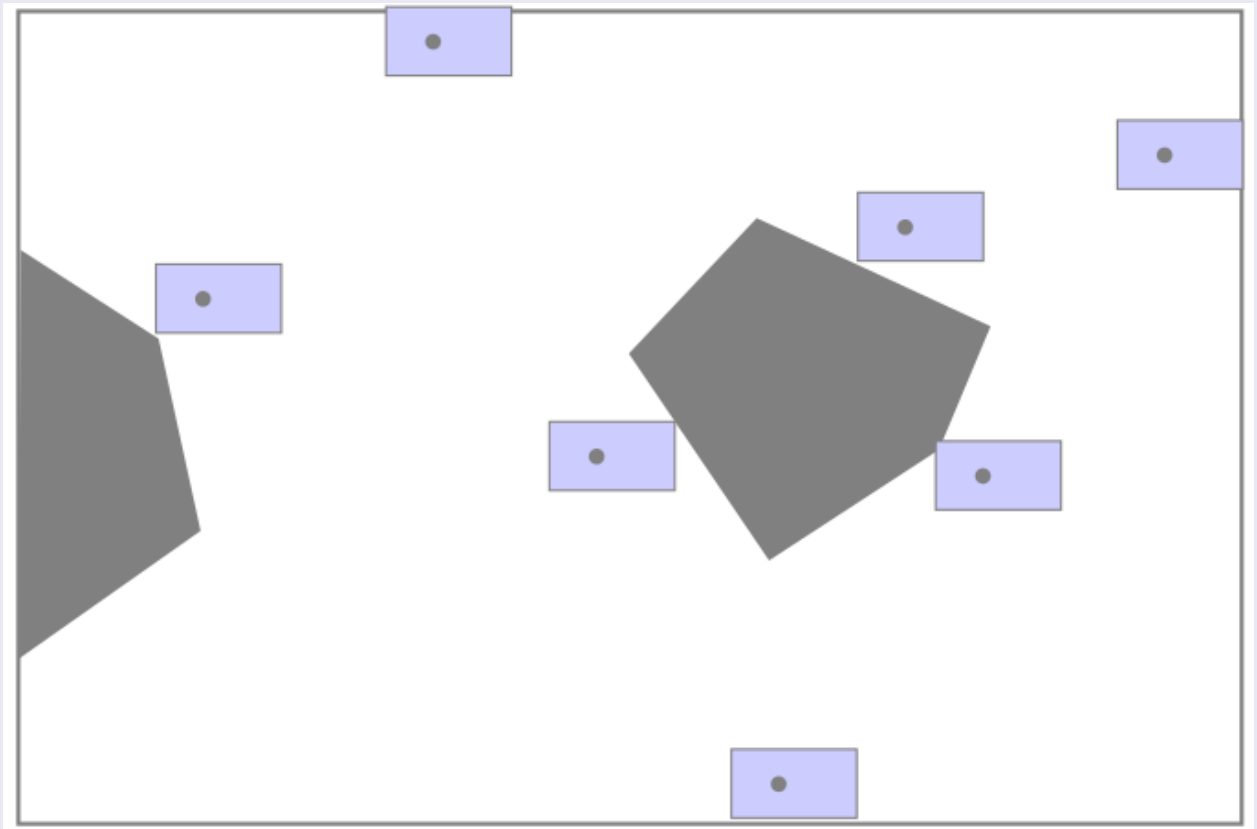
Robot rectangulaire (holonome)

- Comment définit on entièrement et de façon unique une « posture » (configuration) du robot ?
- Degrés de libertés ?
- Dimension de l'espace des configurations ?



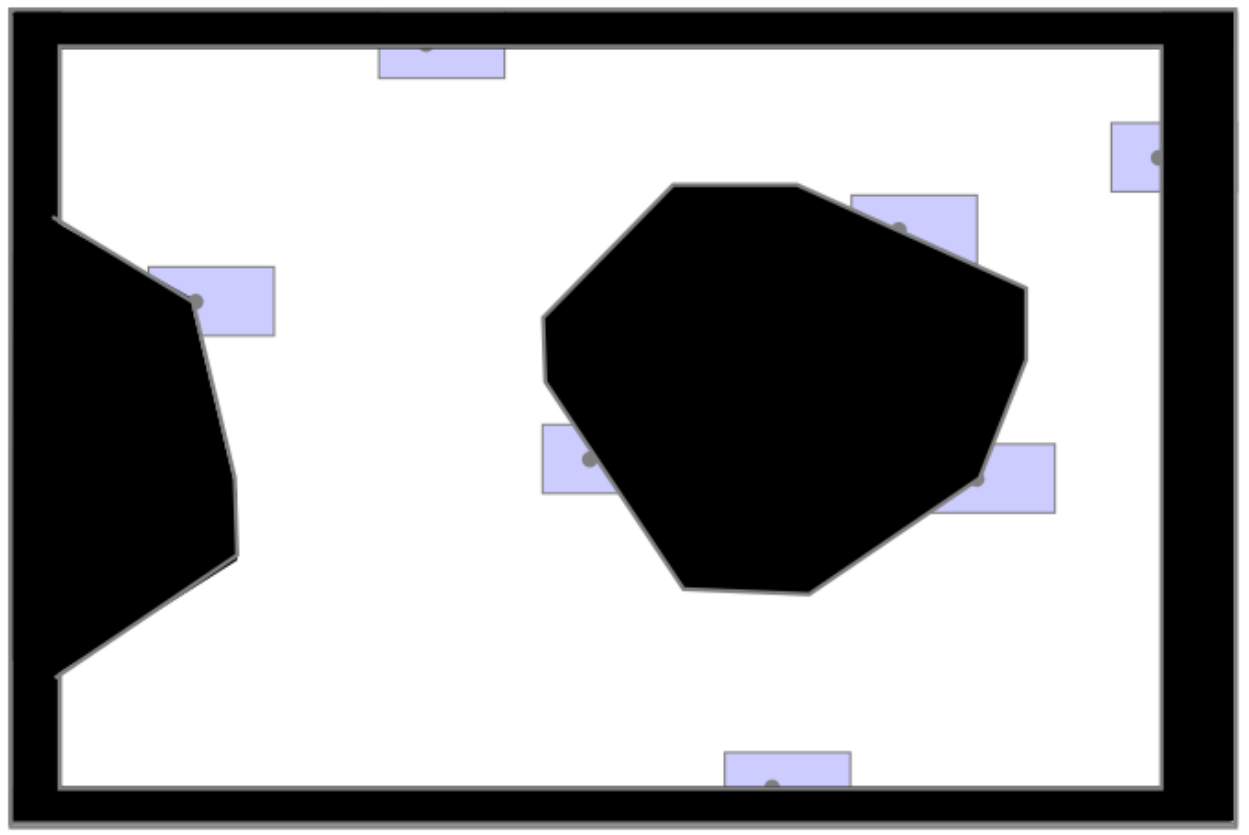
De l'Espace de travail W...

Si on fixe l'orientation du robot $\theta = 0$, on obtient :



...à l'Espace des Configurations *EC*

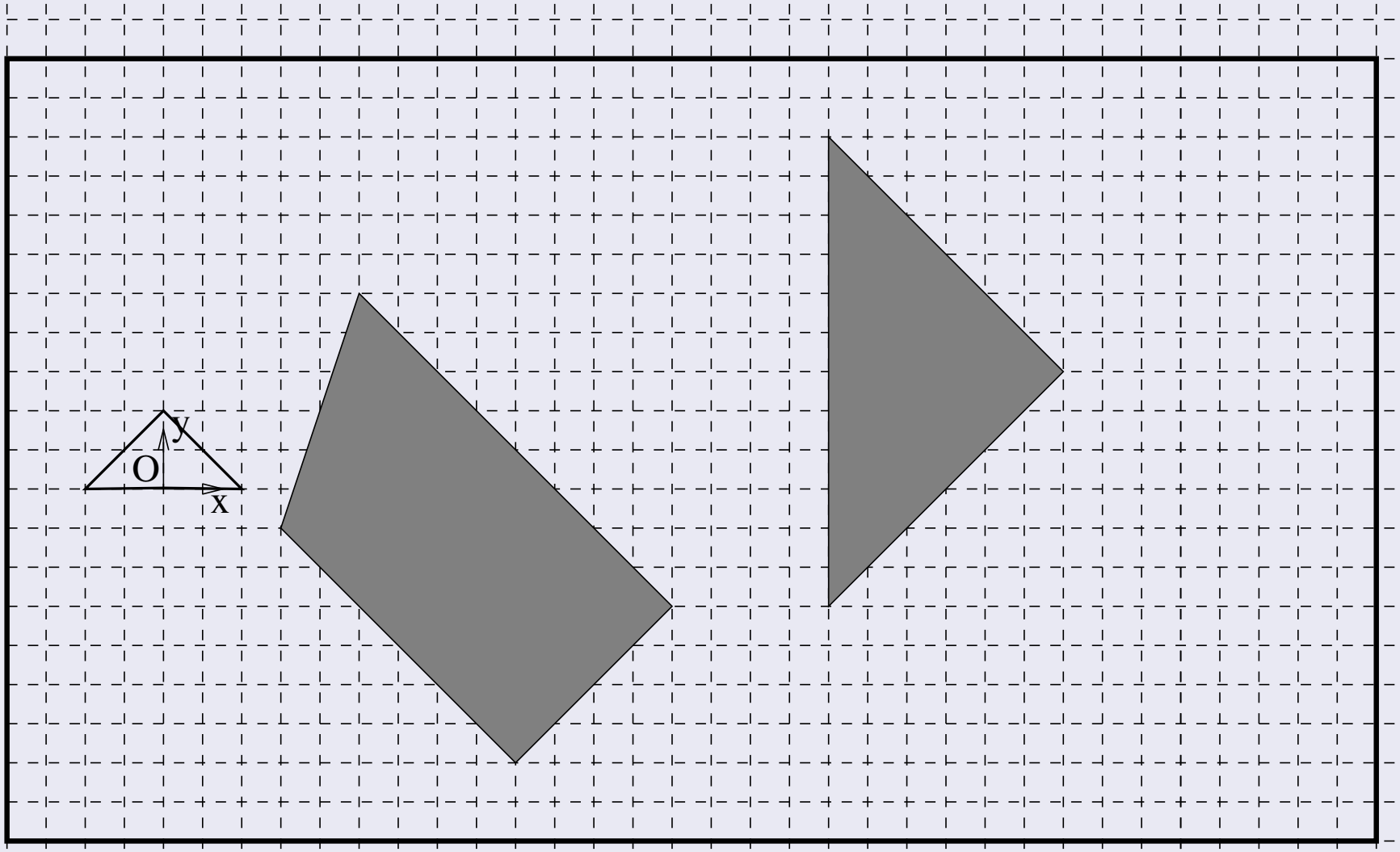
Si on fixe l'orientation du robot $\theta = 0$, on obtient :



