IFT608/IFT702
Planification en intelligence artificielle

Planification de trajectoires avec évitement d'obstacles

Professeur: Froduald Kabanza

Assistants: Jordan Félicien Masakuna

Objectifs

- Se familiariser avec les approches générales de modélisation d'un robot et de son environnement pour planifier les trajectoires.

- Connaître les deux grandes familles d'algorithmes pour planifier les trajectoires d'un robot:
  - » Approches exactes qui modélisent la topologie de l'espace de configuration.
  - » Approches (approximatives) qui échantillonne de l'espace de configuration.

Sujets couverts

- Énoncé du problème
- Cadre de résolution générale
- Représentation et transformation géométriques
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- OMPL

Reference: Steven Lavalle. *Planning Algorithms*. Cambridge University Press, 2006. Disponible en ligne: http://planning.cs.uiuc.edu/. Sections couverts: 3 à 6 et 13 à 14.

Plan

- Énoncé du problème
- Exemples d'applications
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Énoncé du problème

- *Planification de trajectoire*: *Calculer une trajectoire géométrique d'un solide (articulé ou non) sans collision avec des obstacles statiques.*

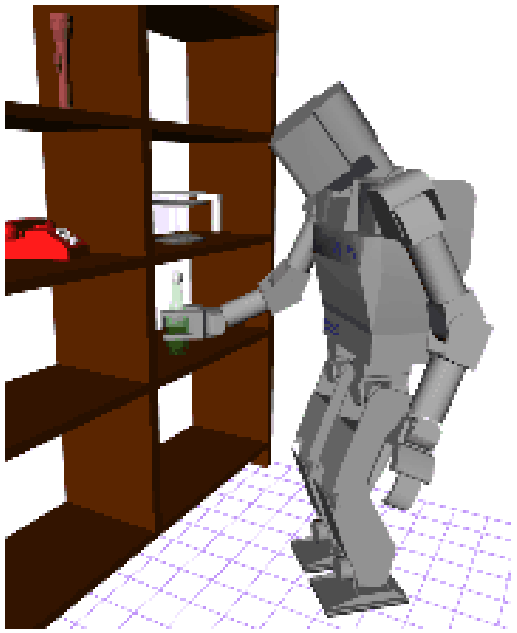
Entrée:
- Géométrie du robot et des obstacles
- Cinétique du robot (degrés de liberté)
- Configurations initiale et finale

↓

Planificateur de trajectoires

↓

Sortie:
- Un chemin sans collision joignant la configuration initiale à la configuration finale



S. Kagami. U of Tokyo

Plan

- Énoncé du problème
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Robots mobiles

ASIMO
Honda

Lignes d'assemblage

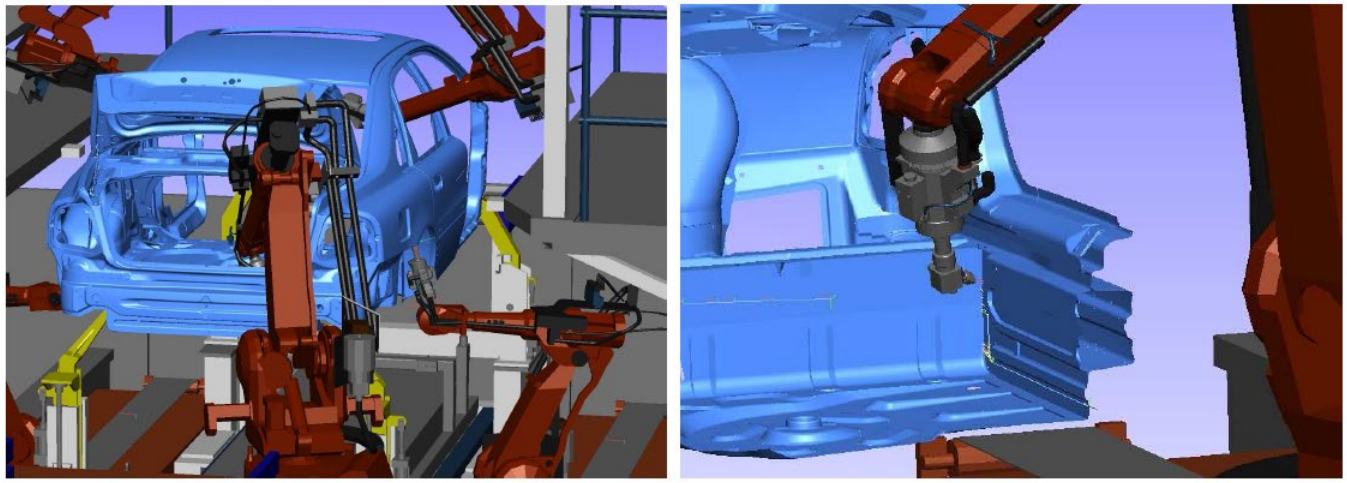
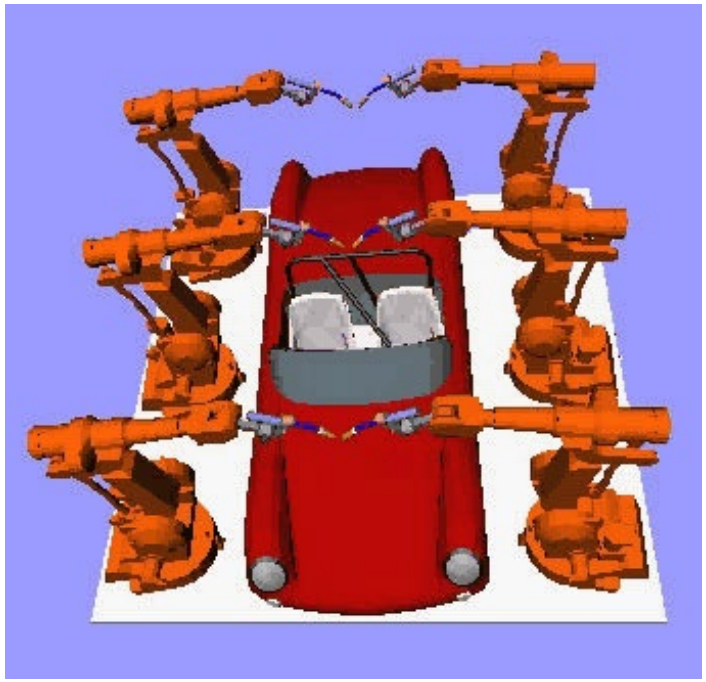
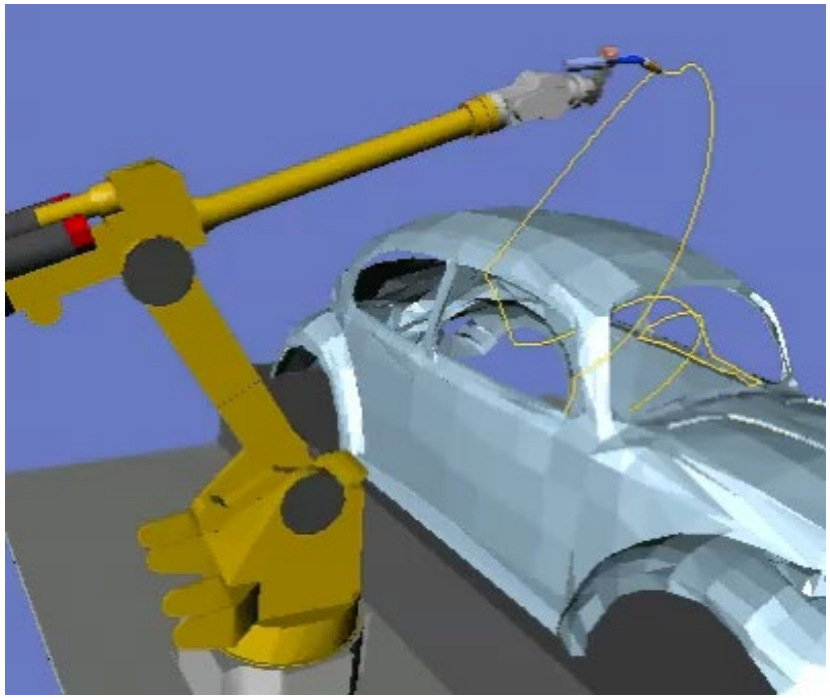


Figure 1.4: An application of motion planning to the sealing process in automotive manufacturing. Planning software developed by the Fraunhofer Chalmers Centre (FCC) is used at the Volvo Cars plant in Sweden (courtesy of Volvo Cars and FCC).

Ligne d’assemblage





Motion Planning Kit (Jean-Claude Latombe – Stanford U.)

Animations cinématographiques

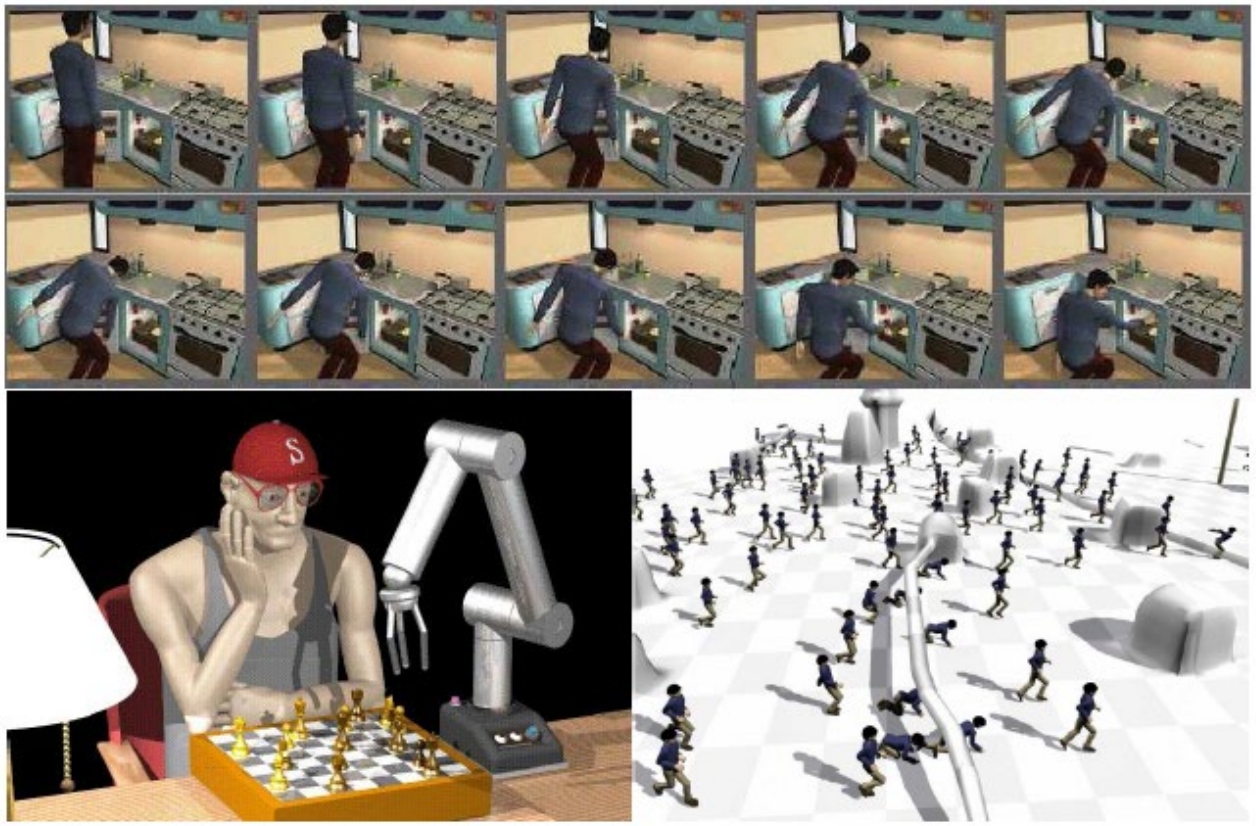
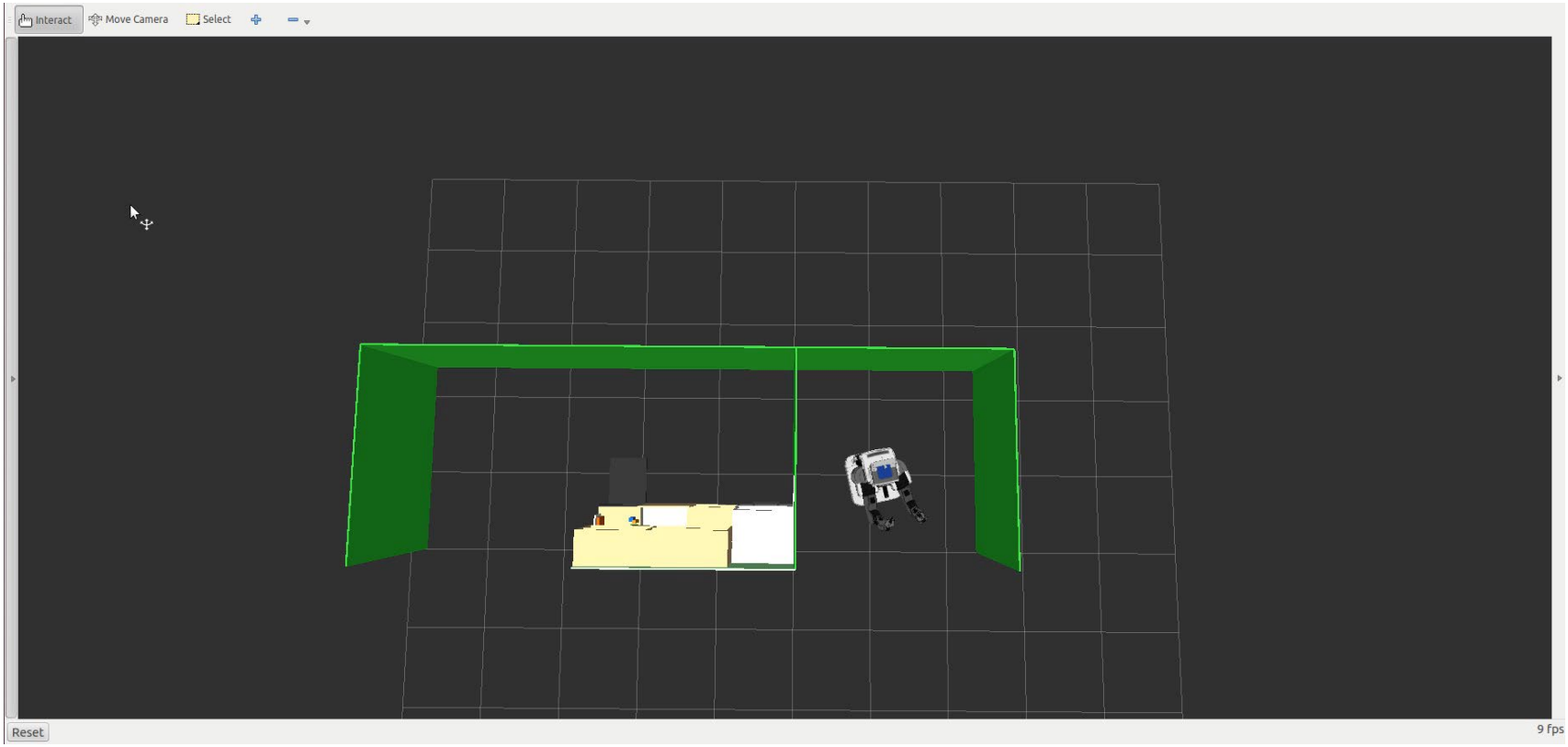
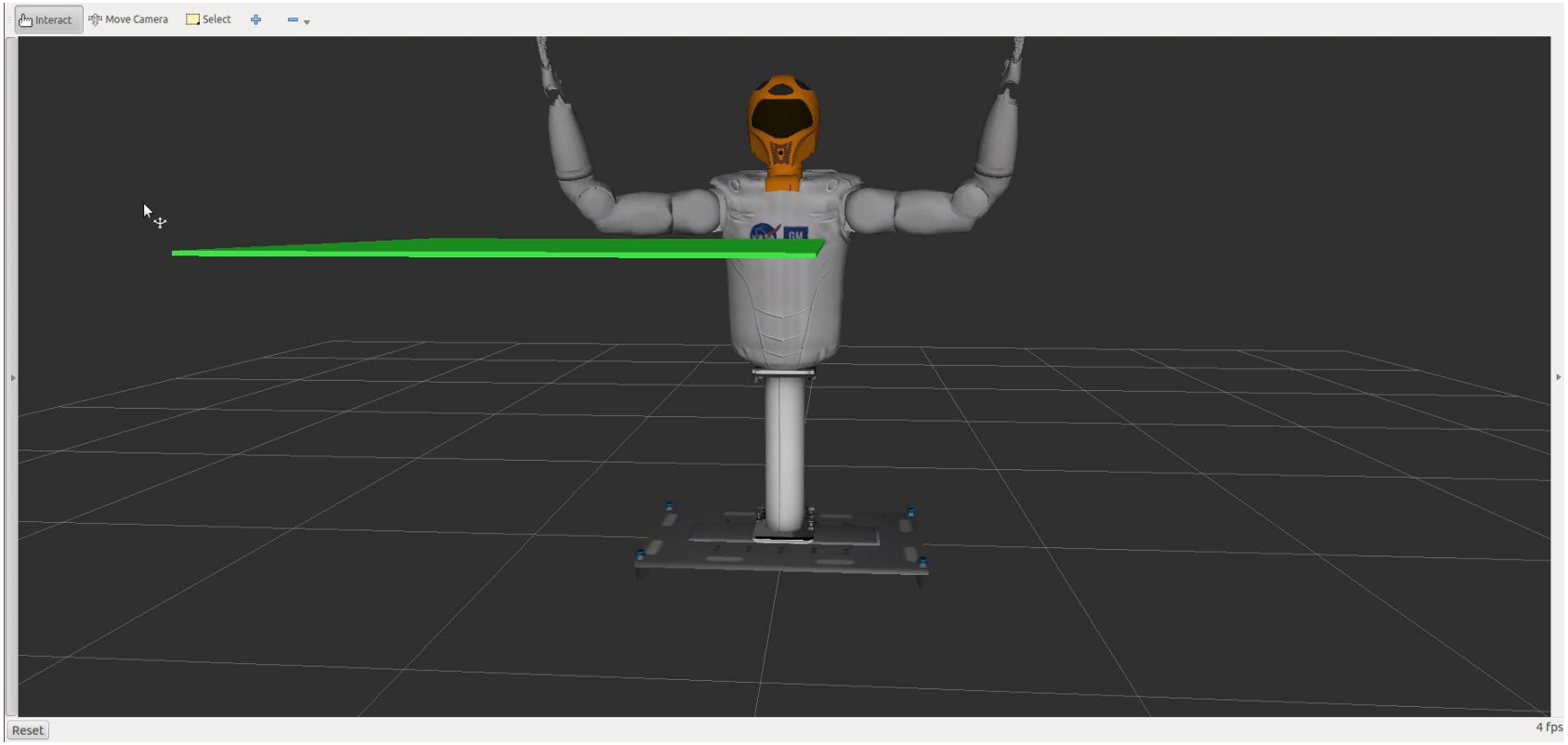


Figure 1.8: Across the top, a motion computed by a planning algorithm, for a digital actor to reach into a refrigerator [499]. In the lower left, a digital actor plays chess with a virtual robot [545]. In the lower right, a planning algorithm computes the motions of 100 digital actors moving across terrain with obstacles [592].

