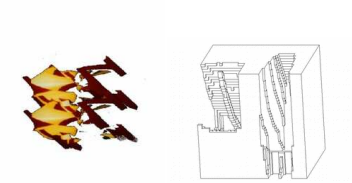
Méthodes Probabilistes

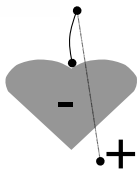
- Pourquoi ?
- Principes des approches probabilistes
- Méthodes PRM
  - Algorithme de base
  - Aspect pratiques de l'implémentation
  - Stratégies d'échantillonnage
- Méthodes de diffusion RRT
  - Algorithme de base
  - Variantes
- Exemples

Motivation d'une nouvelle algorithmique

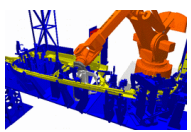
- Algorithme de planification complet -> trop lent, complexité exponentielle

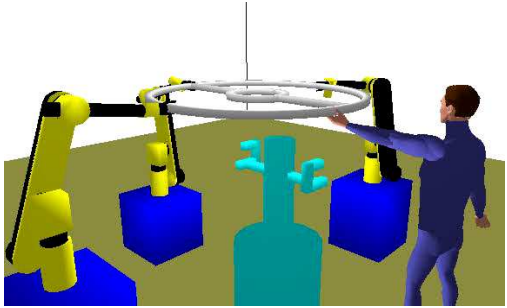


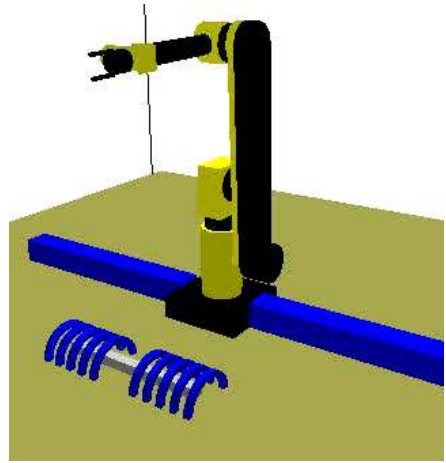
- Méthodes heuristiques -> peu fiable



- Applications réelles: beaucoup de ddl, géométrie complexe raisonnablement, contraintes







Méthode Probabiliste

Méthodes « classiques » = complexité exponentielle en fonction de la dimension de CS

Comment planifier une trajectoire si dimension de CS est grande (10, 100,….) ?

Introduire de l'aléatoire à la place de l'exactitude

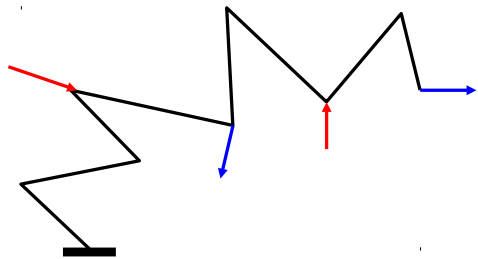
Compromis entre complétude et efficacité calculatoire

Complet en probabilité

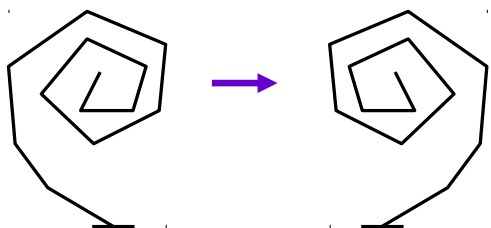
Algorithme complet en probabilité:
S'il existe un chemin solution, la probabilité que l'algorithme trouve la solution est une fonction qui tend vers 1 quand le temps tend vers l'infini.

Comment construire des algorithmes avec une probabilité qui croît rapidement vers 1 ?

Idée initiale:
Potentiel + Random Walk



Mais.....



Méthode Probabiliste

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

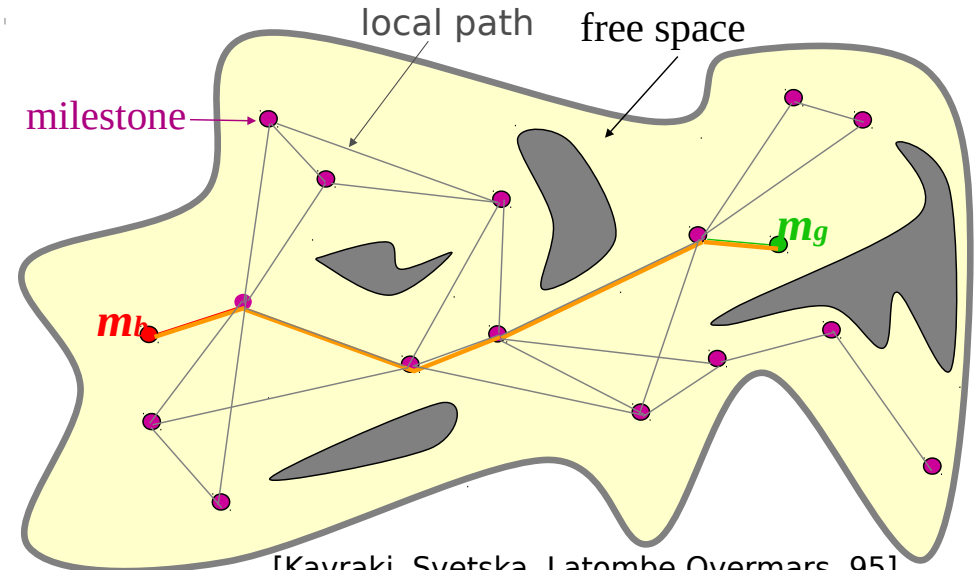
⇒Utilisation de technique aléatoire (au lieu de technique déterministe)

