

The humanoid robot motion planning problem

Karim Bouyarmane

What I will talk about

- The path planning problem
 - Some algorithms
- The robot motion planning problem as a path planning problem
 - Some more algorithms
- The humanoid robot motion planning problem as a multi-modal (path/motion planning) problem
 - An algorithm
- The humanoid robot control problem
 - A universal controller

The path planning problem

Formulation of the problem

- How to go from point A to point B?

Formulation of the problem

- How to go from point A to point B?
 - Intuitively : Launch Google maps

Formulation of the problem

- How to go from point A to point B?
 - Intuitively : Launch Google maps
 - Algorithm
 - get a map (download it)
 - encode relevant information your map (paths) in a graph-like structure (roadmap)
 - run your favorite graph search algorithm (Dijkstra, best-first, A*, D*, etc)
 - done

Formulation of the problem

- How to go from point A to point B?
 - Intuitively : Launch Google maps
 - Algorithm
 - get a map (download it)
 - encode relevant information your map (paths) in a graph-like structure (roadmap)
 - run your favorite graph search algorithm (Dijkstra, best-first, A*, D*, etc)
 - done
- Let's call this the "Google maps meta-algorithm"

Formulation of the problem

- How to go from point A to point B?
- Can we generalize the problem in a sound mathematical formulation?

Formulation of the problem

- How to go from point A to point B?
 - What is “go”?
 - what are “points” A and B?
 - “what” needs to go (subject of the verb)?

Formulation of the problem

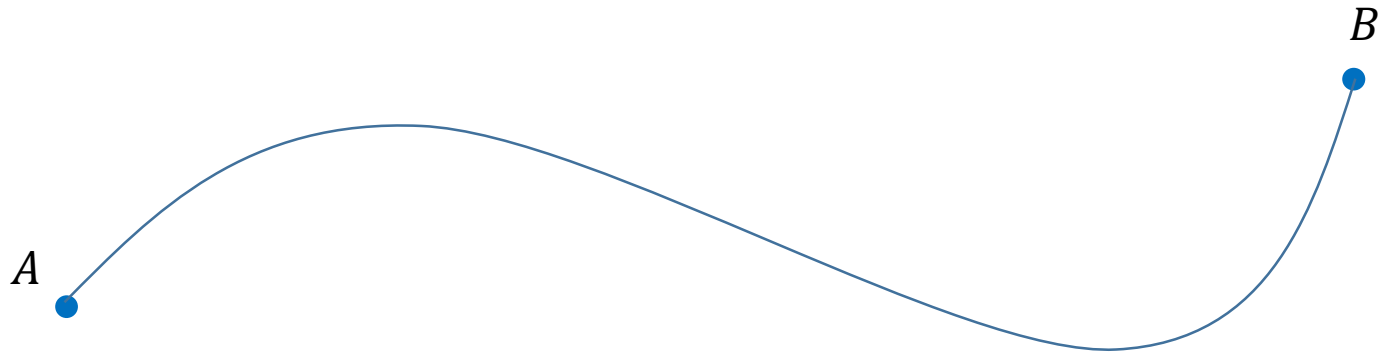
- How to go from point A to point B?
 - What is “go”? → (Google maps) a path along the map
 - what are “points” A and B? → (Google maps) 2D points in the map
 - “what” needs to go (subject of the verb)? → (Google maps) a 2D point constrained to move along the edges of the graph

Formulation of the problem

- Let's consider that the subject of the motion is a “point”. Points are in the ideal mathematical sense, elements of a set, not in the physical point mass sense : no volume, no mass, no physics.
- how to make a point “move” from point A to point B in some set endowed with a notion of “continuity” and a notion of “path” \Rightarrow necessity of operating in a topological *space*
- The notion of motion is therefore associated with the notion of continuity and of topology

Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.



Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.

Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- Question: is there such path? (existence)

Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- Question: is there such path? (existence)
- Answer: property of connectedness of the space

Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- Question: is there such path? (existence)
- Answer: property of connectedness of the space?
 - Definition : a space is connected if it cannot be written as a union of two disjoint open sets/two disjoint closed sets

Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- Question: is there such path? (existence)
- Answer: property of ~~connectedness~~ of the space

Formulation of the problem

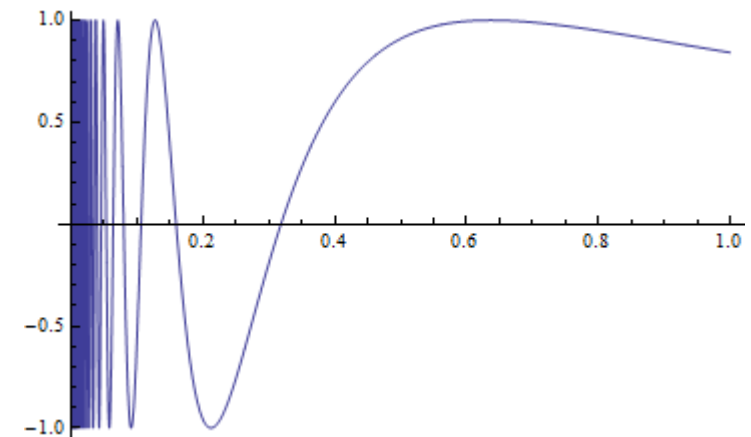
- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- Question: is there such path? (existence)
- Answer: property of *path*-connectedness of the space

Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- path-connected \Rightarrow connected, converse not true in general
 - counter example : the topologist's sine curve

$$\left\{ (x, y) \mid x = 0 \text{ or } y = \sin\left(\frac{1}{x}\right) \right\}$$

connected but not path connected

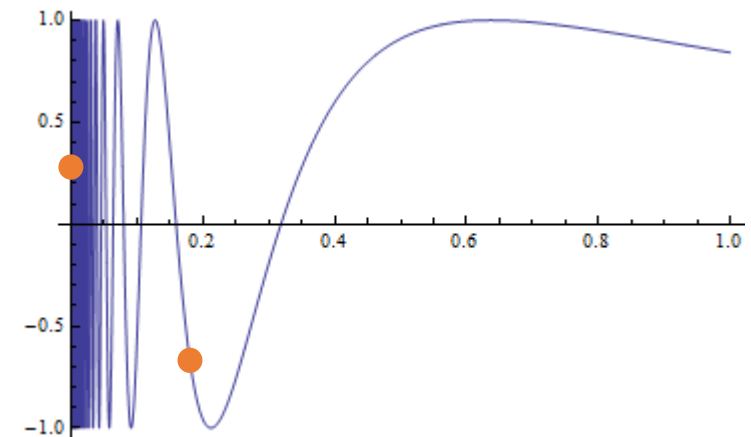


Formulation of the problem

- Let X be some arbitrary topological space and A and B two points in X . A path between A and B is some continuous function f from $[0,1]$ to X such that $f(0) = A$ and $f(1) = B$.
- path-connected \Rightarrow connected, converse not true in general
 - counter example : the topologist's sine curve

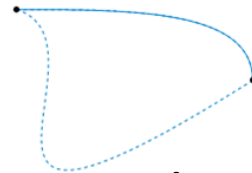
$$\left\{ (x, y) \mid x = 0 \text{ or } y = \sin\left(\frac{1}{x}\right) \right\}$$

connected but not path connected



Formulation of the problem

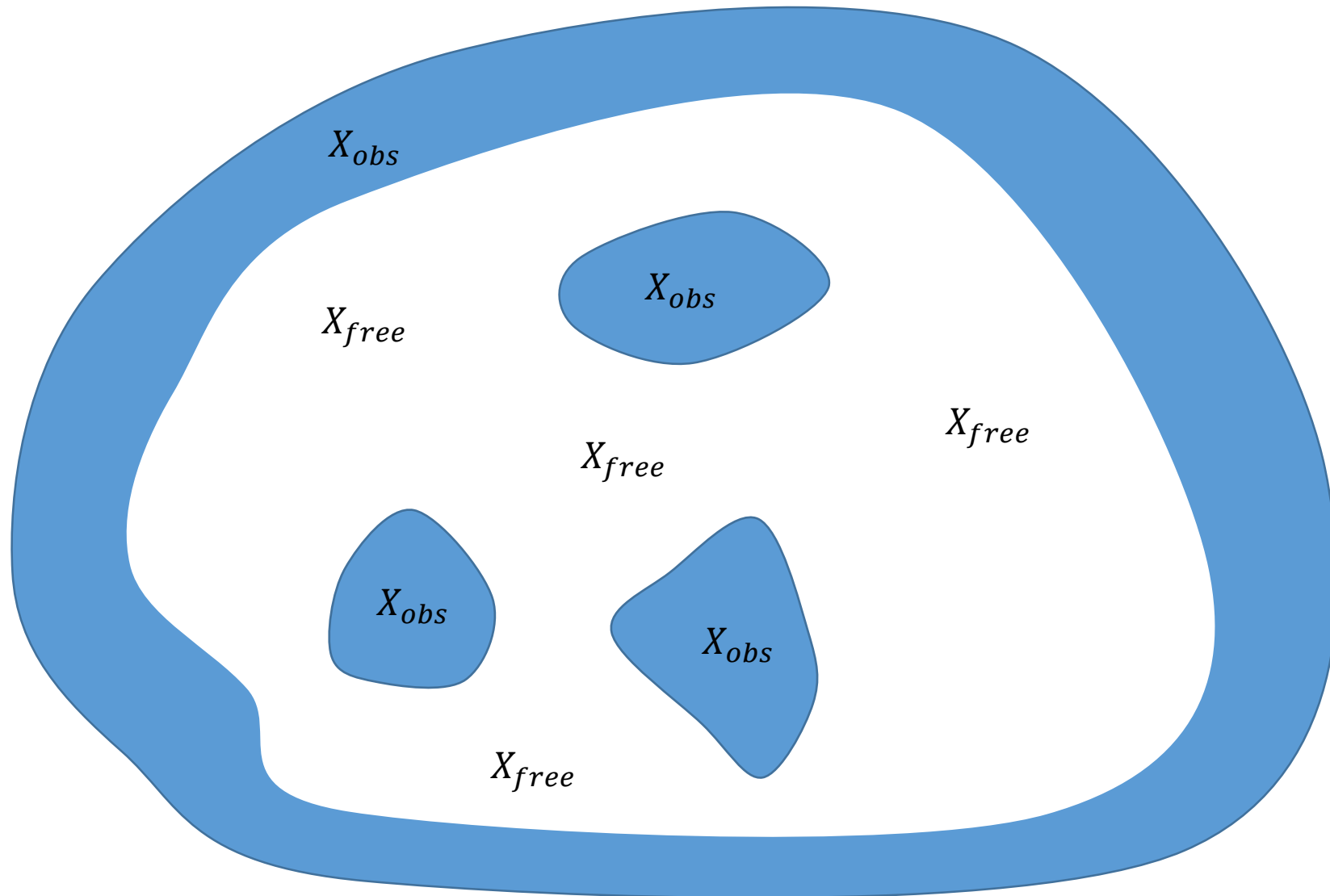
- For convenience, we will consider additional structure on our space X : X is a manifold of fixed dimension n (subset of \mathbb{R}^m locally homeomorphic in each of its points to \mathbb{R}^n)
- In this case: connected \Leftrightarrow path-connected
- Additional interesting properties worth studying in our context for the space X :
 - simply connected or multiply connected
 - Homotopy classes of paths (can a path be continuously deformed into another path) ?
 - Fundamental group (How many topologically different (non homotopic) ways to go from a point to another)



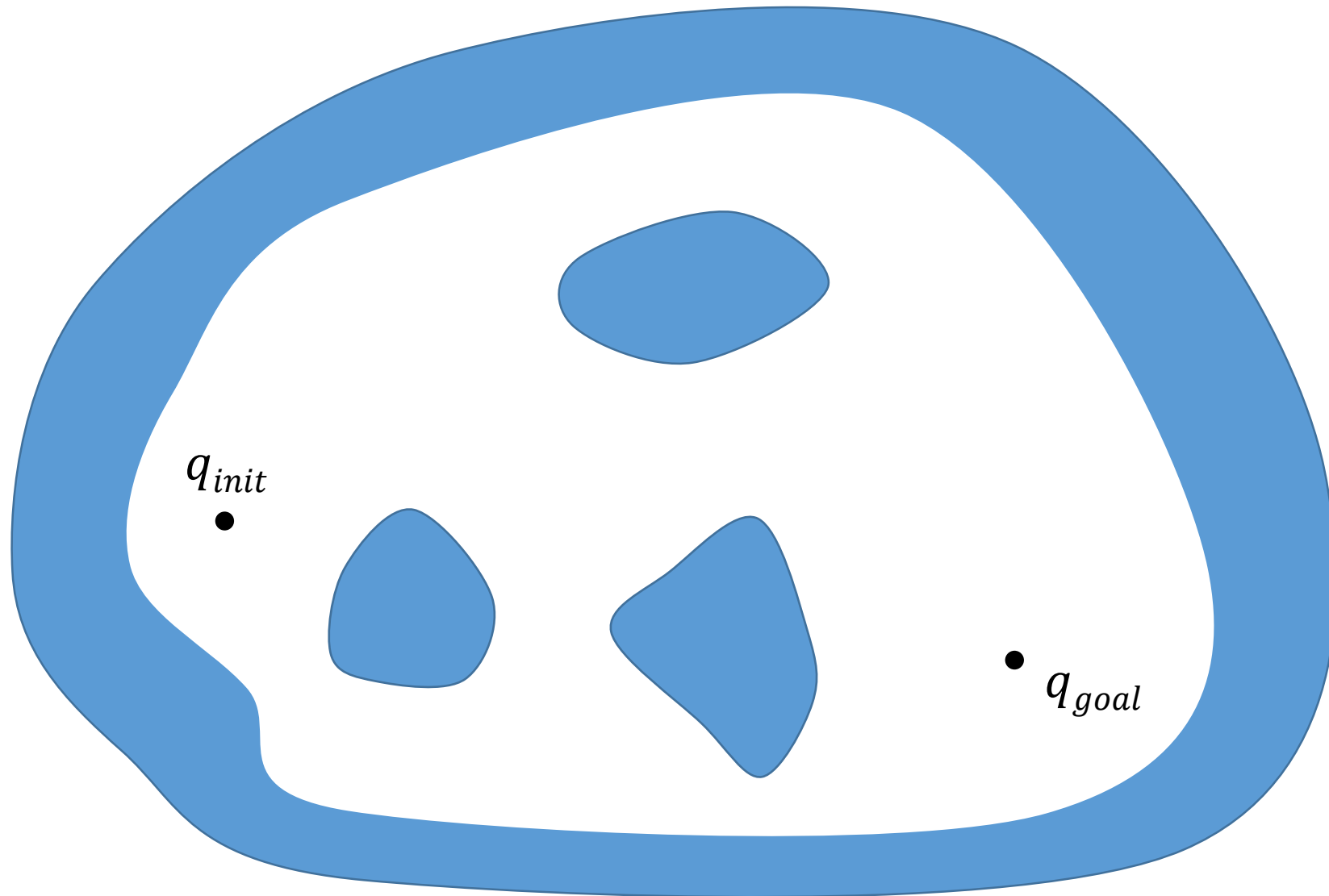
Obstacles and Free Space

- Obstacles are compact subsets of X : X_{obs}
- The space X will be partitioned in two sub sets: $X = X_{obs} \cup X_{free}$
- Path planning algorithms will operate in X_{free} , avoiding X_{obs}
- X_{free} is an open set \Rightarrow no “optimal” path in general, we are not allowed to “touch” the obstacles

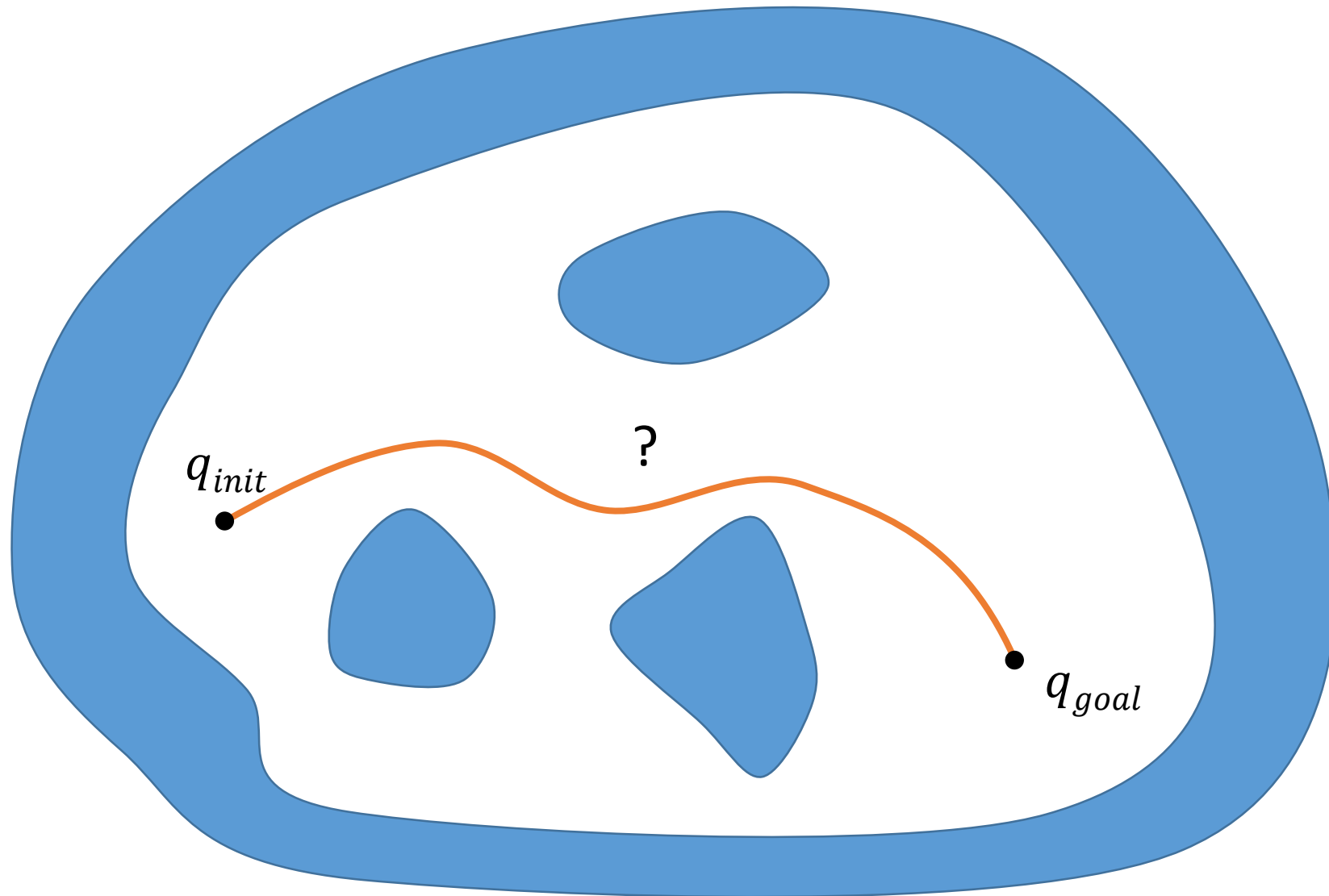
Formulation of the problem



Formulation of the problem



Formulation of the problem

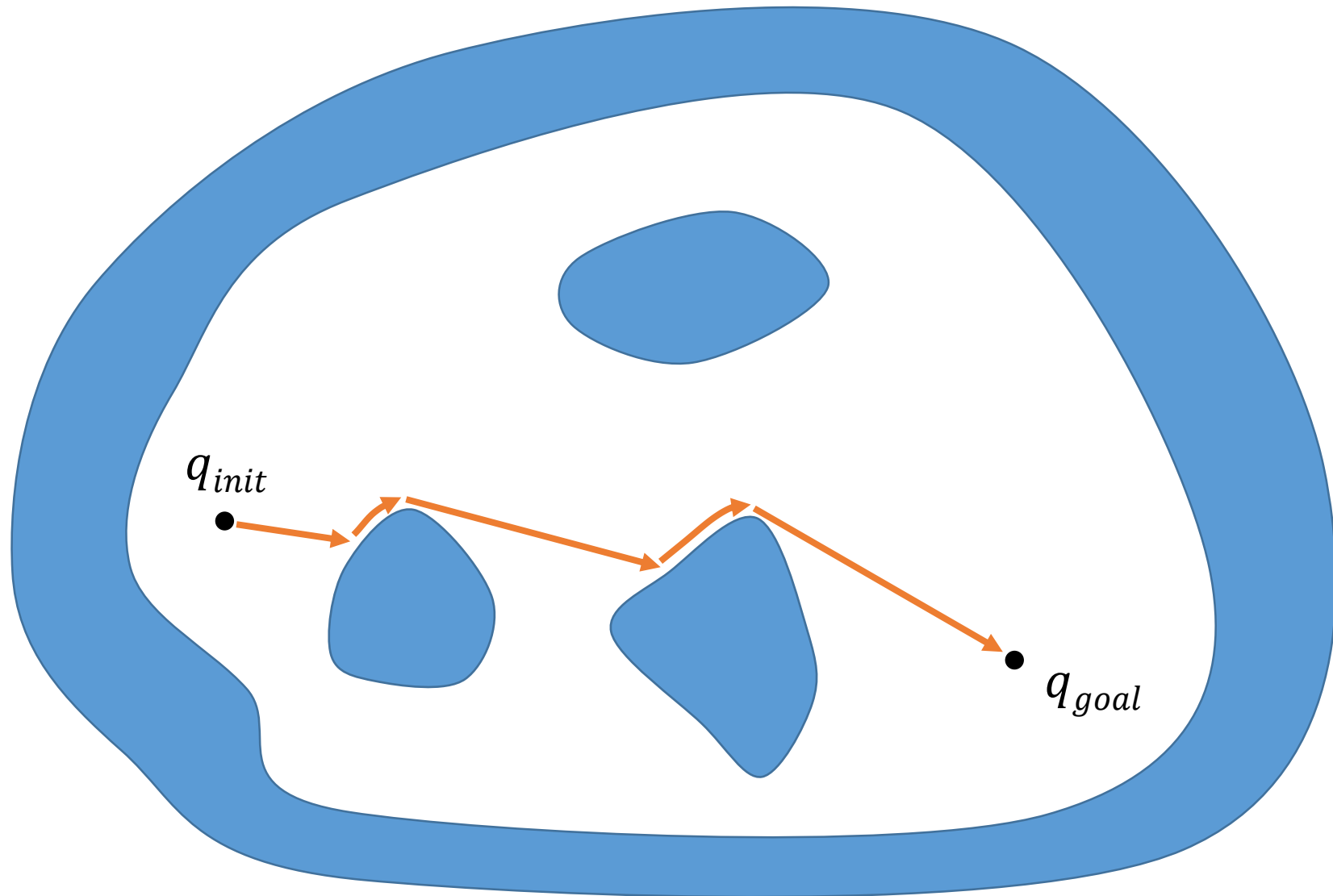


First algorithms

First algorithms

- Intuitive ideas:
 - The bug algorithm
 - Potential fields
 - Potential fields with occasional random walks