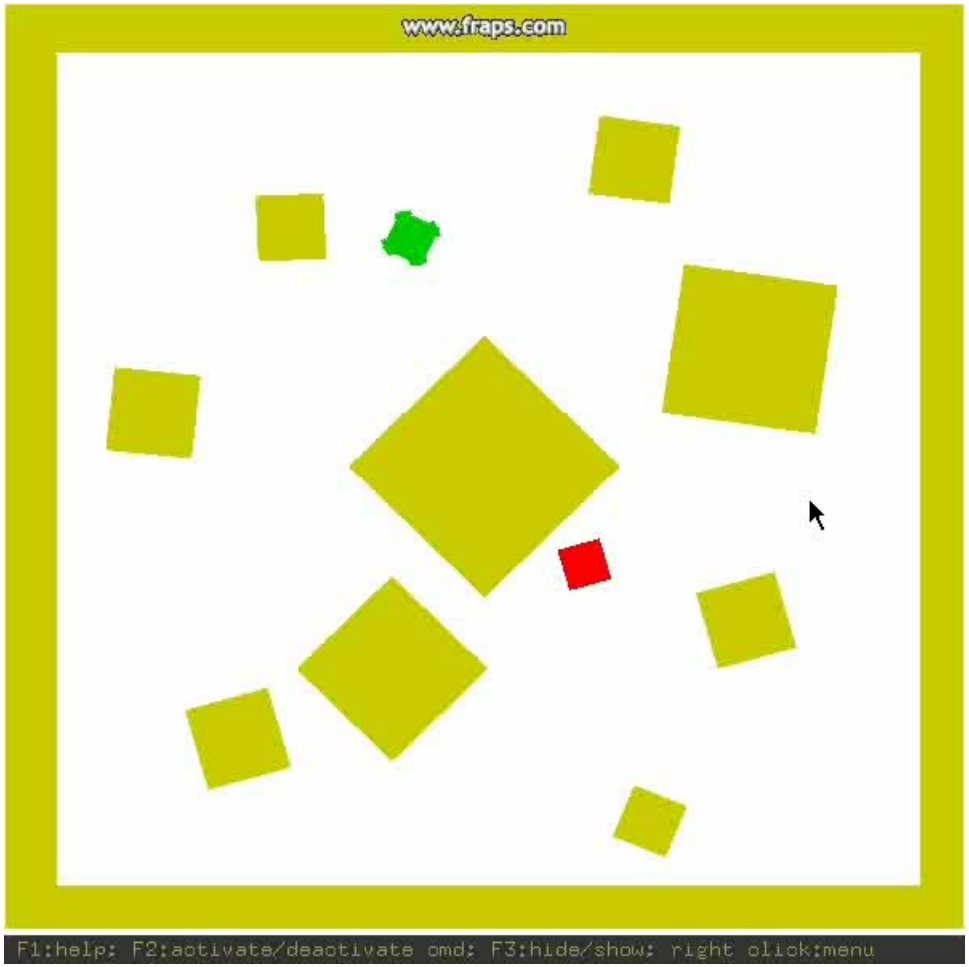
Projet ROS - IFT 608 / IFT702



Projet ROS - IFT 608 / IFT702

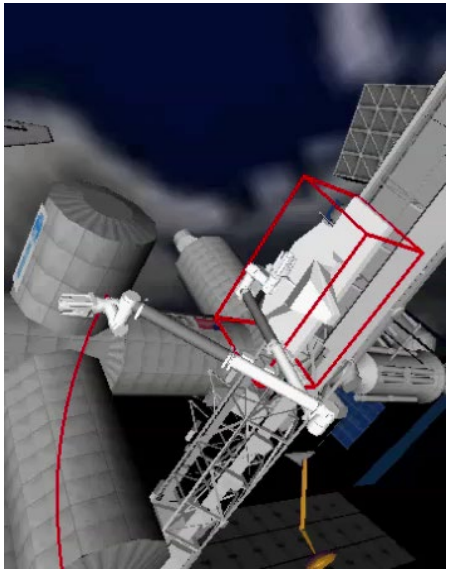


Projet IFT 608 + Cours-Projet



S. Chamberland &
D. Castonguay (2010)

Projet doctorat



Khaled Belghith (2010)

Plan

- Énoncé du problème
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Énoncé du problème

- *Planification de trajectoire*: *Calculer une trajectoire géométrique d'un solide (articulé ou non) sans collision avec des obstacles statiques.*

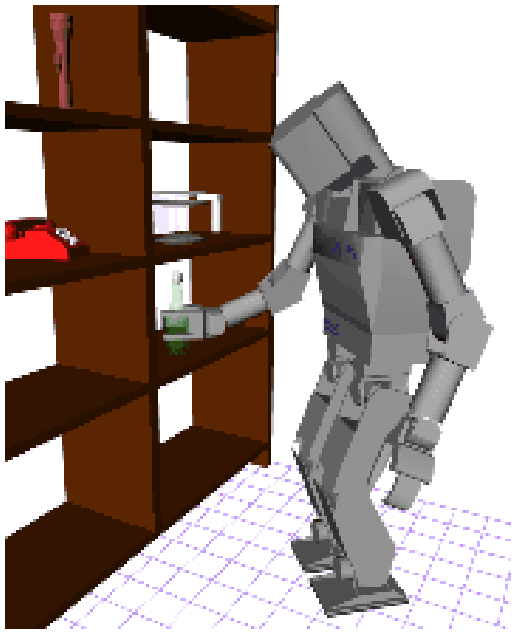
Entrée:
- Géométrie du robot et des obstacles
- Cinétique du robot (degrés de liberté)
- Configurations initiale et finale

↓

Planificateur de trajectoires

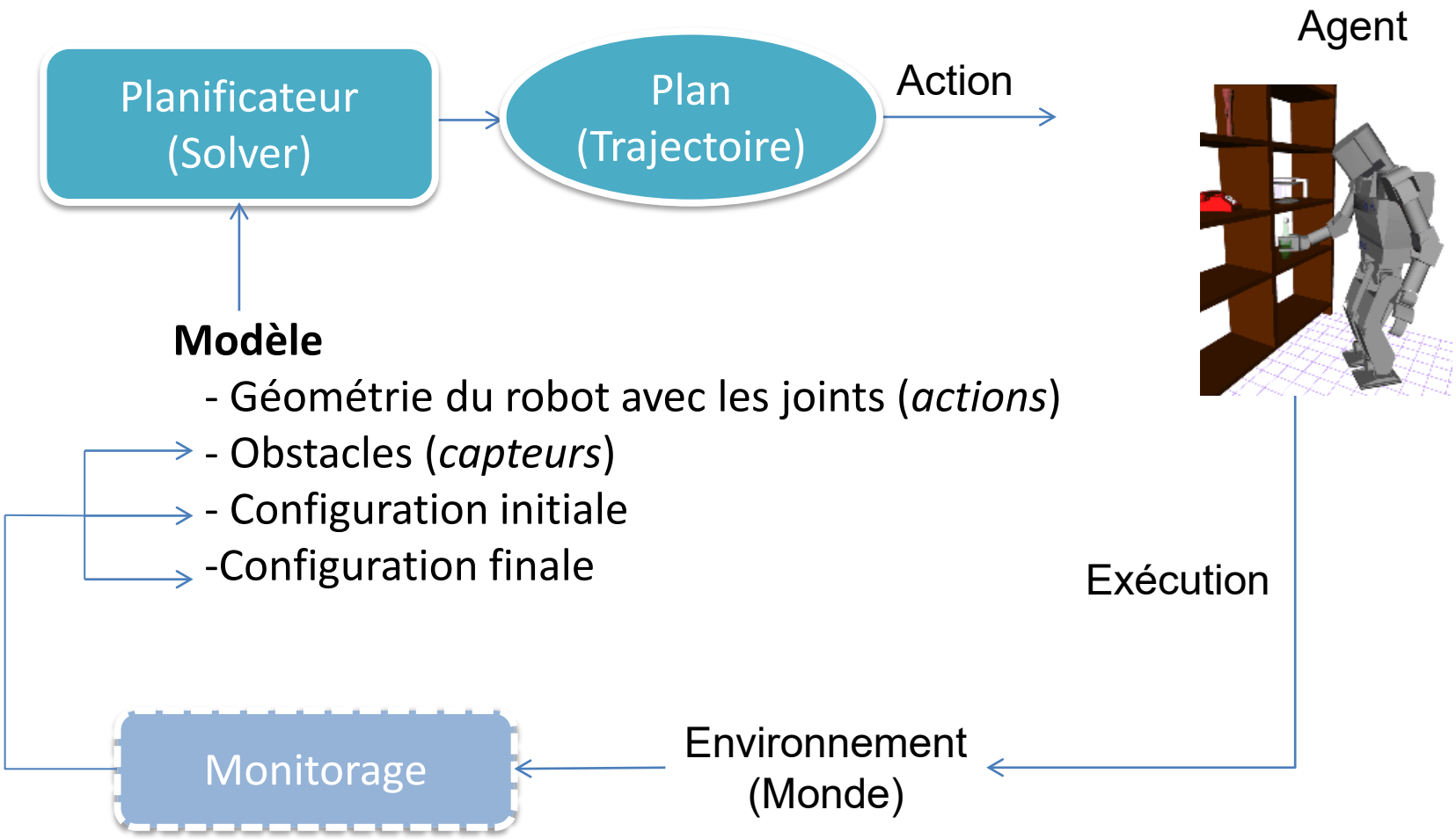
↓

Sortie:
- Un chemin sans collision joignant la configuration initiale à la configuration finale

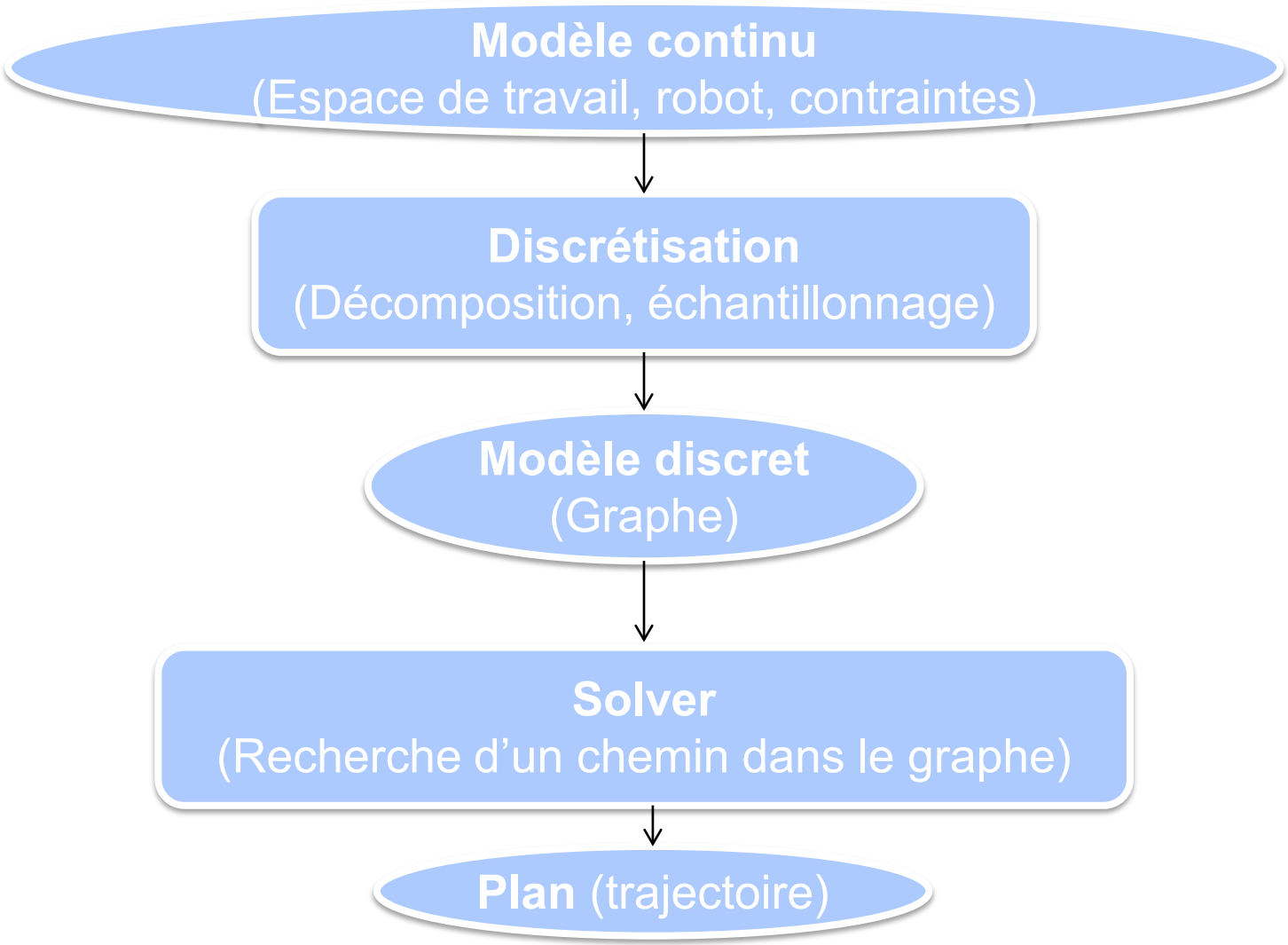


S. Kagami. U of Tokyo

Boucle de contrôle : planification, exécution, monitorage



Planificateur



Planification de trajectoire (*path*) vs Planification des déplacement (*motion*)

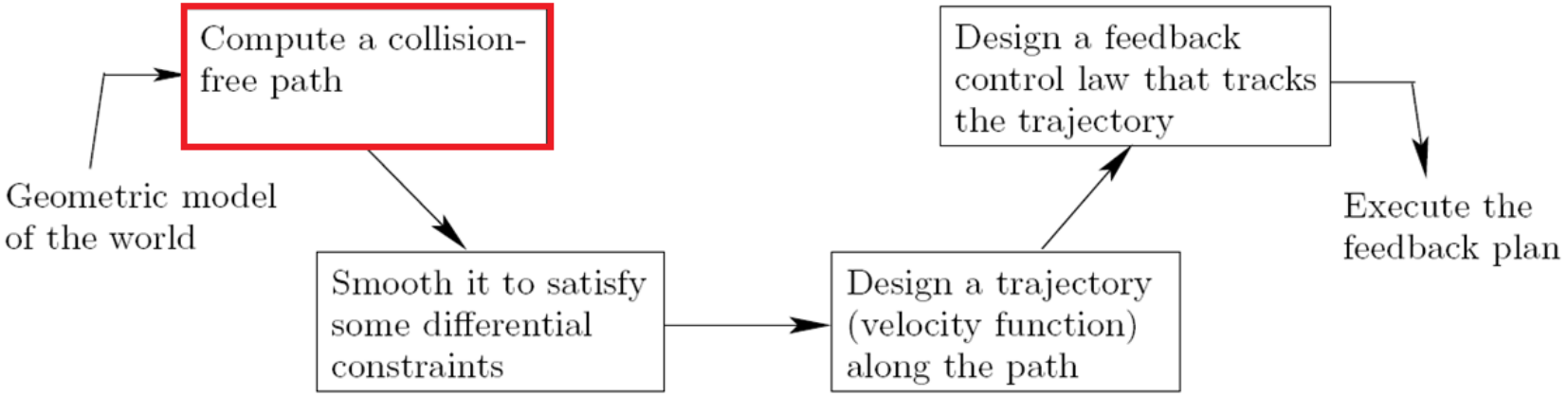


Figure 1.19: A refinement approach that has been used for decades in robotics.

Plan

- Énoncé du problème
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Nécessité des représentations géométriques

Les représentations géométriques sont nécessaires pour

- Modéliser le robot
- Modéliser les obstacles
- Détecter les collisions entre le robot et les obstacles

Pour ce cours

- Besoin juste des concepts et notions de base (infographie)
- Le projet utilisera la libraire OMPL. Elle utilise des outils de détection de collision : MoveIt!, FCL ou PQP

Notation

- $W = \mathbb{R}^2$ ou $\mathbb{R}^3$: espace de travail (*working space*) ou monde (*world*).
- $O \subseteq W$: obstacles (fixes)
- $A \subseteq W = A_1 \times \ldots \times A_n$: robots (mobiles)

Représentation géométrique

- Polygones / polyèdres
- Semi-algébriques
- Triangles
- Carte de coûts discrétisée
  - » Cas particulier: carte binaire (*bitmap*)

Polygones convexes

Un objet 2D qui est qui est un polygone convexe peut être représenté par une intersection de plans.

Convexe: un sous-ensemble X, si pour tout x_1, x_2 dans X et λ dans [0,1]: $\lambda x_1 + (1 - \lambda)x_2 \in X.$

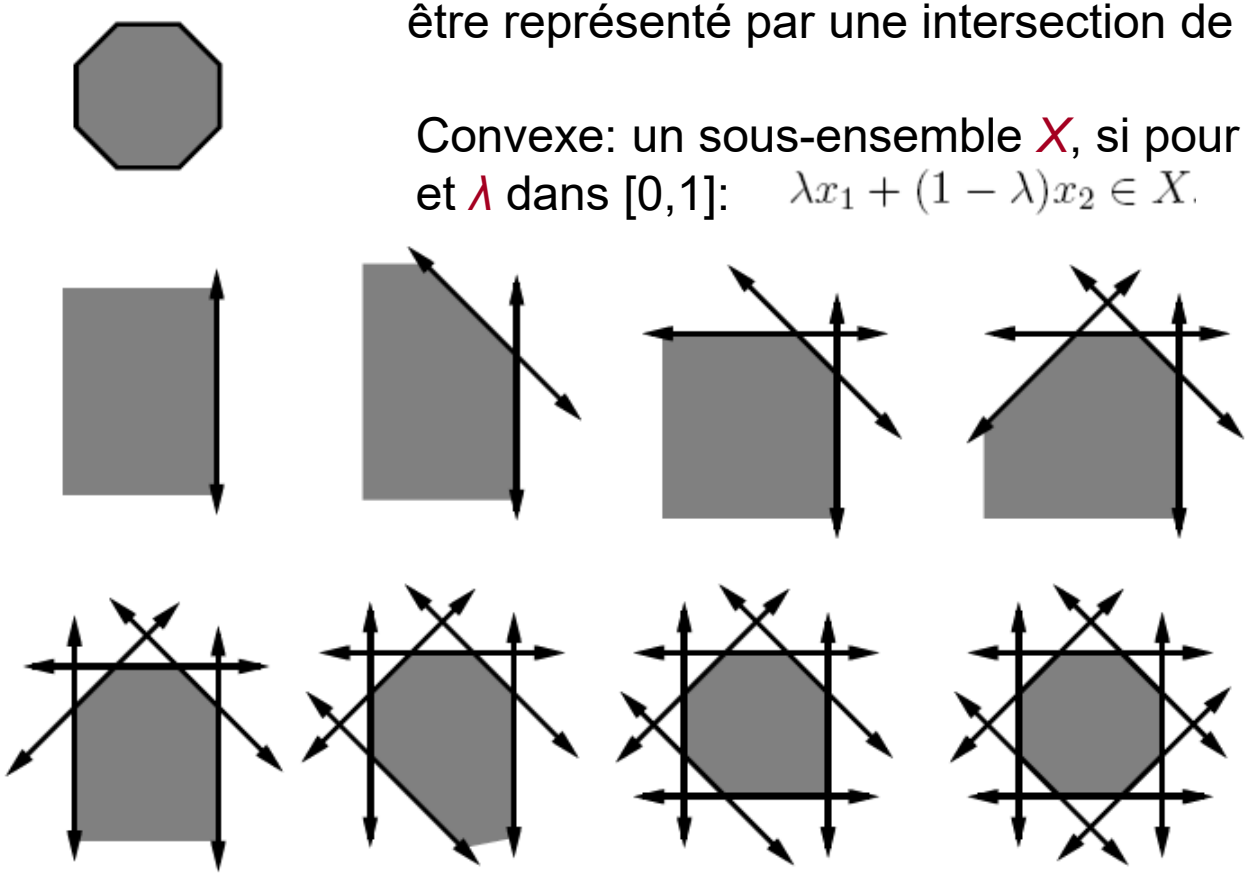


Figure 3.1: A convex polygonal region can be identified by the intersection of half-planes.

Polygones non convexes

- Un objet 2D non convexe, peut être représenté par une union d'objet convexes.

Polyèdres

- Pour des environnements 3D, les concepts précédents sont généralisés en remplaçant les polygones par des polyèdres et les demi-plans par des demi-espaces.

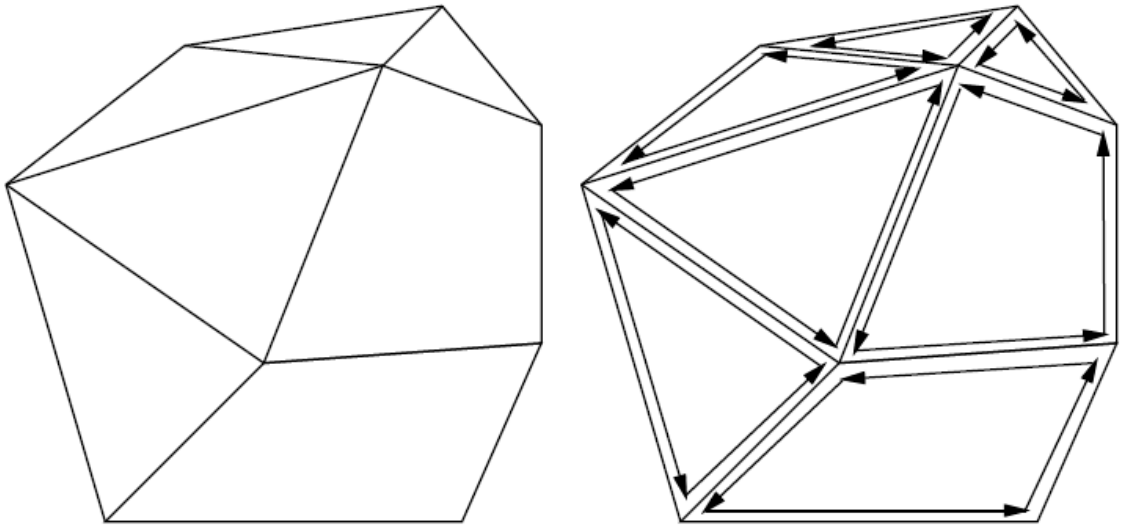


Figure 3.3: (a) A polyhedron can be described in terms of faces, edges, and vertices. (b) The edges of each face can be stored in a circular list that is traversed in counterclockwise order with respect to the outward normal vector of the face.

Triangles

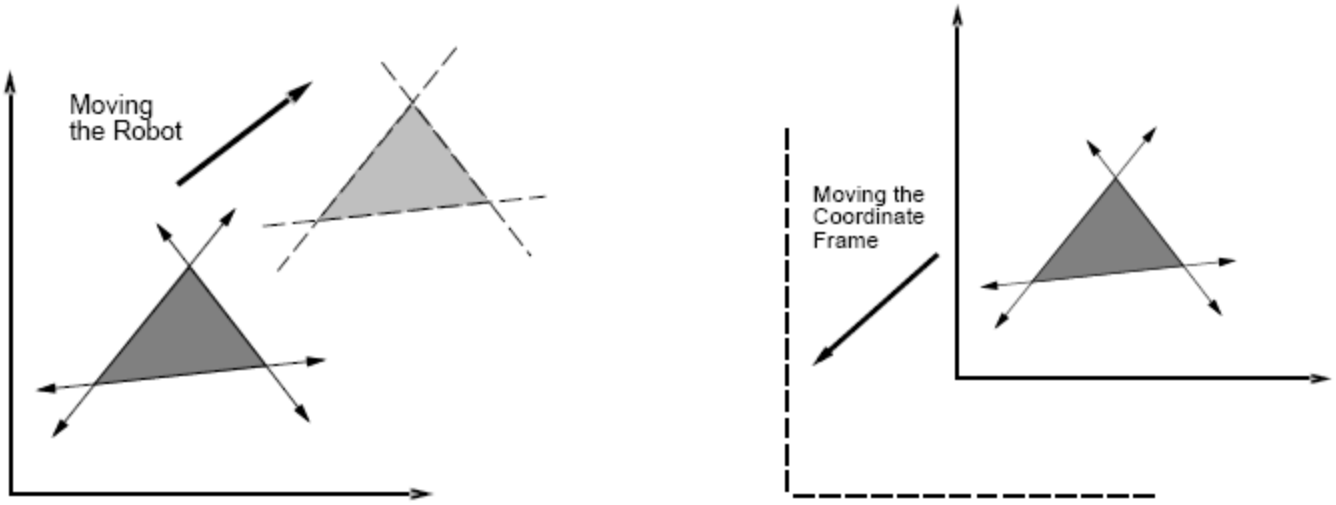
- Un autre modèle très populaire pour les représentations 3D est d'utiliser un ensemble de triangles, chaque triangle étant représenté par trois points (ses sommets).