Obstacles et C-Obstacles

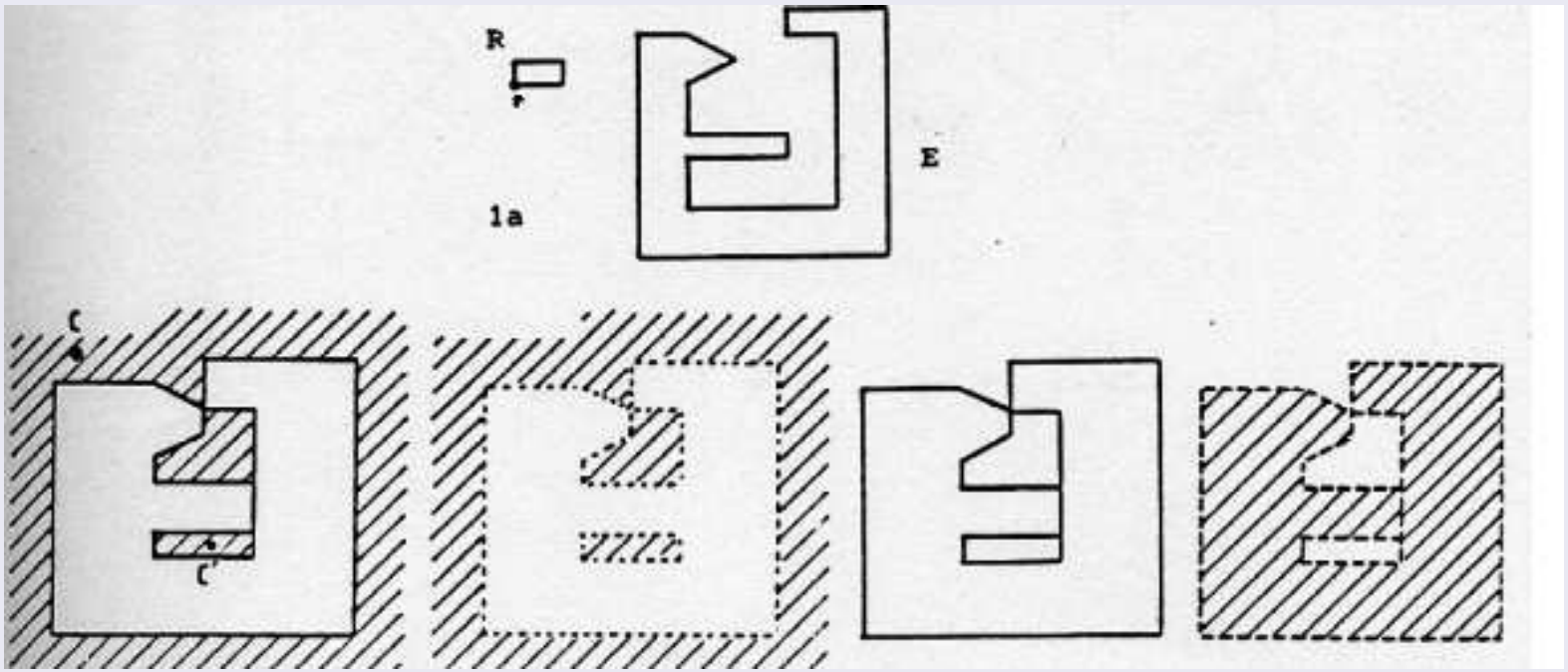
- $CB_i = \{q \in EC | A(q) \cap (\cup B_i) \neq \emptyset\}$, C-Obstacle.
- $ECB = \cup CB_i$, région des C-Obstacles.

Exemple



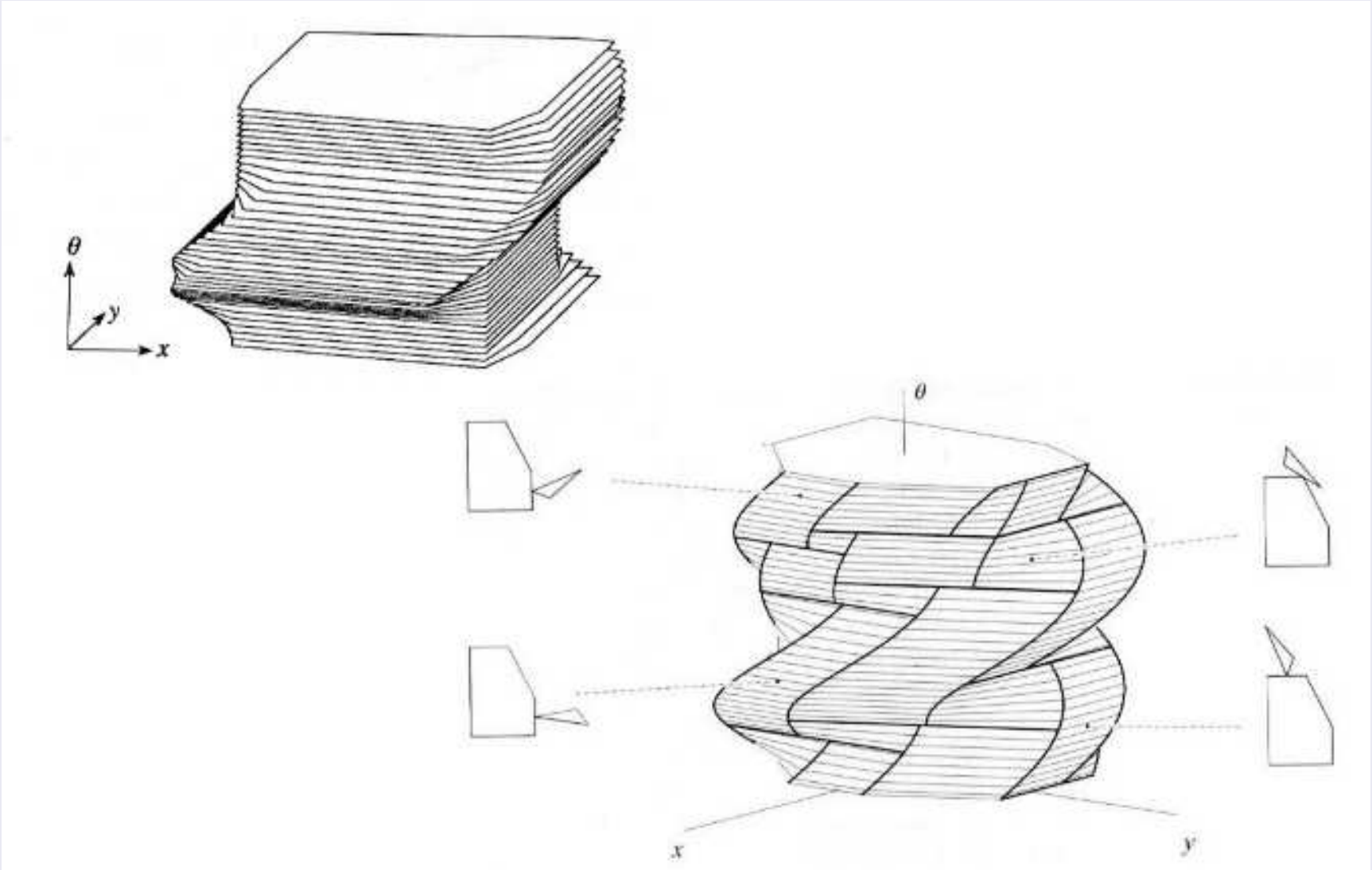
Différents sous-espaces de EC

- $ECL = EC \backslash ECB = \{q \in EC | A(q) \cap (\cup B_i) = \emptyset\}$
- $ECC = \{q \in EC | A(q) \cap (\cup B_i) \neq \emptyset \text{ et } int(A(q)) \cap int((\cup B_i) = \emptyset\}$
- $ECA = ECL \cup ECC$

