Intelligence artificielle de mouvement humanoïde

Karim Bouyarmane

[Video] DARPA Robotics Challenge 2015, Team IHMC Robotics
(Florida Instiute for Human and Machine Interaction

Sujets abordés lors de cette présentation

I. Géométrie/Algorithmes : Rappels planification de chemin / planification de mouvement en robotique

II. Robotique Humanoïde/IA Humanoïde/CG : Cas du robot humanoïde : framework d'intelligence artificielle de mouvement

III. Automatique/Control theory : Correction, complétude et stabilité des contrôleurs humanoïdes

IV. Projet à moyen/long terme : X-AgH-Mint

Rappels planification de chemin / planification de mouvement en robotique

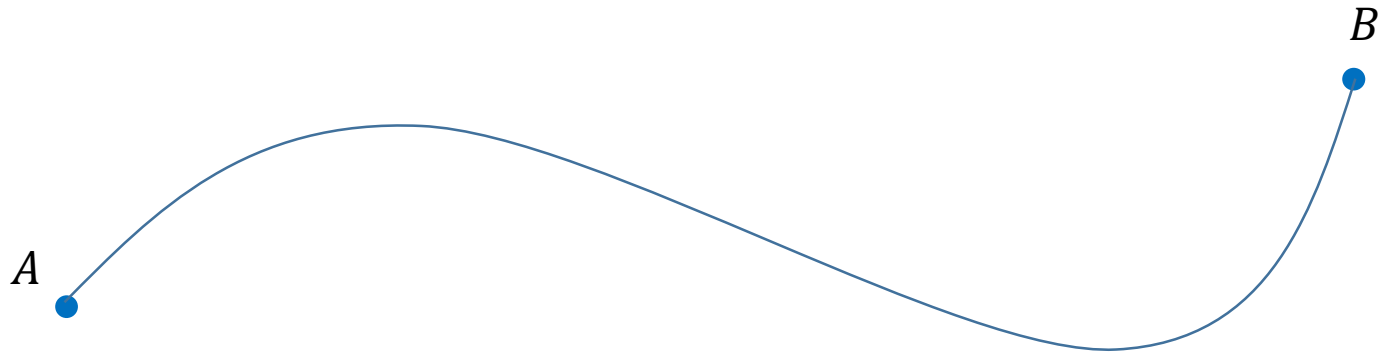
I. Géométrie/Algorithmes

Rappels sur le problème de planification de chemin

Formulation du problème

- Planification d'un chemin continu d'un point A à un point B dans un espace topologique X connexe par arcs (garantissant existence d'une solution)

- *i.e.* trouver une fonction continue f de $[0,1]$ dans X telle que $f(0) = A$ et $f(1) = B$.



Hypothèses, contraintes

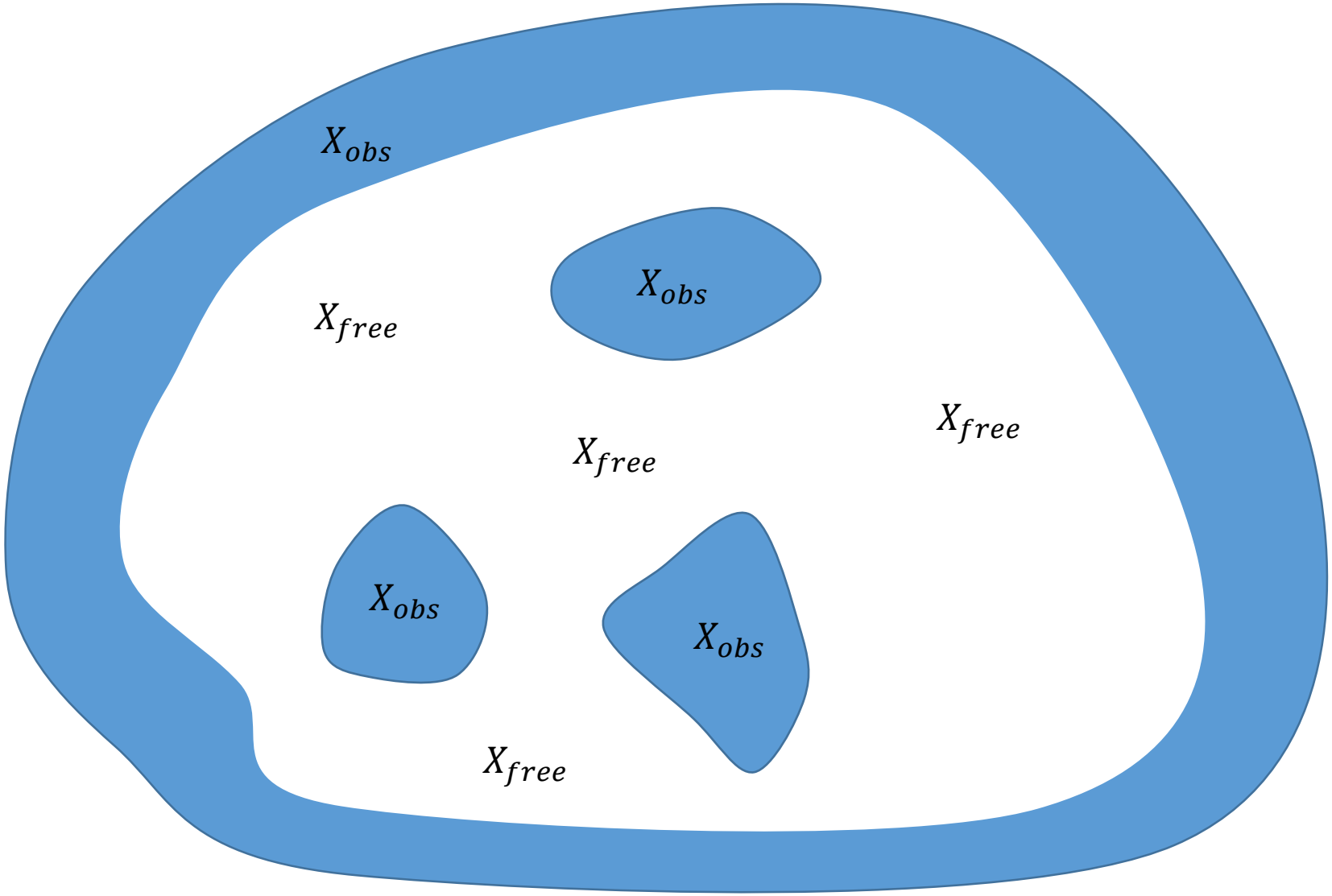
- On va supposer que X est une variété de dimension n (sous-espace de $\mathbb{R}^m$ localement homéomorphe en chaque point à $\mathbb{R}^n$)

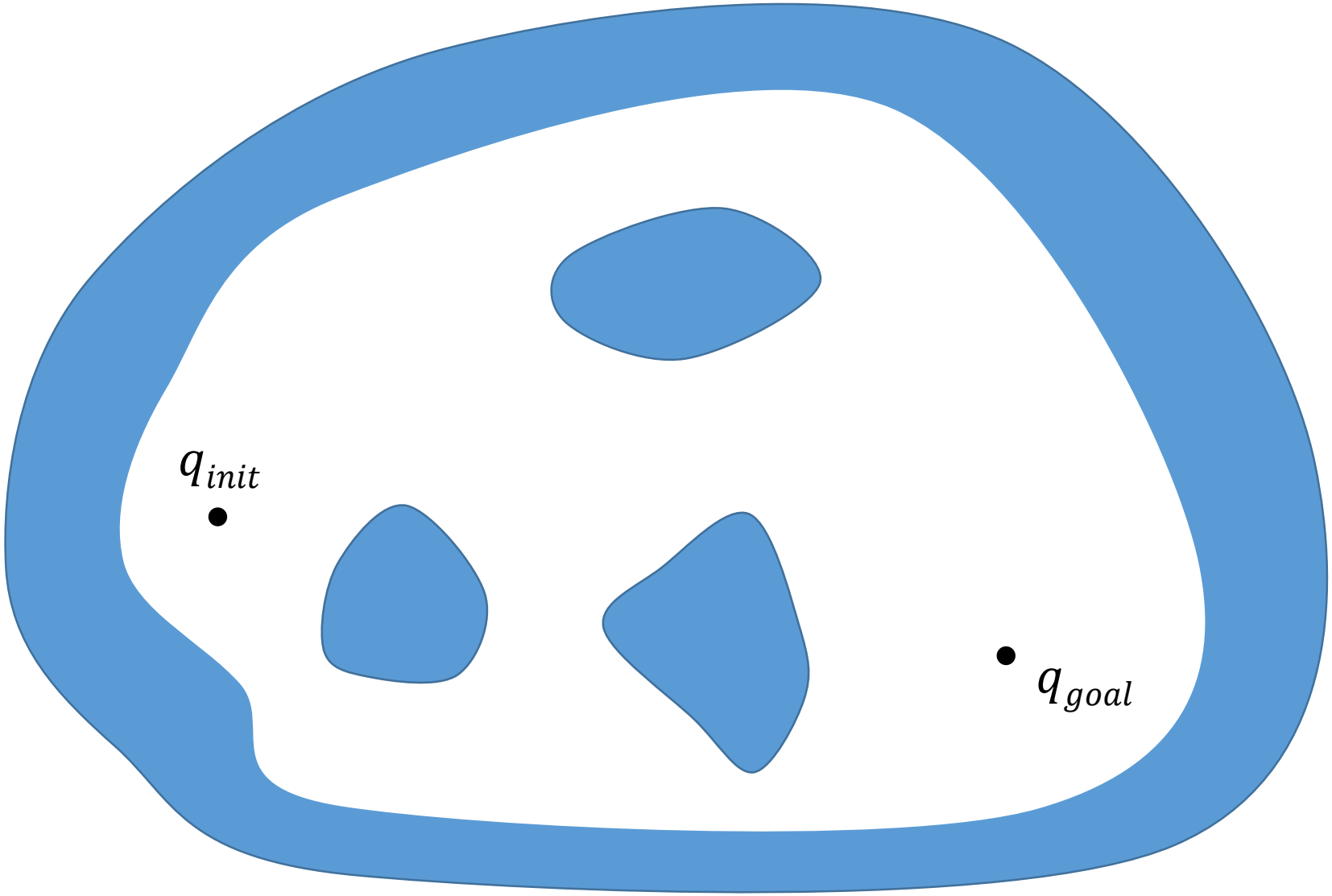
- l'espace contient des **obstacles** définis comme un **sous-ensemble compact** (fermé et borné) de X: X_{obs}

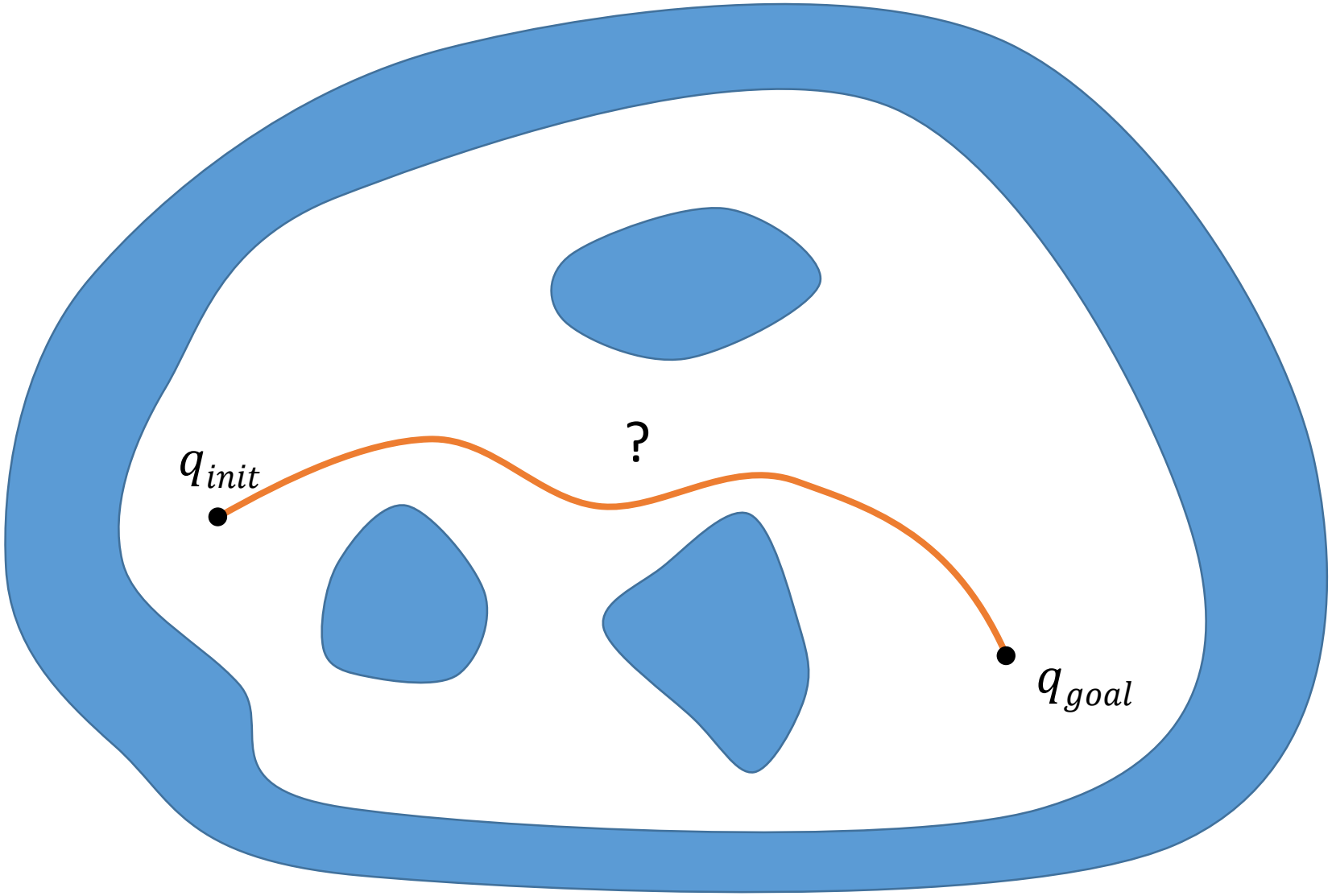
  - L'espace X est alors partagé en deux sous-ensembles complémentaires $X = X_{obs} \cup X_{free}$

  - La planification de chemin aura lieu dans X_{free}, en évitant X_{obs}

- X_{free} ensemble ouvert ⇒ **pas de chemin « optimal » en général**, nous ne nous autorisons pas à toucher des obstacles


X_{obs}
X_{obs}
X_{free}
X_{free}
X_{free}
X_{obs}
X_{obs}
X_{free}

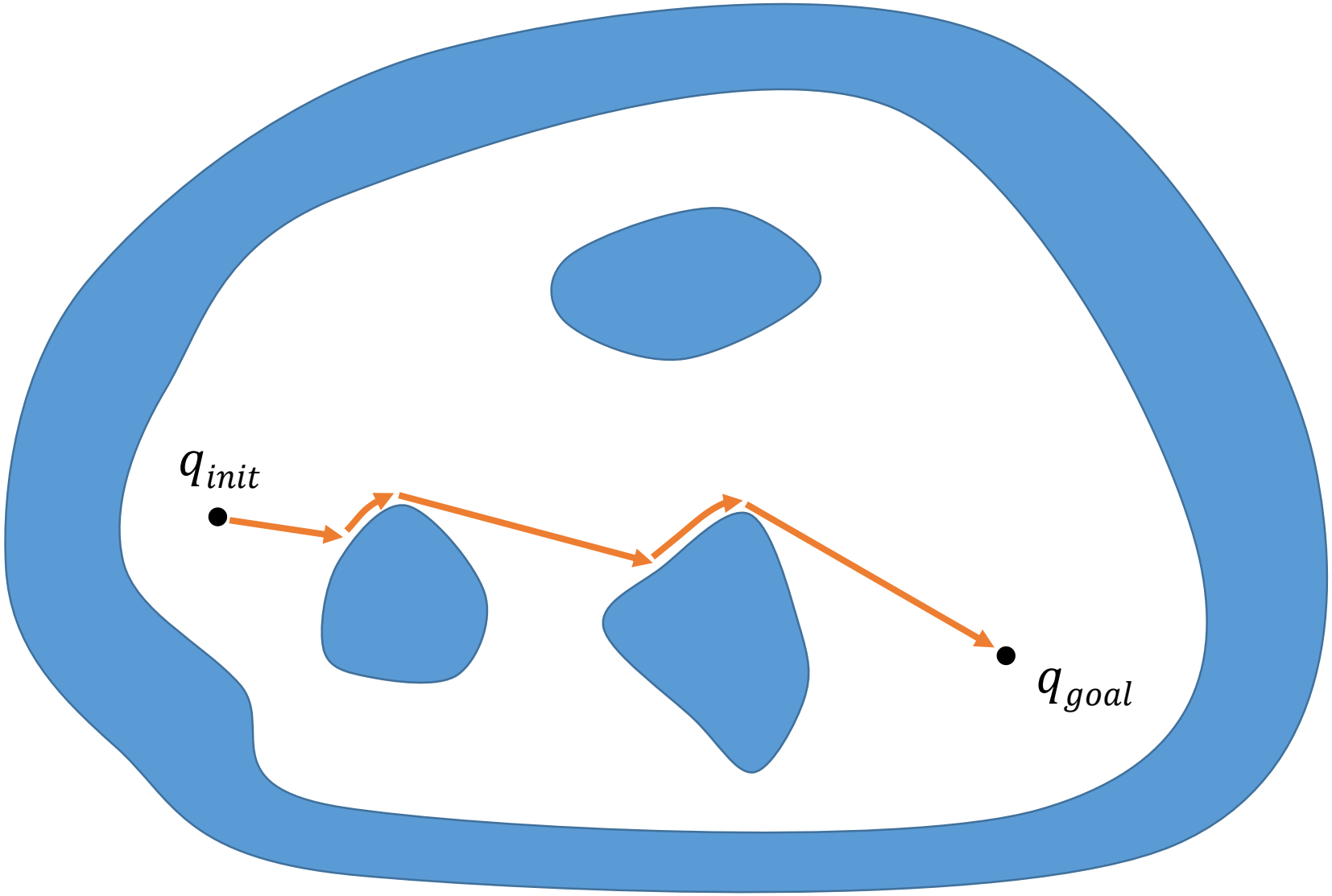

q_{init}
q_{goal}


?
q_{init}
q_{goal}

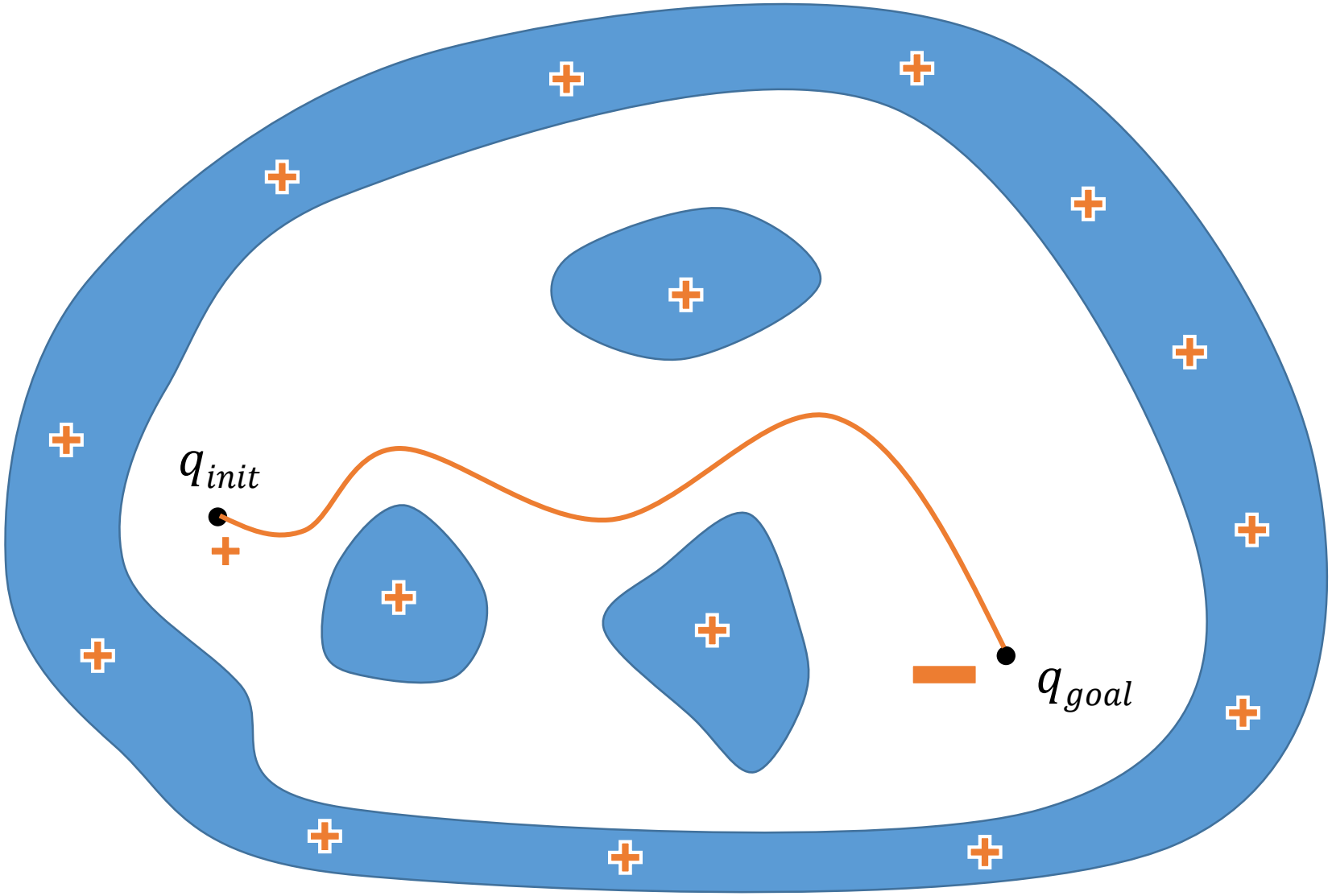
Algorithmes de planification de chemin

Approches intuitives

Bug algorithm



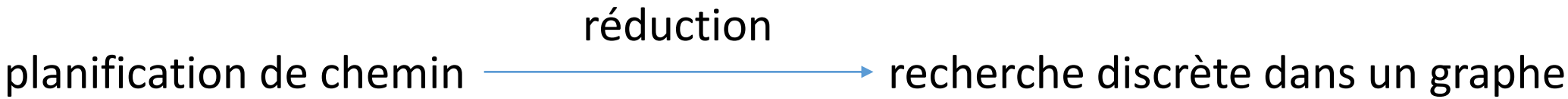
Potential field



Approches par graphes

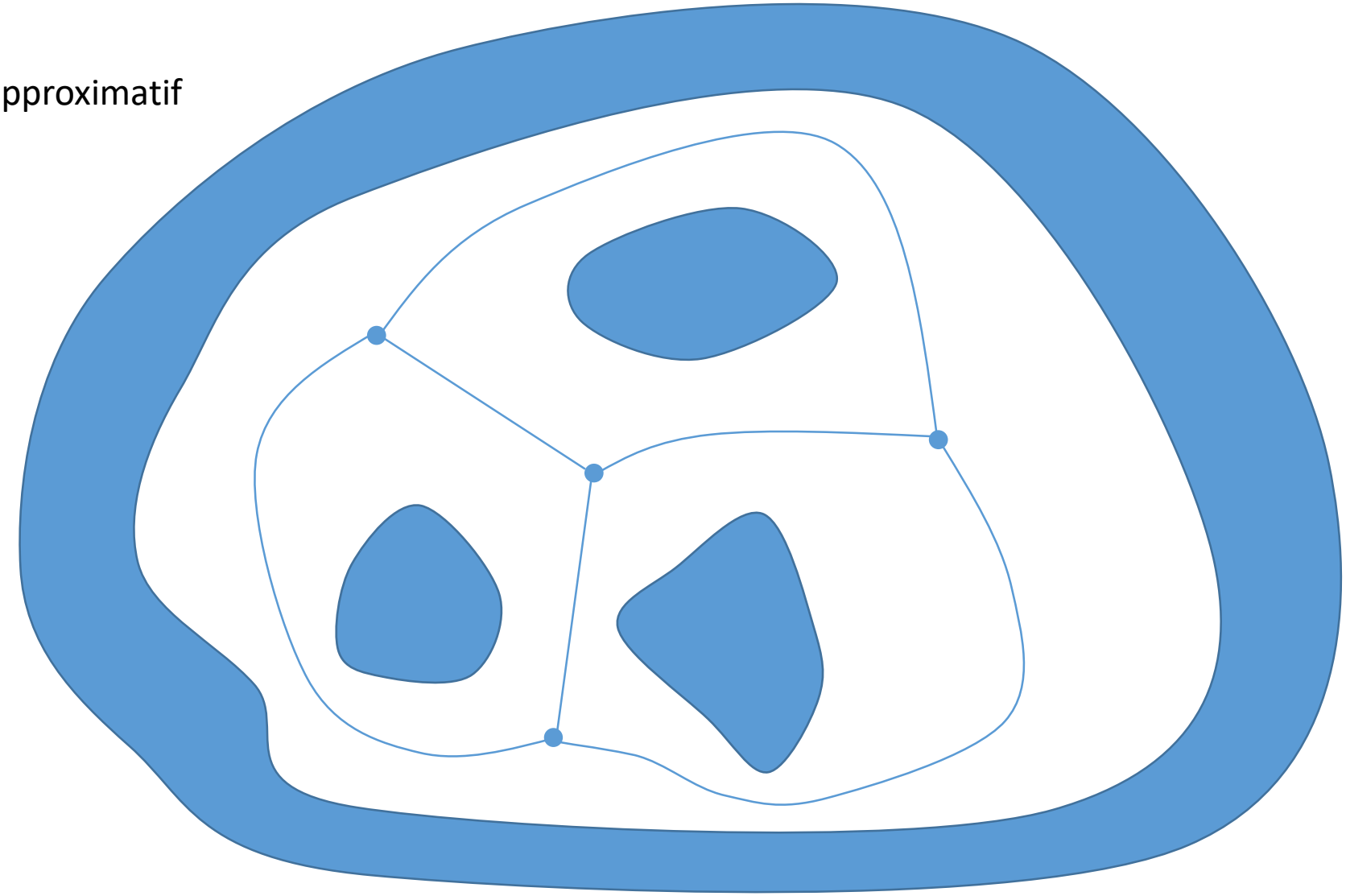
Approches par graphes

- Idée : encoder la connexité de l'espace libre dans une structure discrète de graphe

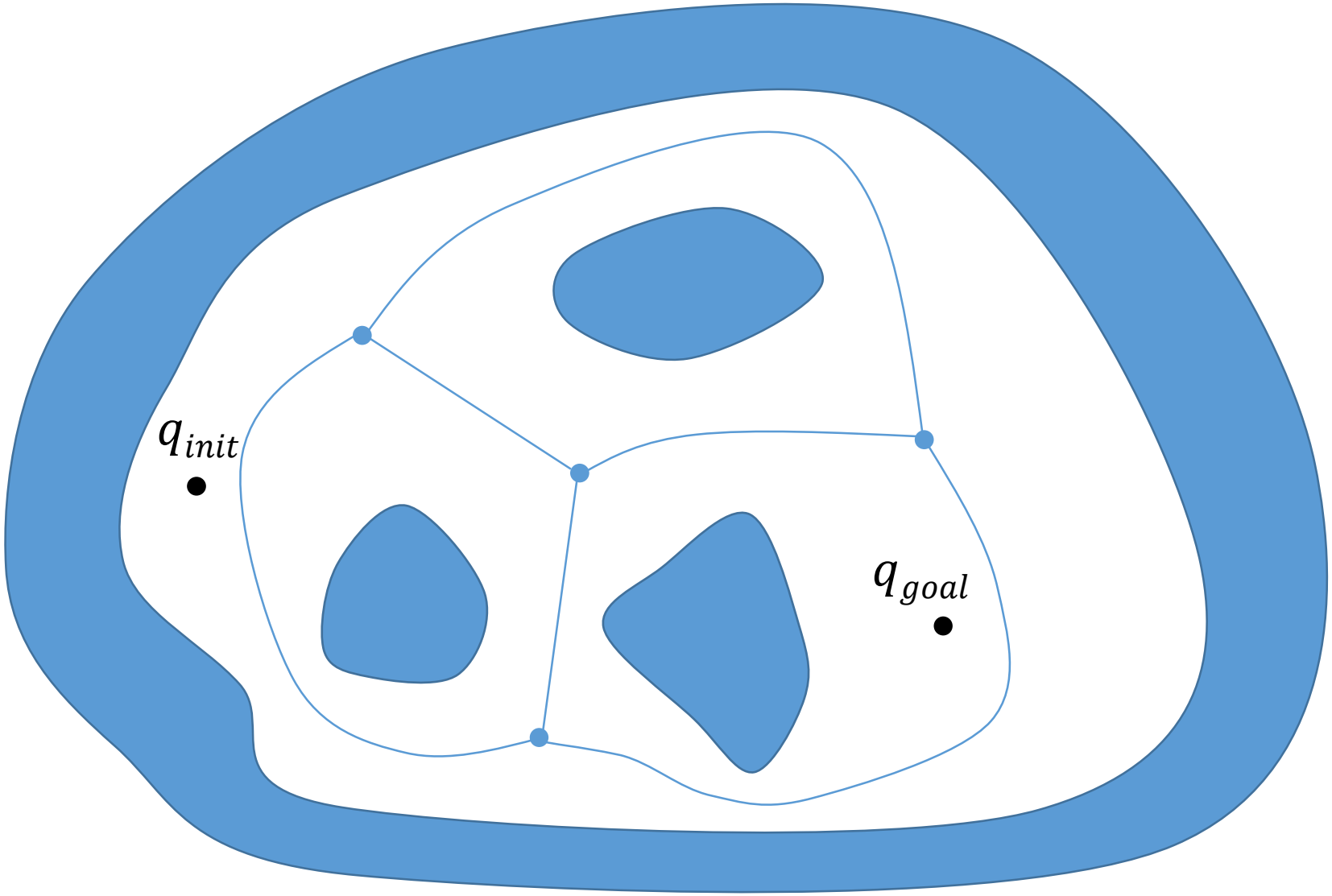


Extended Voronoi diagrams

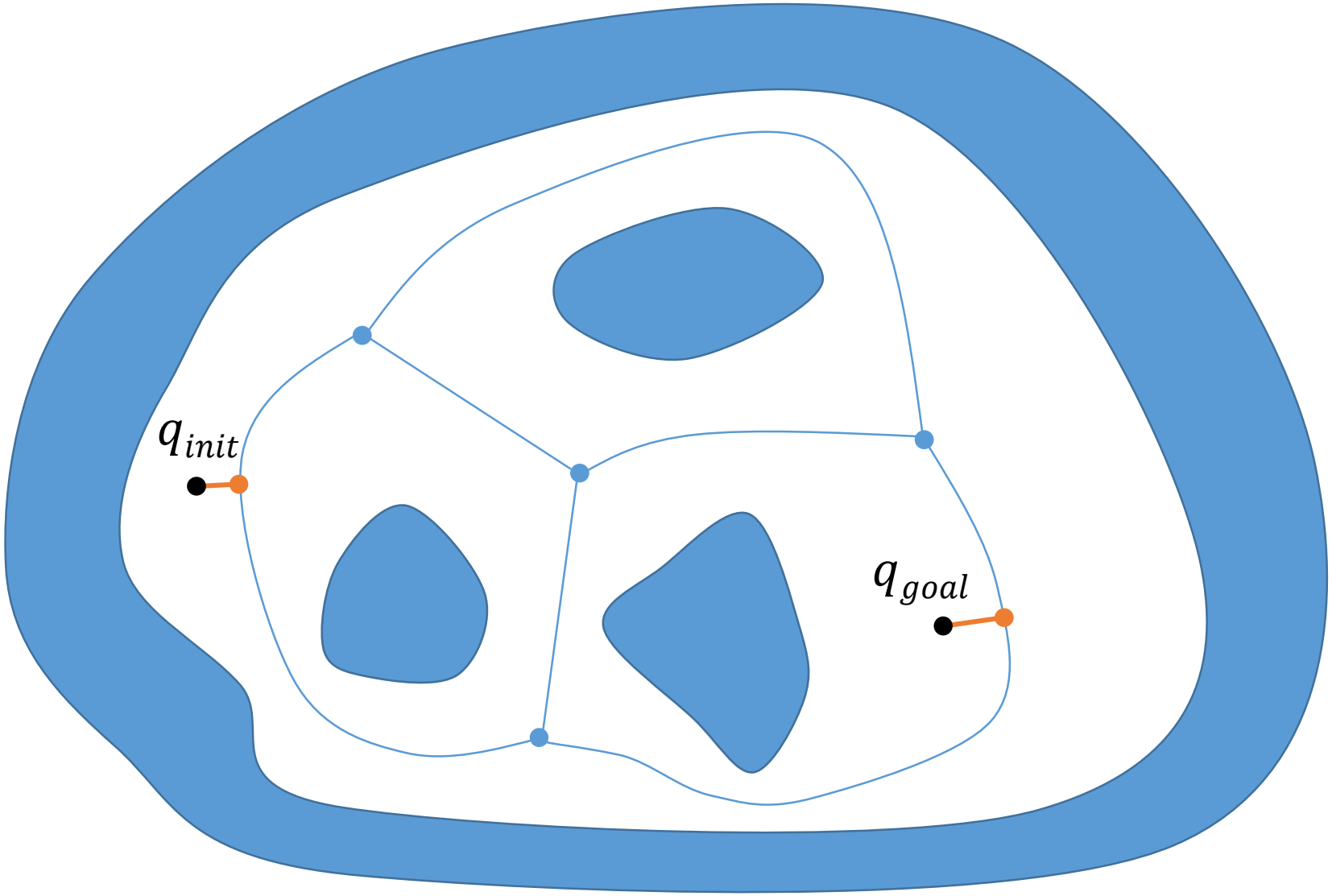
* dessin très approximatif



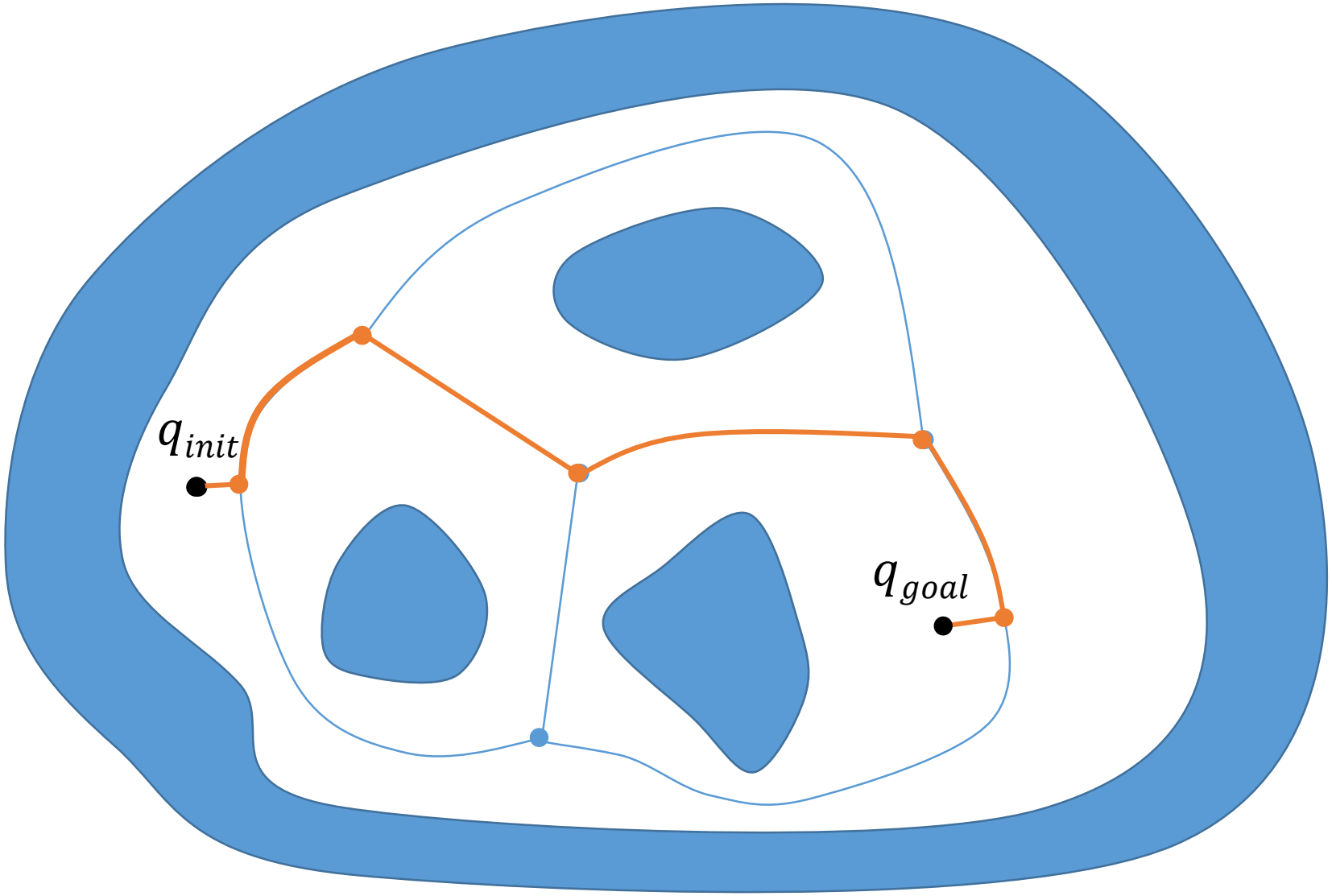
Extended Voronoi diagrams



Extended Voronoi diagrams

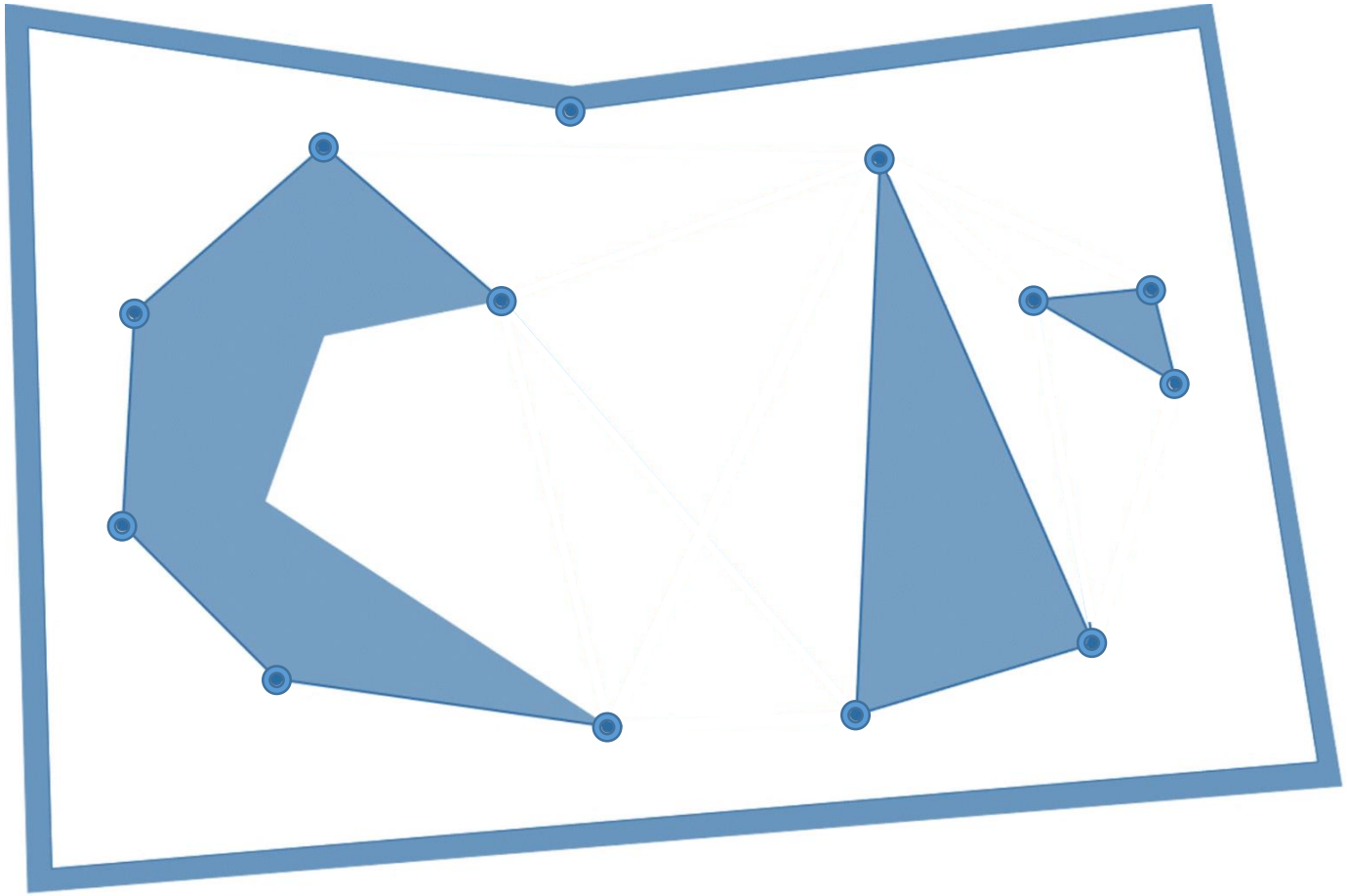


Extended Voronoi diagrams



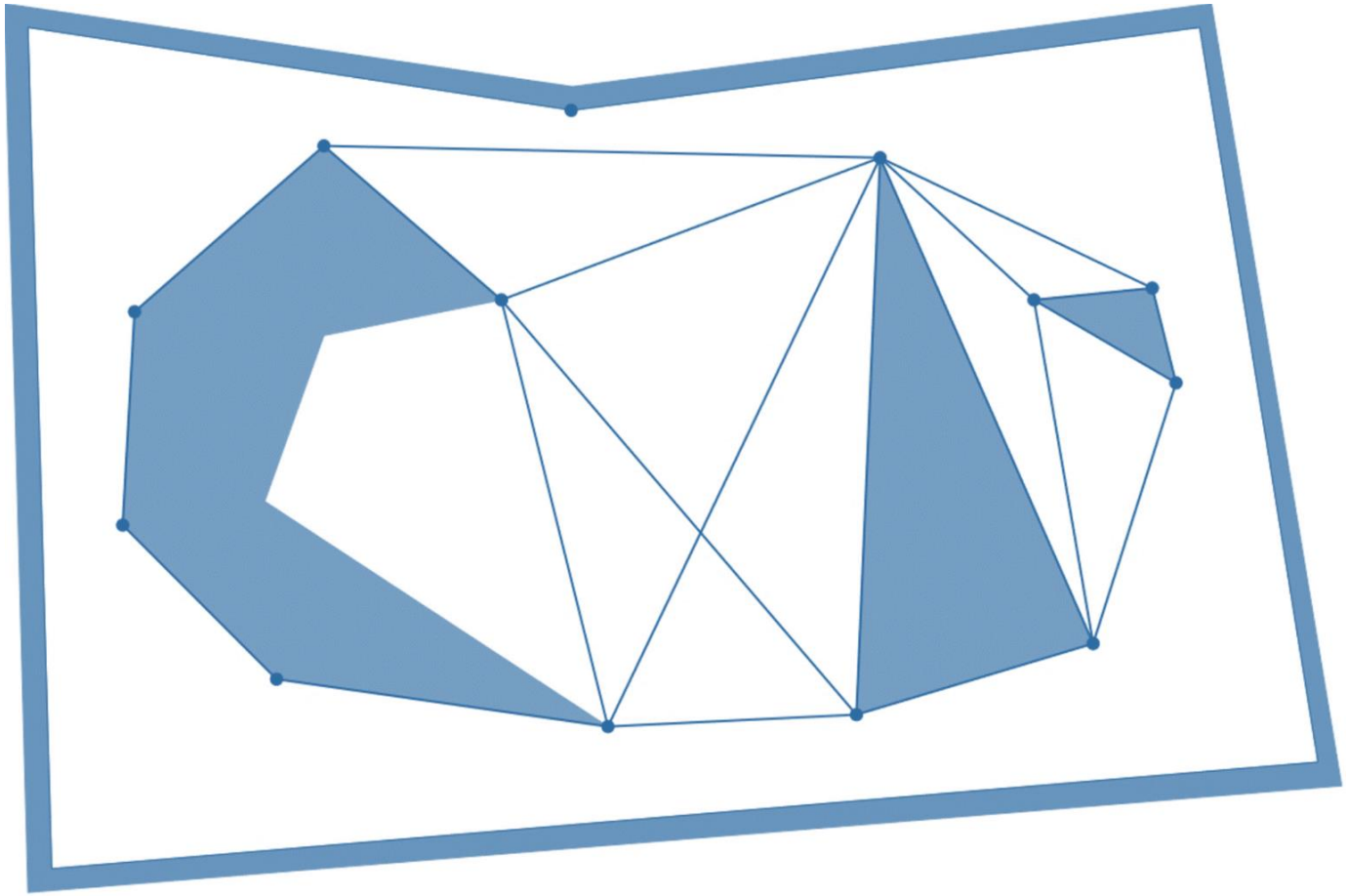
Reduced visibility roadmaps

- trouver les sommets de reflexion

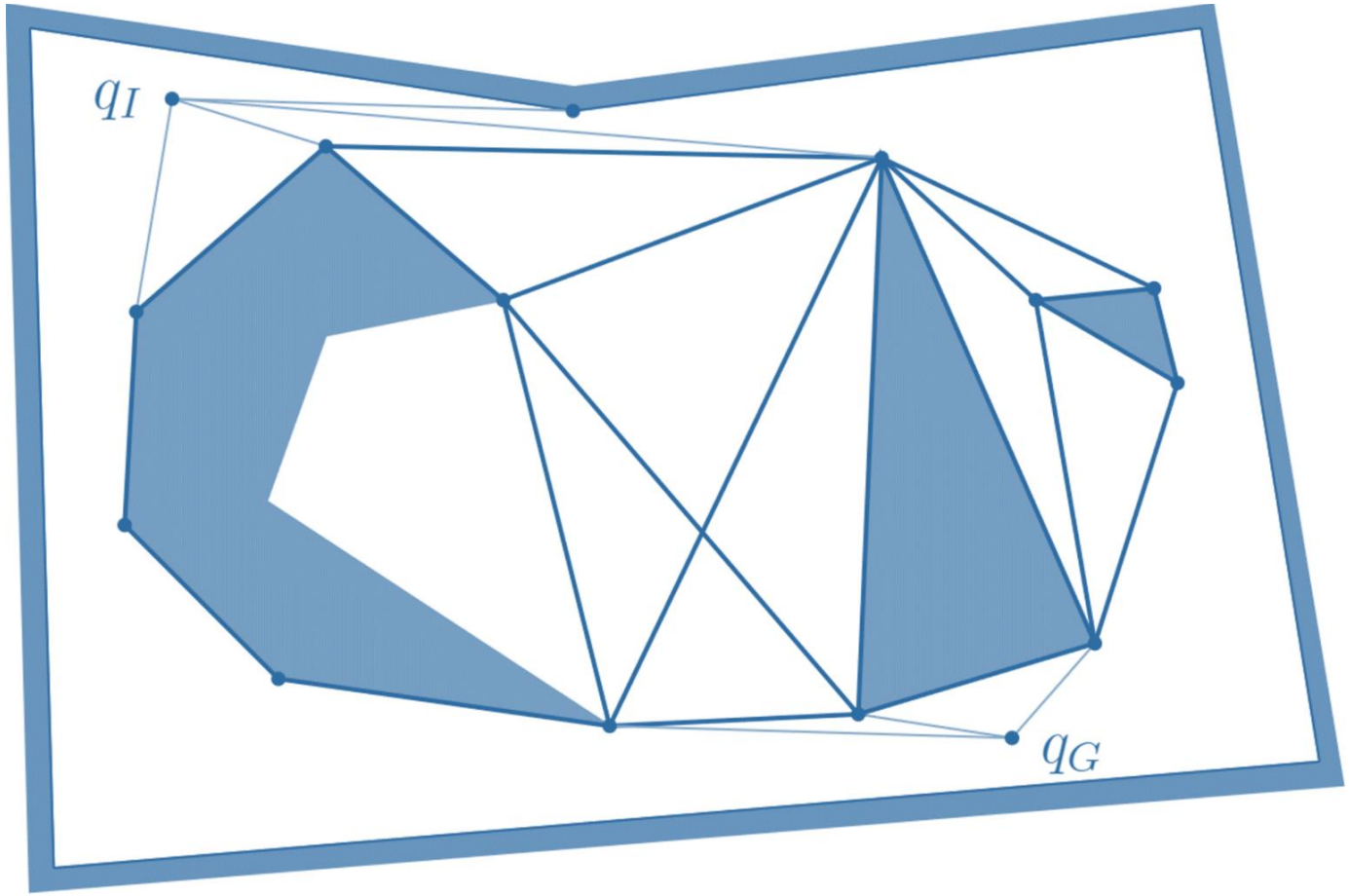


Reduced visibility roadmaps

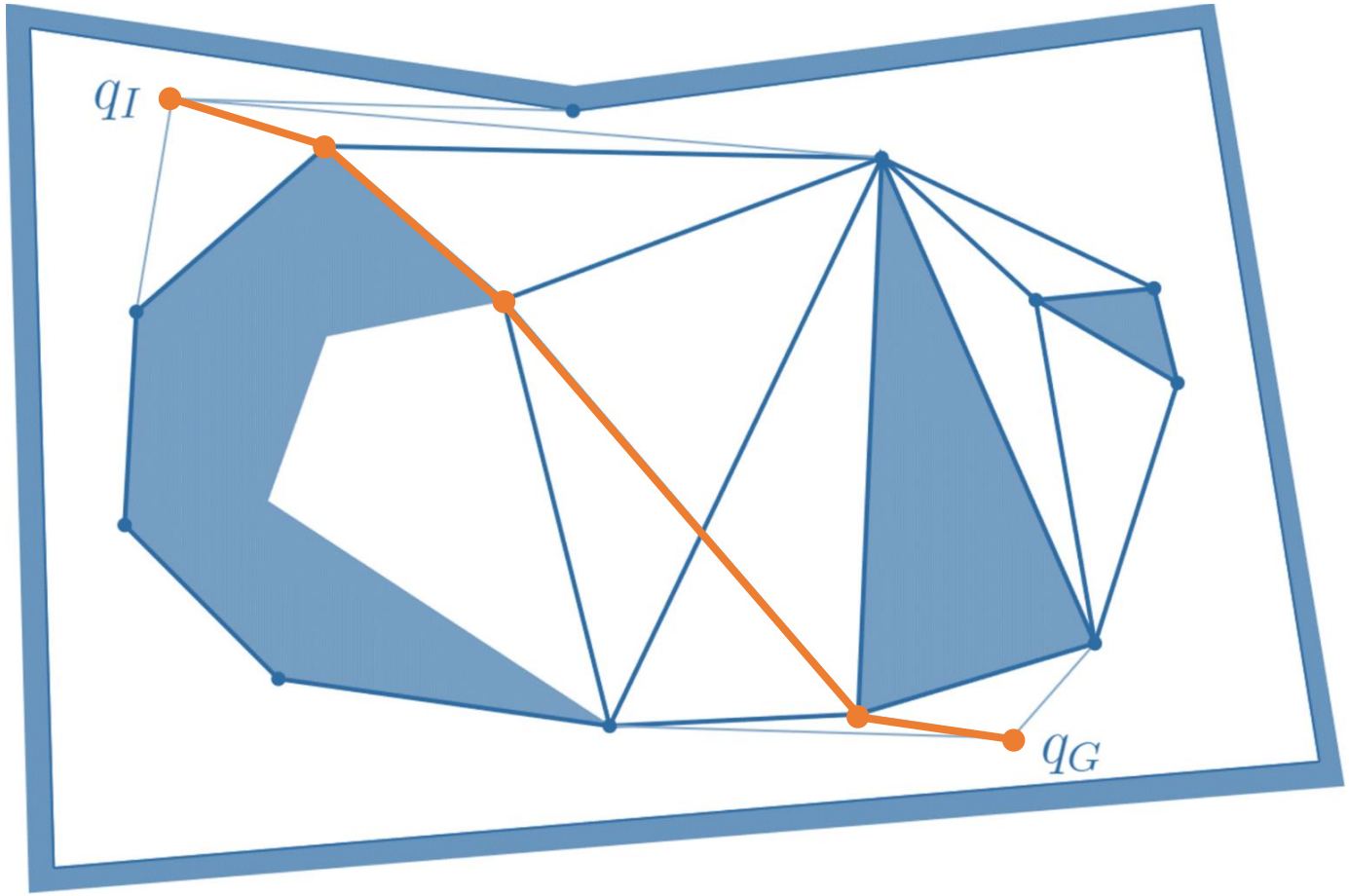
- Connecter:
  - Les sommets de reflexion consécutifs sur un même polygone
  - les arrêtes bitangeantes