Deux types de problème:

- Requêtes multiples: on va calculer plusieurs chemins pour le même type de problème
  - Pré-calculer une roadmap (graphe) qui traduira la connexité de CS
  - Réutiliser cette roadmap à chaque requête de solution
  - Réseaux probabilistes/ sampling (PRM= Probabilistic RoadMap)
- Requêtes unique: on calcule un chemin pour chaque problème.
  - Résoudre un problème à partir d'un nœud spécifique par propagation de roadmap

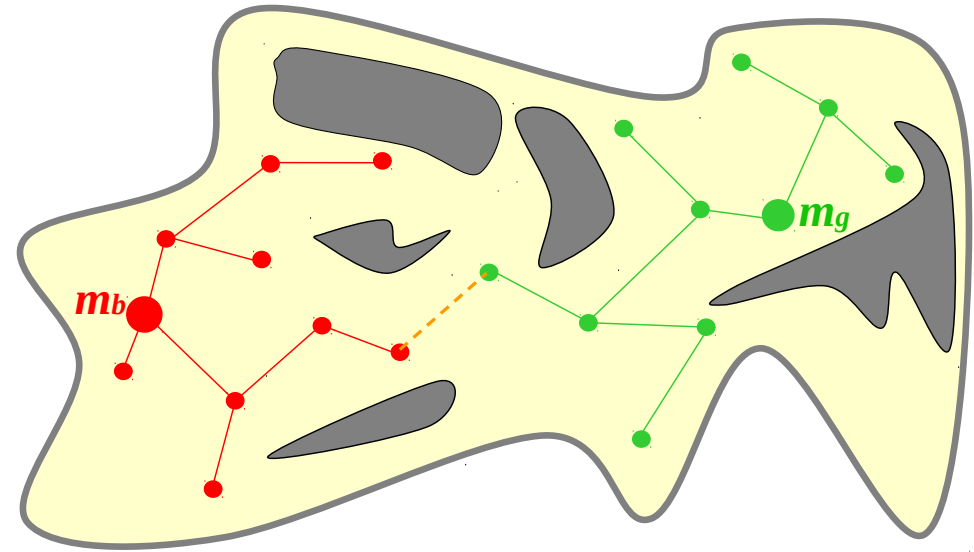
  Méthodes de diffusion/ diffusion (RRT= Rapidly-exploring Random Trees)

Probabilistic Roadmap (PRM)



[Kavraki, Svetska, Latombe,Overmars, 95]

Single-Query Planning (RRT)



[Lavalle, 00]

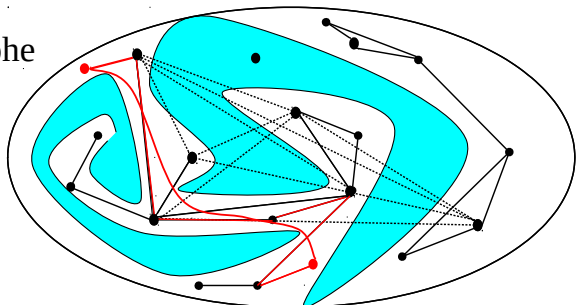
Méthode PRM: principes généraux

1/ Apprentissage du Graphe

2/ Connexion au Graphe

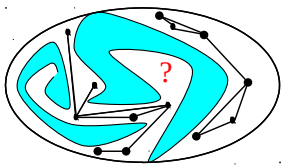
3/ Recherche chemin

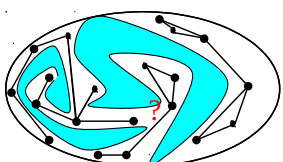
4/ Optimisation



Problème des Roadmaps PRM

- Couverture de CS :
  - toute configuration de CS peut être connecté à la roadmap
- Connexité :
  - Bijection entre les composantes connexes de la roadmap et les composantes connexes de F (free-space).