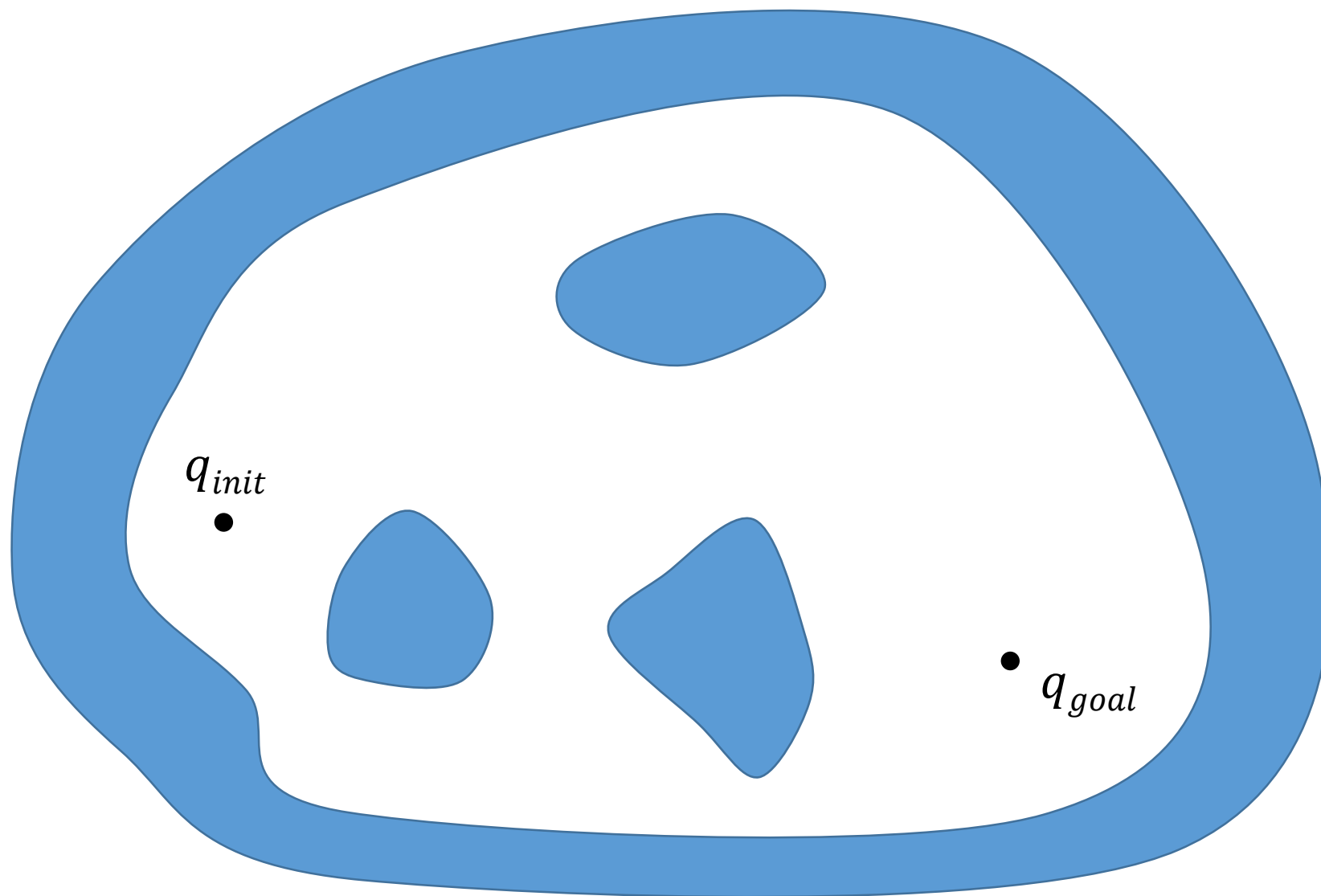
Bug algorithm



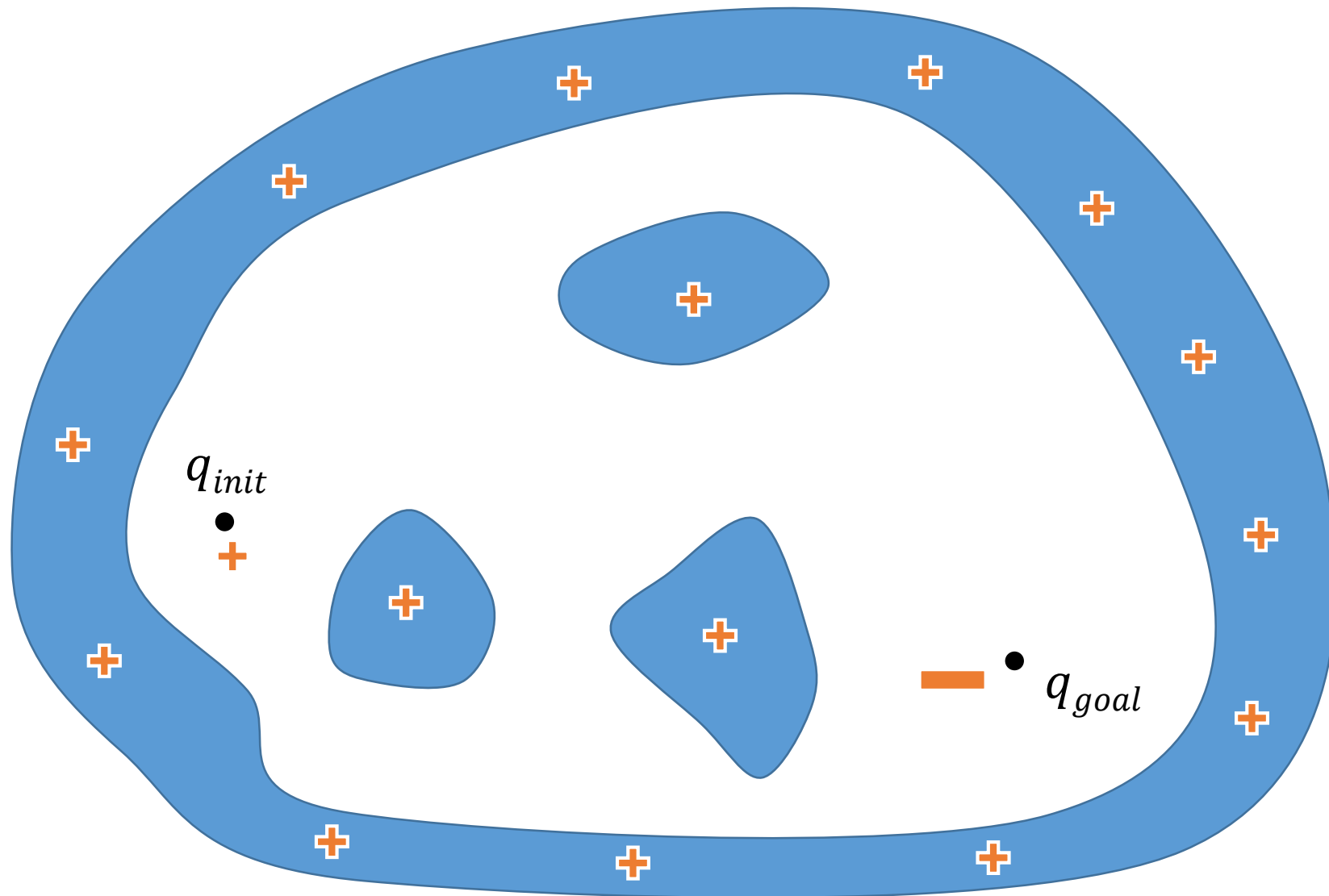
Bug algorithm

- Easy to find instances that “confuse the bug”
- We can add “memory” to the bug so that it circumvent all the obstacle before committing to leave it at best possible point
 - bug1 algorithm
- Add so-called m-line (straight line between initial and goal point through obstacles) as preferred direction to follow when leaving obstacle
 - bug2 algorithm
- “Cheating” → we allow ourselves to touch obstacles

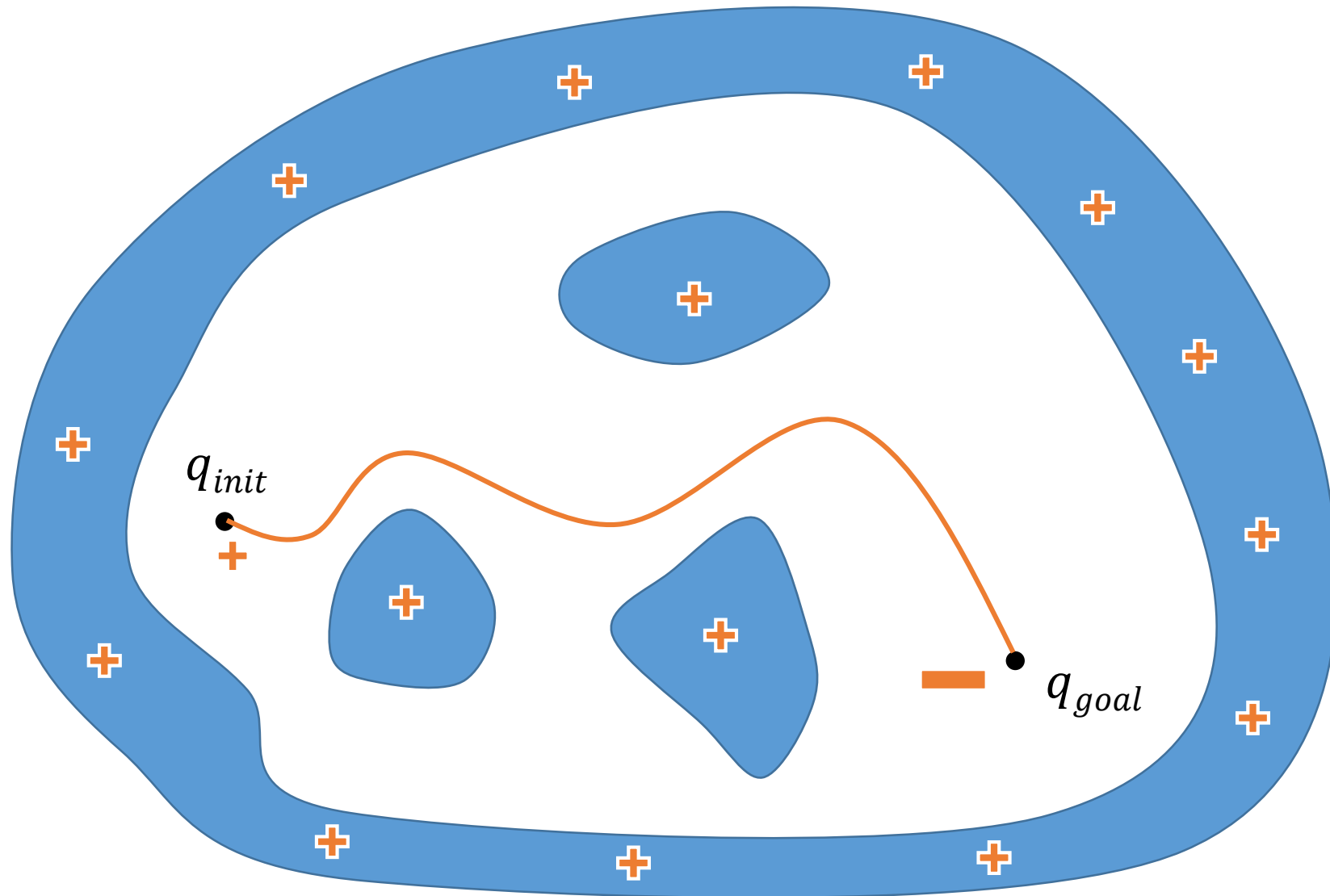
Potential field



Potential field



Potential field

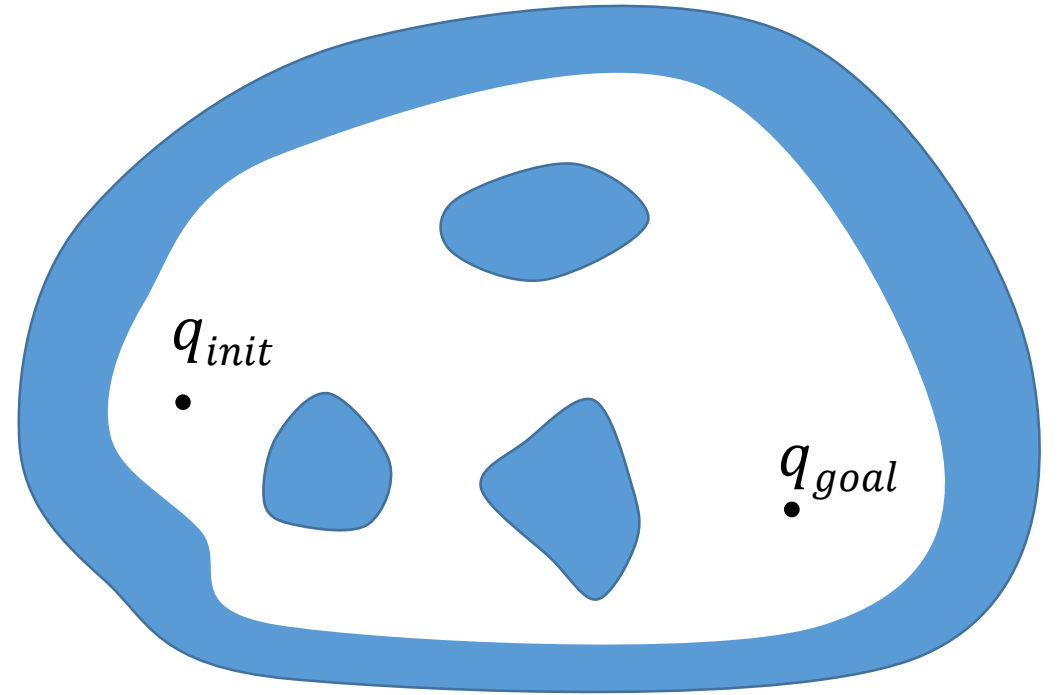


Potential field

- Algorithm:
 - Charge particle and obstacles with given sign charge
 - Charge goal with opposite sign charge
 - Simulate the system and let it flow
- Problem: difficult to calibrate correctly the charges, the forces, etc, to avoid local minima in combined field (parameter tuning)
- Improvement : couple it with occasional local random walks when trapped in local minima

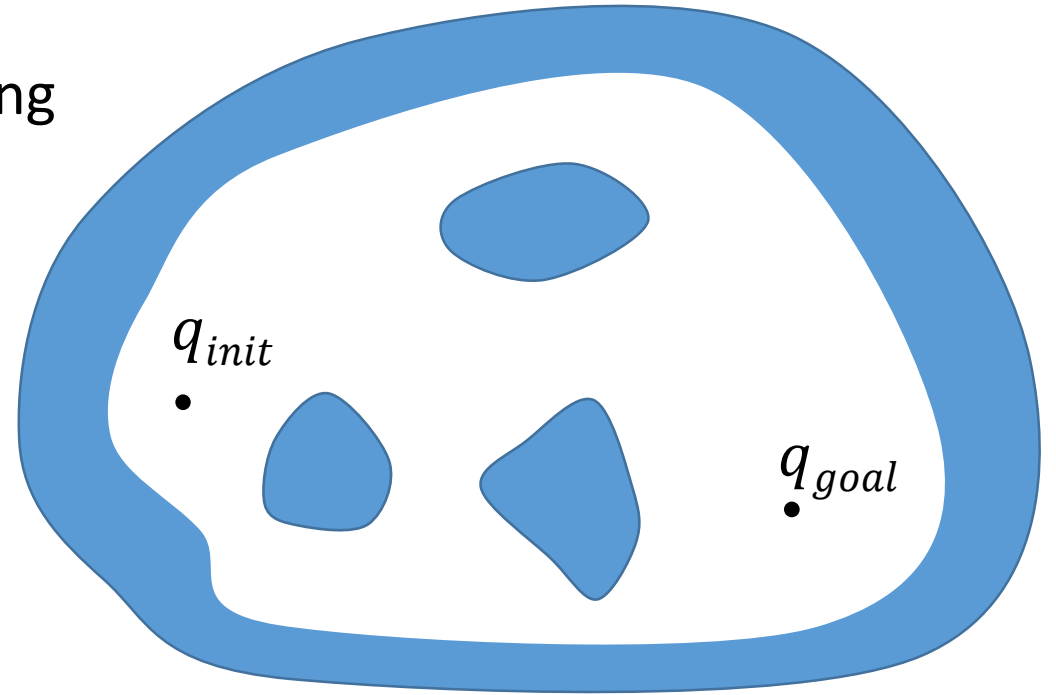
Google map meta algorithm

- Can the Google maps meta-algorithm work here?