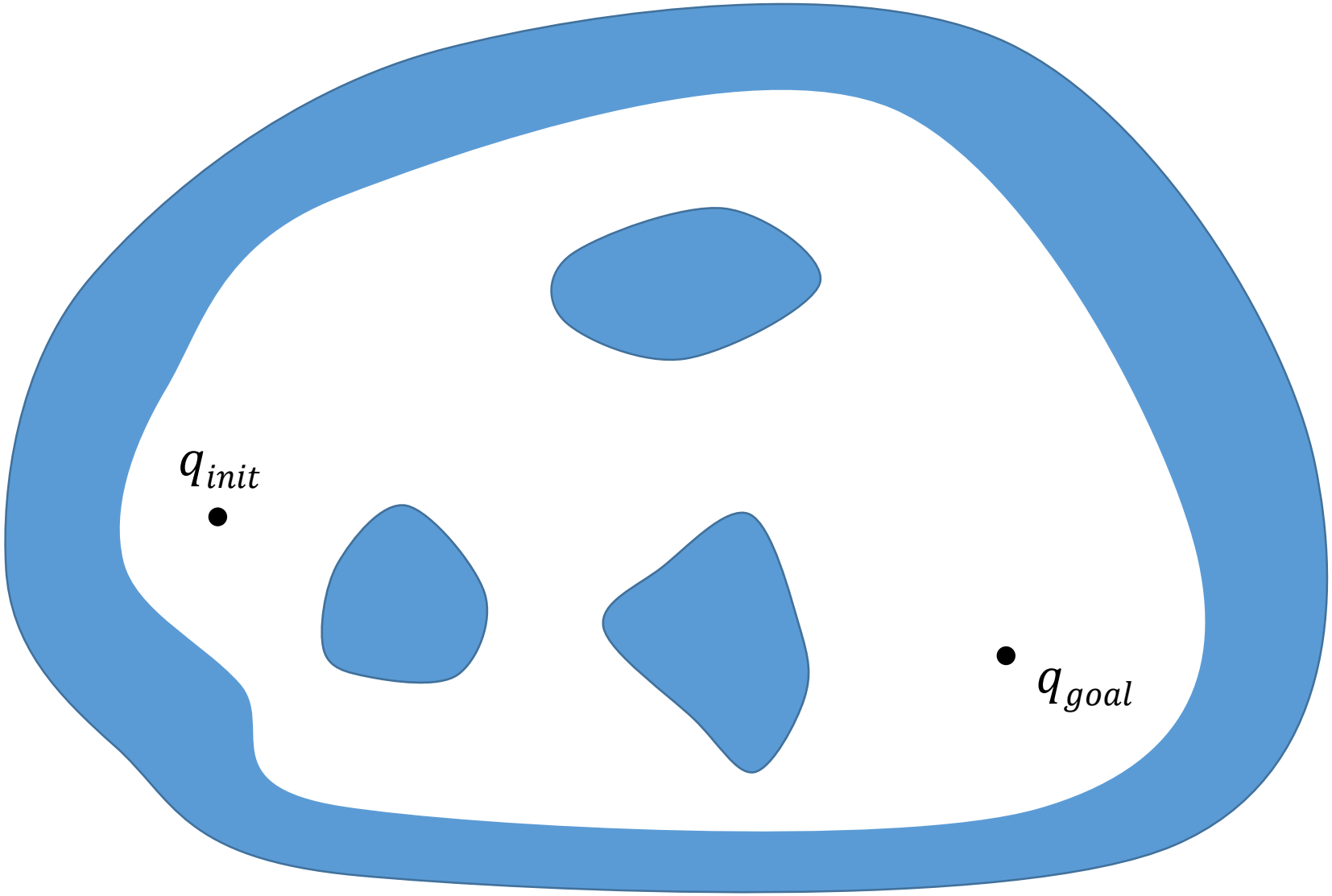
Reduced visibility roadmaps



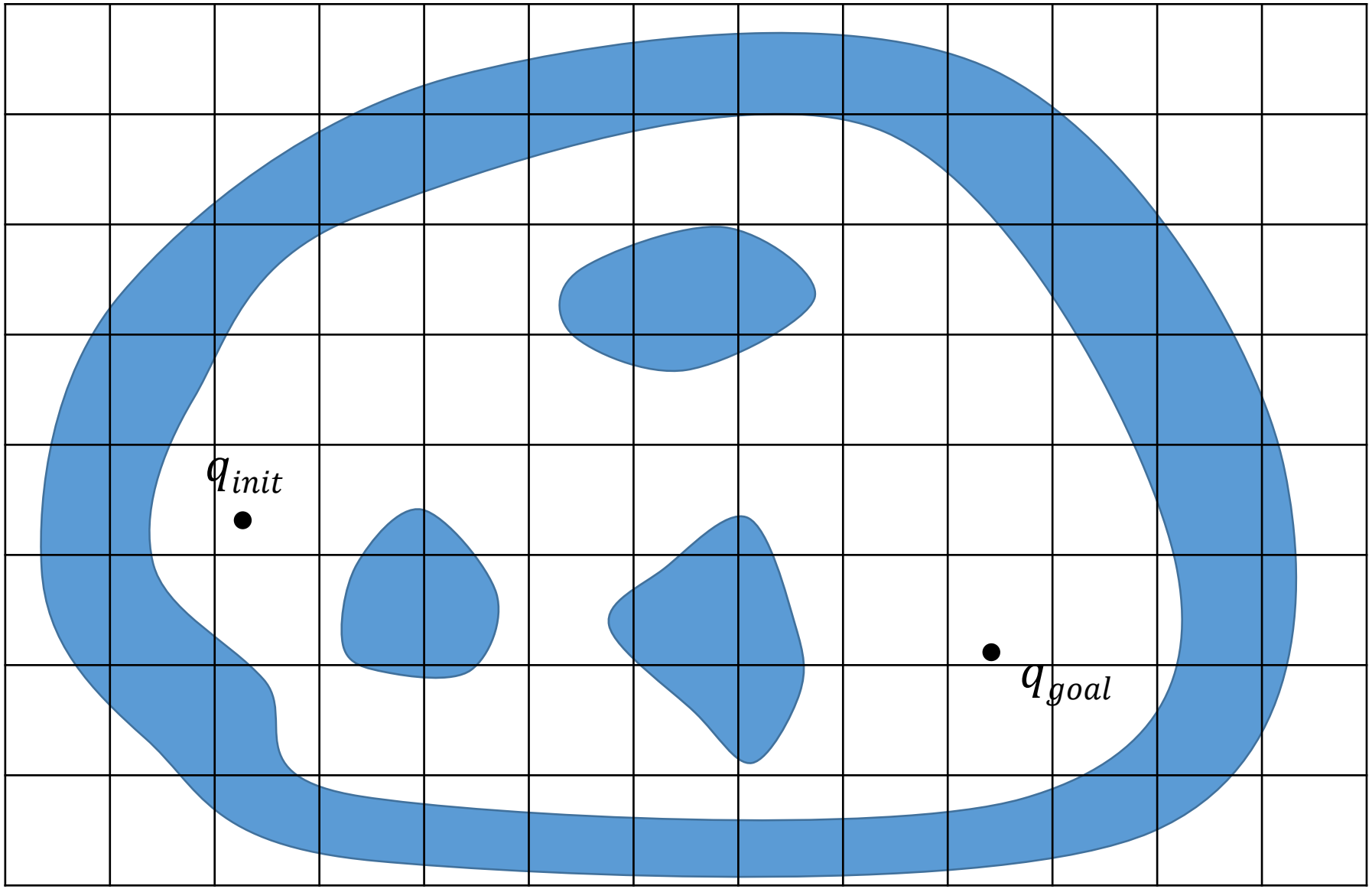
Reduced visibility roadmaps



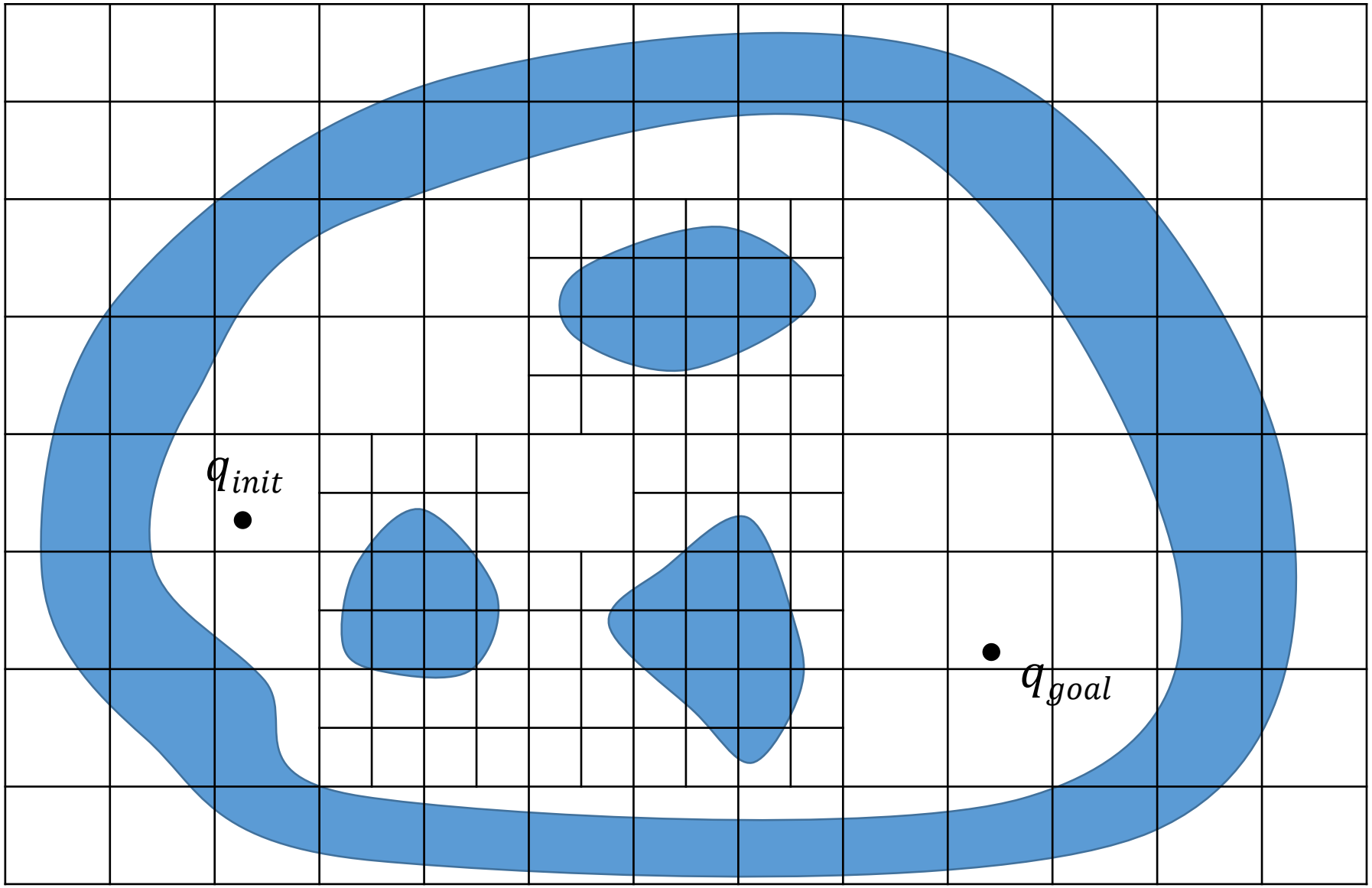
Approximate cell decomposition



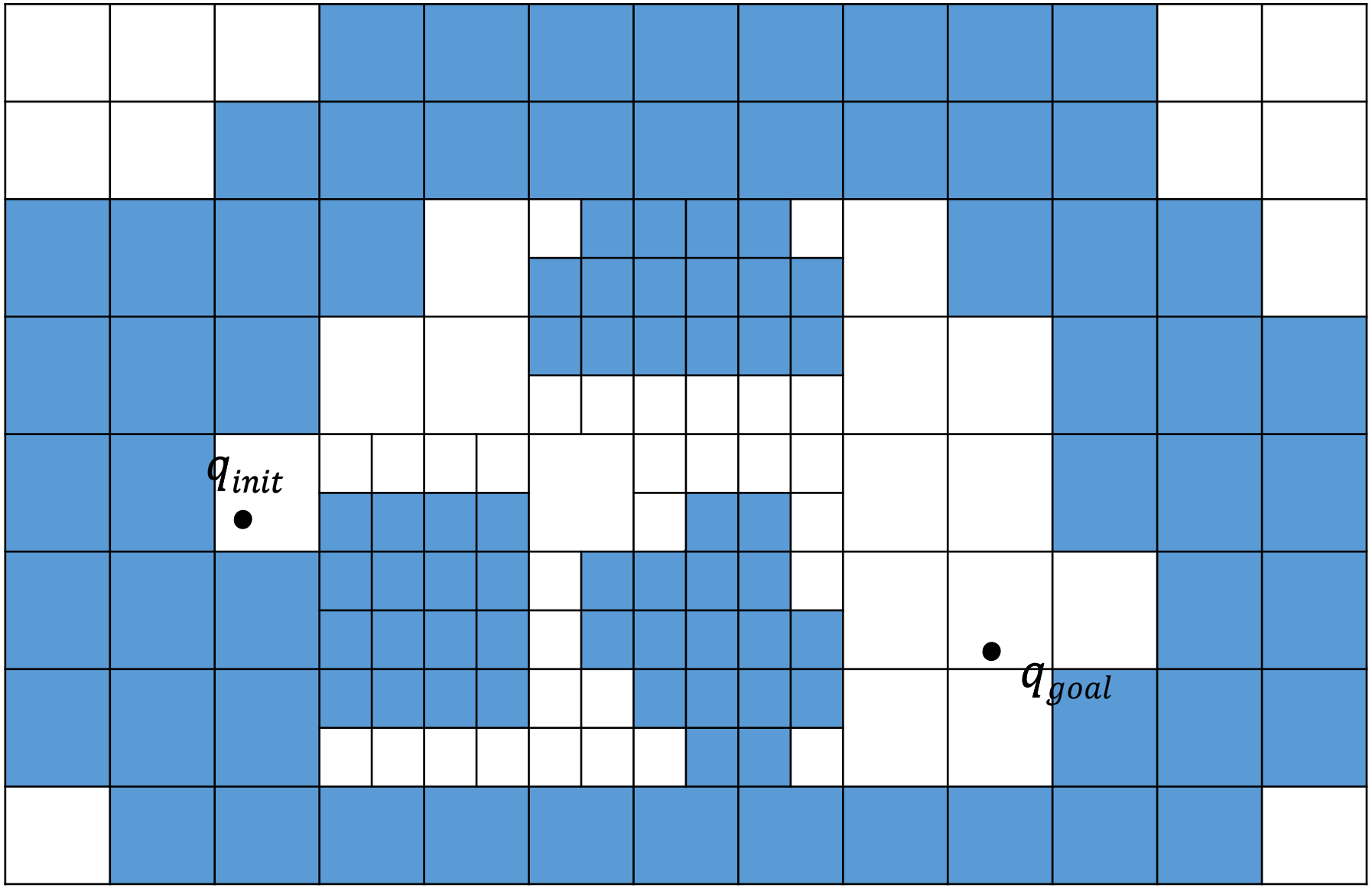
Approximate cell decomposition



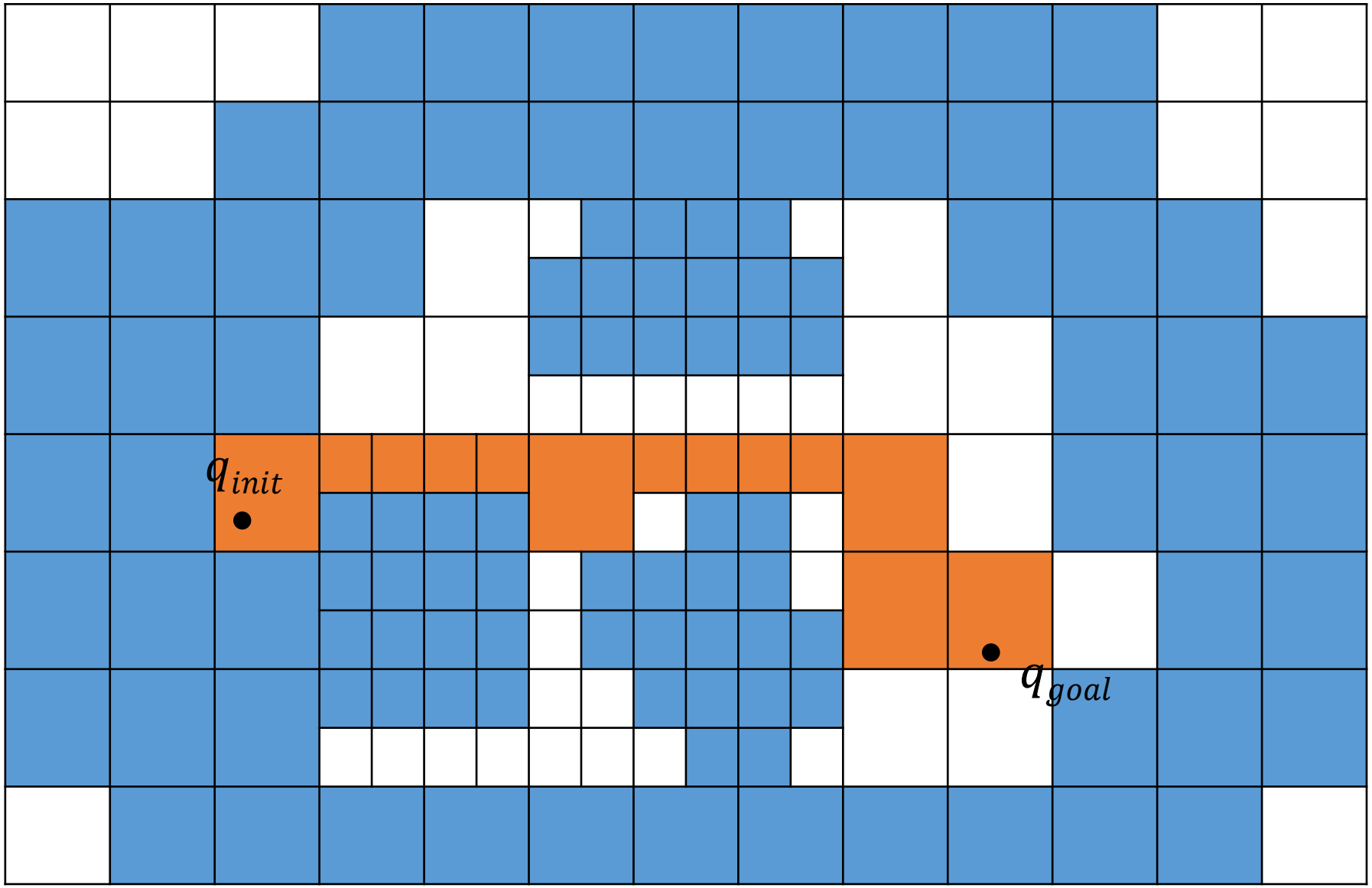
Approximate cell decomposition



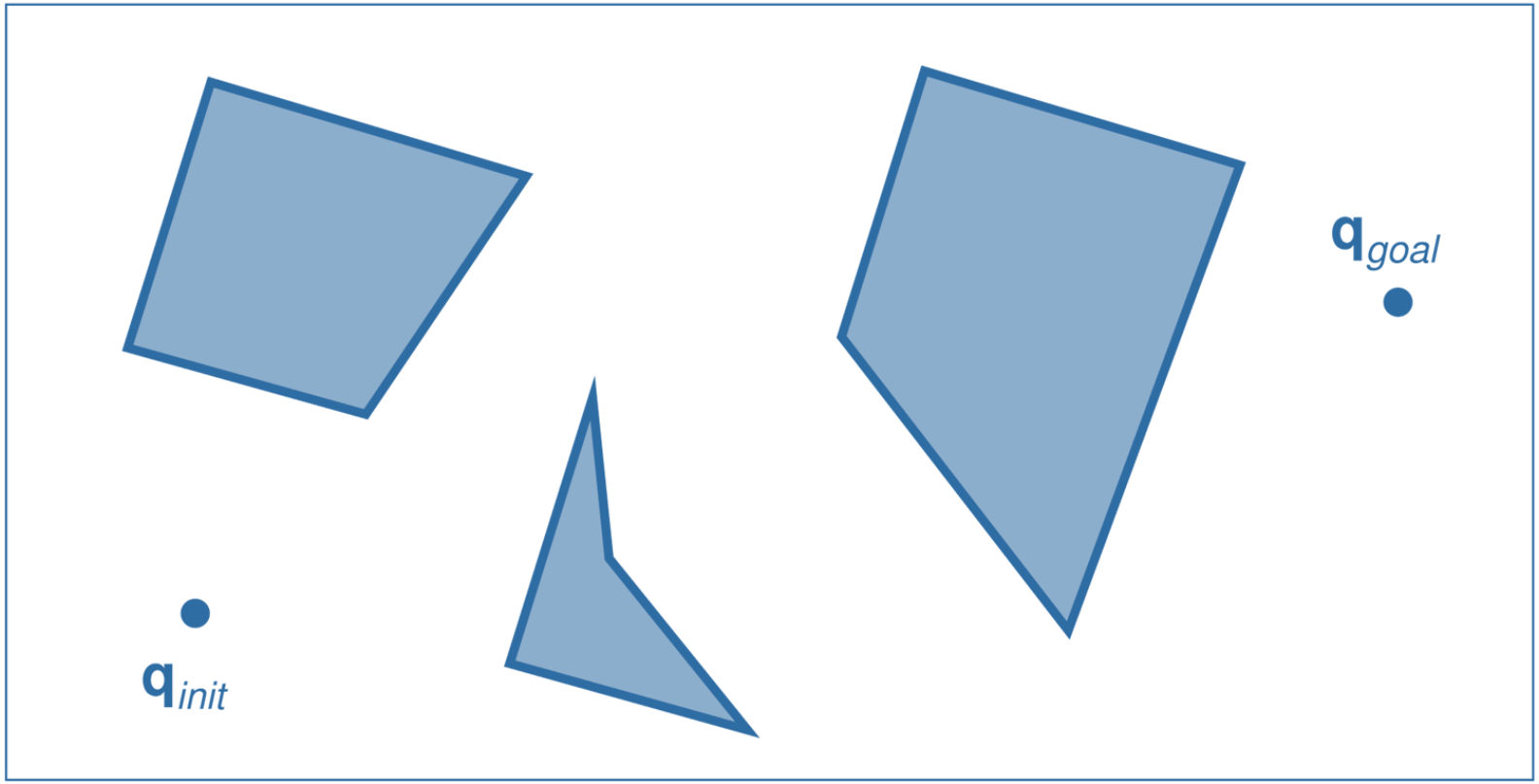
Approximate cell decomposition



Approximate cell decomposition

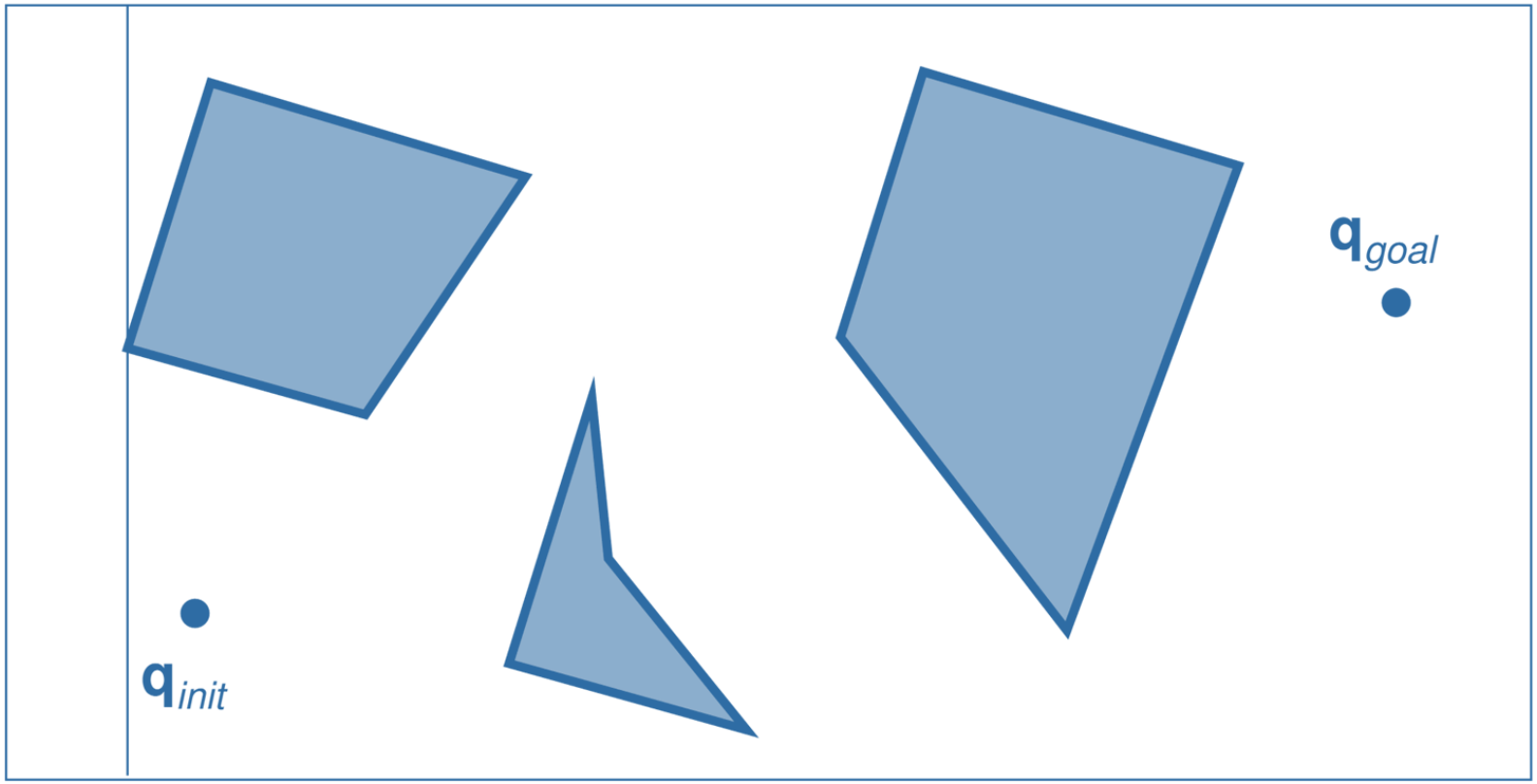


Vertical (exact) cell decomposition

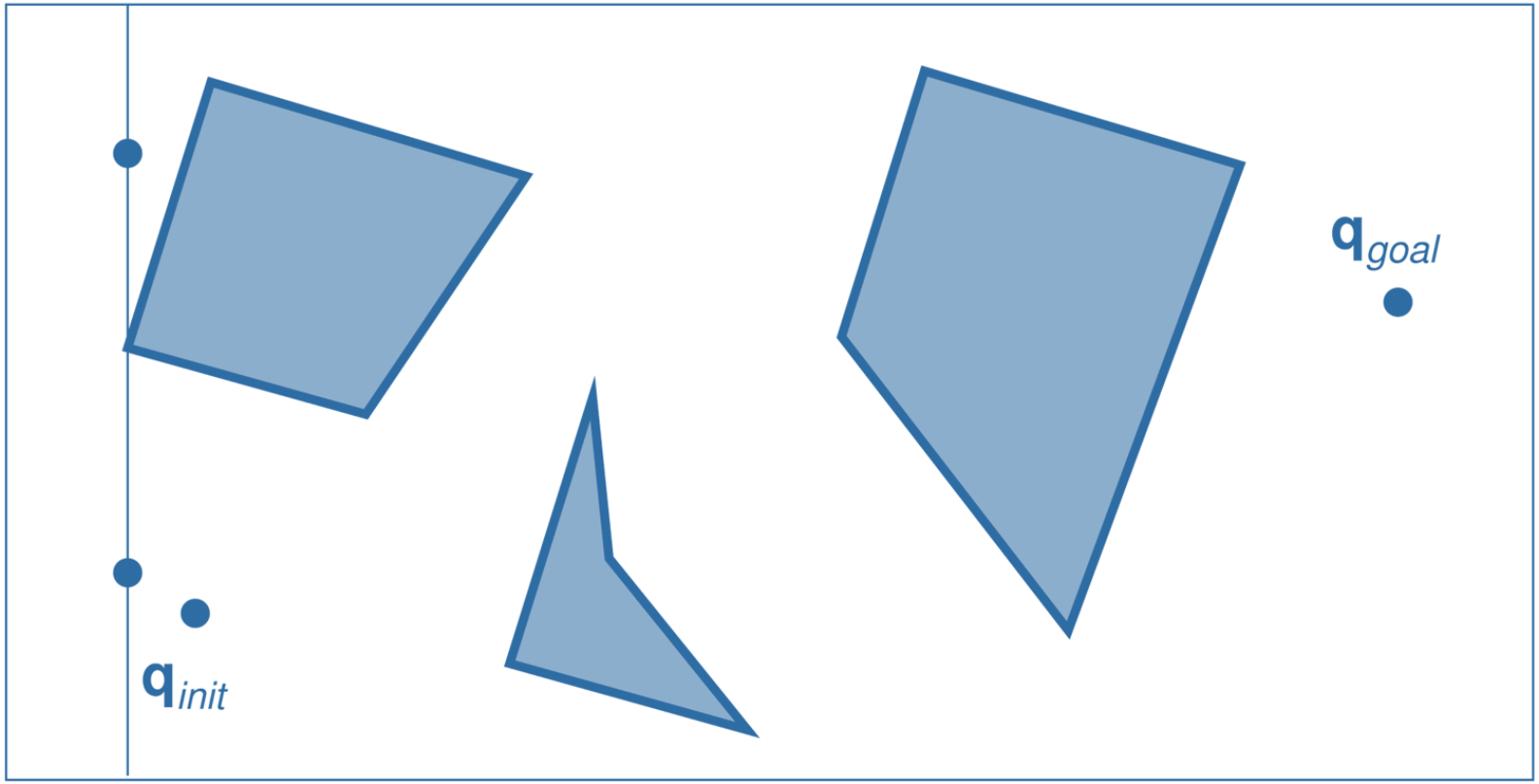


Illustrations de F. Lamiraux

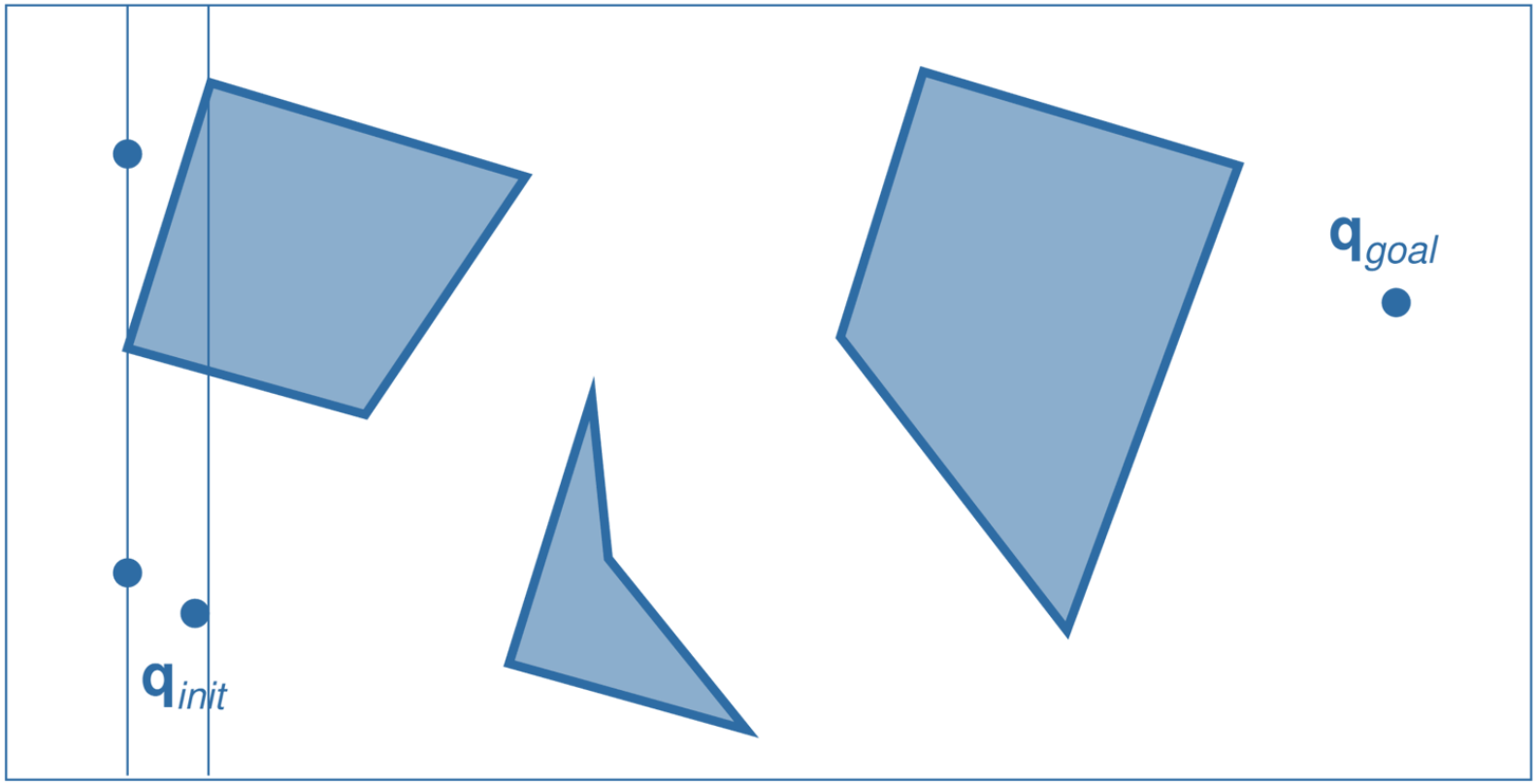
Vertical (exact) cell decomposition



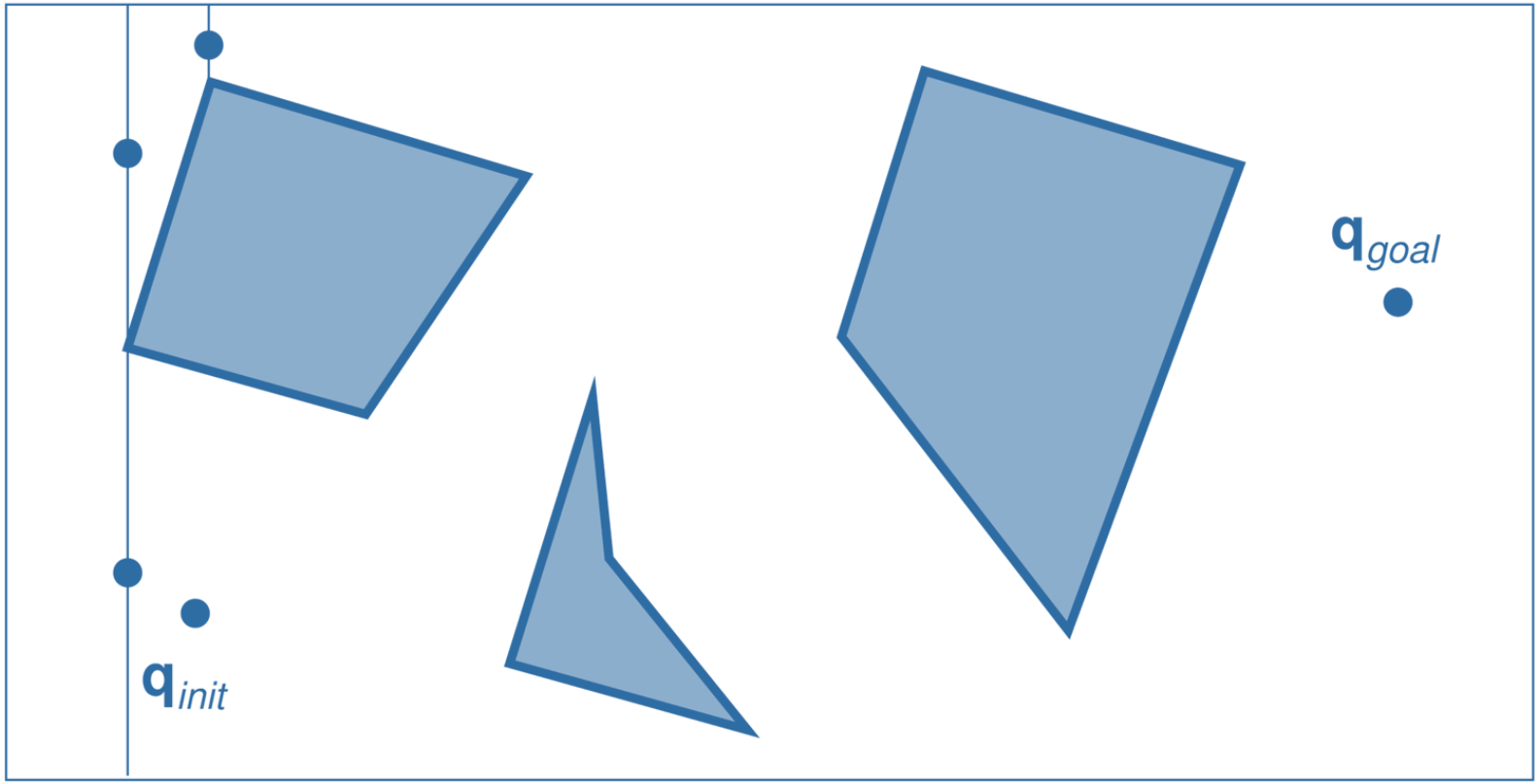
Vertical (exact) cell decomposition



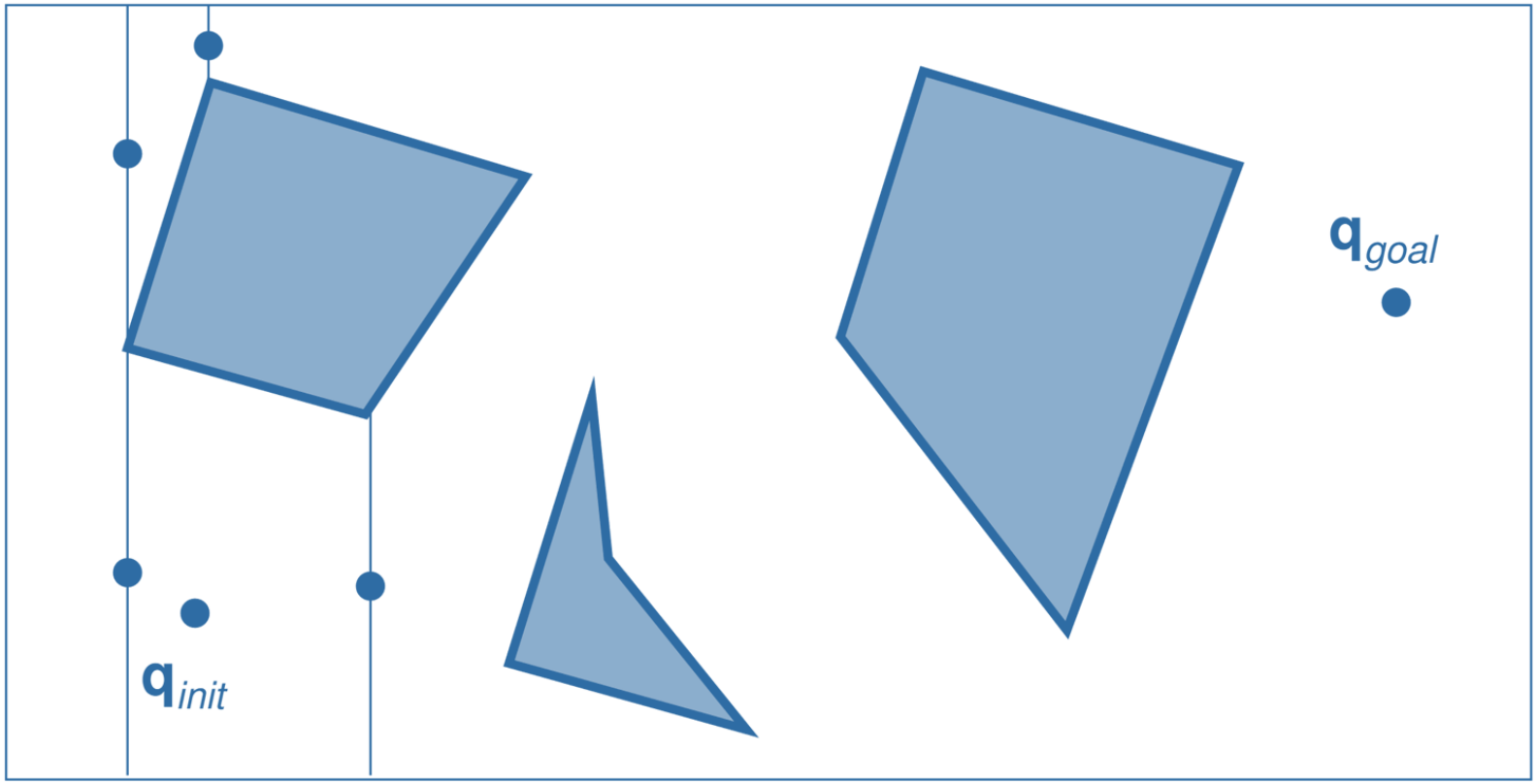
Vertical (exact) cell decomposition



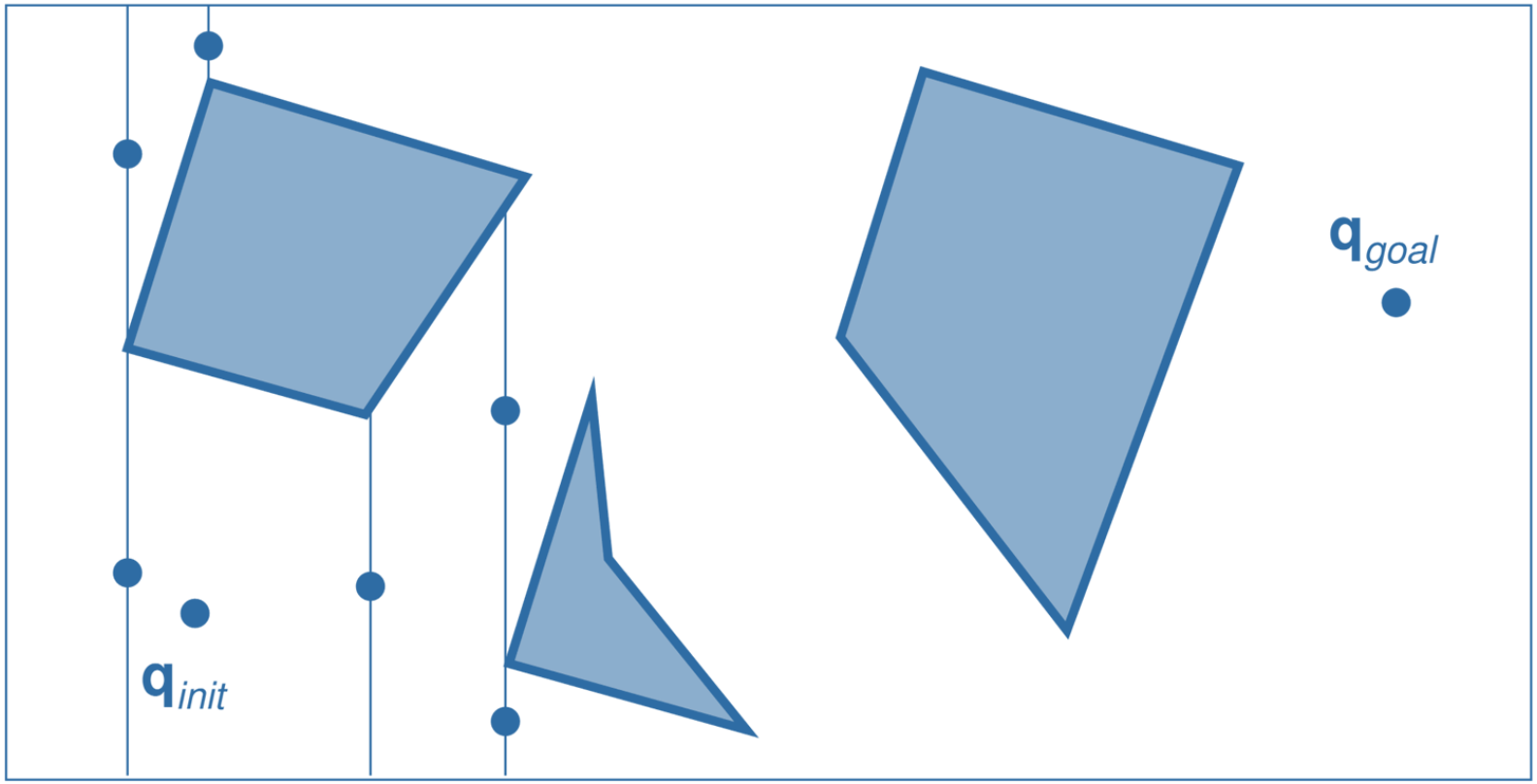
Vertical (exact) cell decomposition



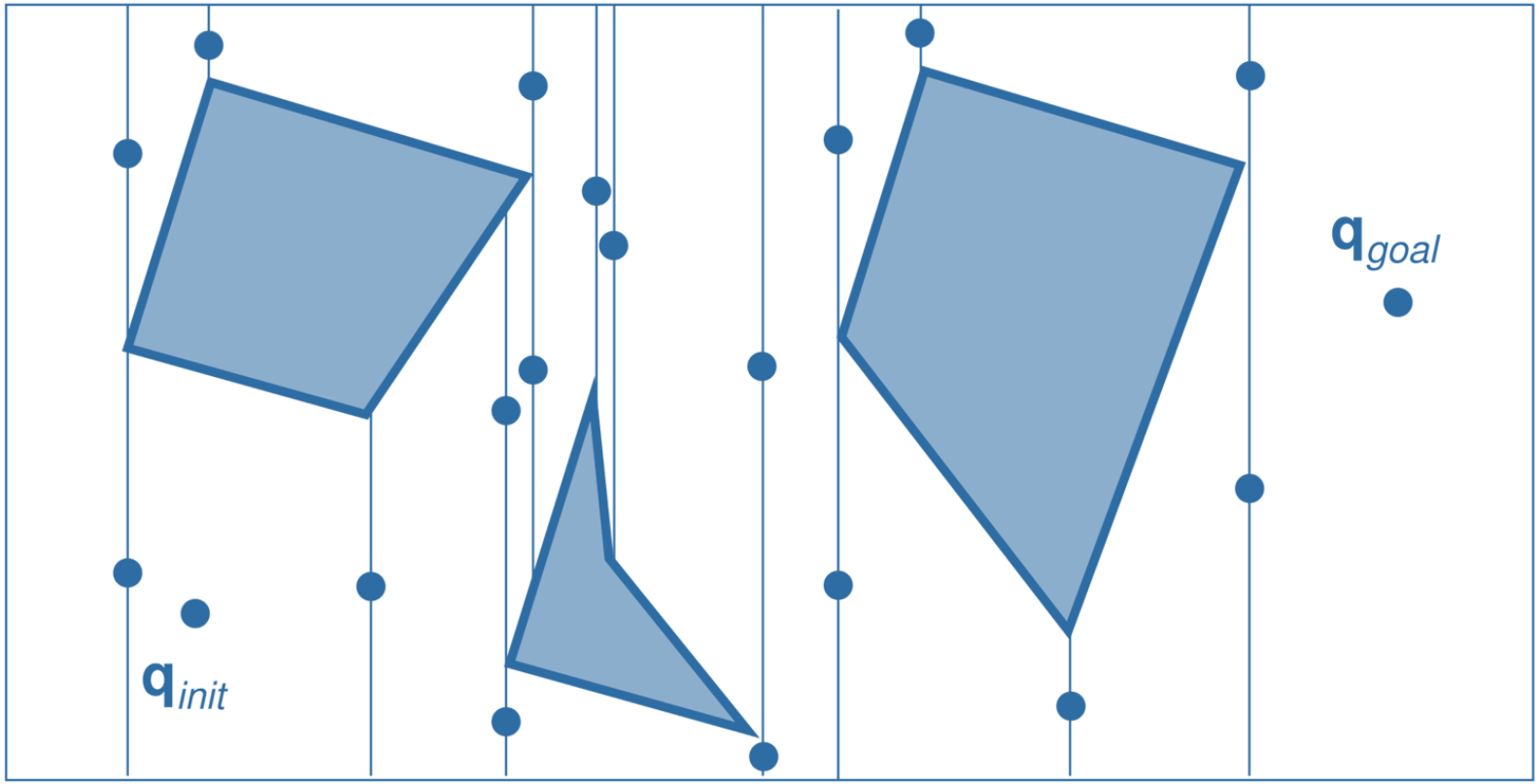
Vertical (exact) cell decomposition



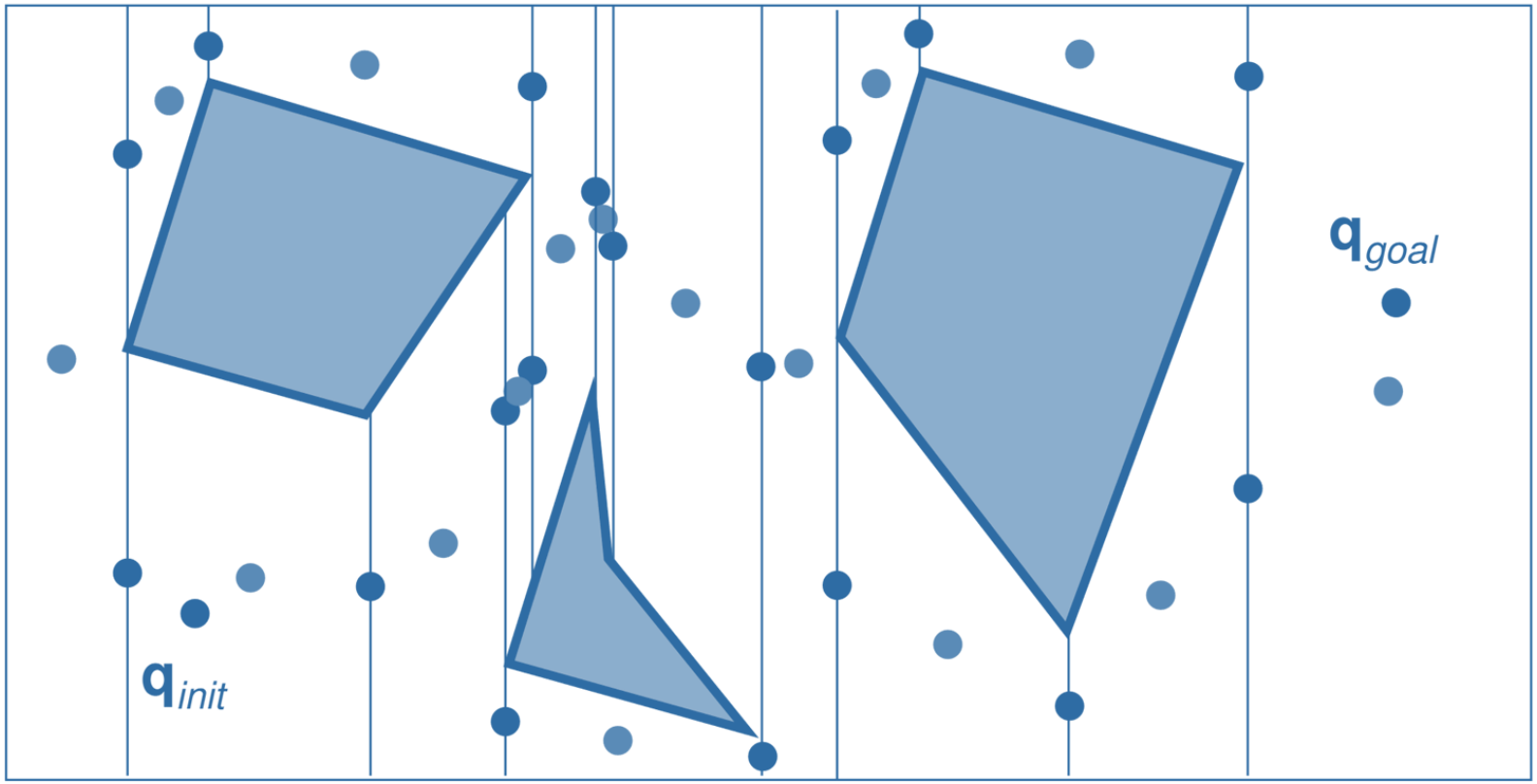
Vertical (exact) cell decomposition



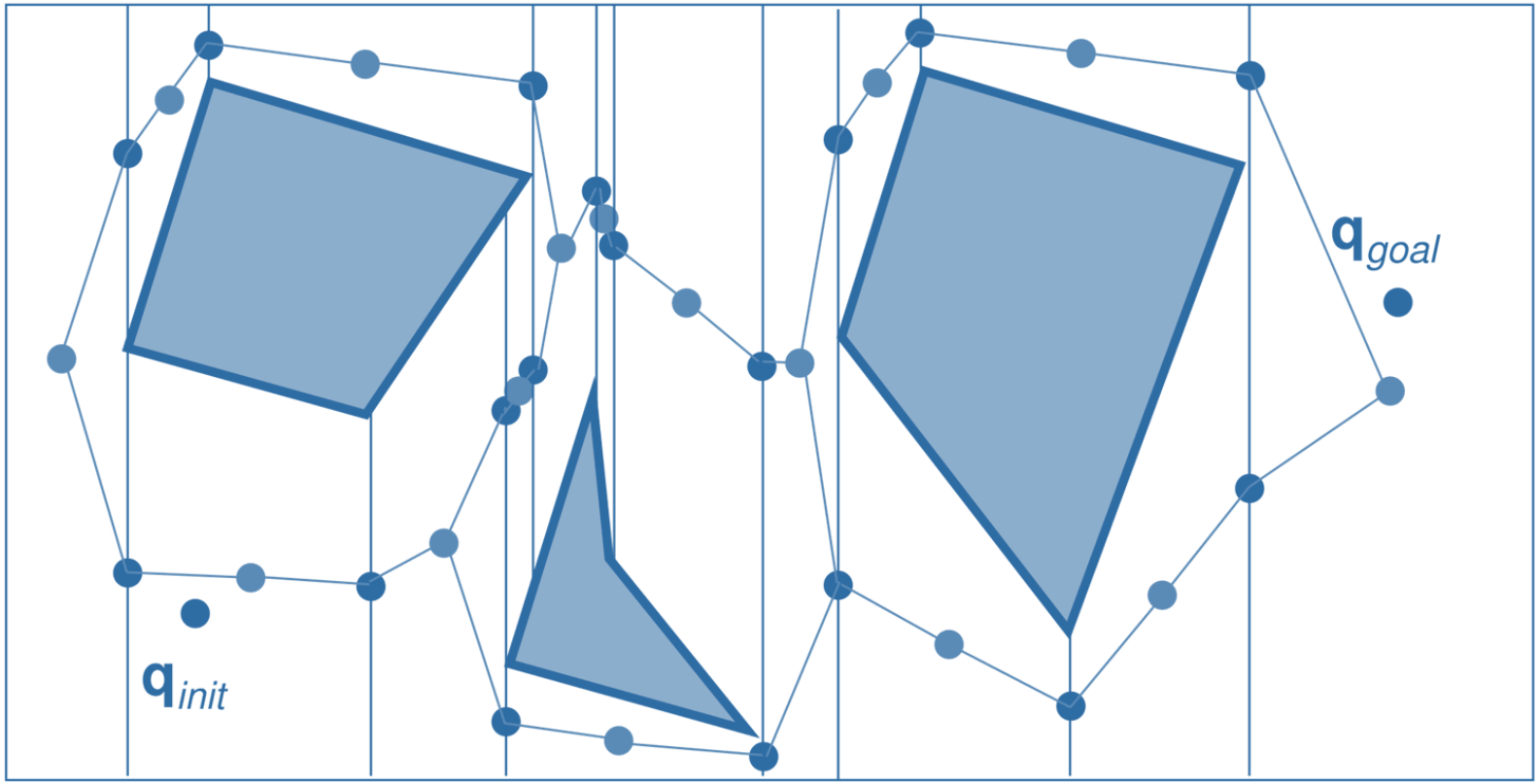
Vertical (exact) cell decomposition



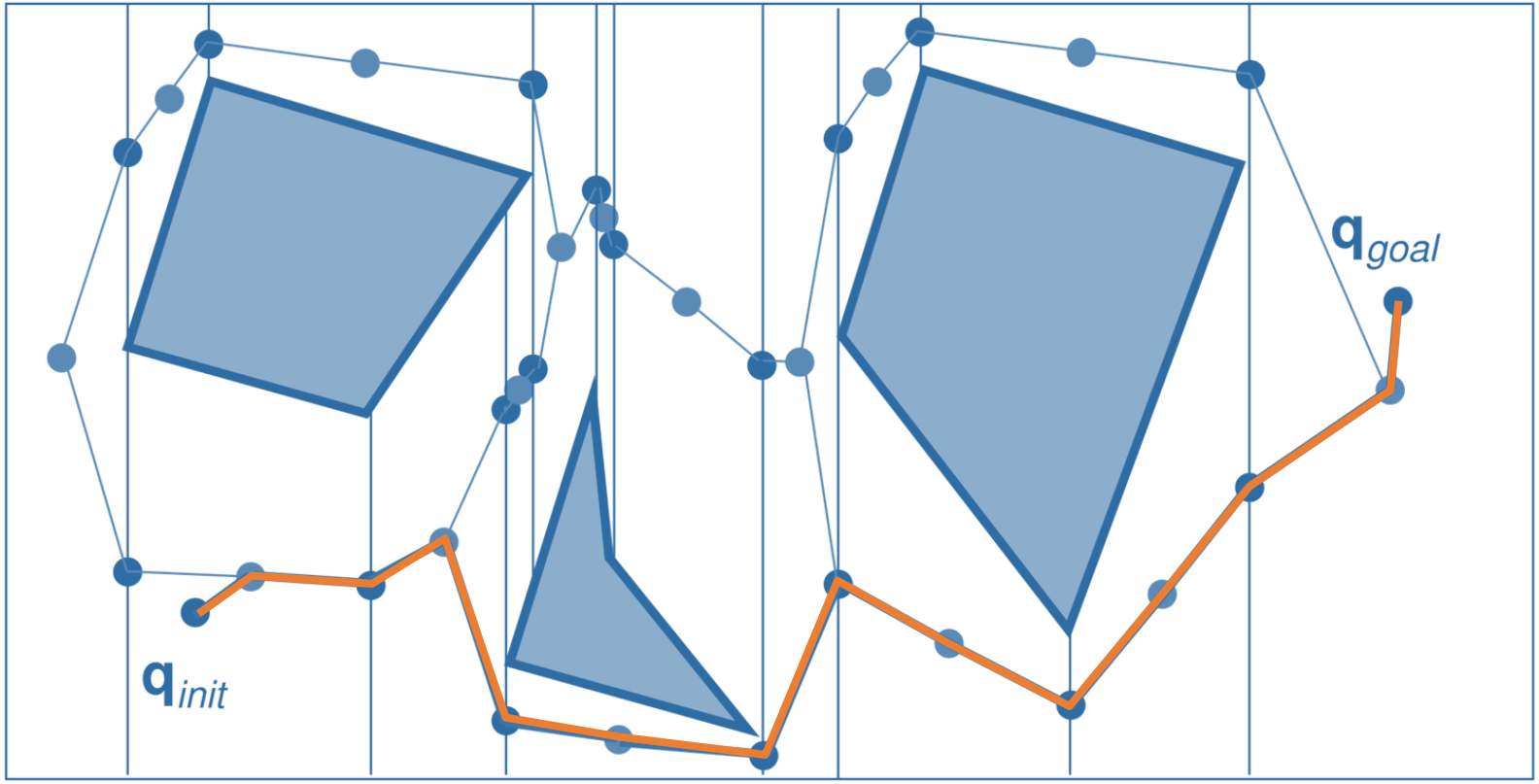
Vertical (exact) cell decomposition



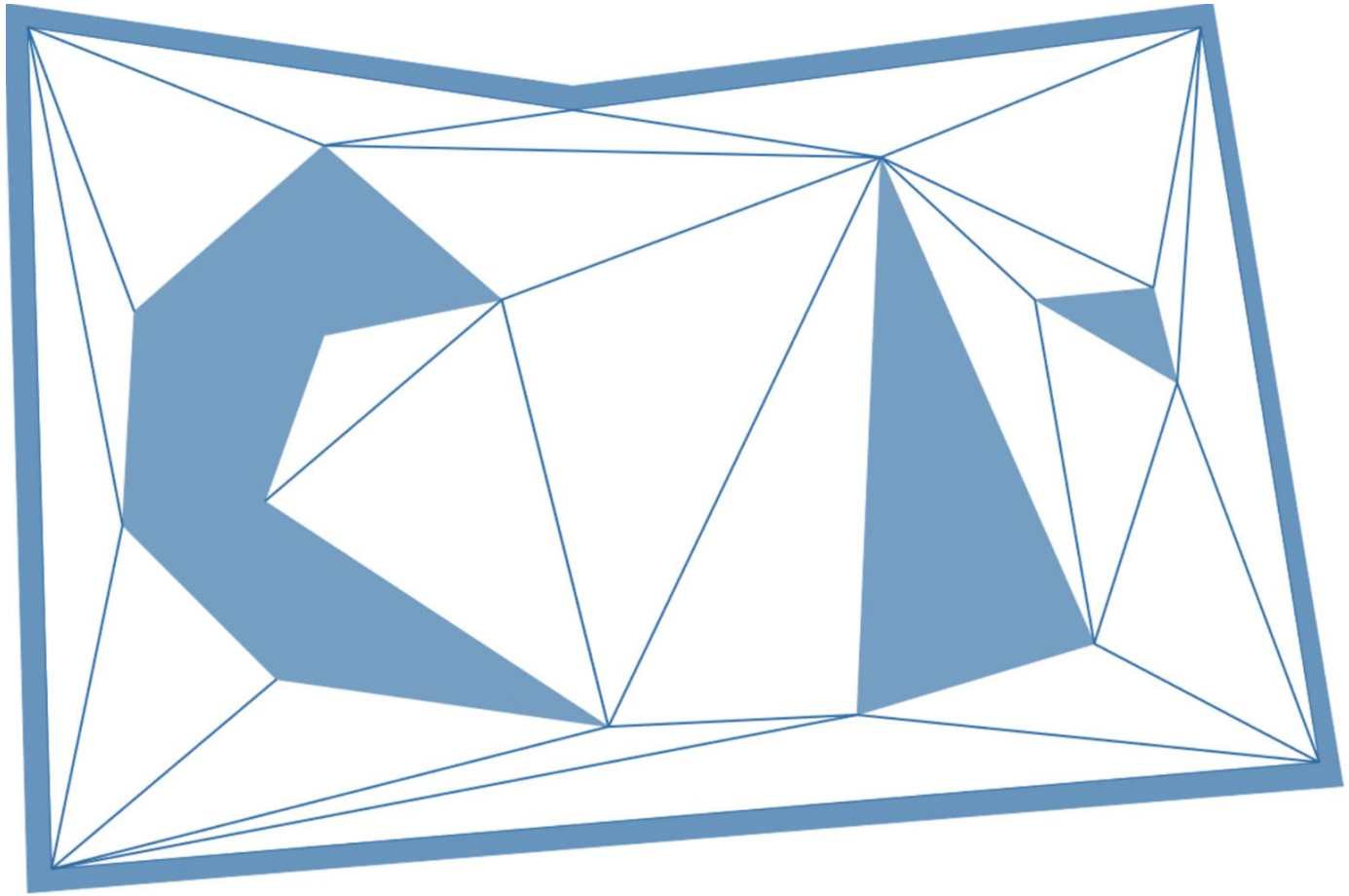
Vertical (exact) cell decomposition



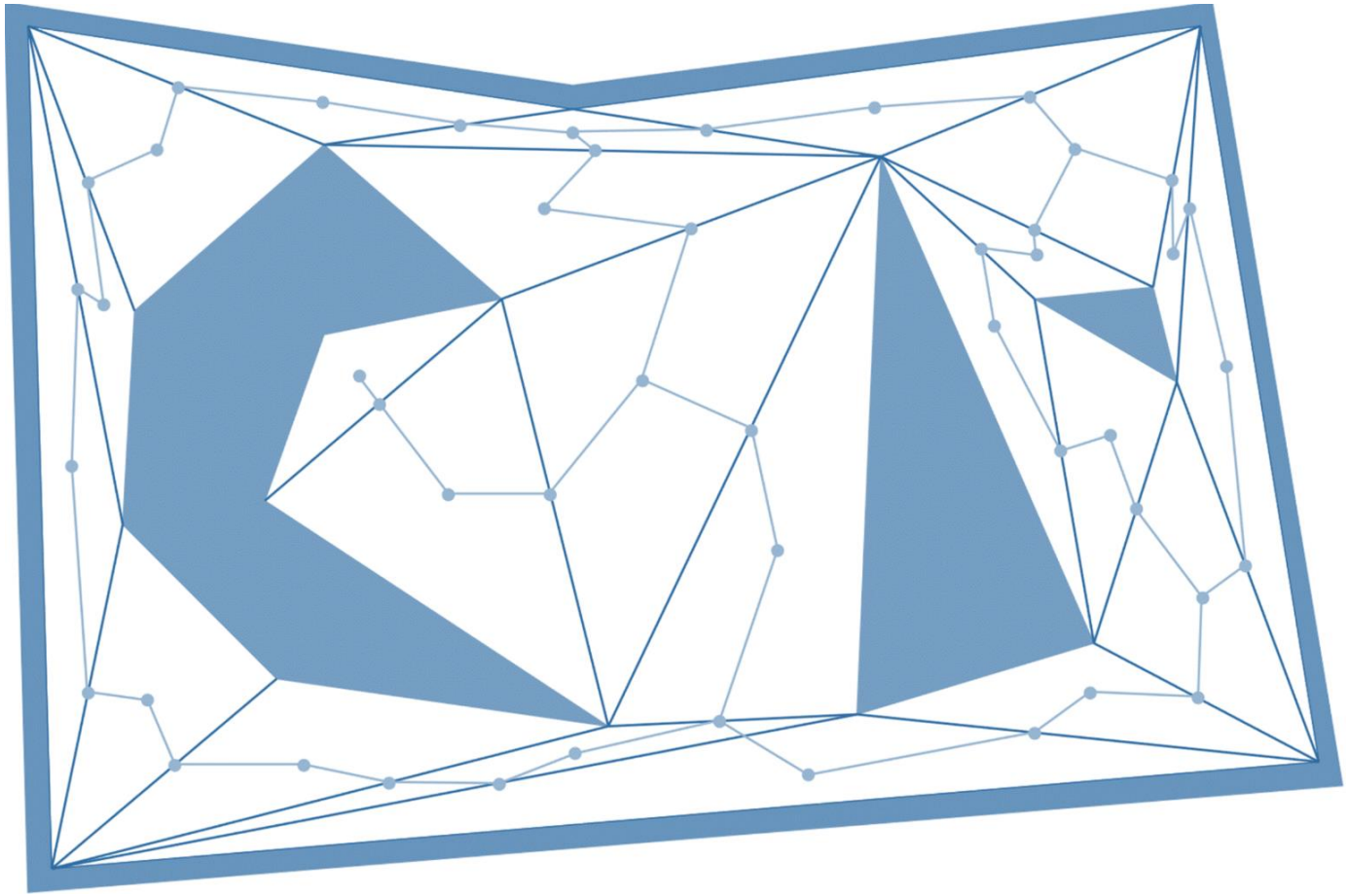
Vertical (exact) cell decomposition



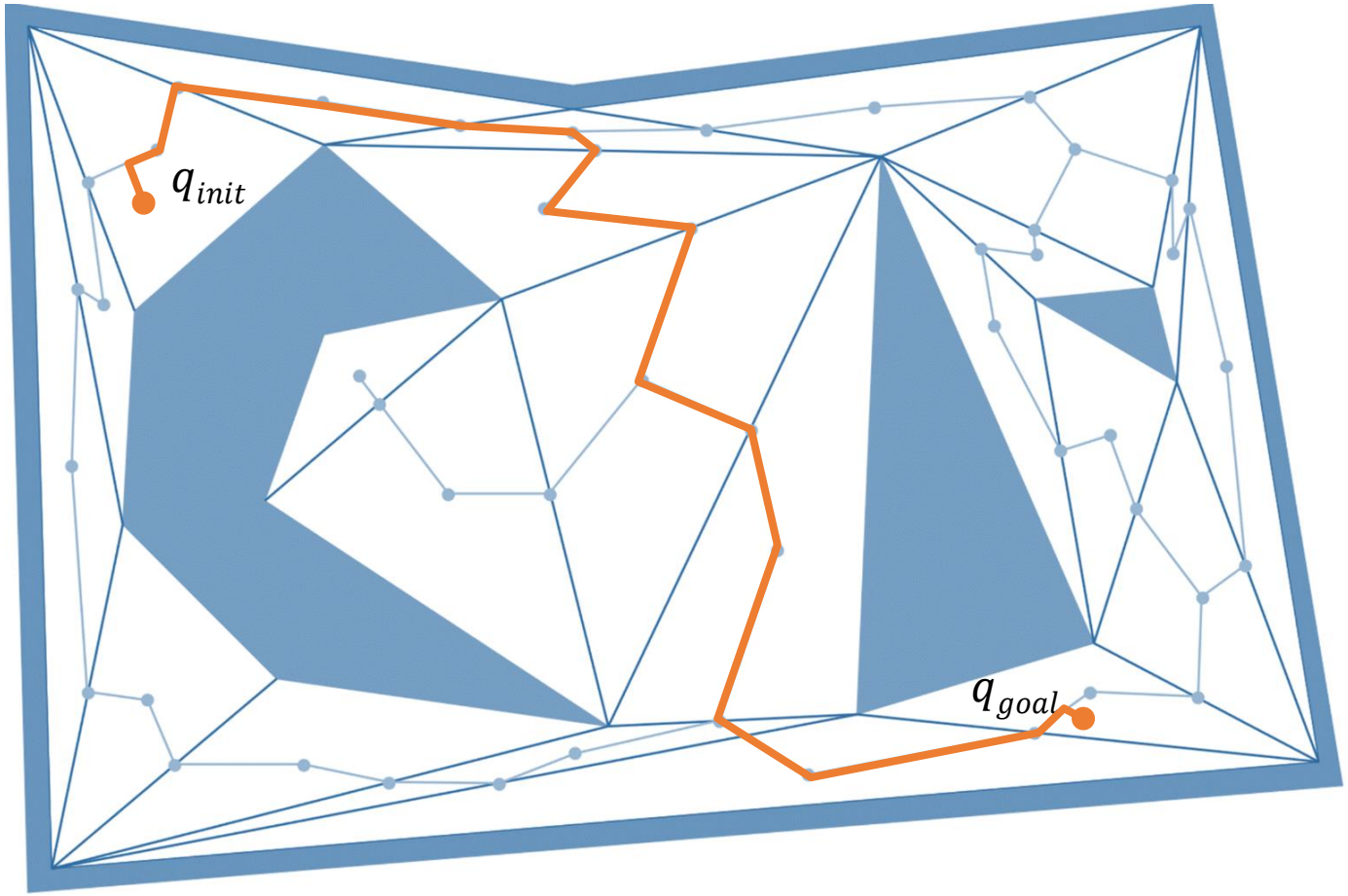
Triangulation



Triangulation



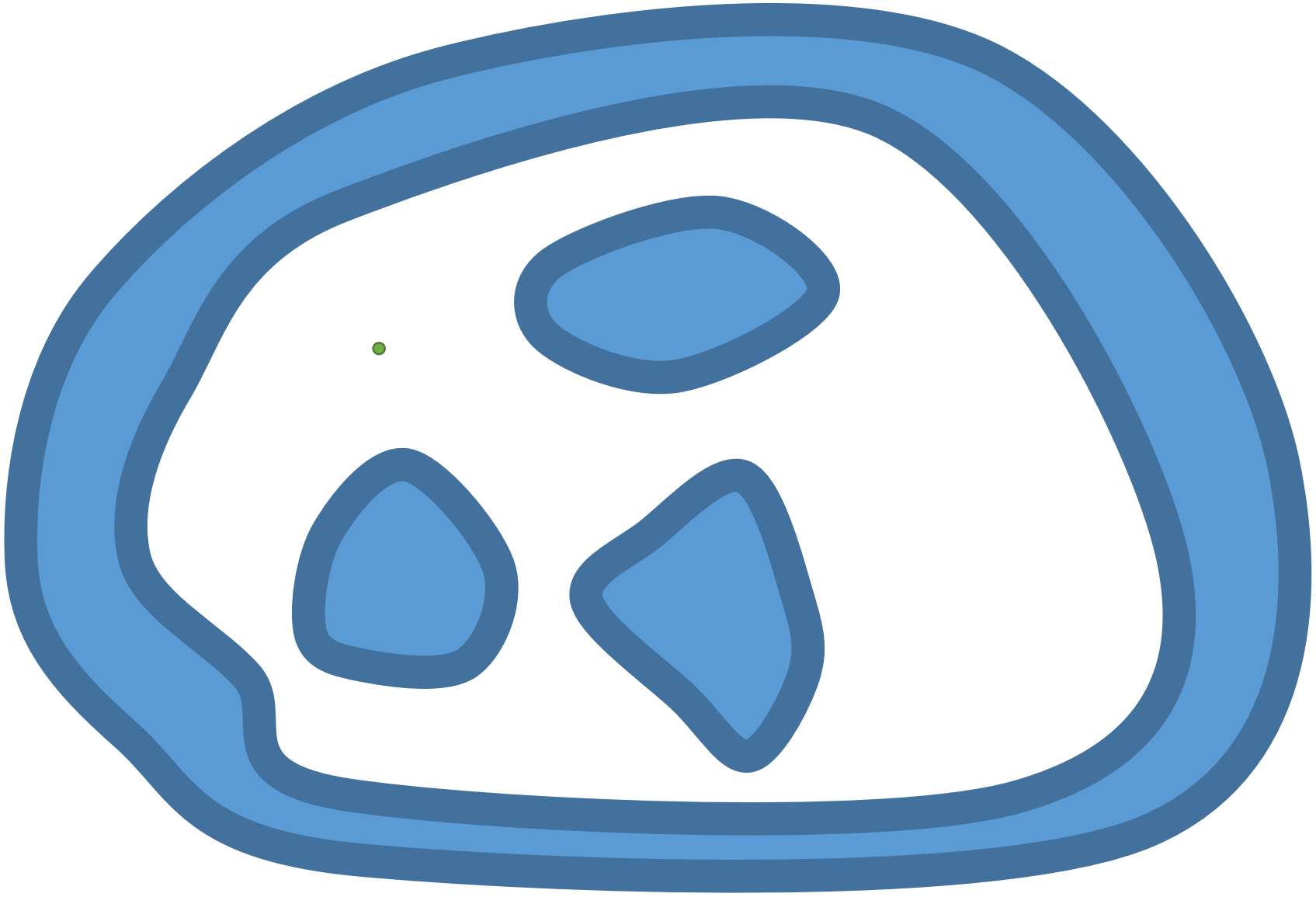
Triangulation



De la planification de chemin à la planification de mouvement

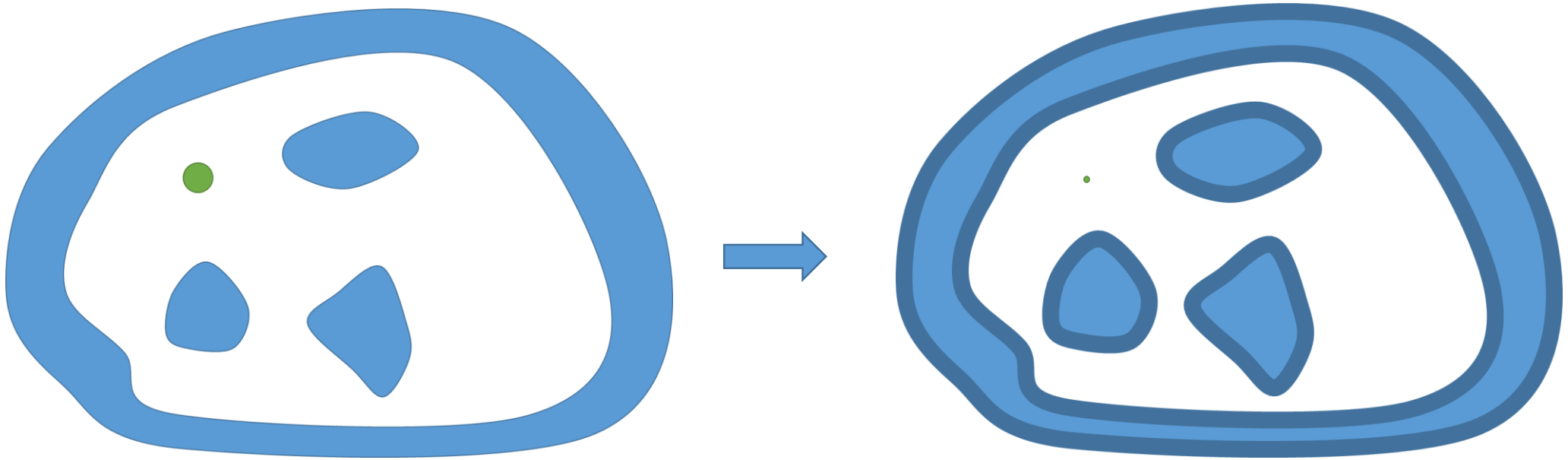
Si notre robot n’est plus un point

Si notre robot n’est plus un point

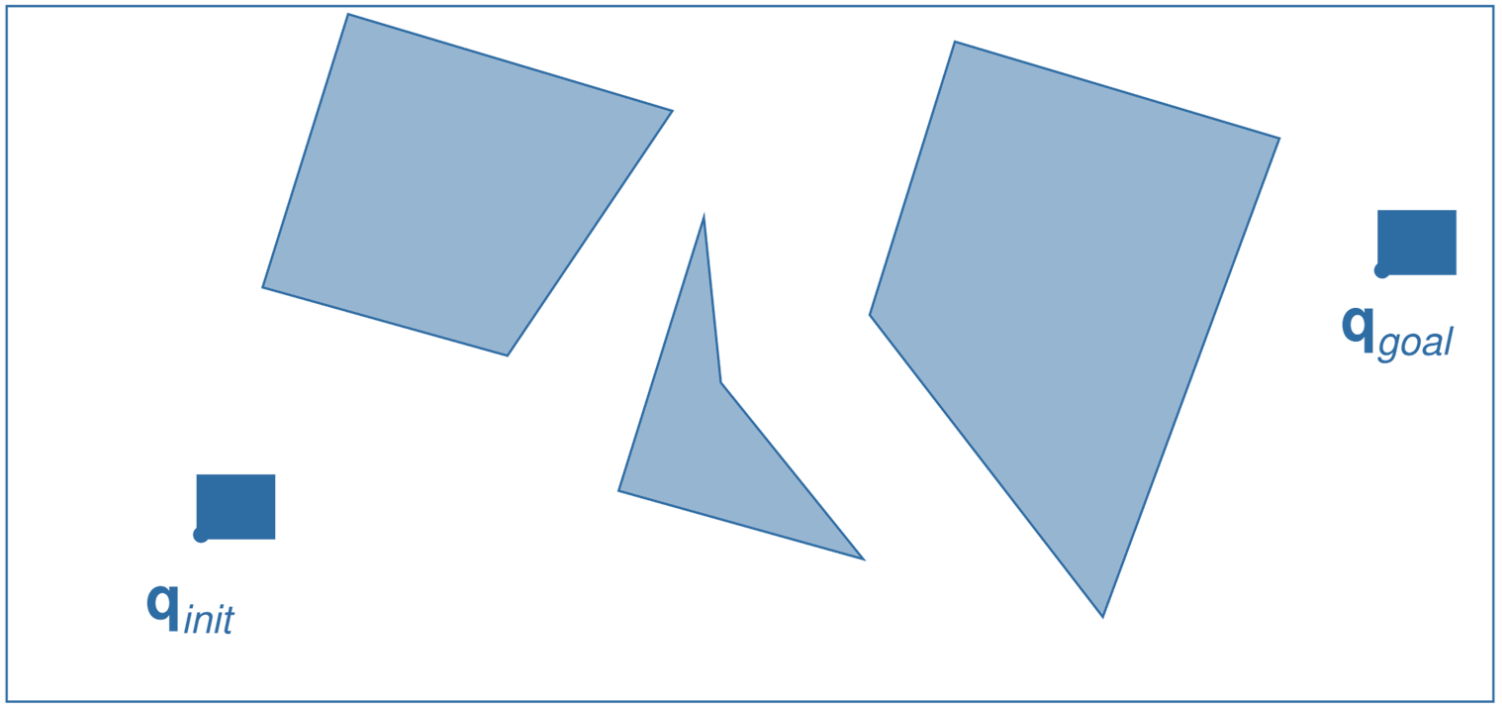