Deux règles impératives des méthodes PRM

- Les tests de collision des configurations et des connexions (chemins locaux) doivent être efficaces d'un point de vue temps CPU.
  ✉ Test de collision hiérarchique

- Un faible nombre de configurations et de chemins locaux sont suffisant pour capturer la connexité de l'espace libre F.
  ✉ Convergence exponentielle de l'exploration de F (probabilistic completeness)

Méthode Probabiliste PRM

Création d'un graphe pour capturer la connexité de l'espace libre associé à W par création aléatoire de :

*nœuds correspondants à des configurations libres

*arcs reliant ces nœuds correspondants à des chemins libres de collision

Fonctions essentielles:

- Stratégie d'échantillonnage
- Stratégie de connexion
- Détecteur de collision (géométrie)
- Méthodes locales (mouvement réalisable)

Variante algorithme de base

```
1.    Input = (nb, k)
2.    V <- 0; E <- 0; nb <- 0;
3.    Tant que (nb < Nmax) faire
4.          Tirer au hasard q dans CS
5.          Si q est libre alors
6.           V <- V ∪ { q }
7.           nb = nb + 1
8.           Vq <- voisins de q inclus dans V ( Dmax)
9.           Pour tout v de Vq par d(v, q) croissante faire
10.                if (q, q') ∉ E et L(q,q') ≠ Nil alors E <- E ∪ { (q,q') }
```

Paramètres à choisir pour « guider » l'aléatoire:

- Choix de q dans CS
- Choix des voisins: Valeur de k ou Seuil de distance maximum pour voisins (Dmax)

* Choix de n ou de Nmax = critère d'arrêt ?

* Méthode Locale L : construction d 'un chemin local entre 2 configurations

* Détecteur de collision

Influence importante sur temps de calcul et couverture de l 'espace libre

Echantillonnage

Echantillonnage uniforme sur F (CS free)

- distribution de probabilité uniforme sur les ddl
- générateur aléatoire de nombre

- Echantillonnage quasi aléatoire:

  - générateur aléatoire de nombre à partir d'une séquence déterministe

Comment mesurer l'échantillonnage ?

Soit P un ensemble de points d'un espace X et N le nombre de points de P
Comment évaluer la répartition des points de P dans X ?

Echantillonnage

Soit P un ensemble de points d'un espace X et N le nombre de points de P
Comment évaluer la répartition des points de P dans X ?

Soit une collection de sous-ensemble de X, R (espace d'intervalles) et soit une fonction de mesure (volume) μ d'un ensemble.

(le volume relatif de A p/r à X) – (fraction d'échantillons contenue dans $A \in R$) est:

$$\left| \frac{\mu(A)}{\mu(X)} - \frac{|P \cap A|}{N} \right|$$

La Disparité/Discrepancy pour k échantillons d'un ensemble de point P avec une collection d'espace d'intervalles R définie sur X, est définie par:

$$D(P,R) = \sup_{A \in R} \left\{ \left| \frac{\mu(A)}{\mu(X)} - \frac{|P \cap A|}{k} \right| \right\}$$

Exemple

Discrepancy donne une mesure sur comment sont distribuées uniformément les points sur X

Echantillonnage

La répartition/dispersion fournit une mesure sur la plus large portion de X qui ne contient aucun point de P.
Soit ρ la fonction de distance alors $\min_{p \in P} \rho(x, p)$ donne la distance de x au point le plus proche de P

La répartition/dispersion δ d'un ensemble de point P avec la métrique ρ est définie par:

$$\delta(P, \rho) = \sup_{x \in X} \min_{p \in P} \rho(x, p)$$

Exemple

Pour X = [0, 1] le séquence qui minimise la répartition/dispersion et la disparité/discrepancy est la séquence de Van der Corput

Echantillonnage

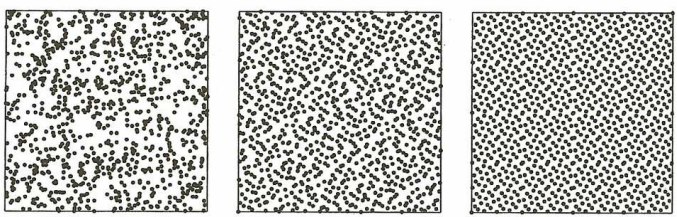
Séquence de Van der Corput pour X = [0, 1]

Soit $a_i \in \{0, 1\}$ alors $n = \sum a_i \cdot 2^i$

Le nième élément de la séquence de VdC est:

$$\Phi(n) = \sum a_i \cdot 2^{-(i+1)}$$

i	Sequence	Binary	Binary	Corput	Points in $[0, 1]/\sim$
1	0	.0000	.0000	0	
2	1/16	.0001	.1000	1/2	
3	1/8	.0010	.0100	1/4	
4	3/16	.0011	.1100	3/4	
5	1/4	.0100	.0010	1/8	
6	5/16	.0101	.1010	5/8	
7	3/8	.0110	.0110	3/8	
8	7/16	.0111	.1110	7/8	
9	1/2	.1000	.0001	1/16	
10	9/16	.1001	.1001	9/16	
11	5/8	.1010	.0101	5/16	
12	11/16	.1011	.1101	13/16	
13	3/4	.1100	.0011	3/16	
14	13/16	.1101	.1011	11/16	
15	7/8	.1110	.0111	7/16	
16	15/16	.1111	.1111	15/16	