Google map meta algorithm

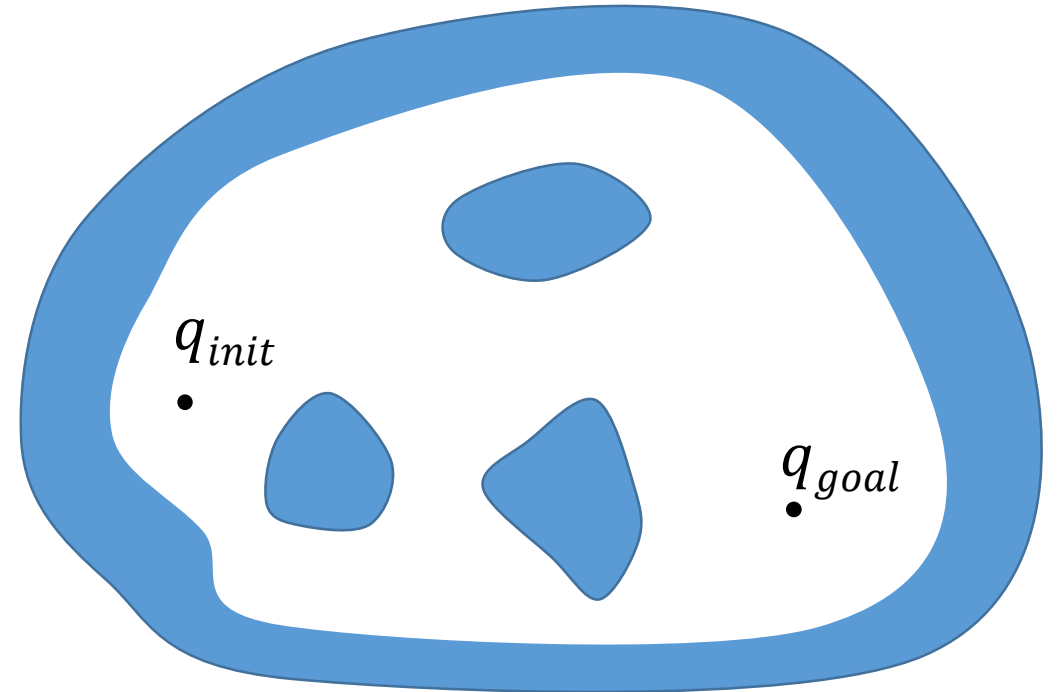
- Can the Google maps meta-algorithm work here?
 - Yes if we can build a “roadmap”
 - that is: a discrete structure (graph) encoding the connectedness of the free space



Google map meta algorithm

- Example algorithms for roadmap building:

- Extended voronoi diagrams
- Visibility roadmap
- Cell decomposition
 - Exact cell decompositions
 - vertical cell decomposition
 - cylindrical cell decomposition
 - Approximate cell decomposition
 - fixed resolution
 - adaptive resolution

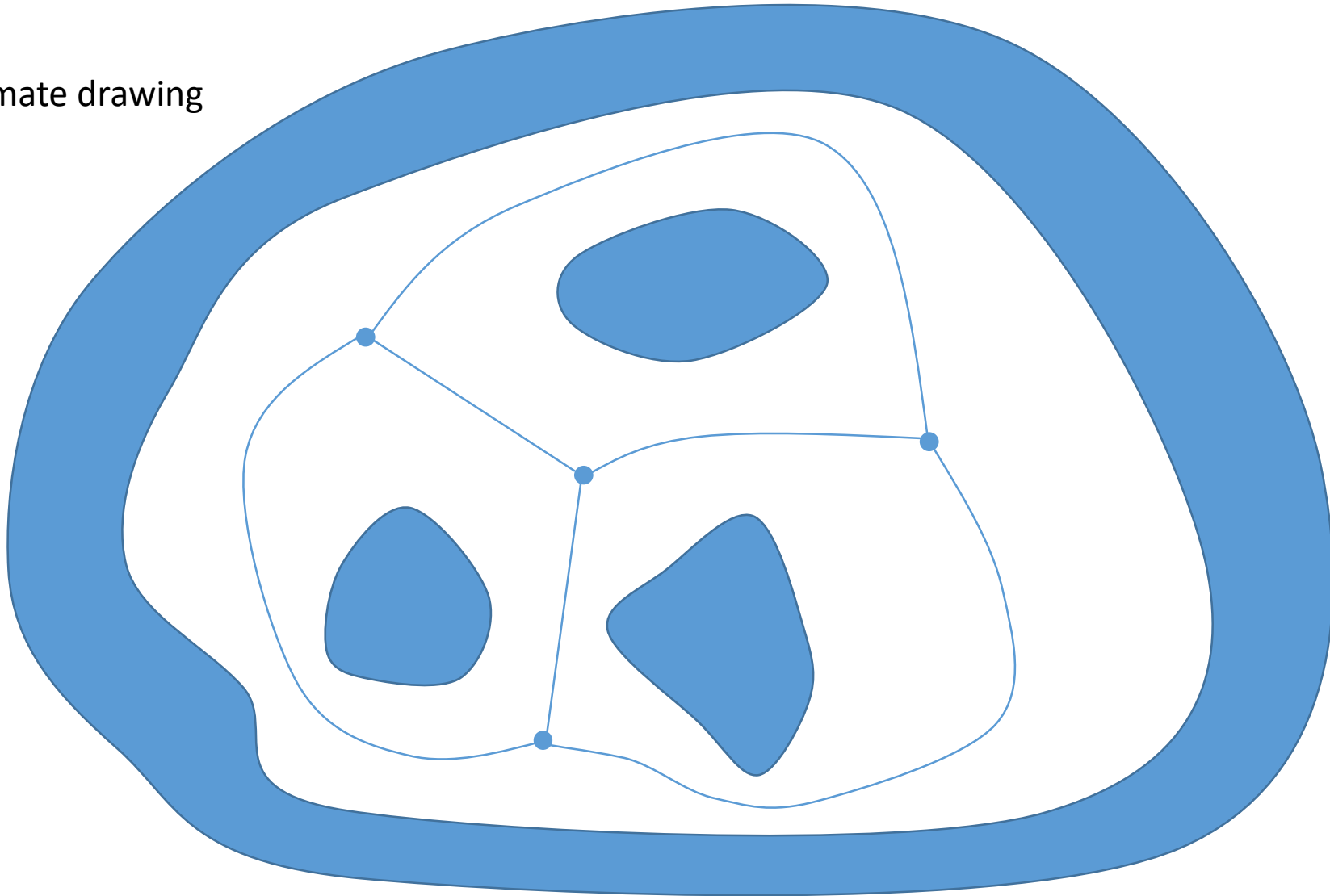


Extended Voronoi diagrams

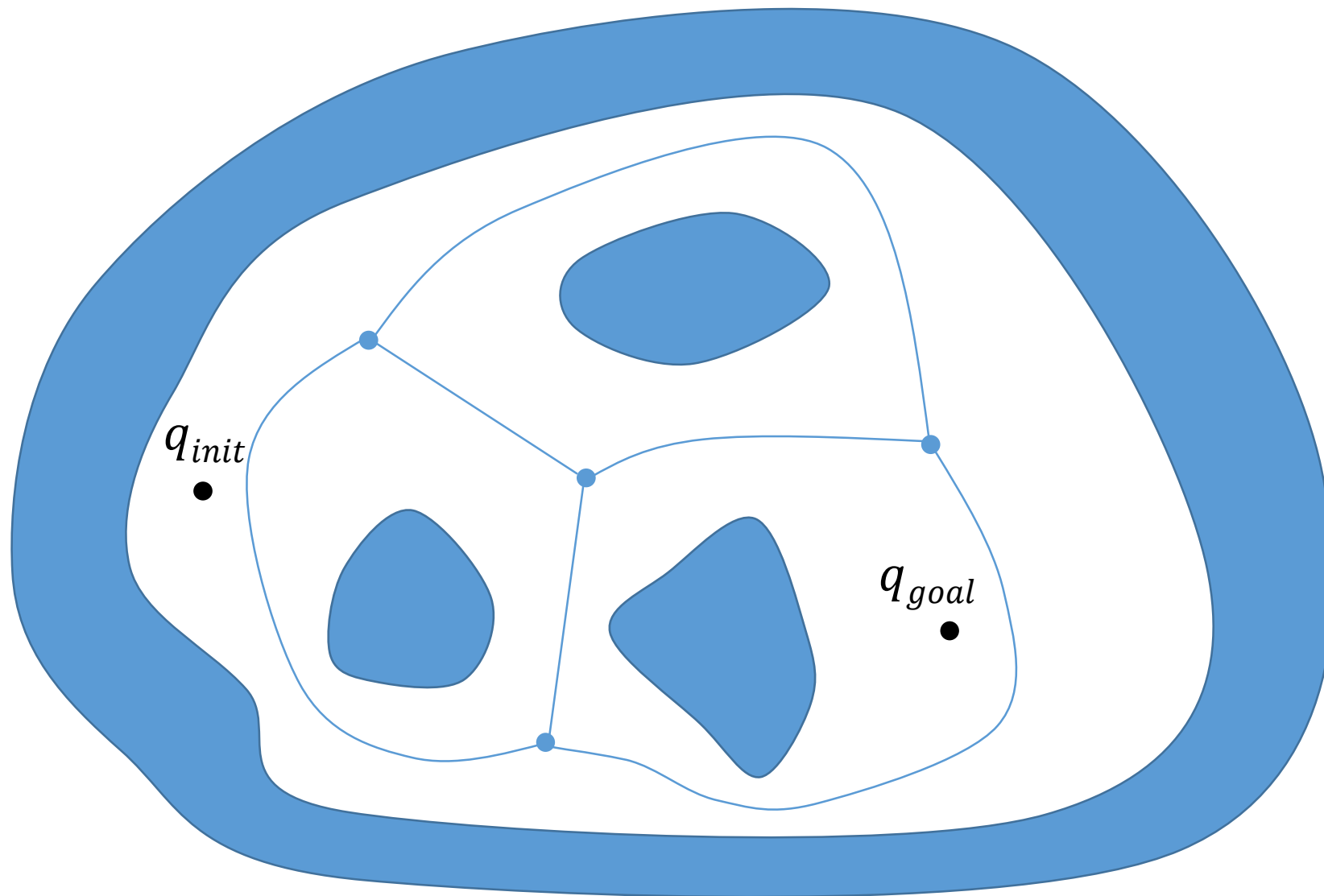


Extended Voronoi diagrams

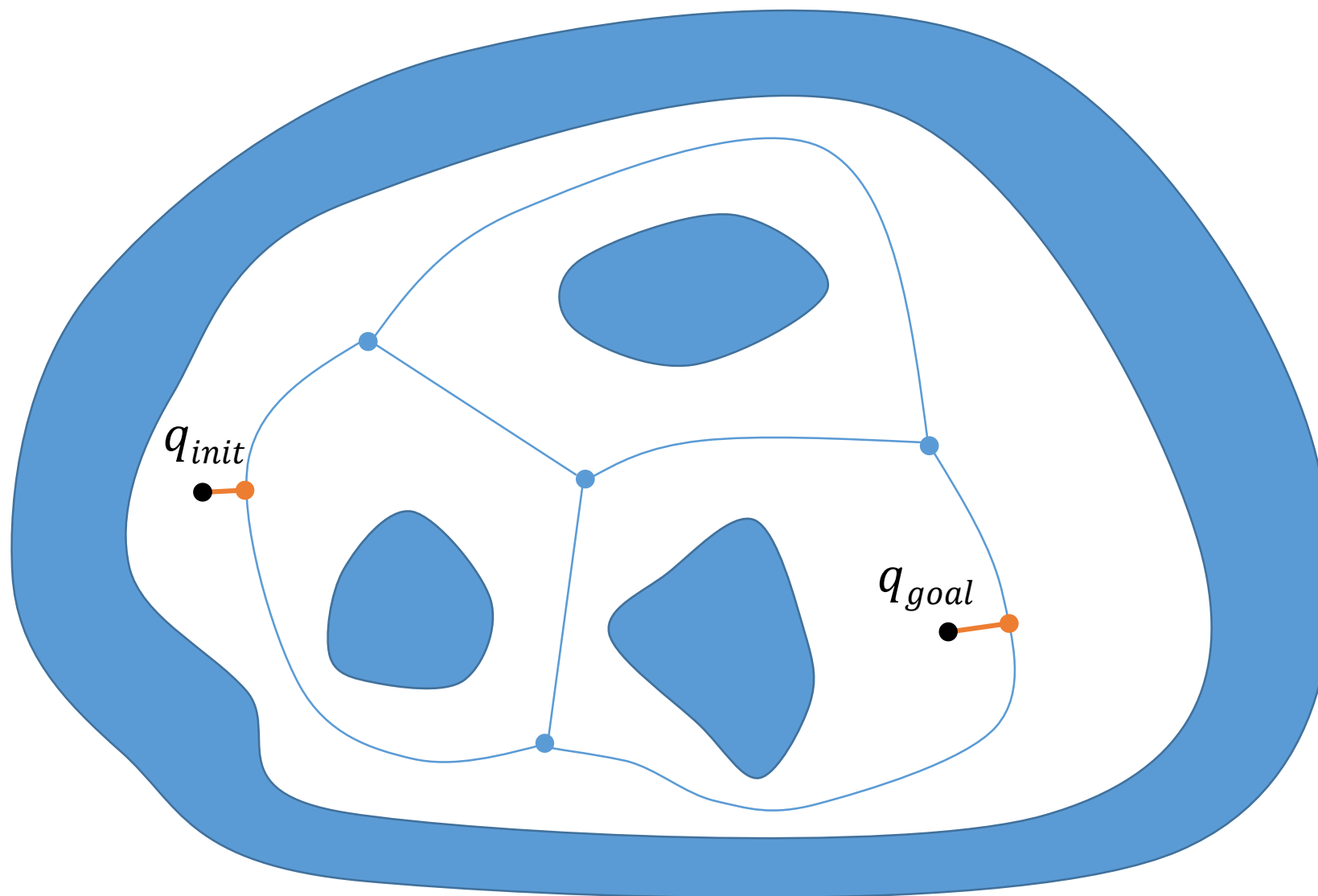
* very approximate drawing



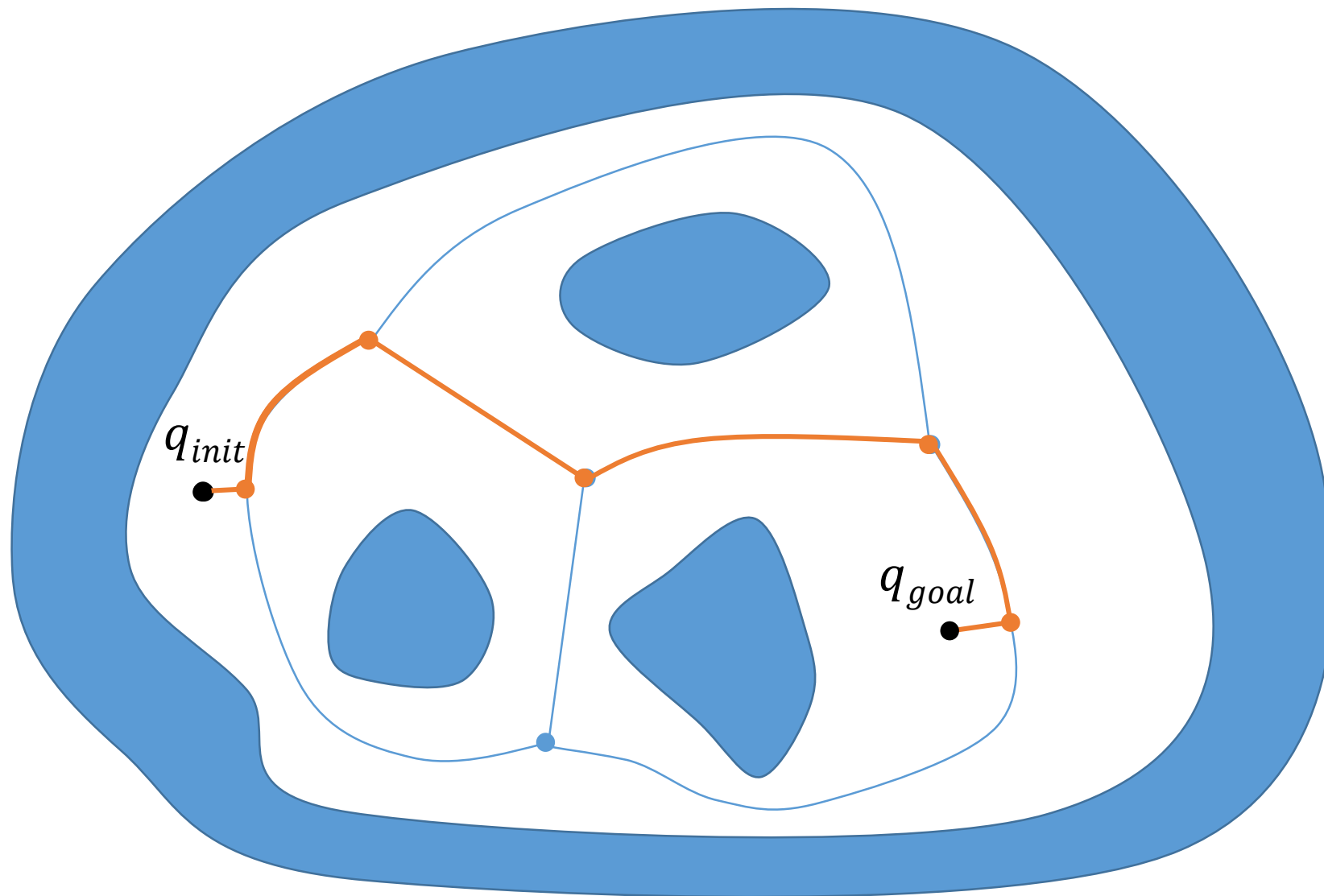
Extended Voronoi diagrams



Extended Voronoi diagrams

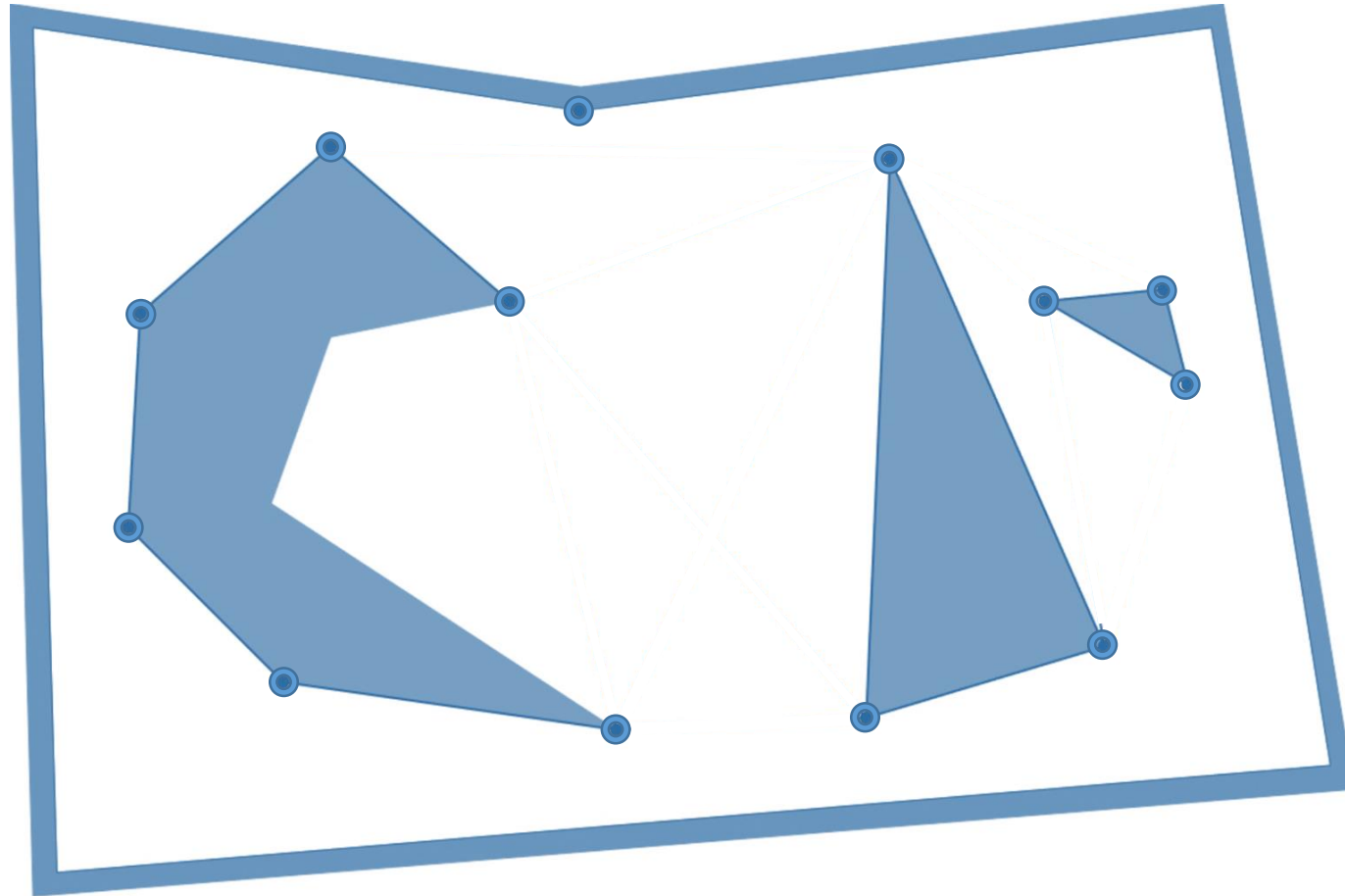


Extended Voronoi diagrams



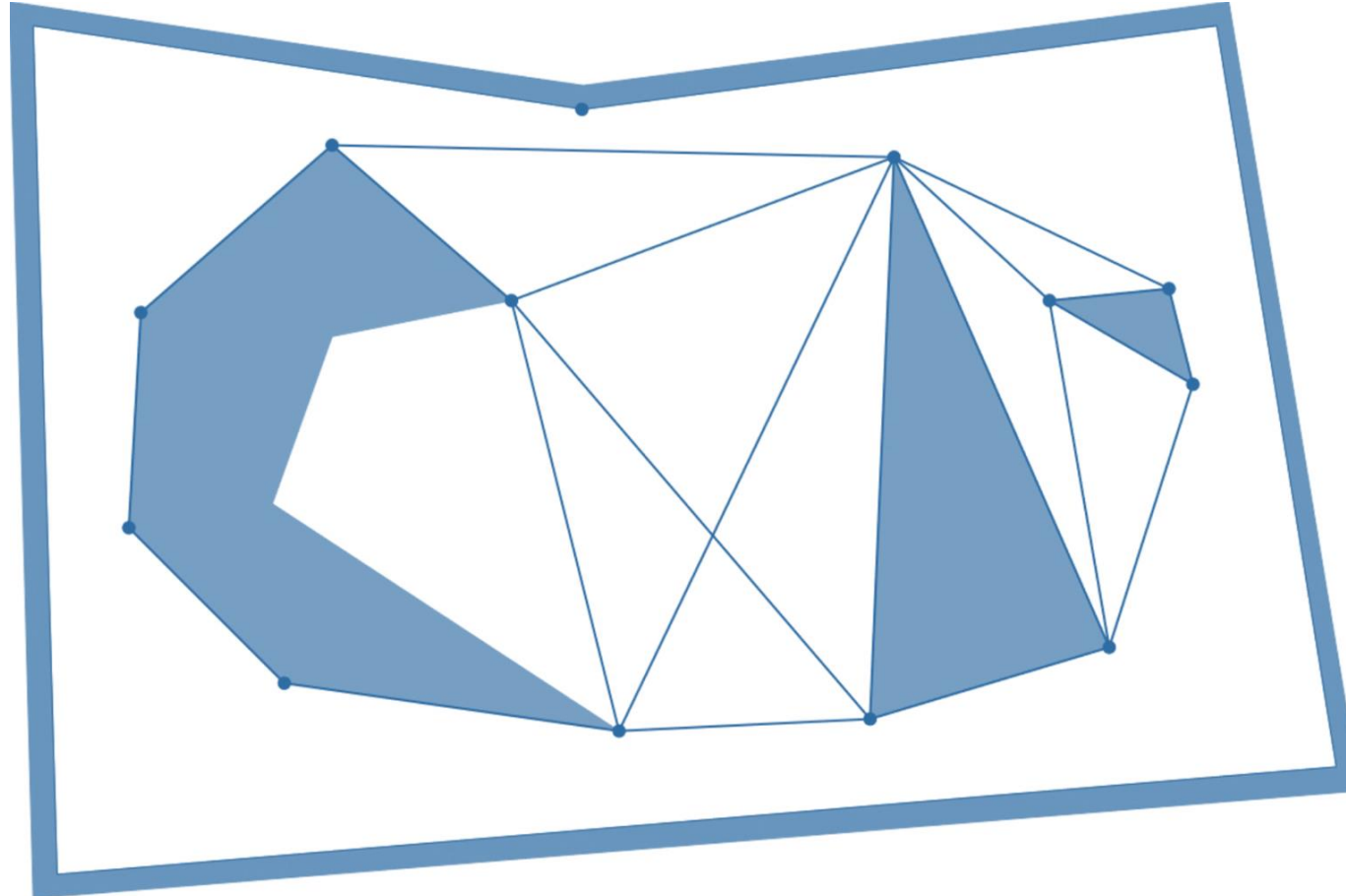
Reduced visibility roadmaps

- Find reflex vertices (vertices where interior polygon angle are less than π)