Si notre robot n’est plus un point

Planification de mouvement —réduction→ Planification de chemin

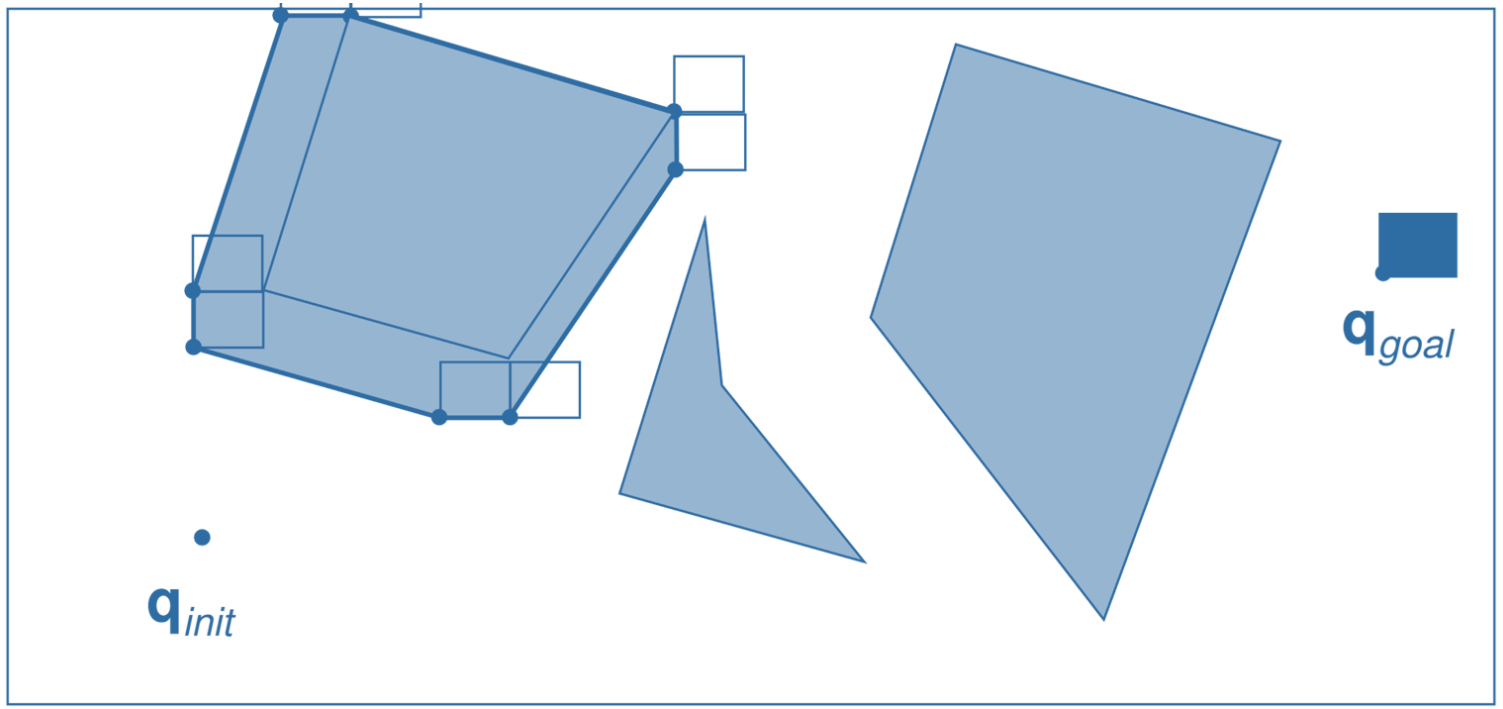


Robot rectangulaire qui ne peut que translater



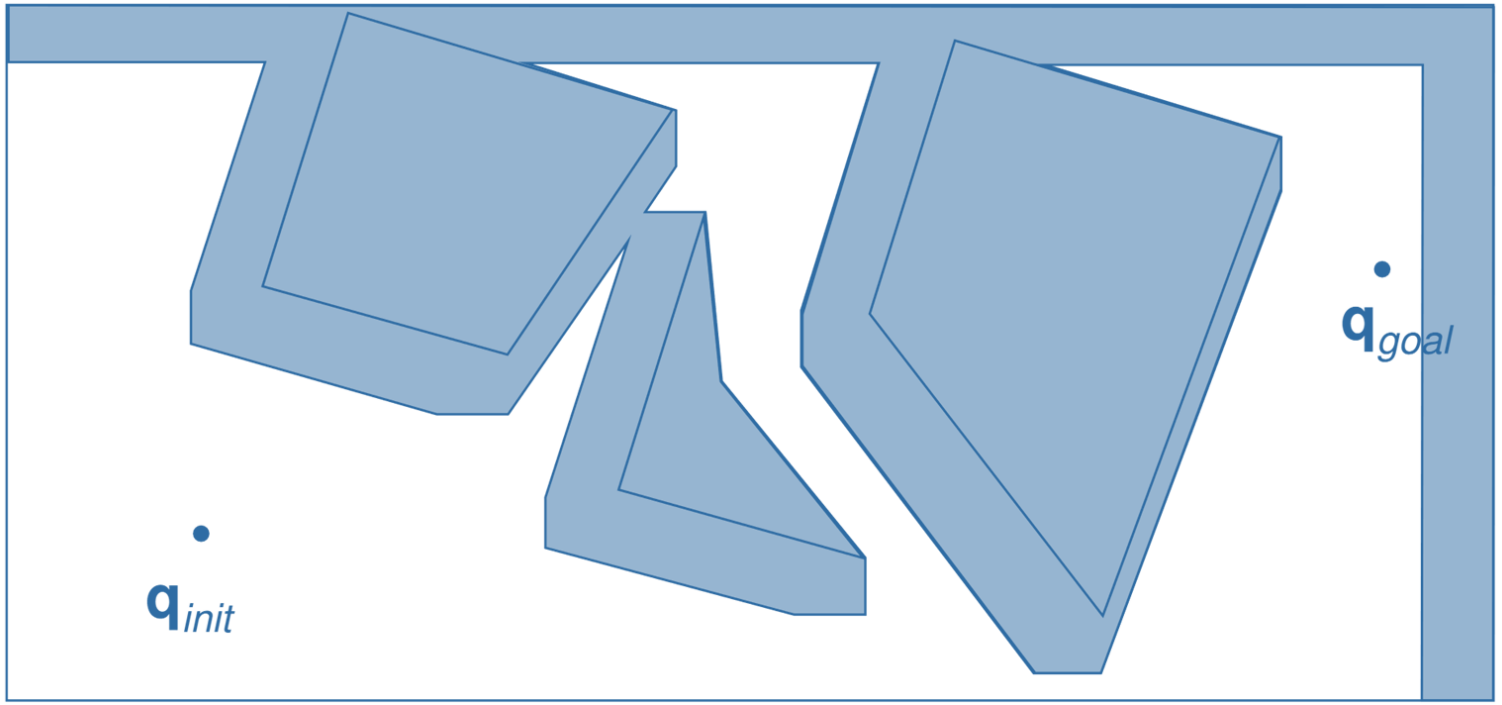
Illustrations de F. Lamiraux

Robot rectangulaire qui ne peut que translater

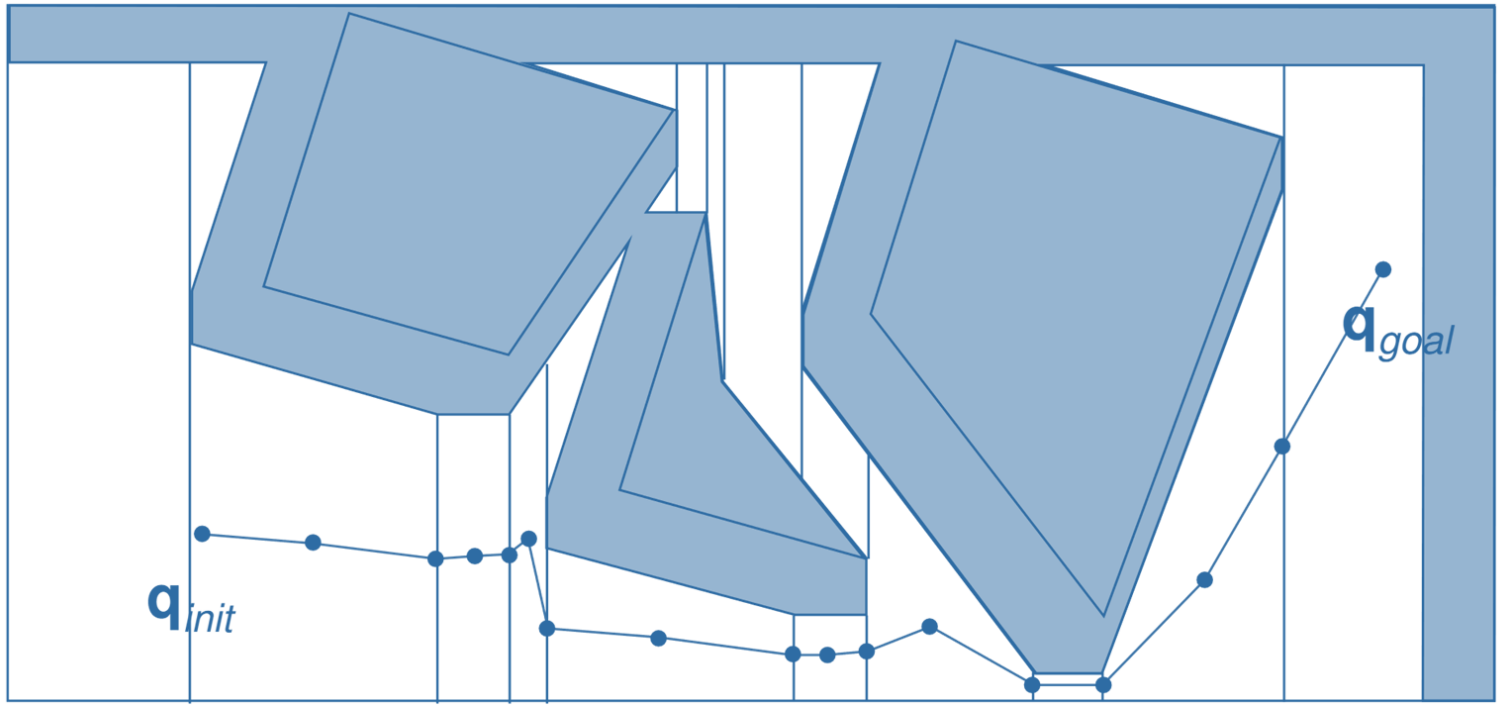


Robot rectangulaire qui ne peut que translater

Espace libre devient la difference de Minkowski entre l'espace libre et le robot

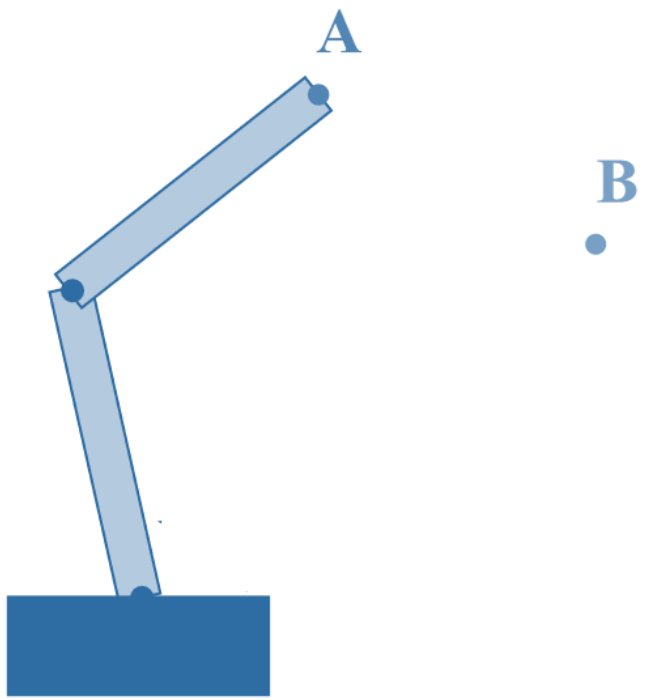


Robot rectangulaire qui ne peut que translater

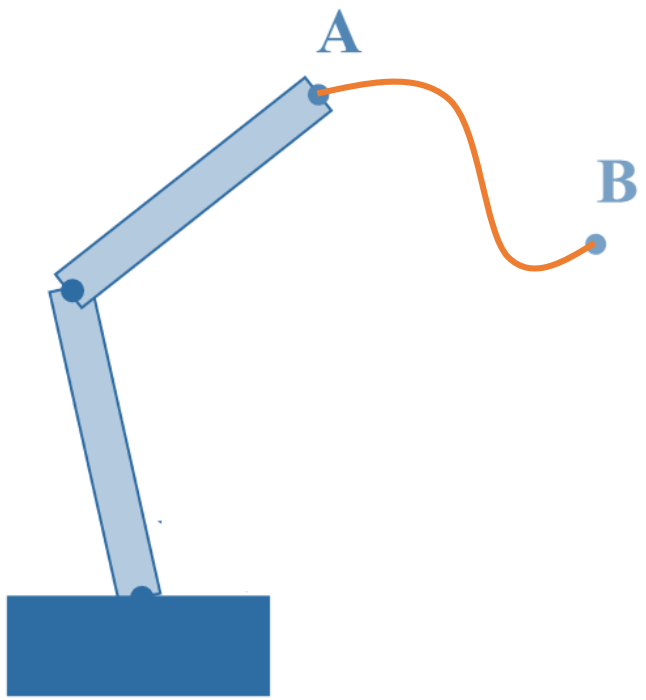


La planification de mouvement pour bras robotique (manipulator arm)

Exemple classique

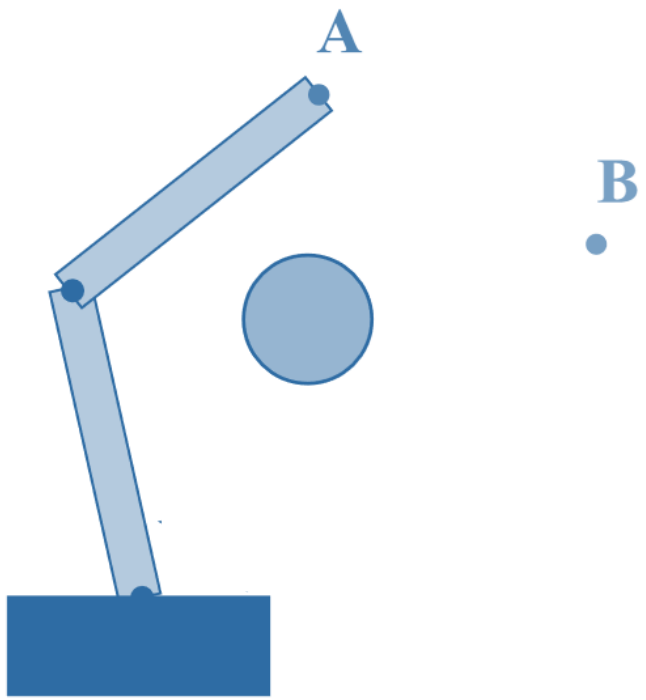


Exemple classique

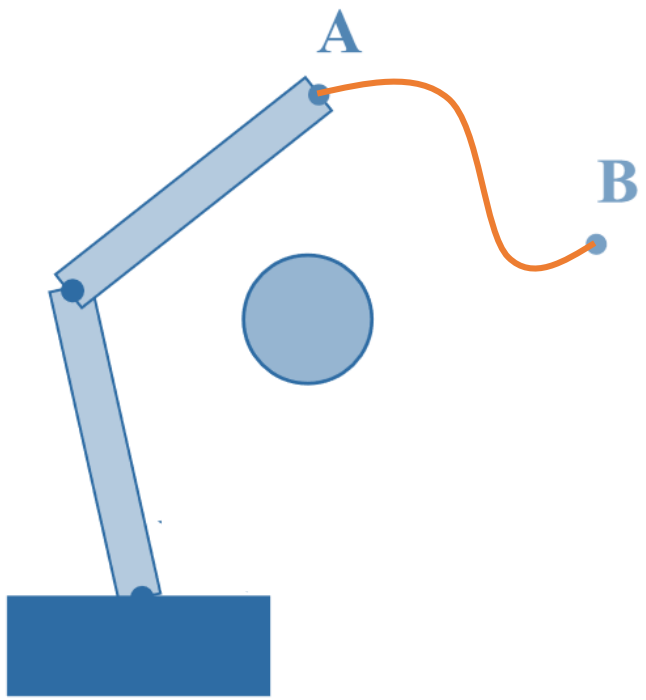


planification de chemin ?

Exemple classique

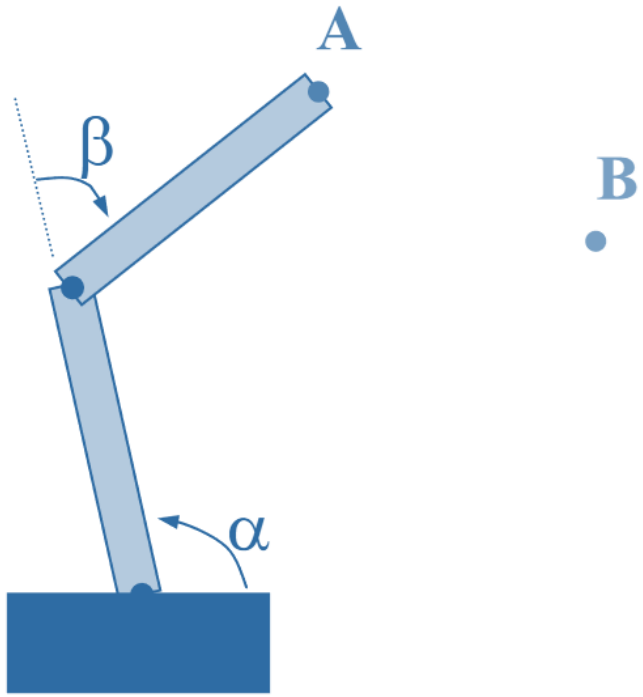


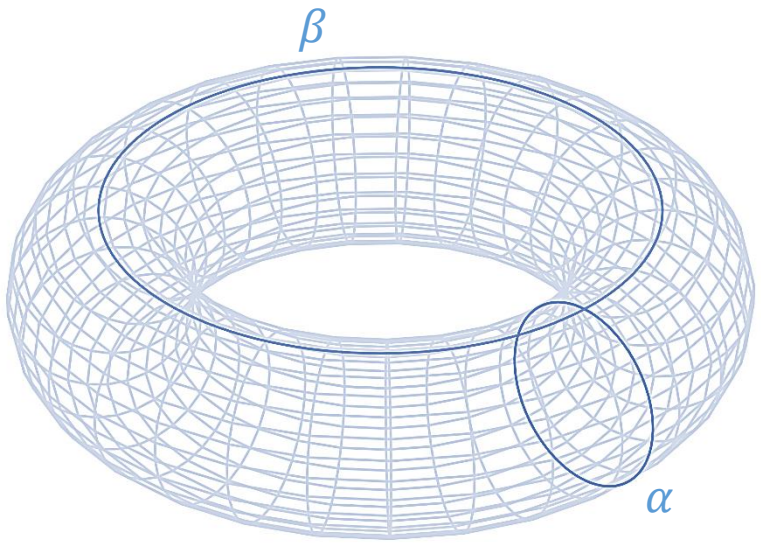
Exemple classique



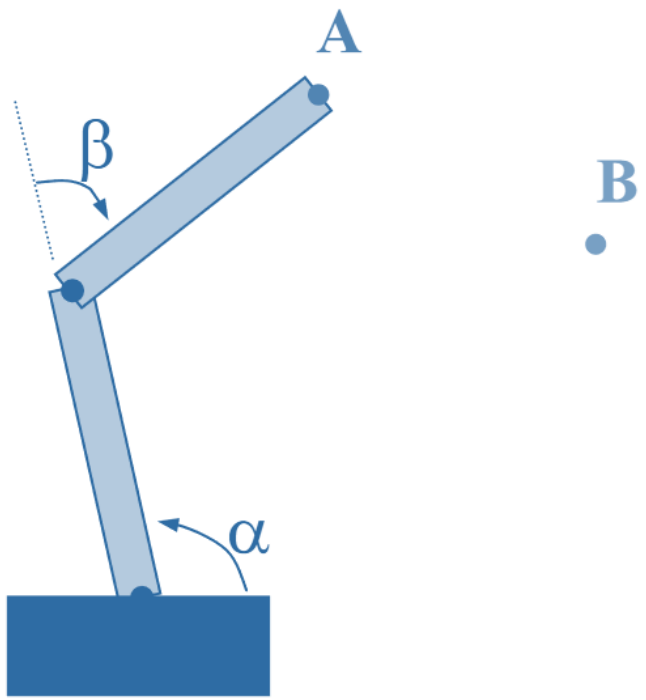
Une simple planification de chemin mènera à une solution infaisable

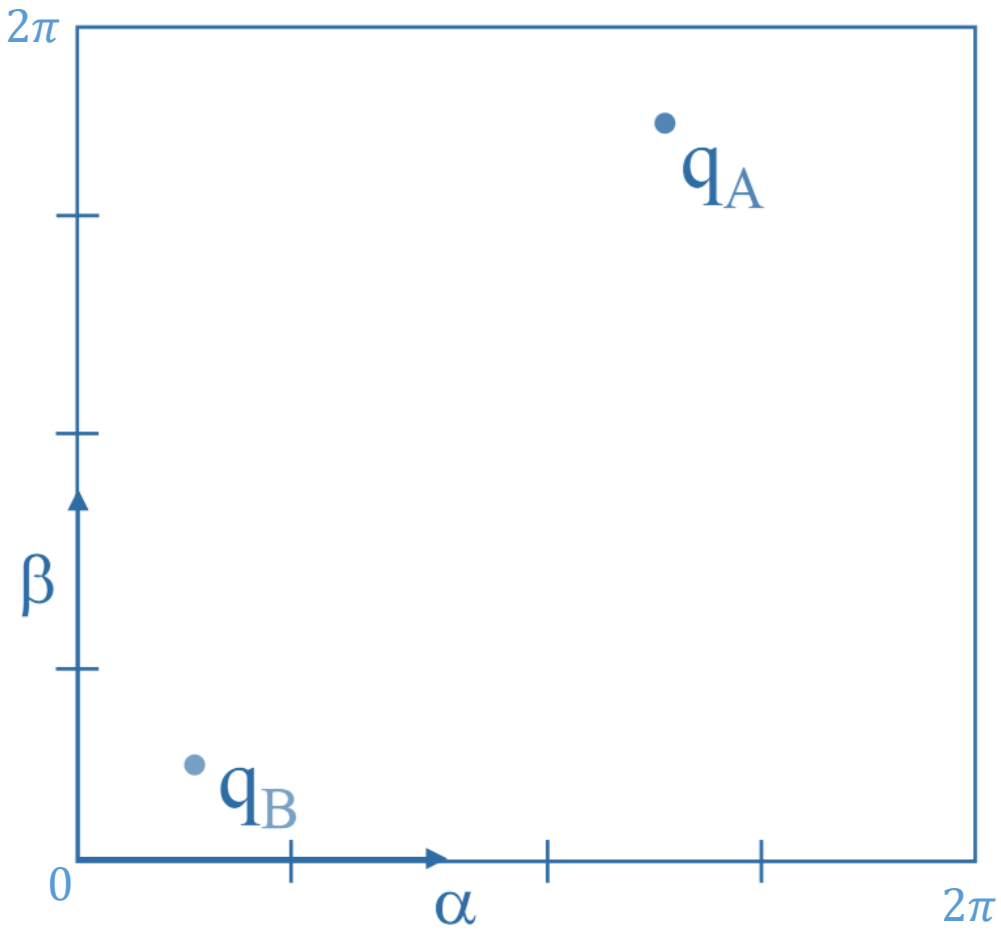
Exemple classique



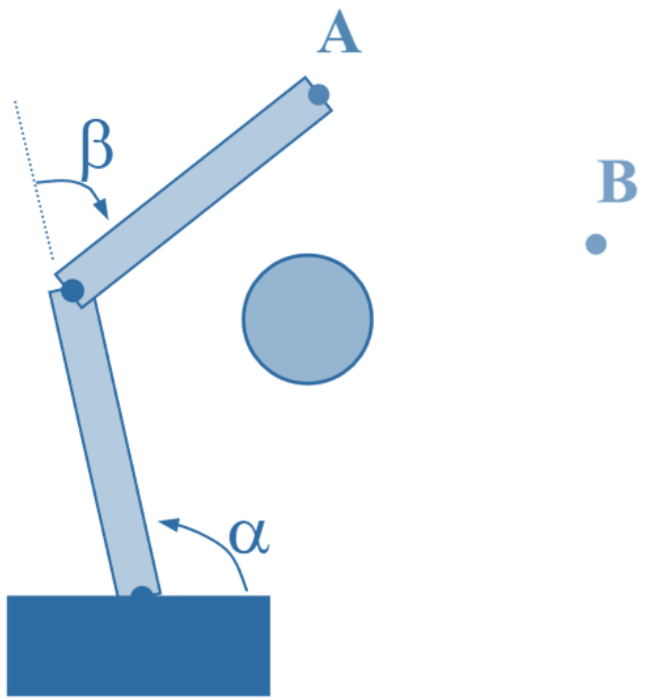


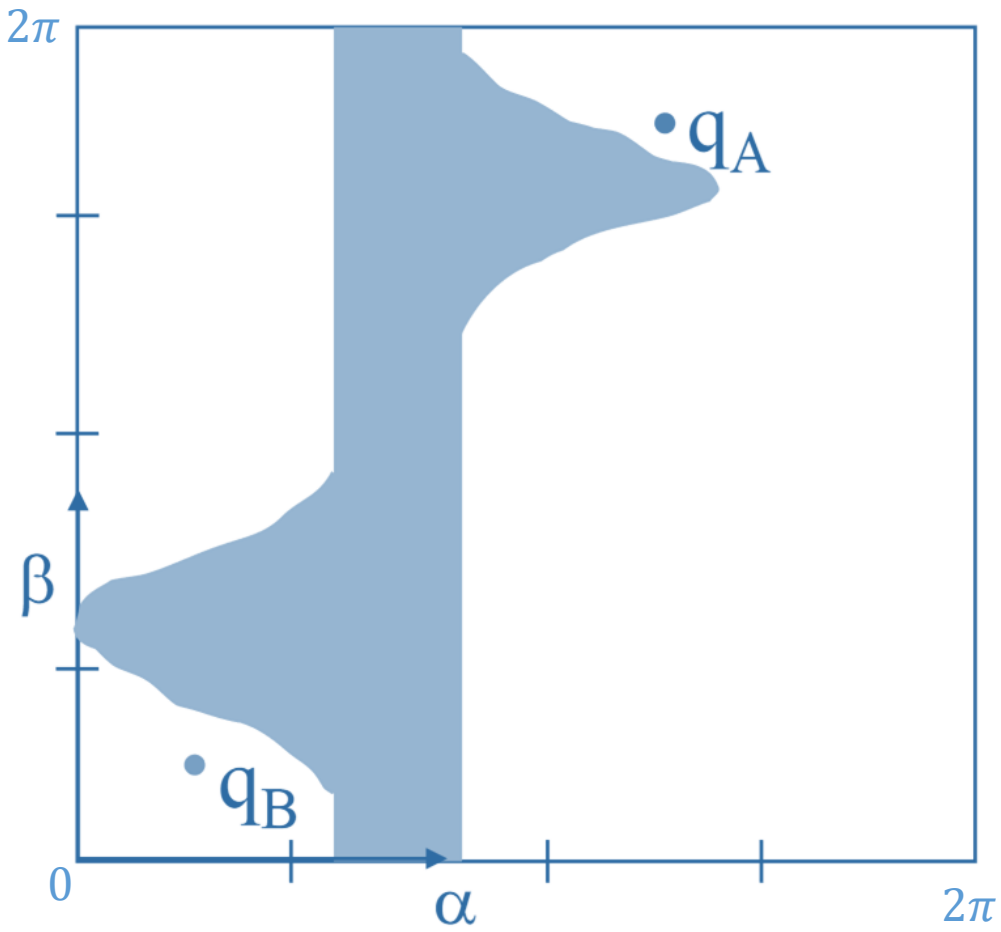
Exemple classique



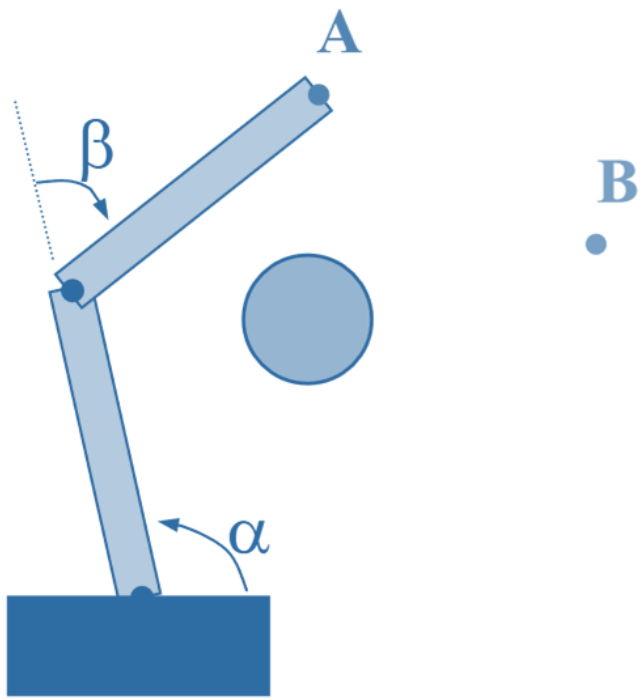


Exemple classique

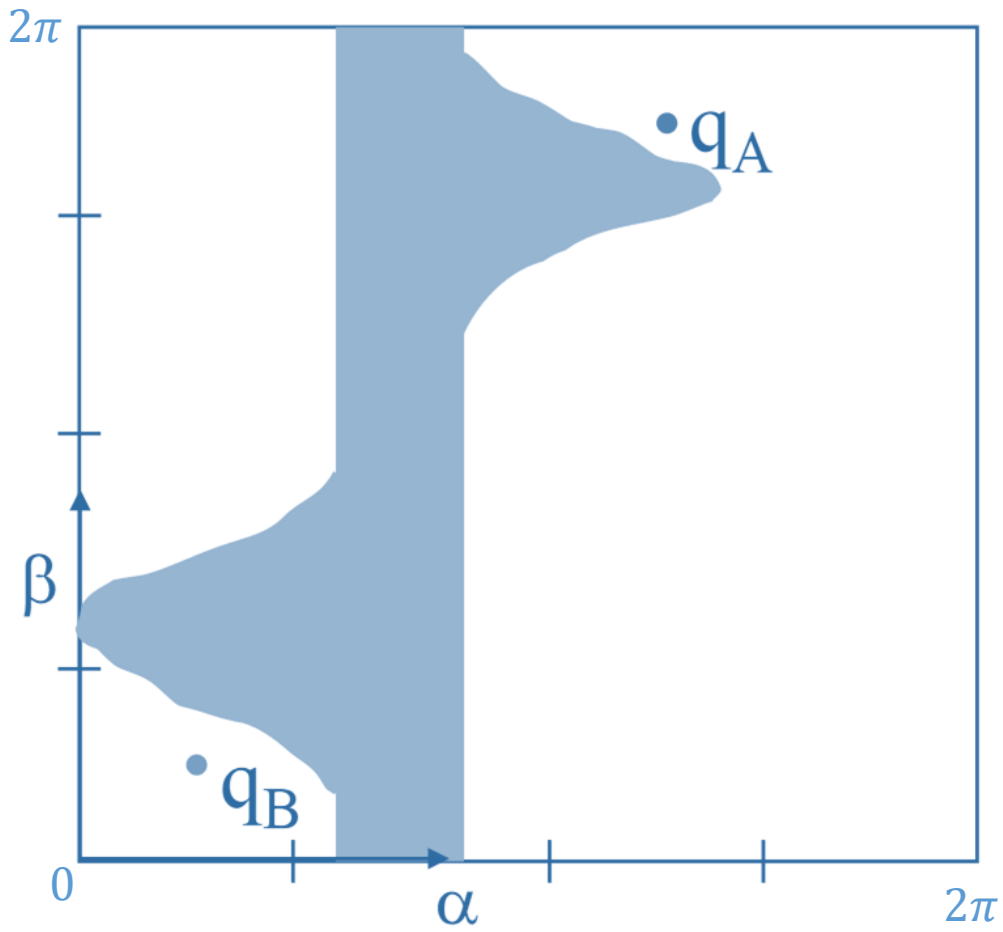




Exemple classique

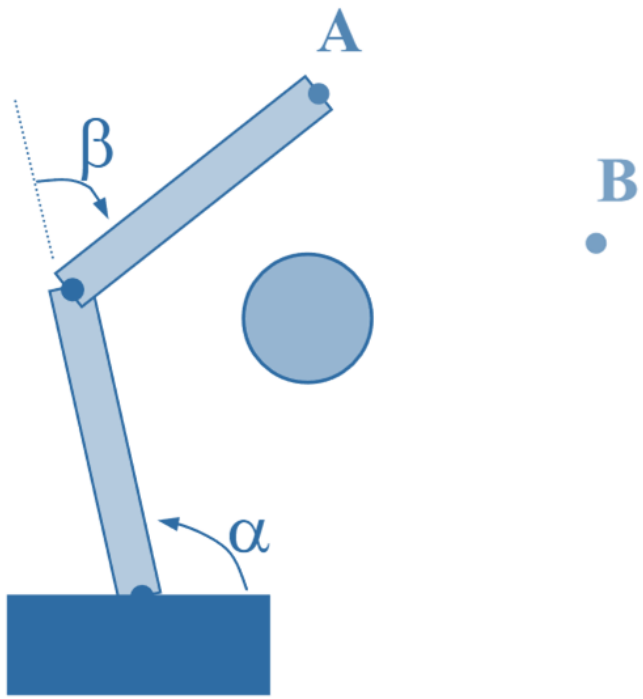


Planification de mouvement

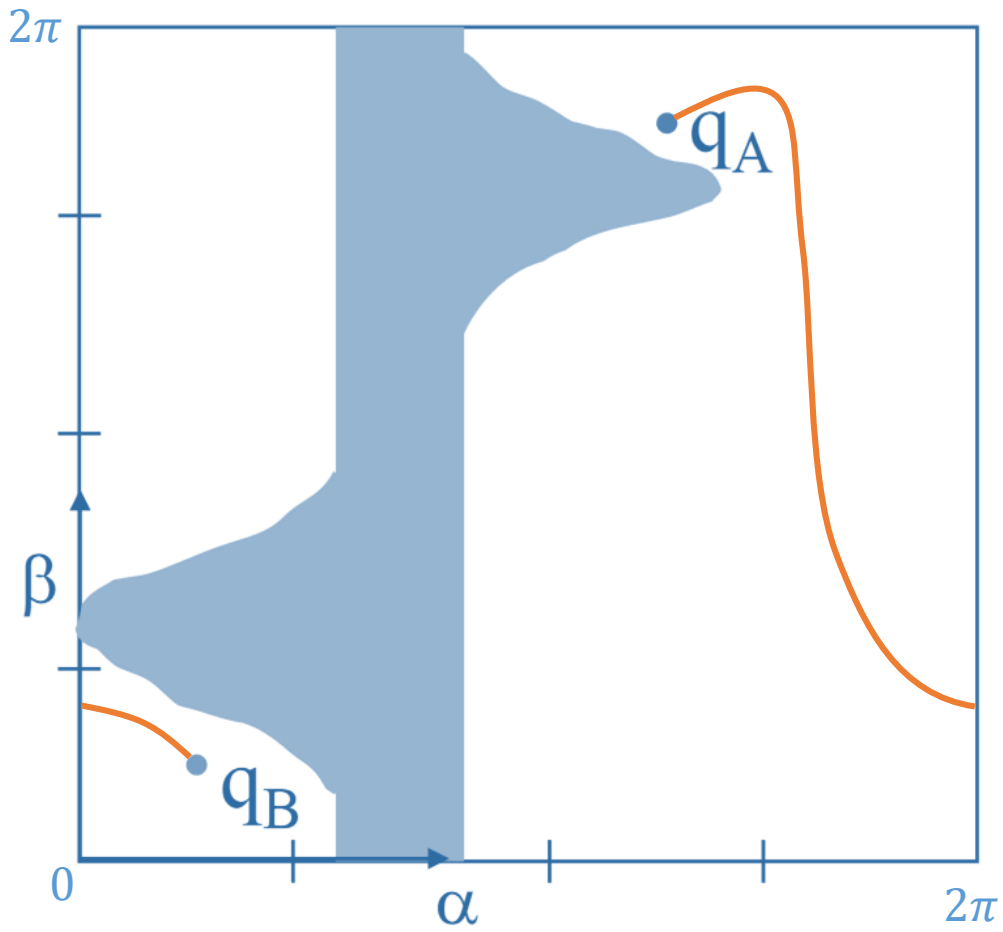


Planification de chemin

Exemple classique

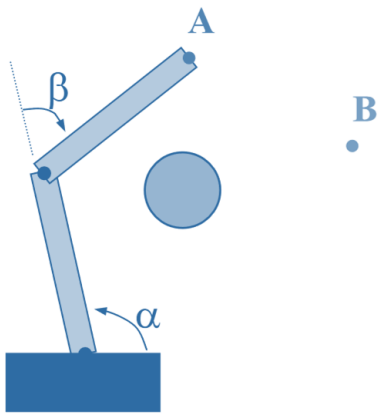


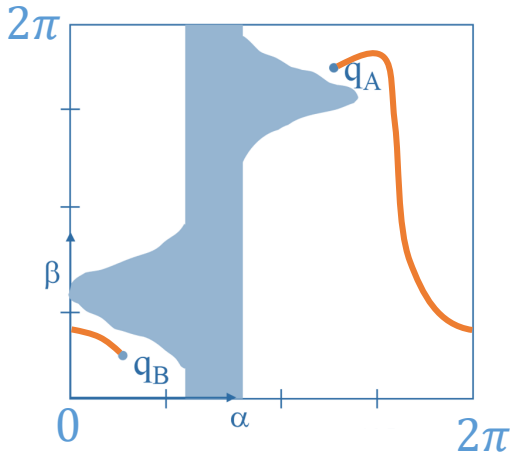
Planification de mouvement



Planification de chemin

Exemple classique





Planification de mouvement —réduction→ Planification de chemin

L'espace des configurations $\mathcal{C}$ (a.k.a C-space)