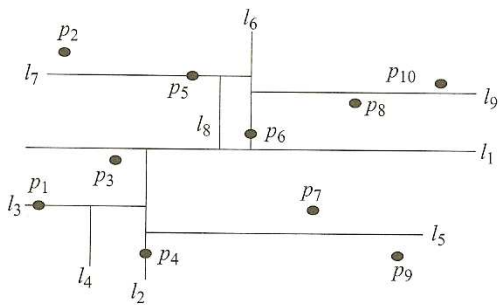


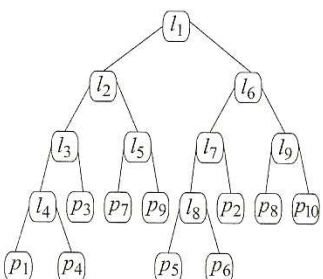
Généralisation de la séquence de VdC => séquence de Halton, séquence de Hammersley

Choix des voisins

Choix des voisins: comment déterminer les N plus proches voisins ? => Kd-tree

- Entrée = un ensemble S de n points en dimension d
- Sortie: un arbre binaire qui décompose S en cellules contenant un nombre de points semblables. Découpage récursif de S en deux sous-ensemble de dimension « égale », sur chaque dimension de S qui change à chaque niveau.
- Kd-tree de n points dans un espace de dimension d utilise O(nd) place mémoire et la construction du Kd-tree est en O(dn. Log(n))
- La recherche est en $O(n^{(1-1/d)} + m)$ avec m nombre de points retournées. Lorsque d devient grand, le coût devient linéaire.
- Exemple en 2D:



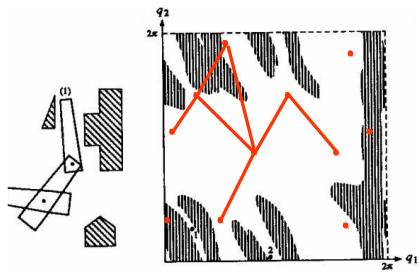


Test de Collision

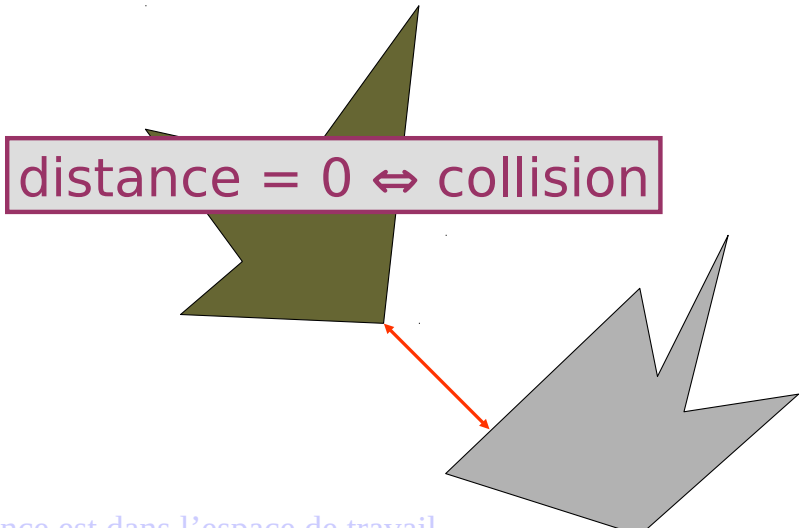
- Echantillonnage de CS

  => nécessité d'avoir des algorithmes efficaces

- Collision vs. Distance
- Statique vs. Dynamique



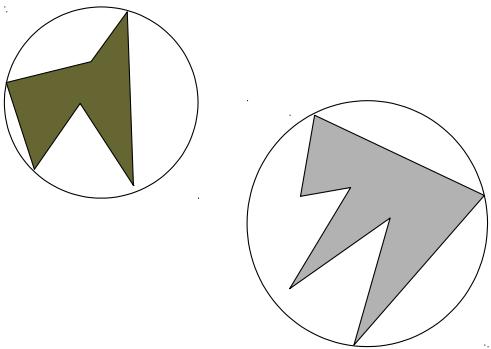
Test de Collision vs. calcul de Distance



Distance est dans l'espace de travail
la distance entre les points les plus proches

Pour deux objets A et B
- soit on peut calculer la distance entre eux
- soit on détermine s'il sont en collision ou non

Il est plus facile de calculer la collision que la vraie distance



… mais la distance donne une information supplémentaire très utile (même approximativement)

Algorithme général

1. d= ∞
2. Pour chaque paire d'objet(L,O) du robot et des obstacles de W faire:
   a. Calculer la distance δ entre L et O
   b. Si δ = 0 alors retourner collision
   c. Si δ < d alors mettre d à δ
3. Retourner d

$$\rho(x,x') = \left(\sum_{i=1}^{n} \left| x_i - x_i' \right|^p \right)^{1/p}$$

Distance euclidienne p=2, L_2
Distance de Manhattan p=1, L_1

$$L_\infty(x,x') = \max_{1 \le i \le n} \left\{ \left| x_i - x_i' \right| \right\}$$