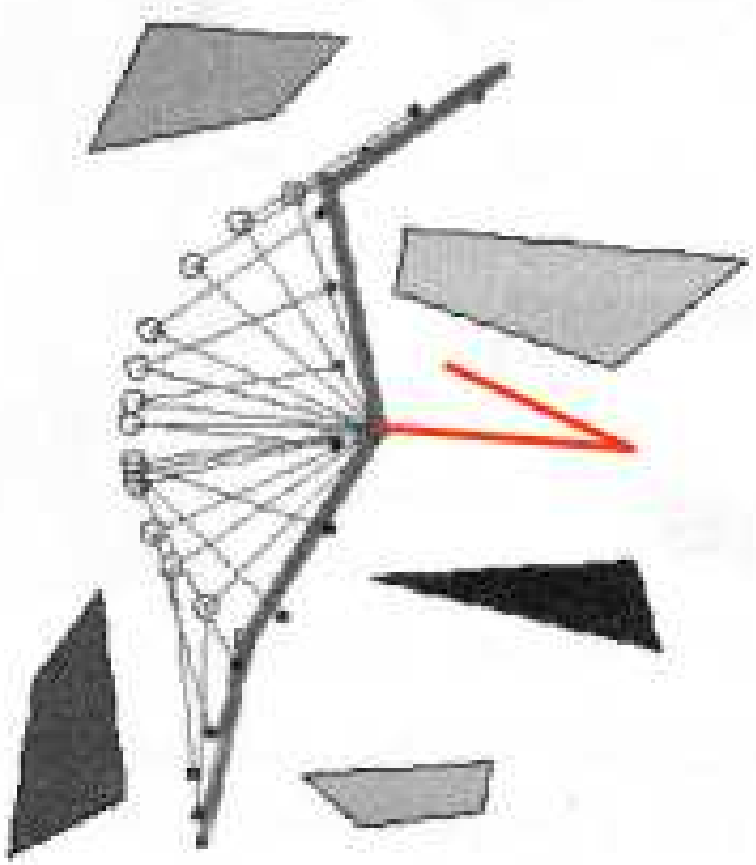


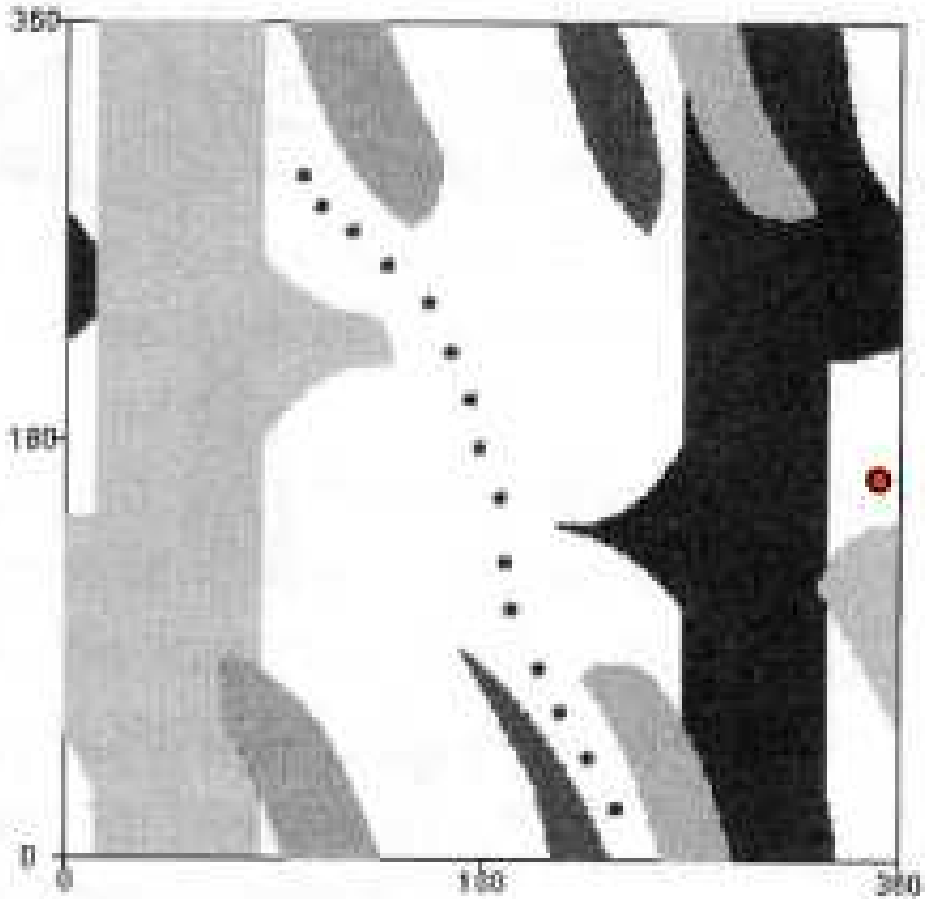
EC de dimension 3 pour un robot mobile

On peut construire l'espace des configurations en "empilant" toutes les représentations pour les valeurs successives de $\theta \in [0, 2\pi[$



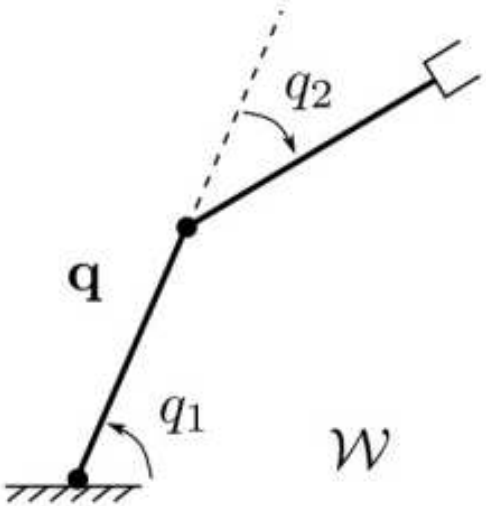
De l'Espace de travail W...à l'Espace des Configurations EC

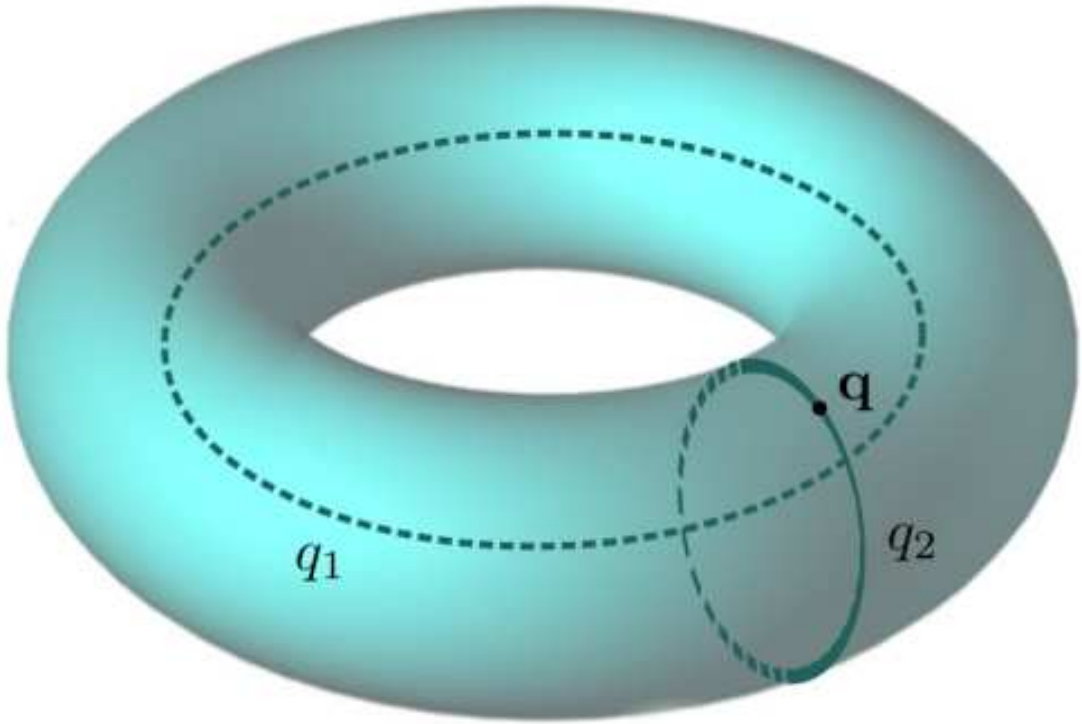




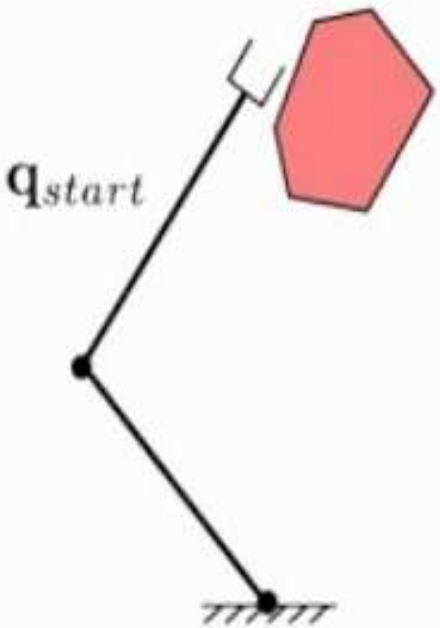
EC est-il un plan borné ?

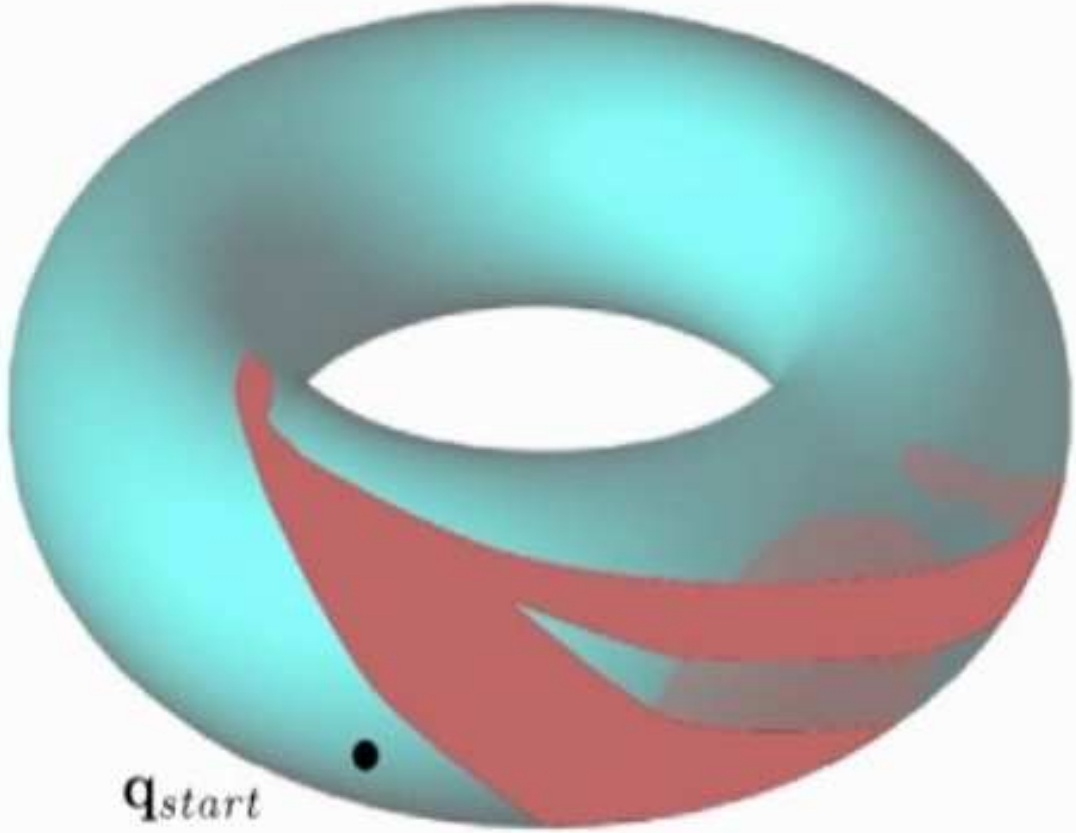
De W à EC





De W à EC





Existence d'un chemin solution

Chemin géométrique

- Un chemin géométrique de A dans W entre 2 situations est un chemin dans EC entre les 2 configurations correspondantes (q_i , q_f).
- Chemin : fonction continue $\Pi : [0,1] \longrightarrow EC$ avec $\Pi(0) = q_i$ et $\Pi(1) = q_f$

Métrique

Distance (ou métrique) : fonction $d : ECxEC \longrightarrow \mathbb{R}^+$
Exemple

Calcul de chemin $\Longrightarrow$ transformation des obstacles O_i dans EC ?
Absence de O_i => tout chemin est réalisable pour un robot libre de mouvement (free-flyer)
EC est connexe si $\forall(q_i, q_f)\ \exists$ un chemin T entre q_i et q_f

Existence d'un chemin solution

Il existe un chemin sans collision entre q_i et q_f
si et seulement si
q_i et q_f appartiennent à la même composante connexe de EC_{free}
(ou EC_{adm} si le contact est autorisé).