- point dans $\mathbb{R}^n$: $\mathbb{R}^n$
- solide en 2D : $SE(2) = \mathbb{R}^2 \times SO(2) = \mathbb{R}^2 \times \mathbb{S}^1$
- liaison pivot : $\mathbb{S}^1$ (cercle)
- liaison rotule : $SO(3) = \mathbb{RP}^3$(espace projectif réel)
  $=$ hémisphère quaternions unitaires
  avec identification antipodale
- solide en 3D : $SE(3) = \mathbb{R}^3 \times SO(3)$
- Bras à n pivots : $\mathbb{T}^n = \left(\mathbb{S}^1\right)^n$ (tore de dimension n)

Le problème du « déménageur de piano »

Défini par la donné de :

- Un **monde physique** $\mathcal{W} = \mathbb{R}^2$ or $\mathbb{R}^3$
- Un **robot** $\mathcal{A} \subset \mathcal{W}$ comme une collection de corps rigides articulées entre elles dans des configurations $q \in \mathcal{C}$, où $\mathcal{C}$ est une variété de dimension n
- Une transformation $g: \mathcal{C} \rightarrow 2^{\mathcal{W}}$(cinématique directe) qui met le robot dans une configuration donnée, on dénotera par abus de notation $g(q)$ comme $\mathcal{A}(q)$
- Une region d'**obstacles** compactes $\mathcal{O} \subset \mathcal{W}$

Le problème de **planification de mouvement** de $\mathcal{A}$ dans $\mathcal{W}$ se réduit à un problème de **planification de chemin** dans $C_{free} = C \setminus C_{obs}$ où $C_{obs} = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$

Un problème de géométrie algébrique computationnelle

- Lemme:

Si on suppose que $\mathcal{A}$ **et** $\mathcal{O}$ sont définis comme des **régions semi-algébriques** de $\mathcal{W}$ (*i.e.* définis comme une union finie d'intersections finies de régions délimitées par des équations polynomiales, ce qui inclut polygones/polyèdres, cercles/sphères, ellipses/ellipsoïdes, etc), **alors** il est possible de démontrer que $\mathcal{C}_{obs}$ et $\mathcal{C}_{free}$ sont **également** des ensembles **semi-algébriques** de $\mathbb{R}^n$ pour tous les types de robots dont les espaces de configurations sont définis comme des produits Cartésiens des variétés précédemment listées (en particulier : $\mathbb{R}, SO(2), SO(3)$)

  - idée : utilisation des formules trigonométriques de l'arc moitié pour se « débarrasser » des fonctions trigonométriques dans les formules de cinématique directe

$$t = \tan\frac{\theta}{2}, \quad \sin\theta = \frac{2t}{1+t^2}\ ,\ \cos\theta = \frac{1-t^2}{1+t^2}\ ,\ \tan\theta = \frac{2t}{1-t^2}$$

- Exemple : robot polyédrique dans un monde polyédrique

Un problème de géométrie algébrique computationnelle

- Les **régions semi-algébriques** sont définies comme des **expressions de Tarski** (formules logiques sur des expressions polynomiales en les coordonnées de la variété $\mathcal{C}$ avec quantificateurs et variables libres)

- La **décomposition cylindrique algébrique** (a.k.a décomposition de Collins ou CAD, 1975) utilisée pour l'**élimination de quantificateurs** dans les expressions de Tarski (et pour décider de la **satisfiabilité** des expressions de Tarski) produit une décomposition en cellules de $\mathcal{C}_{free}$ (similaire à la decomposition en cellules verticale évoquée précédemment)

- C'est donc une solution au problème de planification de mouvement (Schwartz et Sharir, 1990)
  - Complexité doublement exponentielle en la dimension de $\mathcal{C}$

Où en sommes-nous ?

Malheureusement…

Curse of dimensionality

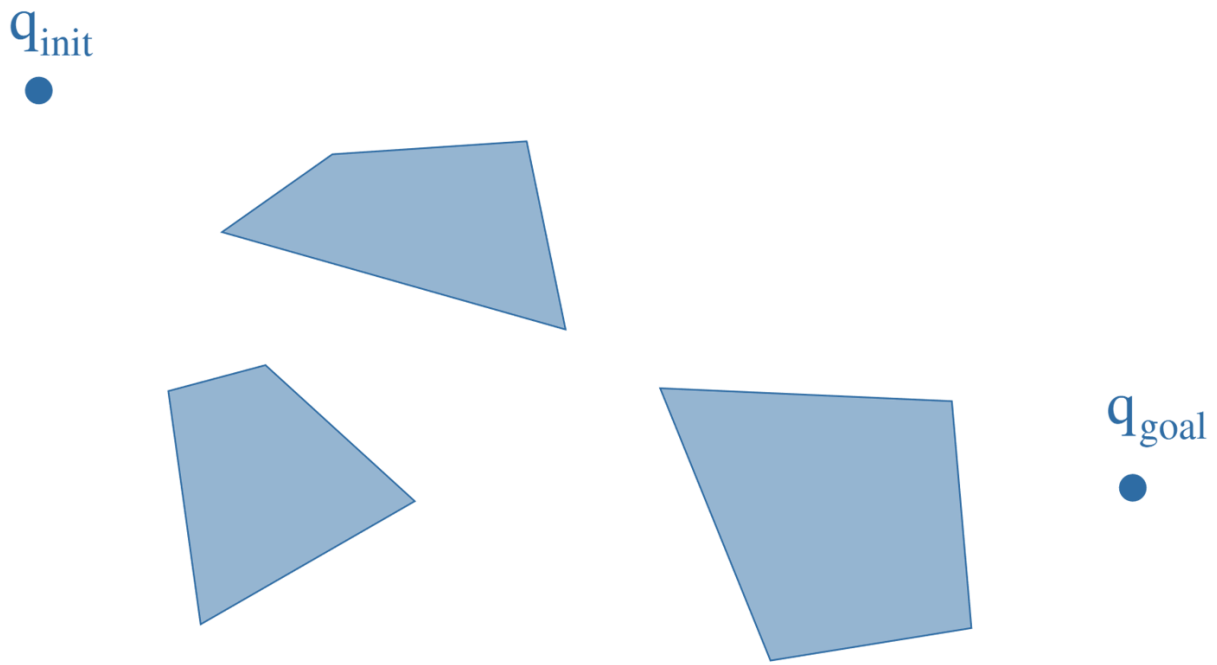
- Problème 1 : plusieurs des algorithmes de planification de chemin vus précédemment se basent sur **des hypothèses fortes** sur la forme des obstacles, *e.g.* des **polygones/polyèdres** (bien qu'on puisse faire de telles hypothèses sur pour $\mathcal{O}$, il est difficile d'en dire autant pour $\mathcal{C}_{obs}$)

- Problème 2 : Ces algorithmes nécessitent une **représentation explicite de $\mathcal{C}_{obs}$** (qu'on ne peut pas construire en general, **$\mathcal{C}_{obs}$ est défini implicitement**)

- Problème 3 : Même en supposant 1 et 2 résolus, les algorithmes présentés ne sont **pas** du tout **utilisables en pratique au-delà de 3 dimensions** (voire de 2 dimension pour certains). Or un espace de configurations d'un bras robotique typique est de dimension 6.

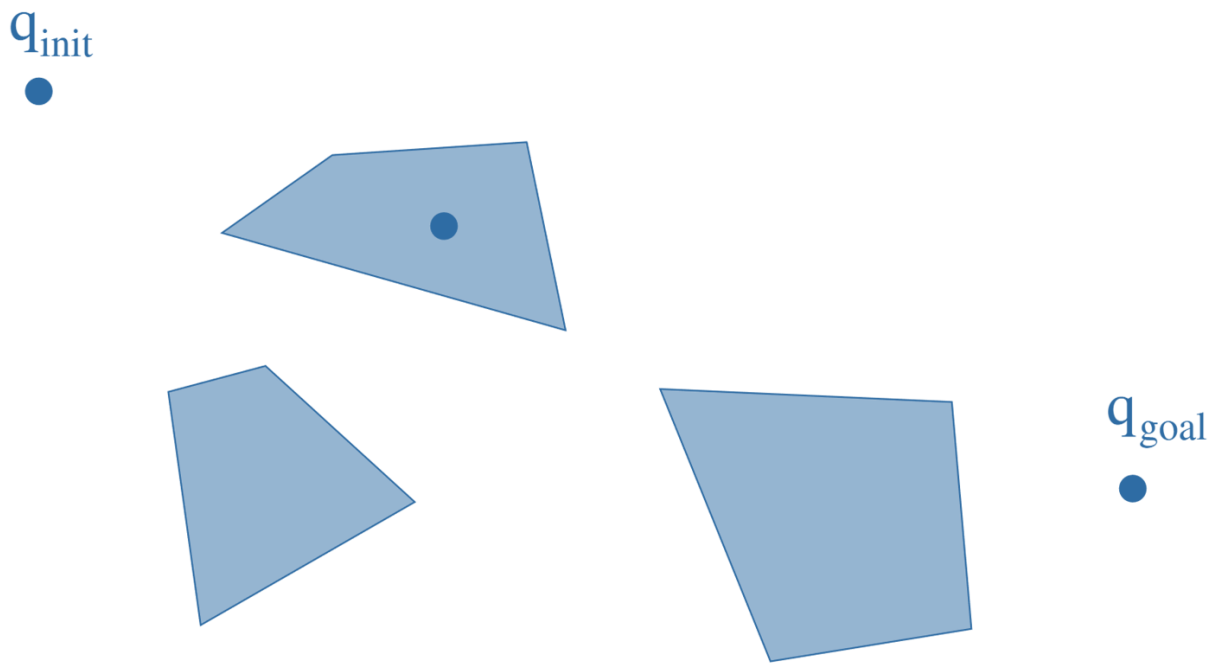
Algorithmes par échantillonnage aléatoire (années 90)

Probabilistic roadmap (PRM)

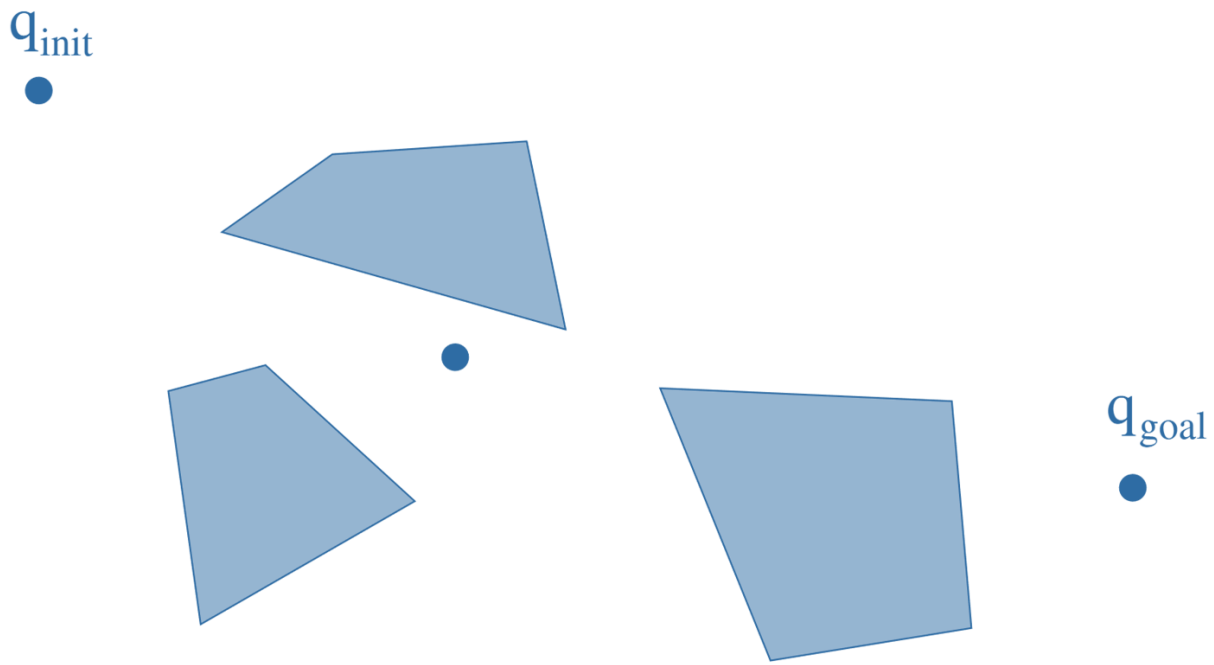


Illustrations de F. Lamiraux

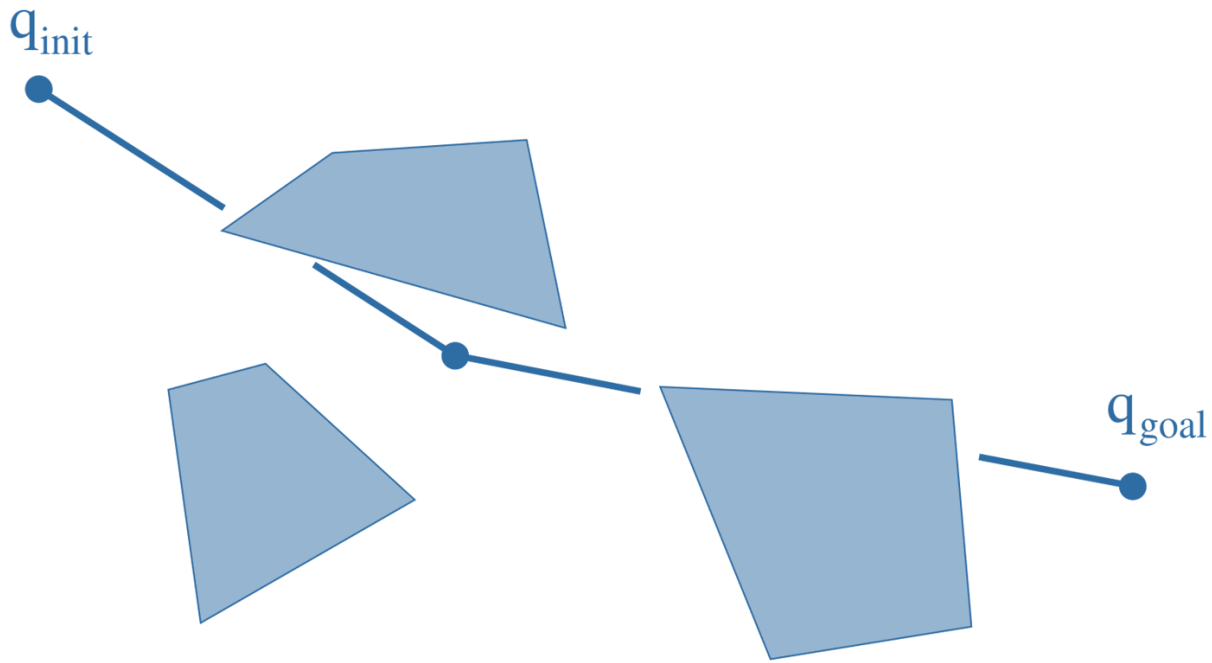
Probabilistic roadmap (PRM)



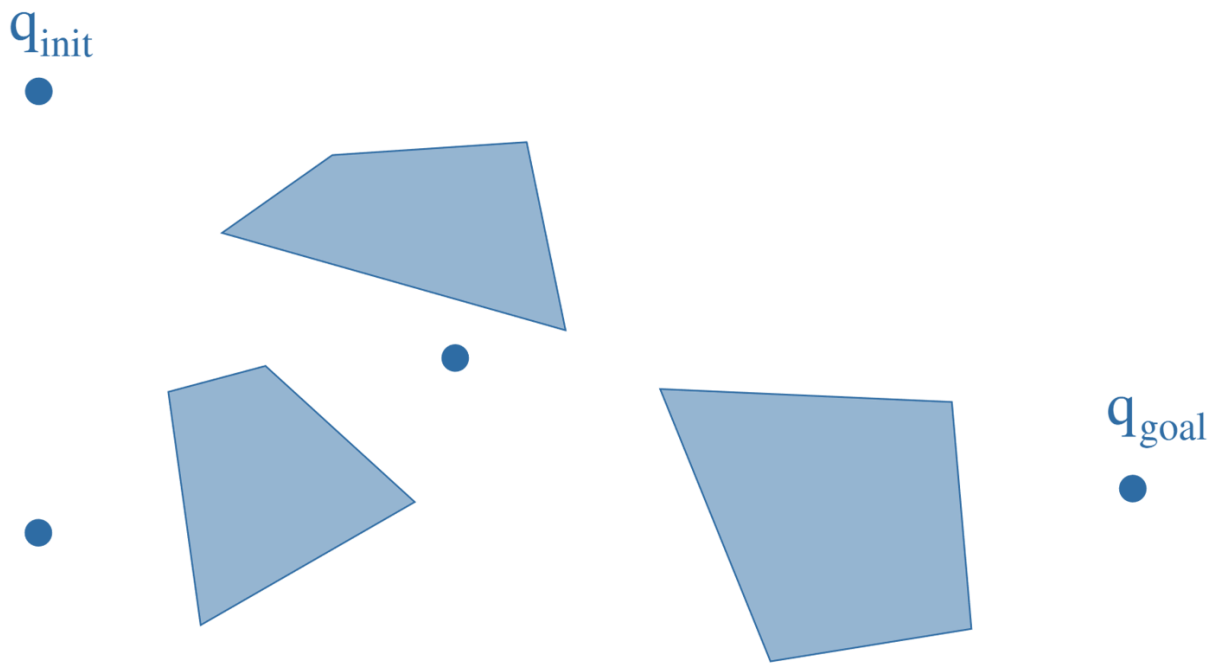
Probabilistic roadmap (PRM)



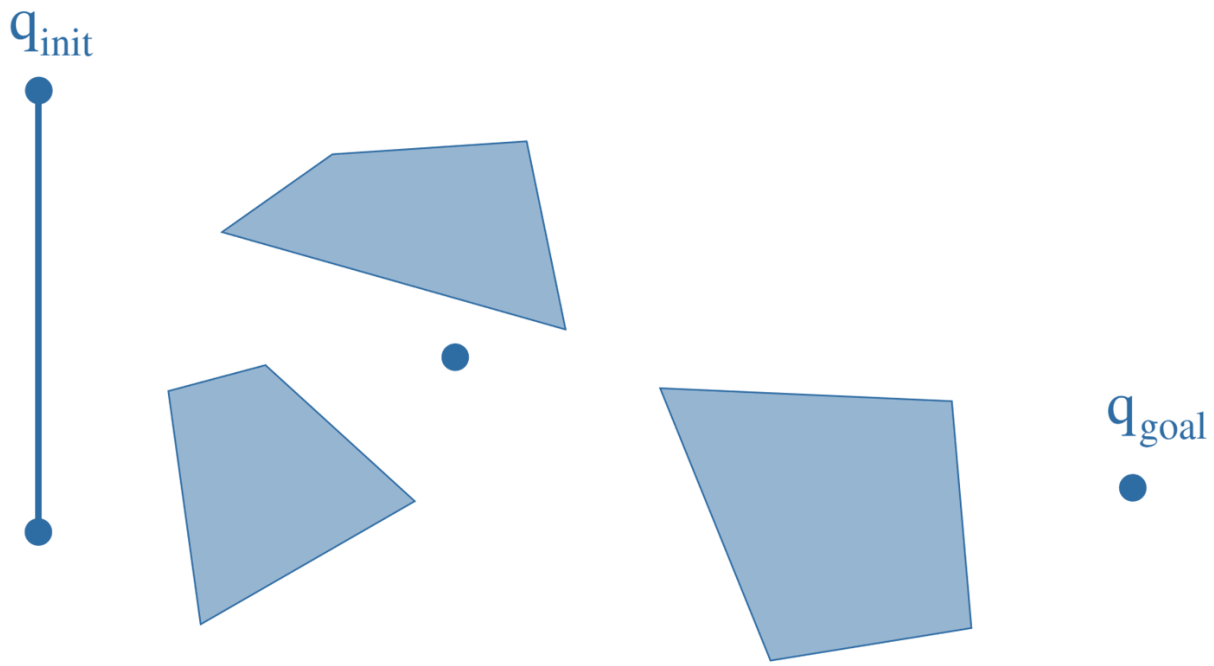
Probabilistic roadmap (PRM)



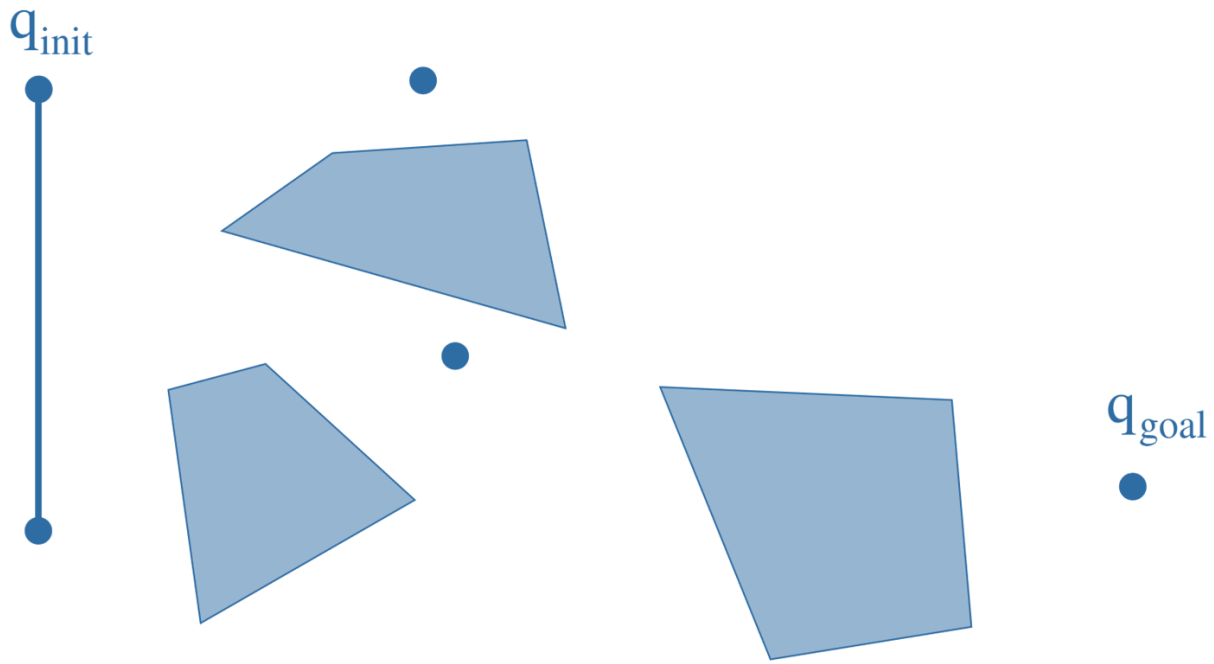
Probabilistic roadmap (PRM)



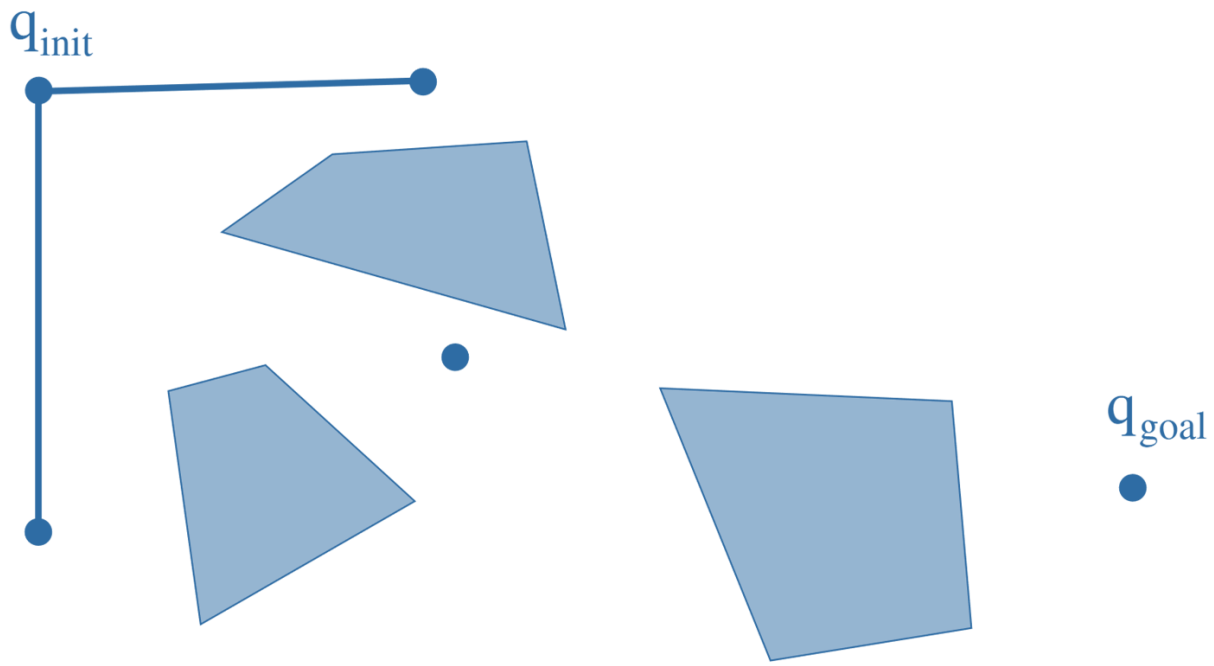
Probabilistic roadmap (PRM)



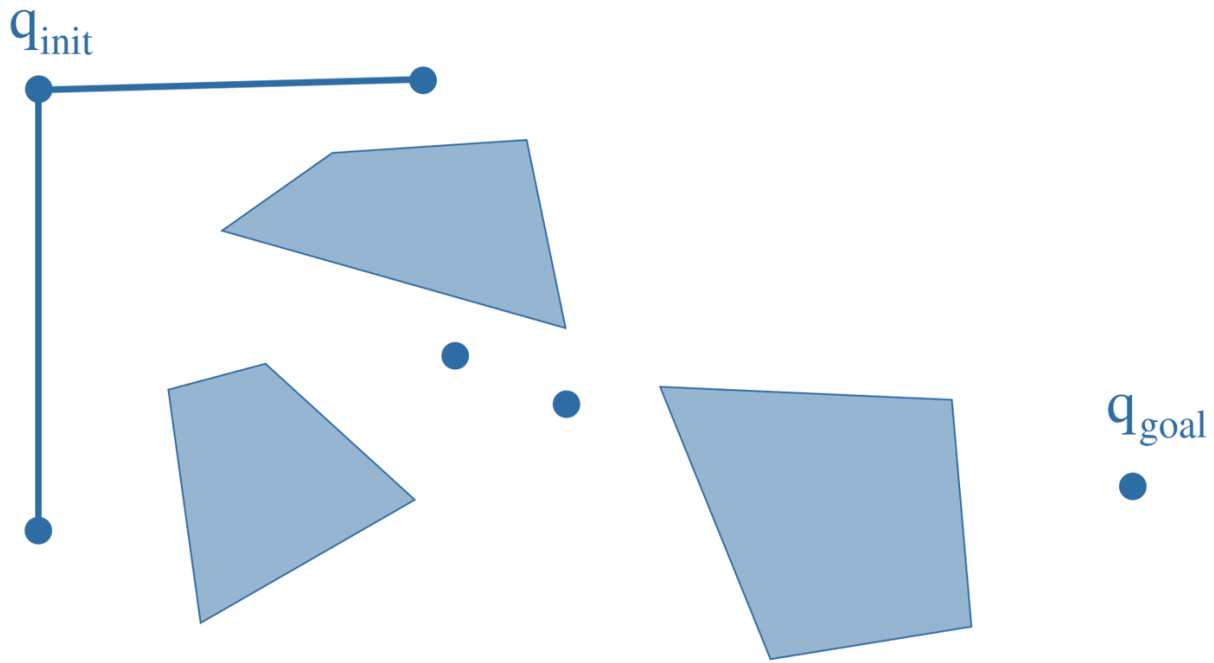
Probabilistic roadmap (PRM)



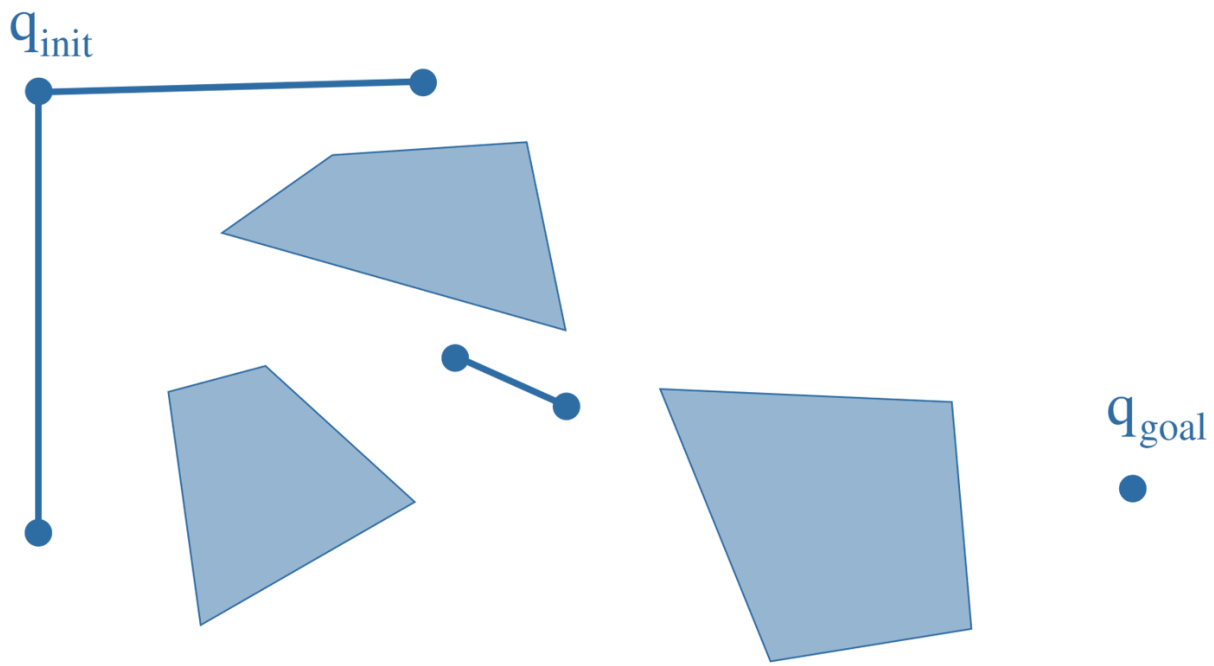
Probabilistic roadmap (PRM)



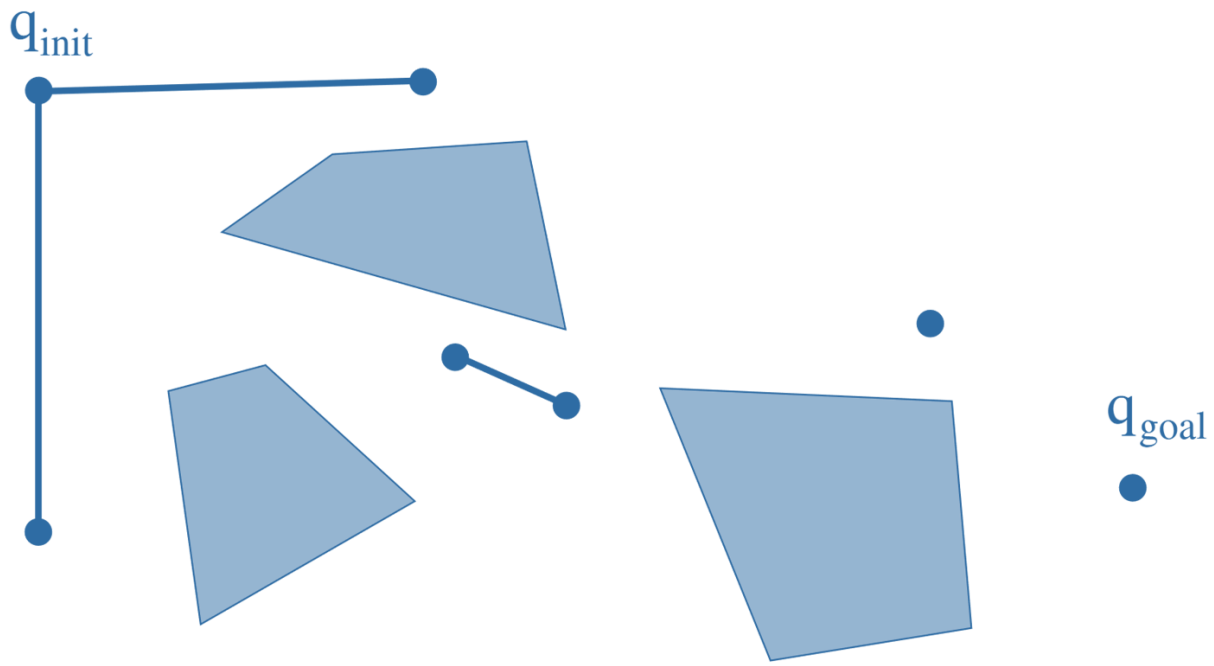
Probabilistic roadmap (PRM)



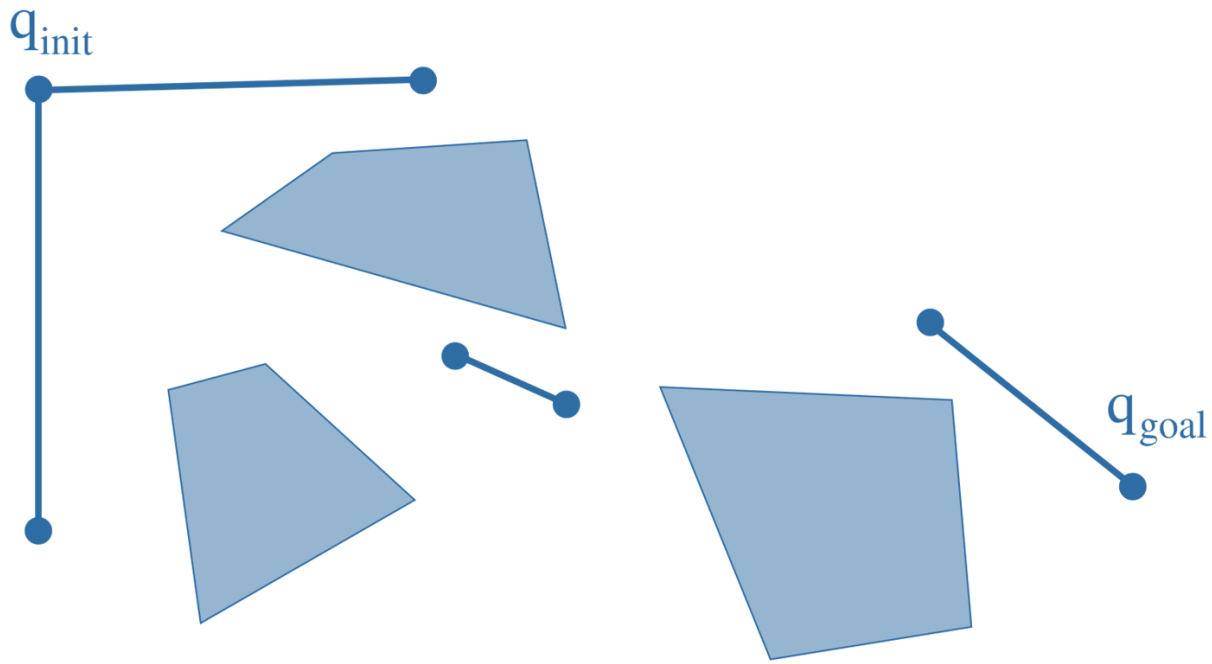
Probabilistic roadmap (PRM)



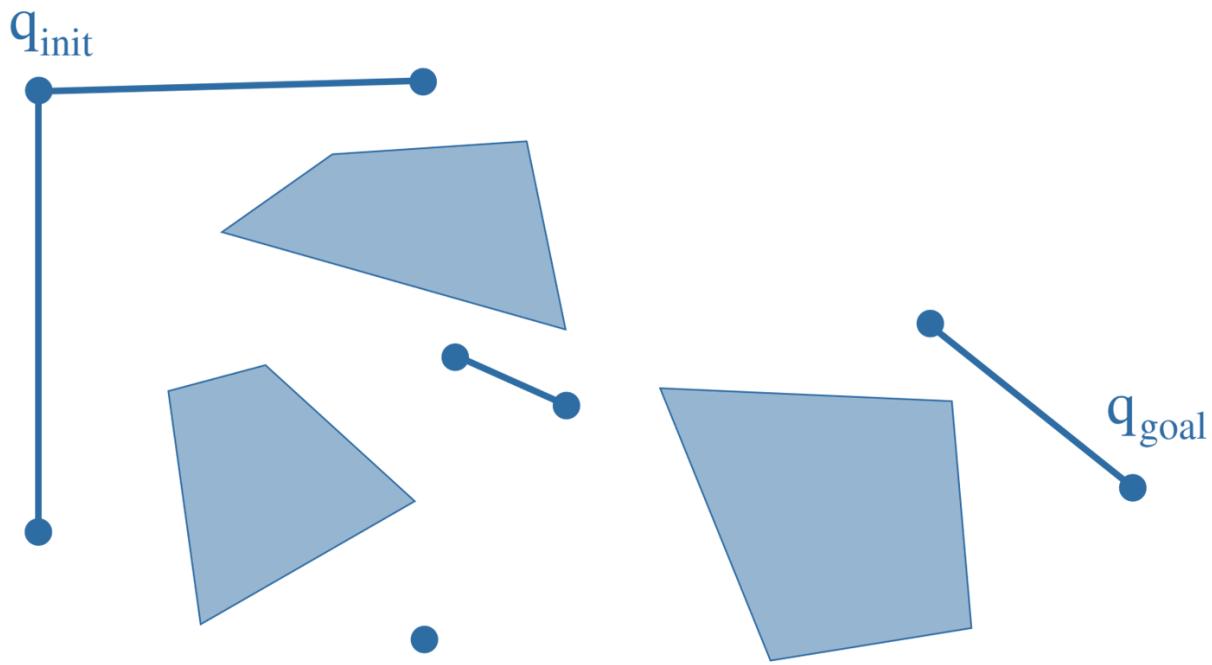
Probabilistic roadmap (PRM)



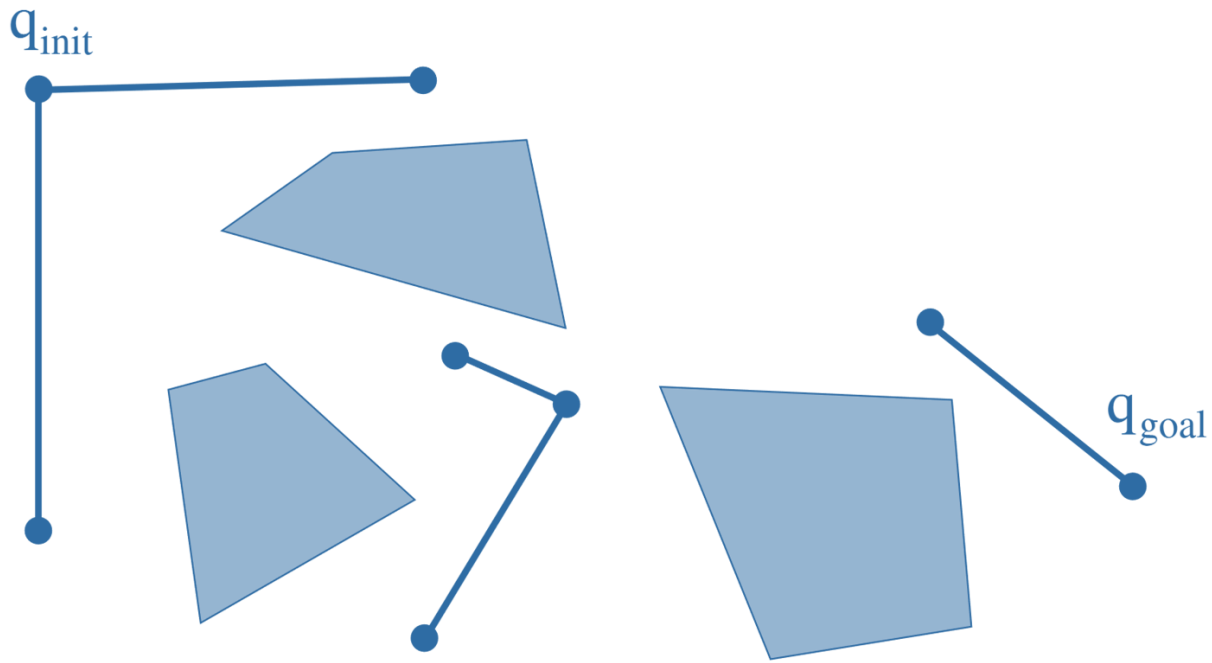
Probabilistic roadmap (PRM)



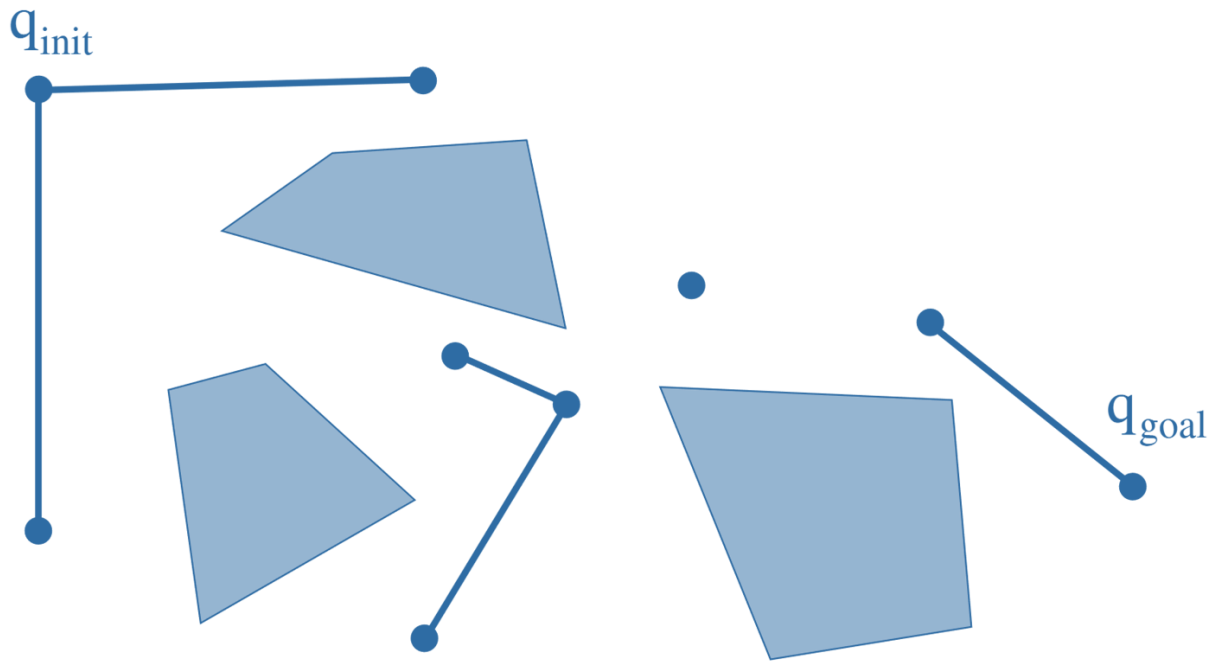
Probabilistic roadmap (PRM)



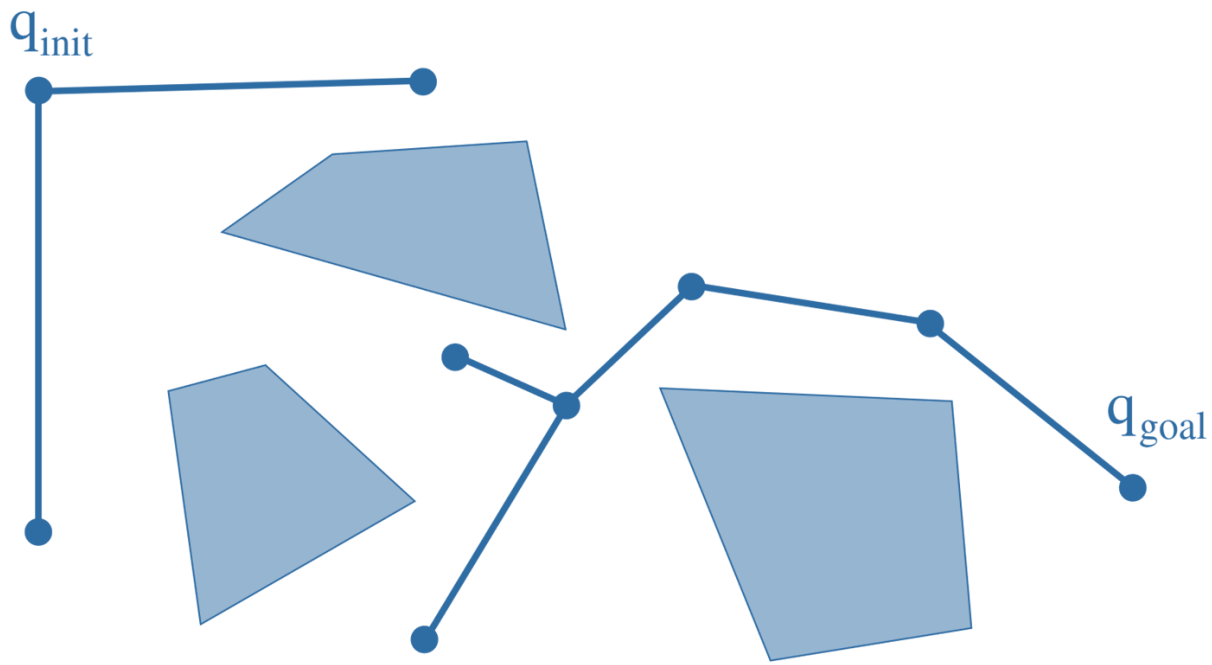
Probabilistic roadmap (PRM)