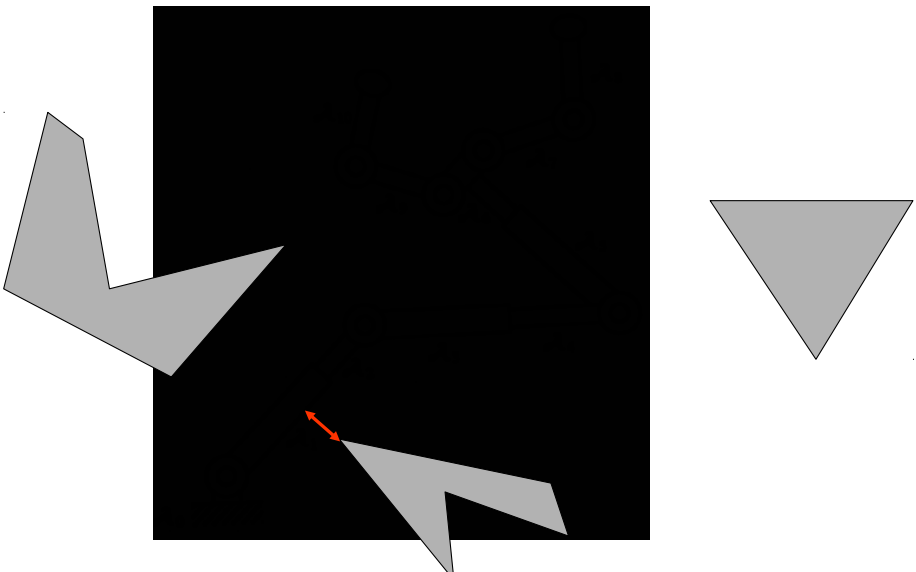
Distance infini, L_∞

Détection de collision pour:

- Deux objets:
  - Objets convexes
  - Forme arbitraire d'objet
- Collection d'objets, e.g., robot articulé + obstacles non statiques+ …
- Objets déformable
- Auto-collision (self-collision)

Exemple: https://github.com/flexible-collision-library /fcl

Articulated Robot

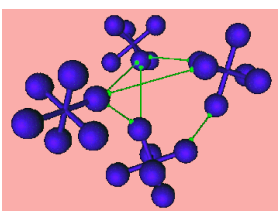


Principales approches

- Hierarchical bounding volume
  (pre-calcul: sphères, ABB, OBB, kd-tree)

- Feature tracking

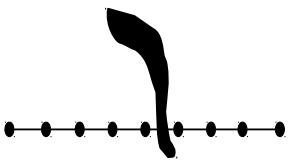
(paires de plus proches objets)



Test de Collision sur chemin locaux L

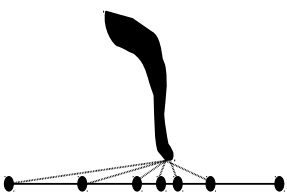
Résolution fixe

(test de collision incrémental)



Résolution dichotomique

(lazzy collision checking)

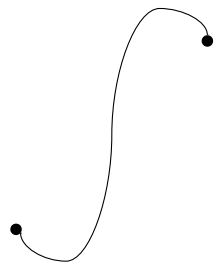


Adaptive dynamic checking

(more expensive distances calculations, guarantee to never miss a collision)

PRM : Méthodes locales

- Calculer un chemin local faisable connectant deux échantillons
- Besoin d'avoir une méthode locale à la fois déterministe et rapide pour de bons résultats.



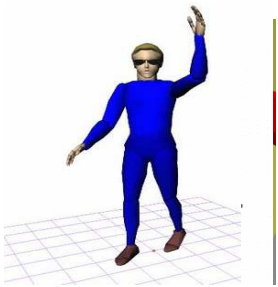
Déterministe : éliminer le besoin de mémoriser les chemins locaux

Rapide : pour construire efficacement les roadmap et pouvoir répondre quasi-instantanément à une requête

- Exemples

PRM : Méthodes locales

- Linéaire
- Manhattan
- Reeds-Sheep
- Combinaisons de méthodes (Lin/RS)
- Chaînes fermées
- Types de mouvements (walk…)



Méthode Locale

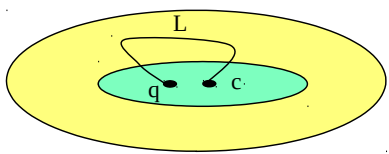
Soit un robot R localement commandable. Une méthode locale L pour R vérifie la TPG (Propriété Topologique Générale) 🔔

$\forall \varepsilon > 0, \exists \delta > 0 \mid \ \forall q \in ECL, B_\delta(q)$ soit contenue dans le domaine d'accessibilité $R_{L,\varepsilon}(q)$

avec $R_{L,\varepsilon}(q) = \{ c \in B_\delta(q), \mid \ L(q,c) \subset B_\varepsilon(q) \}$