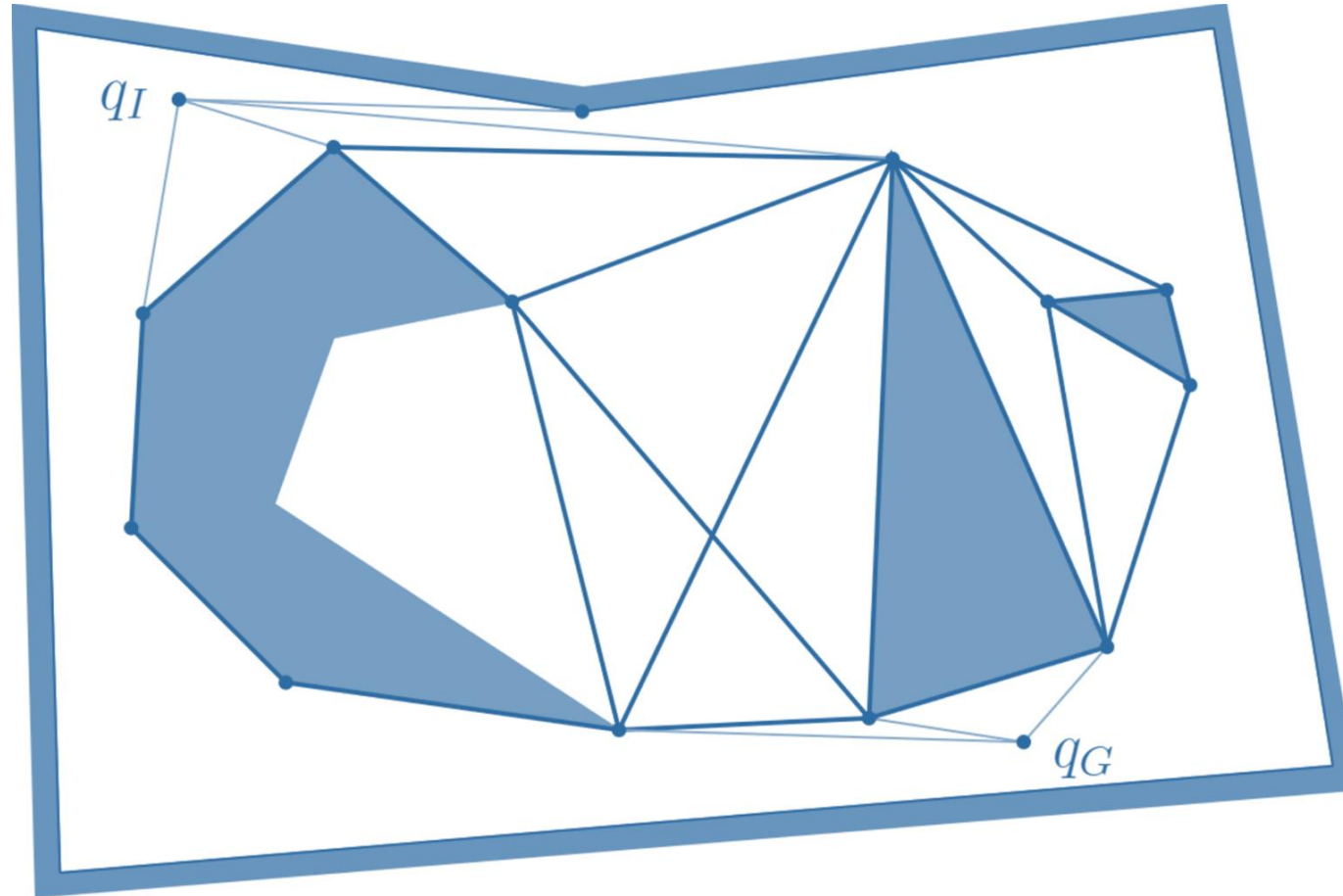
Reduced visibility roadmaps

- Connect:
 - consecutive reflex vertices on given polygon
 - bitangent edges (edges between two reflex vertices on different polygons such that both polygons are on the same side)