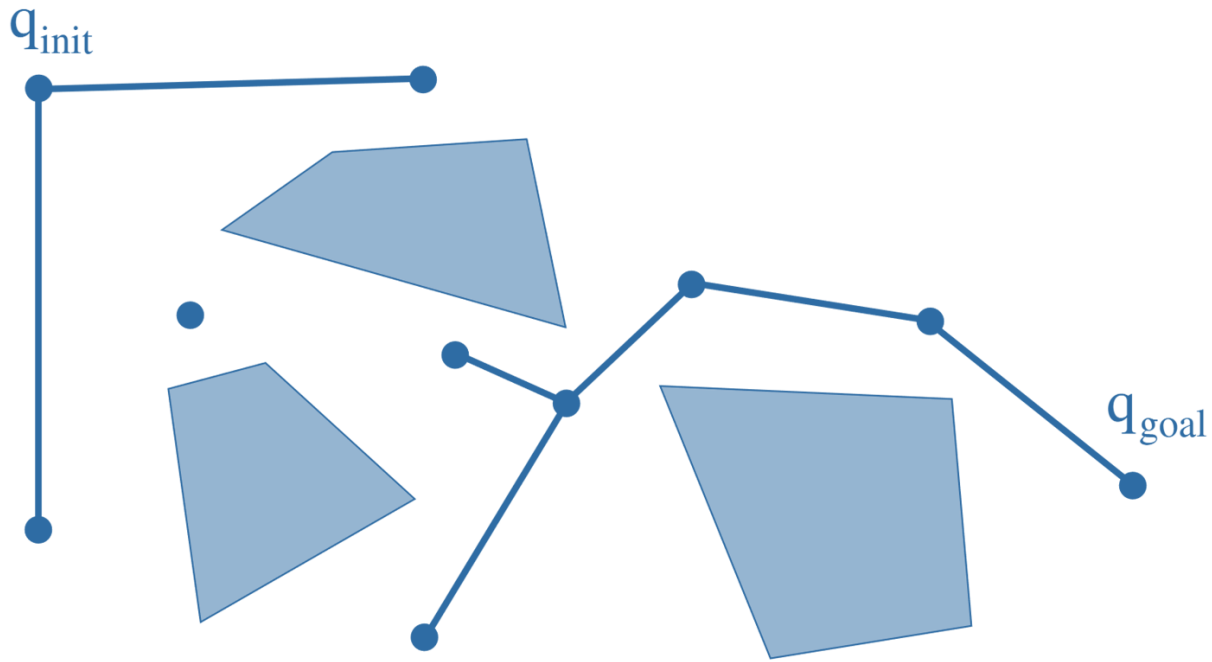
Probabilistic roadmap (PRM)



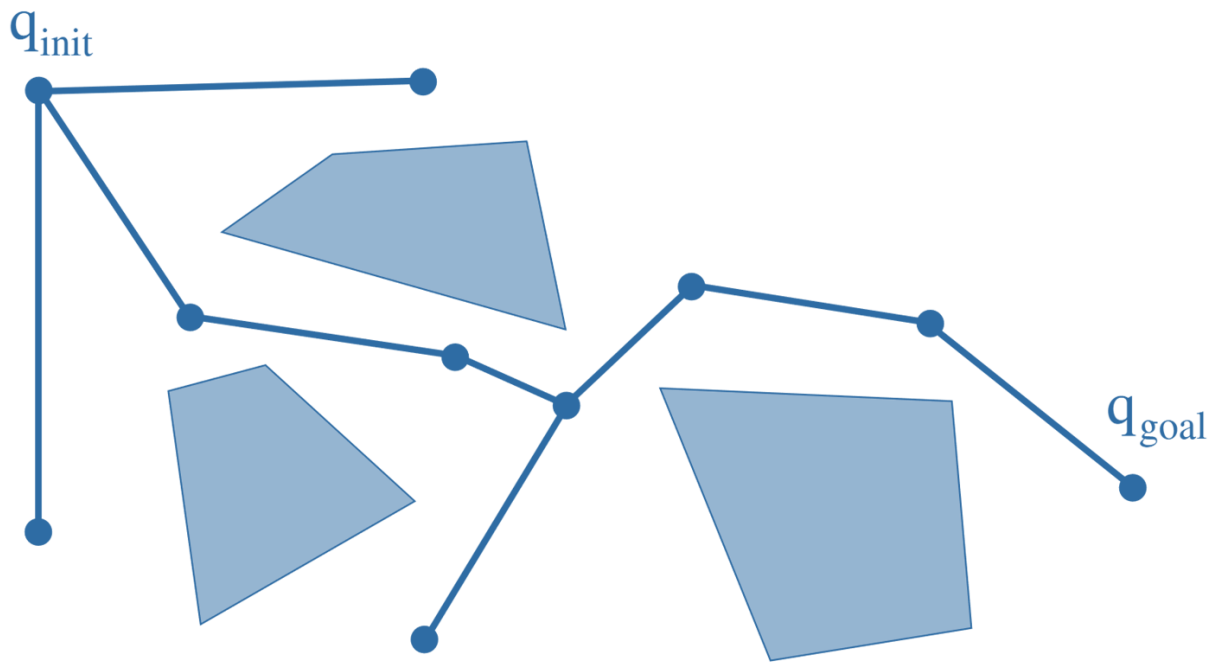
Probabilistic roadmap (PRM)



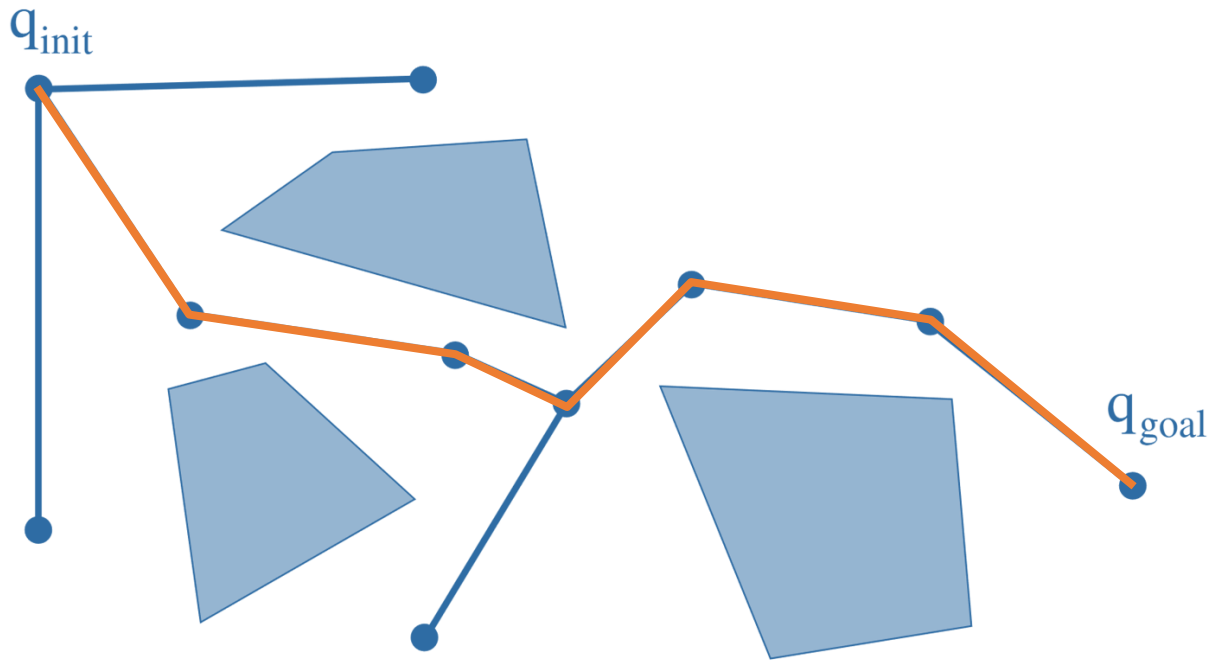
Probabilistic roadmap (PRM)



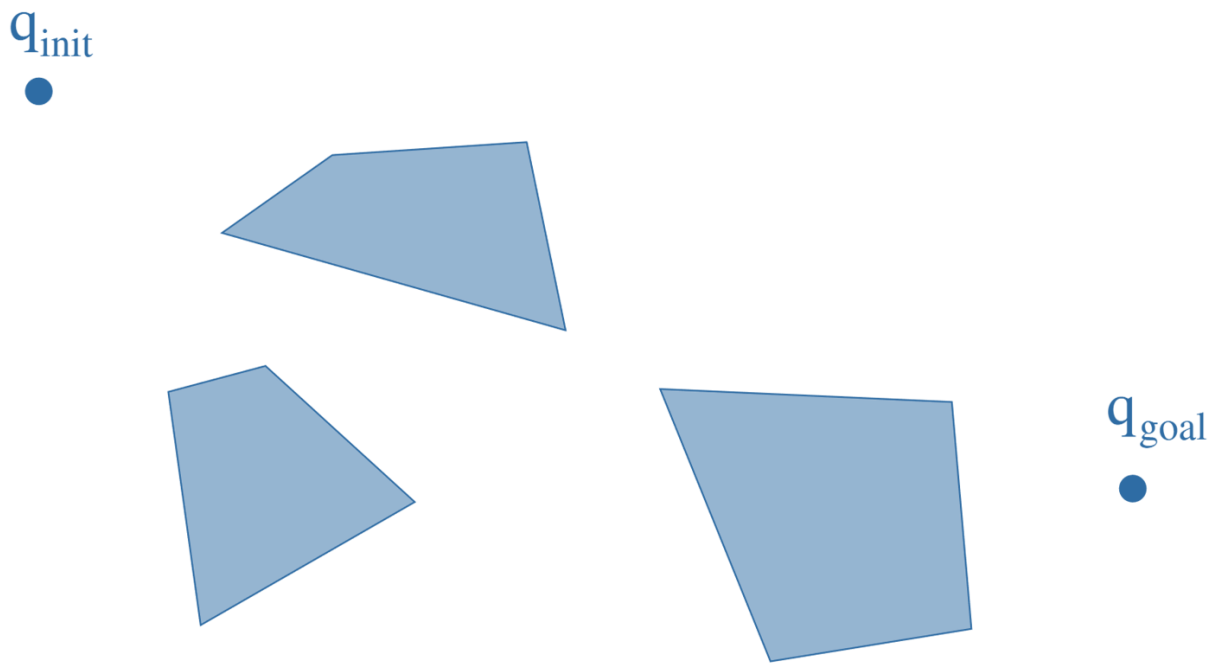
Probabilistic roadmap (PRM)



Probabilistic roadmap (PRM)

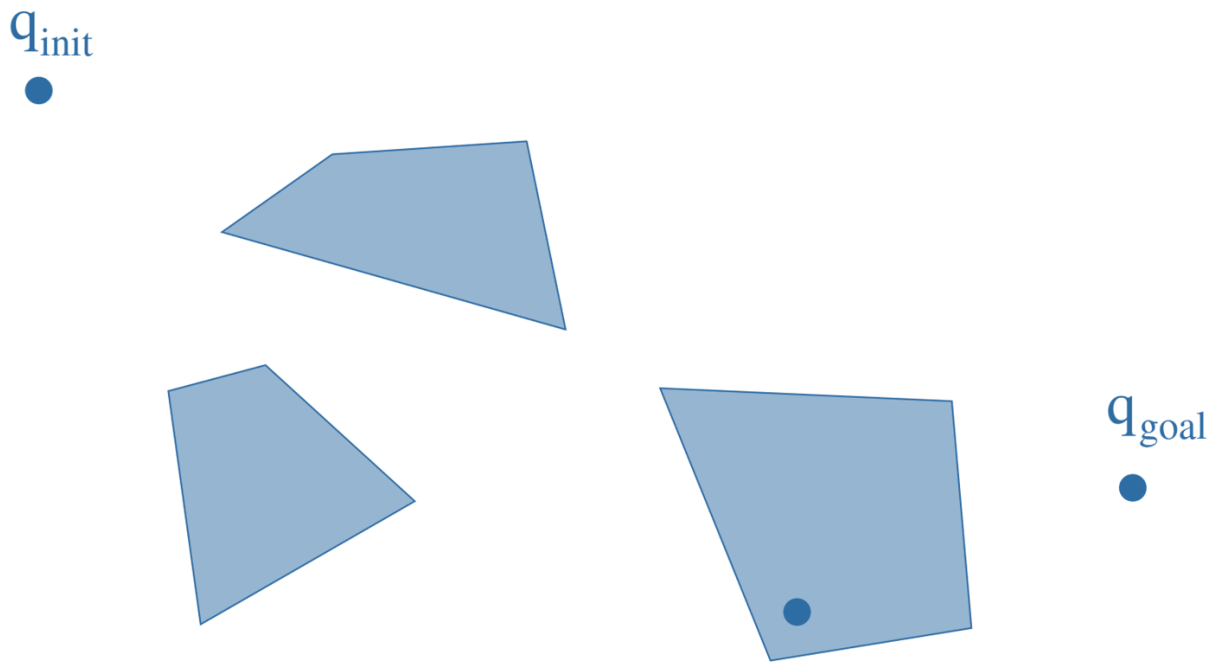


Rapidly-exploring Random Tree (RRT)

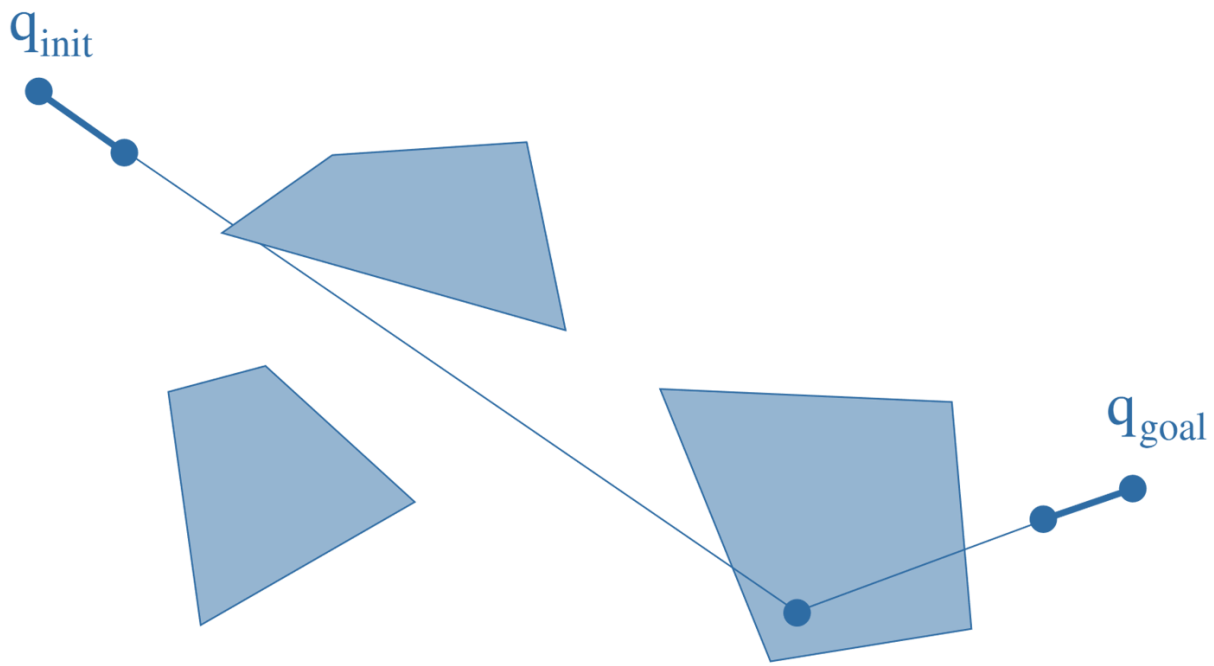


Illustrations de F. Lamiraux

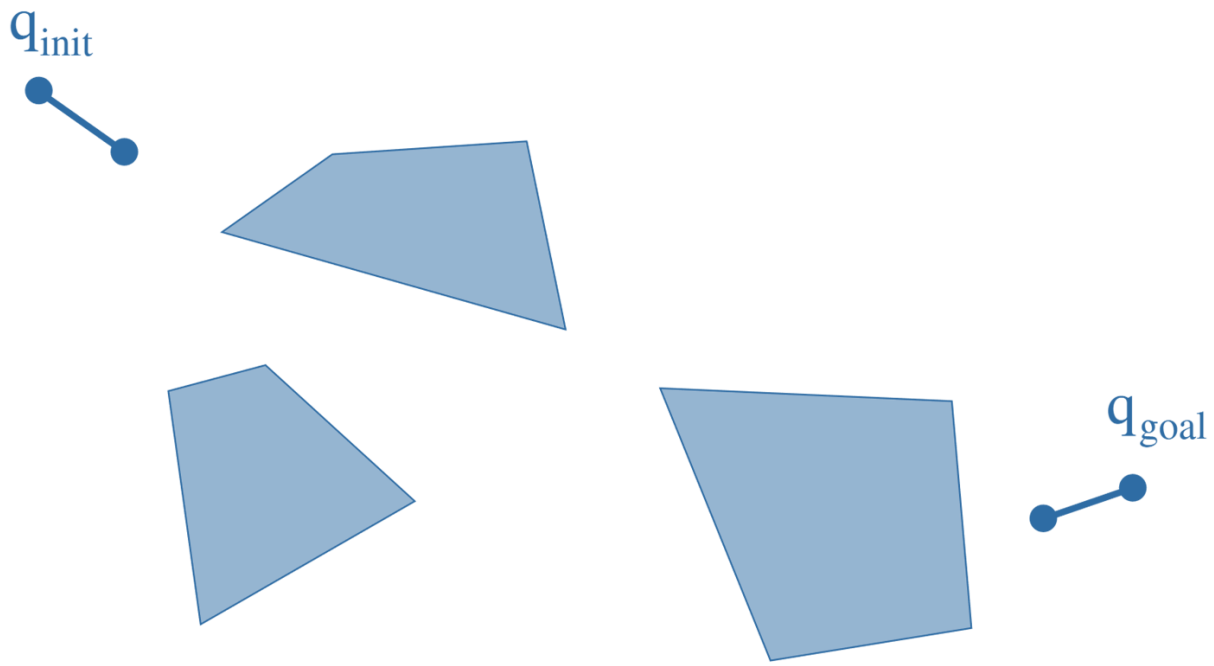
Rapidly-exploring Random Tree (RRT)



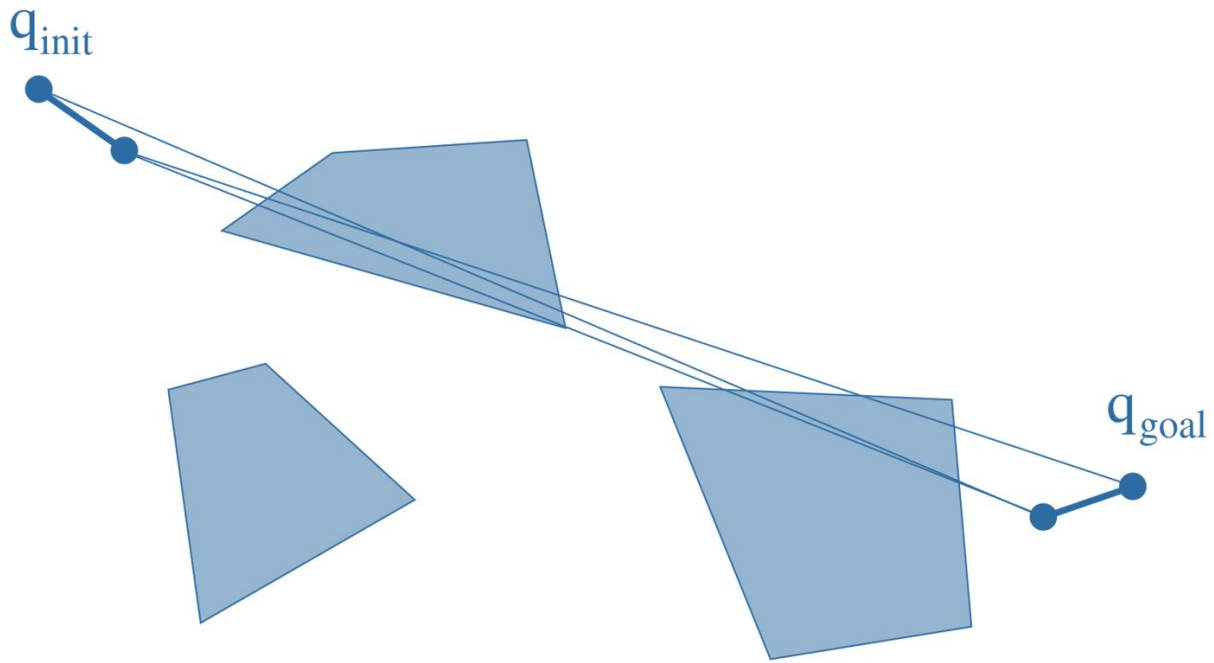
Rapidly-exploring Random Tree (RRT)



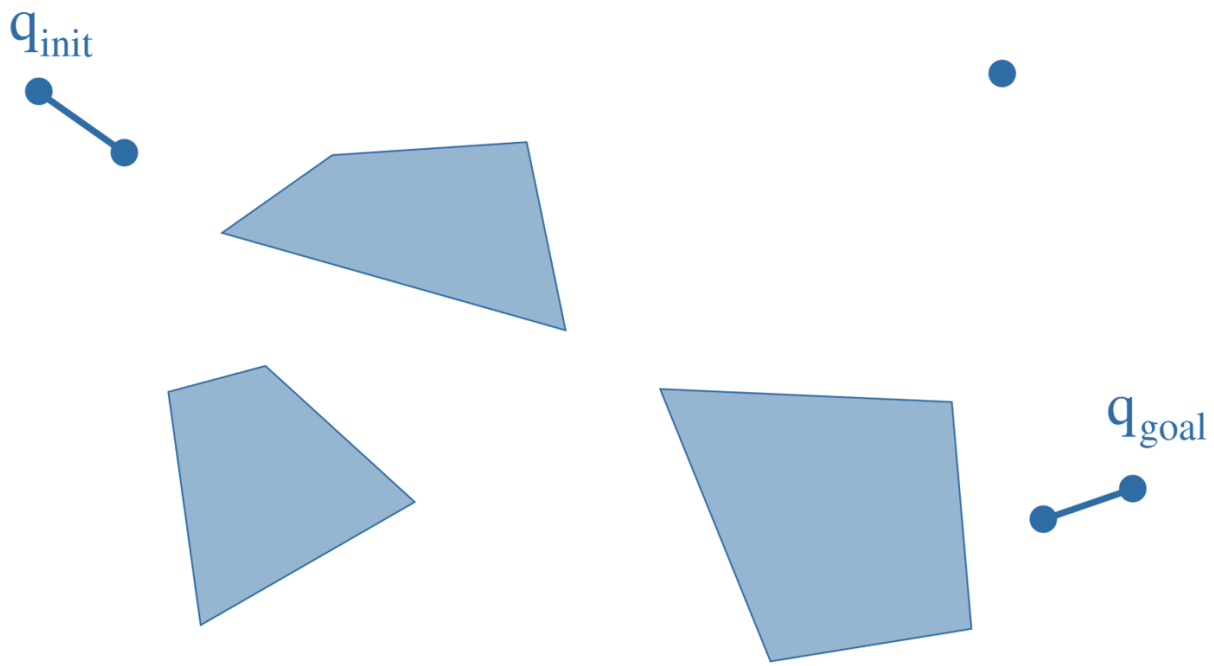
Rapidly-exploring Random Tree (RRT)



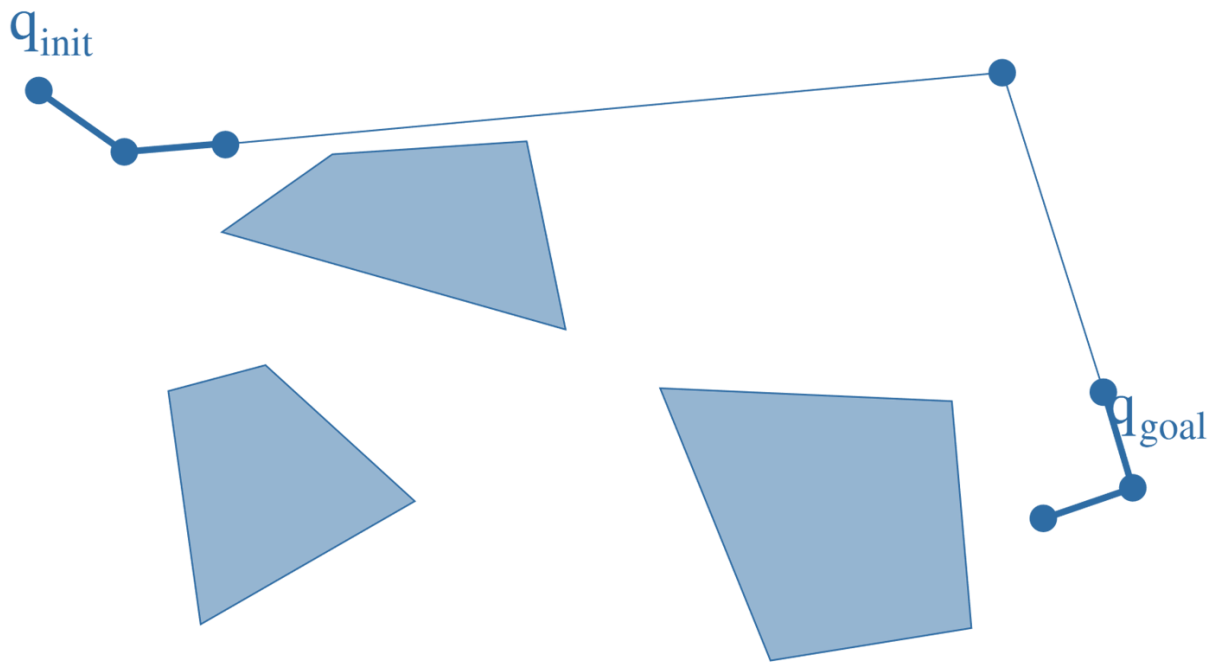
Rapidly-exploring Random Tree (RRT)



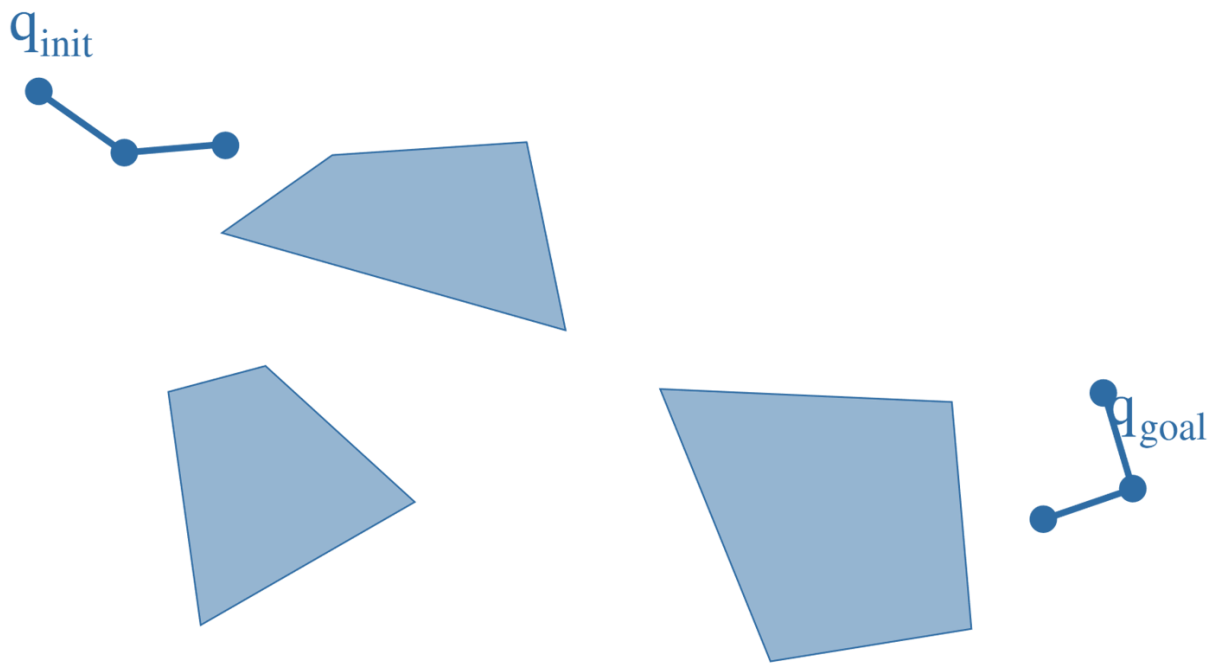
Rapidly-exploring Random Tree (RRT)



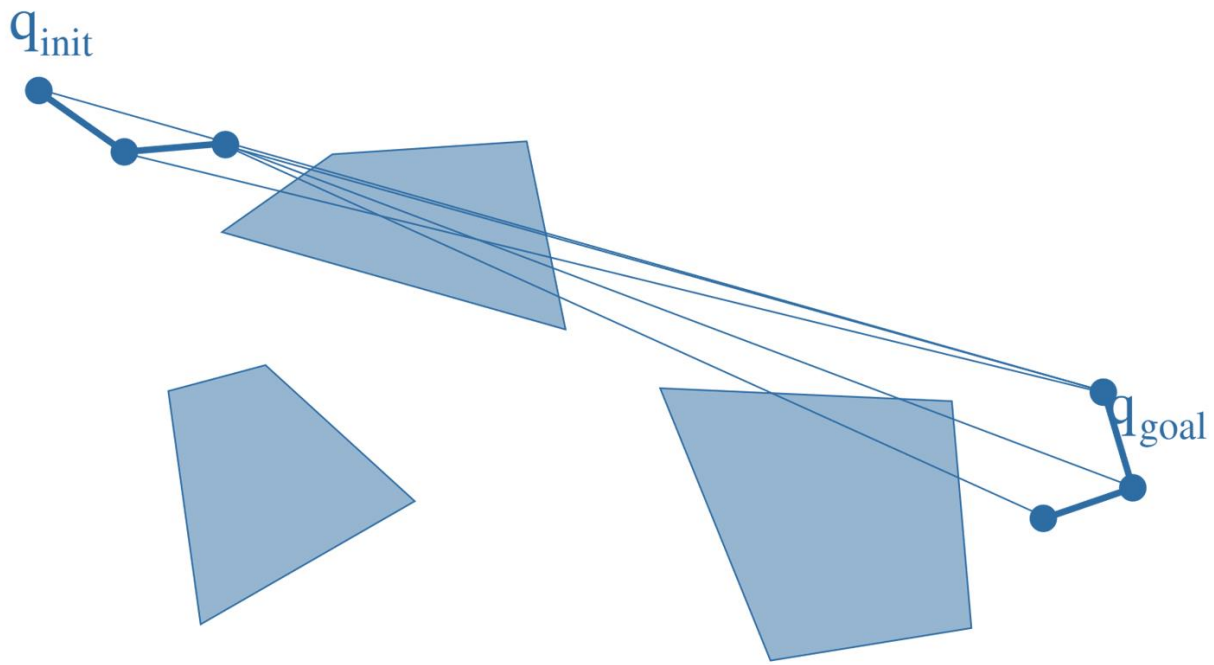
Rapidly-exploring Random Tree (RRT)



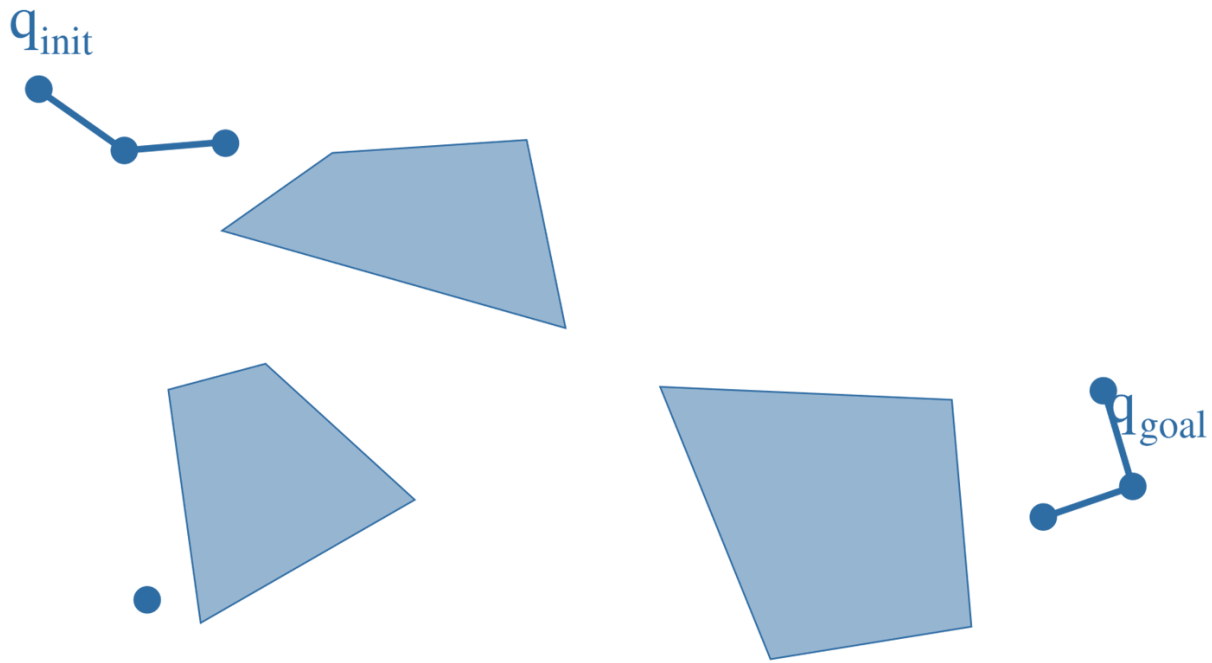
Rapidly-exploring Random Tree (RRT)



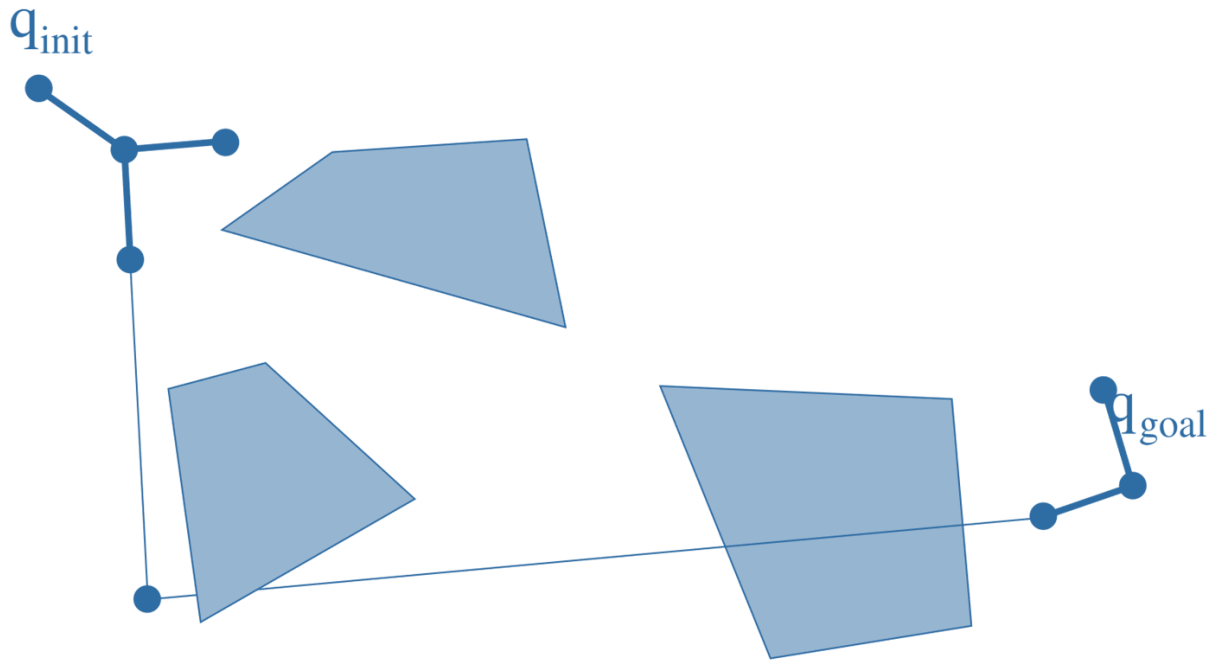
Rapidly-exploring Random Tree (RRT)



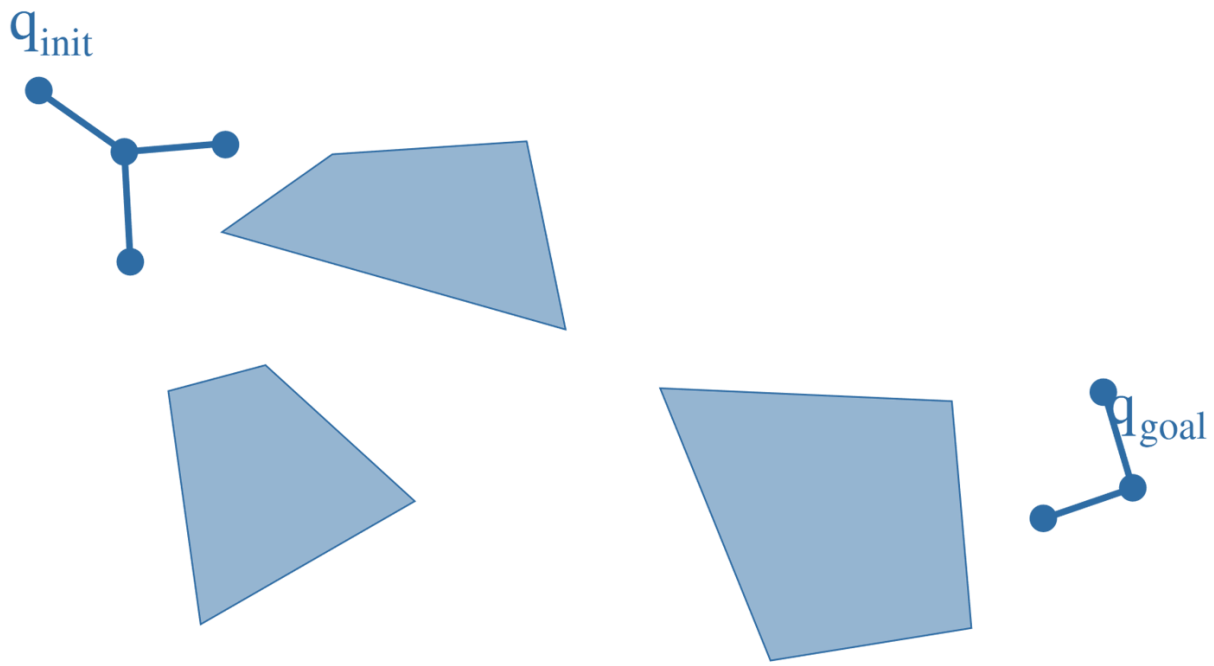
Rapidly-exploring Random Tree (RRT)



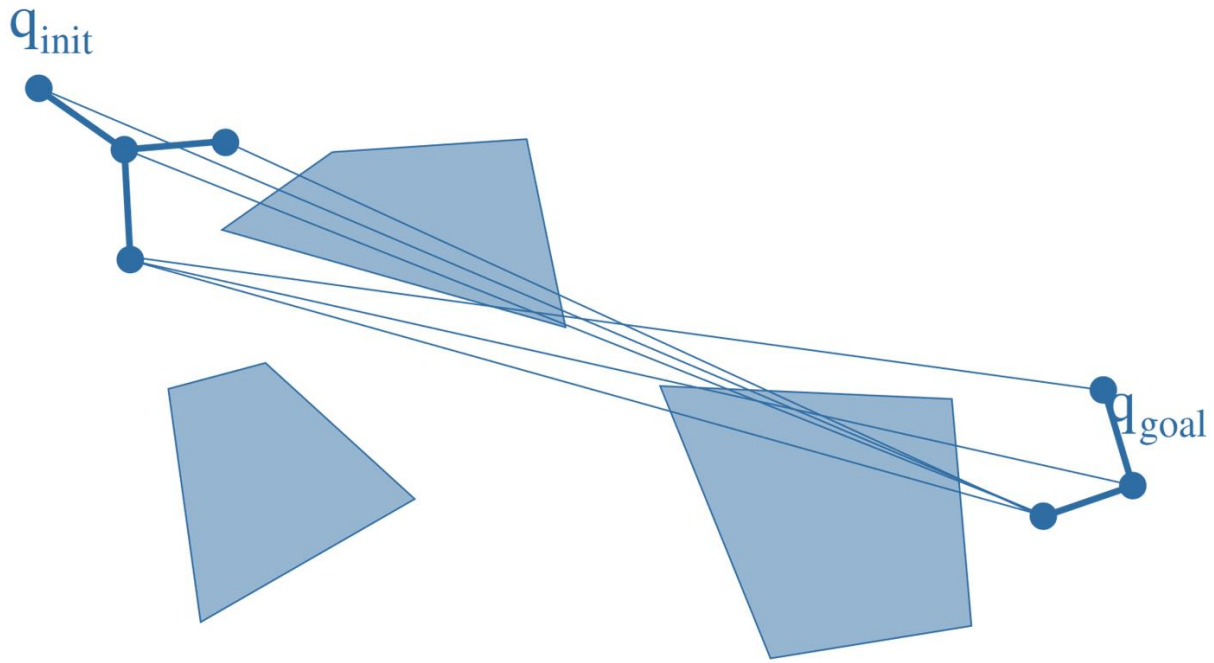
Rapidly-exploring Random Tree (RRT)



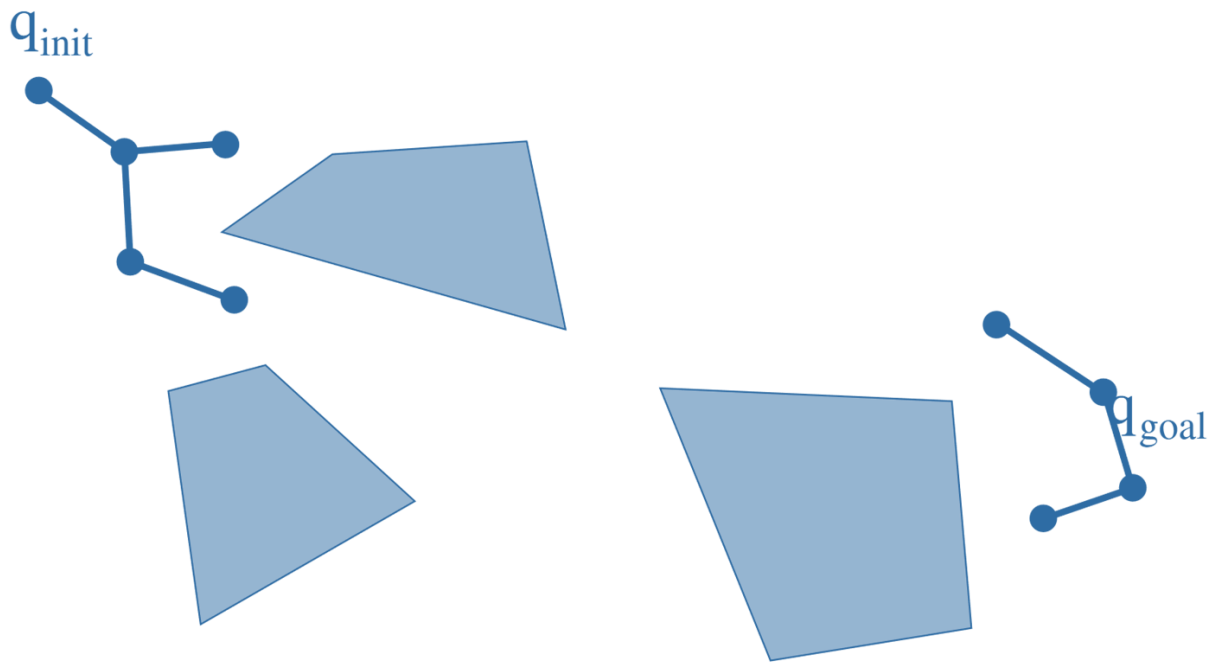
Rapidly-exploring Random Tree (RRT)



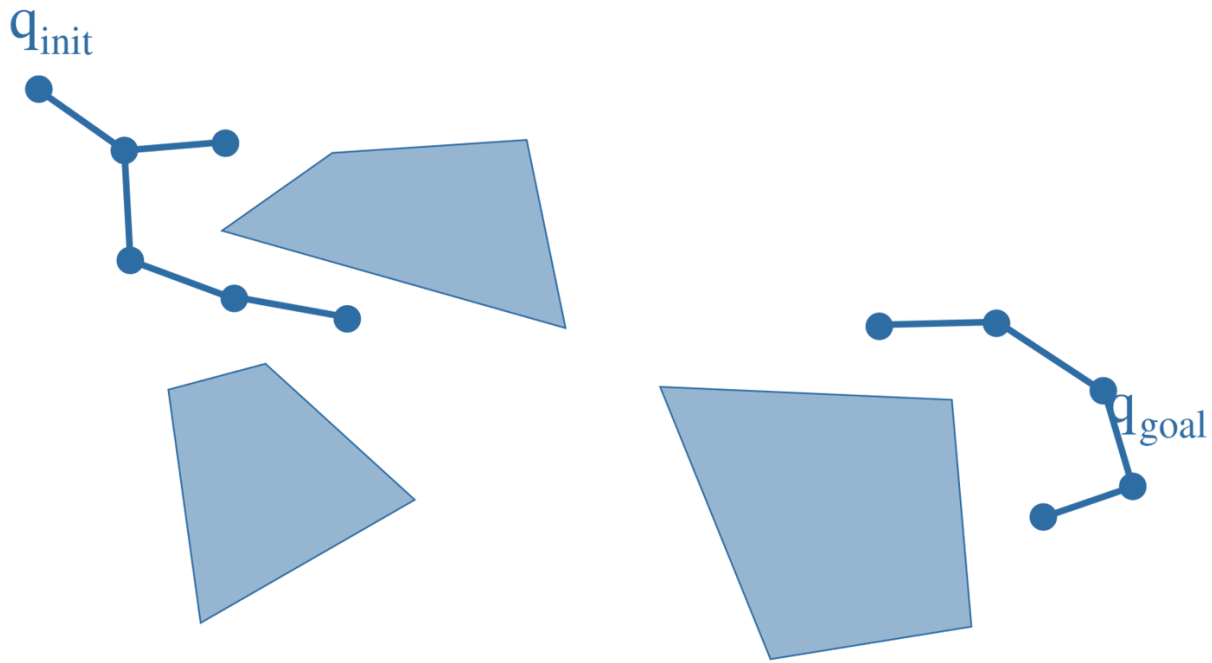
Rapidly-exploring Random Tree (RRT)



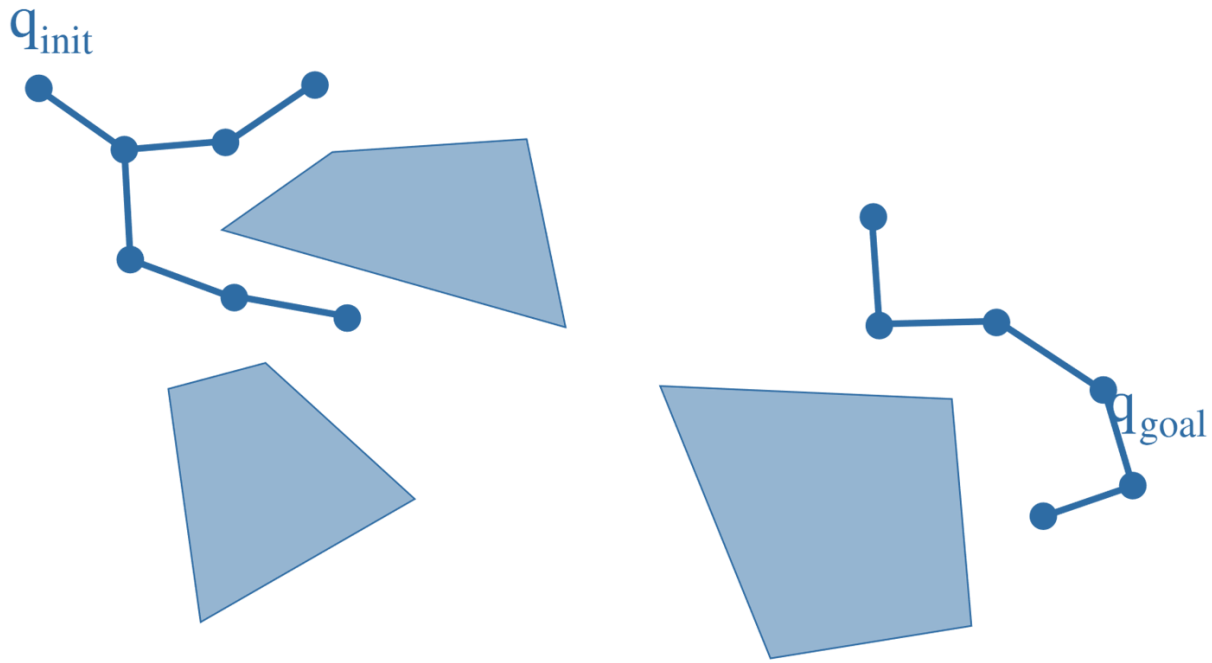
Rapidly-exploring Random Tree (RRT)



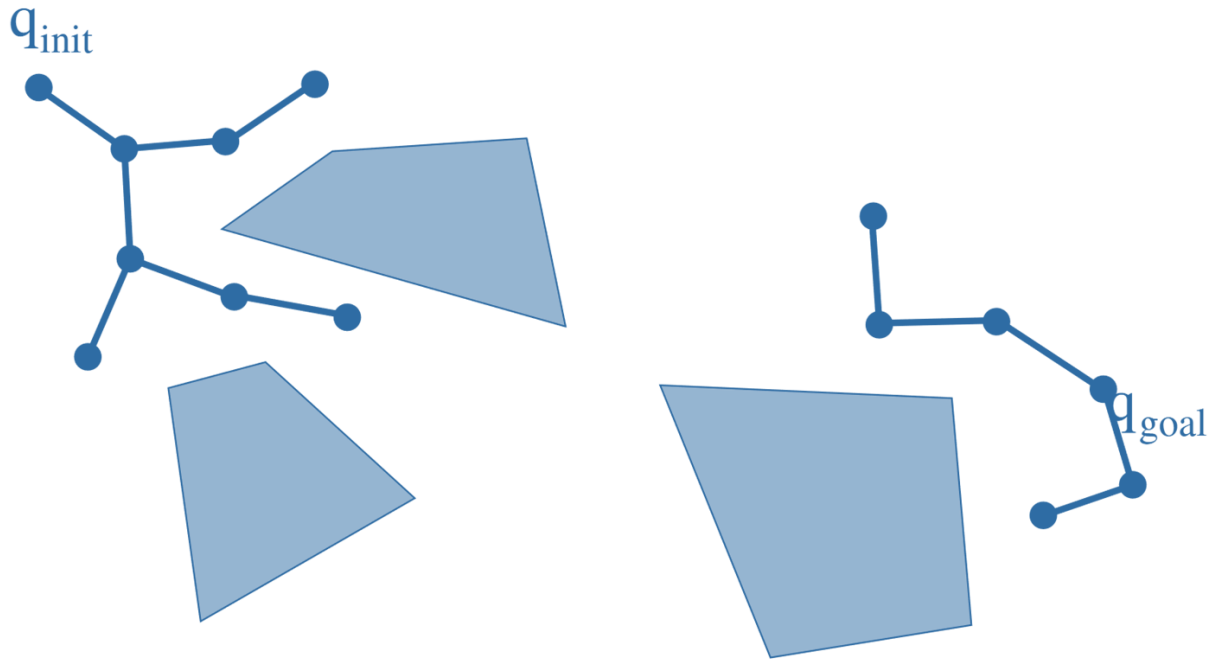
Rapidly-exploring Random Tree (RRT)



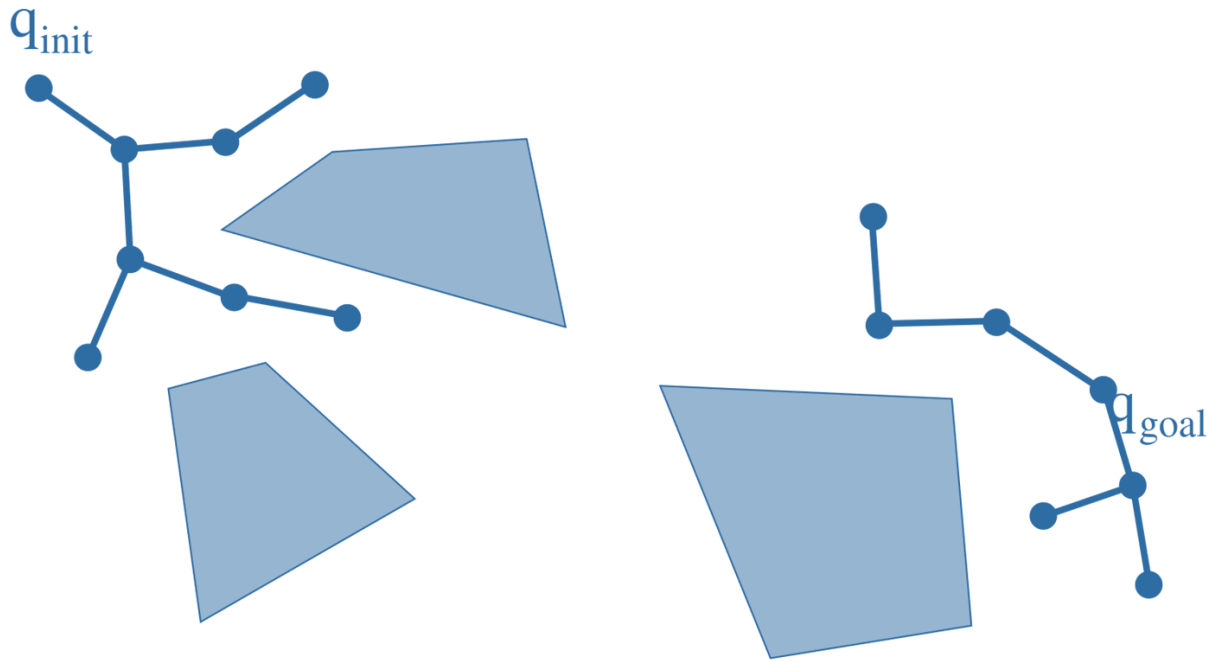
Rapidly-exploring Random Tree (RRT)



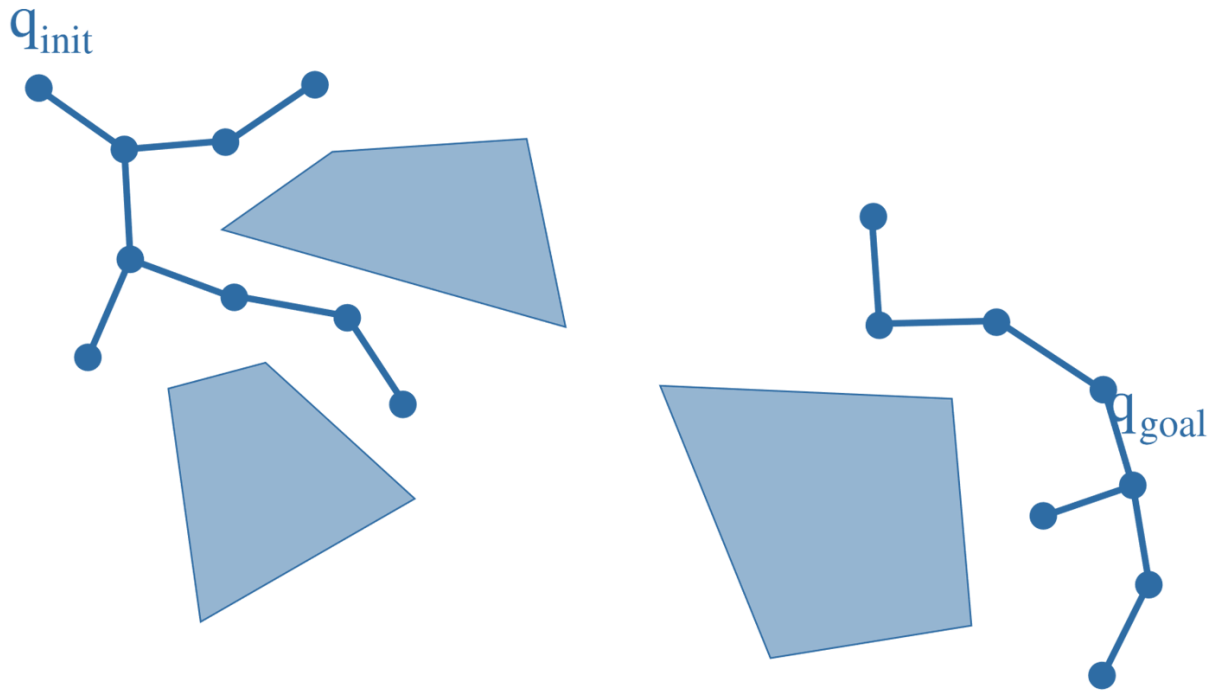
Rapidly-exploring Random Tree (RRT)