Ce chemin est toujours faisable si A est libre de mouvement (aucune contrainte sur les directions de déplacement du robot)

Principe des méthodes de recherche globale de trajectoires

- Modélisation de EC_{free} (ou EC_{adm}) sous forme d'un graphe
- Il existe un chemin entre q_i et q_f si elles appartiennent à la même composante connexe du graphe

Planification déterministe

- Roadmap
  - Graphe de visibilité
  - Diagramme de Voronoï
- Décomposition cellulaire
  - Décomposition cellulaire exacte
  - Décomposition cellulaire approchée
- Champs de potentiel
  - Approche générale
  - Grille de potentiel
- Robots non holonome
  - Discrétisation des commandes
  - Approximation de chemin holonome

On suppose que nous avons un modèle numérique de W ou de EC.

Méthodes basées *Roadmap*

L'union d'un ensemble de courbes (de dimension 1) est une Roadmap, *Rmap*, si pour toutes configurations q_i et q_f de *ECL* qui peuvent être connectés par un chemin, les propriétés suivantes sont vérifiées :

- il existe un chemin de q_i a une configuration q^* de *Rmap*
- il existe un chemin d'une configuration $\tilde{q}$ de *Rmap* à q_f
- il existe un chemin sur *Rmap* entre q^* et $\tilde{q}$

Principe des méthodes *Roadmap*

Capturer la connexité de l'espace libre dans un graphe (courbe de dimension 1) appartenant a *ECL* ou *ECA*

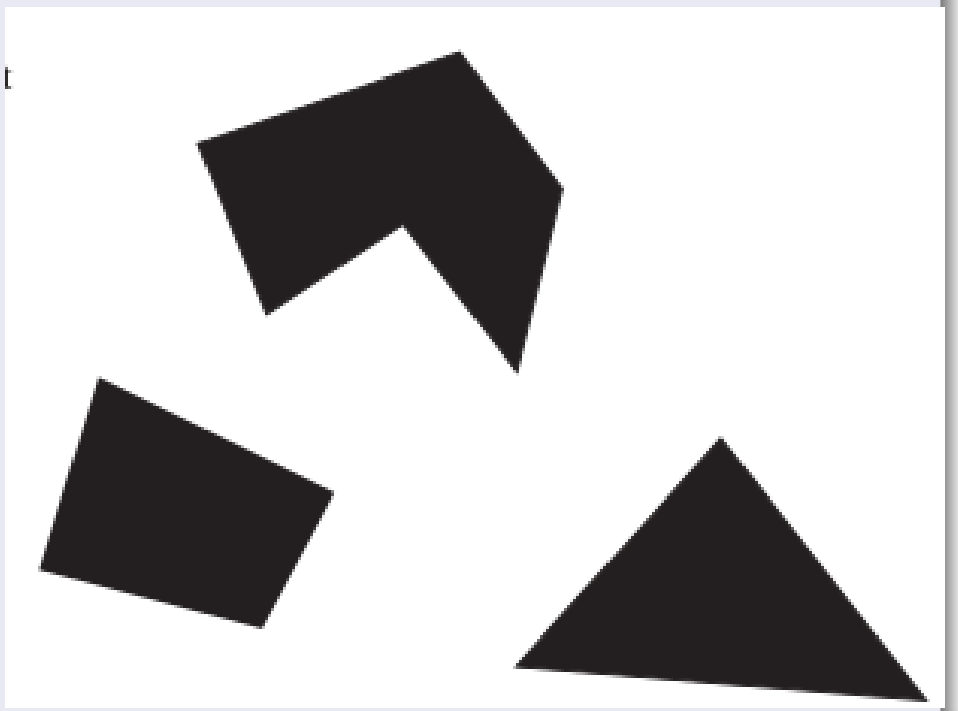
- Construire un graphe *GR*
- Connecter q_{init} et q_{but} à *GR*
- Chercher chemin dans *GR*

Exemple de méthodes *roadmap* :Graphe de visibilité

Calculer toutes les lignes brisées entre les sommets de la frontière de l'espace libre/admissible (notion de visibilité)

- Sommets = ?
- Arcs = ?

Construction en
$O(n^3) \Longrightarrow O(n^2 log(n))$
Recherche en $O(E_{GV} + n.log(n))$



Intérêt – inconvénient ?

Exemple de méthodes *roadmap* :diagramme de Voronoï

Diagramme de Voronoï Vor_S d'un ensemble de n points $S = \{p_i\}$.

- Région de Voronoï : $Vor_S(p_i) = \{x \in \mathbb{R}^2 s.t. \forall p_{j,j \neq i} \in S, ||x - p_i|| \leq [[x - p_j||\}$
- Diagramme de Voronoï = ensemble des cellules (régions + adjacence)
- Sommets = ? Arcs = ? Cas particuliers ?

Triangulation de Delaunay $Del(S)$

- (p_i, p_j, p_k) est un triangle de Delaunay de S si et seulement si le cercle circonscrit à (p_i, p_j, p_k) ne contient aucun point de S
- Triangulation de Delaunay de S : triangulation de S où chaque triangle est un triangle de Delaunay

Vor_S et $Del(S)$ sont des structures duales
Calcul en $O(n^2) \Longrightarrow O(nlog(n))$

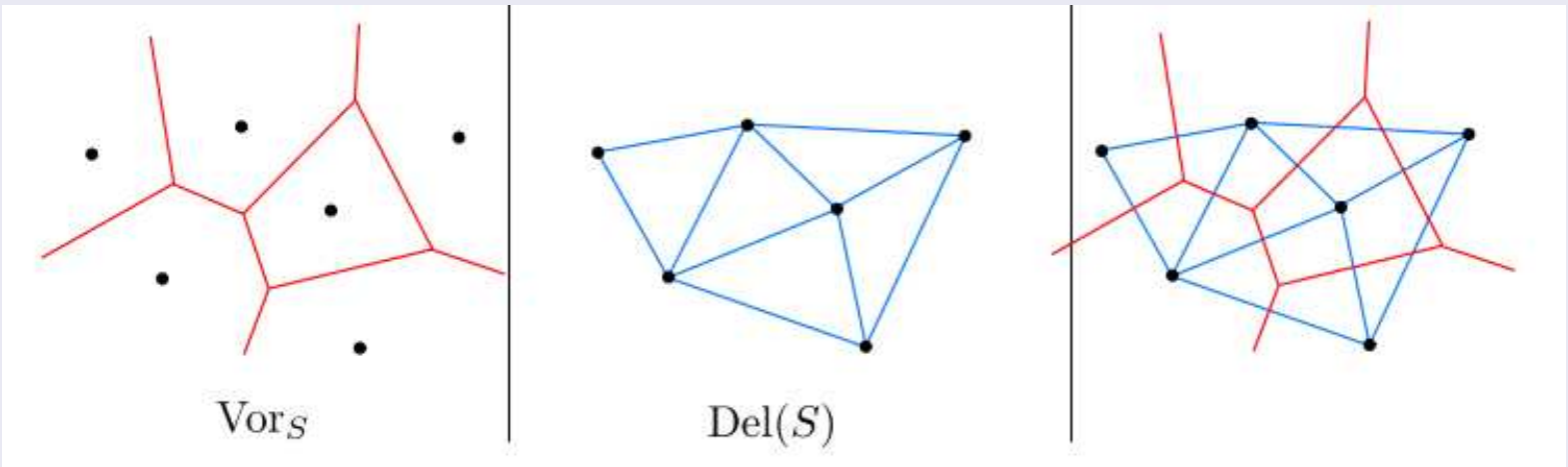


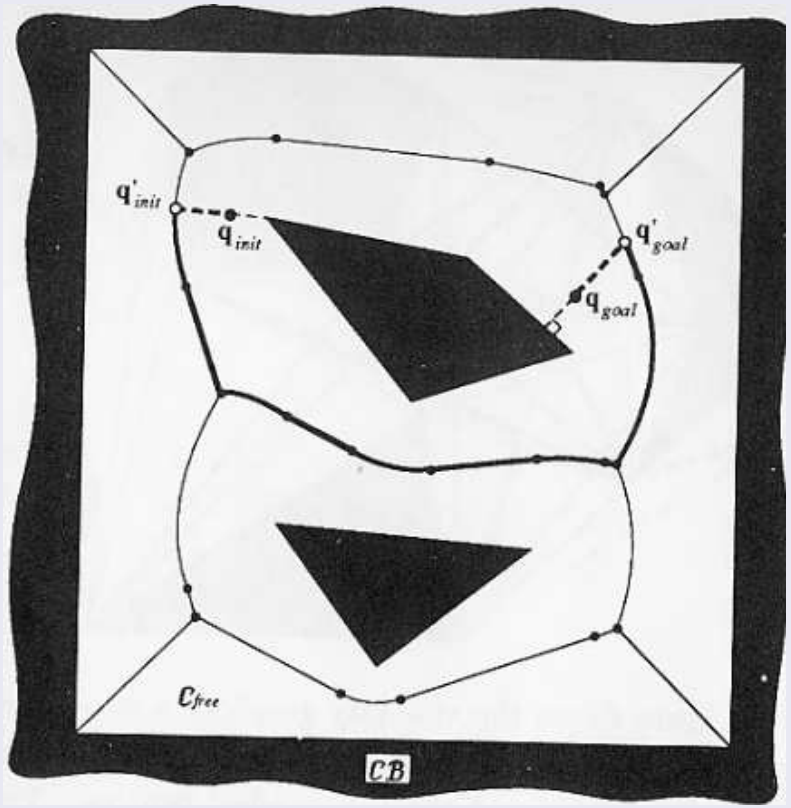
Diagramme de Voronoï - Généralisation aux polygones

Diagramme de Voronoï : Calculer les configurations les plus éloignées des obstacles

- Sommet - sommet $\Longrightarrow$ segment
- Arête - arête $\Longrightarrow$ segment
- Sommet - arête $\Longrightarrow$ arc de parabole

Pour toutes paires (s,s), (E,s), (s, E) :
Construction en
$O(n^4) \Longrightarrow O(n^2) \Longrightarrow O(n.log(n))$

- Calculer les courbes (segment,parabole)
- Calculer l'intersection de ces courbes



Intérêt – inconvénient :

- Chemin le plus « sûr »
- Calcul d'intersection délicats
- Optimisation post-calcul nécessaire

Décomposition polygonale exacte

Représenter exactement un espace par un ensemble de cellule polygonale.