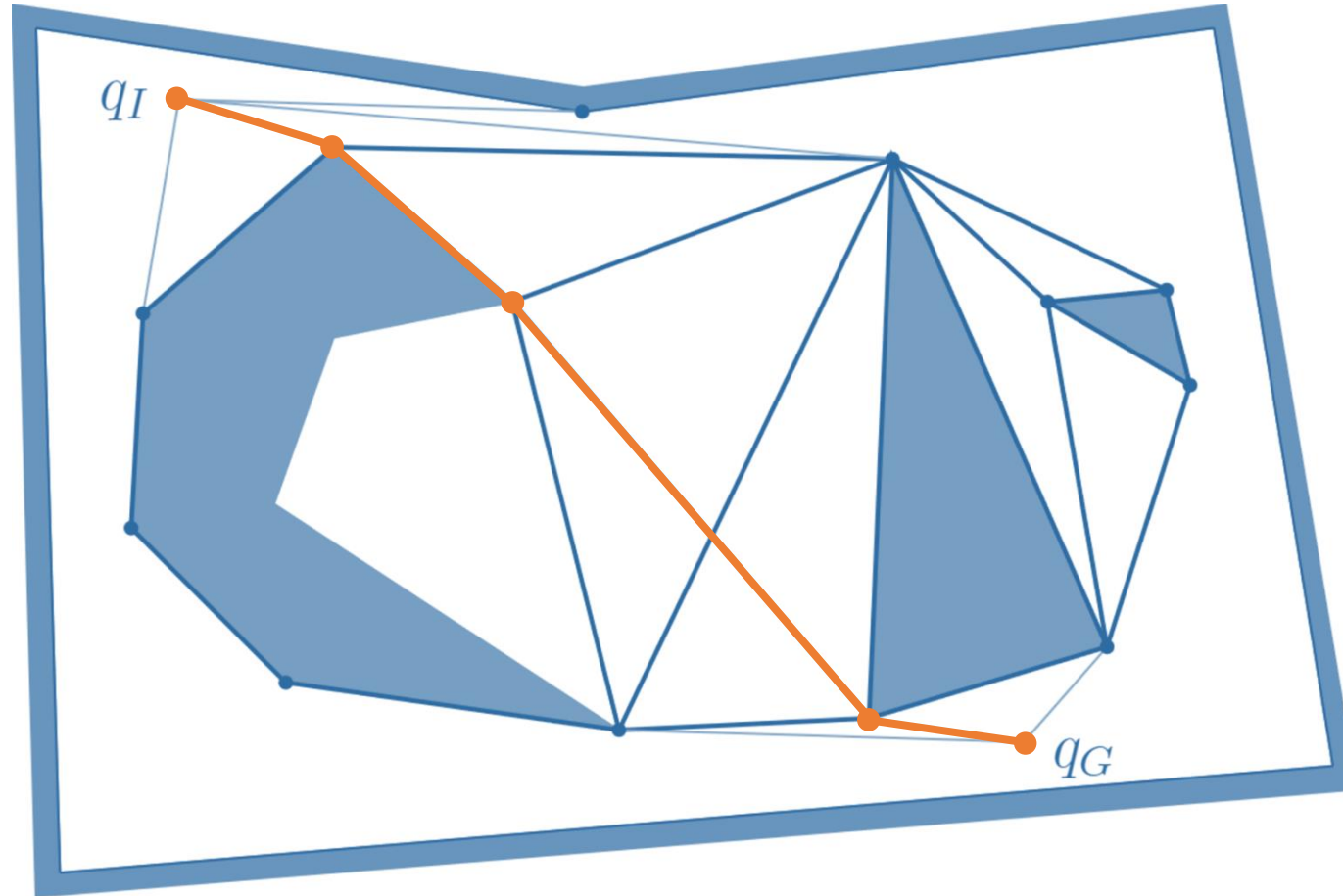
Reduced visibility roadmaps

- Connect initial and goal points to all visible reflex vertices

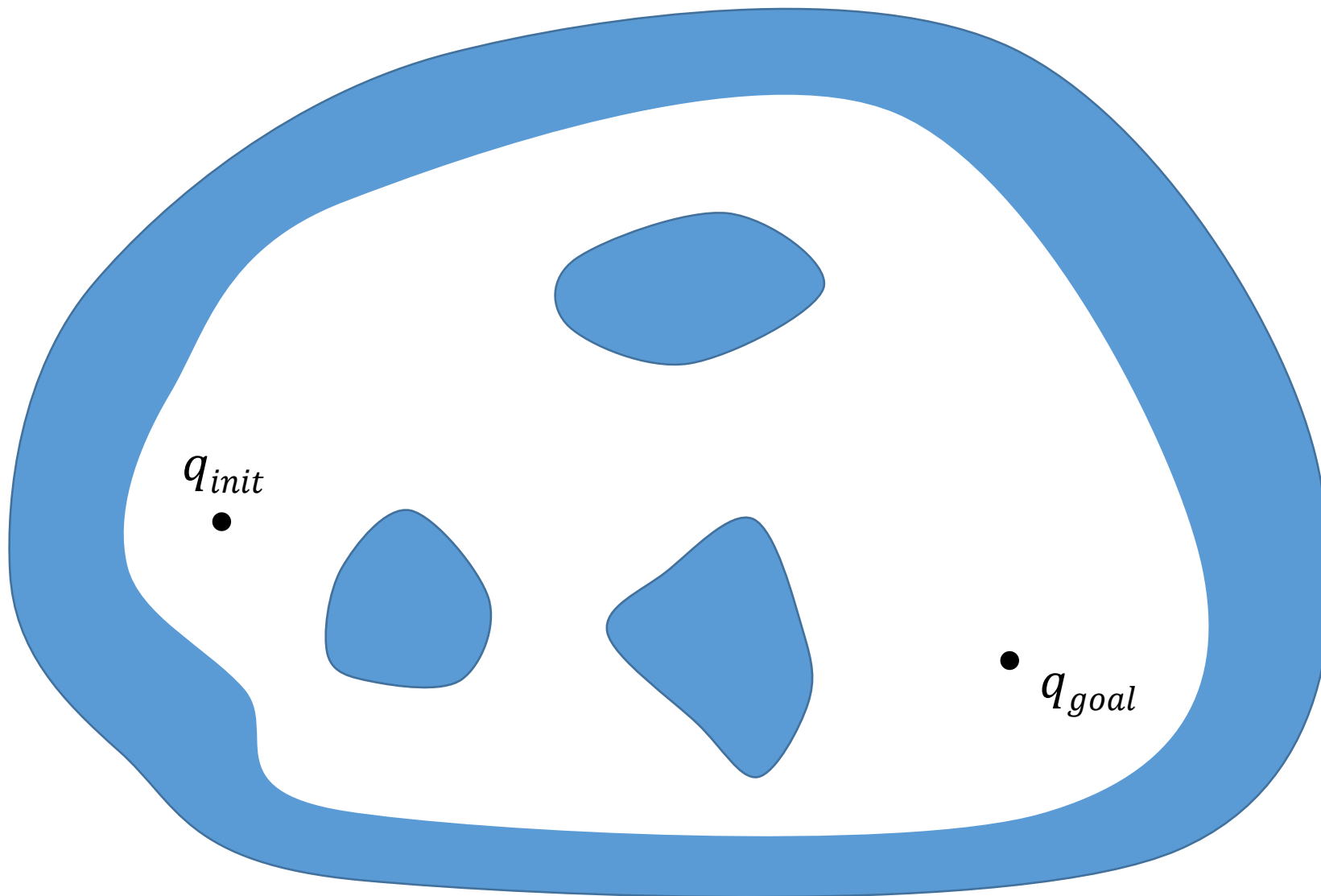


Reduced visibility roadmaps

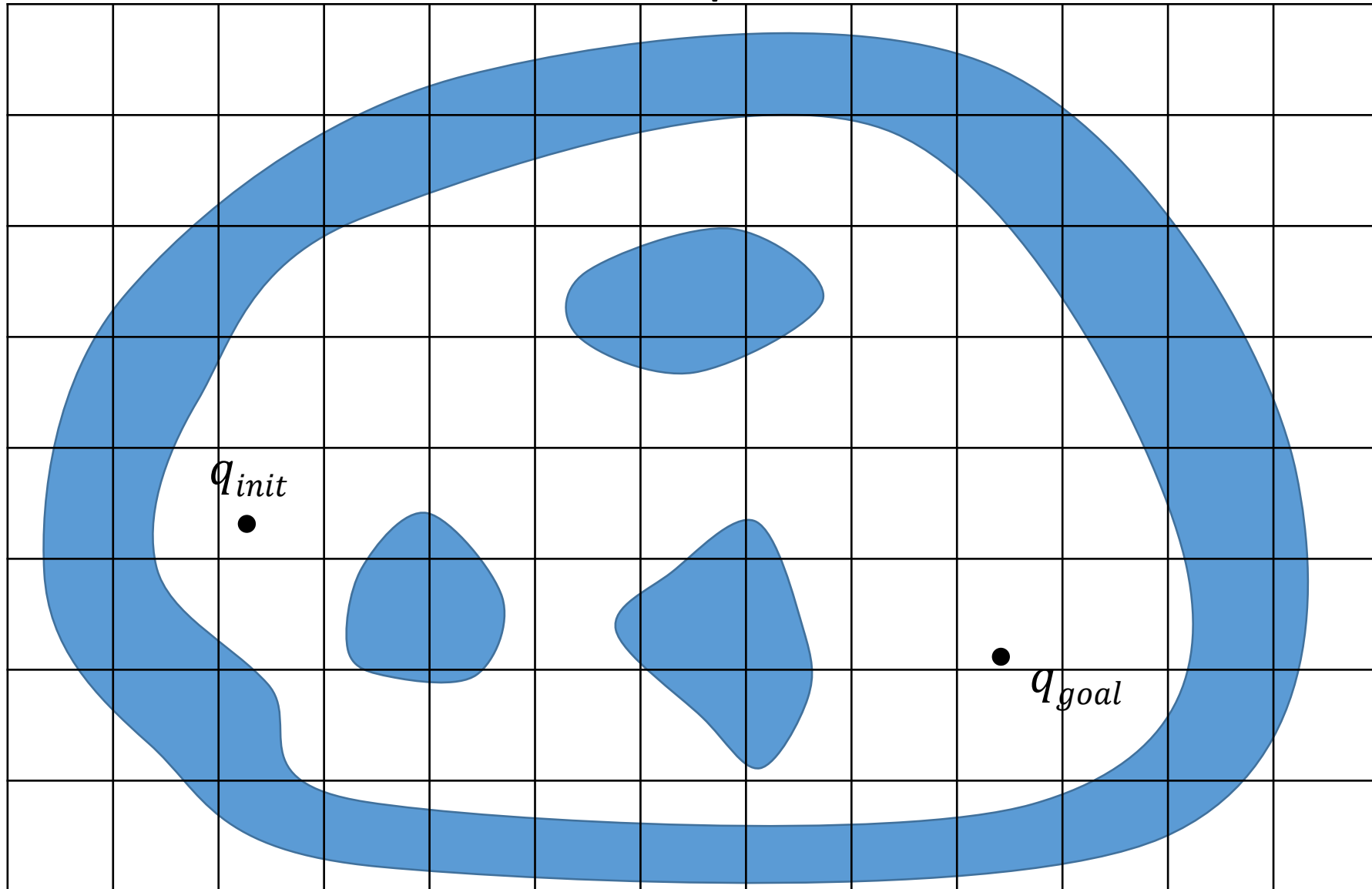
- run favorite graph search algorithm



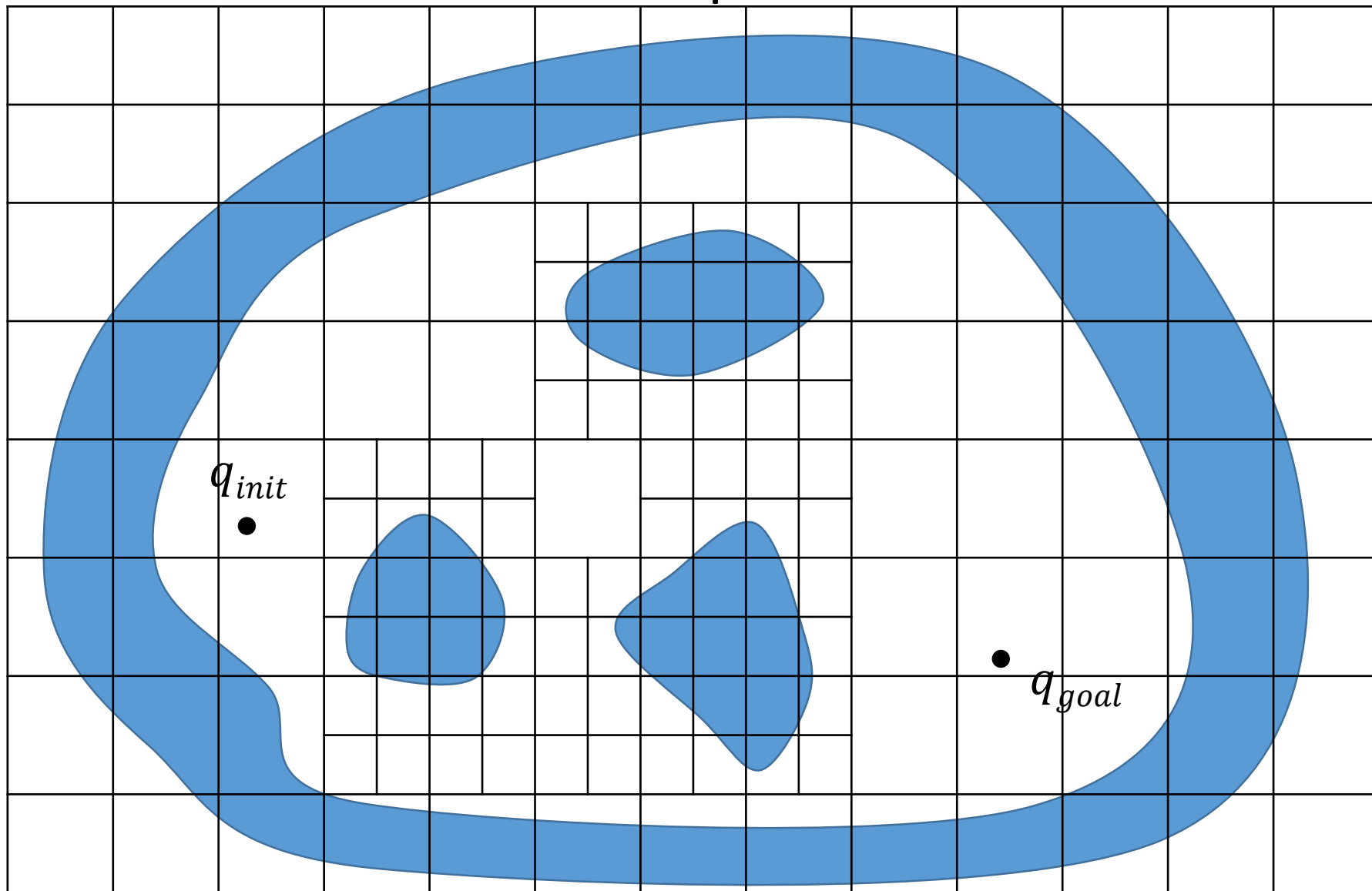
Cell decomposition



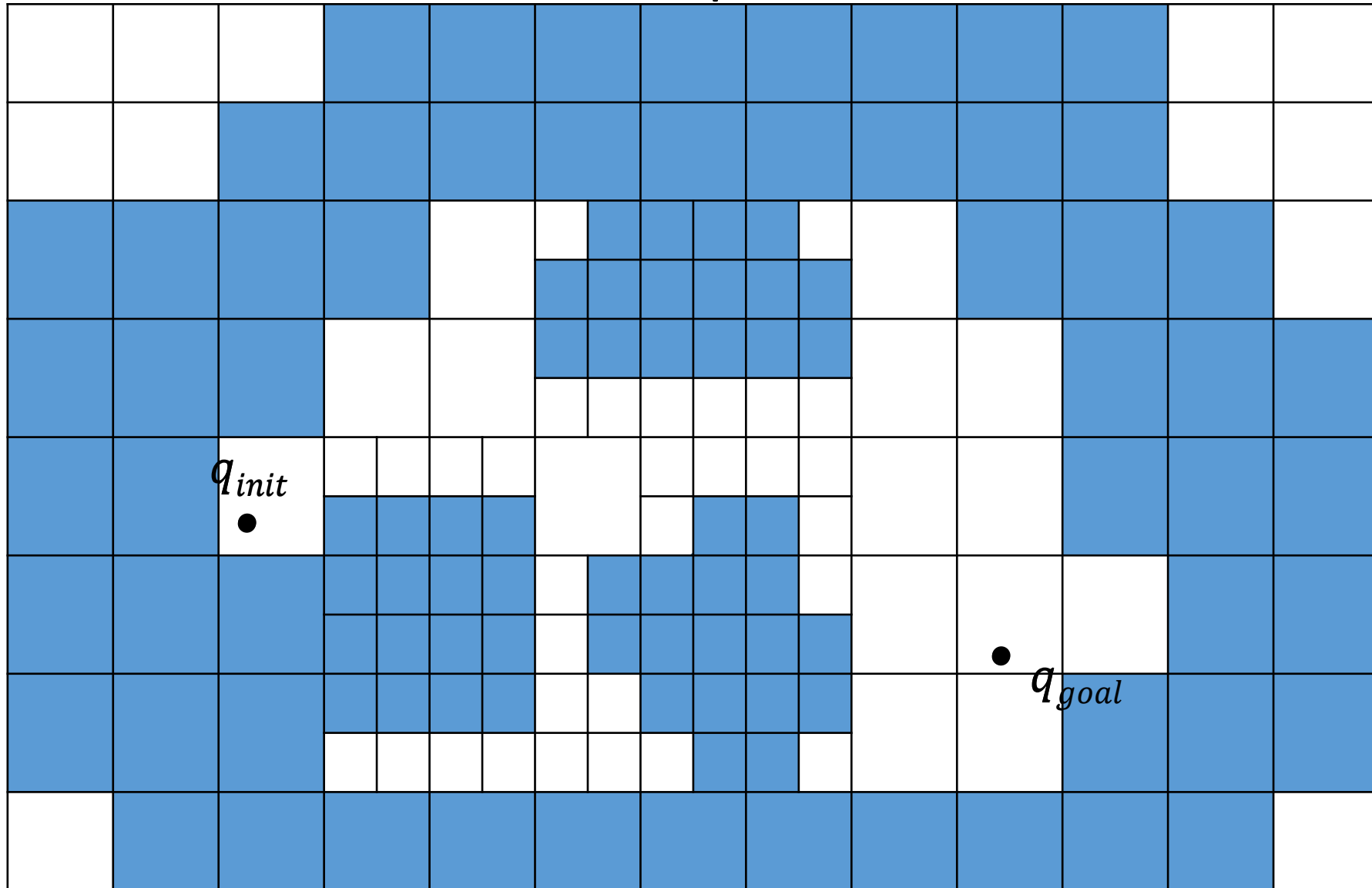
Approximate cell decomposition



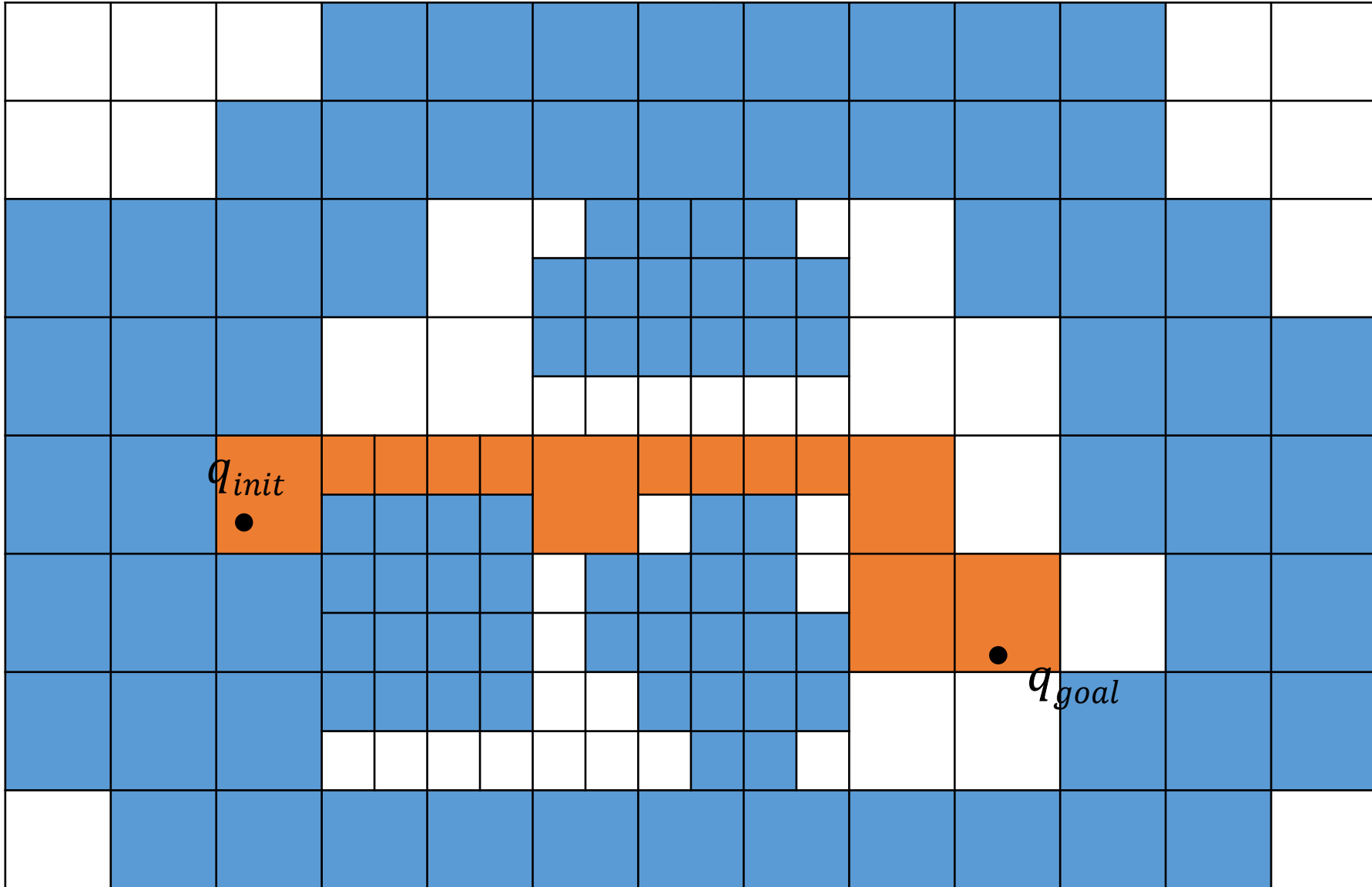
Approximate cell decomposition



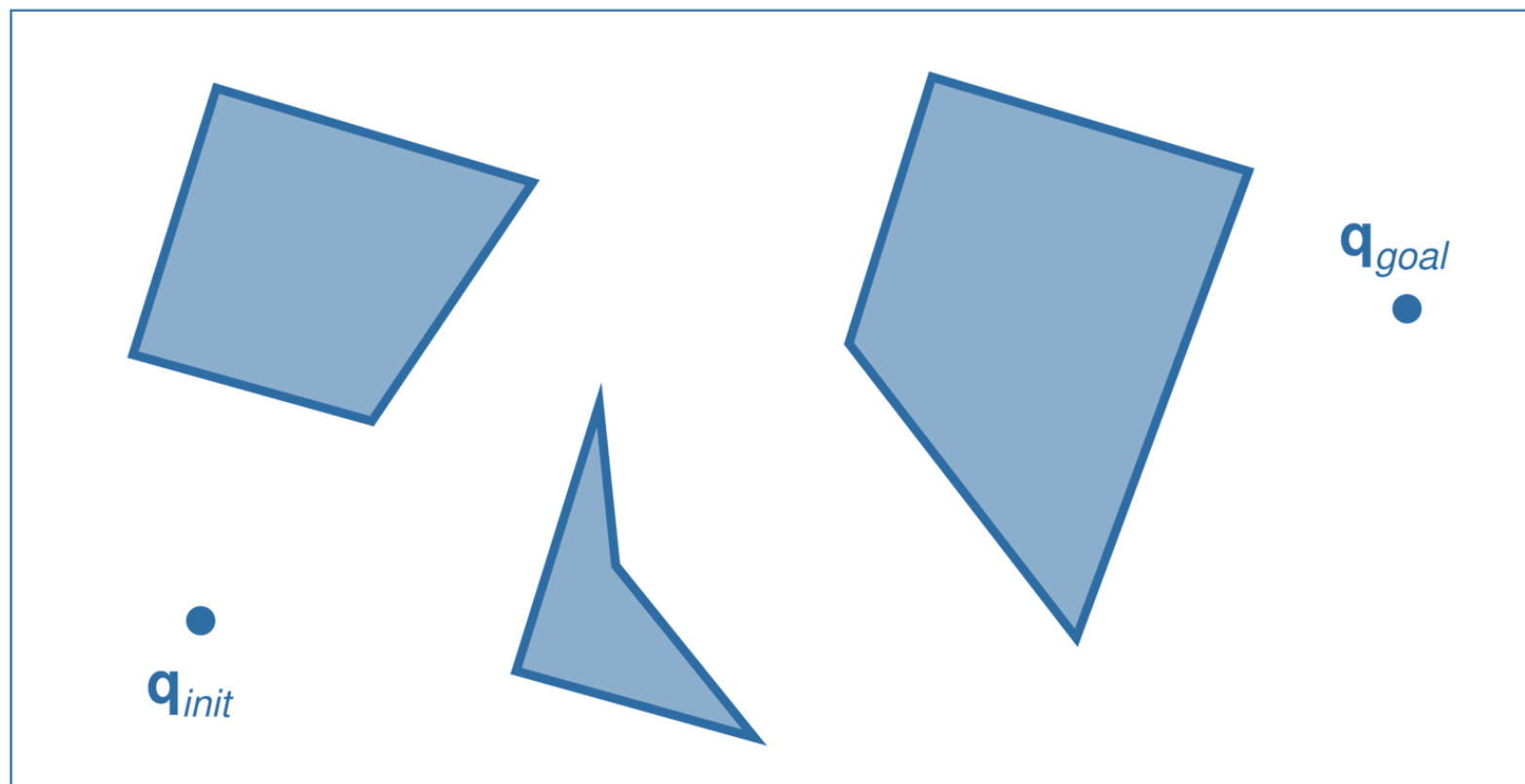
Approximate cell decomposition