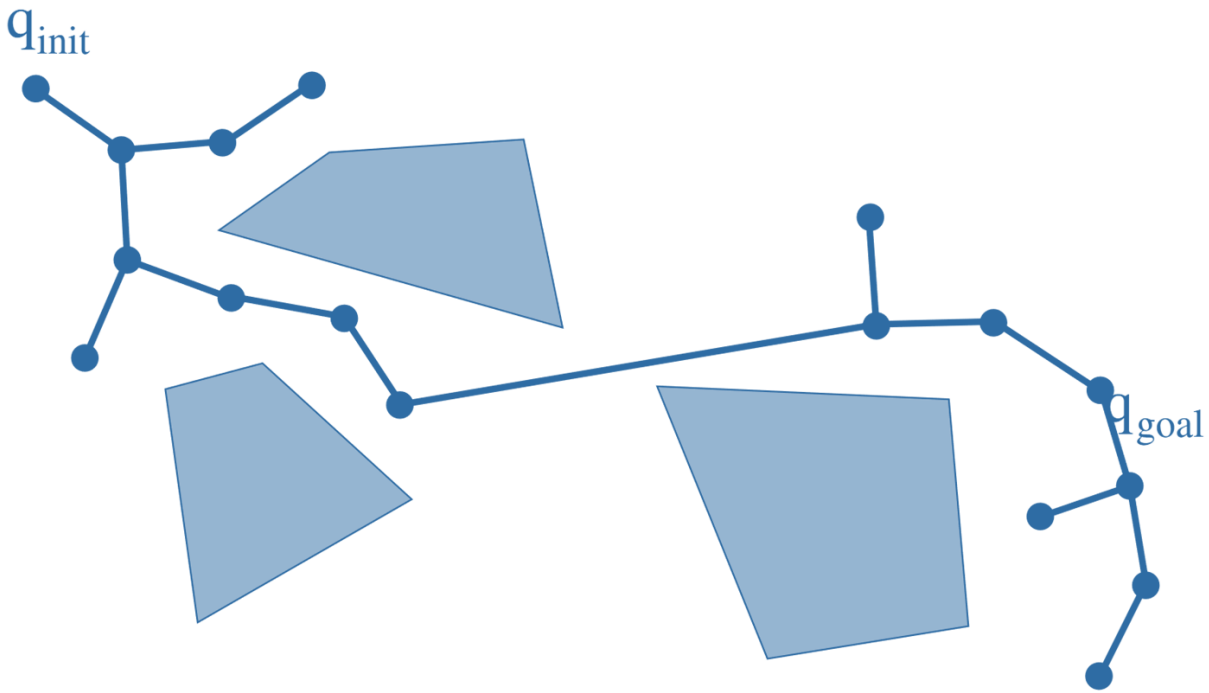
Rapidly-exploring Random Tree (RRT)



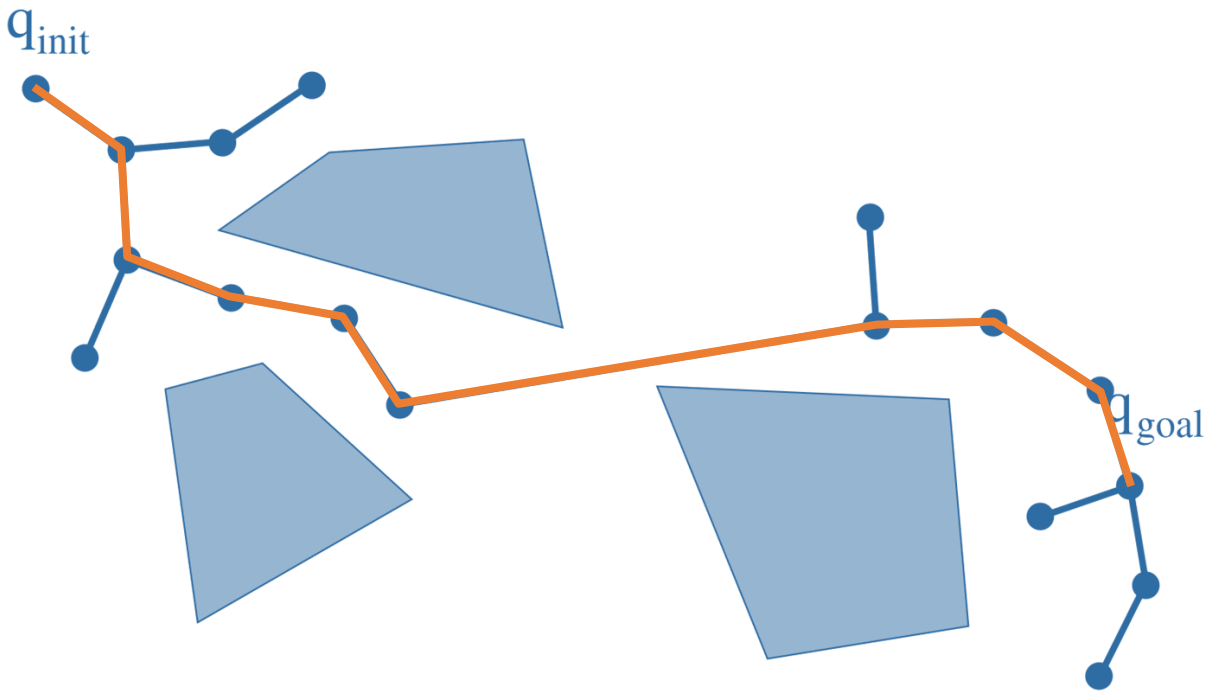
Rapidly-exploring Random Tree (RRT)



Rapidly-exploring Random Tree (RRT)



Rapidly-exploring Random Tree (RRT)



Correction, terminaison et complétude

- Complétude
  - L'algorithme peut trouver une solution en temps fini, ou rapporter en temps fini l'absence de solution
- Complétude en résolution
  - Si l'échantillonnage est fait selon une séquence déterministe dense, alors l'algorithme trouvera en temps fini une solution si elle existe
  - L'algorithme ne peut pas rapporter en temps fini l'absence de solution
- Complétude en probabilité
  - Si une solution existe, alors la probabilité pour que l'algorithme trouve la solution tend vers 1 quand le nombre d'échantillons tend vers l'infini
  - L'algorithme ne peut pas rapporter en temps fini l'absence de solution

Correction, terminaison et complétude

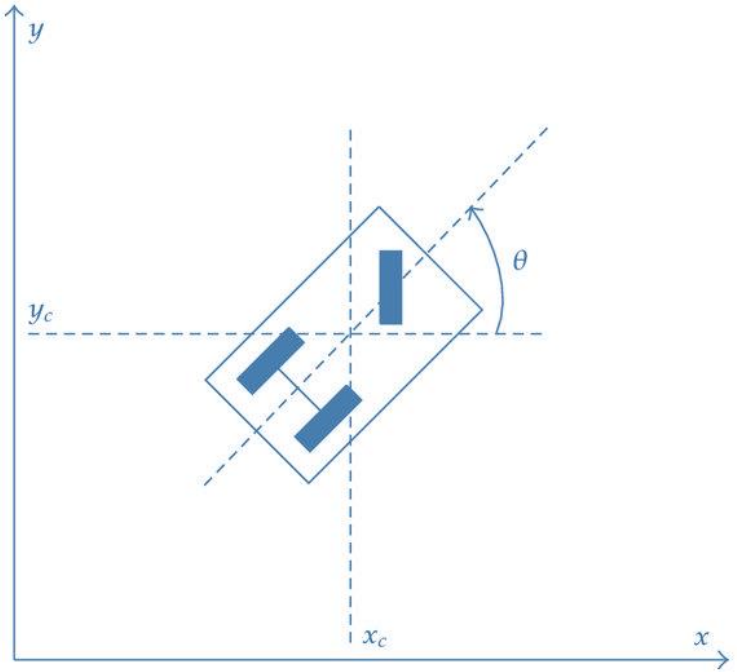
- Les **algorithmes exacts** (décomposition en cellules verticales, décomposition en cellules cylindriques ou décompsition de Collins)
  - sont **complets**

- Les **algorithmes par échantillonnage** (PRM, RRT)
  - Ne sont **pas complets** en général
  - sont **complets en probabilité** si un échantillonnage aléatoire est utilisé
  - sont **complets en résolution** si une séquence d'échantillonnage déterministe dense est utilisée

Systèmes non-holonomes

Systèmes non-holonomes

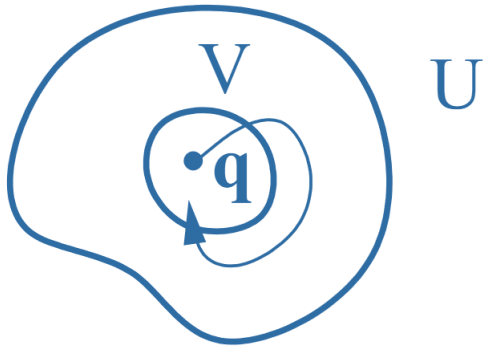
- Introduisent des contraintes différentielles dans l'espace des configurations
- Robots mobiles à roues. voitures autonomes



Illustrations de F. Lamiraux

Contrôlabilité locale des systèmes non-holonomes

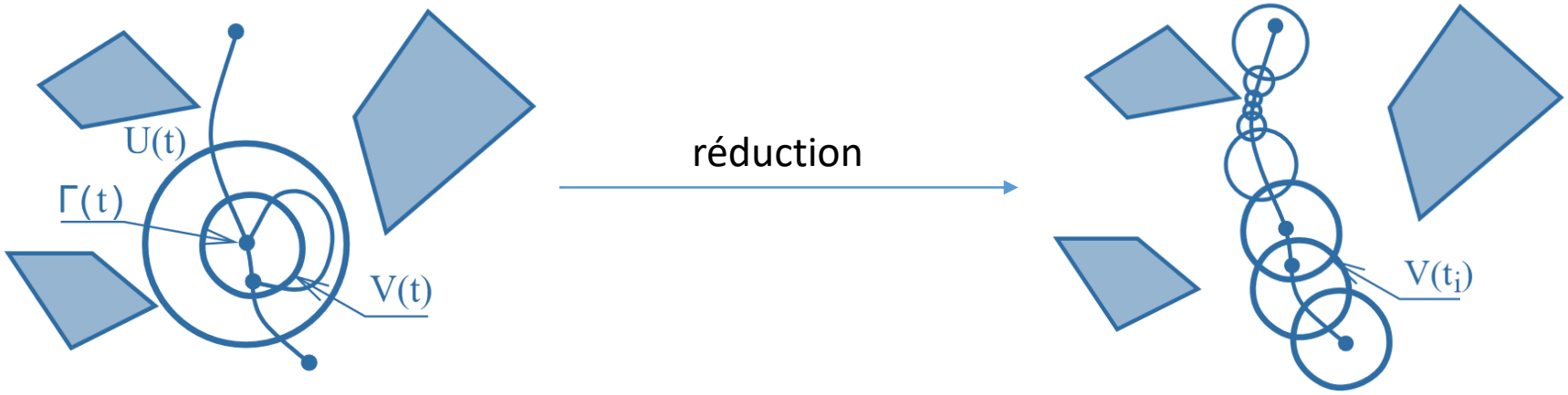
- Un système non-holonome est localement contrôlable ssi pour tout q dans $\mathcal{C}$ et pour tout voisinage U de q il existe un voisinage V de q complètement atteignable à partir de q par des chemins admissibles dans U



- On prouve la contrôlabilité locale par la construction des crochets de Lie des champs de vecteurs de contrôle (algèbre de Lie de ce champs de vecteur de dimension n)

Propriété de réduction

- Si le système est localement contrôlable, alors l'existence d'un chemin continu non contraint libre de collision implique l'existence d'un chemin admissible libre de collision



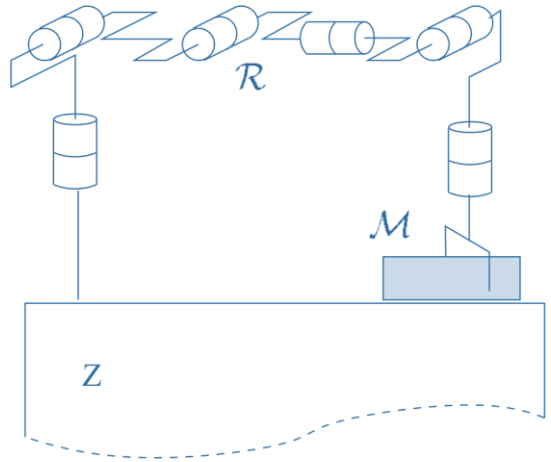
Illustrations de F. Lamiraux

Planification de manipulation

Planification de manipulation

- Planification de movement dans $SE(3)$ pour un objet manipulé $\mathcal{M}$ qui ne peut pas bouger de lui-même

- Deux états discrets : Peut soit
  - bouger indirectement lorsqu'il est saisi par un robot $\mathcal{R}$
  - ou être immobile en étant pose en equilibre statique sur un environnement $\mathcal{Z}$



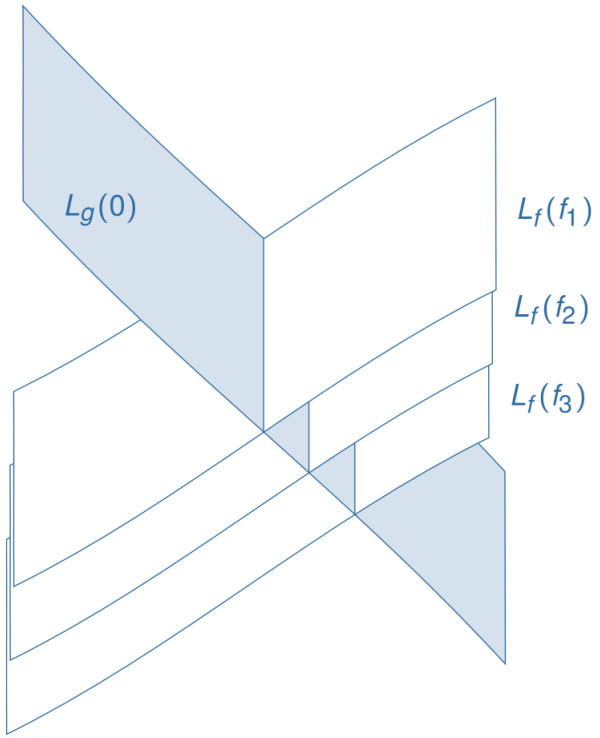
Stratification et feuilletage de l'espace des configurations

- L'espace des configurations $\mathcal{C} = \mathcal{C}_{\mathcal{R}} \times \mathcal{C}_{\mathcal{M}}$ est **stratifié en deux strates**:
  - Ensemble des configurations dans lesquelles le robot saisit l'objet
  - Ensemble des configurations dans lesquelles l'objet est posé en équilibre statique

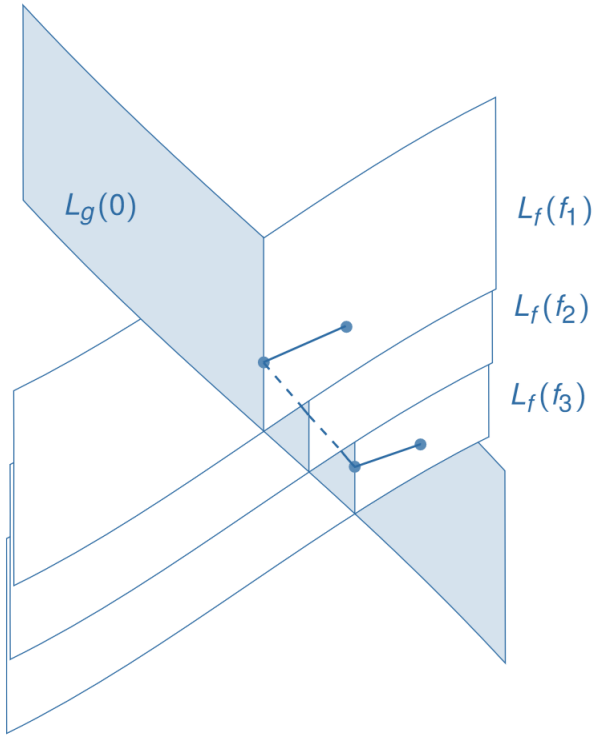
- Chaque strate est **feuilletée**
  - Le feuilletage de la strate de saisie est induit par les différentes positions de saisies du robot sur l'objet
  - Le feuilletage de la strate de repos est induit par les différentes positions de repos de l'objet

- Planification **discrète-continue**
  - Planification discrète entre les feuilletages des deux strates
  - Planification continue au sein de chaque feuille du feuilletage

Planification de manipulation



Planification de manipulation



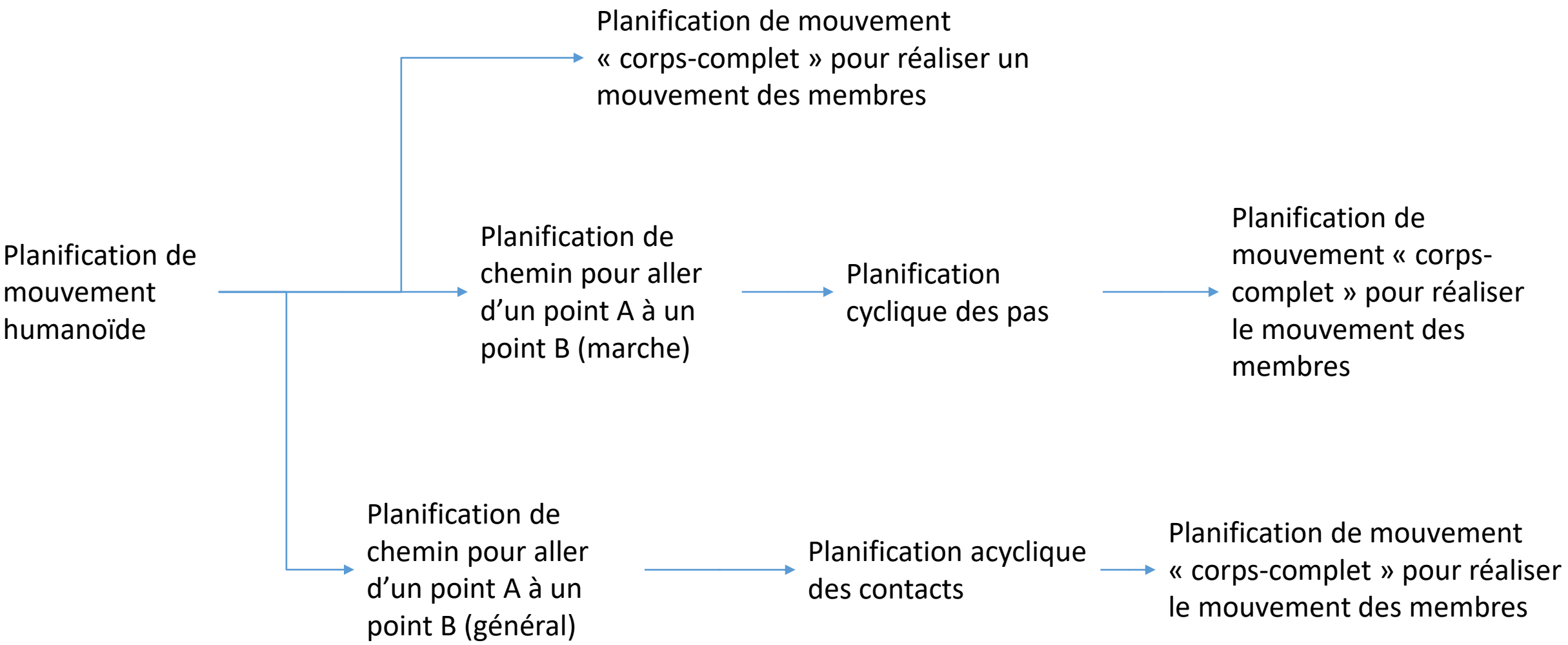
Propriété de réduction

- Sous certaines conditions sur les feuilletages : l’existence de chemin libre de collision qui viole la structure de stratification et de feuilletage implique l’existence d’un chemin admissible libre de collision

Cas du robot humanoïde : framework d’intelligence artificielle de mouvement

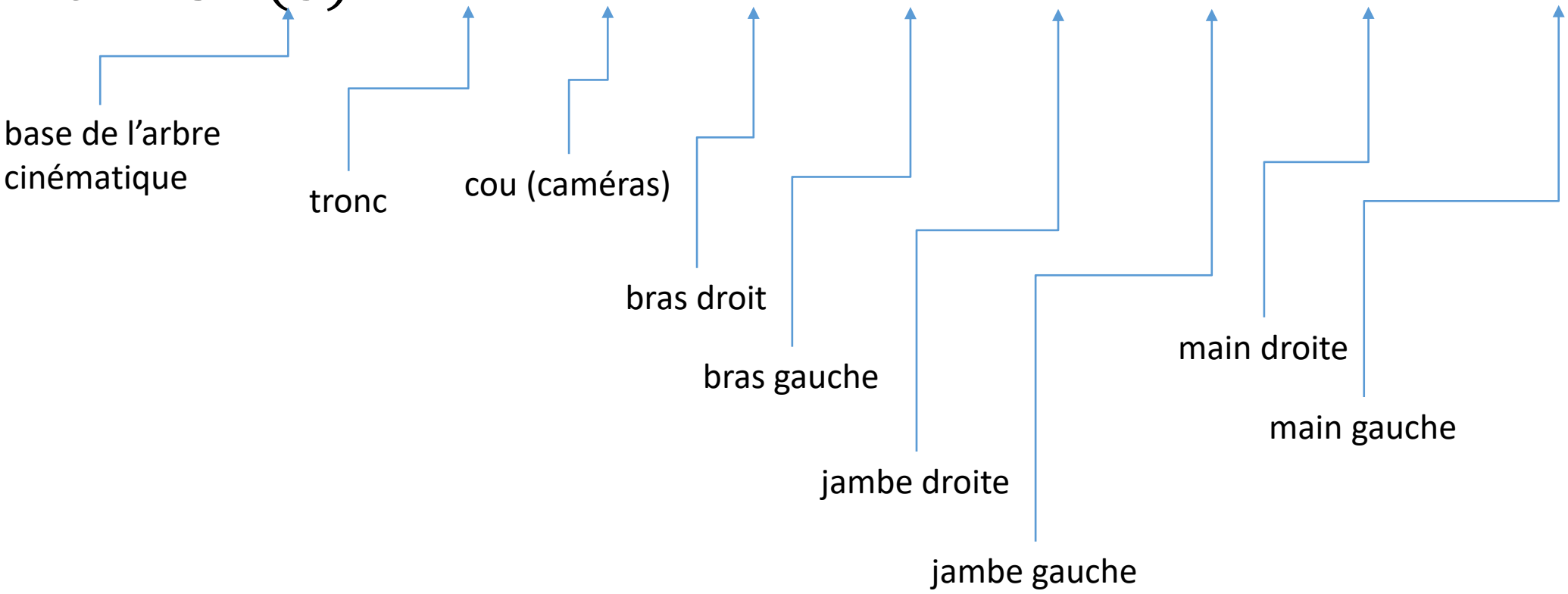
II. Robotique Humanoïde/IA Humanoïde

Planification de mouvement humanoïde



Robot humanoïde, espace des configurations

$$\mathcal{C} = SE(3) \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^7 \times \mathbb{R}^7 \times \mathbb{R}^6 \times \mathbb{R}^6 \times \mathbb{R}^{19} \times \mathbb{R}^{19}$$



Robot humanoïde, espace des configurations

- Espace de très **haute dimension** (+de 50D, en comparaison d'un bras industriel 6D ou d'un robot mobile 2D+1D)
- Espace **sous-actionné stratifié** en en autant de configurations de contacts possibles, chaque strate de mesure nulle (échantillonnage aléatoire difficile)
- Chaque strate **feuilletée** en des positions infiniment nombreuses de chaque contact
- **Problème de l'équilibre**
  - non-encodable géométriquement dans l'espace des configurations (nécessité de passage dans l'espace d'état et planification kino-dymanique)
  - Probabilité nulle qu'une configuration ou un mouvement géométriquement admissible maintiennent l'équilibre du robot (par opposition à un robot multi-pattes ou un manipulateur mobile à roues)
  - Système intrinsèquement instable (pendule inverse en équilibre instable)

Approches pour résoudre le problème

- Découplage fonctionnel : haut du corps pour la manipulation, bas du corps pour la locomotion

- Planification **bas du corps**, aka **« contrôleurs de marche »**
  - Basés sur la physique de pendule inverse, relation linéarisée entre dynamique du centre de pression et dynamique du centre de masse (Kajita et al, 2003)
  - Boucle de stabilisation bipède de bas-niveau haute fréquence

- Planification **haut du corps**, aka **« contrôleurs de manipulation »**
  - Ne pas se préoccuper de l'équilibre (faire confiance au stabilisateur)
  - Voir l'humanoïde comme deux bras manipulateurs fixes

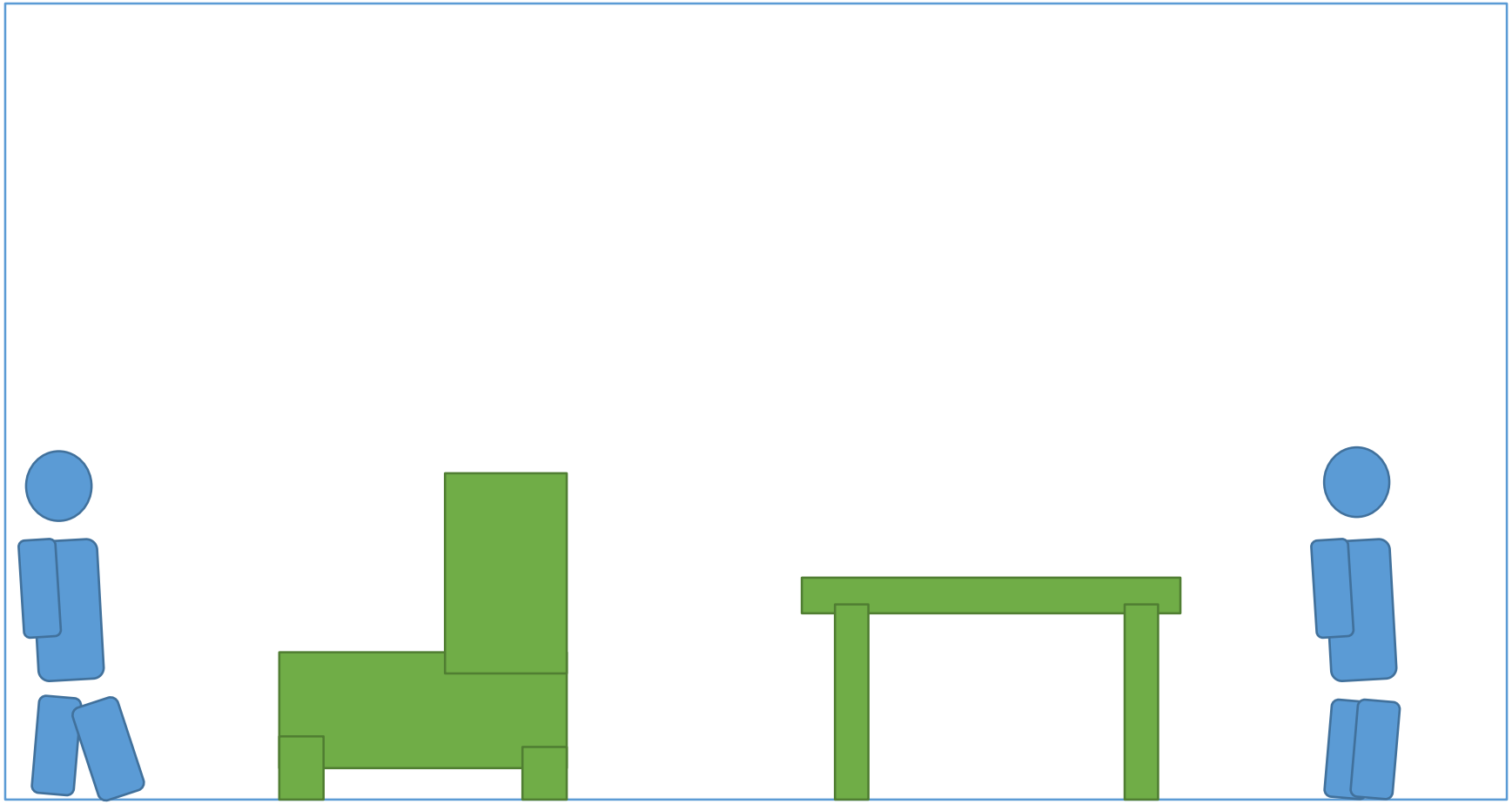
Approche multi-contact d'intelligence artificielle de mouvement

- Notre approche
  - Ne pas découpler
  - Possibilité d'utiliser le haut du corps pour la locomotion (montée d'échelle, marcher à quatre pattes)
  - Possibilité d'utiliser les jambes pour la manipulation (faire des pas supplémentaires pour faciliter la manipulation ou la rendre possible, déplacer un objet avec les jambes)
  - Aucune connaissance a priori sur la nature humanoïde du robot, juste un système de dimension élevée qui se trouve accidentellement avoir une forme humanoïde
  - Notre philosophie : **la marche devrait émerger de manière autonome** de ce framework, sans être codée en dur dans le robot comme une connaissance

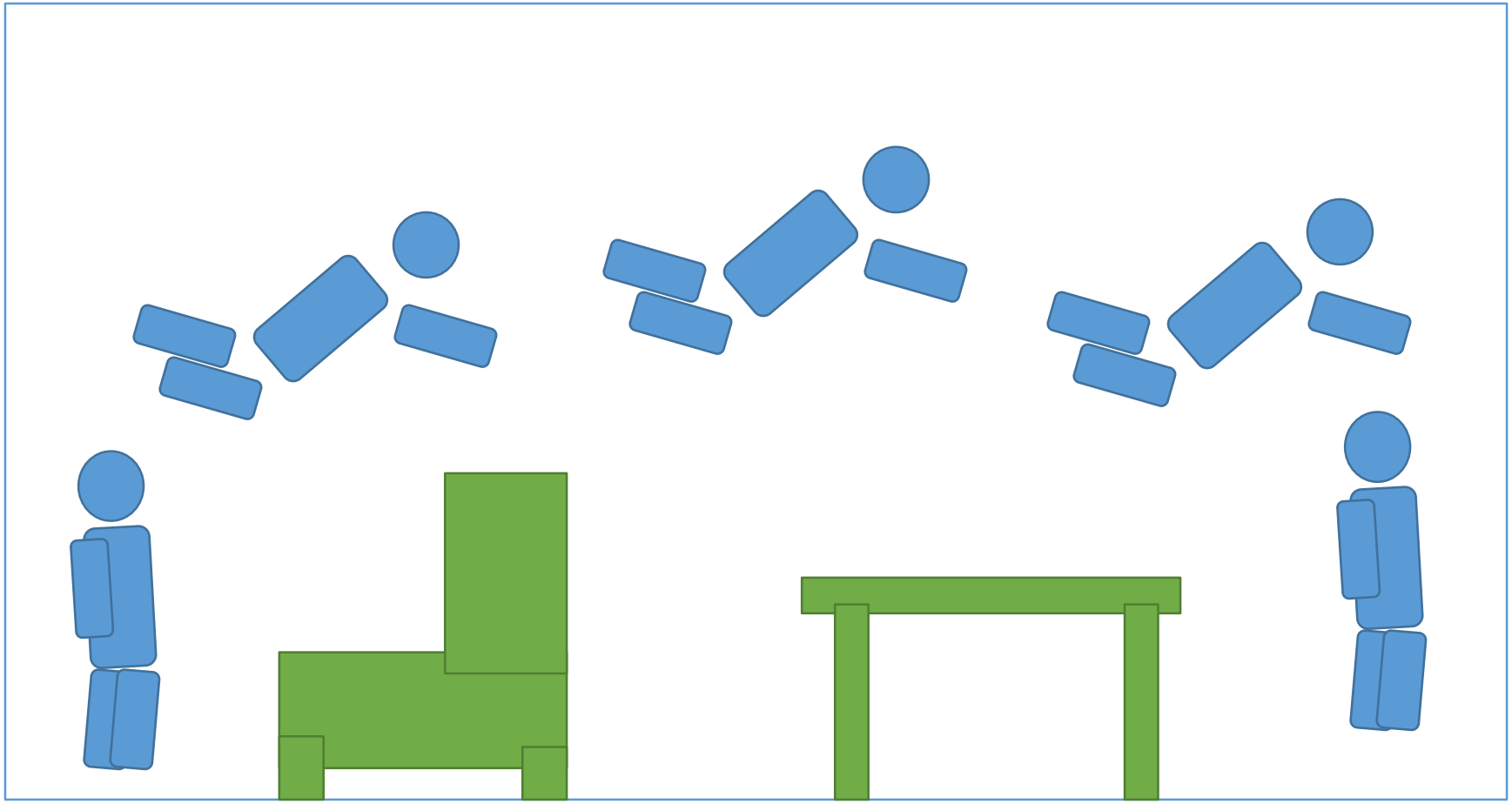
- Intelligence artificielle « générale » de mouvement

Planification multi-contact humanoïde

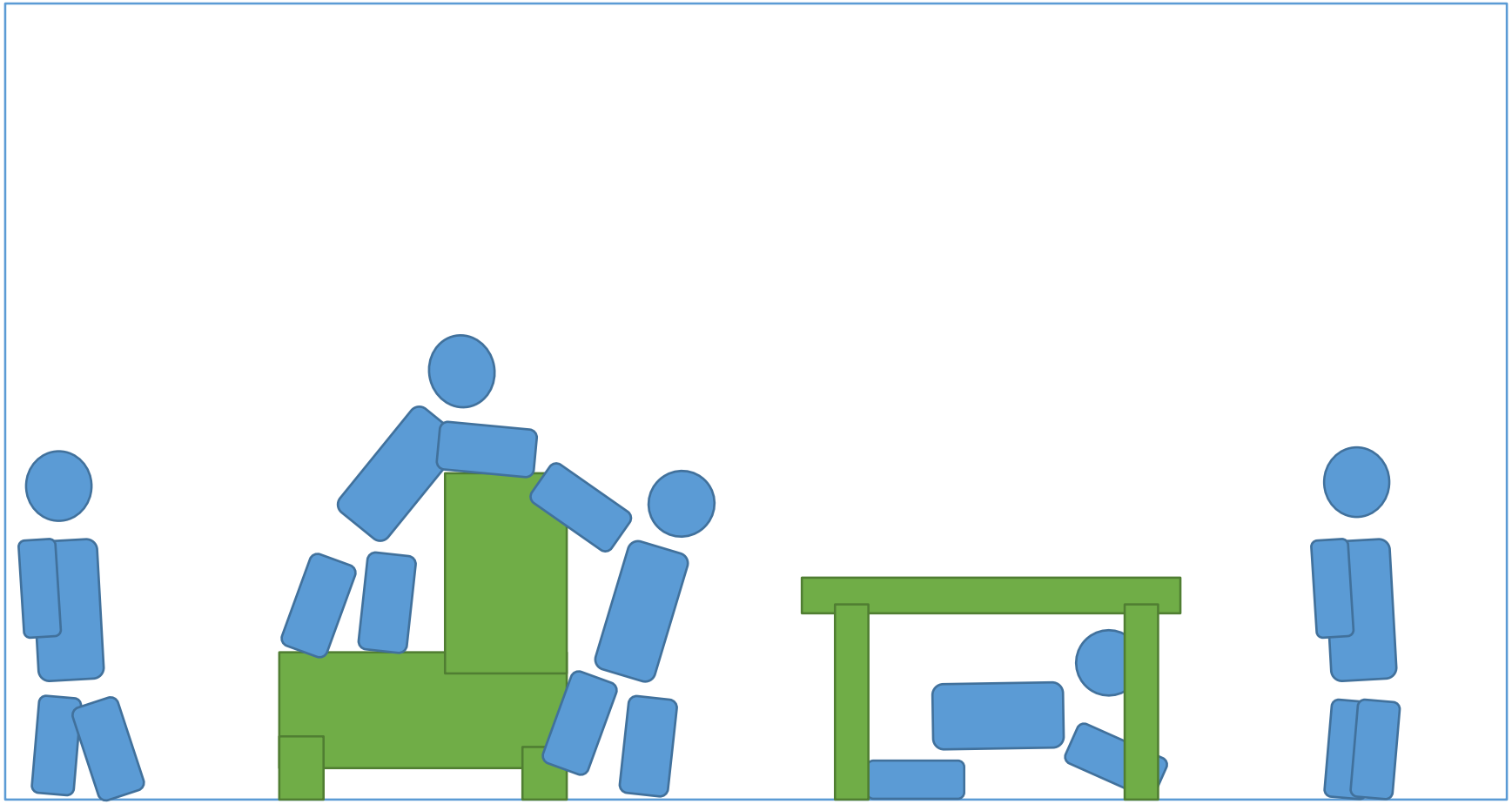
Une instance du problème



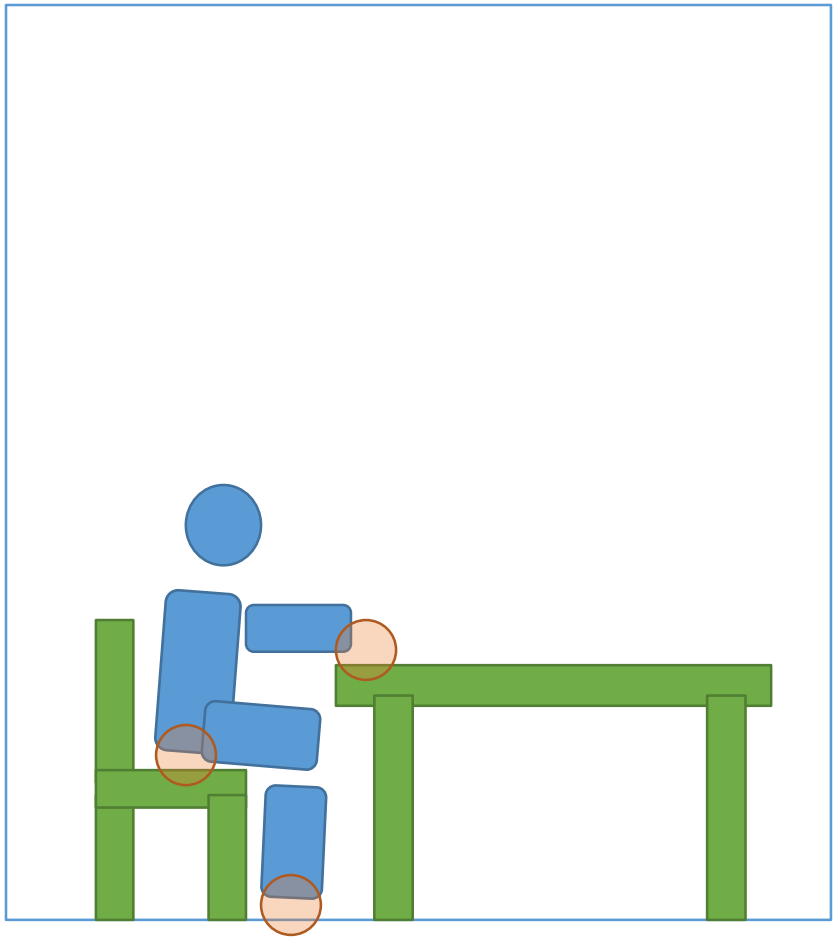
Solution par PRM/RRT

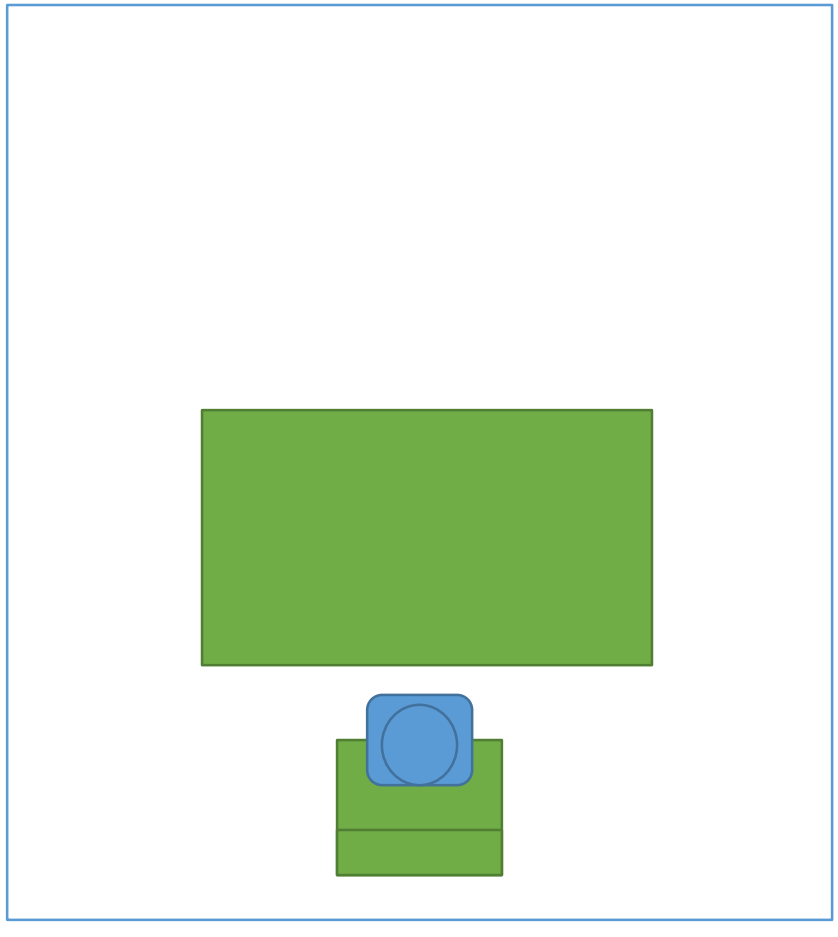


Solution par planification multi-contact

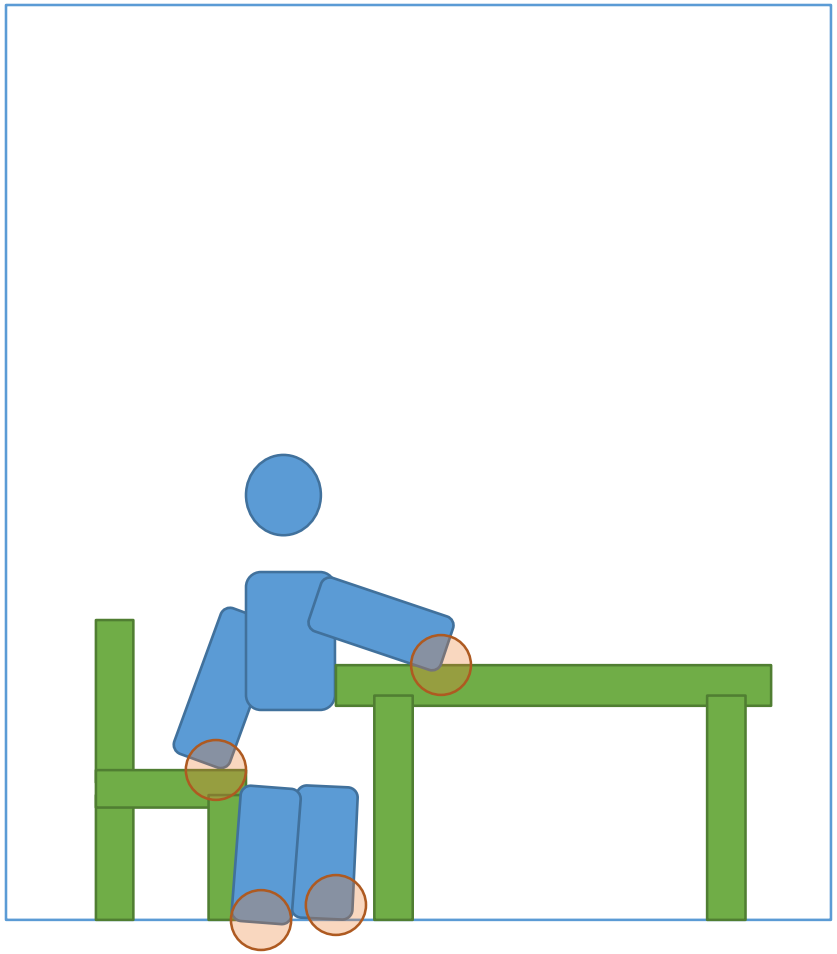


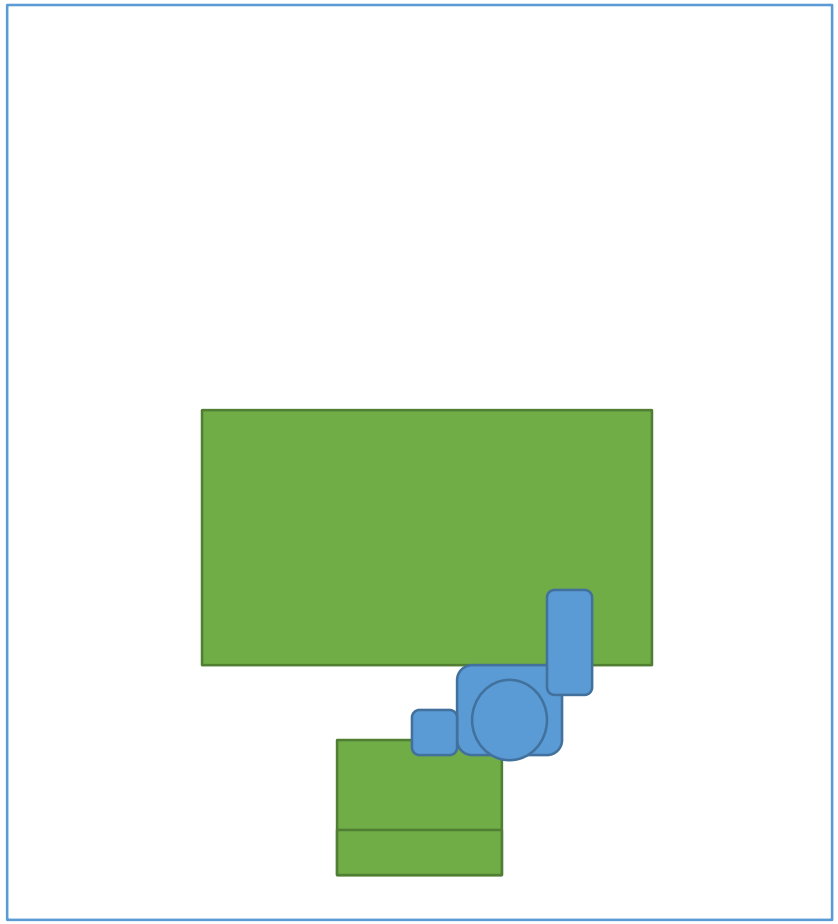
Planification multi-contact avec planification de chemin



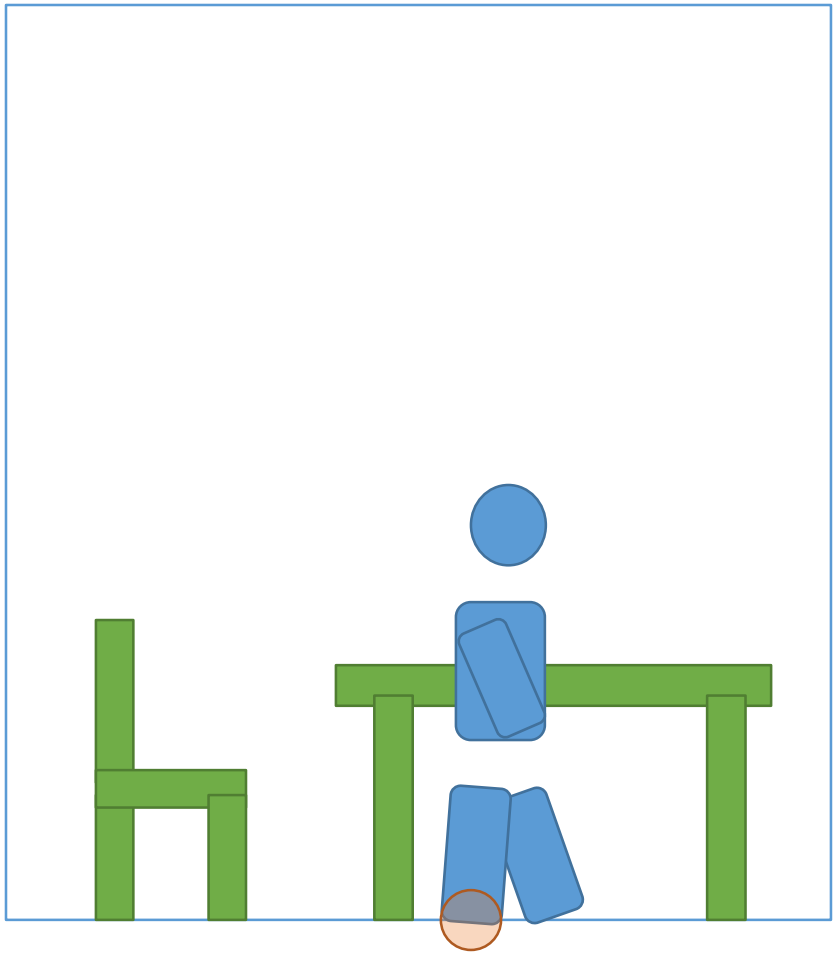


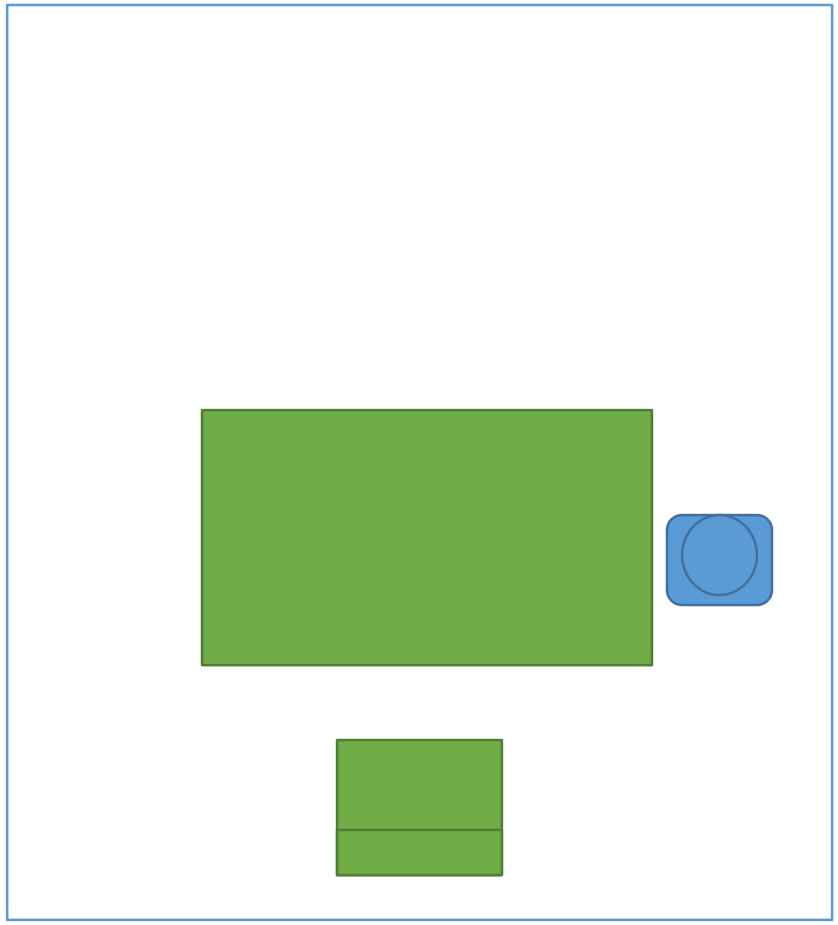
Planification multi-contact avec planification de chemin