Figure 3.6: Triangle strips and triangle fans can reduce the number of redundant points.

Transformations géométriques

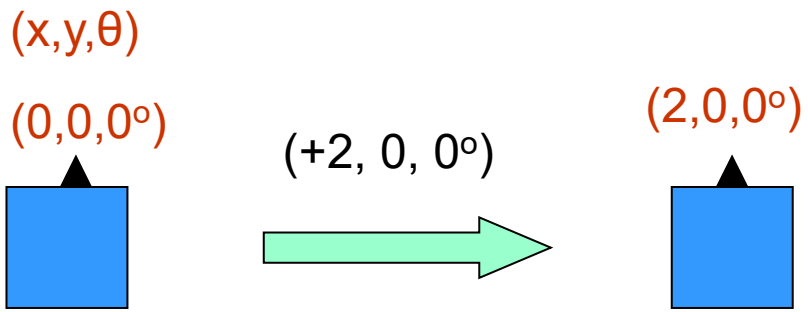
- Transformations entre cadres de référence
  - » Au moins un fixé sur le robot (*A*)
  - » Au moins un fixé sur l'espace de travail (*W*).
- Types de transformations
  - » Translations
  - » Rotations

Translations



(a) Translation of the robot (b) Translation of the frame

Figure 3.7: Every transformation has two interpretations.



Rotations

Rotation The robot, $\mathcal{A}$, can be *rotated* counterclockwise by some angle $\theta \in [0, 2\pi)$ by mapping every $(x, y) \in \mathcal{A}$ as

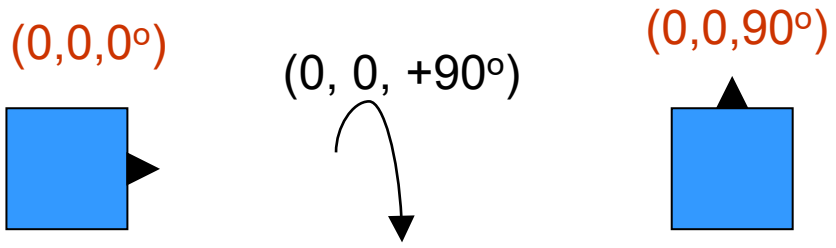
$$(x, y) \mapsto (x \cos\theta - y \sin\theta,\ x \sin\theta + y \cos\theta). \tag{3.30}$$

Using a 2×2 rotation matrix,

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

the transformation can be written as

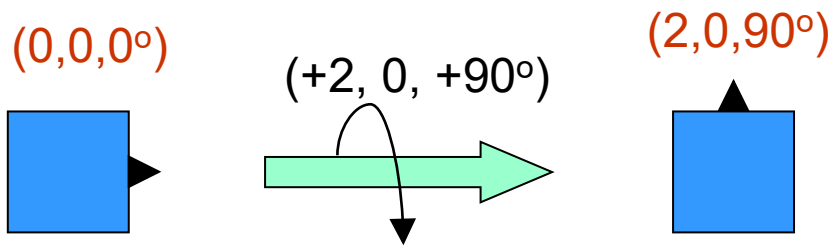
$$\begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{pmatrix} = R(\theta) \begin{pmatrix} x \\ y \end{pmatrix}.$$



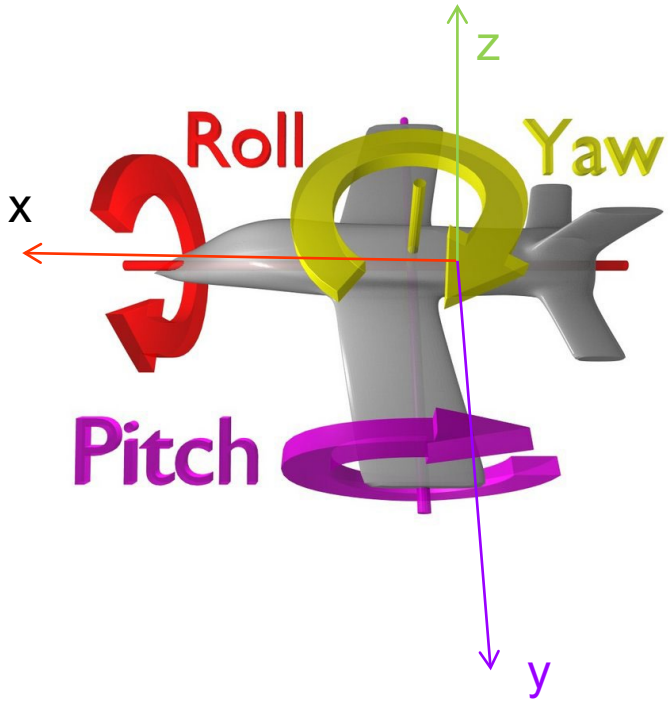
Rotations + Transformations

Combining translation and rotation Suppose a rotation by θ is performed, followed by a translation by x_t, y_t. This can be used to place the robot in any desired position and orientation. Note that translations and rotations do not commute! If the operations are applied successively, each $(x, y) \in \mathcal{A}$ is transformed to

$$\begin{pmatrix} x\cos\theta - y\sin\theta + x_t \\ x\sin\theta + y\cos\theta + y_t \end{pmatrix}. \tag{3.33}$$



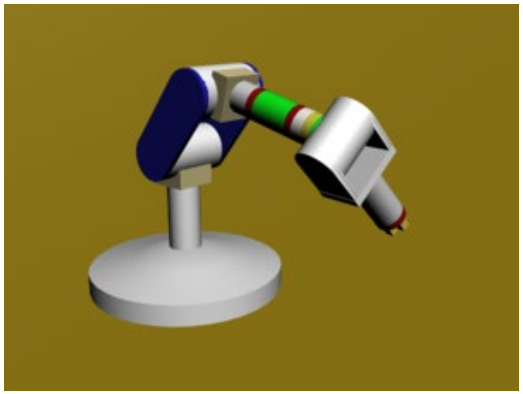
Yaw, Pitch and Roll

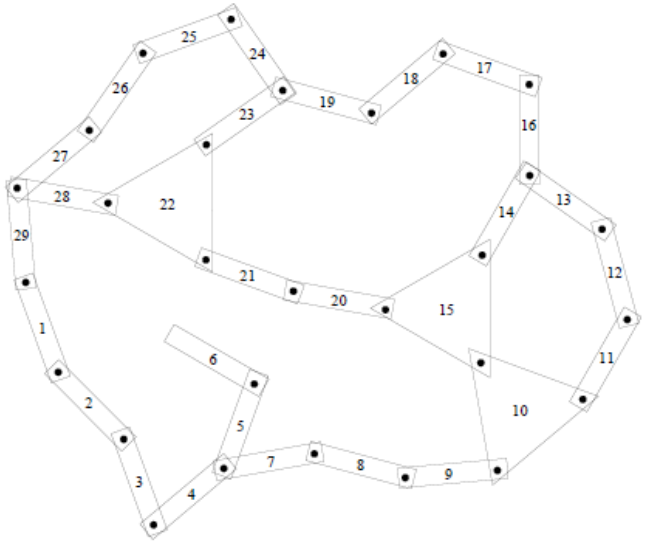


N’importe quelle rotation dans l’espace à trois dimension peut être représentée par une séquence de *yaw, pitch, roll*

Chaîne cinématique

- Ensemble de segments rigides articulés reliés par des joints (typiquement en translation ou en rotation)





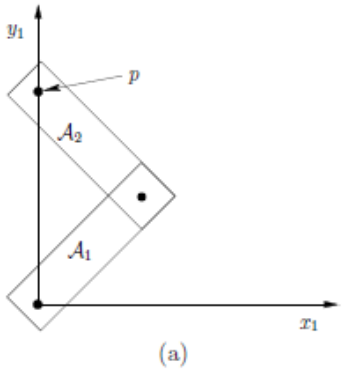


Figure 4.27: A complicated linkage that has 29 links, several loops, links with more than two bodies, and bodies with more than two links. Each integer i indicates link $\mathcal{A}_i$.

Transformations géométriques pour les chaînes cinématiques

- Les transformations géométriques de bases sont combinées pour obtenir des transformations d'un corps articulé: Voir Section 3.3 du livre.

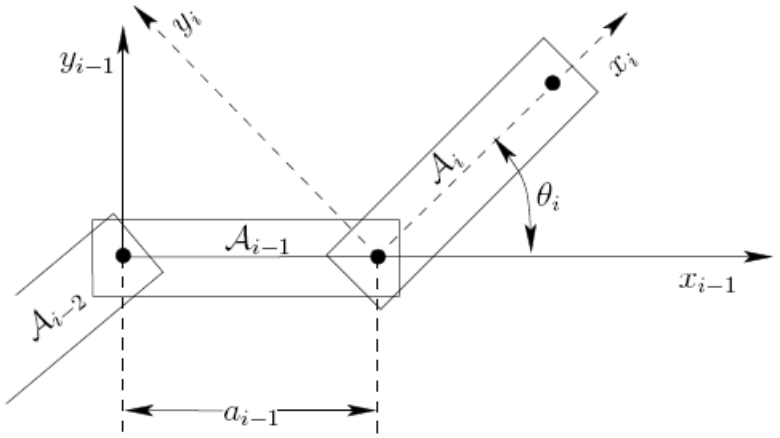
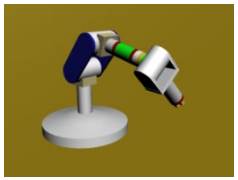
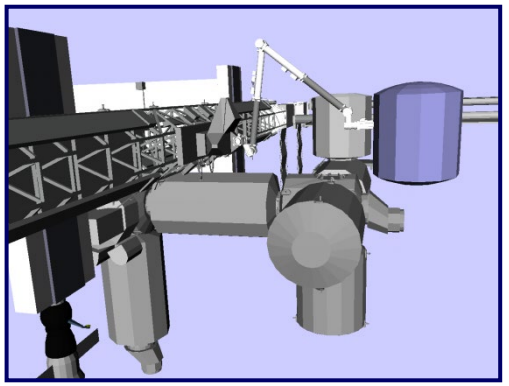
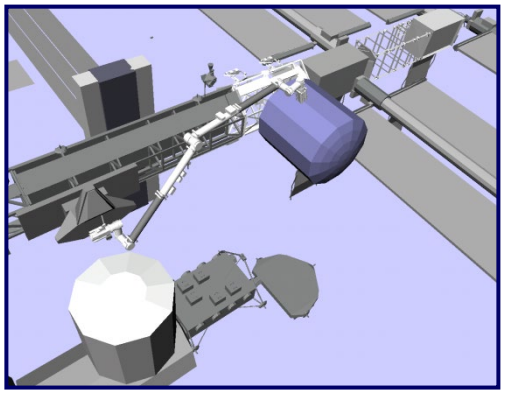


Figure 3.10: The body frame of each $\mathcal{A}_i$, for $1 < i < m$, is based on the joints that connect $\mathcal{A}_i$ to $\mathcal{A}_{i-1}$ and $\mathcal{A}_{i+1}$.



Cinématique inverse





- Déplacement pour les **chaînes cinématiques lorsqu'il n'y a pas d'obstacle.**

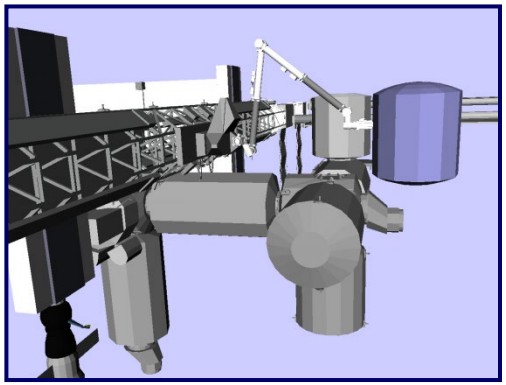
- Déterminer les rotations nécessaires pour amener l'effecteur terminal (*end effector*) dans une position donnée
  - » Problème algébrique (optimisation non-linéaire)
    - Utilisation de la matrice jacobienne: relate la vélocité des mouvements des joints par rapport aux différentes positions des effecteurs terminaux.
    - Ne prend pas en compte les obstacles externes.

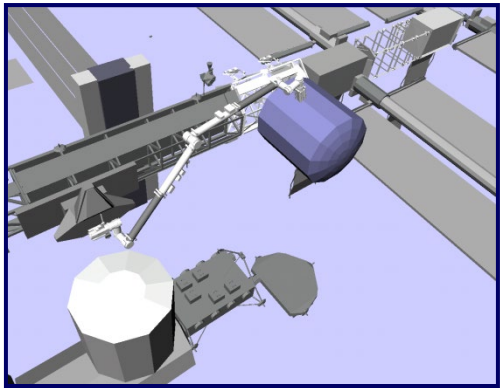
Plan

- Énoncé du problème
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Espace de configuration

- Ensemble des configurations possibles pour un robot.
  - » Le passage d'une configuration à une autre est le résultat d'une transformation.

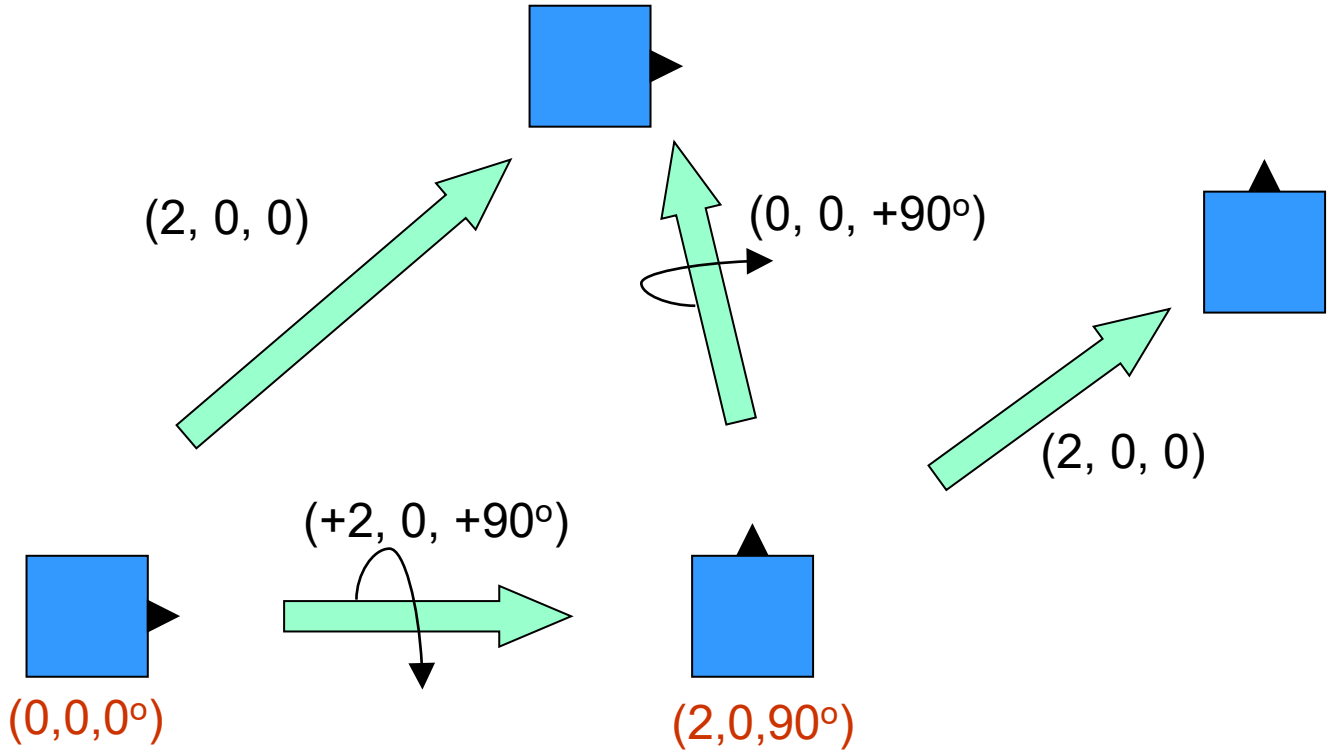




- Ainsi:
  - » Espace de configuration est synonyme de l'espace d'états.
  - » Transformation est synonyme d'action.

États/Actions

- États = configurations
- Actions = transformations



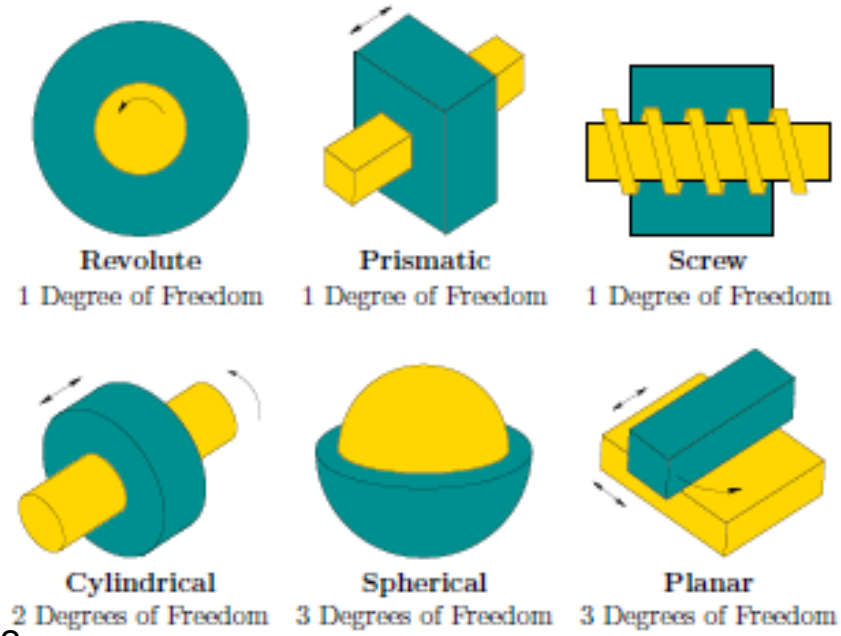
Notation

- C : C-space (espace de configuration)
- $C_{obs} \subseteq C$: configurations en collision avec des obstacles
- $C_{free} \subseteq C$ *:* configurations sans collision

Degrés de liberté

- Degrés de liberté = dimension de l'espace de configuration *C*
  - » Un corps rigide, capable de translation et de rotation selon tous les axes a six degré de liberté.

  - » Pour une chaîne cinématique, le nombre de degré de liberté dépend des joints.
    - Les joints limitent le nombre de degrés de libertés.