Décomposition polygonale exacte K de E

K est une décomposition polygonale convexe exacte de E si K est un ensemble fini de polygones convexes, appelés cellules cel, telles que :

- $Int(cel_i) \cap Int(cel_j) = \emptyset \ \ \forall i \neq j$
- $\cup(cel) = \overline{E}$ (fermeture ou adhérence)

Deux cellules sont adjacentes si et seulement si leur intersection est un segment non nul

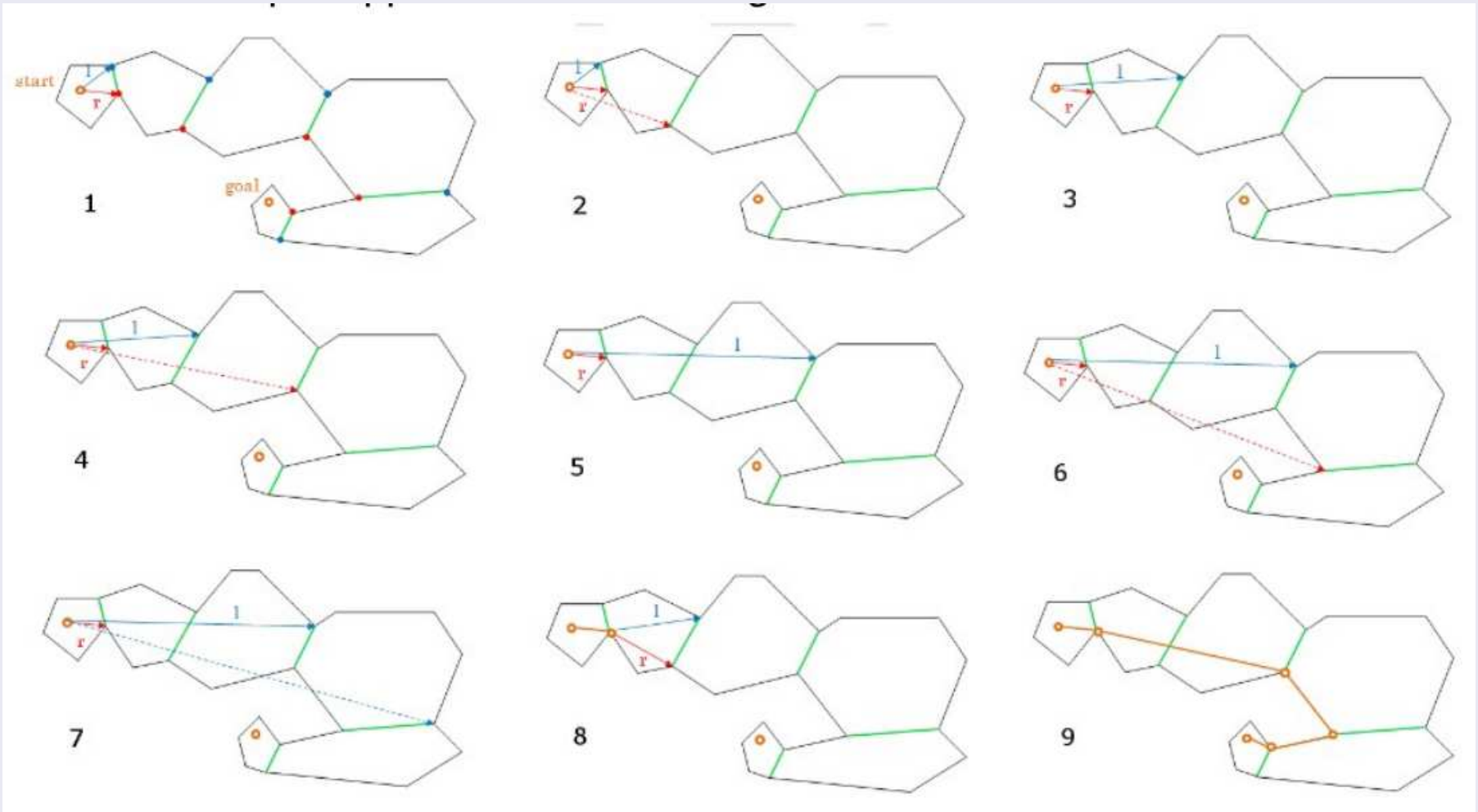
Graphe de connexité G :

- Noeud = cel
- Arc/arêtes = adjacence

Attention : certains algorithmes sont simples mais une implémentation **robuste** est difficile.

Planification de chemin dans une décomposition polygonale

Planification locale : Depuis la séquence de polygones (« corridor ») obtenue dans la phase globale, trouver le plus court chemin
Par exemple, application du funnel algorithm



Décomposition approchée

Cellules de forme prédéfinie qui ne permettent pas de représenter exactement ECL ou W.

La forme des cellules doit permettre :

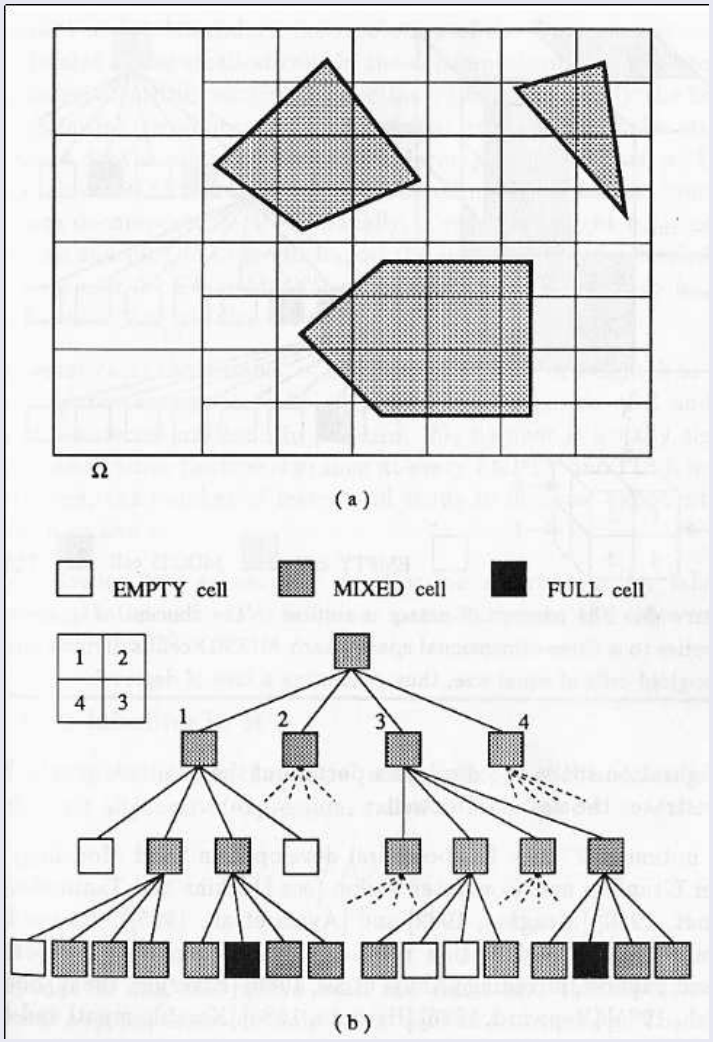
- de calculer itérativement la décomposition
- d'être robuste numériquement

Représentation sous forme de graphe (arbre) appelée « quadtree »

La taille des cellules solutions du chemin donne une information sur l'espace libre.

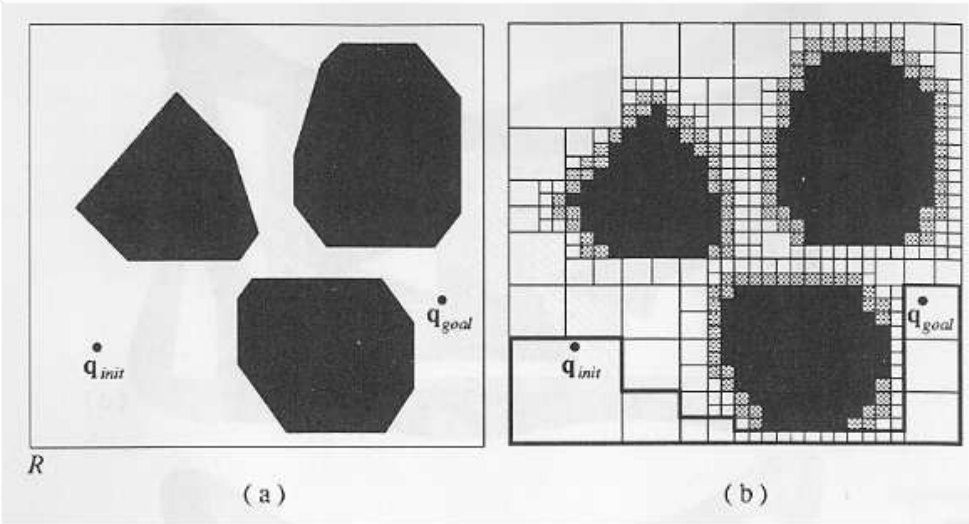
Algorithmes quasi-complet

Exemple : « Marquer et Diviser »



- Nombre de feuilles pour décomposition de hauteur h en dimension n ?