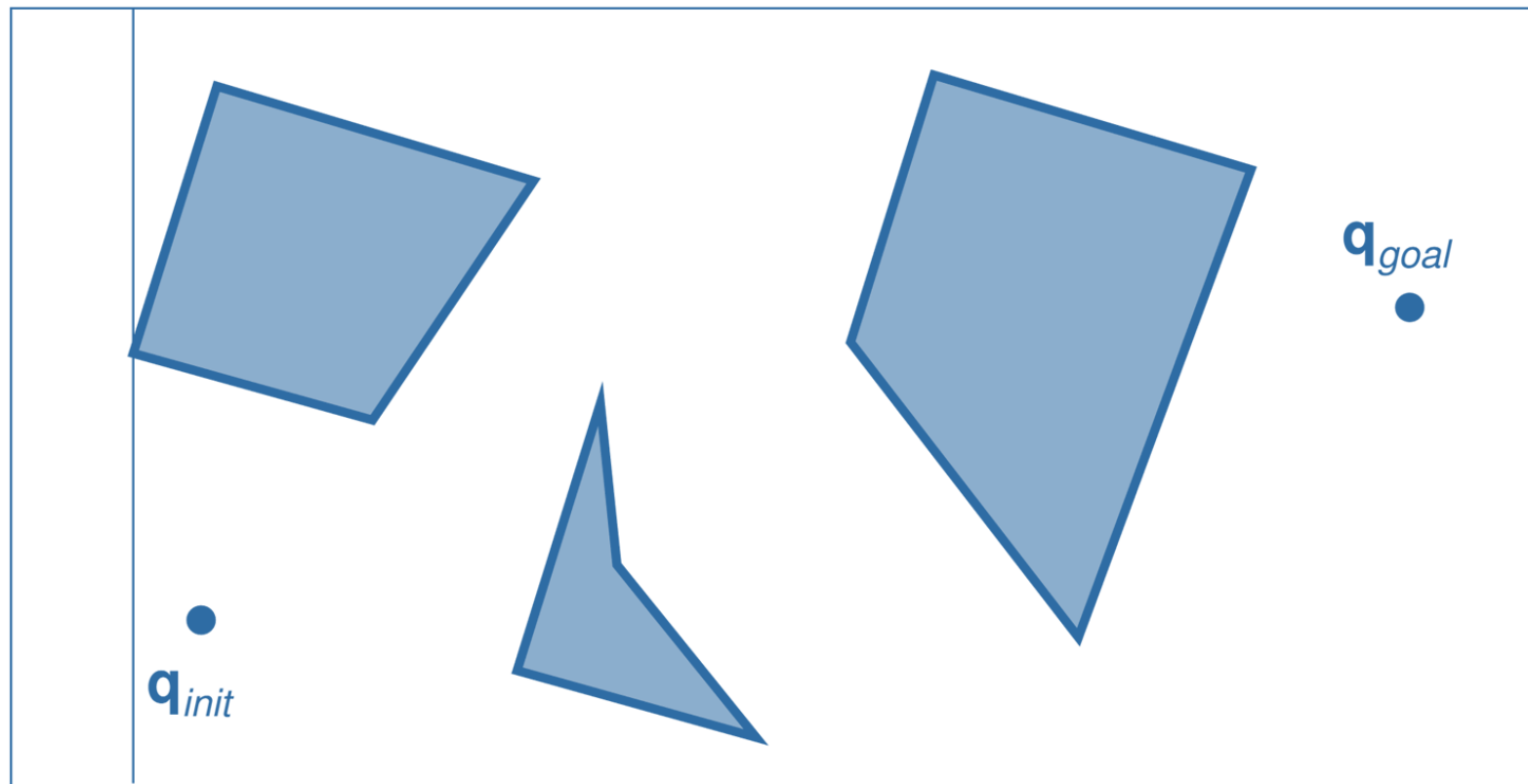
Approximate cell decomposition



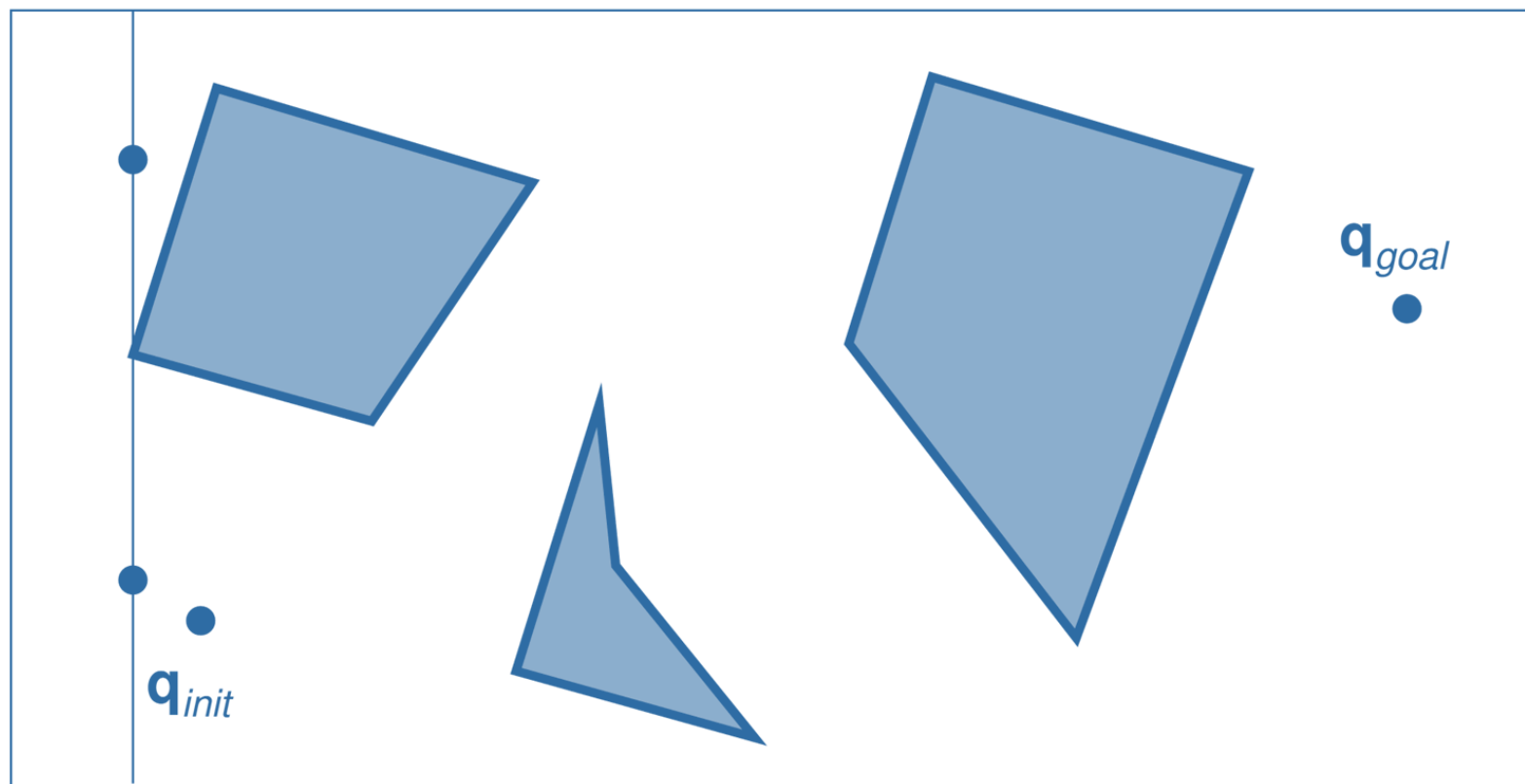
Vertical cell decomposition



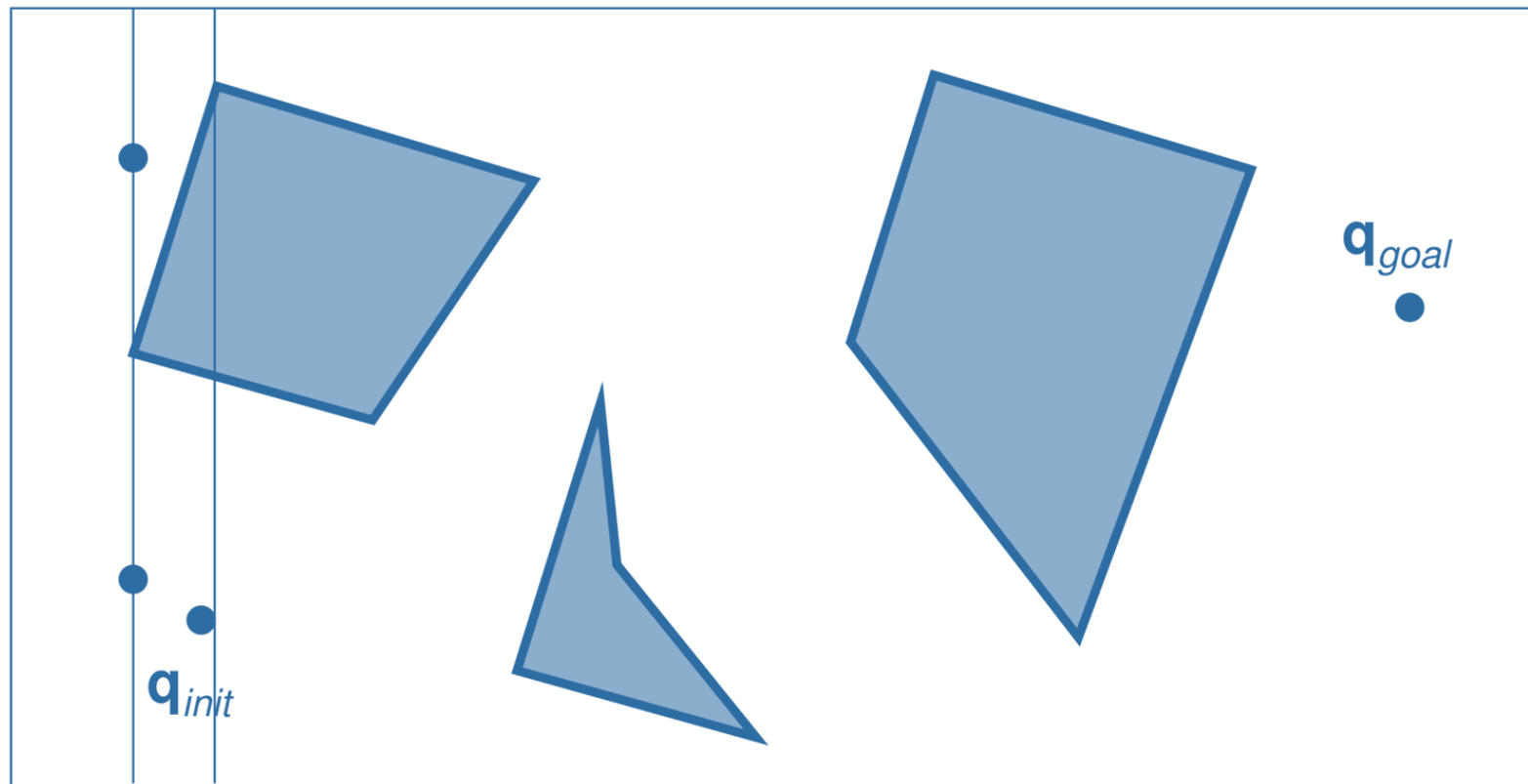
Vertical cell decomposition



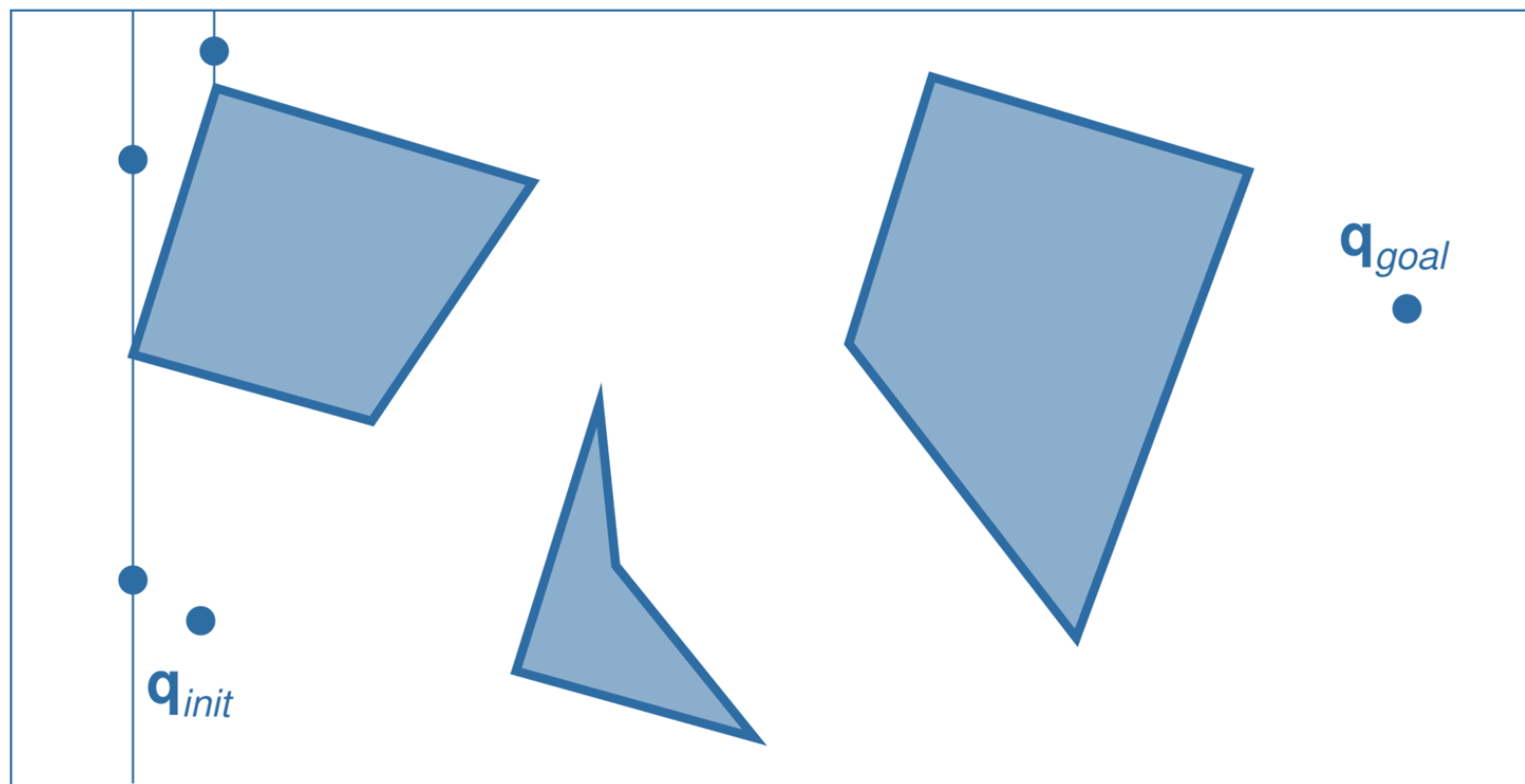
Vertical cell decomposition



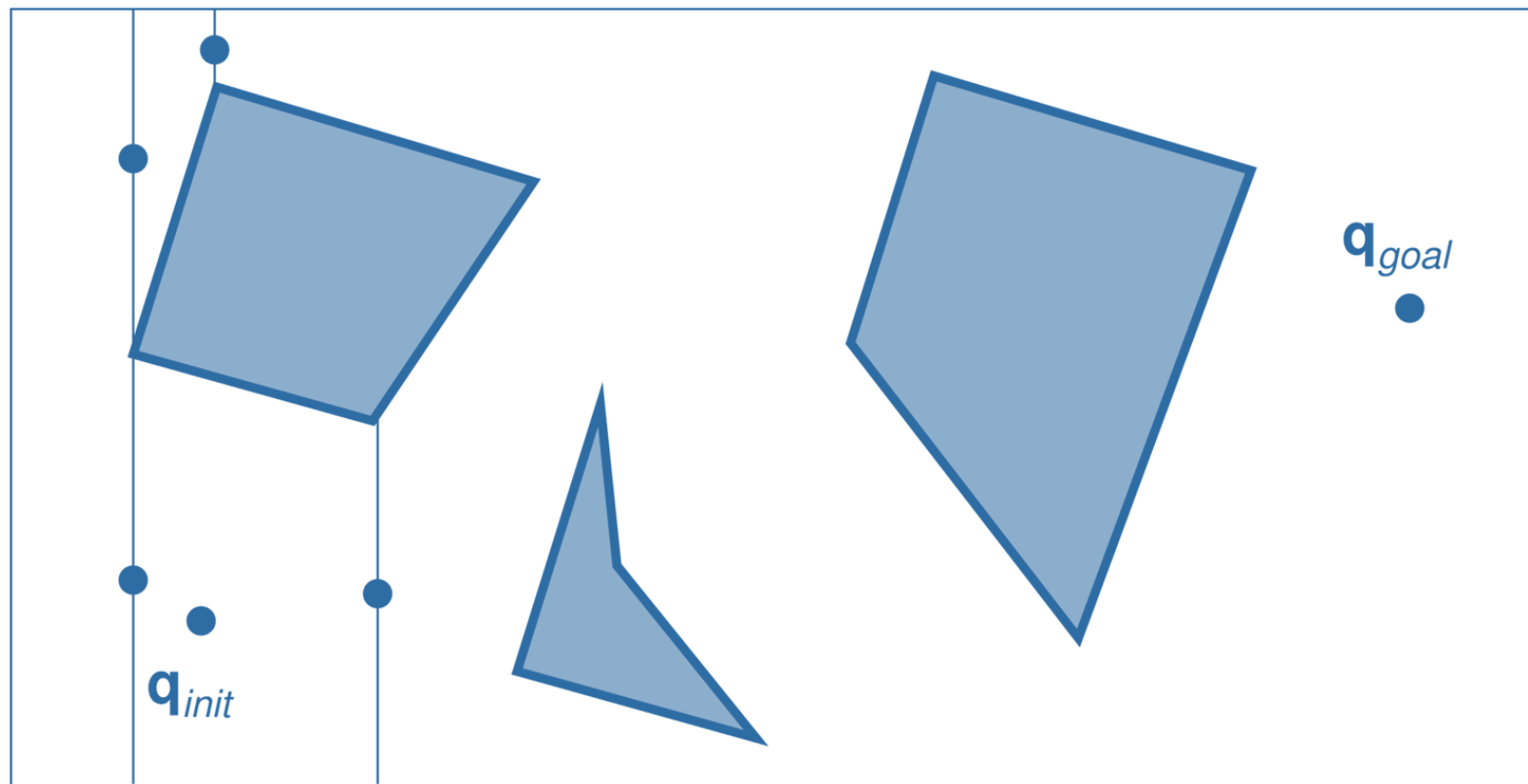
Vertical cell decomposition



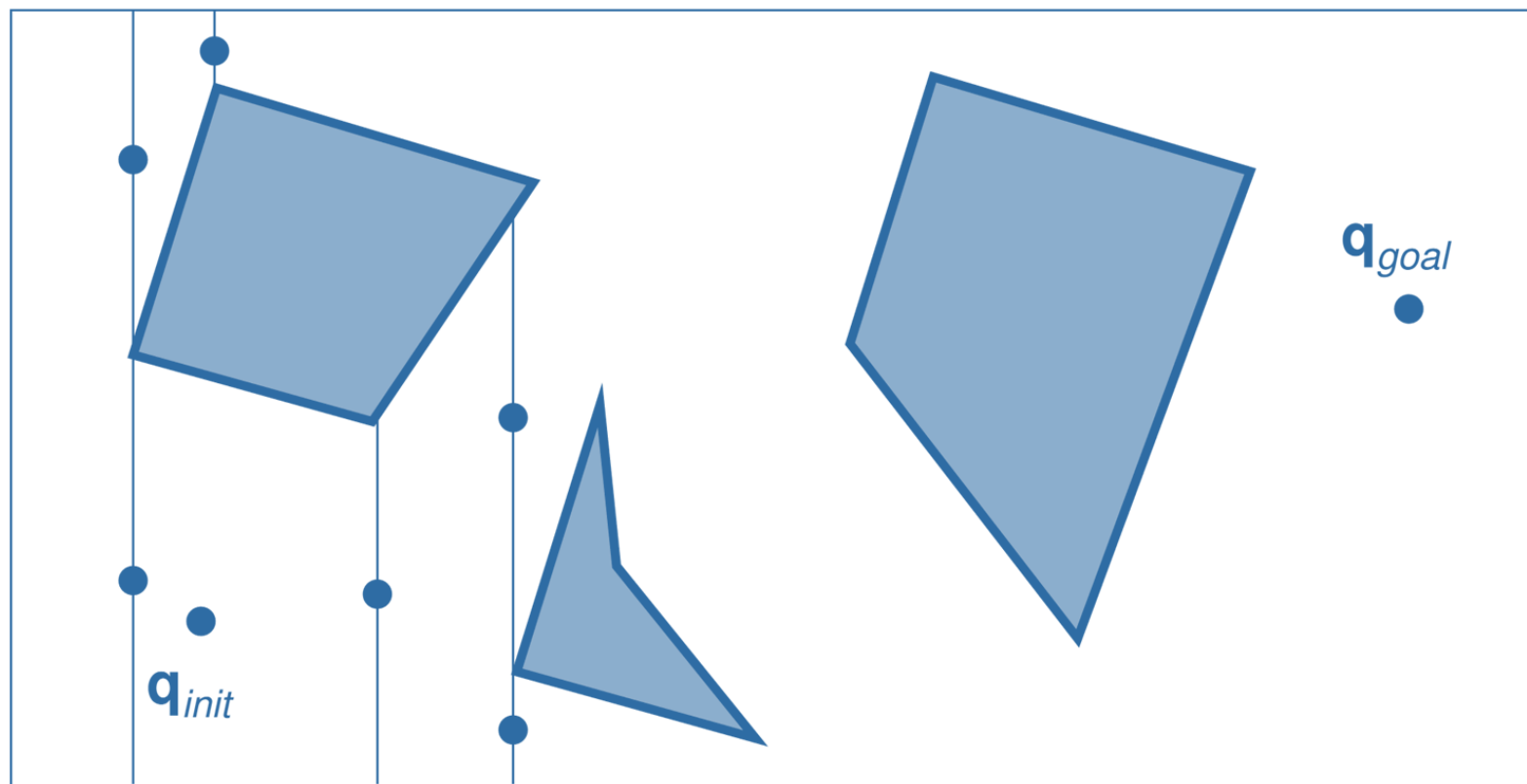
Vertical cell decomposition



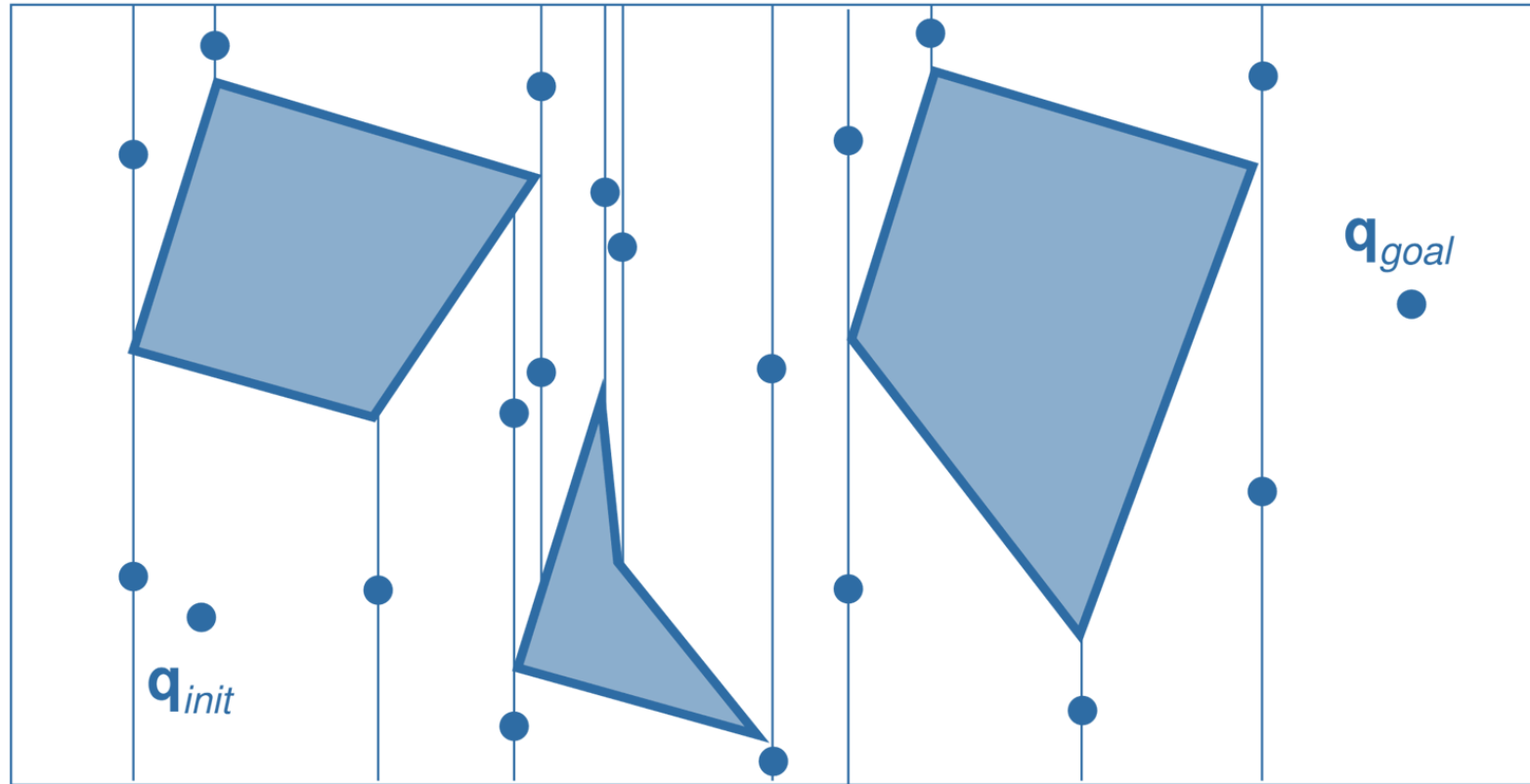
Vertical cell decomposition



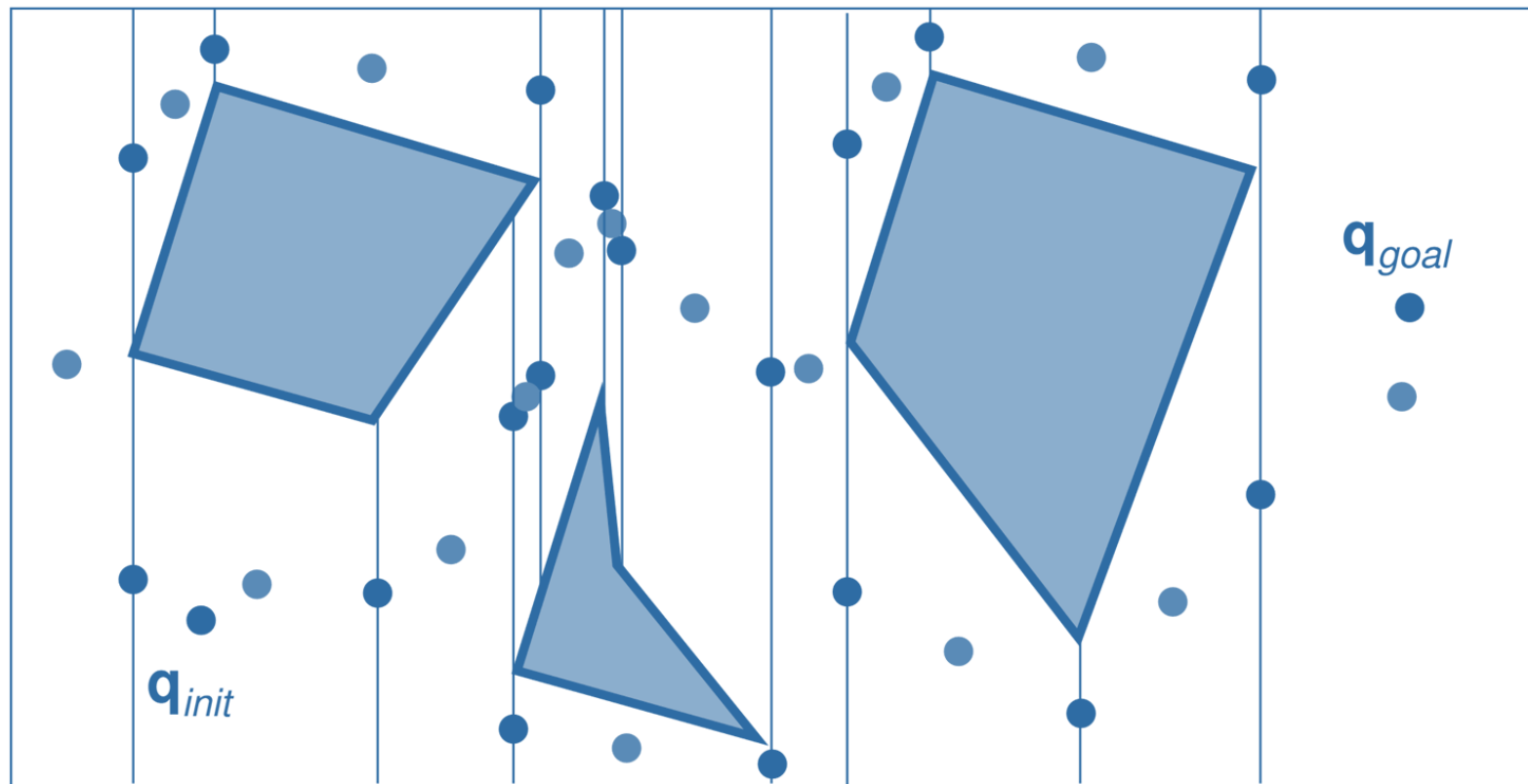