Plan

- Énoncé du problème
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Énoncé du problème (Rappel)

- *Planification de trajectoire*: *Calculer une trajectoire géométrique d'un solide (articulé ou non) sans collision avec des obstacles statiques.*

Entrée:

- Géométrie du robot et des obstacles
- Cinétique du robot (degrés de liberté)
- Configurations initiale et finale

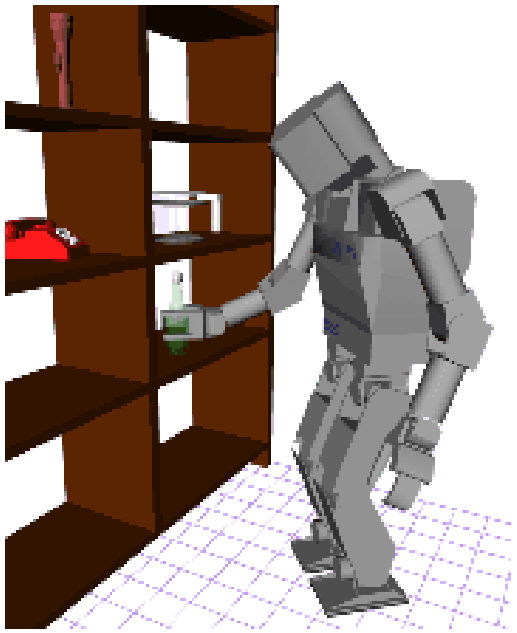
↓

Planificateur de trajectoires

↓

Sortie:

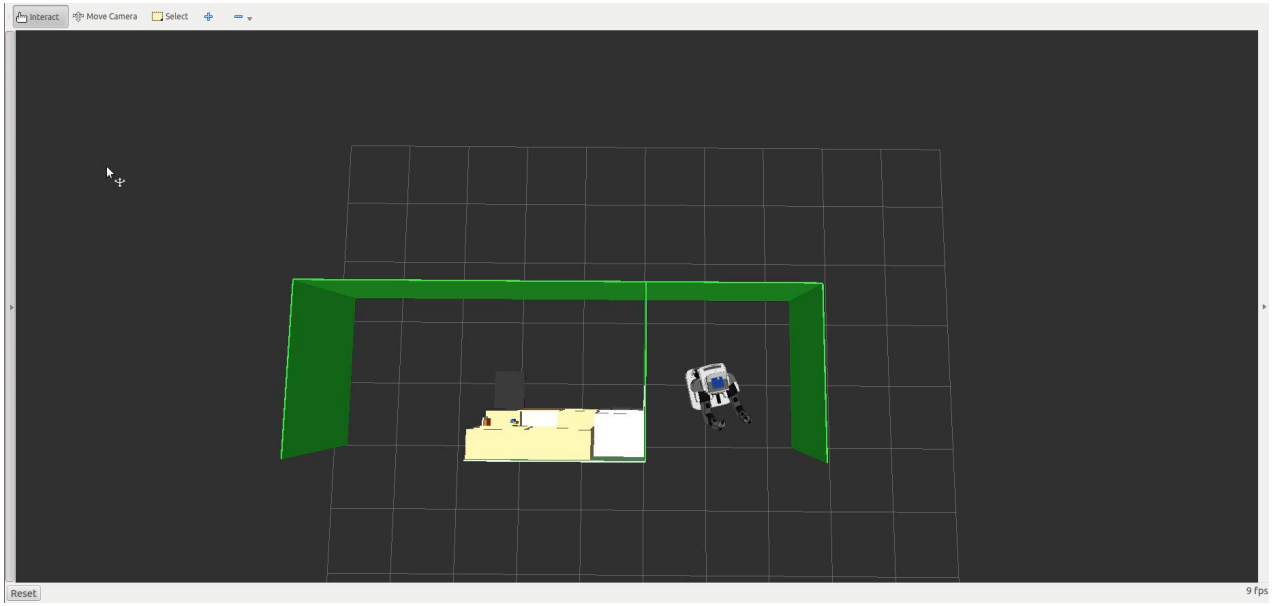
- Un chemin sans collision joignant la configuration initiale à la configuration finale



S. Kagami. U of Tokyo

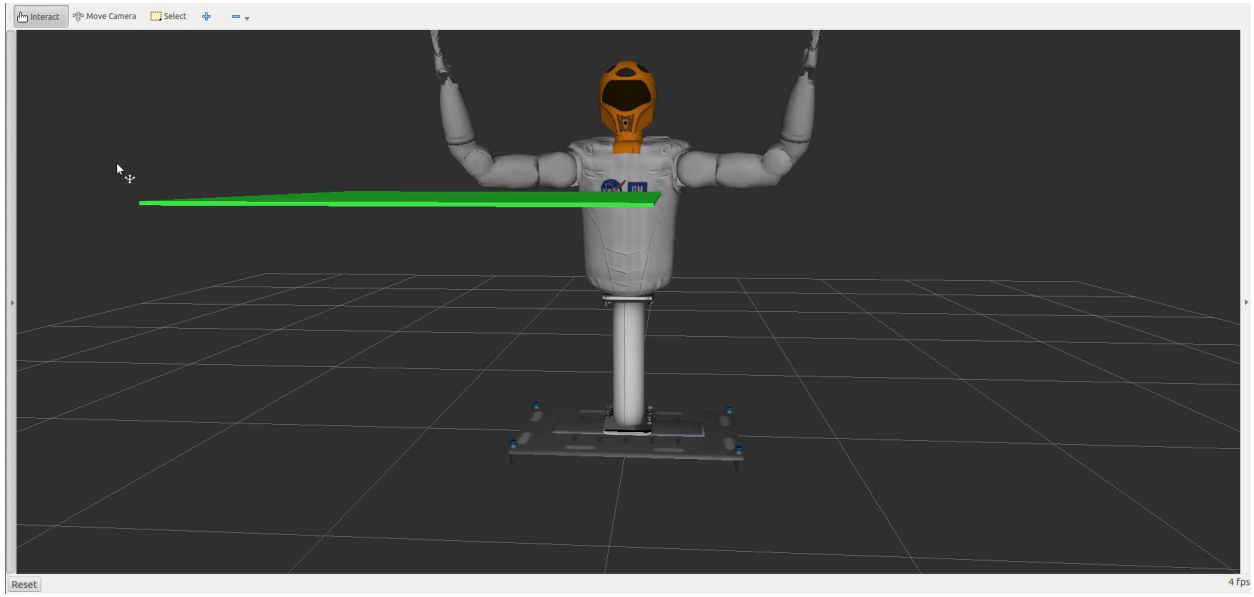
Hypothèses

- Transformations géométriques seulement (pas de contraintes différentielles pour l'instant)
- Obstacles statiques
- Environnement déterministe



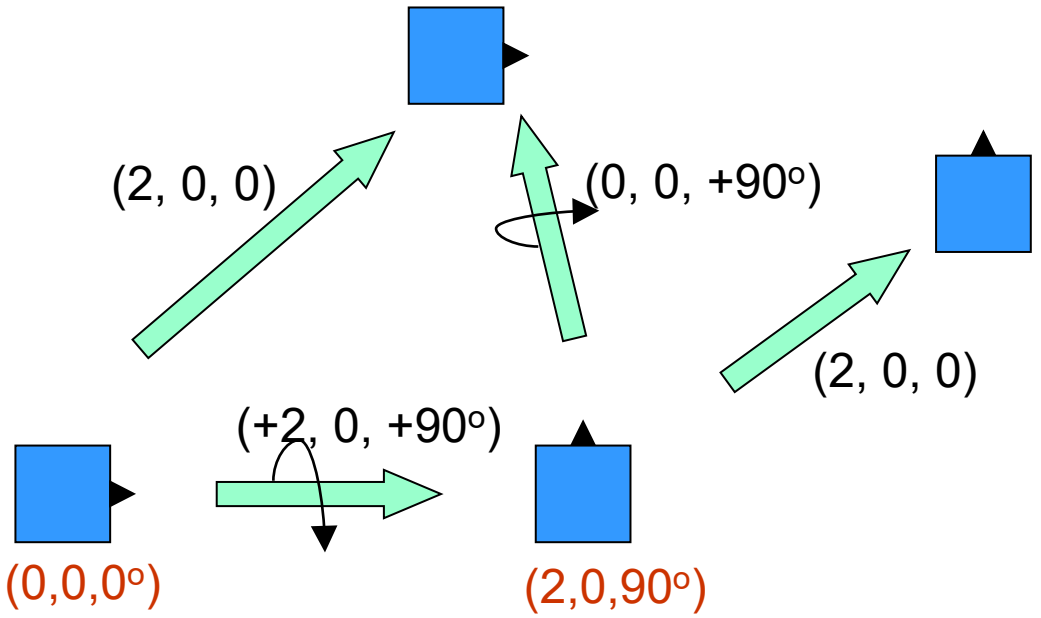
Hypothèses

- Transformations géométriques seulement (pas de contraintes différentielles pour l'instant)
- Obstacles statiques
- Environnement déterministe



Roadmaps

- Ces approches construisent des *roadmaps*, qui représentent des configurations atteignables par des transformations géométriques sans collisions
- États = configurations discrètes
- Actions = chemins



Sujets couverts

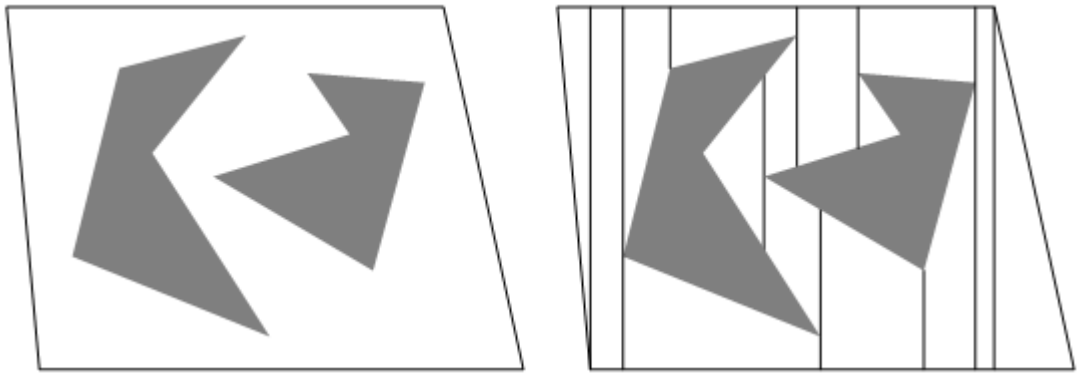
- Introduction
- Exemples d'applications
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Approches exactes

- Approches permettant de trouver des chemins sans approximations
  - » Par opposition aux approches par échantillonnage

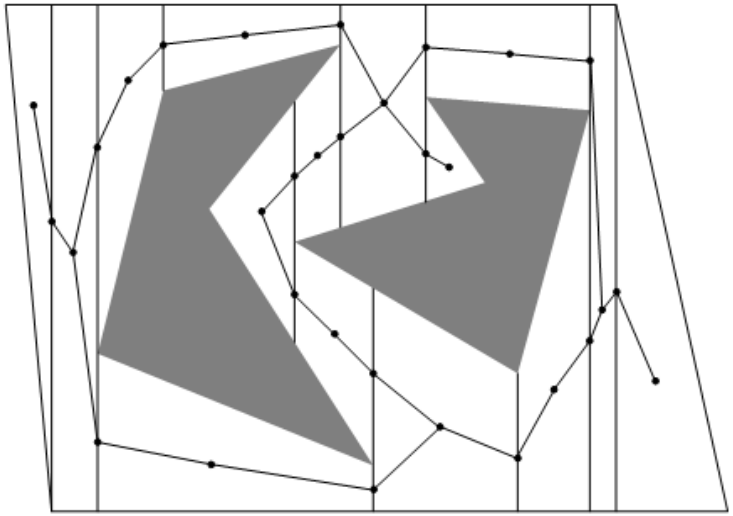
Décomposition verticale

- Construction du *roadmap* capturant complètement la topologie de l'espace de configuration



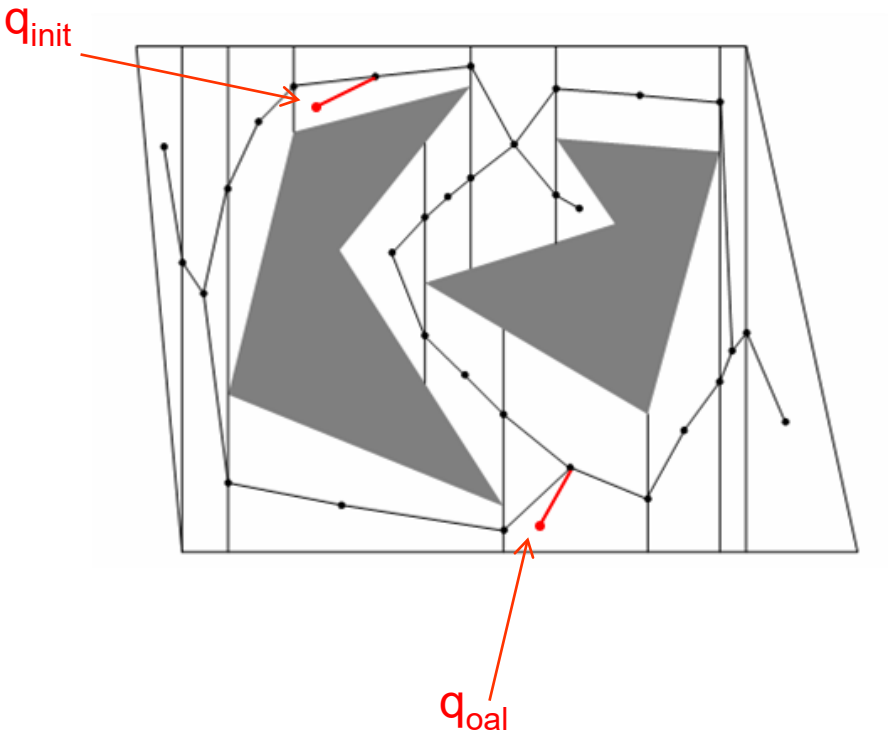
Décomposition verticale

- Construction du *roadmap* capturant complètement la topologie de l'espace de configuration



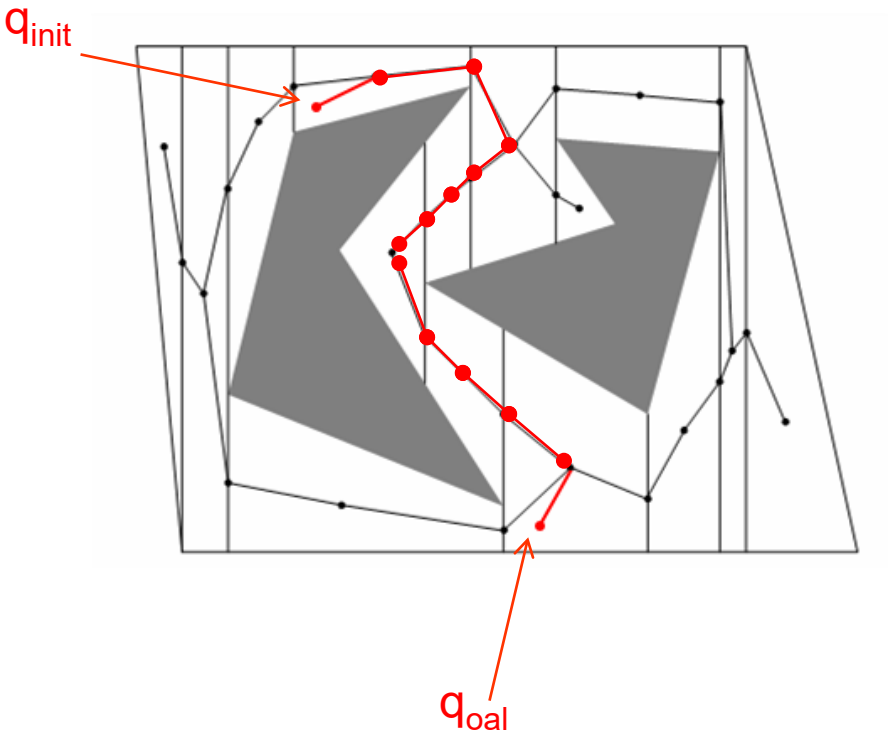
Décomposition verticale

- Réduit le problème de planification d'une trajectoire à la recherche d'un chemin sur le roadmap.



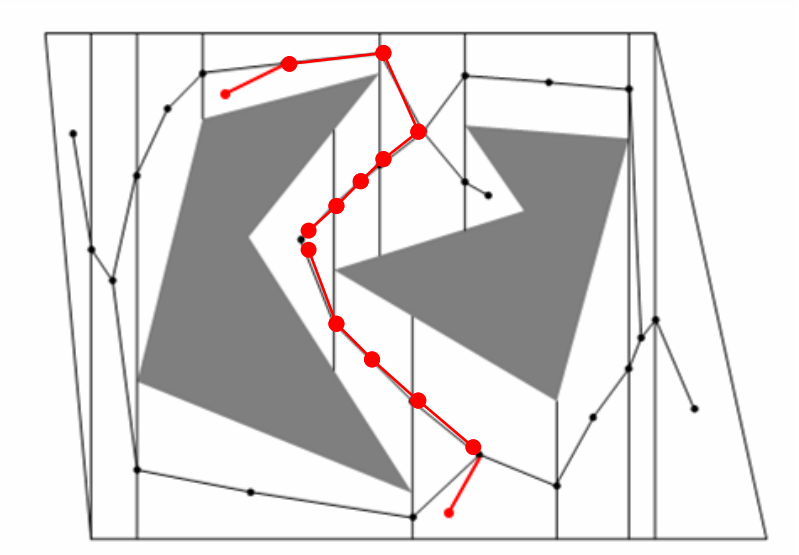
Décomposition verticale

- Réduit le problème de planification d'une trajectoire à la recherche d'un chemin sur le roadmap.



Approches exactes

- **Propriétés**:



1. Accessibilité: possible de connecter facilement n'importe quelle configuration viable (pas dans un obstacle) à la *roadmap*
2. Conserve la connectivité: si un chemin existe entre deux états, un chemin (potentiellement différent) qui passe par la *roadmap* existe aussi

Cadre générale de résolution du problème

Problème continu

(espace de configuration + contraintes)

↓

Discrétisation

(décomposition, échantillonnage**)**

↓

Recherche heuristique dans un graphe

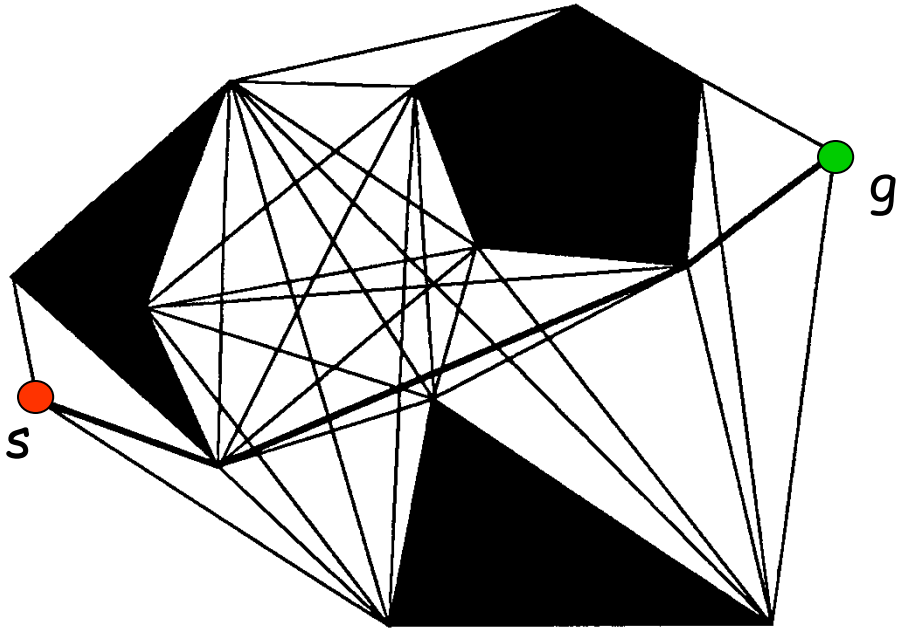
(A* ou similaire)

Algorithm (sketch)

1. Compute cell decomposition down to some resolution
2. Identify start and goal cells
3. Search for sequence of empty/mixed cells between start and goal cells (A*)
4. If no sequence, then exit with no path
5. If sequence of empty cells, then exit with solution
6. If resolution threshold achieved, then exit with failure
7. Decompose further the mixed cells
8. Return to 2

Approches avec un graphe de visibilité

- Introduit avec le robot Shakey à SRI dans les années 60.
- Efficace pour l'espace 2D



Algorithme simplifié

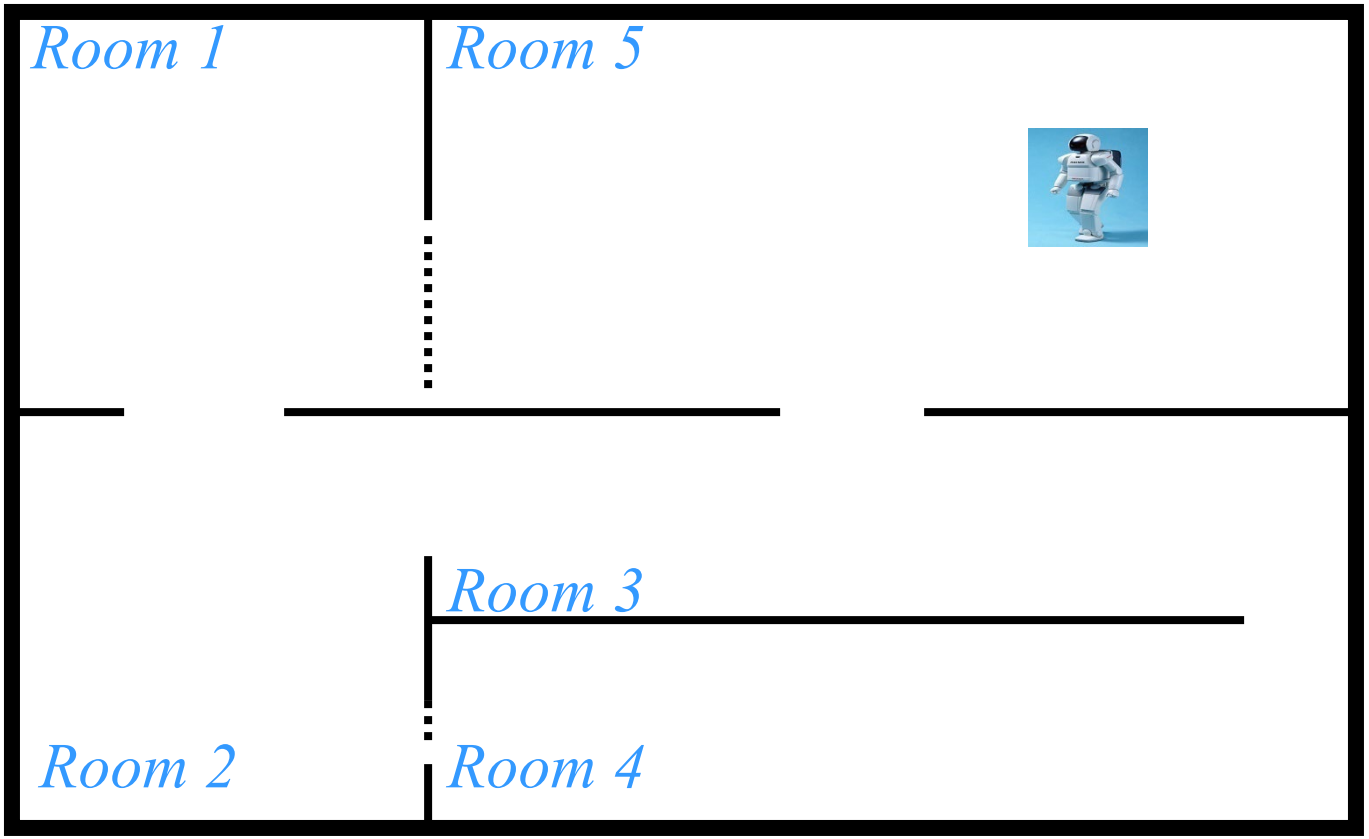
VG signifie « *Visibility Graph* » (Graphe de visibilité)

Algorithm

- Install all obstacles vertices in *VG*, plus the start and goal positions
- For every pair of nodes u, v in *VG*
- If segment(u,v) is an obstacle edge then
- insert (u,v) into *VG*
- else
- for every obstacle edge e
- if segment(u,v) does not intersects e
- insert (u,v) into *VG*
- Search *VG* using *A**

Complexité : $O(n^3)$, avec n = nombre de sommets des obstacles.
Il existe des versions améliorés $O(n)$.