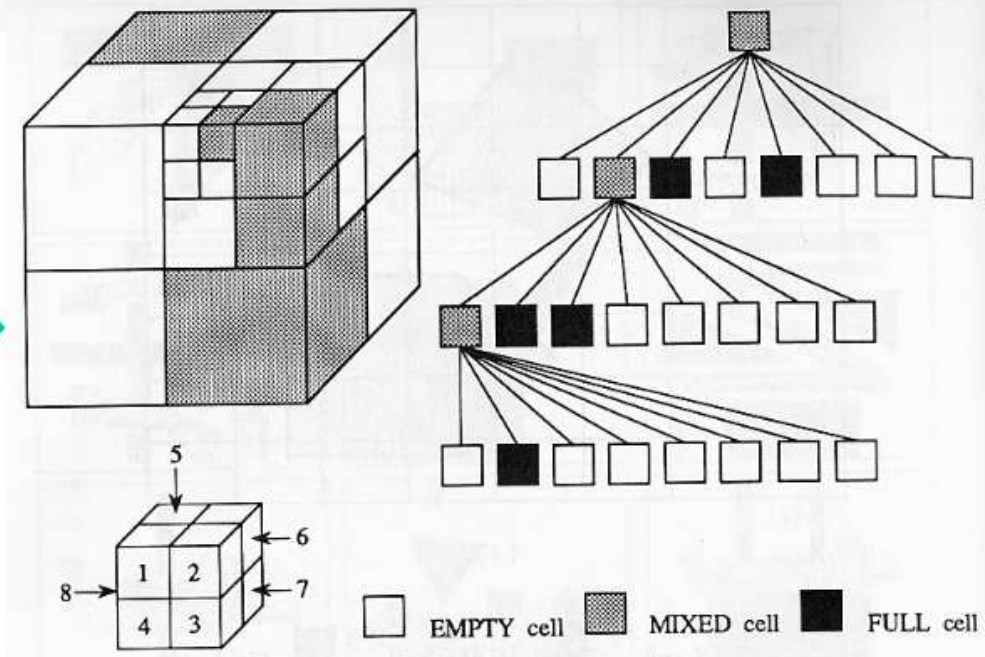
Décomposition approchée



Recherche de chemin

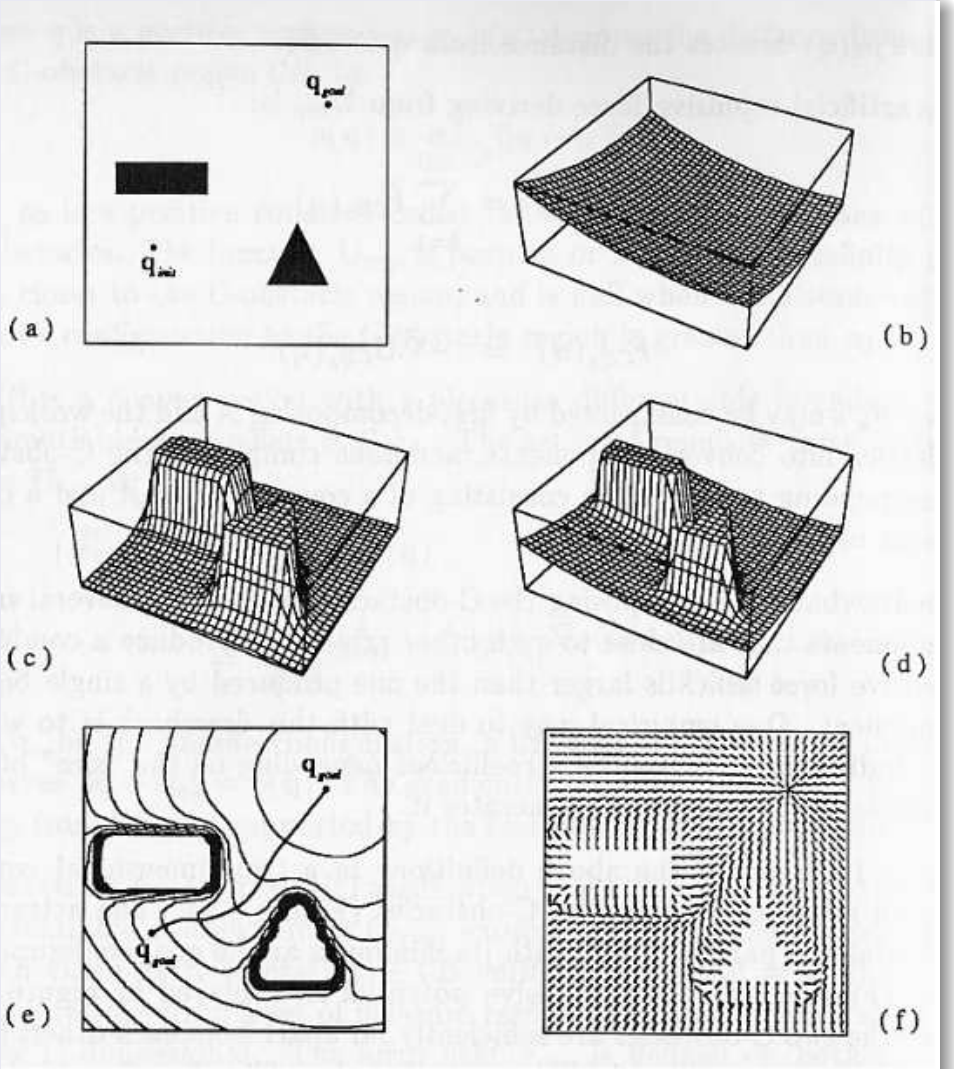
Extension en dimension 3 (**octree**)

Généralisable en dimension n, mais complexité exponentielle...



Champs de potentiel

- Mouvement sous l'action d'un champ de potentiel U.
  - potentiel attractif vers le but U_a
  - potentiel répulsif proche des obstacles U_r
- Forces induites par U_a et U_r : $F(q) = -\nabla U(q) = -\nabla U_a - \nabla U_r$
- Puits de potentiel
- Réactivité importante



Comment utiliser ce principe pour faire de la planification globale ?

Champs de potentiel : exemple de méthode globale

Idée :

- Suivre le gradient jusqu'au but ou minimum local.
- Si minimum alors remplir le puit de potentiel

Discrétisation de l'espace (ECou W)
Complexité ?
open : liste de cellules triées par valeur croissante de potentiel

« Best First Planner »

open ← $\{q_{init}\}$
q_{init}.visited ← True
WHILE open != {}
  q←best(open)
  IF $q == q_{goal}$
    return SUCCESS
  FOR n ∈ neighbors(q)
    IF NOT n.visited AND NOT isInCollision(n)
      open.insert(n)
      n.visited← True
return FAILURE

Champs de potentiel : exemple de méthode globale par propagation

Idée : Suivre le gradient jusqu'au but ou minimum local

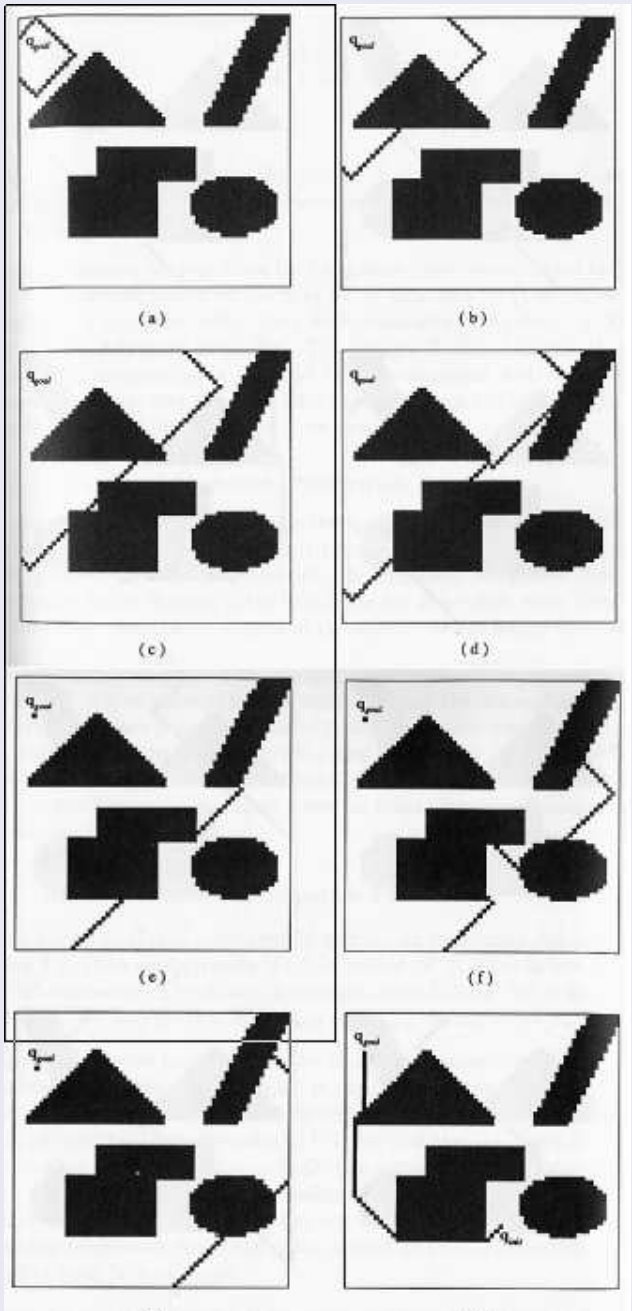
Si minimum alors remplir le puit de potentiel

Discrétisation de l'espace (ECou W)

Complexité ? Généralisation $n > 2$?