(chaque configuration c peut être atteinte par un chemin L en ne s'éloignant pas plus de ε de q)

- Small time controllable systems
  - Méthodes locales STC



Si ECL est borné, si le robot est localement commandable et si la TPG sur L est vérifiée alors la méthode est complète en probabilité

Distribution aléatoire uniforme => Probabilité(Boule Bi contient un nœud) -> 1 quand t-> infini

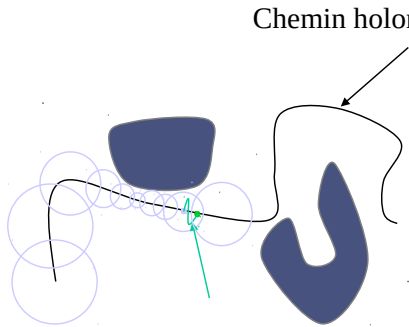
Méthode Locale

Existence d'un chemin libre de collision

=

Existence d'une composant connexe dans F (CS libre)

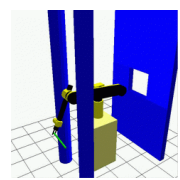
Tout chemin libre de collision peut être approximé par une séquence finie de chemin faisable par une méthode locale STC

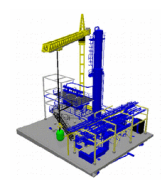


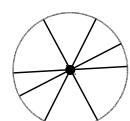
Méthode Locale

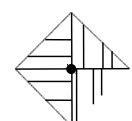
- Contrôlabilité vs. Topologie de CS
- Système STLC
- STLC: méthode locale

Système holonome: aucun problème



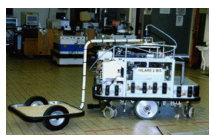






Méthode Locale

- Controllabilité vs. Topologie de CS
- Système STLC
- STLC:méthode locale



Système non holonome : plus difficile

Exemple: voiture, avion, camion,…..

Stratégies d'échantillonnage

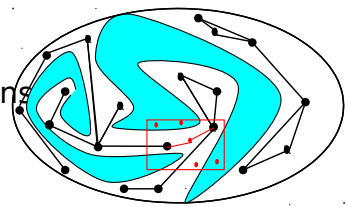
Espace contraint

- Dilatation de l'espace libre
  - Construction de G' autorisant le robot à pénétrer les obstacles
  - Repousser les nœuds de G' dans ECL -> G
- Utilisation des obstacles
  - Engendrer des nœuds proches des surfaces des obstacles
- Echantillonnage gaussien
  - Ajouter des configurations dans des régions difficiles de ECL. Idée: la probabilité d'ajouter une q dépend du nombre de conf. interdites autour de q

- Visibilité
  - Le domaine de visibilité est lié à la méthode locale L (on peut atteindre tous les nœuds de Visi à partir d'une configuration gardienne)
  - Domaine de Visibilité: $Visi_L(q^*) = \{ q \in ECL, \mid \quad L(q^*, q) \subset ECL \}$
  - Forcer l'algorithme à explorer en priorité les régions non couvertes

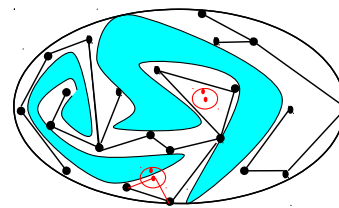
Stratégie d'échantillonnage

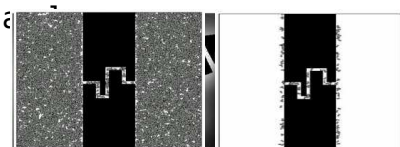
- Echantillonnage uniforme
- Echantillonnage multi-étape
- - étendre la roadmap dans les "régions difficiles"
- utiliser une fonction de poids heuristique w(q)
- [Kavraki-Svetska]



Stratégie d'échantillonnage

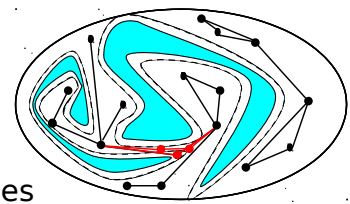
- Echantillonnage uniforme
- Echantillonage multi-étape
- Obstacle-sensitive sampling
- - placer beaucoup d'échantillons proches de la frontière des obstacles OBPRM [Amato],
- Echantillonnage Gaussien [Overma





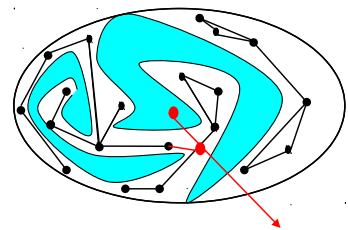
Stratégie d'échantillonnage

- Echantillonnage uniforme
- Echantillonage multi-étape
- Obstacle-sensitive sampling
- Dilatation de l'espace libre F
- - permettre une pénétration dans les obstacles pour franchir (widen) les passages étroits [Kavraki]
- - mais les calculs de distance de pénétration sont plus complexe