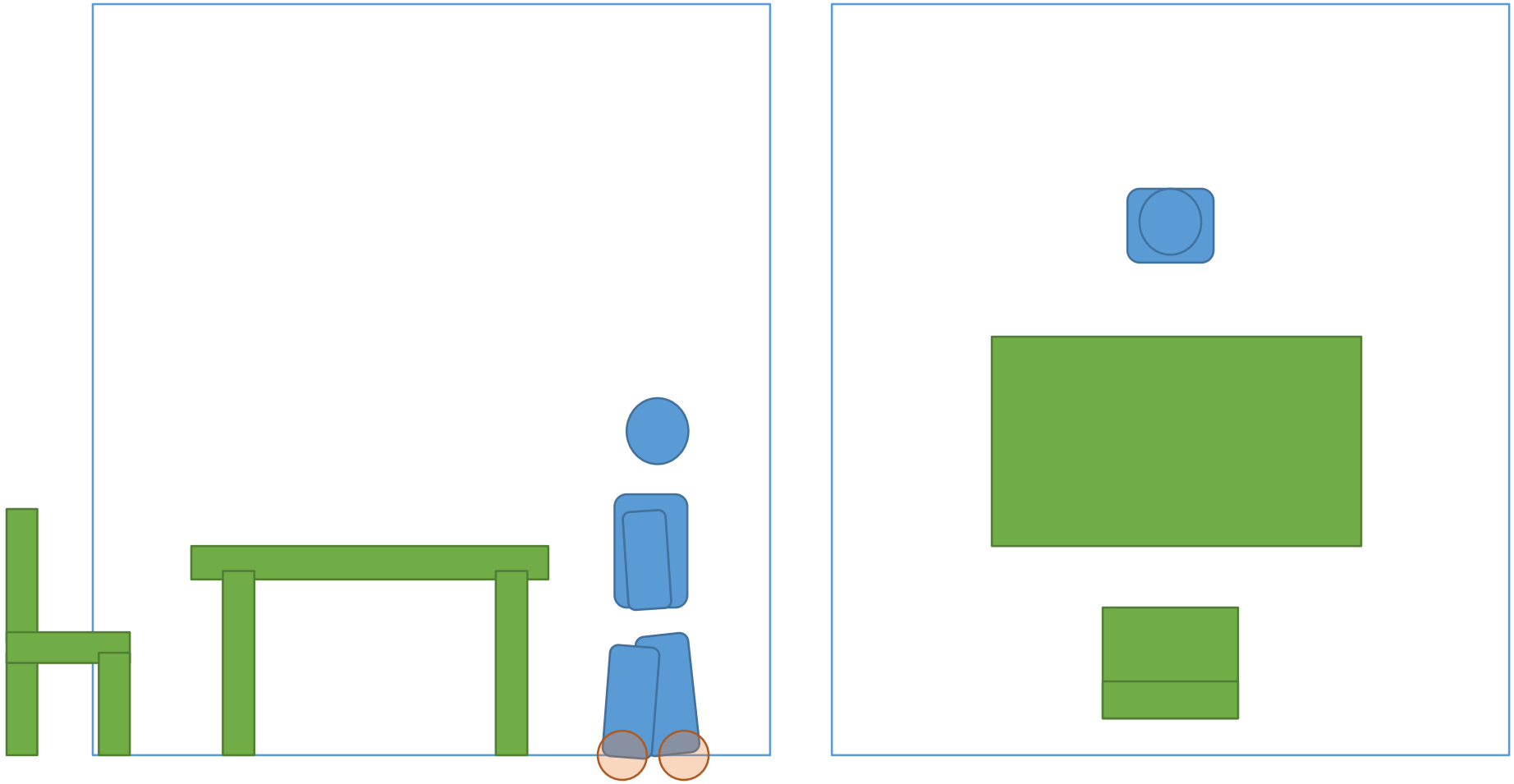


Planification multi-contact avec planification de chemin





Planification multi-contact avec planification de chemin



Formulation du problème

- La planification multi-contact induit une **stratification/feuilletages** similaires à ceux de la planification de manipulation (avec une combinatoire beaucoup plus élevée cependant)

- Une **stance** σ est un **ensemble de contacts :** Chaque feuille d'une strate du C-space correspond à une **stance** σ

- Un **contact** c est défini comme un élément de $E = \mathbb{N}^4 \times SE(2)$

- **L'ensemble des stances** est noté $\Sigma \subset 2^E$

- Deux **stances** σ et σ' sont **adjacentes** si elles diffèrent (au sens ensembliste) d'exactement un contact $\exists c \in \mathrm{E}\ \sigma = \sigma' \cup \{c\}$ ou $\sigma' = \sigma \cup \{c\}$

Formulation du problème

- Chaque configuration q du robot est mappée sur une stance σ via une fonction de **cinématique directe** $\sigma = \gamma(q)$
- inversement à chaque stance σ est associée une sous-variété de **cinématique inverse** $\mathcal{Q}_\sigma = \gamma^{-1}(\{\sigma\})$
- On s'intéresse aux sous-ensembles $\mathcal{F}_\sigma \subset \mathcal{Q}_\sigma$ des configurations physiquement admissibles (existence de forces de contacts réalisant statiquement ou dynamiquement la configuration)
- Une **suite admissible de stances** (chemin dans Σ) est une suite $(\sigma_i)_{i\in\{1,\dots,k\}}$ telle que $\forall i \in \{1,\dots,k-1\}$ σ_i est adjacent à σ_{i+1} et $\mathcal{F}_{\sigma_i} \cap \mathcal{F}_{\sigma_{i+1}} \neq \emptyset$. (la suite des configurations $q_i \in \mathcal{F}_{\sigma_i} \cap \mathcal{F}_{\sigma_{i+1}}$ est alors appelée **suite des configurations de transitions**)

- Problème de planification de contacts pour une requête de planification (q_{init}, q_{goal}):

trouver une suite admissible de stances $(\sigma_i)_{i\in\{1,\dots,k\}}$ telle que $f(\sigma_1) = q_{init}$ et $f(\sigma_k) = q_{goal}$ (et exhiber la suite des configurations de transitions)

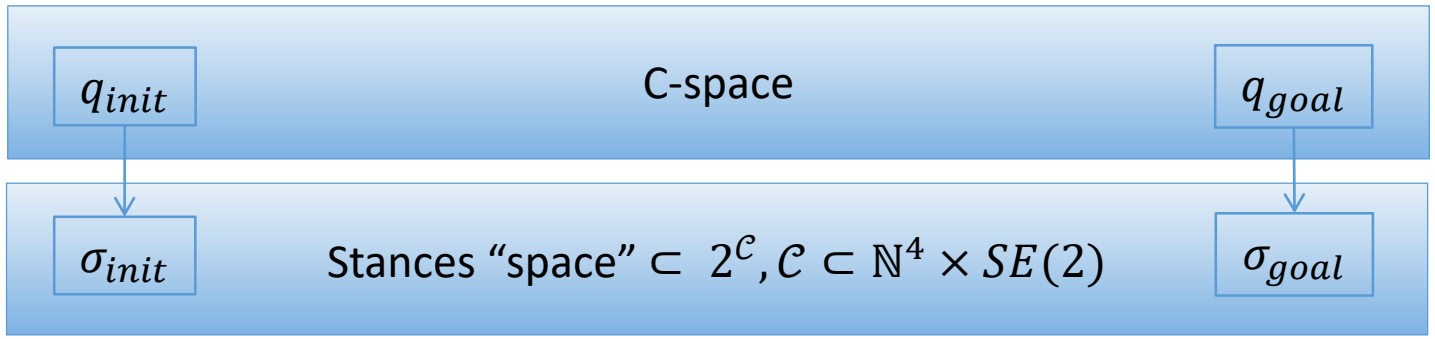
Framework général d’intelligence artificielle de mouvement humanoïde

Requête de planification : q_{init}, q_{goal}

q_{init}

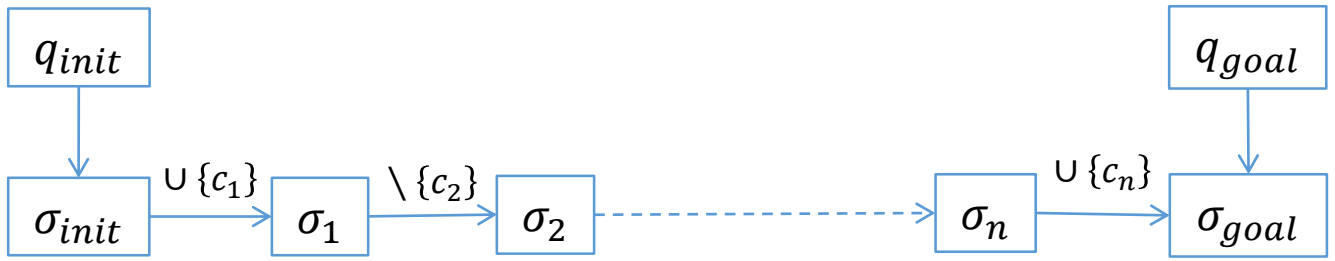
q_{goal}

Requête de planification : q_{init}, q_{goal}

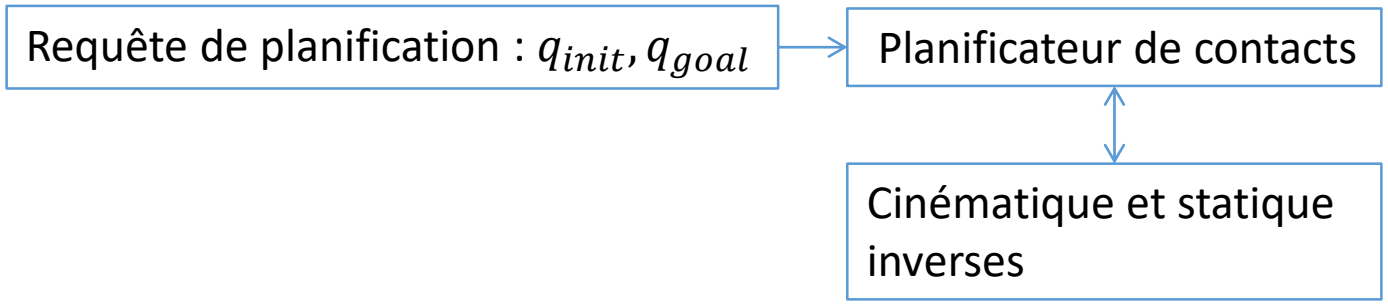


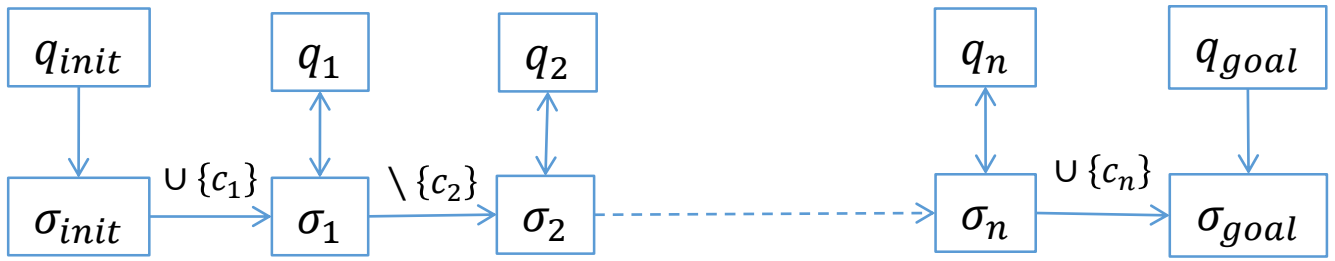
stance σ = ensemble de contacts = état de contact

Requête de planification : q_{init}, q_{goal} → Planificateur de contacts

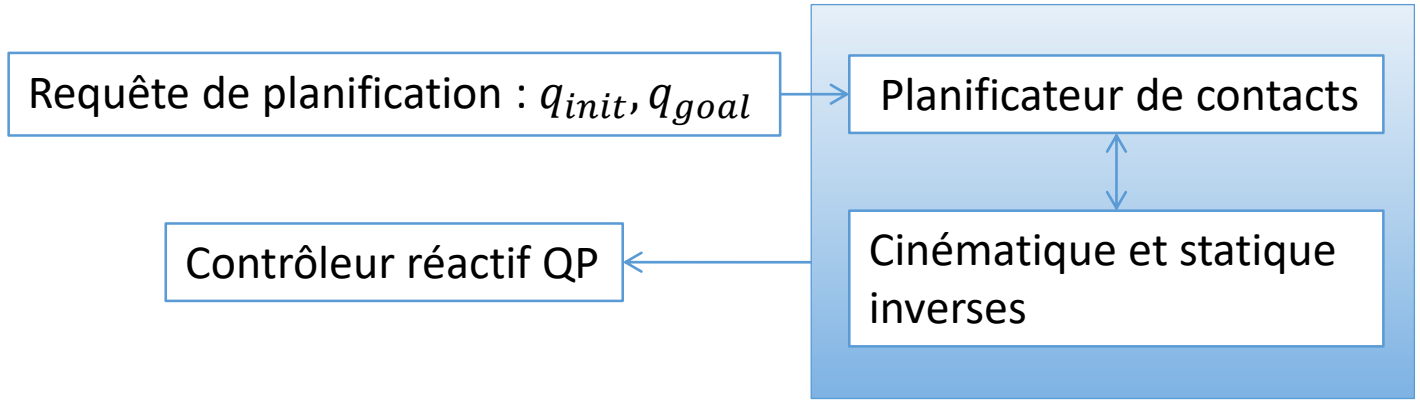


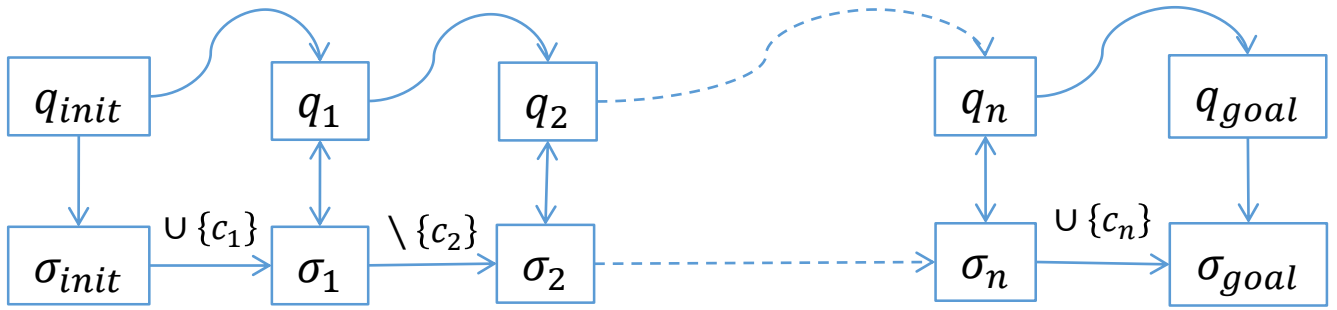
stance σ = ensemble de contacts = état de contact



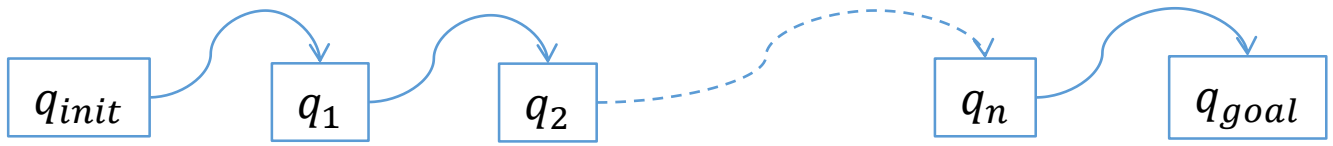


stance σ = ensemble de contacts = état de contact

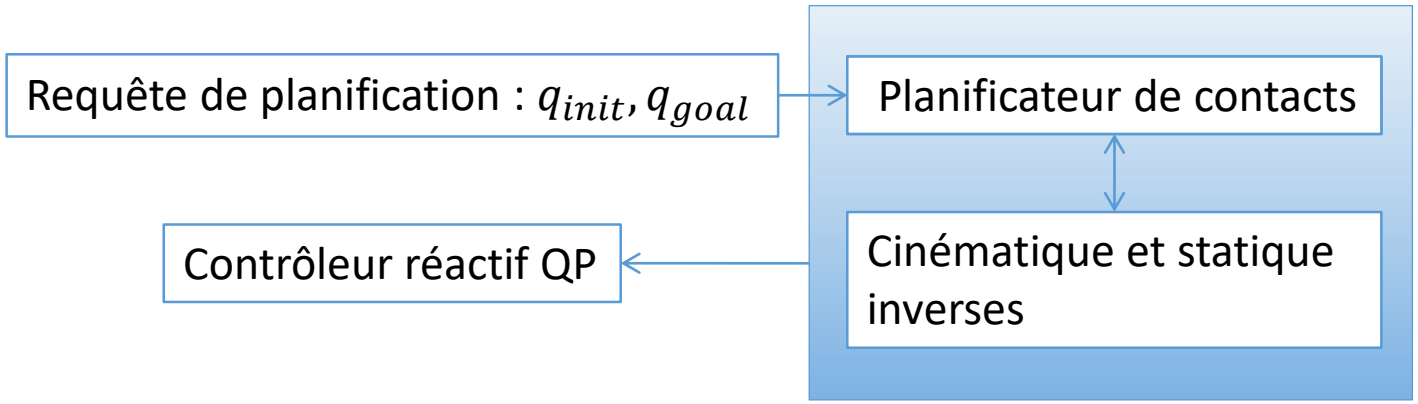




stance σ = ensemble de contacts = état de contact

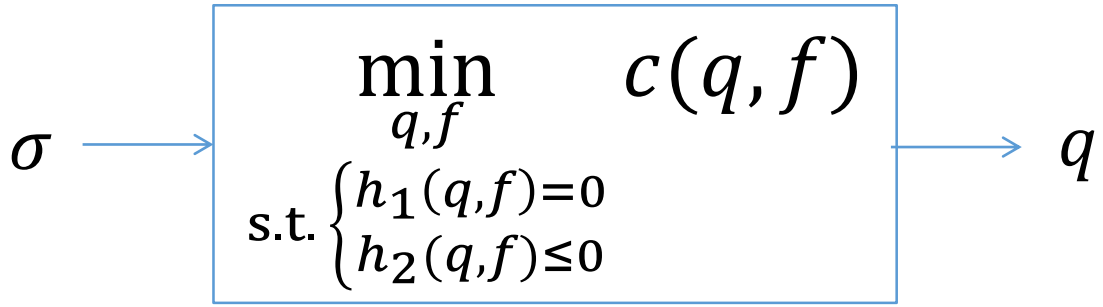


Mouvement final

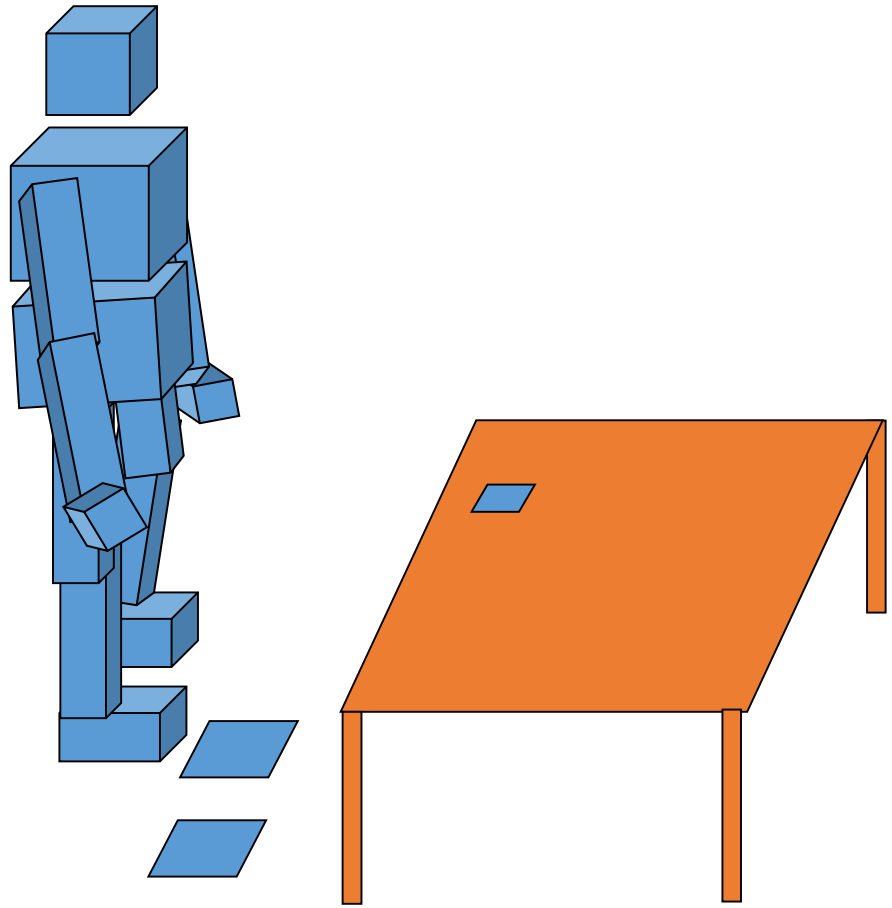

Requête de planification : q_{init}, q_{goal}
Planificateur de contacts
Cinématique et statique inverses
Contrôleur réactif QP

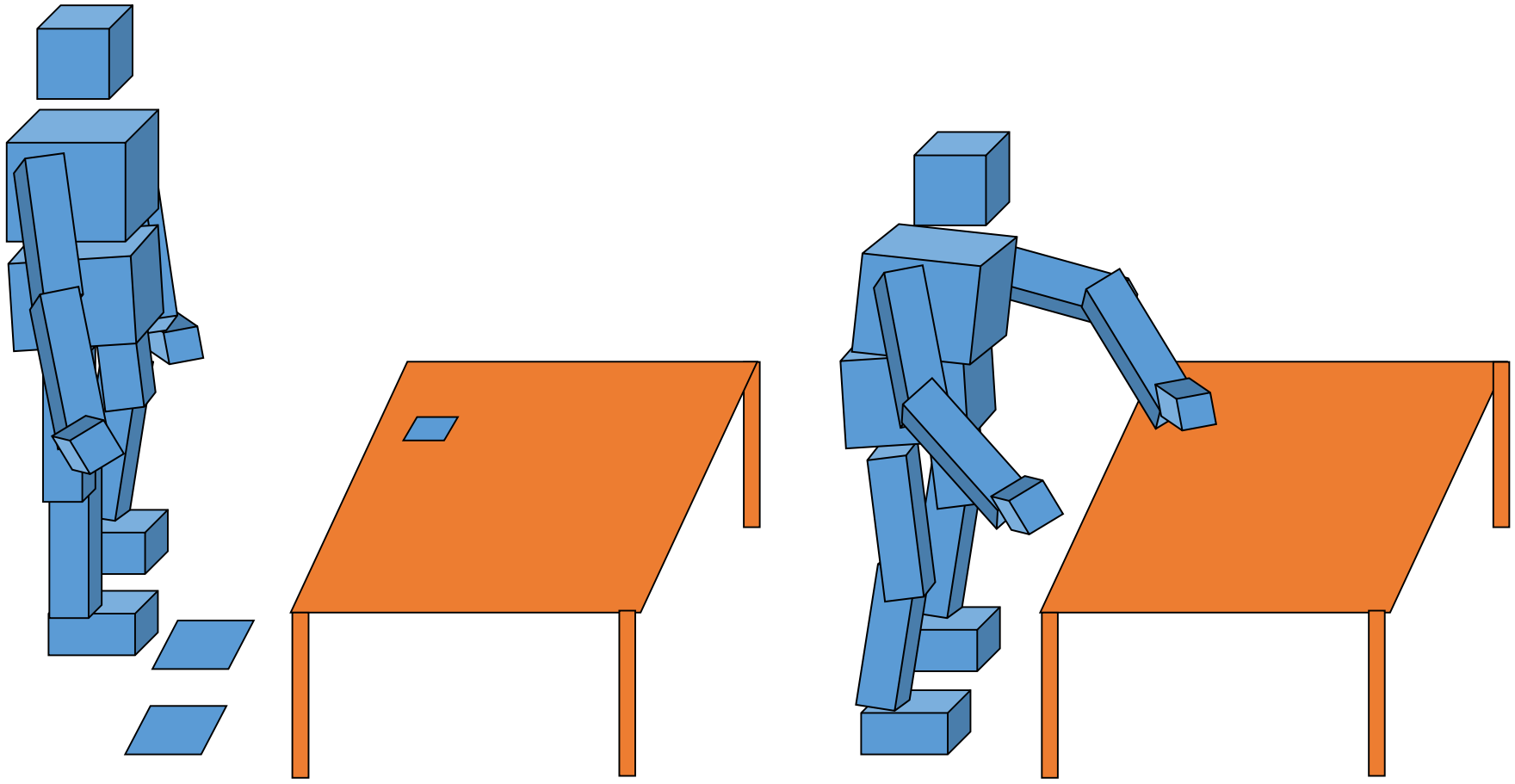
Cinématique et statique inverses

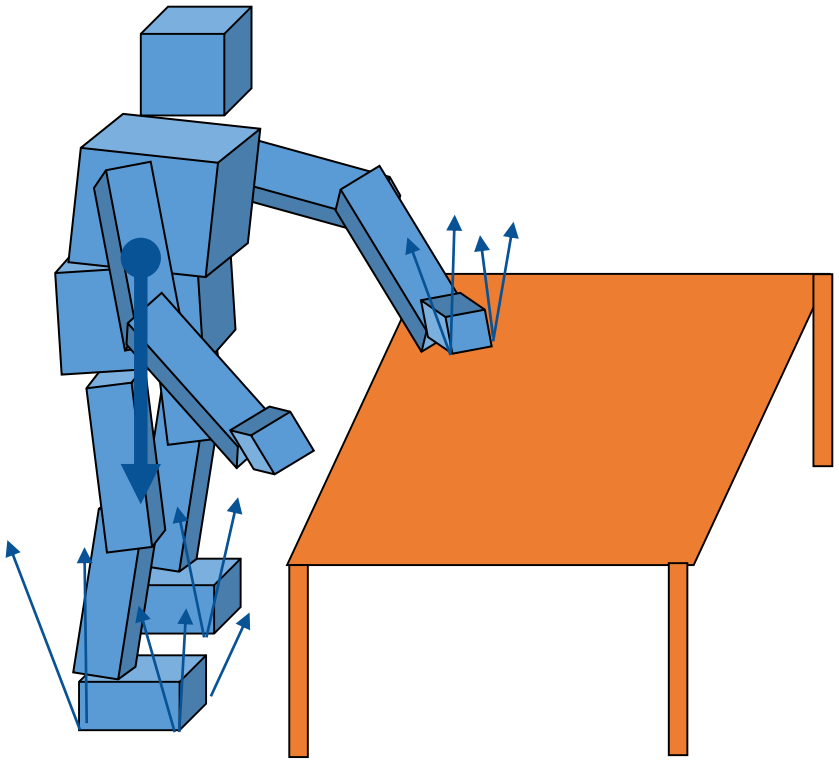
Solveur statique inverse : une optimisation non-linéaire sous contraintes égalité et inégalité

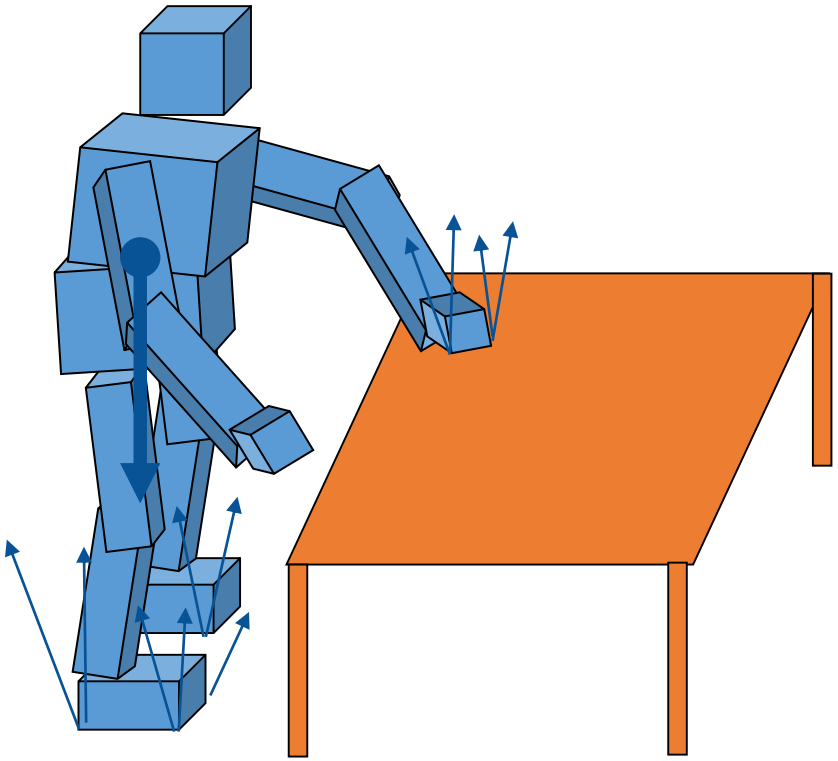


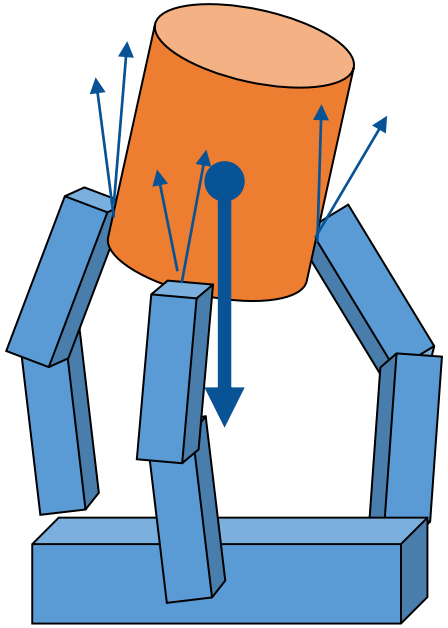
- $c(q,f)$ distance de q à une posture neutre de référence q_{ref} + norme of f
- $h_1(q,f) = 0$ position des contacts fixes, equation d'équilibre statique (base flottante)
- $h_2(q,f) \leq 0$ position des contacts flottants, limites aritculaires, equation d'équilibre statique(couple limites), cone de frication, évitement de collision

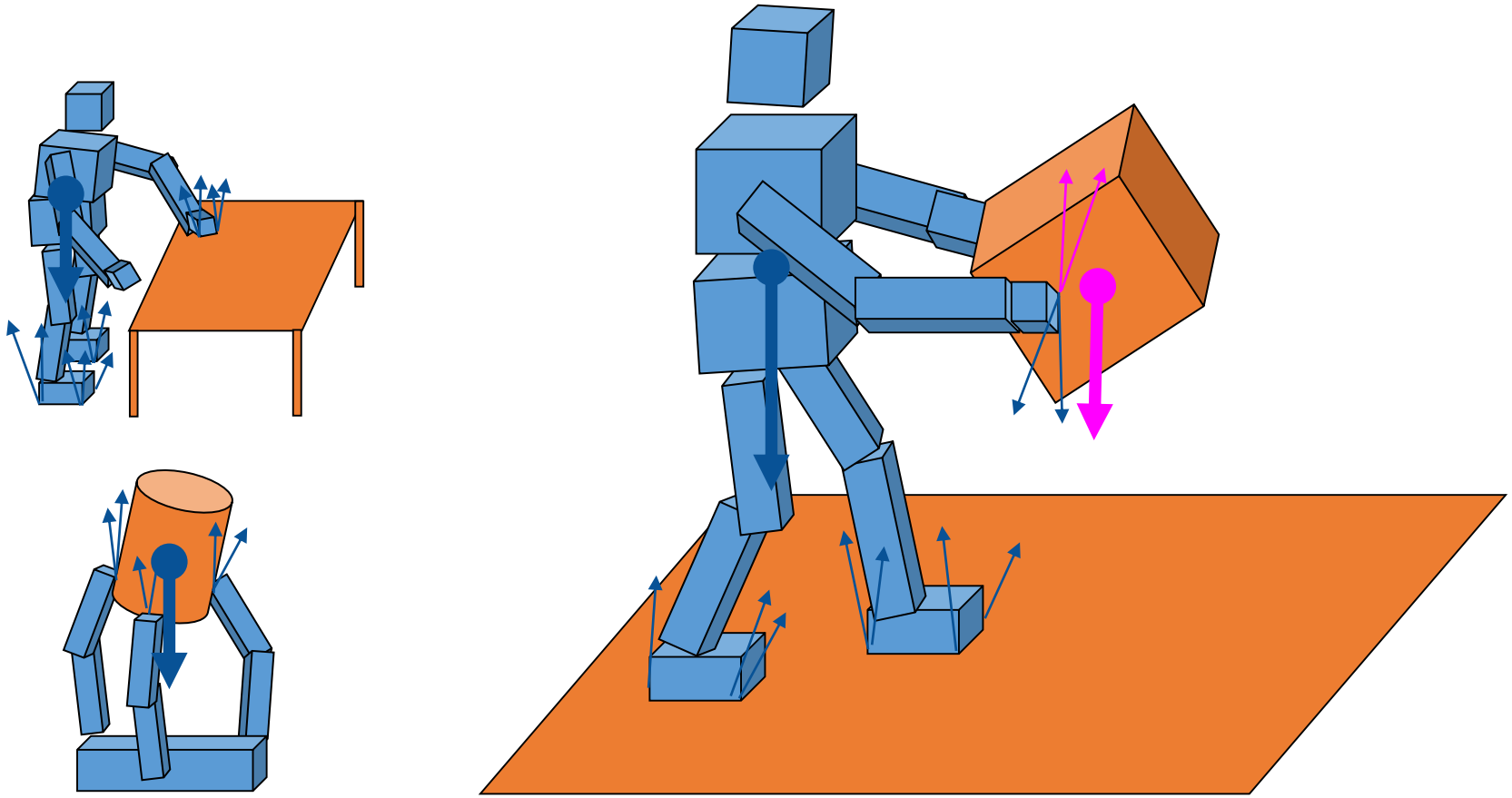


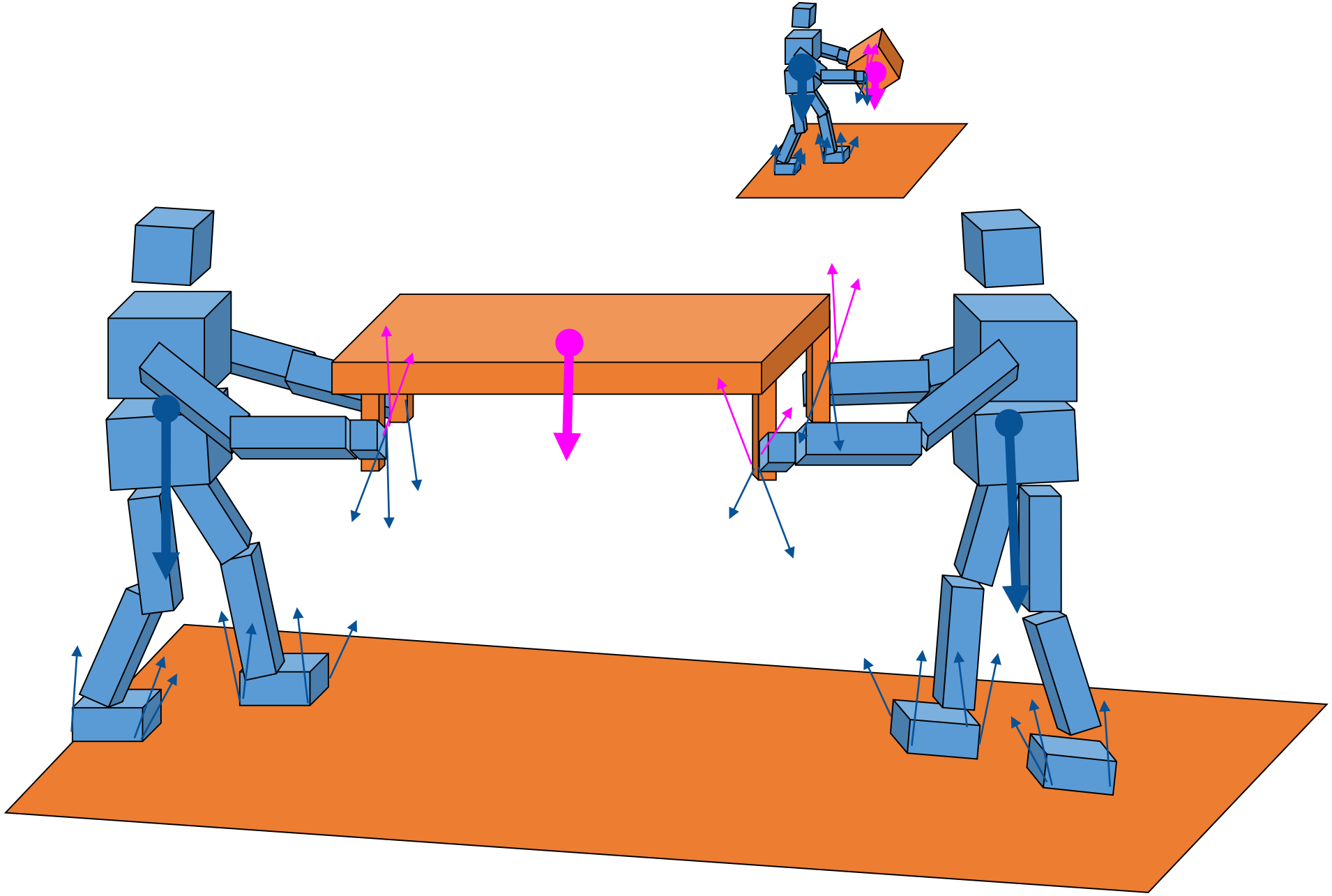




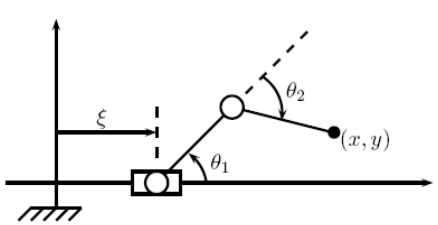


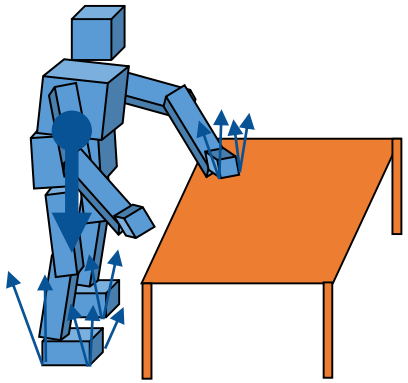




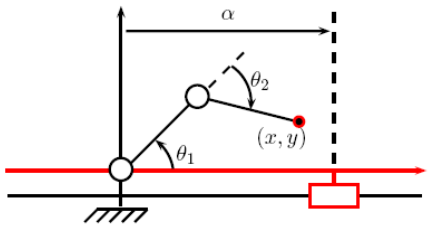


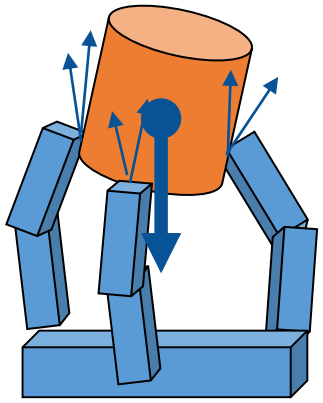
Locomotion



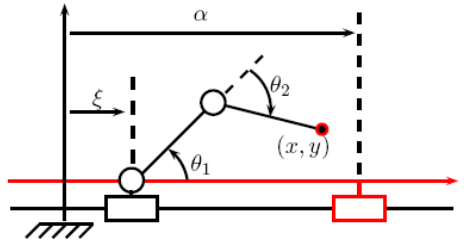


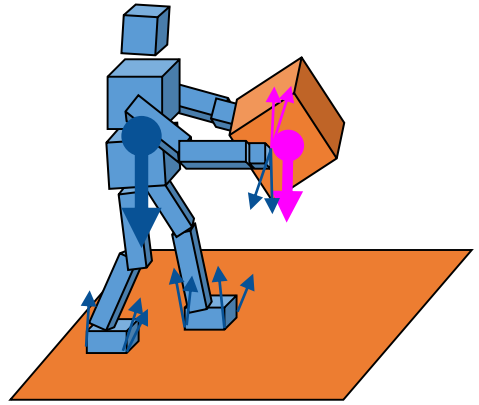
Manipulation

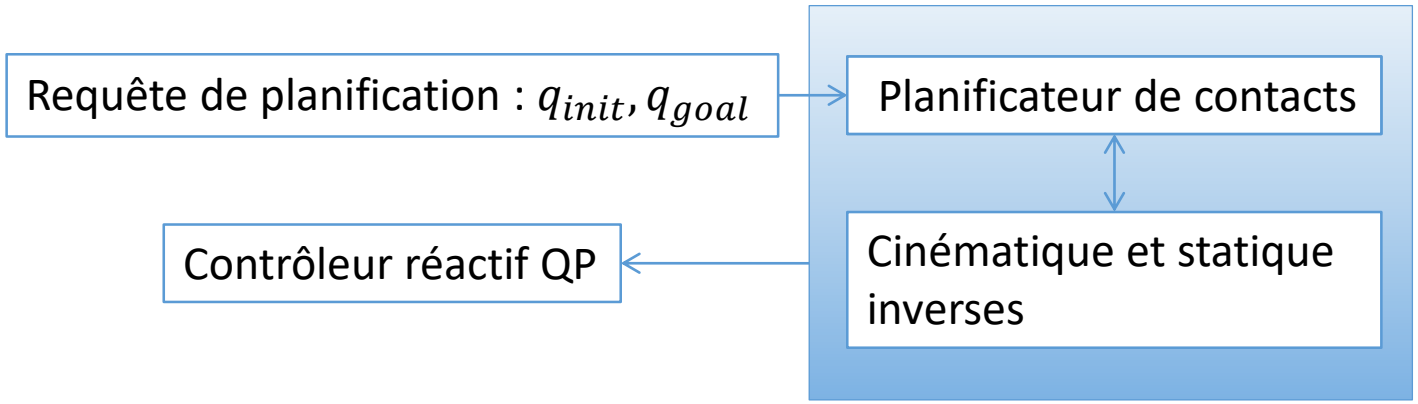




Locomotion-and-Manipulation






Requête de planification : q_{init}, q_{goal}
Planificateur de contacts
Cinématique et statique inverses
Contrôleur réactif QP

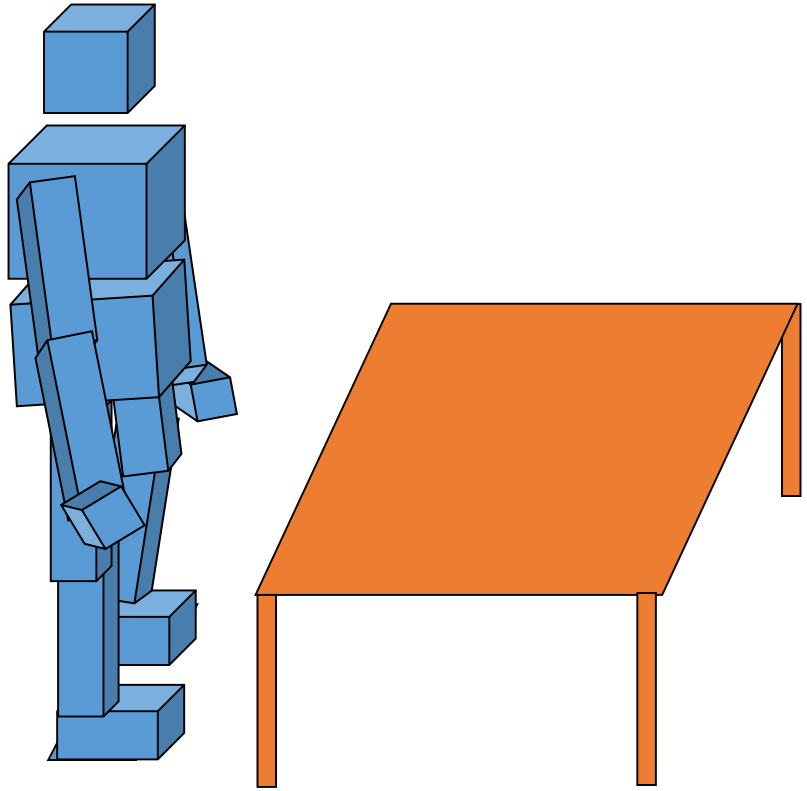
Planificateur de contacts

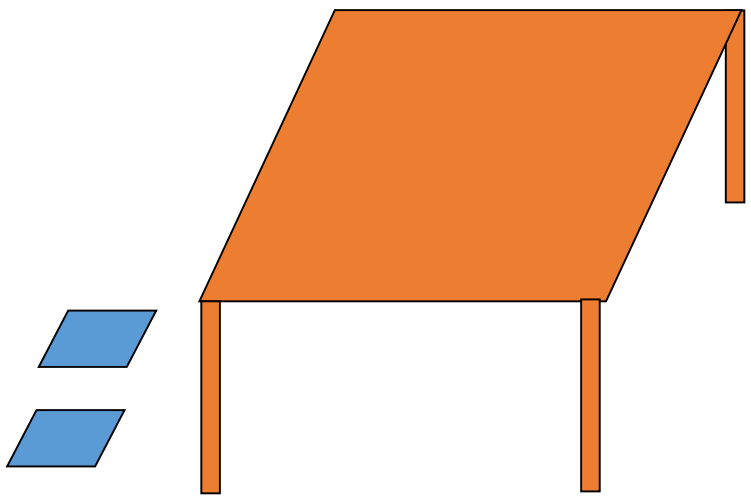
Algorithme de recherche d'une séquence de stances admissibles

Algorithme Best-First

Intitialiser un arbre de recherche et une file de priorité avec la stance initale

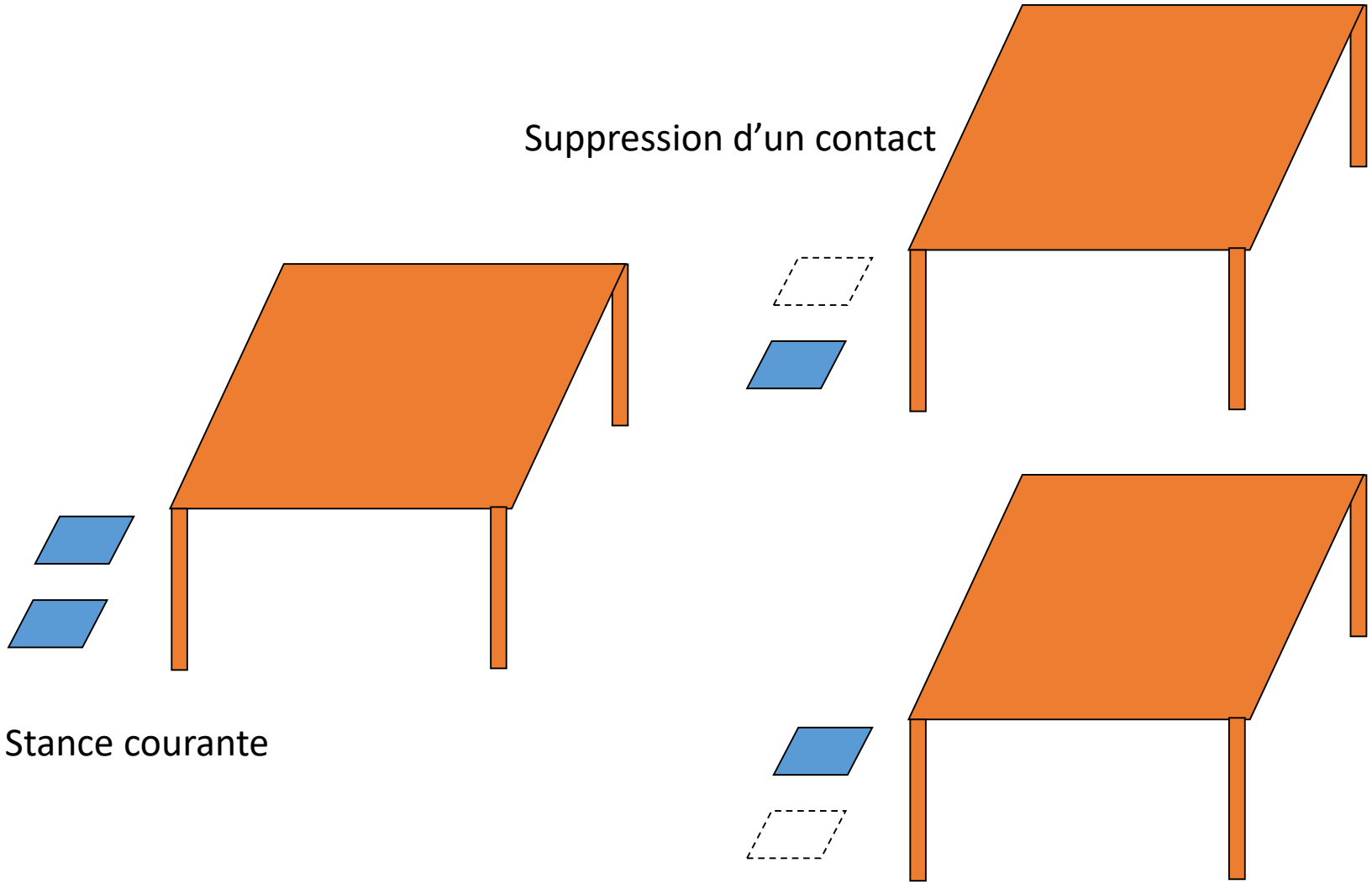
- Boucle:
  - Extraire (pop) la "meilleure" stance de la file de priorité
  - Pour tout contact dans cette stance courante
    - Retirer le contact et tester si le robot peut s'équilibrer sur la stance réduite du contact (CSI)
    - Si vrai, insérer (push) la stance réduite dans la file de priorité et l'ajouter comme fils de la stance courante dans l'arbre
  - Pour tout corps non sollicité dans la stance courante
    - Ajouter différentes paires de contact entre ce corps et différentes surfaces de l'environment et tester si le robot peut l'atteindre en état d'équilibre (CSI)
    - Si vrai, insérer (push) la stance augmentée dans la file de priorité et l'ajouter comme fils de la stance courante dans l'arbre
- Jusqu'à atteindre la stance goal