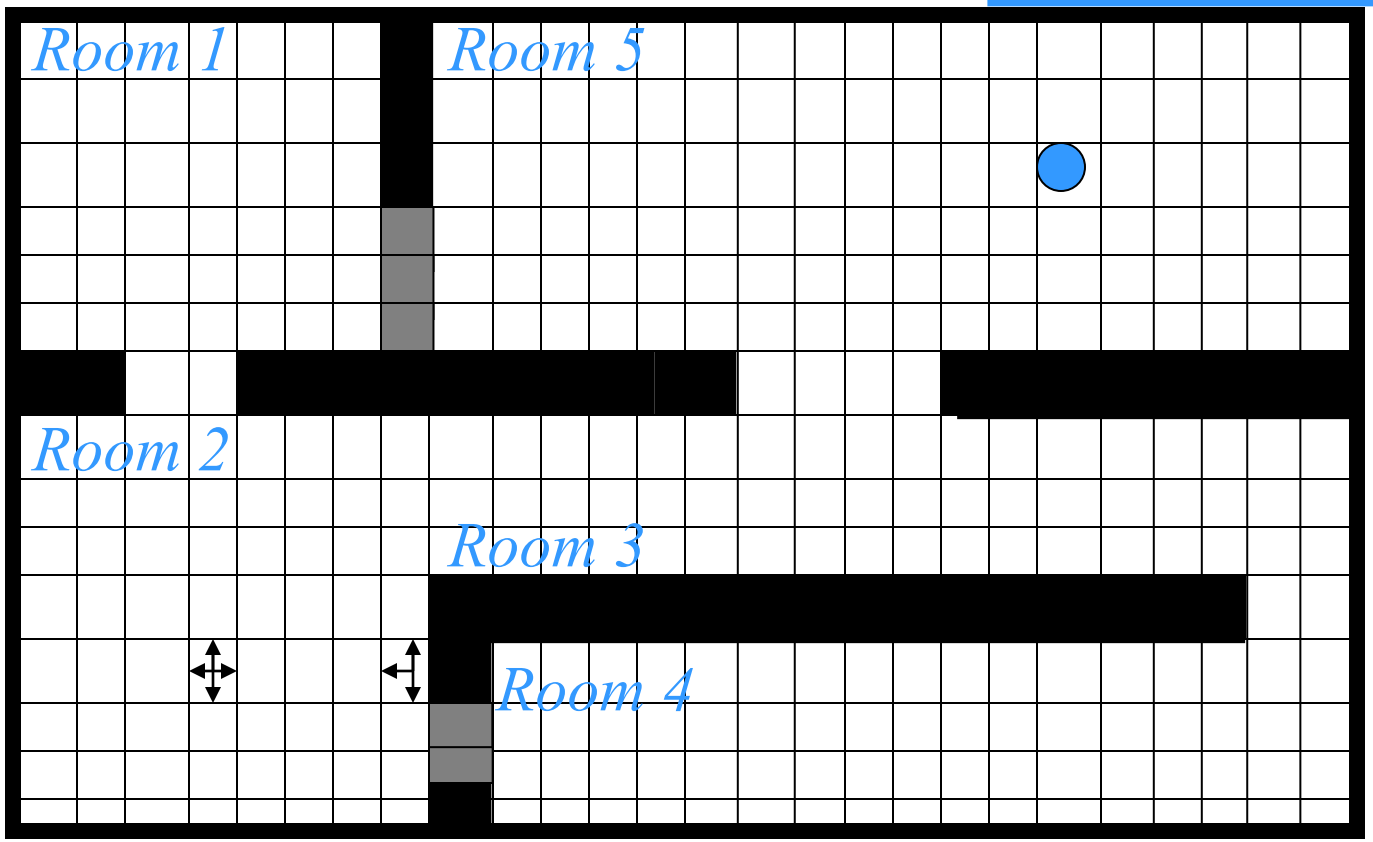
Décomposition en grille



Décomposition en grille

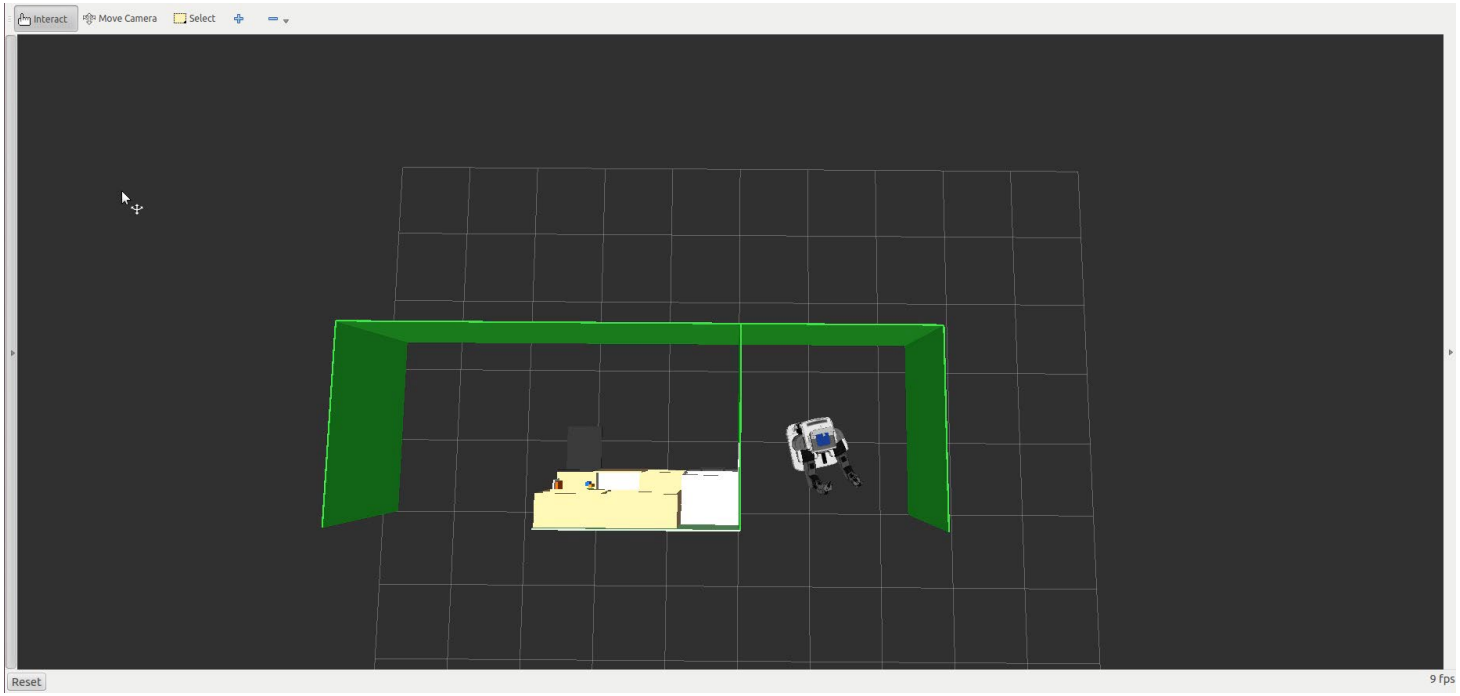
Décomposer la carte en grille (*occupancy grid*):
4-connected (illustré ici) ou 8-connected.

Transitions:
- Tourner à gauche ↰
- Tourner à droite ↱
- Avancer



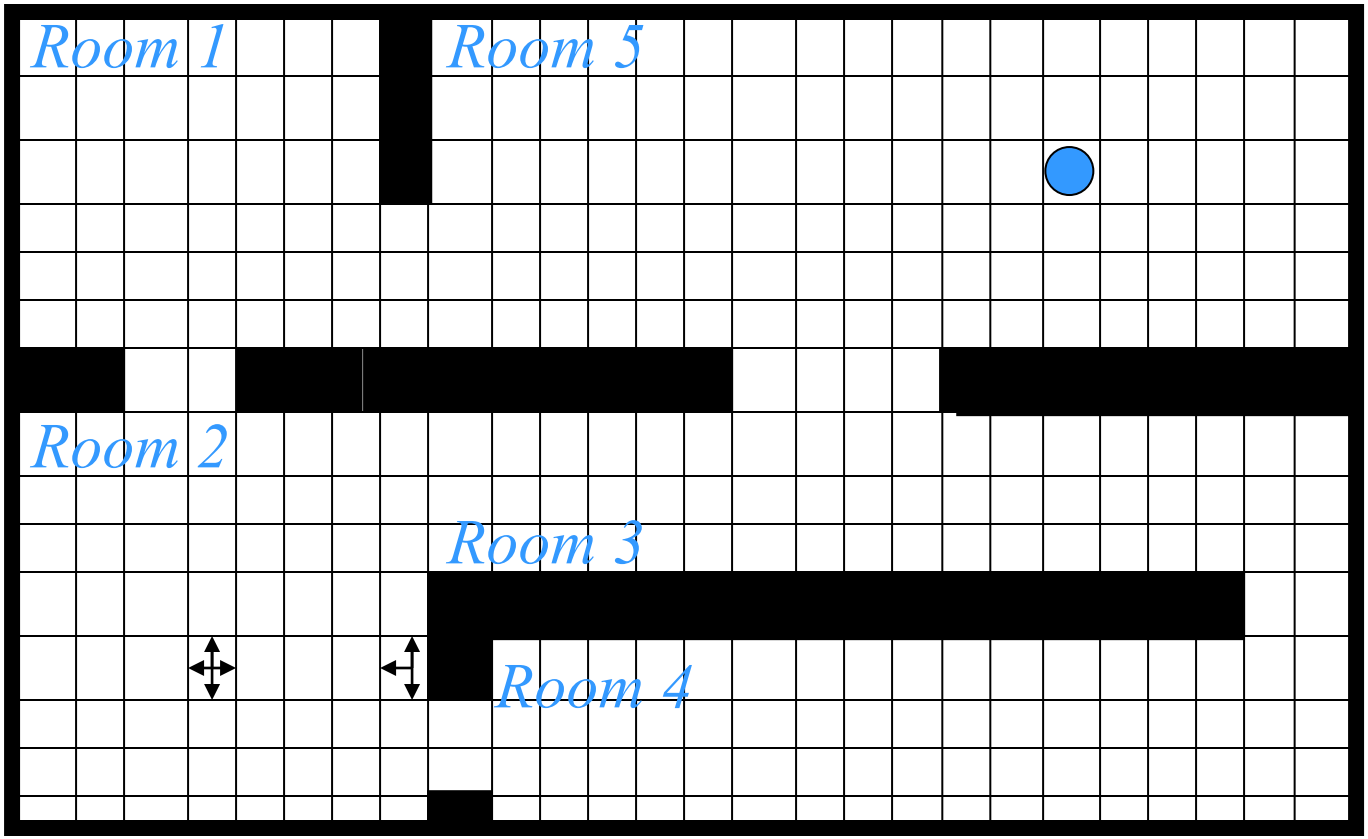
Décomposition en grille

- Fonctionne bien pour des robots mobiles non articulées. Par exemple, déplacer PR2 sur sa base d'un pièce à une autre. Fonctionne généralement moins bien pour les bras-articulés complexes.



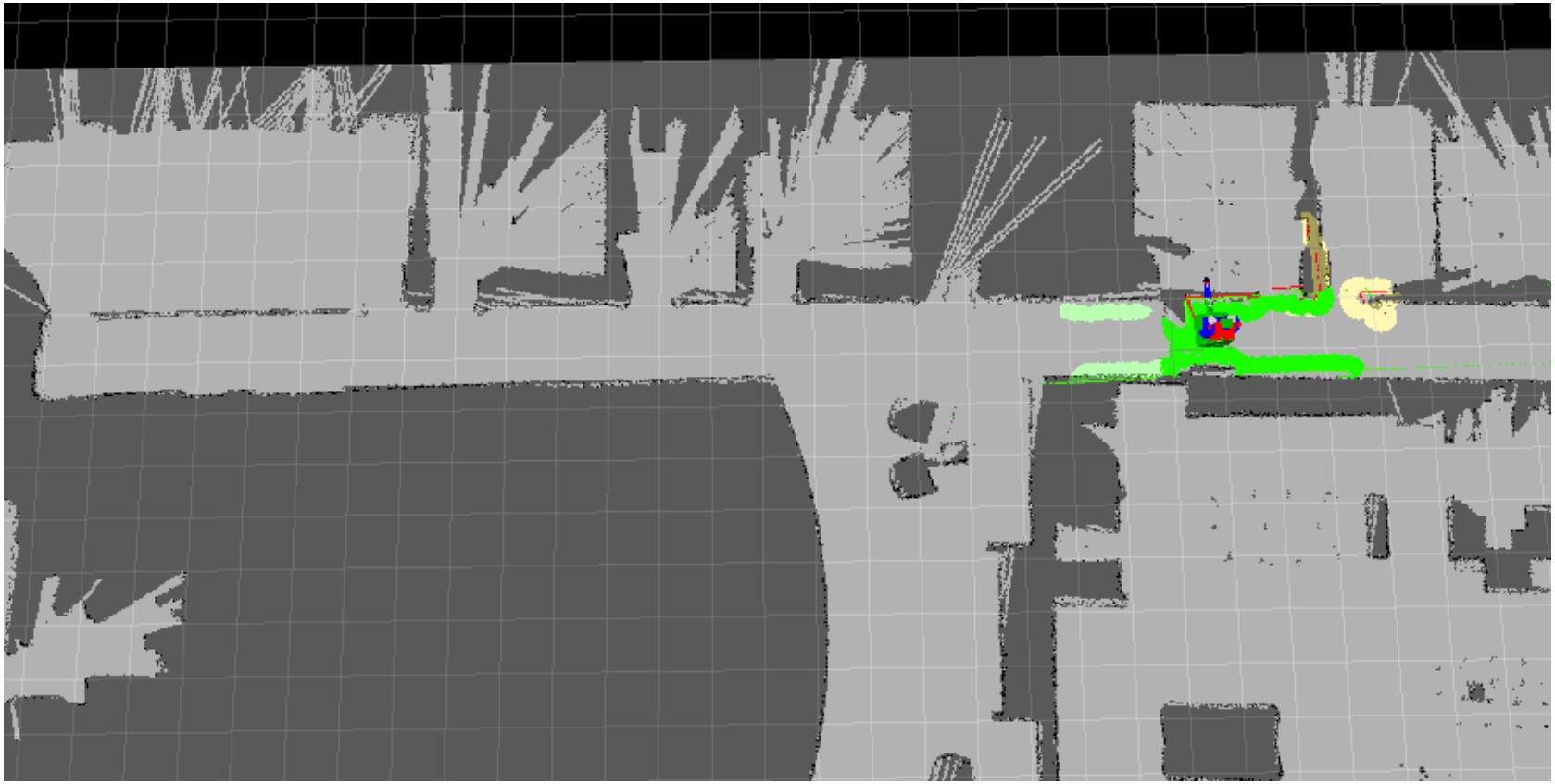
Cartes de coût

- Binaire (case occupée ou non)



Cartes de coût

- Coût basé sur les obstacles fixes et les capteurs (exemple dans ROS)



Décomposition en grille

1. Compute cell decomposition down to some resolution
2. Identify start and goal cells
3. Search for sequence of empty/mixed cells between start and goal cells
4. If no sequence, then exit with no path
5. If sequence of empty cells, then exit with solution
6. If resolution threshold achieved, then exit with failure
7. Decompose further the mixed cells
8. Return to 2

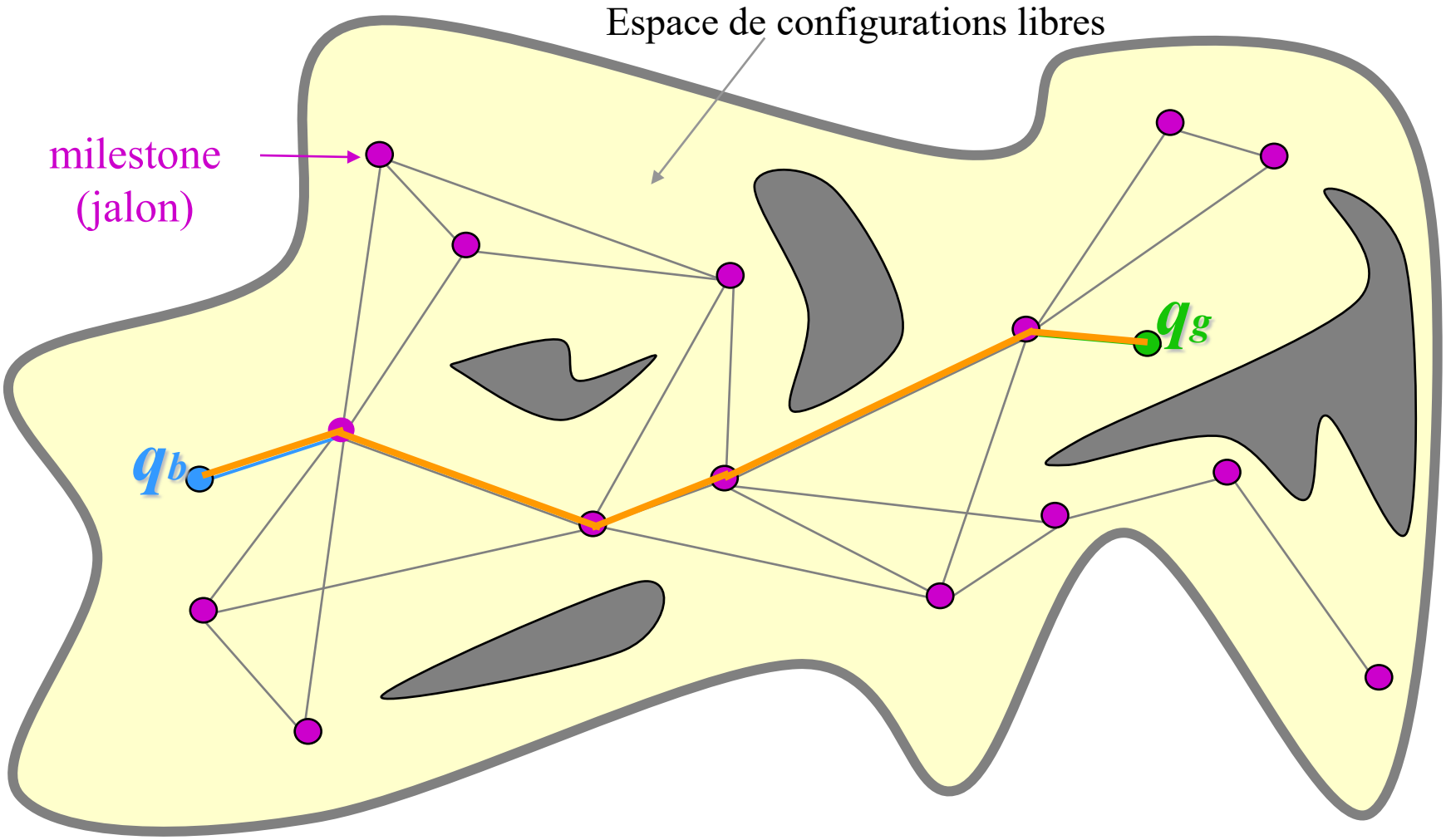
Approches exactes
Avantages / Inconvénients

- Avantages
  - » Efficaces si dimension basse, e.g. *point-robot* dans $C = R^2$
  - » Garanties théoriques (complétude, bornes sur le temps d'exécution...)
  - » Appropriées si plusieurs requêtes seront faites dans l'environnement (*multi-query*)
- Inconvénients
  - » Lent si nombre élevé de dimensions
  - » Implémentation difficile
  - » Représentation algébrique des obstacles

Plan

- Énoncé du problème
- Cadre de résolution général
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Approche *roadmap* probabiliste (*PRM - probabilistic roadmap*)



[Kavraki, Svetska, Latombe, Overmars, 95]

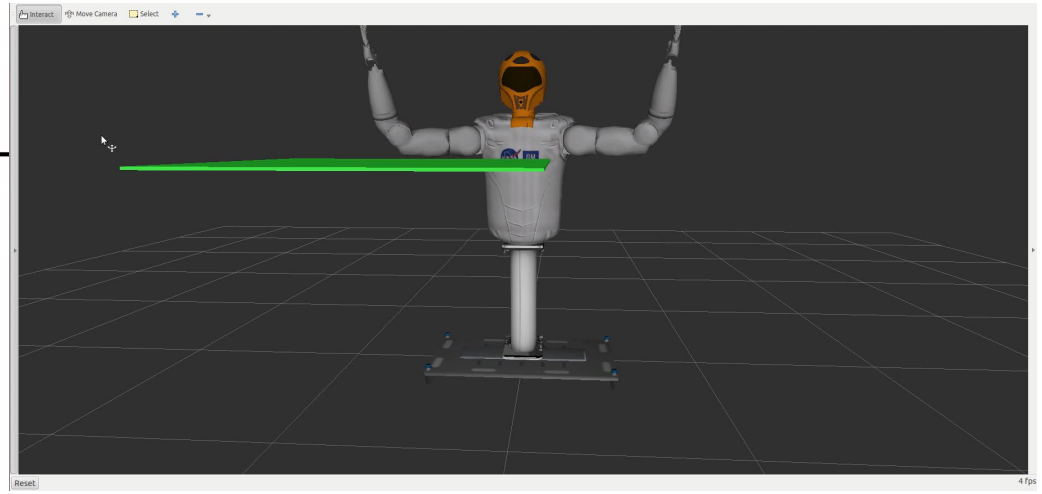
Rapidly Exploring Dense Tree (RDT)

[LaValle, Section 5]

- RRT (*Rapidly Exploring Random Tree*): cas particulier de RDT avec séquence aléatoire.