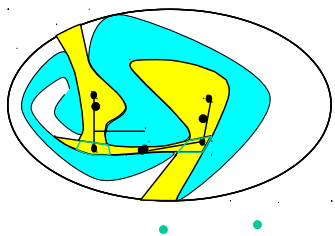
Stratégie d'échantillonnage

- Echantillonnage uniforme
- Echantillonnage multi-étape
- Obstacle-sensitive sampling
- Dilatation de l'espace libre F
- Echantillonnage selon un axe Médian
- - essayer de pousser les échantillons dans l'espace libre le long d'une direction aléatoire [Kavraki]



Stratégie d'échantillonnage

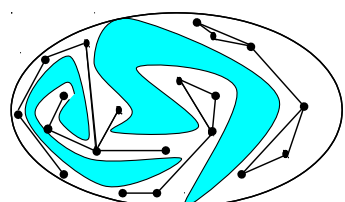
- Echantillonnage uniforme
- Echantillonnage multi-étape
- Obstacle-sensitive sampling
- Dilatation de l'espace libre F
- Echantillonnage selon un axe Médian
- Echantillonnage par Visibilité
- - utiliser la notion d'atteignabilité pour calculer une "petite" roadmap et contrôler l'étape de fin [Siméon/Nissoux]



Stratégie d'échantillonnage

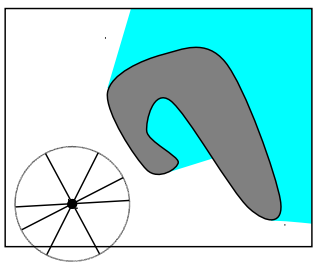
- Connection aux k-plus proches voisins
- connections acycliques
- ...

- But : limiter le nombre de connections
- Problème : quand stopper l'échantillonnage avec des garanties de couverture de l'espace libre

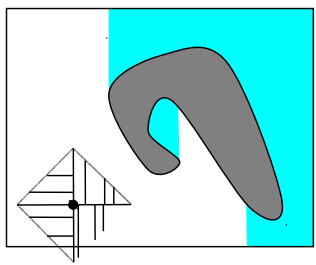


PRM-Visibilité

Ensemble de Visibilité:



Euclidienne

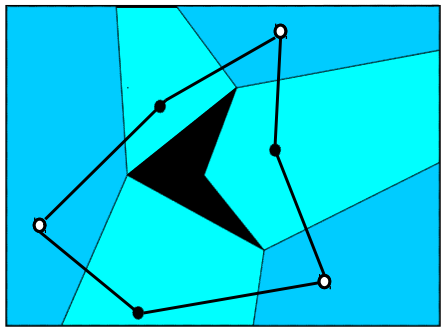


Type Manhattan

PRM-Visibilité

Calcul :

Pas de connaissance explicite des ensembles atteignables (domaines de visibilité)

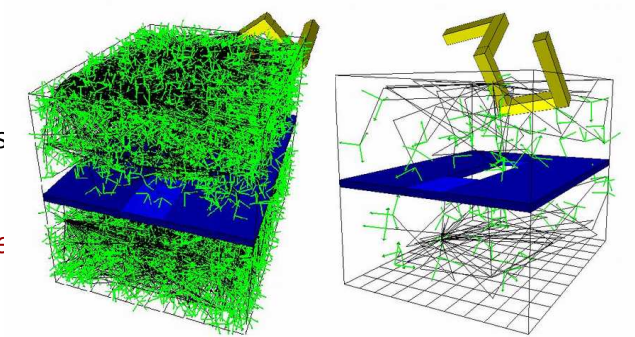


- *Générer des gardiens (qui apportent de l'information et des connecteurs (qui réduisent le nombre de composantes connexes) aléatoirement*
- *Arrêter après #try échecs*

PRM-Visibilité

Avantages :

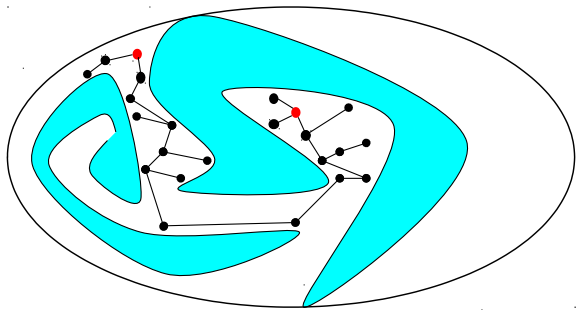
- *connexité des espaces complexes peut être représenté par des roadmaps de faible taille*
- *contrôler l'algorithme par une estimation de la couverture de CS libre (#try $^{-1}$)*



PRM vs. PRM-Visibilité

Puzzle

Diffusion : principe général



Méthode RRT= Rapidly-exploring Random Tree

Simple_RRT(q0)