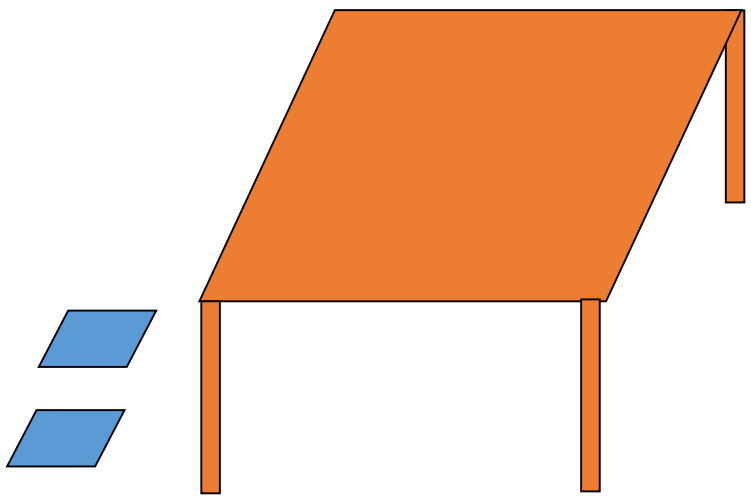


Stance courante

Suppression d’un contact

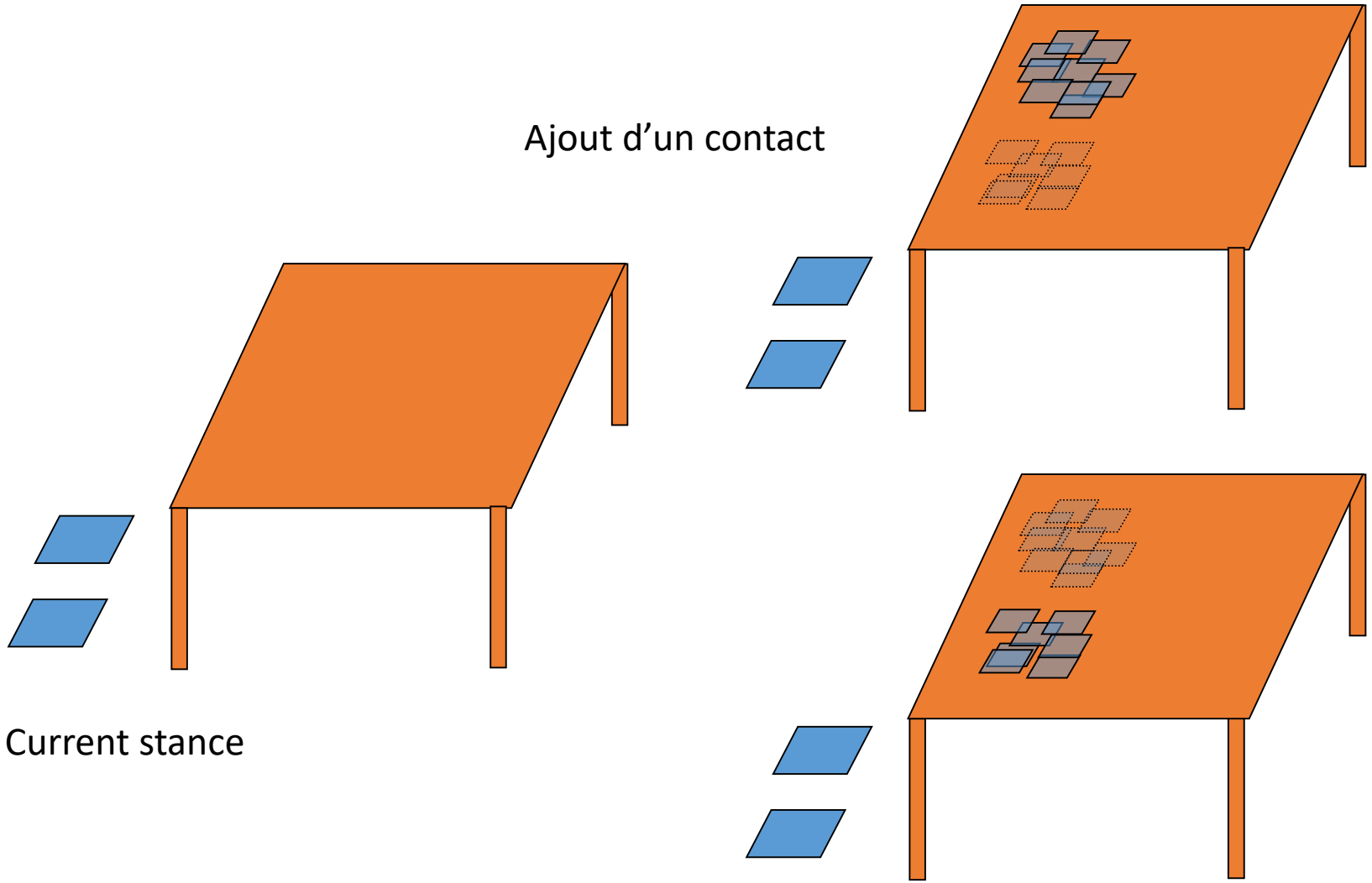


Stance courante



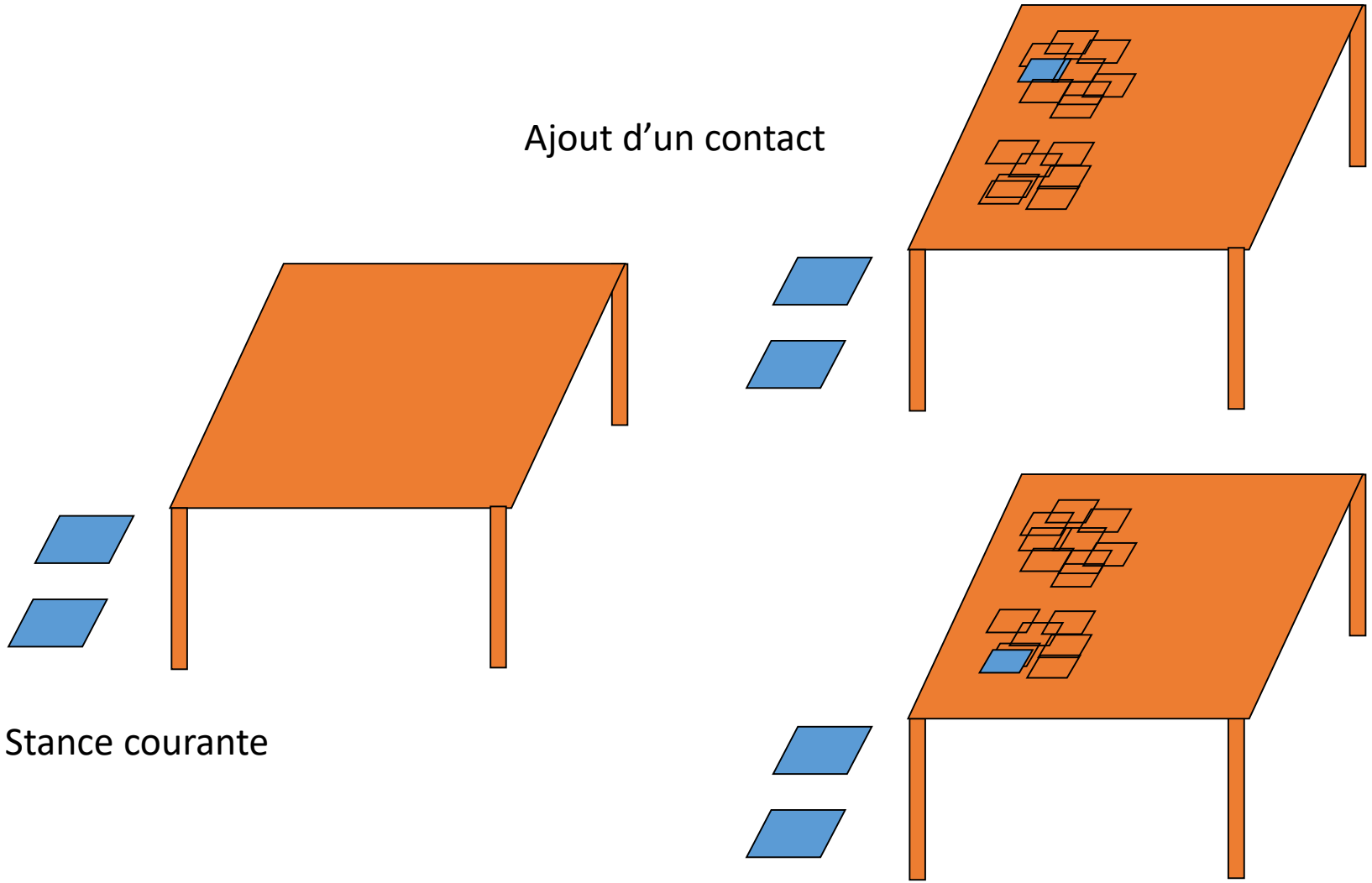
Stance courante

Ajout d’un contact

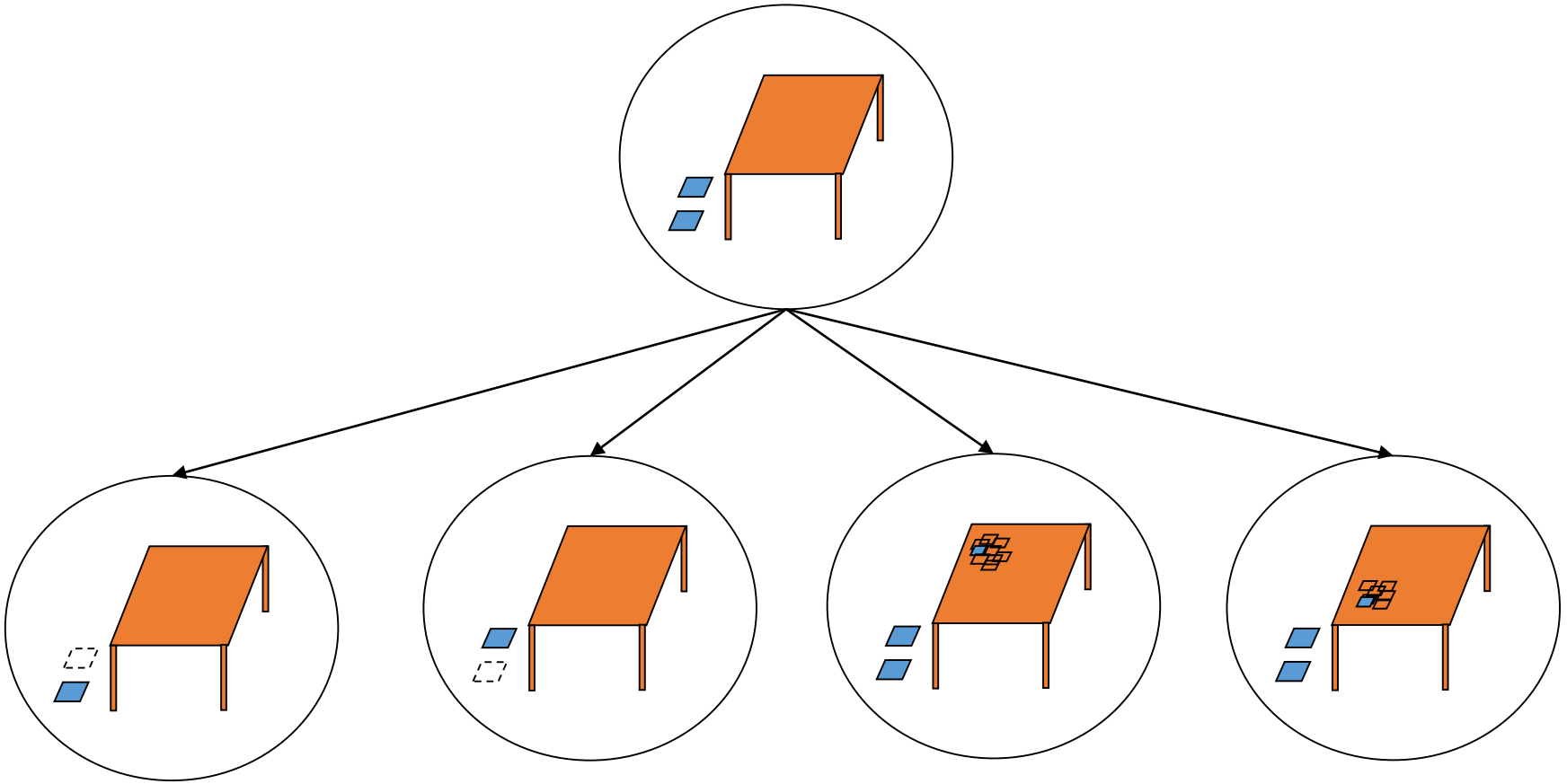


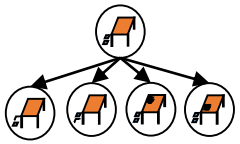
Current stance

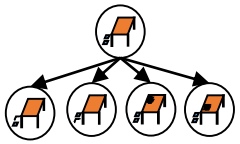
Ajout d’un contact

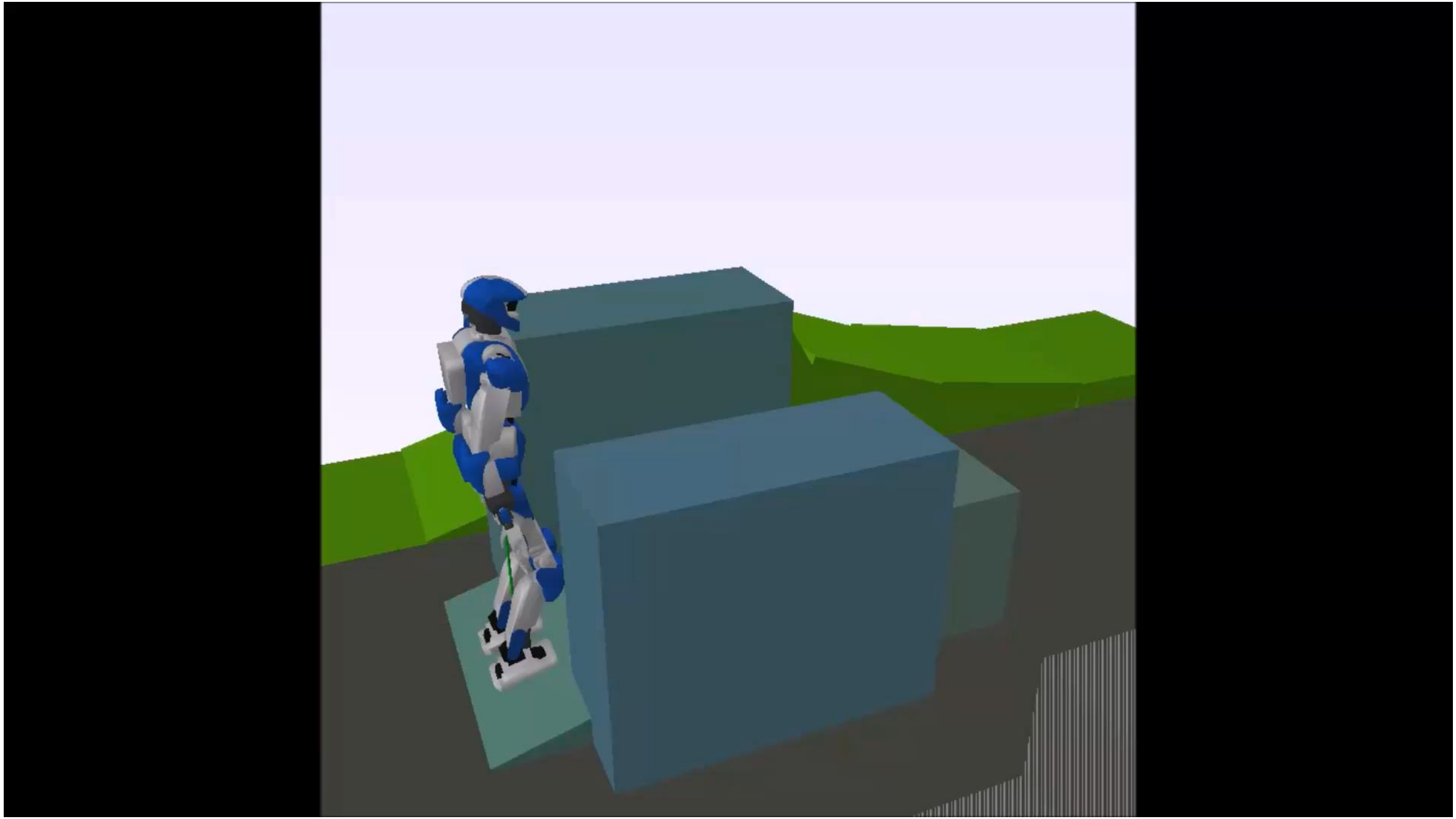


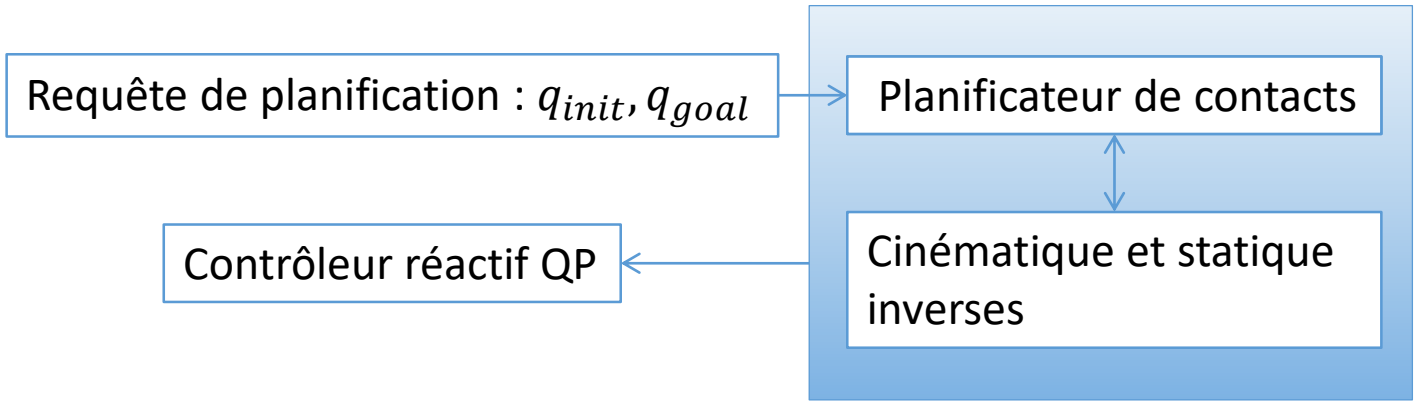
Stance courante










Requête de planification : q_{init}, q_{goal}
Planificateur de contacts
Cinématique et statique inverses
Contrôleur réactif QP

Contrôleur réactif QP

Contrôleur par programmation quadratique

Observation : en un état donné $q, \dot{q}$, l'equation de la dynamique est linéaire en $\ddot{q}, u, f$:

$$M(q)\ddot{q} + N(q, \dot{q}) = Su + J^T f$$

Contrôleur par programmation quadratique

$$\min_{\ddot{q},f,u} \sum_k w_k \left\| \ddot{\tau}_k - \ddot{\tau}_k{}^d \right\|^2$$

$$\begin{cases} M(q)\ddot{q}+N(q,\dot{q})=Su+J^T f \\ J\ddot{q}+\dot{J}\dot{q}=0 \\ Kf \geq 0 \\ |u| \leq u_{max} \end{cases}$$

Contrôleur par programmation quadratique

$$\min_{\ddot{q},f,u} \sum_k w_k \left\| \ddot{\tau}_k - \ddot{\tau}_k{}^d \right\|^2$$

$$\begin{cases} M(q)\ddot{q}+N(q,\dot{q})=Su+J^T f \\ J\ddot{q}+\dot{J}\dot{q}=0 \\ Kf \geq 0 \\ |u| \leq u_{max} \end{cases}$$

$\rightarrow \ddot{q}$

$\rightarrow u$

$\rightarrow f$

Contrôleur par programmation quadratique

$$\min_{\ddot{q},f,u} \sum_k w_k \left\| \ddot{\tau}_k - \ddot{\tau}_k{}^d \right\|^2$$

$$\begin{cases} M(q)\ddot{q}+N(q,\dot{q})=Su+J^T f \\ J\ddot{q}+\dot{J}\dot{q}=0 \\ Kf \geq 0 \\ |u| \leq u_{max} \end{cases}$$

$\rightarrow \ddot{q} \xrightarrow{\int} \dot{q} \xrightarrow{\int} q$

- contrôle en position
- utilisation en tant que simulateur

Contrôleur par programmation quadratique

$$\min_{\ddot{q},f,u} \sum_k w_k \left\| \ddot{\tau}_k - \ddot{\tau}_k{}^d \right\|^2$$
$$\begin{cases} M(q)\ddot{q}+N(q,\dot{q})=Su+J^T f \\ J\ddot{q}+\dot{J}\dot{q}=0 \\ Kf \geq 0 \\ |u| \leq u_{max} \end{cases}$$

$\rightarrow u$

Contrôle en couple

- vers un robot réel
- vers un simulateur graphique physique

(e.g. ODE, Bullet, etc) pour animation

Contrôleur par programmation quadratique

$$\min_{\ddot{q},f,u} \sum_k w_k \left\| \ddot{\tau}_k - \ddot{\tau}_k{}^d \right\|^2$$

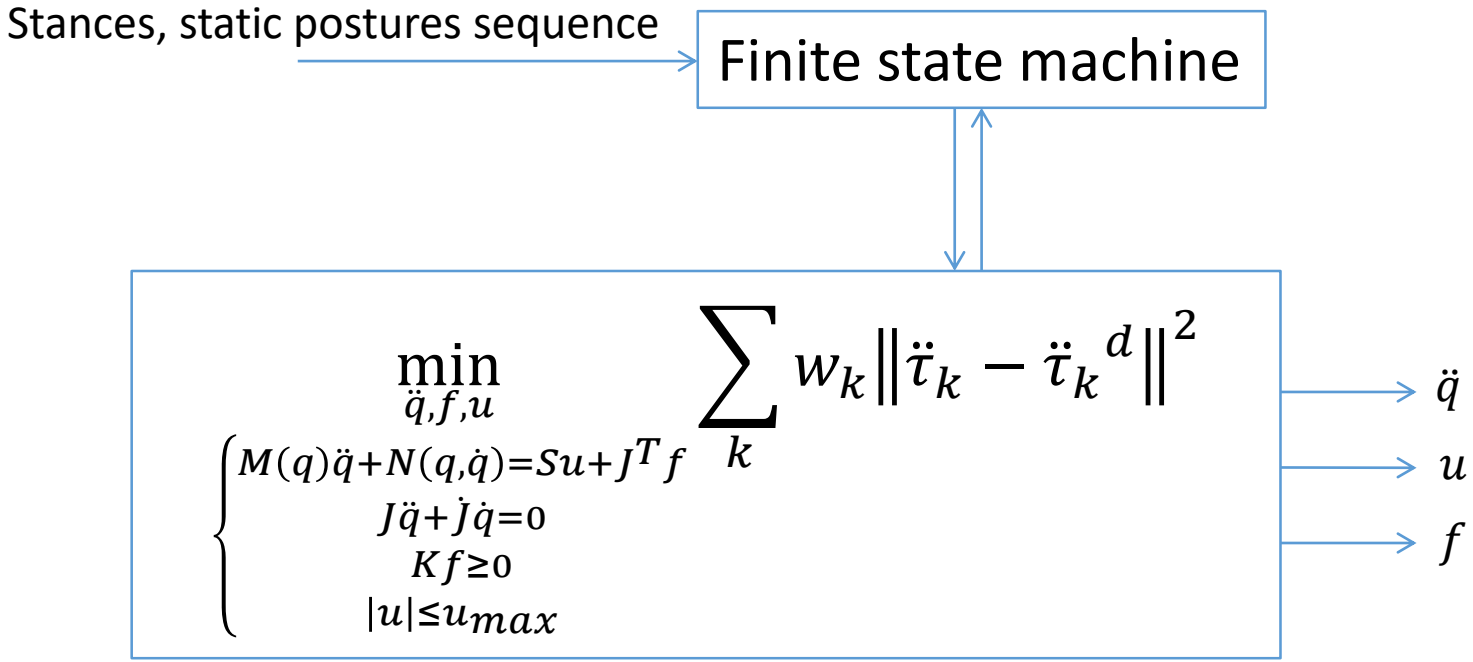
$$\begin{cases} M(q)\ddot{q}+N(q,\dot{q})=Su+J^T f \\ J\ddot{q}+\dot{J}\dot{q}=0 \\ Kf \geq 0 \\ |u| \leq u_{max} \end{cases}$$

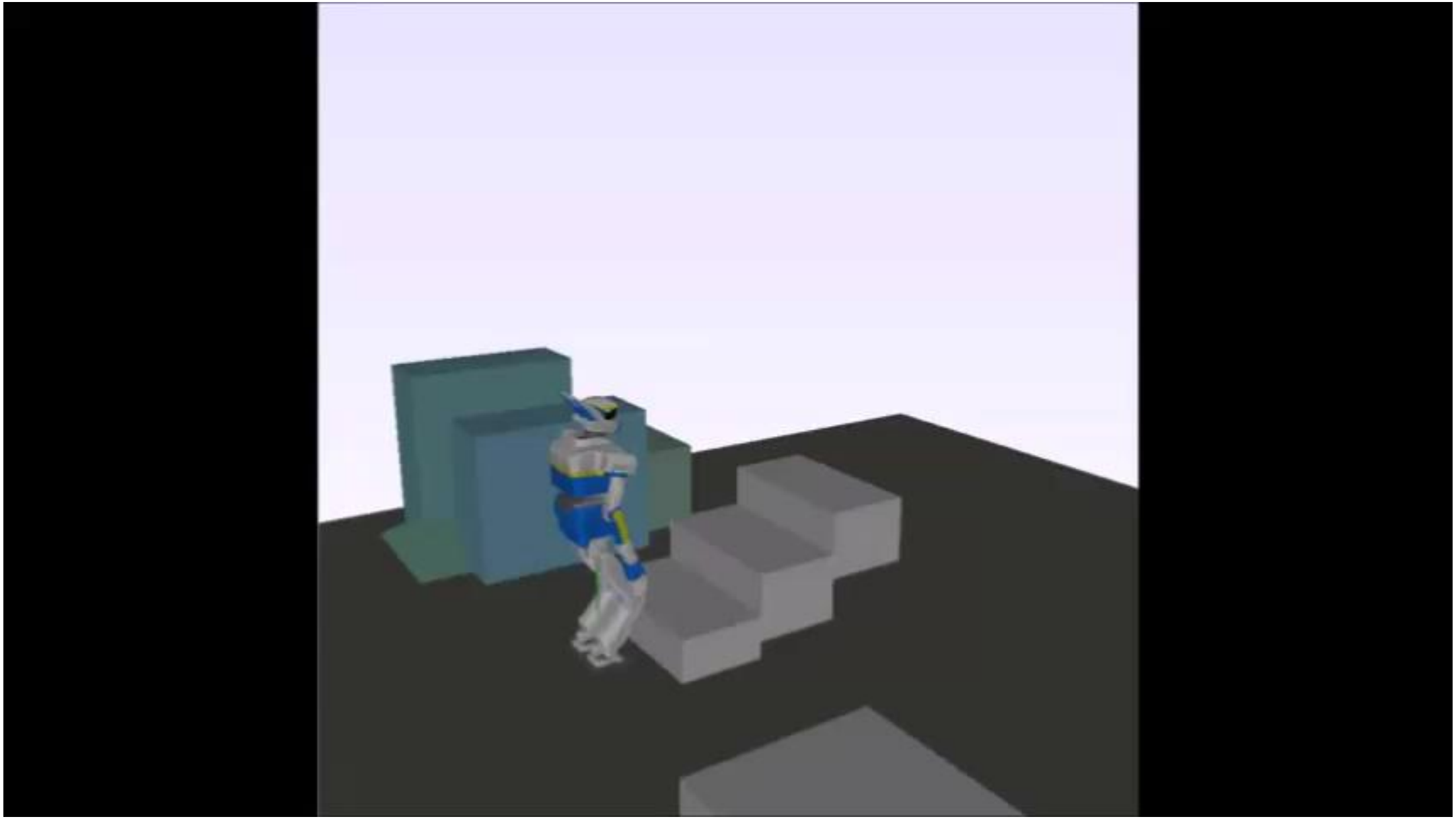
$\rightarrow \ddot{q}$

$\rightarrow u$

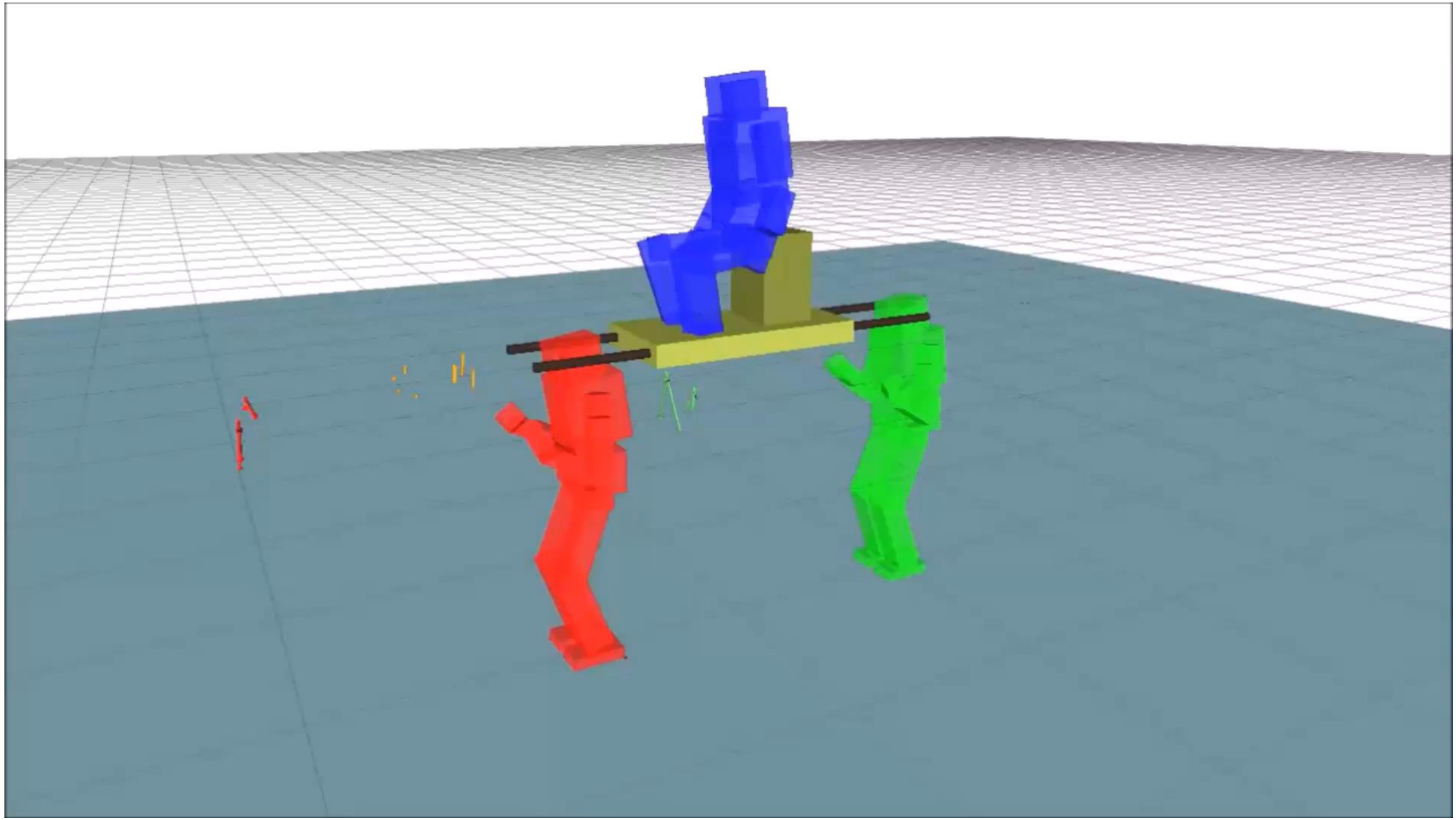
$\rightarrow f$

Contrôleur par programmation quadratique






AZA




FAIRPLEX

Correction, complétude et stabilité des contrôleurs humanoïdes

III. Automatique/Théorie du Contrôle

- Le contrôle multi-tâches d'un robot humanoïde peut être formulé comme

$$\min_{x \in \mathcal{X}} f(x) = \left(\left\| \ddot{\tau}_1 - \ddot{\tau}_1^d \right\|^2, \ldots, \left\| \ddot{\tau}_p - \ddot{\tau}_p^d \right\|^2 \right)$$

- Une tâche est n'importe quelle « feature » (gradeur d'intérêt) que l'on souhaite contrôler sur le robot (position de la main, position du centre de masse, du centre de pression, voire posture complète)

$$\tau_k : \mathbb{R}^n \rightarrow \mathbb{R}^{n_k}$$

- Toutes les tâches ne peuvent pas être résolues simultanément (conflits)
- Nécessité d'un compromis entre les tâches
- Deux « écoles » dans la communauté : l'approche par pondération et l'approche par priorisation stricte

- La formulation précédente est une instance particulière d'un problème d'optimisation multi-objectif

$$\min_{x \in \mathcal{X}} f(x) = \Big(f_1(x), \dots, f_p(x) \Big)$$

- Il n'y a pas de solution parfaite au problème, le point dit idéal n'est en général pas une solution

$$y^I = \Big(\min_{x \in \mathcal{X}} f_1(x), \dots, \min_{x \in \mathcal{X}} f_p(x) \Big) \notin \mathcal{Y} = f(\mathcal{X})$$

- Il existe tout un sous-ensemble de $\mathcal{X}$, ou de manière équivalente de $\mathcal{Y} = f(\mathcal{X})$, de solutions optimales, dénoté $\mathcal{Y}_N$

- Ces solution sont appelées : Pareto-optimales, solutions efficientes (x^*), points non-dominés ($y^* = f(x^*)$)

La pondération des tâches et leur priorisation sont deux méthodes parmi d'autres de résolution du problème d'optimisation multi-objectif

- Pondération (scalarisation linéaire) :

$$y^{ws}(w) = \min_{x \in X} \sum_{k=1}^{p} w_k f_k(x)$$

- Priorisation (optimum au sens de l'odre lexicographique total)

$$y^{lex} = \operatorname*{lexmin}_{x \in X} \ (f_1(x), \dots, f_p(x))$$

- Théorème :

$$y^{lex} \in \mathcal{Y}_N$$

$$\forall w > 0 \ : \ y^{ws}(w) \in \mathcal{Y}_N$$

- Question :

$$\exists w > 0 \ : \ y^{lex} = y^{ws}(w) \ ?$$

- Plus généralement :

$$\forall y \in \mathcal{Y}_N \quad \exists w > 0 \quad : \quad y = y^{ws}(w) \, ?$$

- Une notation utile : l'ensemble des solutions par pondération avec poids strictement positifs

$$\mathcal{S}(\mathcal{Y}) = \left\{ y^* \in \mathcal{Y} \;\middle|\; \sum_{k=1}^{p} w_k y_k^* = \min_{y \in \mathcal{Y}} \sum_{k=1}^{p} w_k y_k \,, w > 0 \right\}$$

- Théorème

$$\mathcal{S}(Y) \subset Y_N$$

- Réciproque ? fausse en général

- Les deux questions précédentes peuvent être reformulées en

$$y^{lex} \in \mathcal{S}(Y) \, ?$$

$$Y_N = \mathcal{S}(Y) \, ?$$

- Théorème (Hartley, 1978) :

Si $\mathcal{Y}$ non vide, $\mathbb{R}^p_{\leqq}$-convexe et $\mathbb{R}^p_{\leqq}$-fermé, alors $\mathcal{Y}_N \subset cl(\mathcal{S}(\mathcal{Y}))$

- cl est la fermeture (adhérence)

- On obtient donc, sous les bonnes conditions

$$\mathcal{S}(\mathcal{Y}) \subset \mathcal{Y}_N \subset cl(\mathcal{S}(\mathcal{Y}))$$

et quelques corollaires, dont le fait que la solution lexicographique peut être approchée à un epsilon arbitraire près par une scalarisation par pondération.

Dans l'espace des solutions, « la scalarisation par pondération est dense »

- Proposition (stabilité des tâches) :

Si $\mu(A_k) < 0$ alors, pour tout $\epsilon > 0$, l'inéquation différentielle :

$$\left\| \ddot{\tau}_k - \ddot{\tau}_k^d \right\|^2 < \epsilon$$

résulte en un $\eta_k(t)$ **globalement ultimement uniformément borné** (UUB). De plus, pour tout $t \to \epsilon(t) > 0$ tel que $\epsilon(t) = \mathcal{O}(e^{2\mu(A_k)t})$, l'inéquation différentielle :

$$\left\| \ddot{\tau}_k - \ddot{\tau}_k^d \right\|^2 < \epsilon(t)$$

implique, pour toute condition intitiale $\eta_k(0)$:

$$\eta_k(t) \xrightarrow[t \to +\infty]{} 0.$$

- On souhaite à présent étudier les propriétés de stabilité (au sens de Lyapunov) du système n-dimenstional d'équations différentielles obtenu par scalarisation par pondération :

$$\ddot{q} = \underset{\ddot{q}}{\operatorname{argmin}} \sum_{k=1}^{p} w_k \left\| \ddot{\tau}_k - \ddot{\tau}_k^d \right\|^2$$

- Proposition :

Si $B > 0$ le système

$$\dot{\xi} = \underset{\ddot{q}}{\operatorname{argmin}} \sum_{k=1}^{p} w_k \|\dot{\eta}_k - A_k \eta_k\|^2$$

admet un point d'équilibre si et seulement si il existe ξ^0 tel que

$$\sum_{k=1}^{p} w_k \mathcal{J}_k(\xi^0)^T A_k \gamma_k(\xi^0) = 0$$

dans ce cas, cet équilibre est exponentiellement stable (au sens de Lyapunov) si et seulement si la matrice

$$B^{-1} \sum_{k=1}^{p} w_k \left((\gamma_k^T A_k^T \otimes I_{2n_k}) K_{2n_k 2n} \Gamma_k + \mathcal{J}_k^T A_k \mathcal{J}_k \right)$$

évaluée en ξ^0 est Hurwitz

- Take away : Tester la stabilité du contrôleur revient à calculer les valeurs propres d'une matrice qu'on peut calculer analytiquement

$$B^{-1}\sum_{k=1}^{p} w_k \left(\left(\gamma_k^T A_k^T \otimes I_{2n_k} \right) K_{2n_k 2n} \Gamma_k + \mathcal{J}_k^T A_k \mathcal{J}_k \right)$$

- Dans le cas d'un robot humanoïde, on démontre qu'on peut construire une représentation analytique de l'ensemble $\mathcal{X}$ qui apparaît dans la formulation

$$\min_{x \in \mathcal{X}} f(x) = \left(\left\| \ddot{\tau}_1 - \ddot{\tau}_1^d \right\|^2, \ldots, \left\| \ddot{\tau}_p - \ddot{\tau}_p^d \right\|^2, \|u\|^2, \|f\|^2 \right)$$

- Lemme :

Pour un humanoïde, les conditions du théorème de Hartley sont vérifiées

- On peut donc considérer la scalarisation par pondération, qui est presque complète dans l'espace des solutions Pareto-optimales, et qui débouche sur un programme quadratique QP

$$\min_{x} x^T Q x + l^T x$$
$$\text{subject to } H_e x = b_e \, , H_i x \leqq b_i$$

- Proposition :

Pour un humanoïde, le programme quadratique précédent atteint son minimum en un unique point, *i.e.* la solution existe et est unique

- Proposition :

Soit $p = (\delta Q, \delta l, \delta H_e, \delta H_i, \delta b_e, \delta b_i)$ une perturbation du programme quadratique précédent. Si H_e et $H_e + \delta H_e$ sont de rangs pleins et si le système de contraintes linéaires du QP est régulier à l'état initial ξ^0, alors il existe $\epsilon > 0$ et $\mathrm{K} > 0$ tels que la solution $x^* = x^0 + \delta x$ du QP perturbé

$$\min_x x^T (Q + \delta Q)x + (l + \delta l)^T x$$
$$\text{subject to } (H_e + \delta H_e)x = b_e + \delta b_e, (H_i + \delta H_i)x \leqq b_i + \delta b_i$$

existe et est unique et vérifie, chaque fois que $\|p\| < \epsilon$

$$\left\|x^* - x^0\right\| < \mathrm{K} \|p\|_\infty$$

- Proposition :

Sous les hypothèses de la proposition précédente, l'application $p \to x^*$ est bien définie en un voisinage de 0 et est continue en 0.

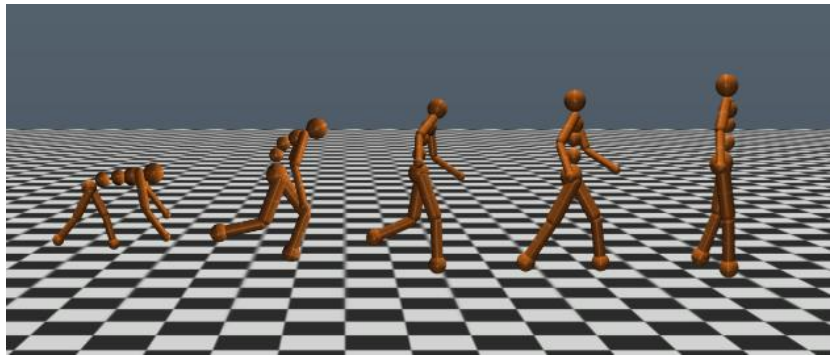
IV. Projets

Sujet non couverts dans cette présentation

- Contrôle de chute d’humanoïde
- Contrôle corps-complet par interface cerveau machine
- Perception
  - SLAM humanoïde (Simultaneous Localization and Mapping)
- Middleware et architectures de contrôle
- Hardware

Sujets « chauds »

- Apprentissage profond, algorithmes évolutionaires et intelligence artificielle générale de mouvement
  - Google Deepmind “Emergence of Locomotion Behaviours in Rich Environments”
  - Uber AI labs “Welcoming the Era of Deep Neuroevolution”



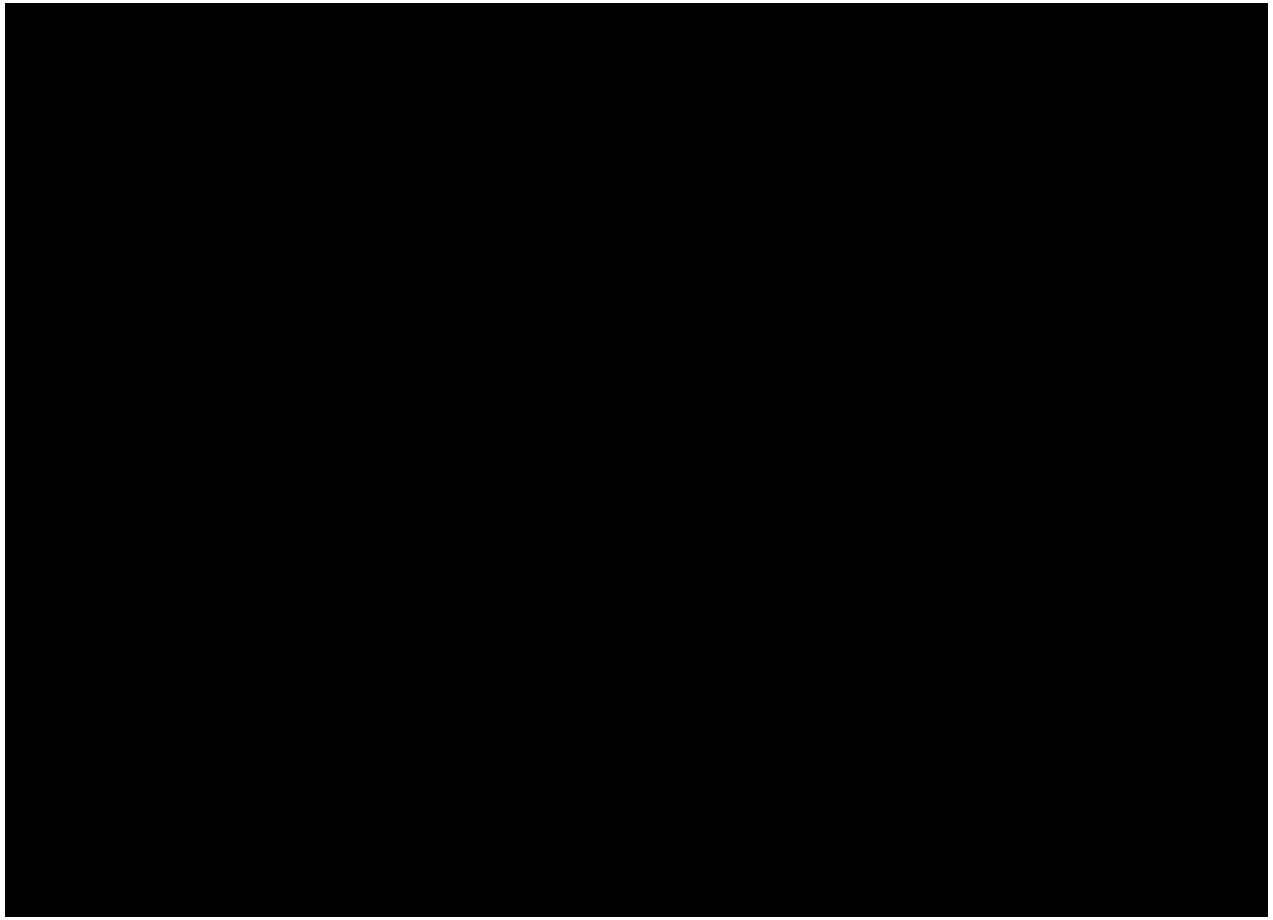
  - Travaux en cours (apprentissage et optimization CMA-ES) (Spitz, Bouyarmane al, Humanoids 2017)

- Nouveaux développements en hardware, ouverture de nouvelles possibilités

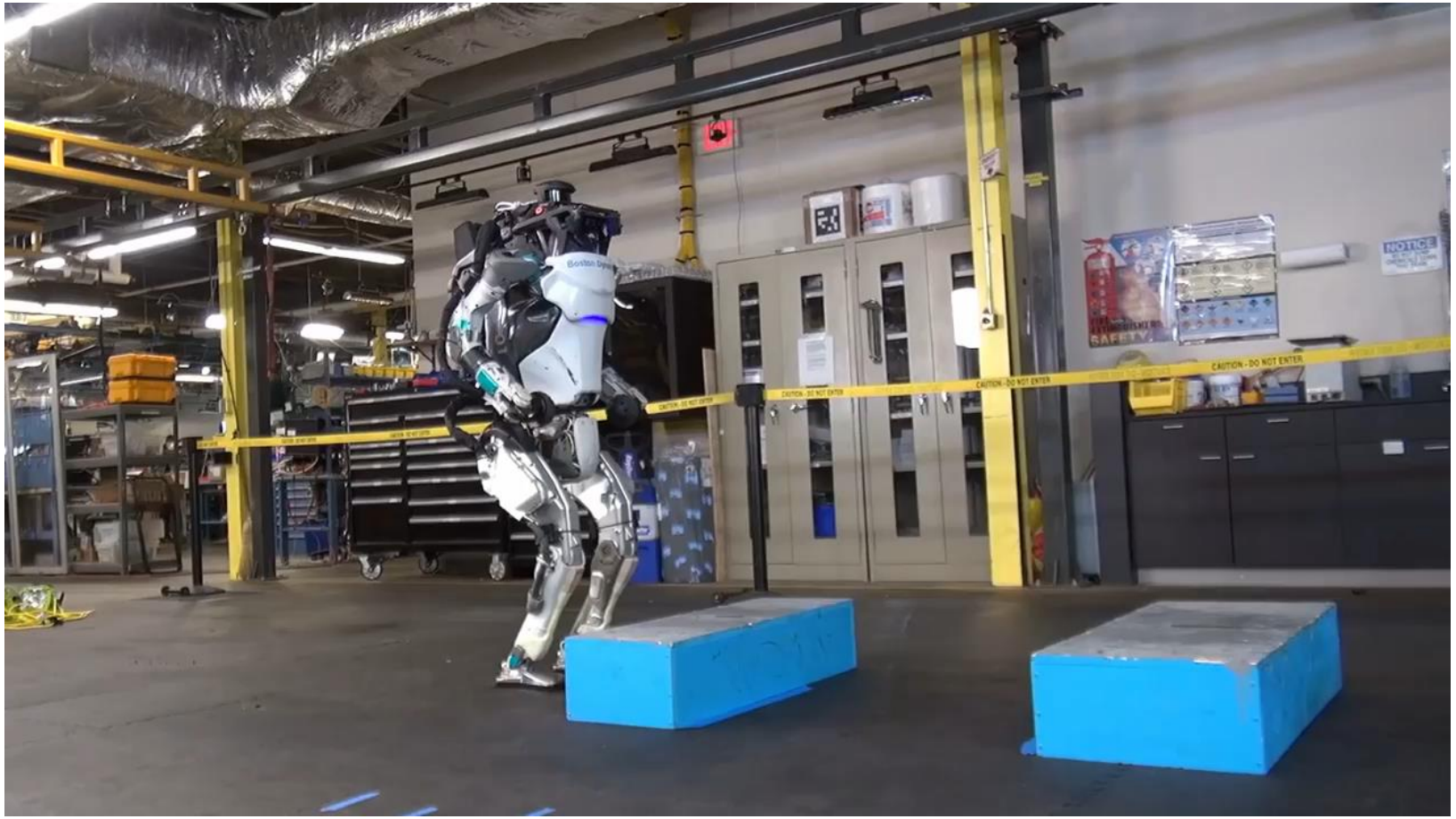
Adaptive Whole-Body Manipulation in Human-to-Humanoid Multi-Contact Motion Retargeting

Kazuya Otani Karim Bouyarmane

Deep Reinforcement Learning



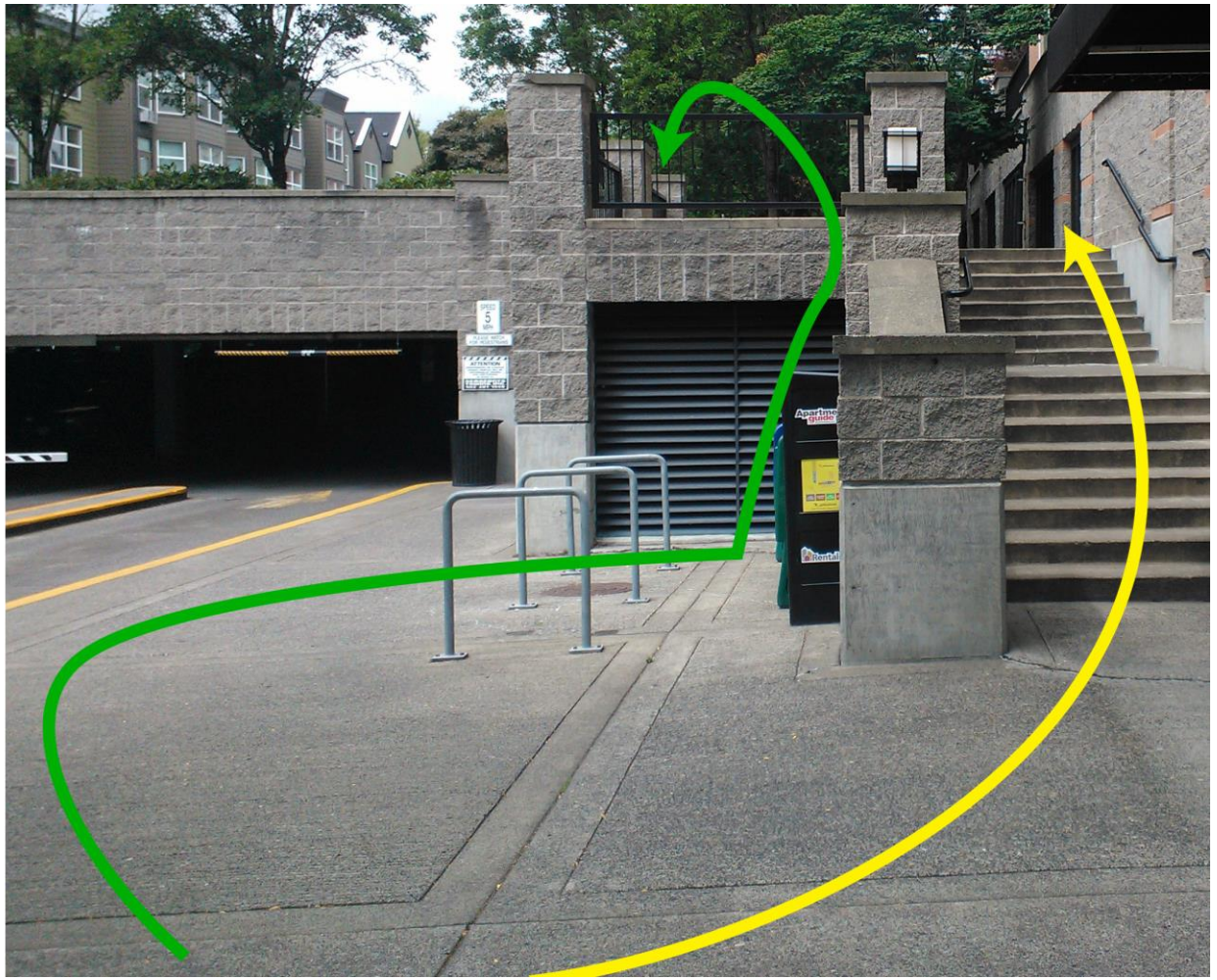
Super hardware (Boston Dynamics)



Un projet à moyen/long terme : X-AgH-Mint

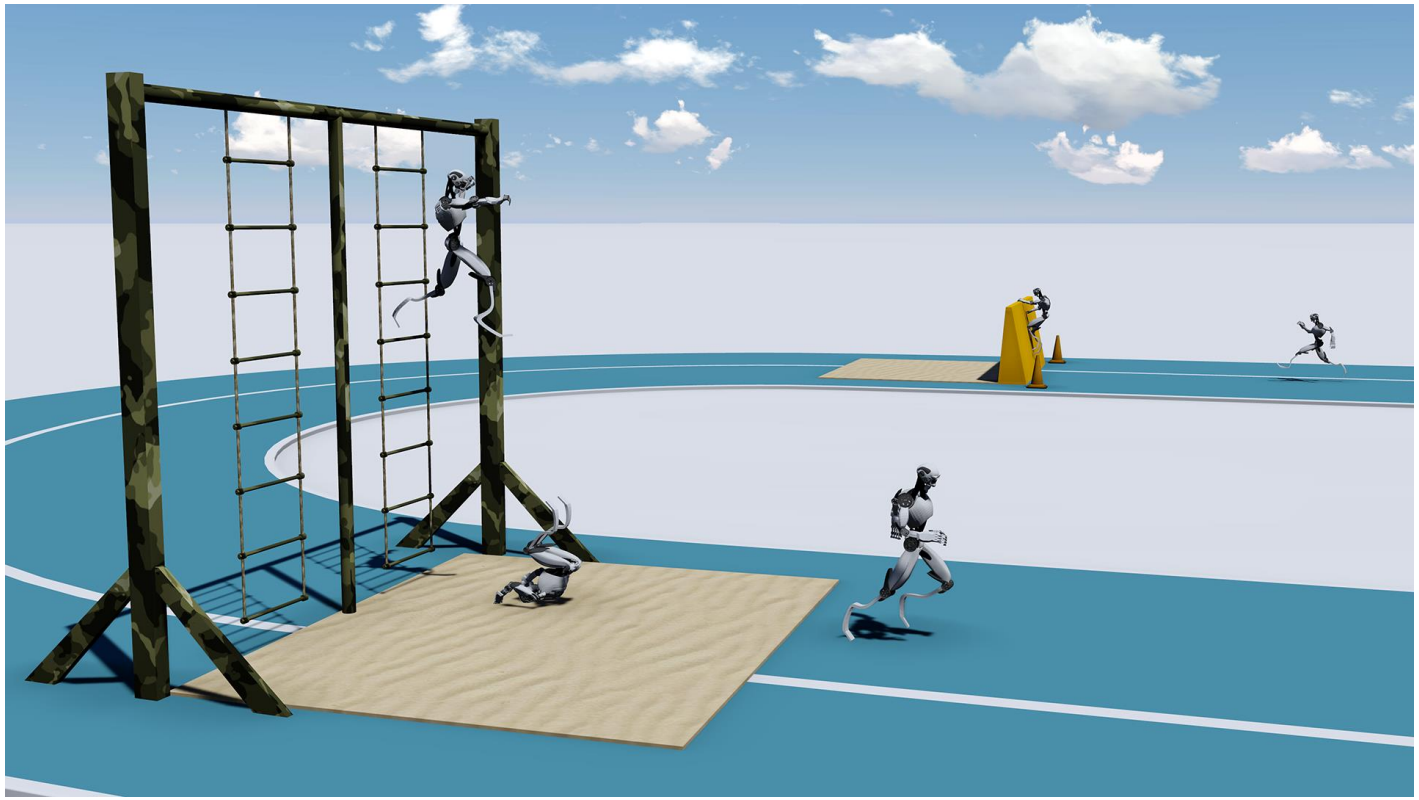
EXtremely Agile Humanoïd Motion Intelligence

X-AgH-Mint EXtremely Agile Humanoïd Motion Intelligence



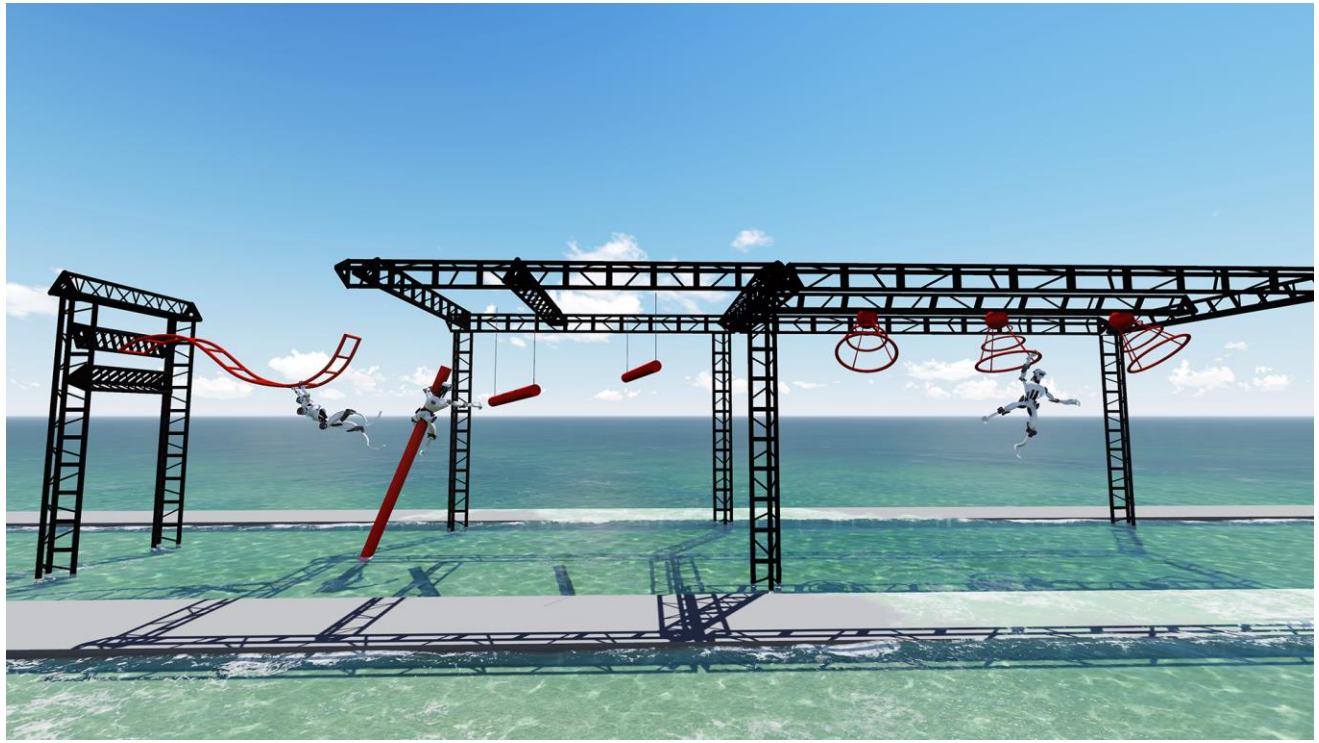
X-AgH-Mint EXtremely Agile Humanoïd Motion Intelligence

X-AgH-Mint EXtremely Agile Humanoïd Motion Intelligence



X-AgH-Mint EXtremely Agile Humanoïd Motion Intelligence

X-AgH-Mint EXtremely Agile Humanoïd Motion Intelligence



Merci pour votre attention !

Papiers, preprints, videos : https://members.loria.fr/kbouyarmane

karim.bouyarmane [at] polytechnique.edu