RDT(q_0)

1 $\mathcal{G}$.init(q_0);
2 **for** $i = 1$ **to** k **do**
3 $q_n \leftarrow$ NEAREST($S, \alpha(i)$);
4 $q_s \leftarrow$ STOPPING-CONFIGURATION($q_n, \alpha(i)$);
5 **if** $q_s \neq q_n$ **then**
6 $\mathcal{G}$.add_vertex(q_s);
7 $\mathcal{G}$.add_edge(q_n, q_s);



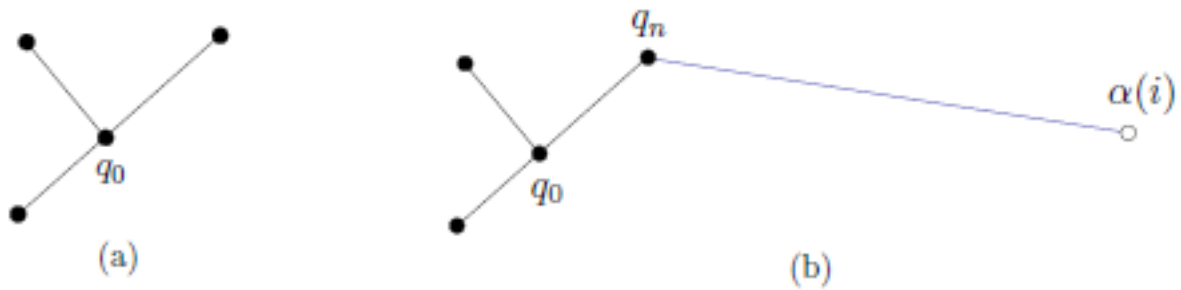
Rapidly Exploring Dense Tree (RDT)

[LaValle, Section 5]

- RRT (*Rapidly Exploring Random Tree*): cas particulier de RDT avec séquence aléatoire.

RDT(q_0)

1 $\mathcal{G}$.init(q_0);
2 **for** $i = 1$ **to** k **do**
3 $q_n \leftarrow$ NEAREST($S, \alpha(i)$);
4 $q_s \leftarrow$ STOPPING-CONFIGURATION($q_n, \alpha(i)$);
5 **if** $q_s \neq q_n$ **then**
6 $\mathcal{G}$.add_vertex(q_s);
7 $\mathcal{G}$.add_edge(q_n, q_s);



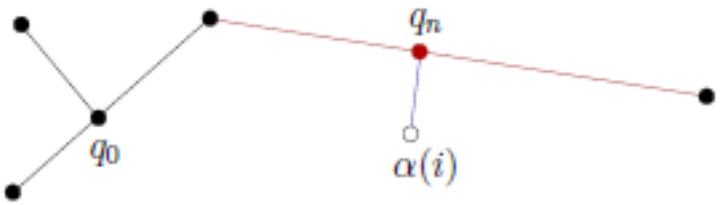
Rapidly Exploring Dense Tree (RDT)

[LaValle, Section 5]

- RRT (*Rapidly Exploring Random Tree*): cas particulier de RDT avec séquence aléatoire.

RDT(q_0)

1 $\mathcal{G}$.init(q_0);
2 **for** $i = 1$ **to** k **do**
3 $q_n \leftarrow$ NEAREST($S, \alpha(i)$);
4 $q_s \leftarrow$ STOPPING-CONFIGURATION($q_n, \alpha(i)$);
5 **if** $q_s \neq q_n$ **then**
6 $\mathcal{G}$.add_vertex(q_s);
7 $\mathcal{G}$.add_edge(q_n, q_s);

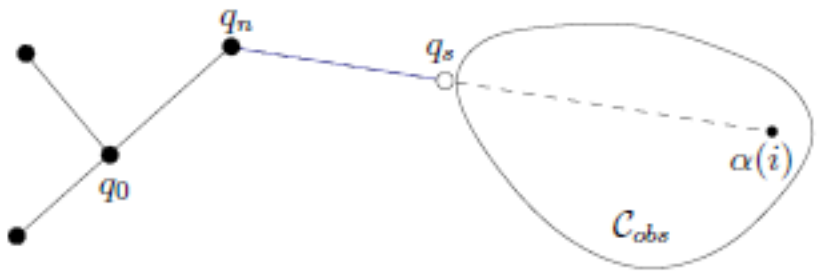


Rapidly Exploring Dense Tree (RDT)

[LaValle, Section 5]

- RRT (*Rapidly Exploring Random Tree*): cas particulier de RDT avec séquence aléatoire.

RDT(q_0)
1 $\mathcal{G}$.init(q_0);
2 **for** $i = 1$ **to** k **do**
3 $q_n \leftarrow$ NEAREST($S, \alpha(i)$);
4 $q_s \leftarrow$ STOPPING-CONFIGURATION($q_n, \alpha(i)$);
5 **if** $q_s \neq q_n$ **then**
6 $\mathcal{G}$.add_vertex(q_s);
7 $\mathcal{G}$.add_edge(q_n, q_s);

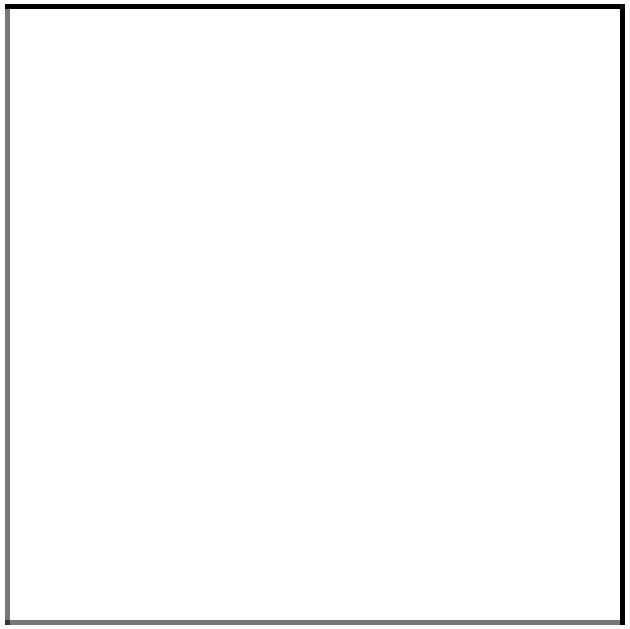


Rapidly Exploring Dense Tree (RDT)

[LaValle, Section 5]

- RRT (*Rapidly Exploring Random Tree*): cas particulier de RDT avec séquence aléatoire.

```
RDT(q_0)
1  G.init(q_0);
2  for i = 1 to k do
3      q_n ← NEAREST(S, α(i));
4      q_s ← STOPPING-CONFIGURATION(q_n, α(i));
5      if q_s ≠ q_n then
6          G.add_vertex(q_s);
7          G.add_edge(q_n, q_s);
```



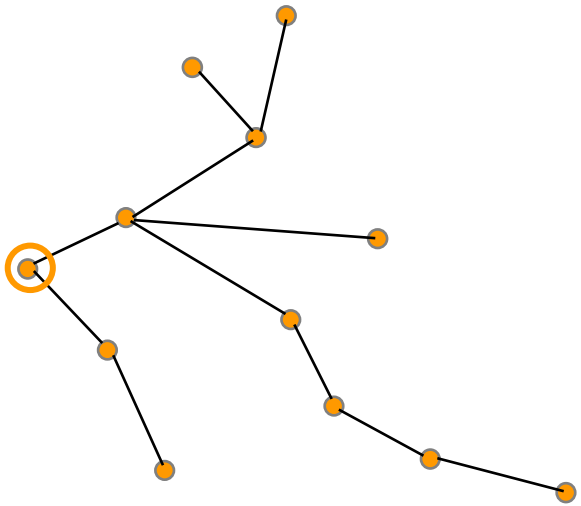
RDT - Bidirectionnel [LaValle, Section 5]

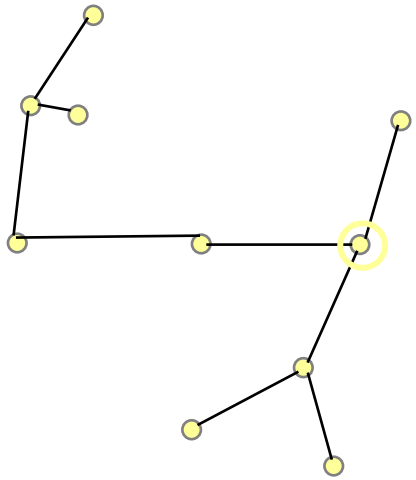
RDT_BALANCED_BIDIRECTIONAL(q_I, q_G)

1 T_a.init(q_I); T_b.init(q_G);
2 **for** $i = 1$ **to** K **do**
3 $q_n \leftarrow$ NEAREST($S_a, \alpha(i)$);
4 $q_s \leftarrow$ STOPPING-CONFIGURATION(q_n,$\alpha(i)$);
5 **if** $q_s \neq q_n$ **then**
6 T_a.add_vertex(q_s);
7 T_a.add_edge(q_n, q_s);
8 $q'_n \leftarrow$ NEAREST(S_b, q_s);
9 $q'_s \leftarrow$ STOPPING-CONFIGURATION(q'_n,q_s);
10 **if** $q'_s \neq q'_n$ **then**
11 T_b.add_vertex(q'_s);
12 T_b.add_edge(q'_n, q'_s);
13 **if** $q'_s = q_s$ **then return** SOLUTION;
14 **if** $|T_b| > |T_a|$ **then** SWAP(T_a, T_b);
15 **return** FAILURE

RDT Bidirectionnel

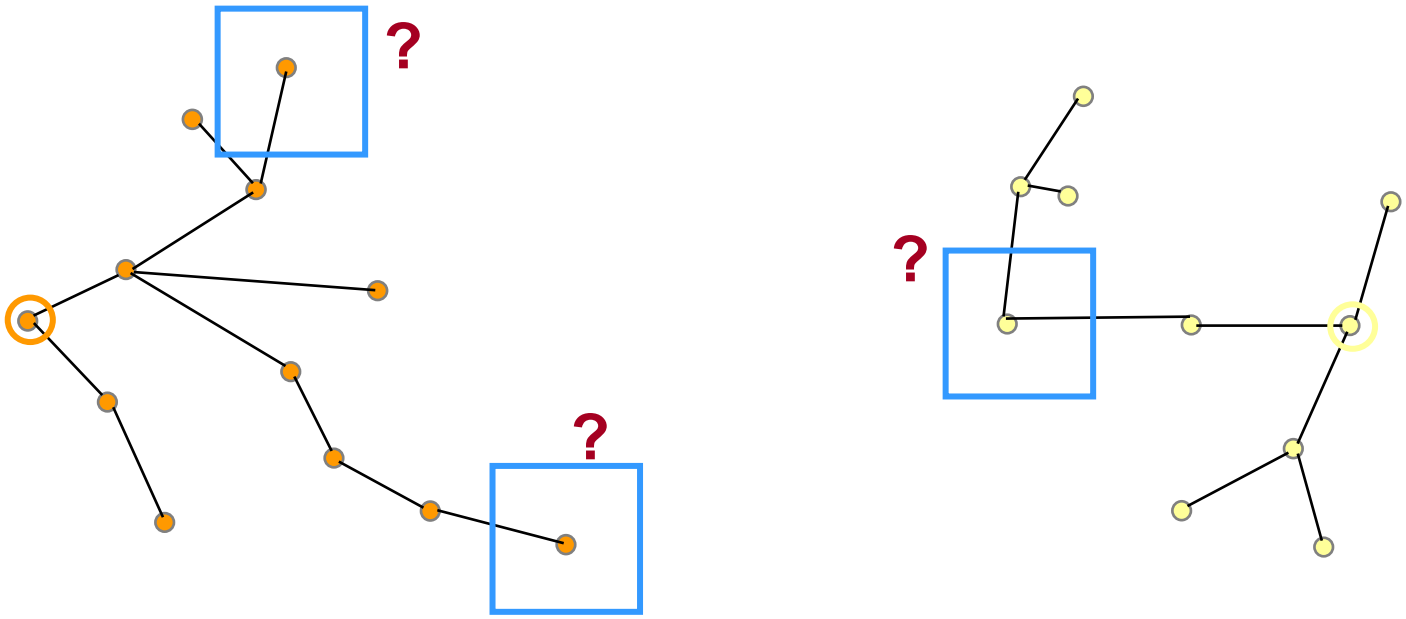
RDT Bidirectionnel





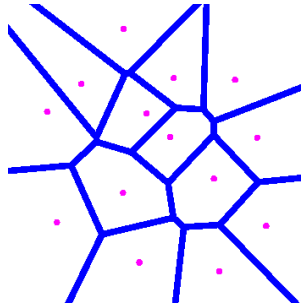
On souhaite faire grandir l'arbre

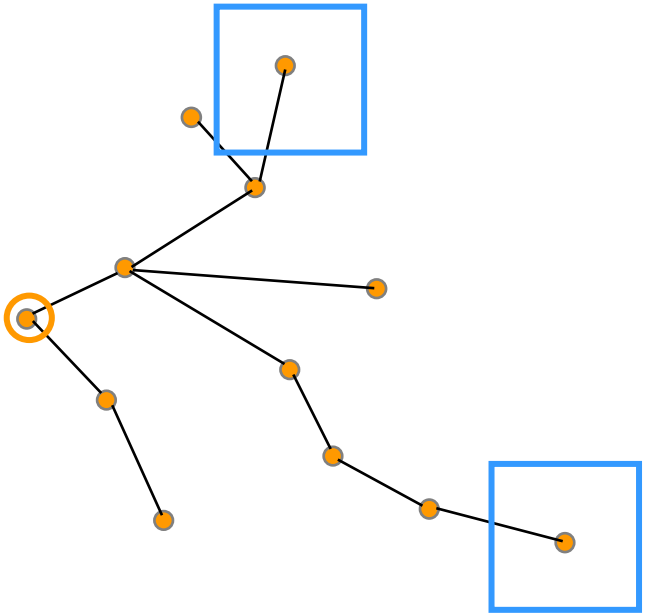
RDT Bidirectionnel

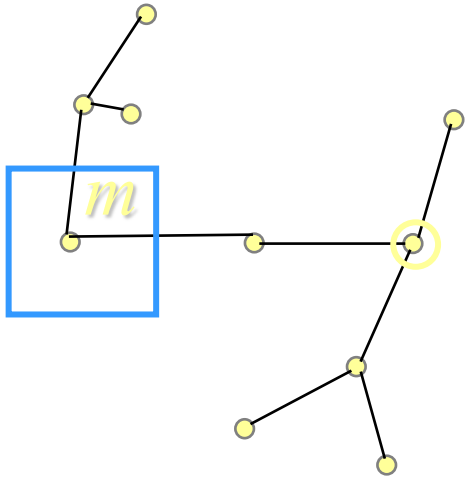


On souhaite faire grandir l'arbre; il faut trouver un noeud à étendre

RDT Bidirectionnel

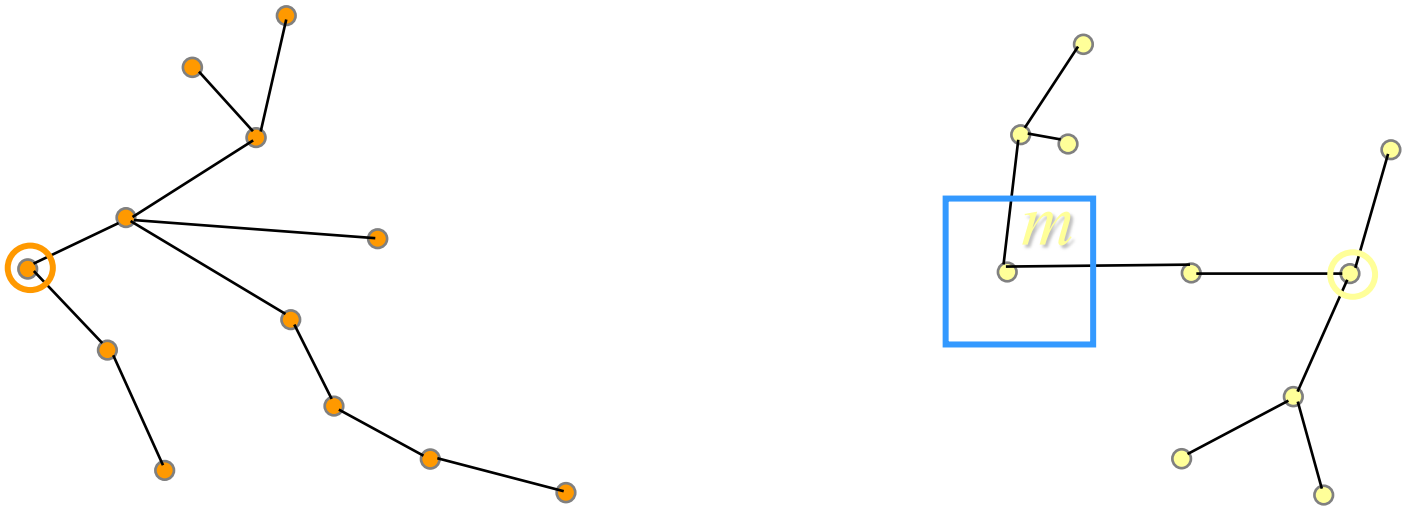






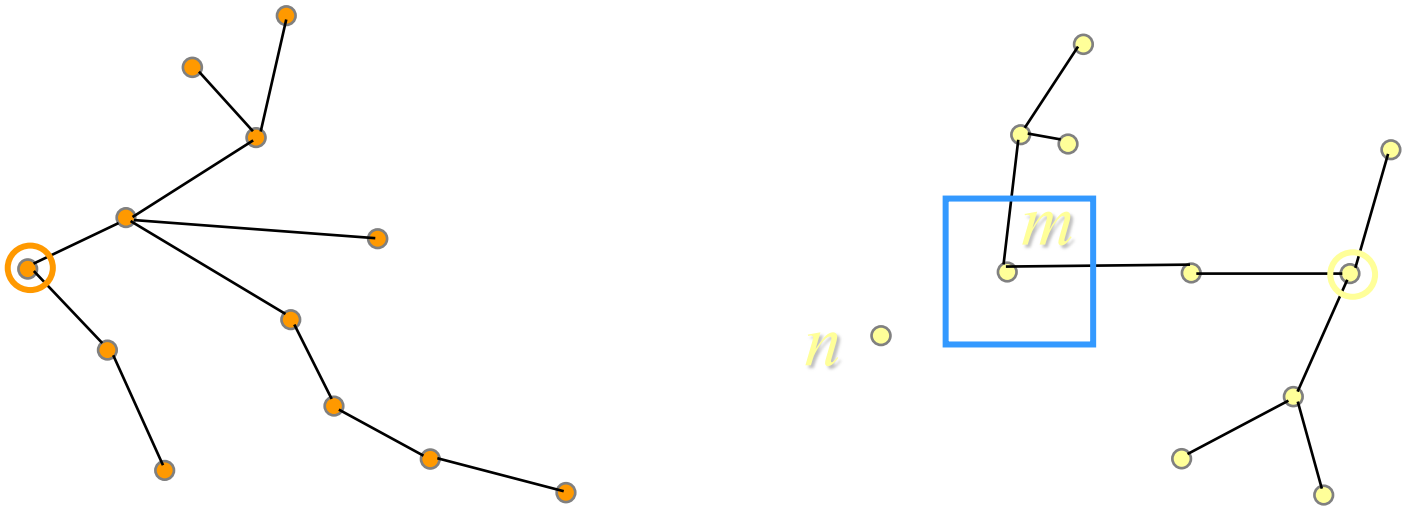
Chaque noeud a une probabilité d'être étendu inversement proportionnelle à la densité de son voisinage.