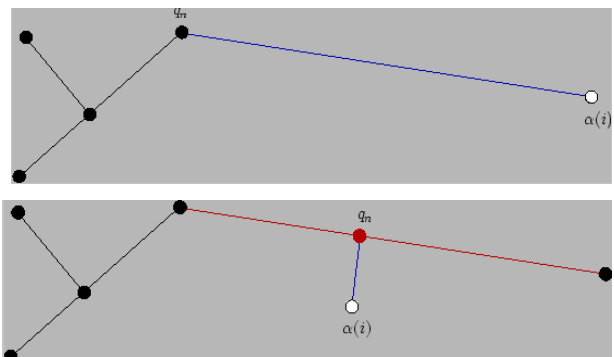
1 G.init(q0)

2 For i=1 to k do

3 G.add_vertex(α (i))

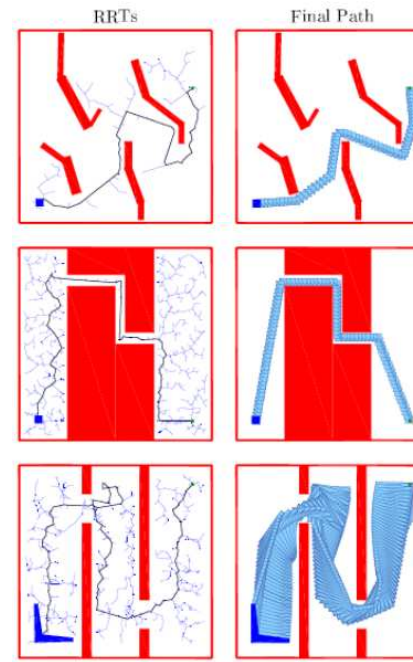
4 qn ⟵ NEAREST(S, α(i))

5 G.add_edge(qn, α(i));



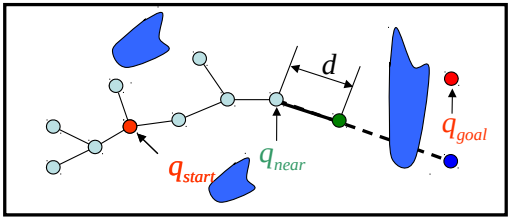
```
CONNECT(T, q)
 1  repeat
 2      S ← EXTEND(T, q);
 3  until not (S = Advanced)
 4  Return S;
```

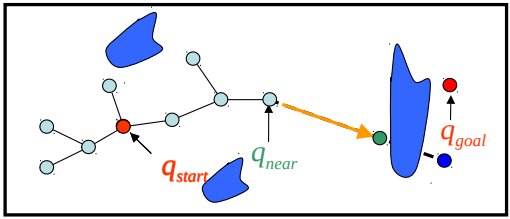
```
RRT_CONNECT_PLANNER(q_init, q_goal)
 1  T_a.init(q_init); T_b.init(q_goal);
 2  for k = 1 to K do
 3      q_rand ← RANDOM_CONFIG();
 4      if not (EXTEND(T_a, q_rand) = Trapped) then
 5          if (CONNECT(T_b, q_new) = Reached) then
 6              Return PATH(T_a, T_b);
 7      SWAP(T_a, T_b);
 8  Return Failure
```

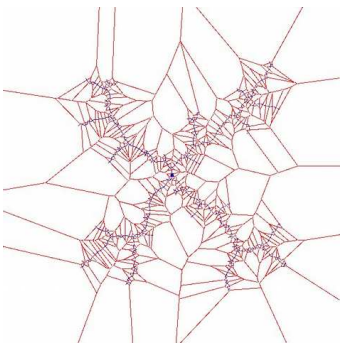


RRT

Deux variantes

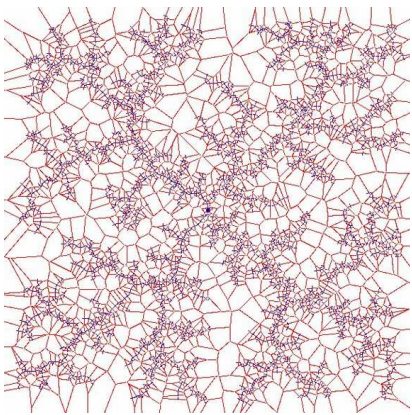






Selection proportionnelle aux régions de Voronoï

Complet en probabilité



Méthodes Probabilistes de MP

Une “rupture” algorithmique :

- *Schéma général de planification*
  pour une large classe problèmes
- *Très efficace en pratique*
  difficulté des problèmes
- *Conceptuellement robuste*
  structures de données simple

➢ *Cadre de travail pratique*

OMPL: Open Motion Planning Library

HPP: Humanoid Path Planner

Motion planning

Kineo

Kineo software components deliver a range of solutions for automatic path planning, high speed collision detection and predictive cable performance. These technologies are used by automotive, aerospace, energy, shipbuilding and manufacturing industries in software applications for robotics and digital mock-up, including assembly/disassembly verification, human simulation and accessibility studies.