Vertical cell decomposition



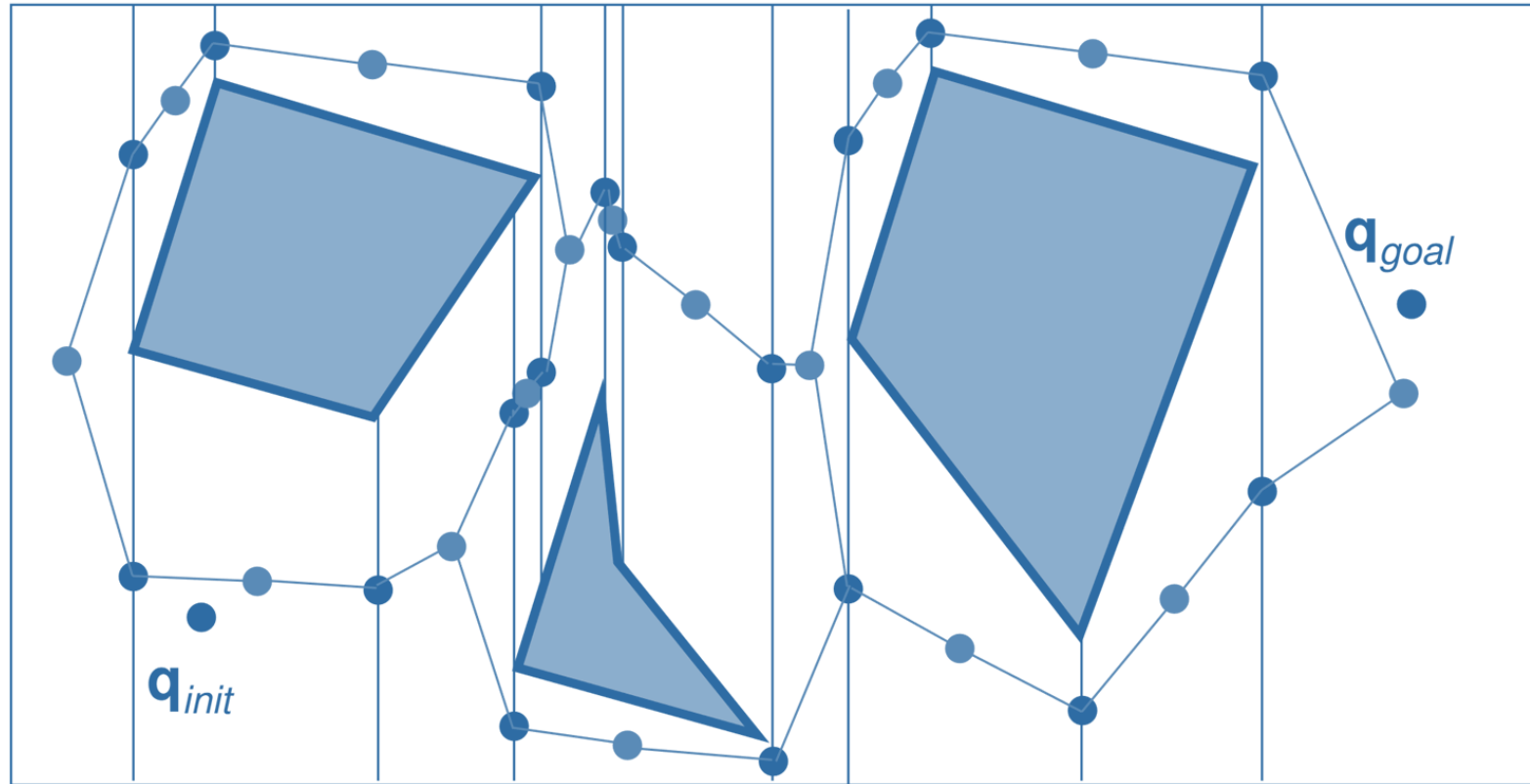
Vertical cell decomposition



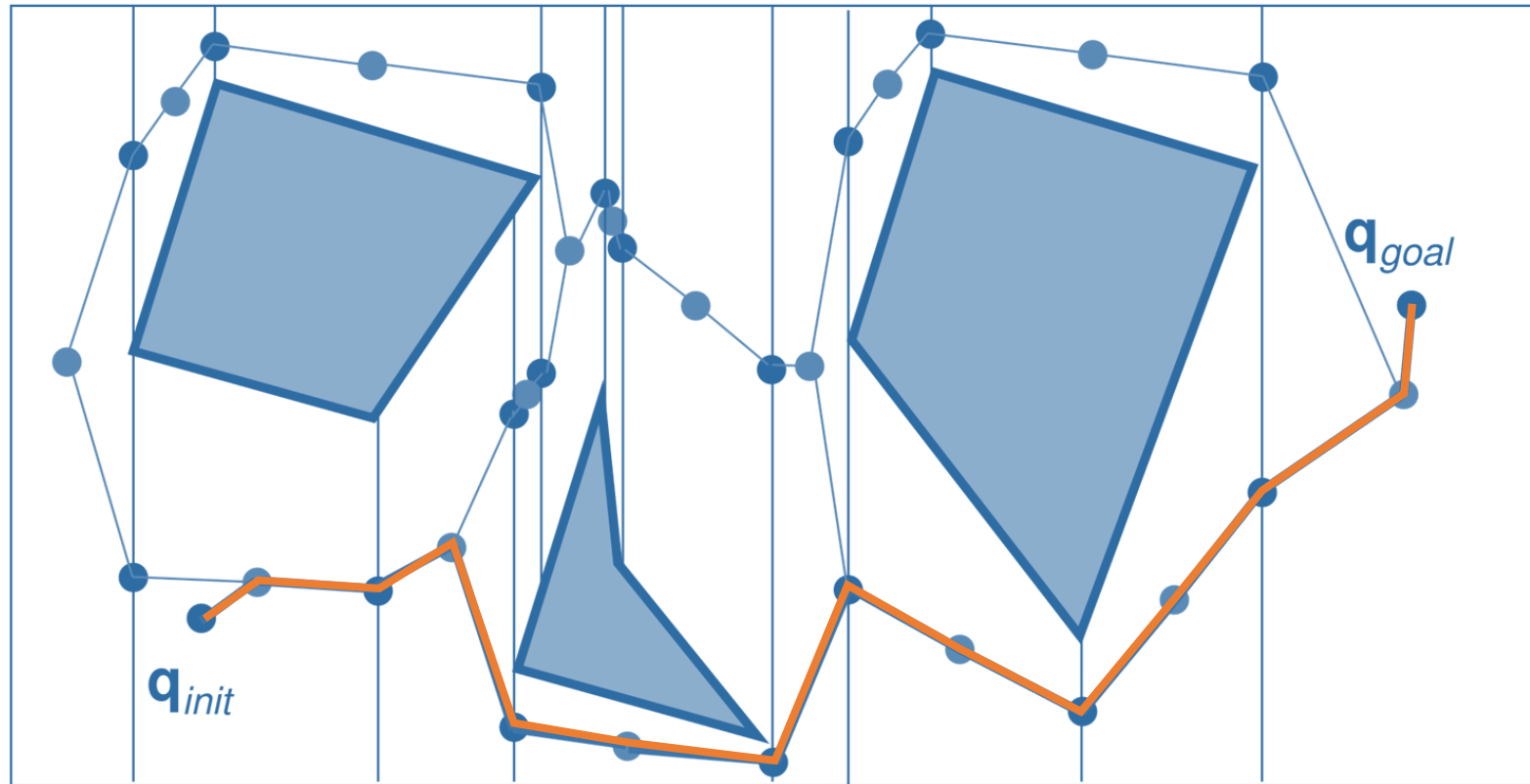
Vertical cell decomposition



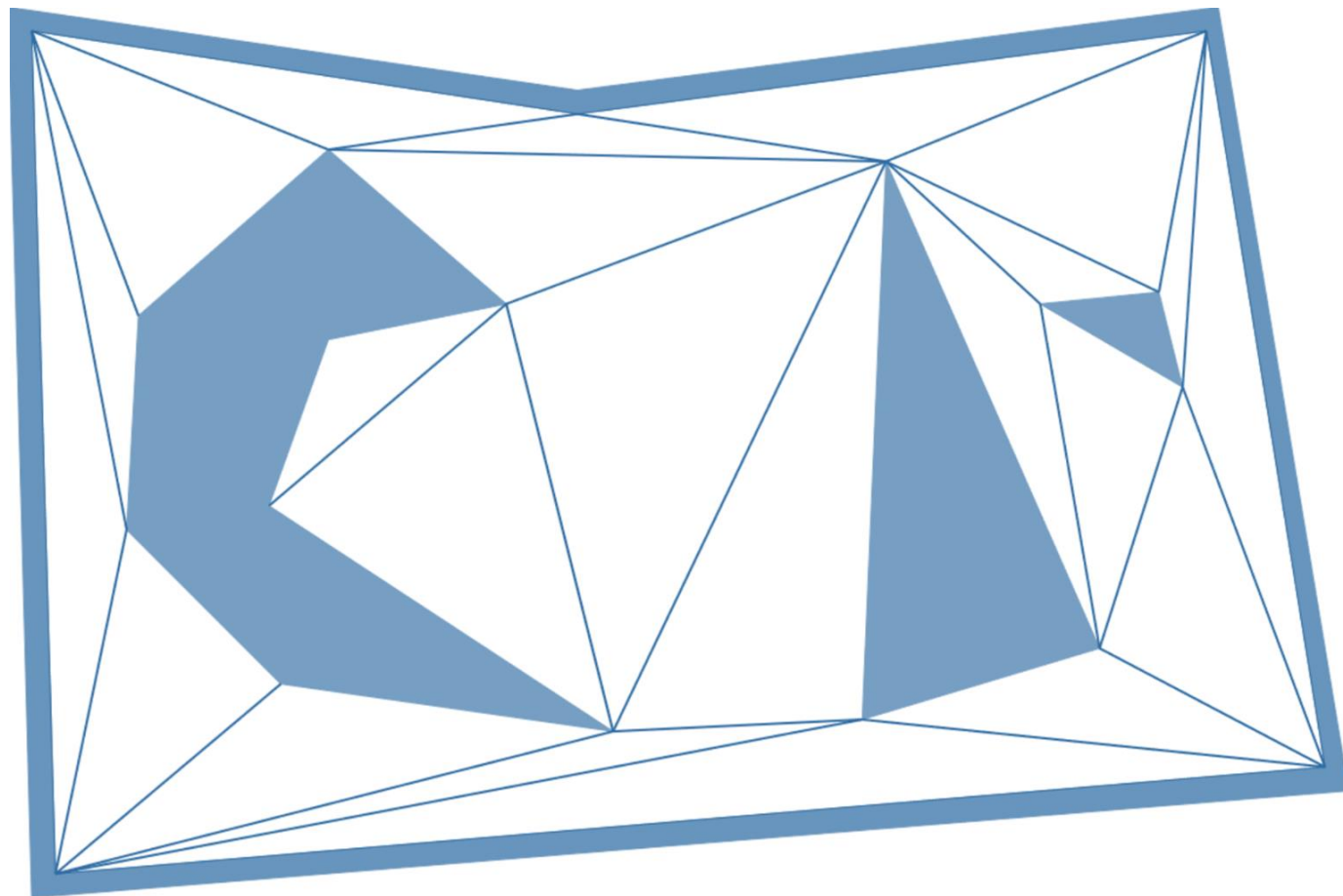
Vertical cell decomposition



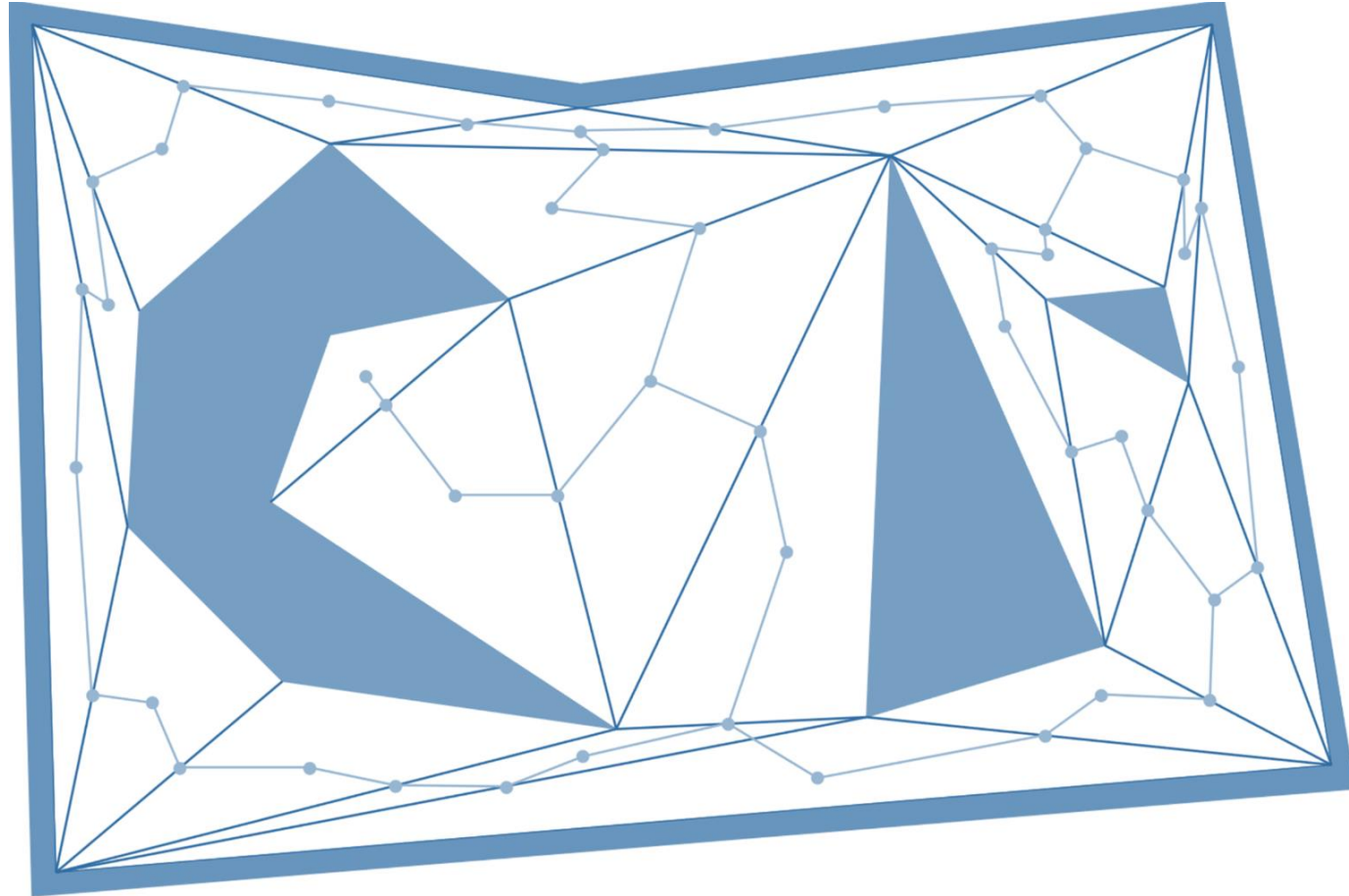
Vertical cell decomposition



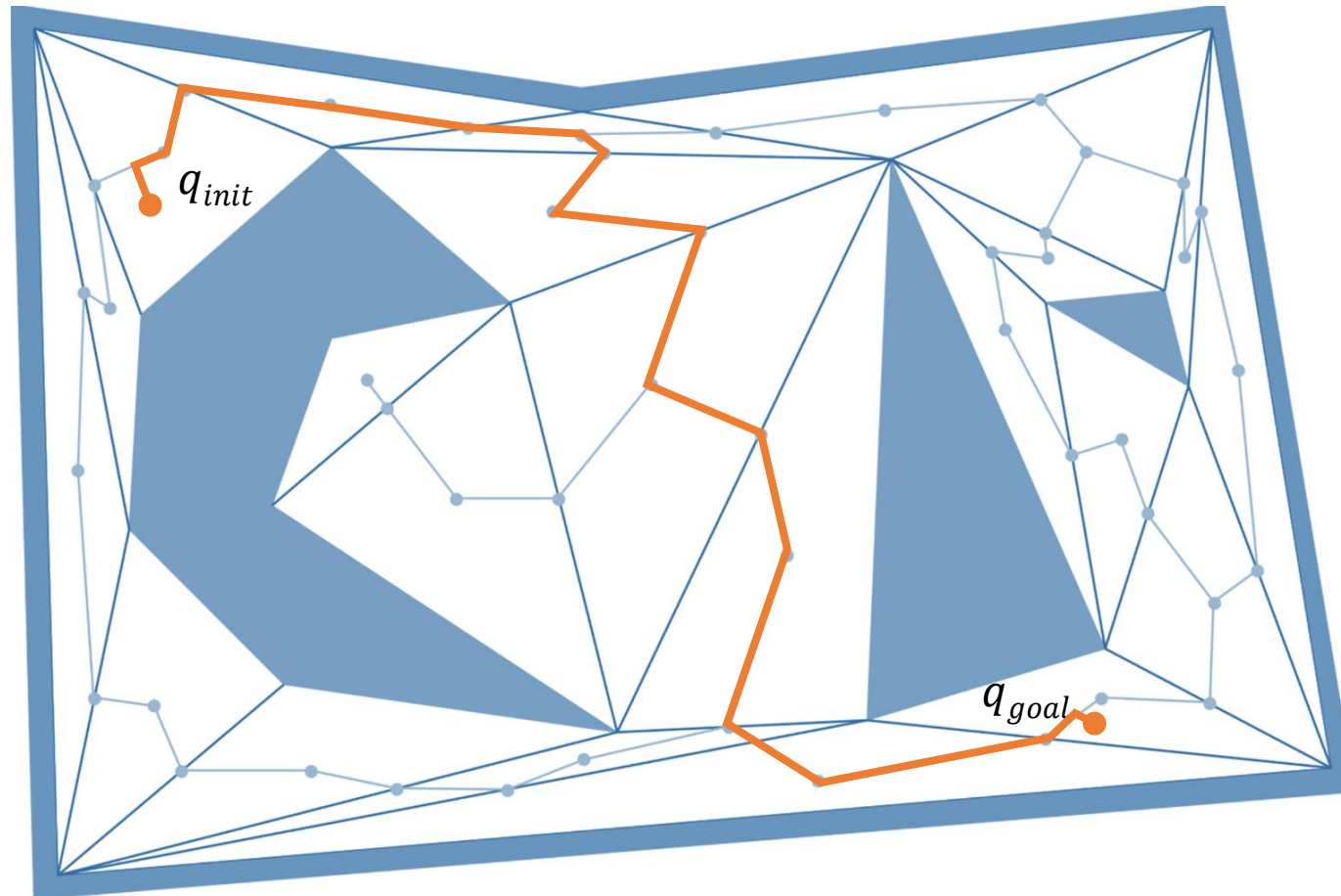
Triangulation



Triangulation



Triangulation

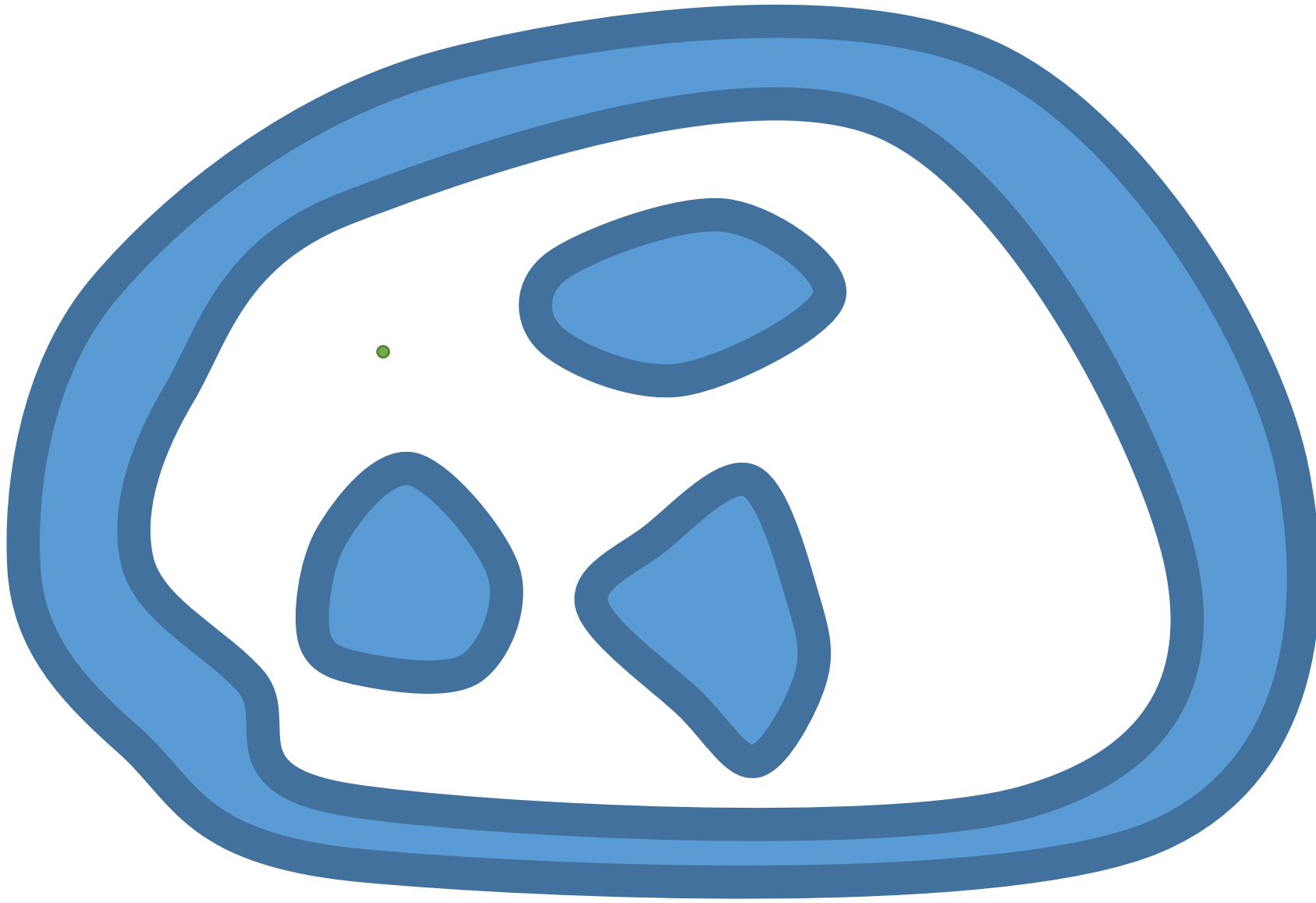


From path planning to motion
planning

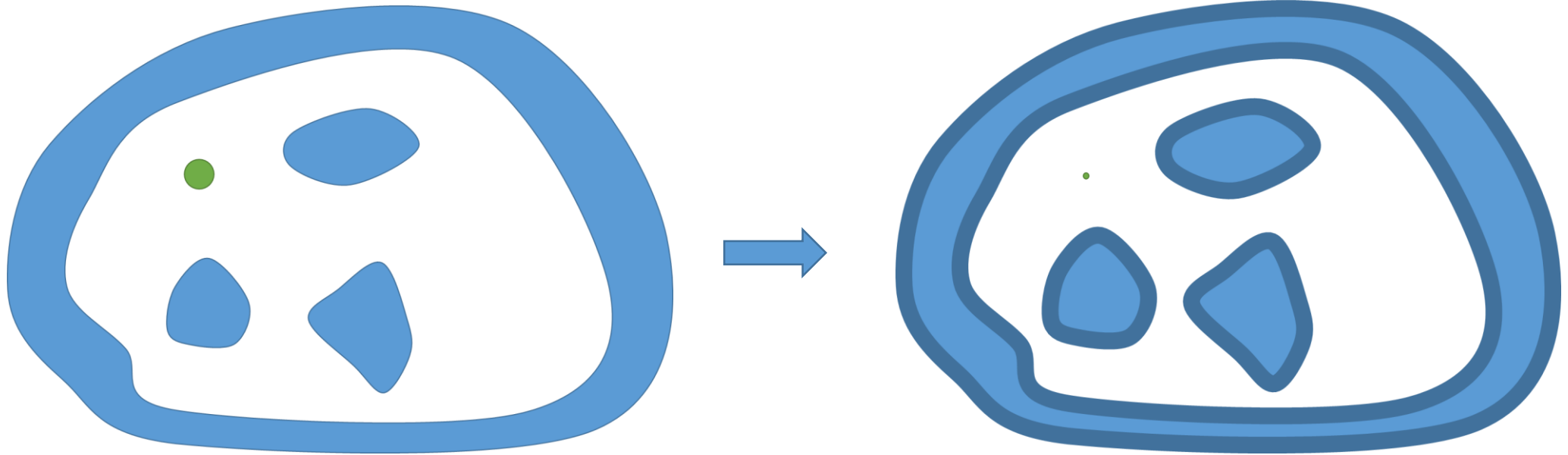
What if our subject is not a point?