Algorithm 1 Wavefront Propagation

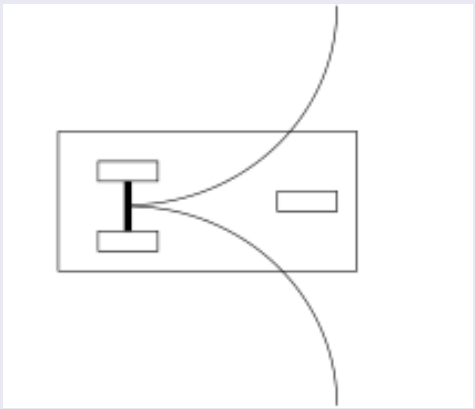
```
1: for all q ∈ GC do                                   ▷ Initialization
2:     if q ∈ GCL then
3:         d(q) ← −1                                   ▷ Obstacles init to −1
4:     else
5:         d(q) ← ∞                                    ▷ Unexplored init to ∞
6:     end if
7: end for
8: Q_cur ← {q_goal}
9: i ← 0
10: repeat                                  ▷ Each iteration is one step of the wave
11:     Q_next ← {∅}
12:     for all q ∈ Q_cur do
13:         d(q) ← i
14:         for all n ∈ neighbors(q) do
15:             if d(n) < 0 then                        ▷ n not explored and in GCL
16:                 Q_next.add(n)
17:             end if
18:         end for
19:     end for
20:     i ← i + 1
21:     Q_cur ← Q_next
22: until Q_cur is empty
```



Planification pour des robots non-holonomes

- Contrainte non-holonome $F(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$: non-intégrable
- Contrainte non-holonome $\Longrightarrow$ contrainte sur les directions de déplacement

- Conséquence sur la trajectoire calculée ?
- Toute trajectoire calculée sans CNH est-elle faisable avec la CNH ?



Deux familles de méthode :

- Approche basée sur chemin holonome
- Approche basée grille discrète

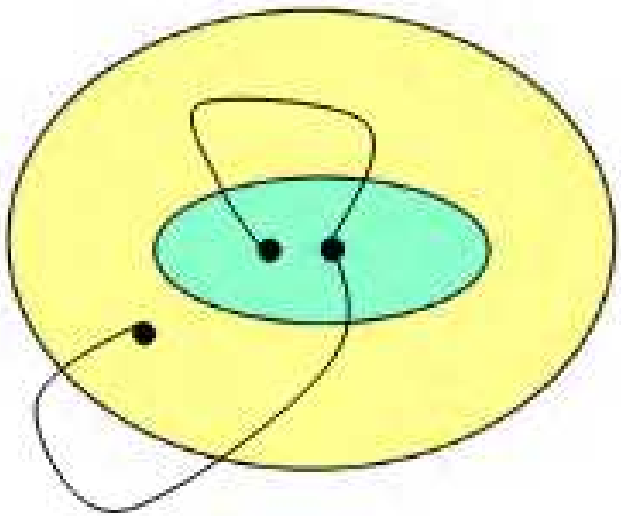
Approche basée sur chemin holonome

La contrainte NH $\Longrightarrow$ chemin local particulier ("steering method")
Propriété STC : Small Time Controllable

Small Time Controllable

Un chemin local entre deux configurations q et q', noté $L(q, q')$, est STC s'il vérifie la propriété suivante :

$$\forall \epsilon | B(q, \epsilon) \in ECL \Longrightarrow \exists \rho | \forall q^* \in B(q, \rho), L(q, q^*) \in B(q, \epsilon)$$



Existence d'un chemin pour un robot NH-STC

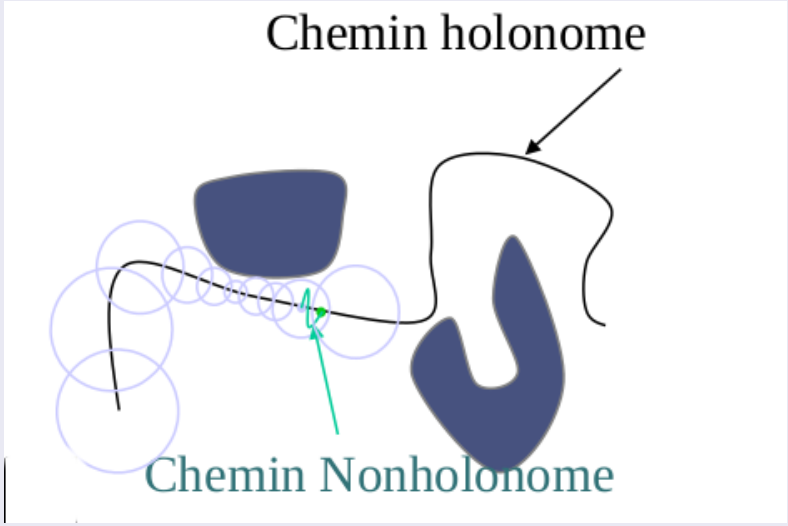
Existence d'un chemin libre de collision pour un robot NH qui est STC.

=

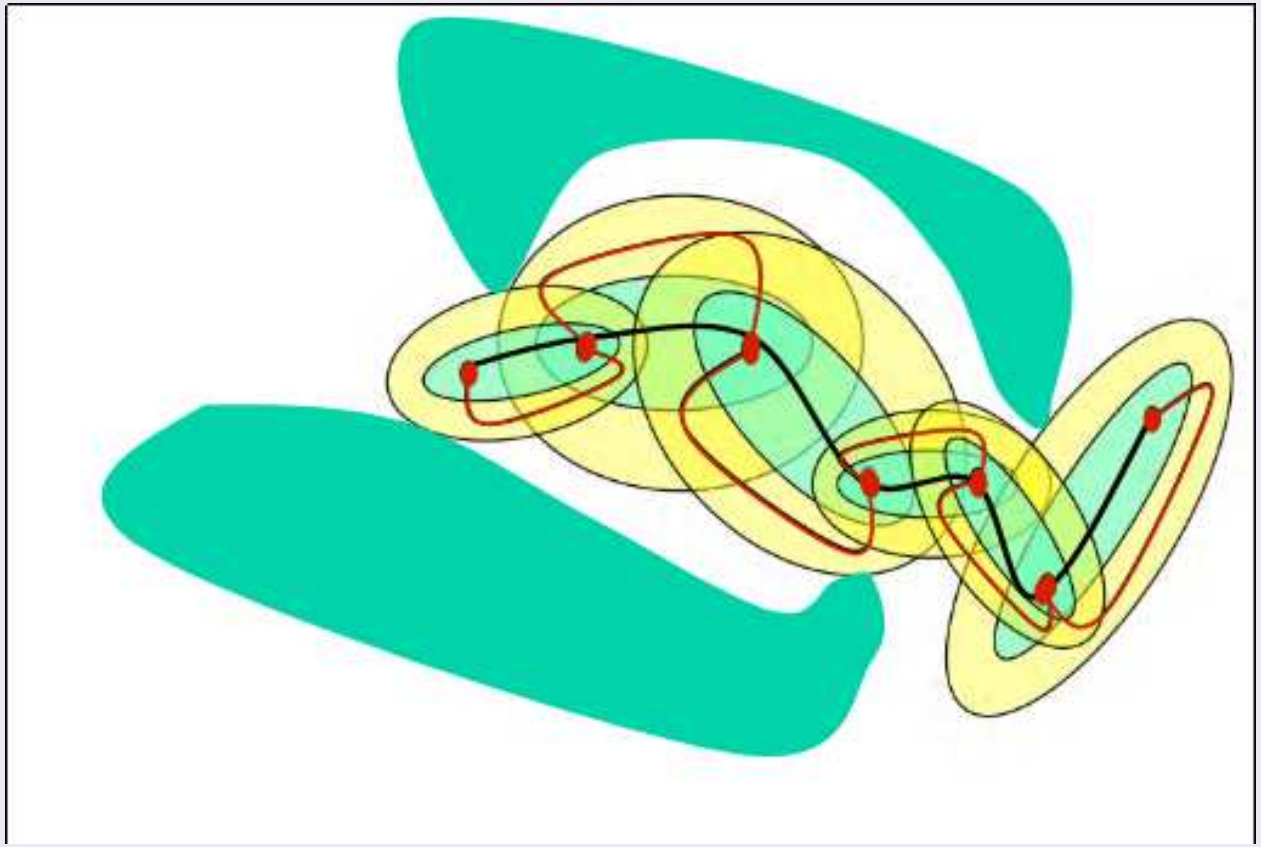
Existence d'une composant connexe dans EC libre

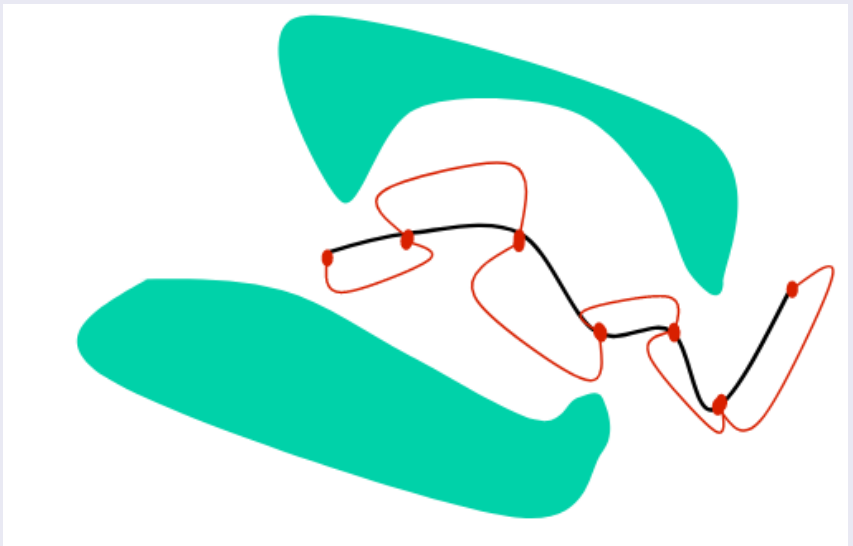
Approche basée sur chemin holonome

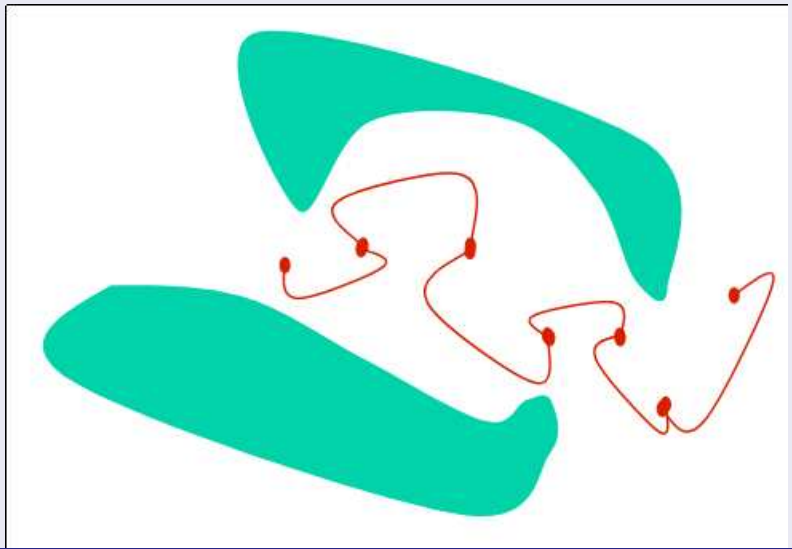
Pour tout système qui est *STC*, l'existence d'un trajectoire admissible est caractérisée par l'existence d'une trajectoire dans l'espace des configuration sans collision ni contact (*ECL*).