RDT Bidirectionnel



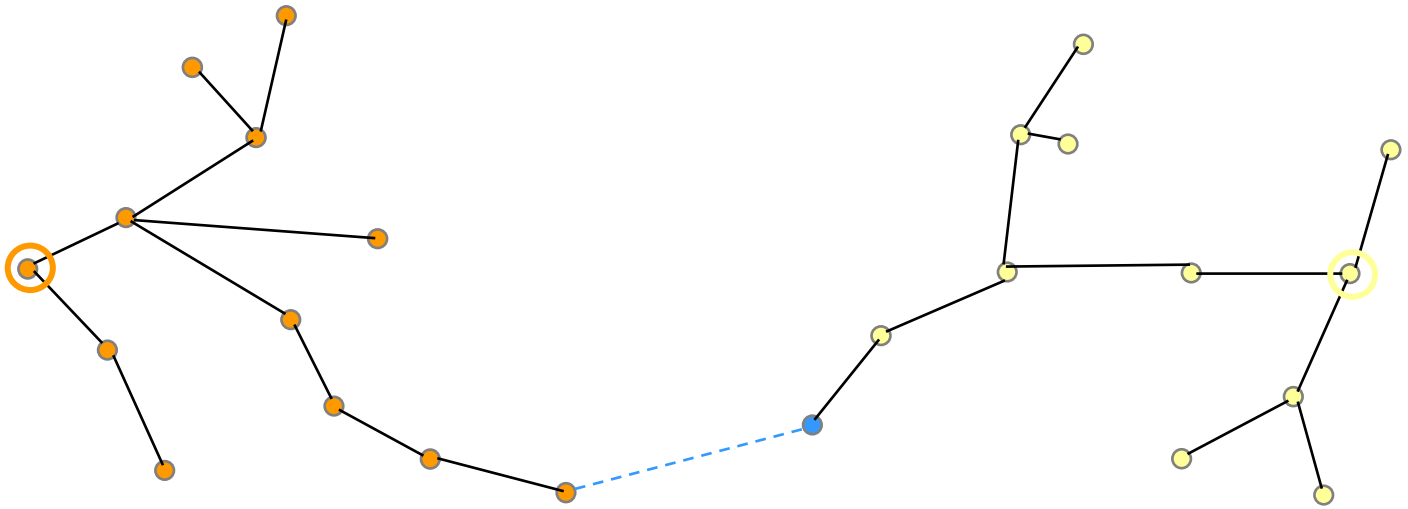
Supposons que n est choisi

RDT Bidirectionnel

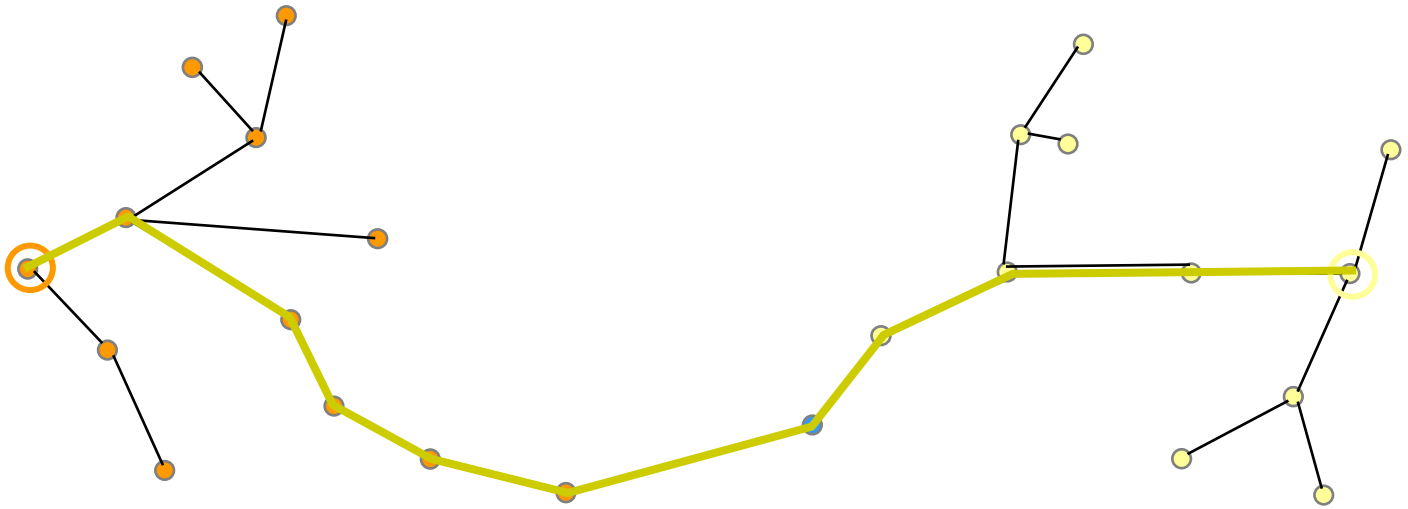


Un nouveau *n* est généré dans un voisinage proche de *m*

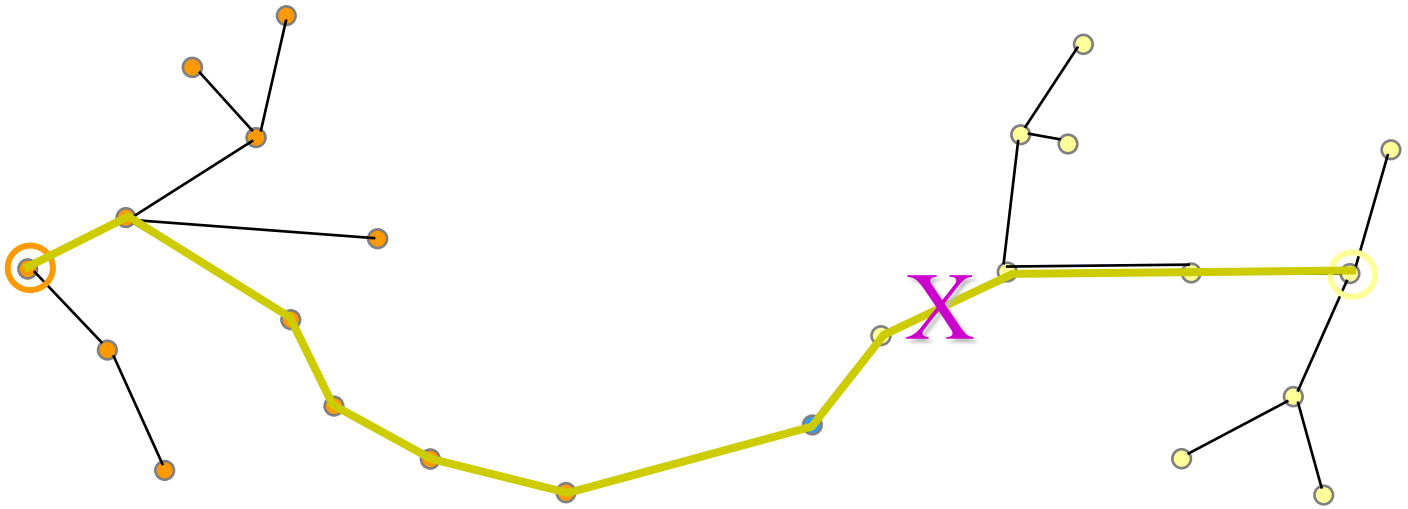
RDT Bidirectionnel



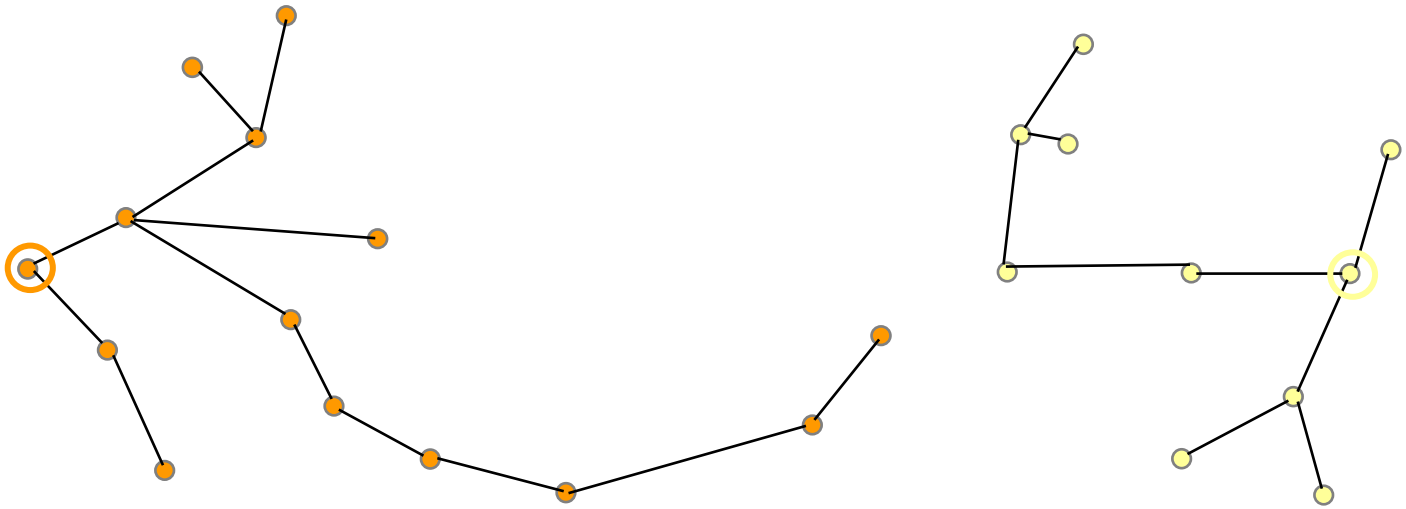
RDT Bidirectionnel



RDT Bidirectionnel

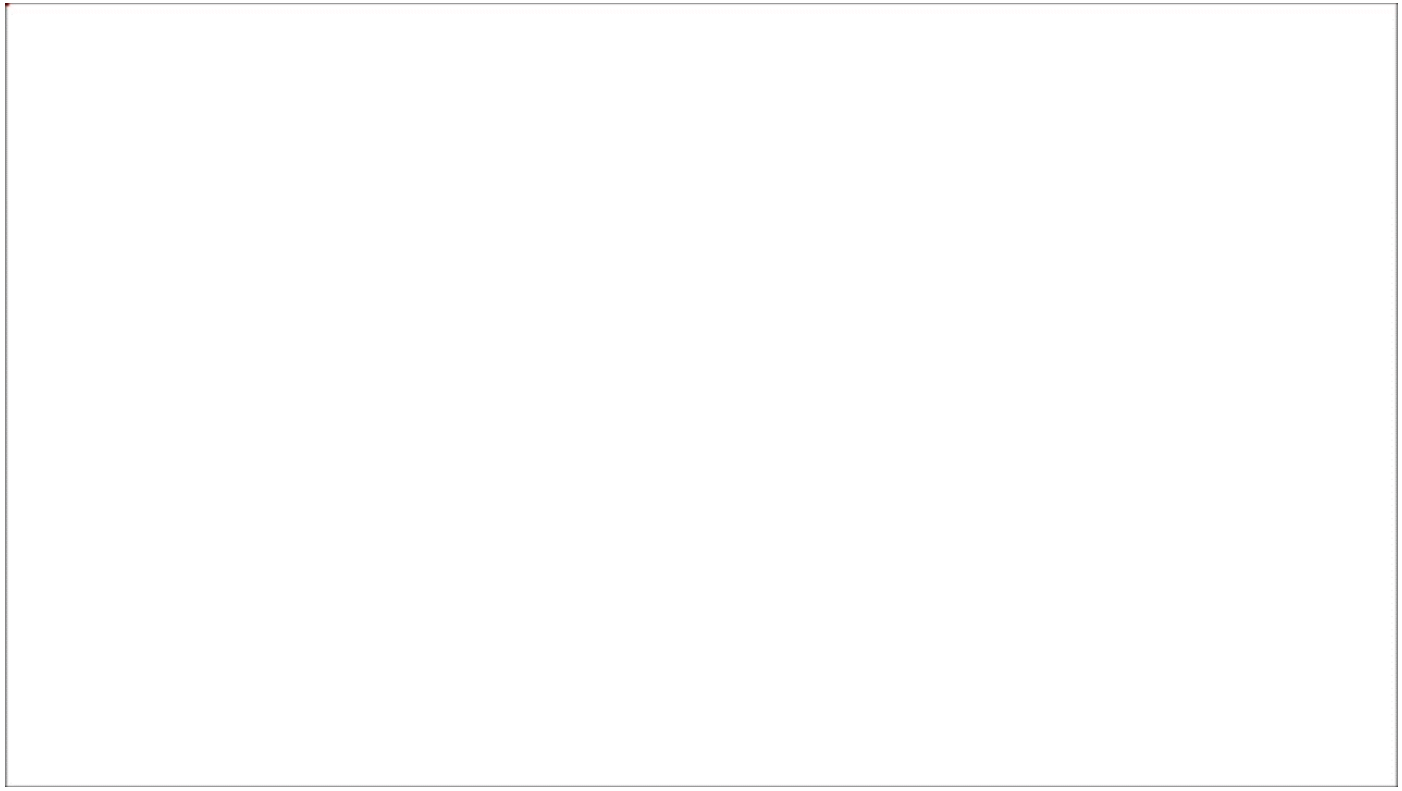


RDT Bidirectionnel



La vérification de collision pour les segments est mémorisée

Demo RDT Bidirectionnel



Plusieurs variantes

- RDT : Manuel de reference : (LaValle et al., 2006)
- RRT : Version de RDT qu'on vient de voir
- RRT*
- SBL [Hsu, Latombe, Motwani, 1997]
- Et d'autres. Voir OMPL https://ompl.kavrakilab.org/

Approches par échantillonnage
Avantages / Inconvénients

- Avantages
  - » Préférables dans les environnements de haute dimension
  - » Représentation arbitraire des obstacles et du robot
- Inconvénients
  - » Résultats potentiellement non-déterministes
  - » Garanties plus faibles de complétude
    - *Resolution complete* avec des approches déterministes
    - *Probabilistically complete* avec des approches aléatoires

Résumé des types d'approches

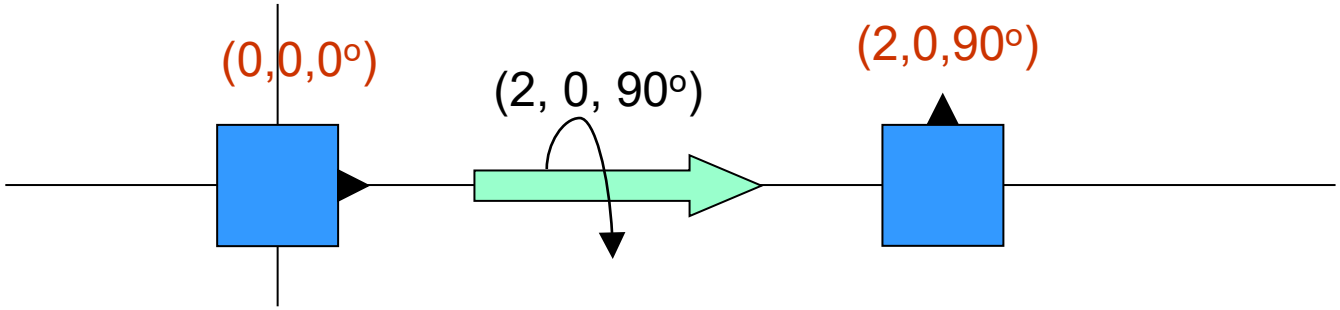
- Par décomposition
  - » Complètes
  - » Appropriées quand l'environnement et les obstacles sont connus d'avance
  - » Requièrent une représentation algébrique des obstacles
- Par échantillonnage
  - » Garanties plus faibles de complétude (résolution/probabiliste)
  - » Plus efficaces dans des espaces de grande dimension
  - » Détection de collisions comme une boîte noire (obstacles arbitraires)

Sujets couverts

- Introduction
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

Contraintes différentielles

- Dans les exemples précédents (géométriques), on assumait:
  - » Qu'il était toujours possible de relier trivialement deux états arbitraires

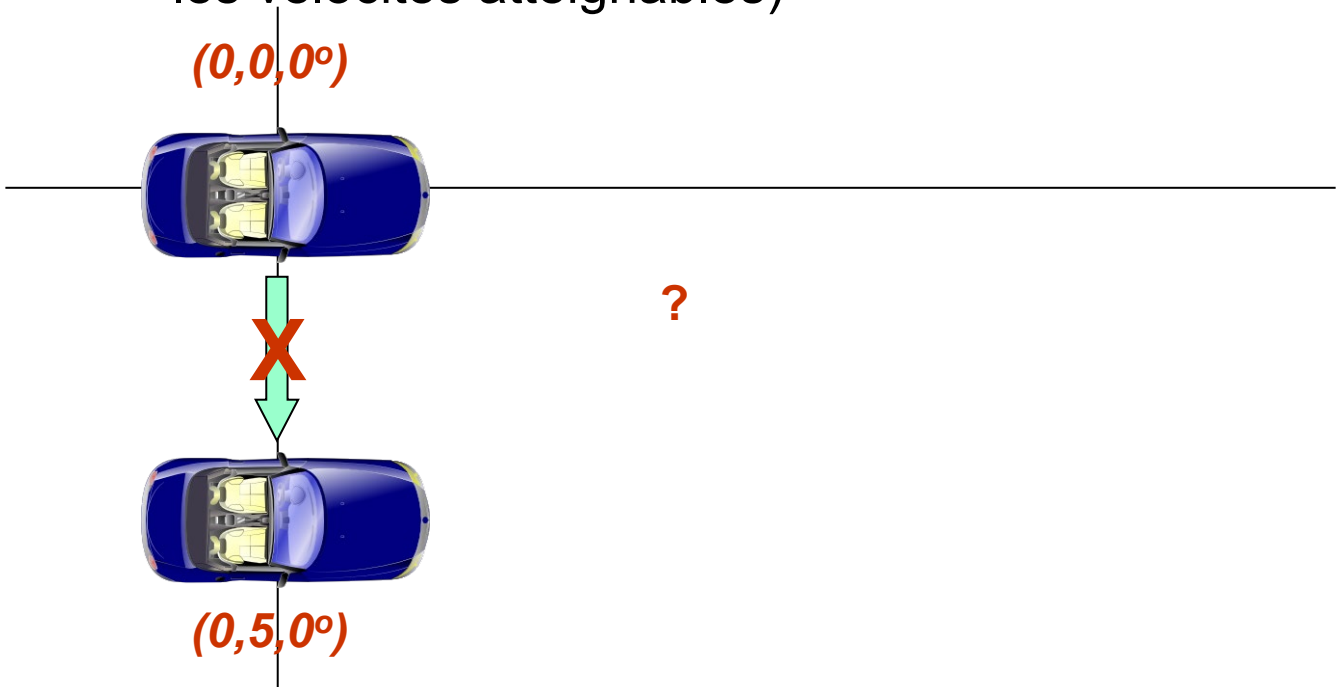


  - » Formellement, on assumait que l'espace des vélocités (dérivées) atteignables n'était pas contraint.

- En pratique, il arrive qu'on veuille modéliser un robot avec des contraintes différentielles.

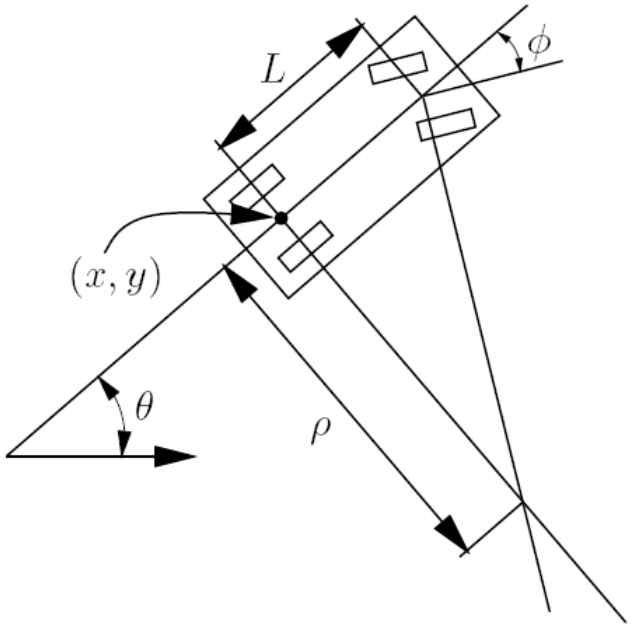
Contraintes différentielles

- Cas typique: une voiture
  - » Impossible de se déplacer latéralement! (i.e. contraintes sur les vélocités atteignables)



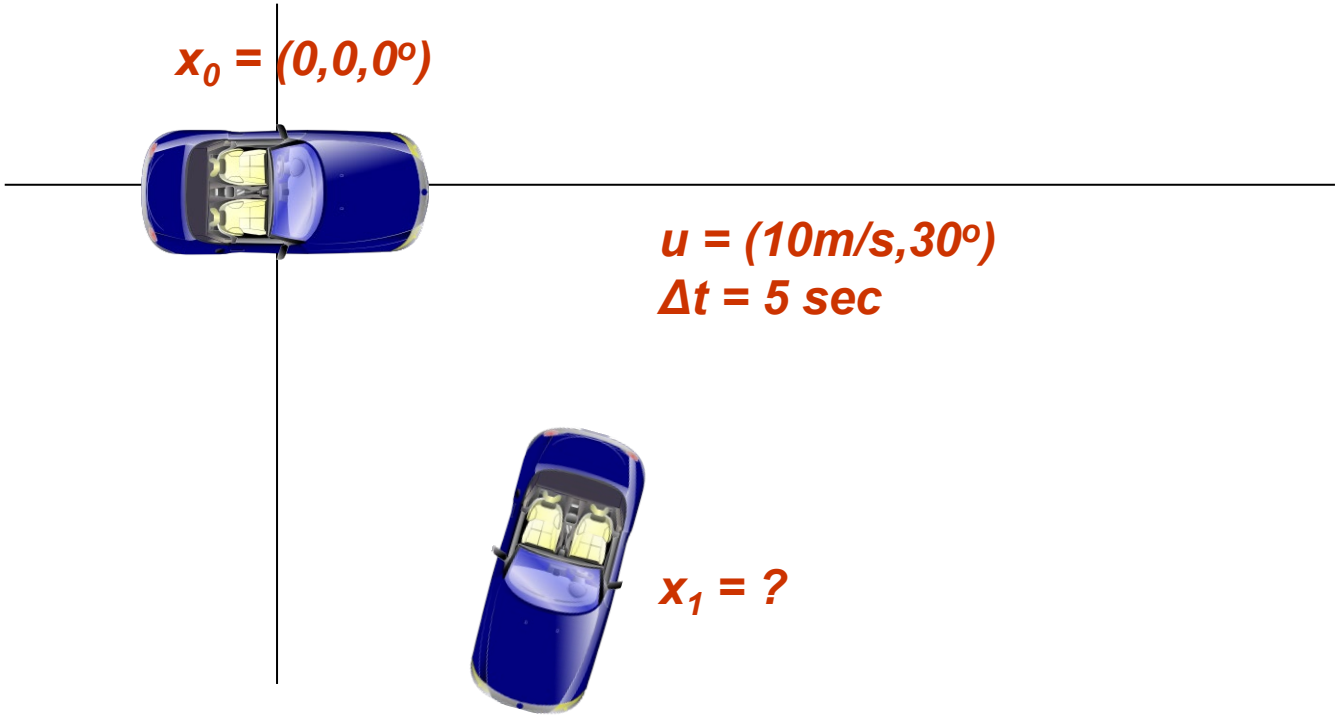
Modèle dynamique d'une voiture

- Exemple: modèle dynamique d'une voiture
- Rendre les déplacements plus *smooth* : accélération et vitesse de rotation du volant
- *Espace de contrôle (Phase space)*: on augmente l'espace de configuration
  - » 3 degrés de liberté + vitesse *s* + *rotation du volant φ*
  - » $q = (x, y, \theta, s, \varphi)$
- Espace d'actions
  - » 2 degrés de liberté (accélération + vitesse de rotation du volant)
  - » $u = (a, s_\varphi)$



Fonction de transition

- Comment obtenir le nouvel état?