La propriété de STC garantit la complétude de la transformation de la trajectoire holonome en une trajectoire faisable par le robot non-holonome. Si la méthode de calcul de la trajectoire dans *ECL* est complète alors la méthode de calcul de la trajectoire pour le robot NH est complète.







Comment calculer un chemin local STC ?

Robot "unicycle" : évident

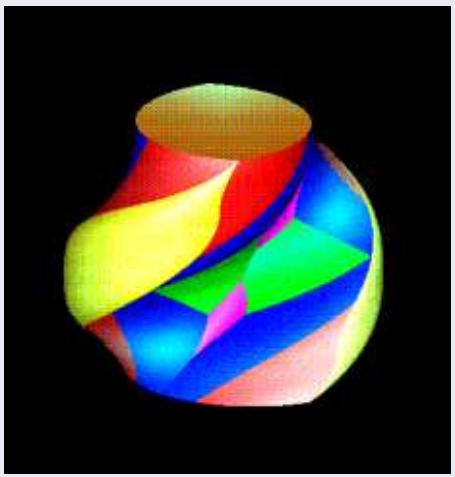
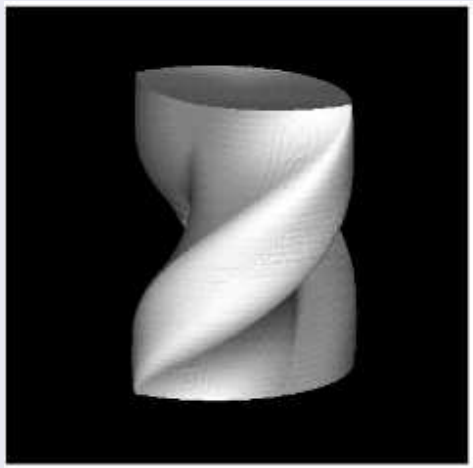
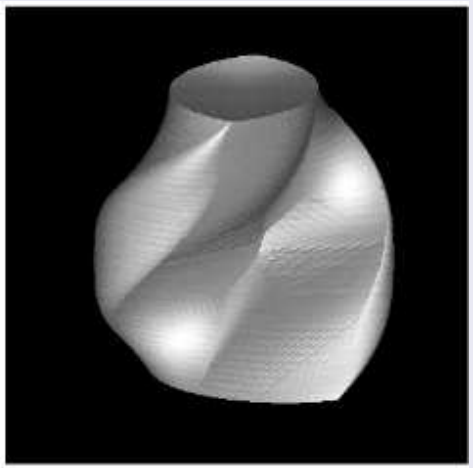
Robot type voiture avec rayon de giration borné ?

STC pour un robot type voiture

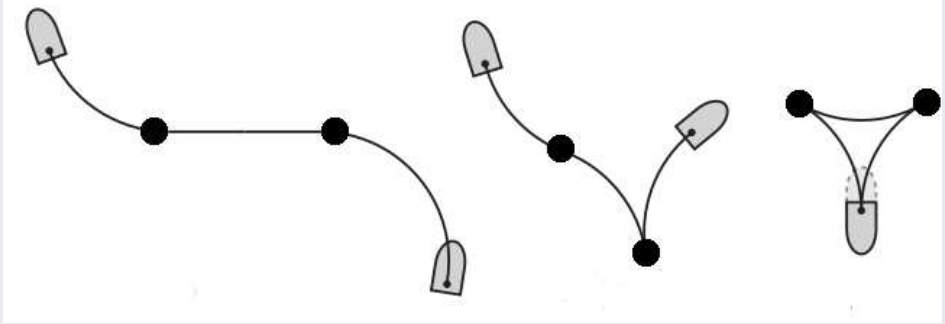
La trajectoire optimale en distance dans le plan sans obstacle entre deux configuration est calculable en temps fini.

Courbes de Reeds & Shepp (RS) de type $C|CSC|C$ avec :

- Arc de cercle C et segment de droite S
- 2 points de rebroussement $|$

Un robot capable de suivre une courbe RS vérifie la propriété de STC.

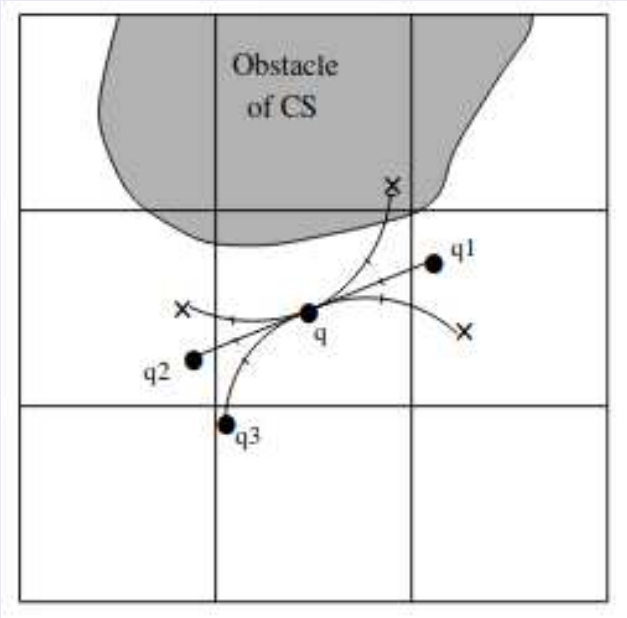


- $C|C|C$
- $CC|C$
- $C|CC$
- $CCa|CaC$
- $C|CaCa|C$
- $C|C_{\pi/2}SC$
- $CSC_{\pi/2}|C$
- $C|C_{\pi/2}SC_{\pi/2}|C$
- CSC

Et si marche arrière impossible ? CSC, CCC, STC ?

Approche par grille discrète

- Le robot est commandé par un nombre **fini** m de commande.
- Principe : propager un arbre à partir de la q_{init} en appliquant les commandes pendant un δt
- Nécessité de discrétiser l'espace de recherche



La propagation est effectuée en appliquant séparément les m commandes pendant δt. Les branches vont relier les différentes cellules discrètes de $ECL(W)$. On construit un arbre de recherche (heuristique). Les extrémités de ces chemins sont des nouvelles feuilles si elles se trouvent dans des cellules non connectées à l 'arbre.

- Algorithme quasi-complet
- Attention au réglage du pas d'intégration et du pas de discrétisation.

Exemple : Approche par grille discrète

Input: q_i et une région but G(but)

Output: une trajectoire Path ou Echec

Initialiser arbre T et liste OPEN avec q_i
While ((OPEN non vide) et Size(T) < Max_Size_Tree) **do**
    q ← first de OPEN et enlever de OPEN
    if (q ∈ G(but)) **then**
        return SUCCESS, return path
    EndIf
    If (q non proche d'une configuration déjà explorée) **then**
        Marquer q
        For All actions sur q **do**
            Intégrer l'action sur un temps t → q_{new}
            If path(q, q_{new}) est sans collision **then**
                q_{new} successeur de q dans T
                Calculer cost pour atteindre q_{new}
                Insérer q_{new} , trié par coût
            EndIf
        EndFor
    EndIf
EndWhile
Return ECHEC

Si $dim(EC)$ est très grande ? $\Longrightarrow$ Méthodes Probabilistes

Méthodes "classiques" : complexité exponentielle en fonction de la $dim(EC)$. Comment planifier une trajectoire si dim(EC) est grande ?

Construire un **graphe** G capturant la connexité de ECL permet de résoudre facilement le problème par une recherche de chemin dans un graphe.

MAIS

Construire une **représentation explicite** de la frontière de ECL est coûteux, difficile,... lorsque n croît.

Idée : Construire un graphe G capturant la connexité de ECL mais sans **construire une représentation explicite** de ECL

- Réseaux Probabilistes (PRM)
  - Apprentissage "aléatoire" d'un Graphe
  - Connexion au Graphe/Recherche chemin/Optimisation
- Méthodes de diffusion (RRT)
  - Propager "aléatoirement" un arbre de recherche vers le but

Exercice

Soit un robot mobile RM, holonome circulaire de rayon r, se déplaçant dans un environnement polygonal W plan parfaitement connu.
On dispose des systèmes ou algorithmes suivants :

- $QUADTREE$: algorithme de décomposition cellulaire approchée de W, indépendant de RM (entrée= W, sortie = quadtree sous forme d'arbre),
- LOC : système de localisation absolue permettant de connaître à chaque instant la configuration du robot sans erreur,
- $COM(q_{but})$: un système de commande du robot parfait qui permet d'atteindre sans erreur une configuration but, q_{but}, en ligne droite,

1. Expliquer comment planifier la trajectoire du robot RM sans collision, entre 2 configurations, avec les éléments fournis.
2. On suppose maintenant que le robot est non-holonome et que la commande COM calcule et suit une trajectoire respectant la contrainte (de type Reeds & Shepp). Expliquer comment planifier la trajectoire du robot RM.