Fonction de transition

- Comment obtenir le nouvel état?
  - » On intègre les vélocités dans le temps.

$$x(t) = x(0) + \int_0^t f(x(t'), u(t'))dt', \qquad (14.1)$$

  - » Intégration
    - Symbolique
    - Numérique (pour équations non-intégrables symboliquement), exemple Runge-Kutta

Récapitulation

- Deux approche possibles traiter les contraintes différentielles
  - » Planifier puis transformer (RDT/RRT sans contraintes différentielles): explorant l'espace des configurations en ignorant les contraintes différentielles et traiter les contraintes différentielles en une phase de post-traitement de la trajectoire obtenue.
  - » Planifier avec des contraintes différentielles (RDT/RRT avec contraintes différentielles): Traiter les contraintes différentielles à la volée en même temps que la détection de collisions (étendre le planificateur local)

Planification de trajectoires

- **Approche A** : Planifier puis transformer
  1) Calcul trajectoire sans tenir compte les contraintes différentielles
  2) Lissage de la trajectoire qui respecte les contraintes

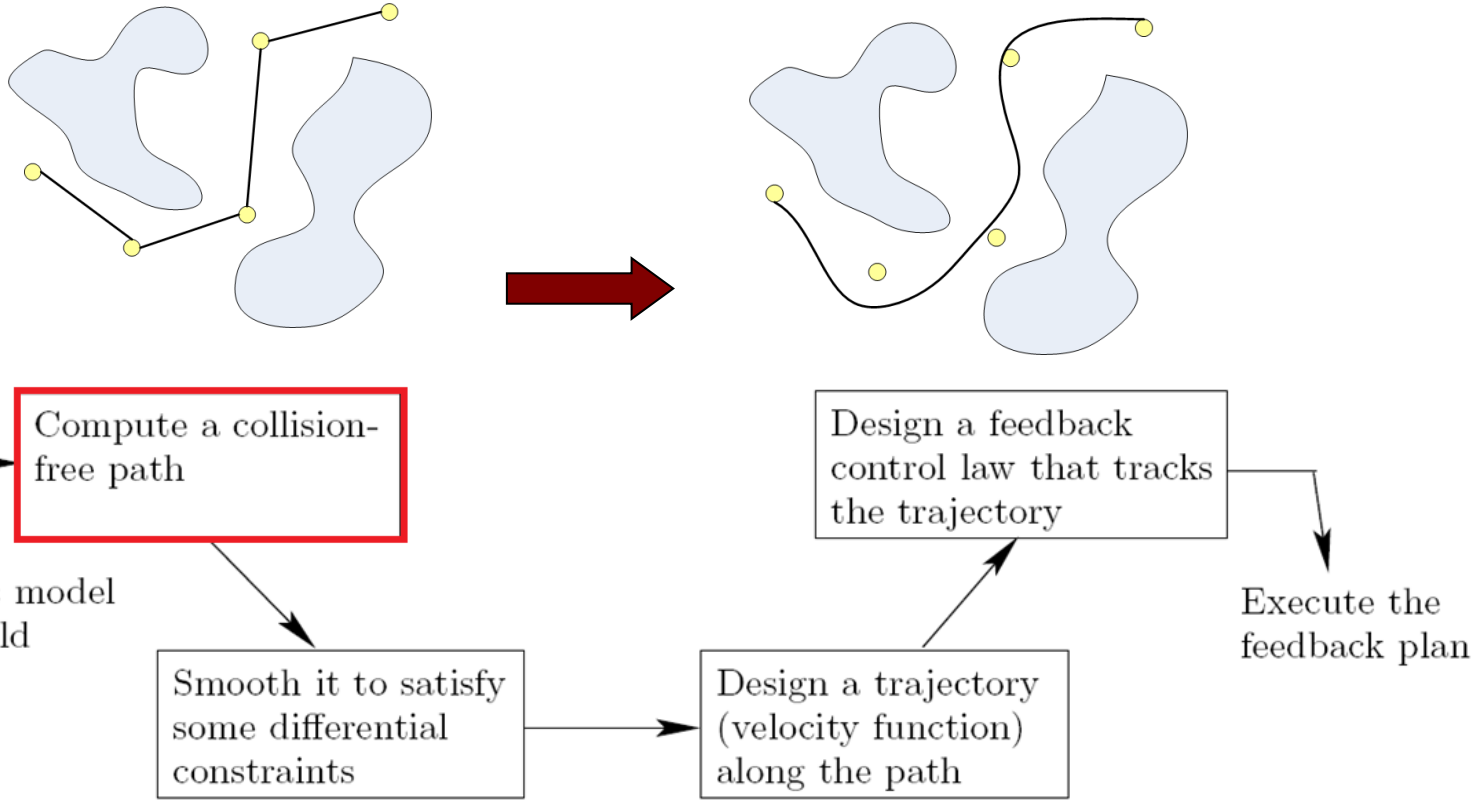
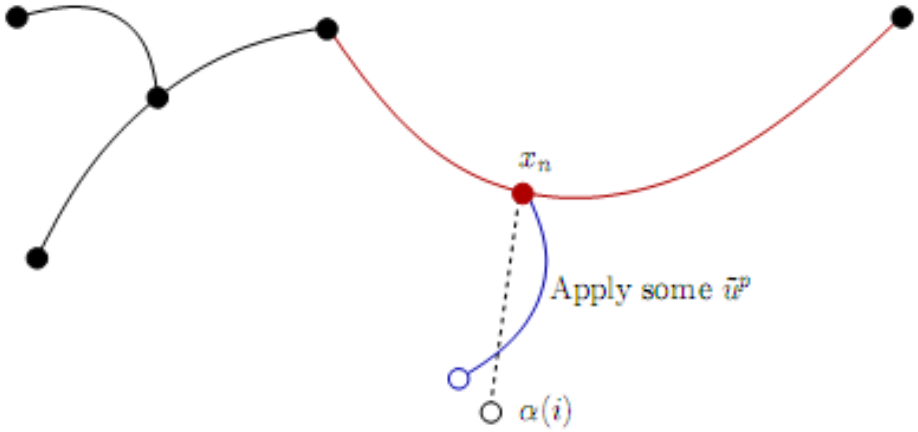


Figure 1.19: A refinement approach that has been used for decades in robotics.

Planification de trajectoires

- **Approche B** : Utiliser les RDT/RRT avec des contraintes différentielles
  - » Étendre l'arbre de façon aléatoire
    - Avec un certain biais vers le but (l'état final)
  - » Échantillonnage de l'espace de contrôle

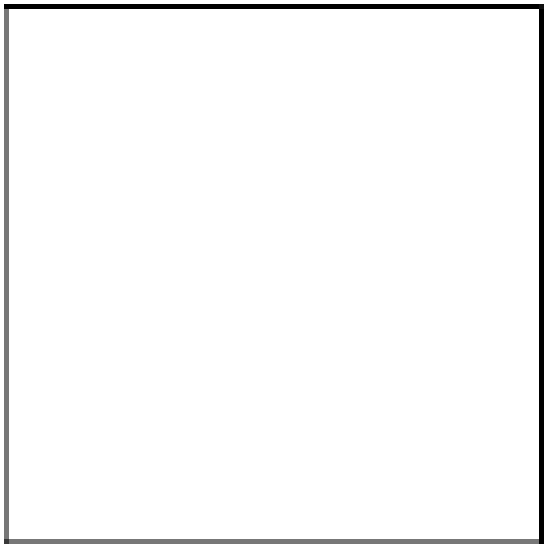


Planification de trajectoires

- Utiliser les RDTs

SIMPLE_RDT_WITH_DIFFERENTIAL_CONSTRAINTS(x_0)
1 $\mathcal{G}$.init(x_0);
2 **for** $i = 1$ **to** k **do**
3 $x_n \leftarrow$ NEAREST($S(\mathcal{G}), \alpha(i)$);
4 $(\tilde{u}^p, x_r) \leftarrow$ LOCAL_PLANNER($x_n, \alpha(i)$);
5 $\mathcal{G}$.add_vertex(x_r);
6 $\mathcal{G}$.add_edge($\tilde{u}^p$);

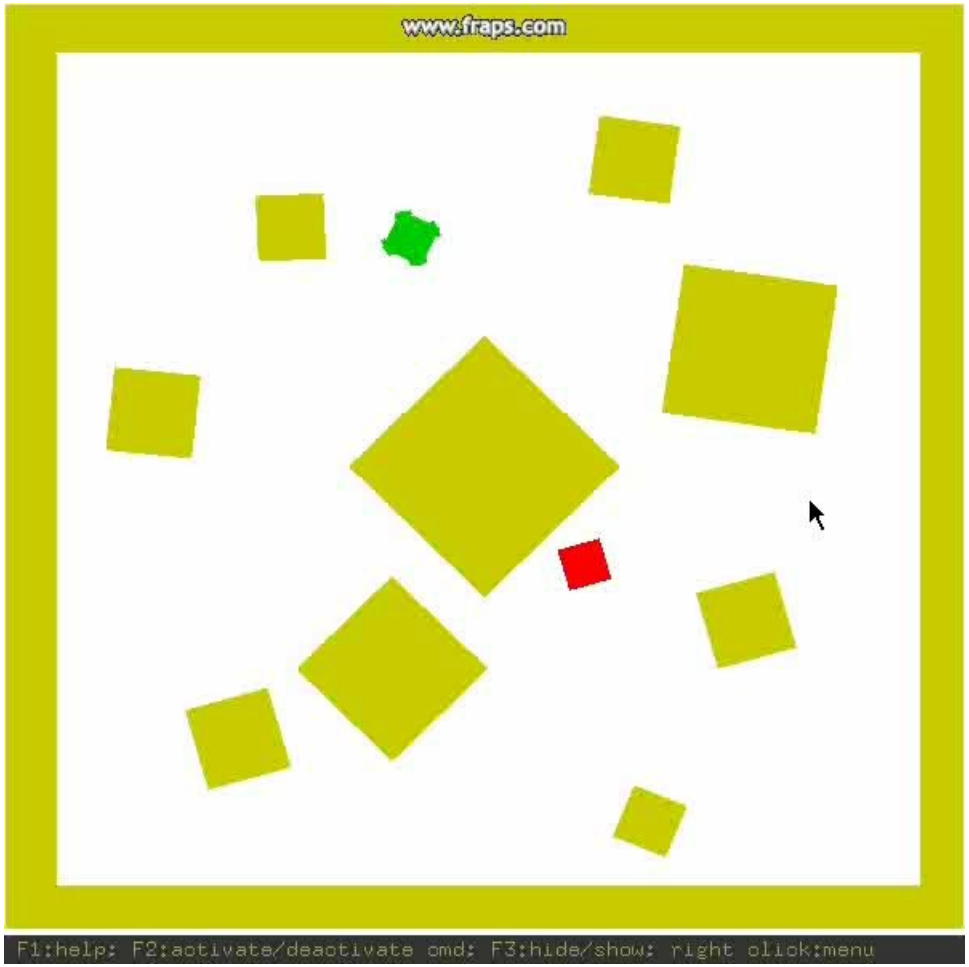
Figure 14.19: Extending the basic RDT algorithm to handle differential constraints. In comparison to Figure 5.16, an LPM computes x_r, which becomes the new vertex, instead of $\alpha(i)$. In some applications, line 4 may fail, in which case lines 5 and 6 are skipped.



- Différence principale avec la version géométrique: le *local planner* s'assure que la trajectoire générée respecte les contraintes différentielles

Planification de trajectoires

- Exemple avec RDT



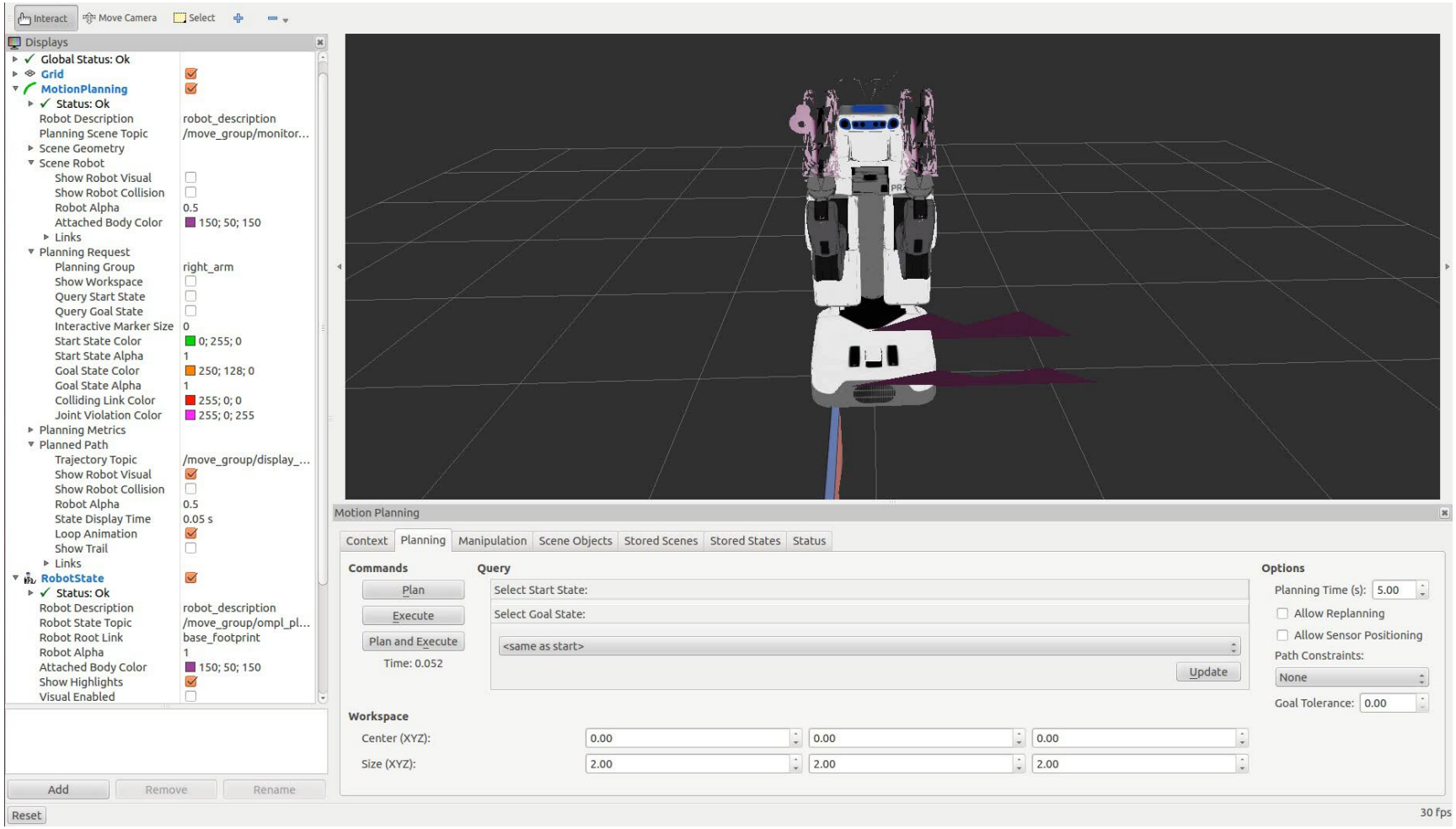
Plan

- Introduction
- Représentation et transformation des entités
- Espace de configuration
- Approches de planification
  - » Exactes
  - » Par échantillonnage
- Contraintes différentielles
- Outils suggérés

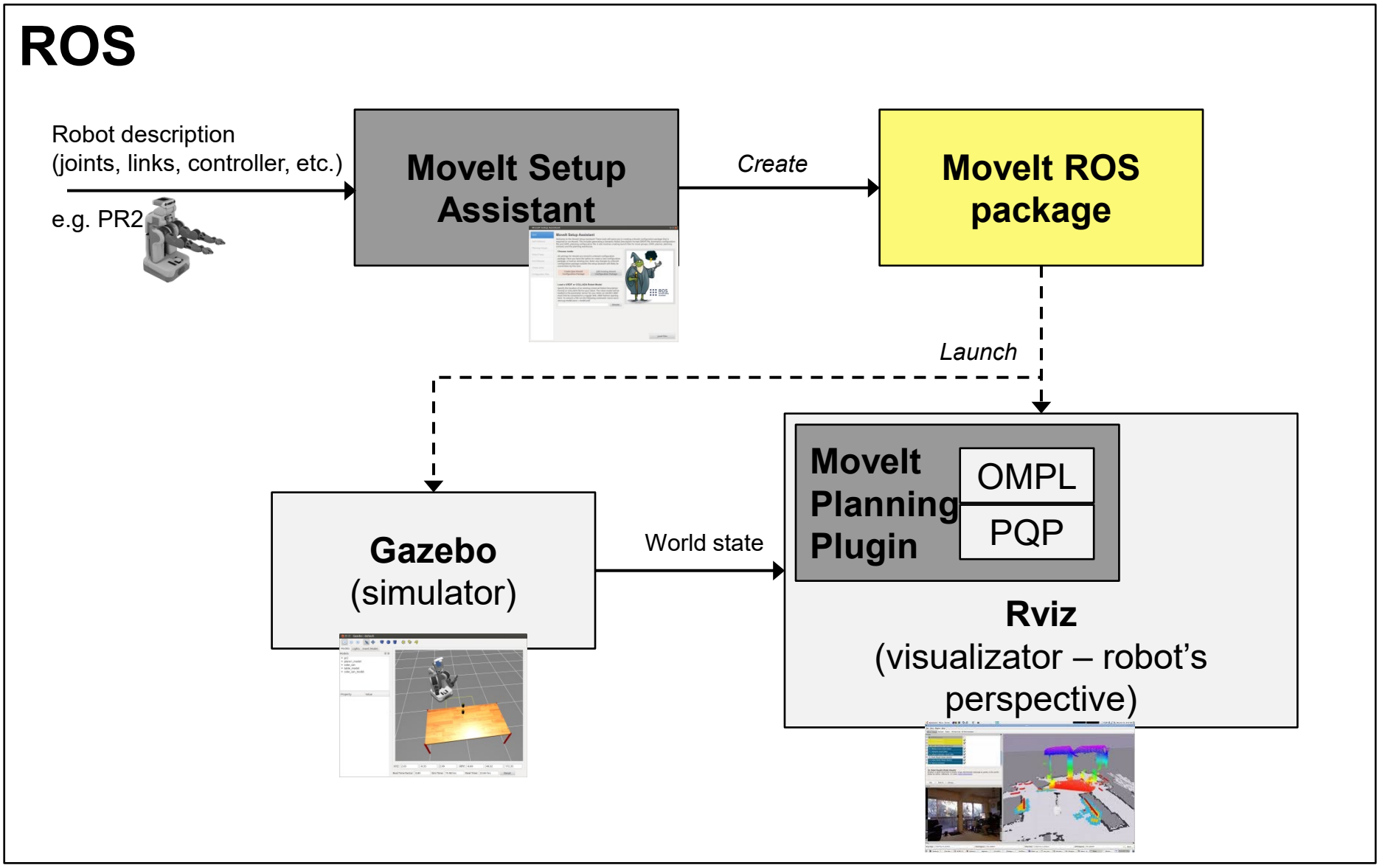
Outils

- ROS (Robotics Operating System)
  - Facilite la programmation bas-niveau du contrôle d'un robot
- OMPL (Open Motion Planning Library)
  - Planification de trajectoires
- MoveIT
  - Intégration de ROS, OMPL, un détecteur decollision, un simulateur (e.g., Gazebo), et d'autres composantes.

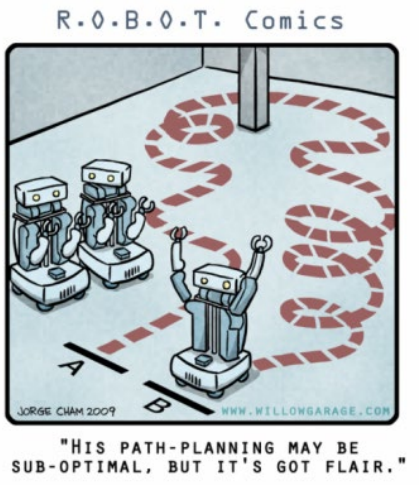
Démo de planification avec MoveIT



Architecture d'intégration pour MoveIT



Ce qu'il fau retenir



- Planification de trajectoires
  - » Problème continu et discret
  - » Approches par décomposition exacte
    - Trapézoidale
    - Graphe de visibilité
    - Grille
  - » Algorithme par échantillonnage : RRT
  - » Algorithme par échantillonnage avec des contraintes différentielles: RDT
- Non couvert…
  - » Environnement non-déterministe
  - » Obstacles dynamiques
  - » Contraintes temporelles
- Initiation au packages OMPL, ROS et MoveIt