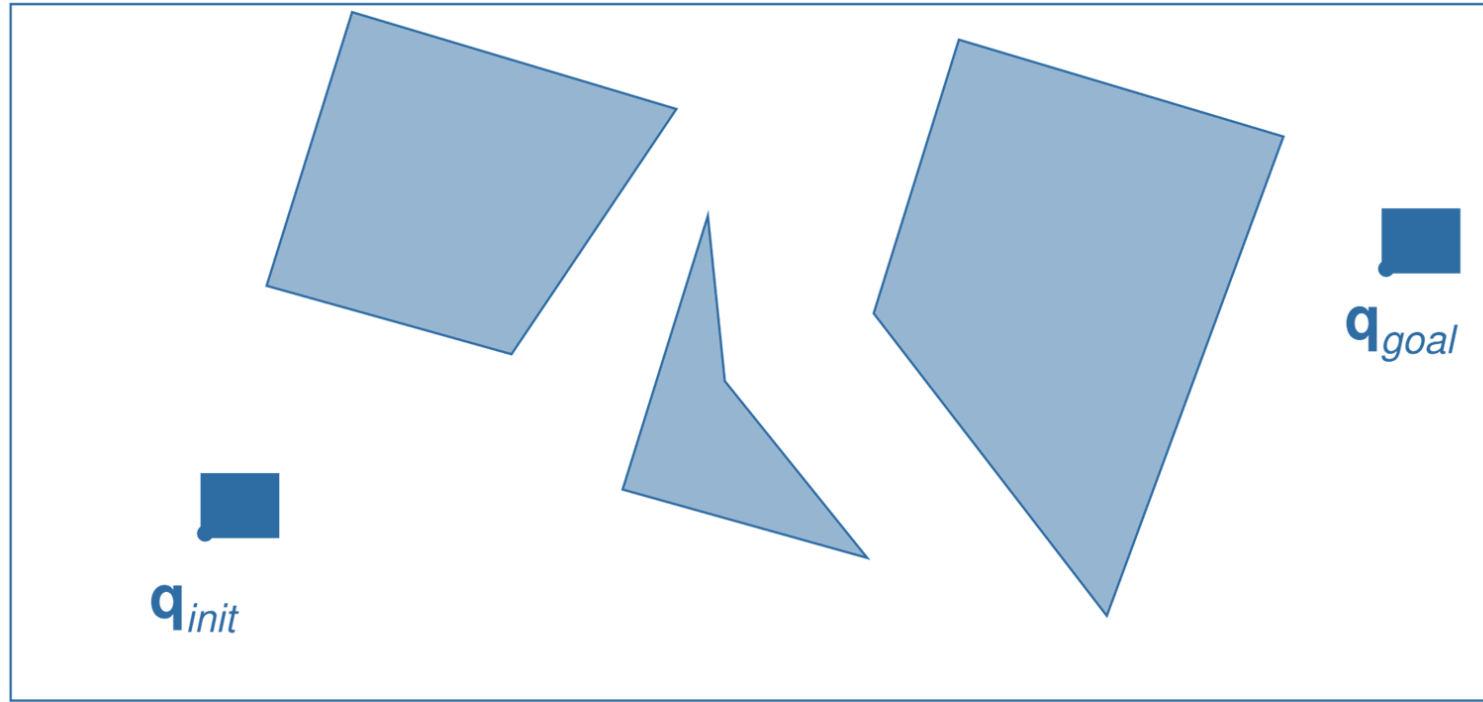
What if our subject is not a point?



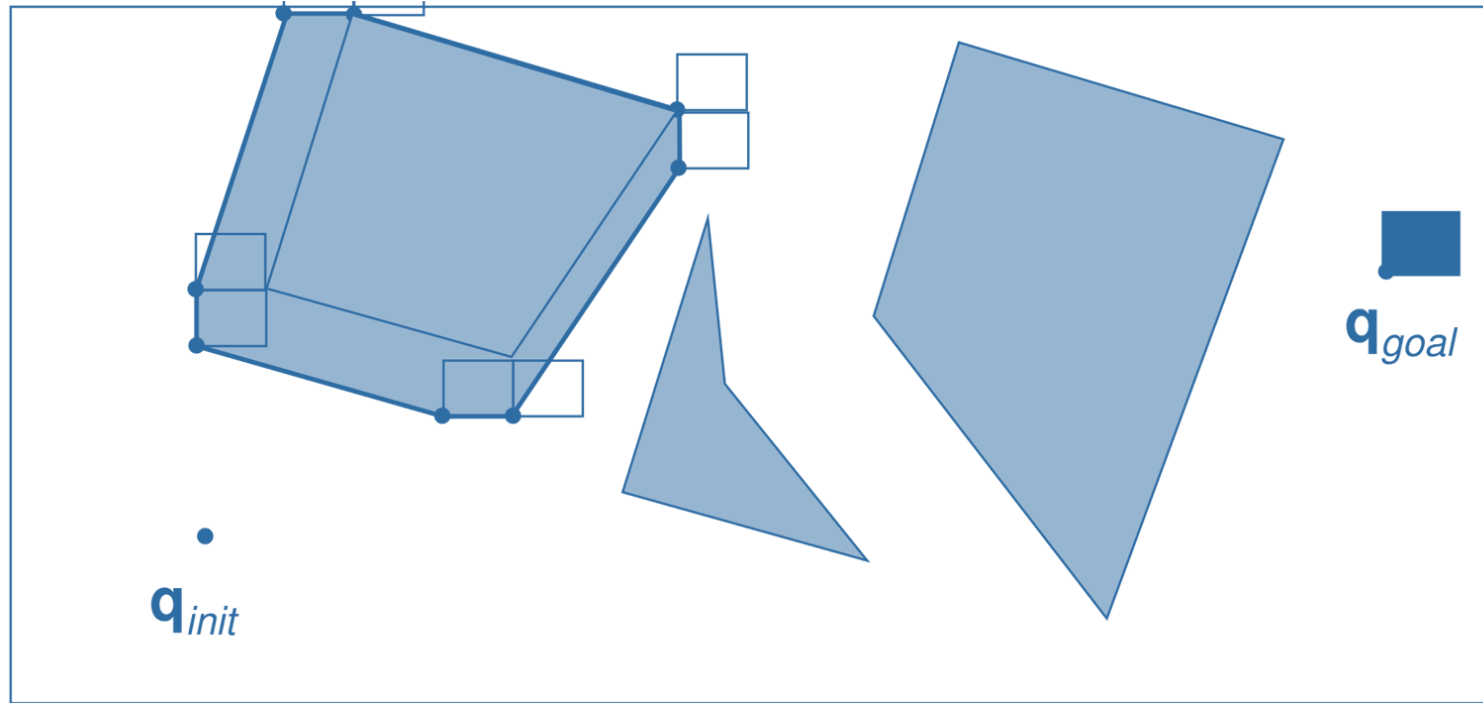
Towards motion planning



Rectangular subject that can only translate in 2D

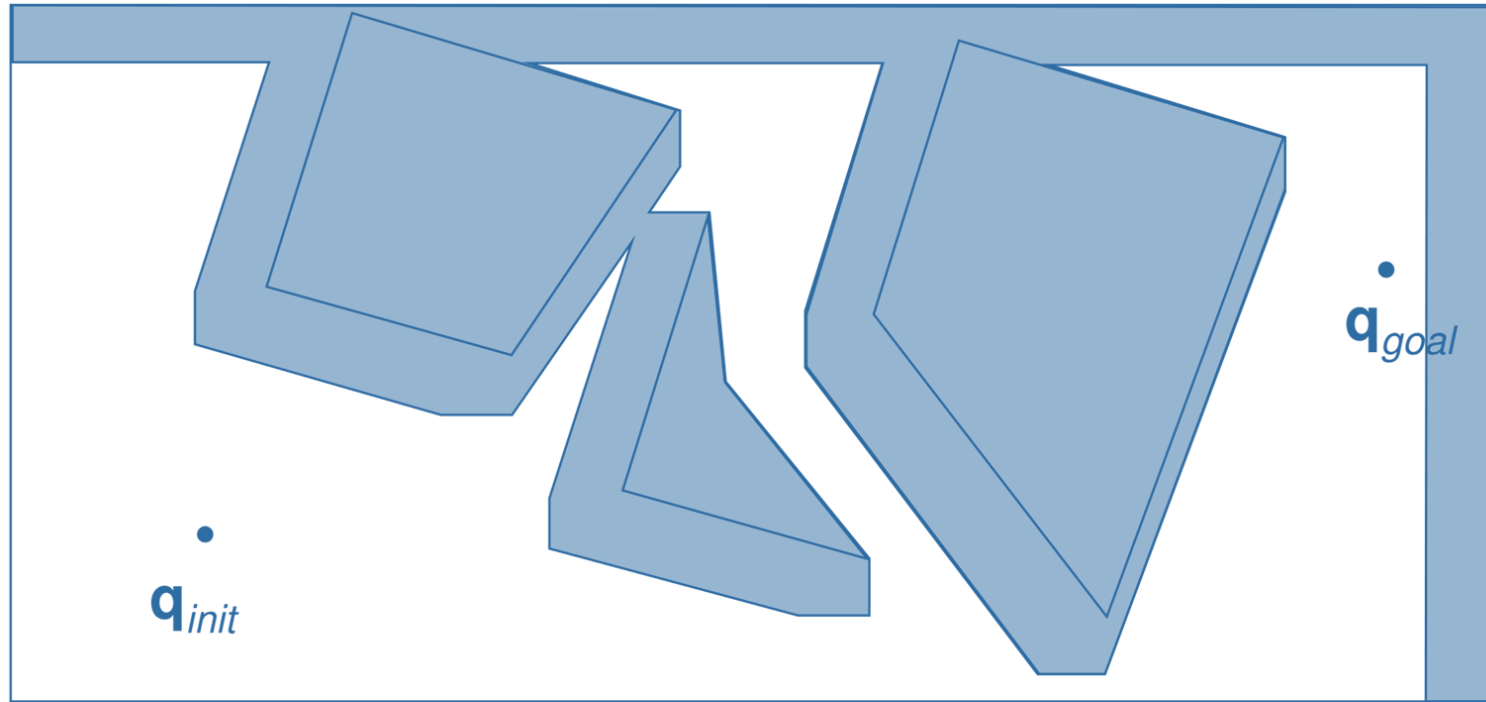


Rectangular subject that can only translate in 2D



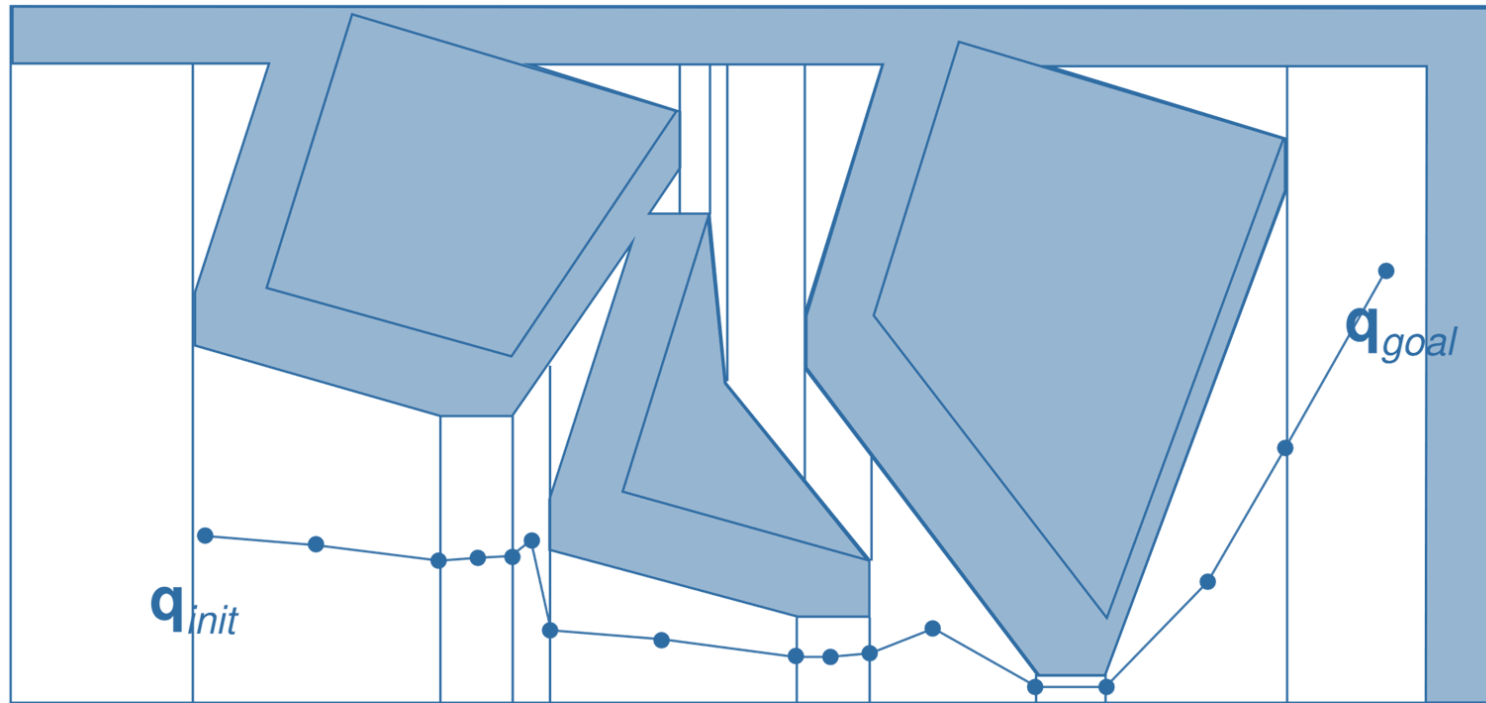
Rectangular subject that can only translate in 2D

Replace subject with a point,
and obstacles with Minkowski
difference
between
obstacles and
subject



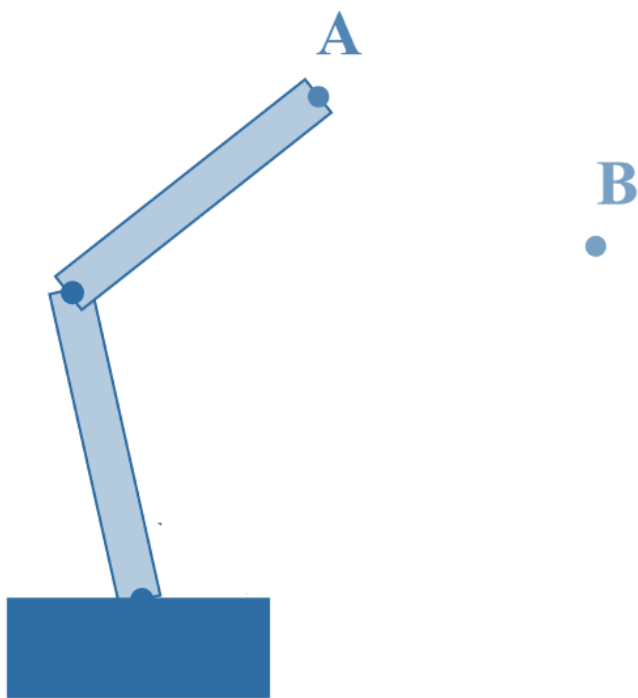
Rectangular subject that can only translate in 2D

Replace subject
with a point,
and obstacles
with Minkowski
difference
between
obstacles and
subject

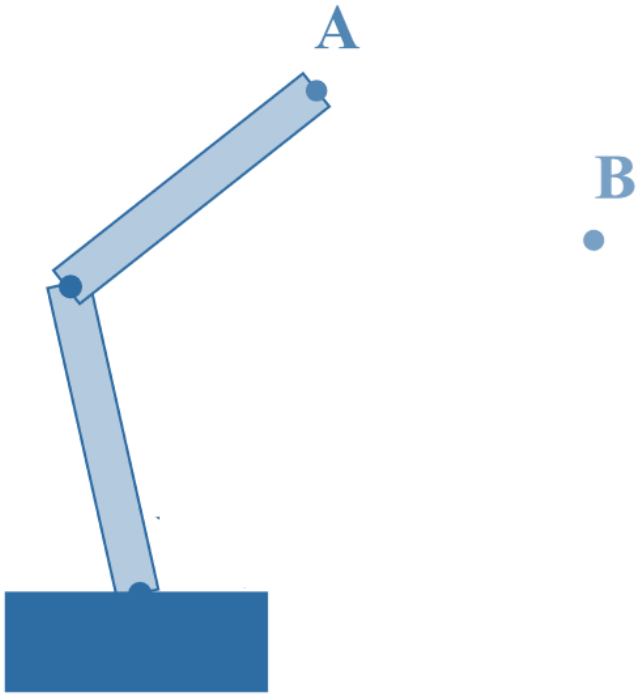


The motion planning problem as
a path planning problem

Motion planning problem



Motion planning problem



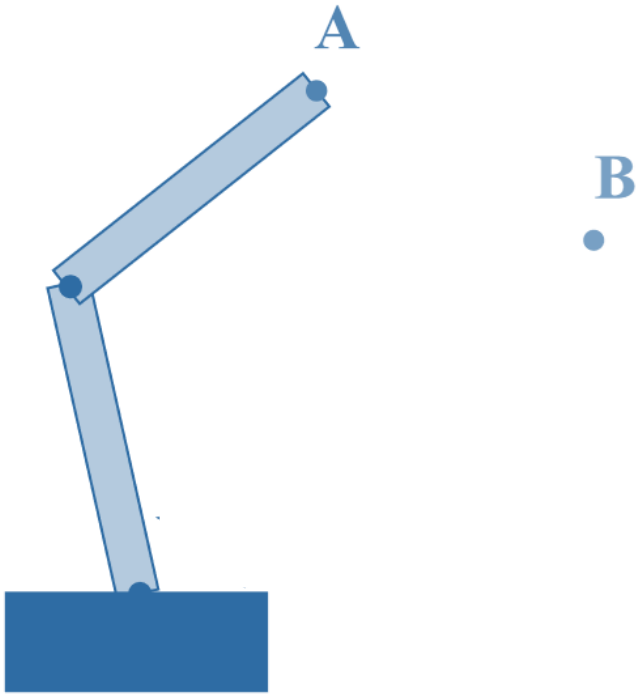
How to go from point A to point B?

What is “go”?

what are “points” A and B?

“what” needs to go (subject of the verb)?

Motion planning problem



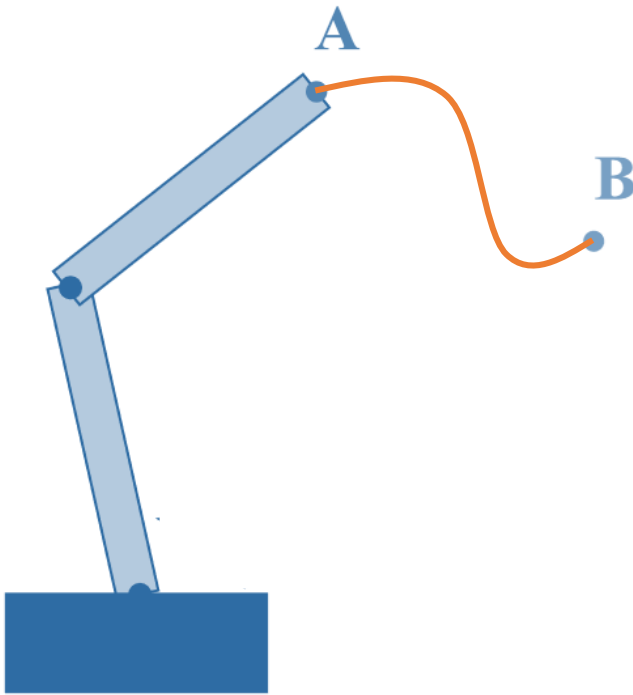
How to go from point A to point B?

What is “go”? -> move the whole robot ?

what are “points” A and B? -> points of the 3D space ?

“what” needs to go (subject of the verb)? -> point on the robot ?

Motion planning problem



How to go from point A to point B?

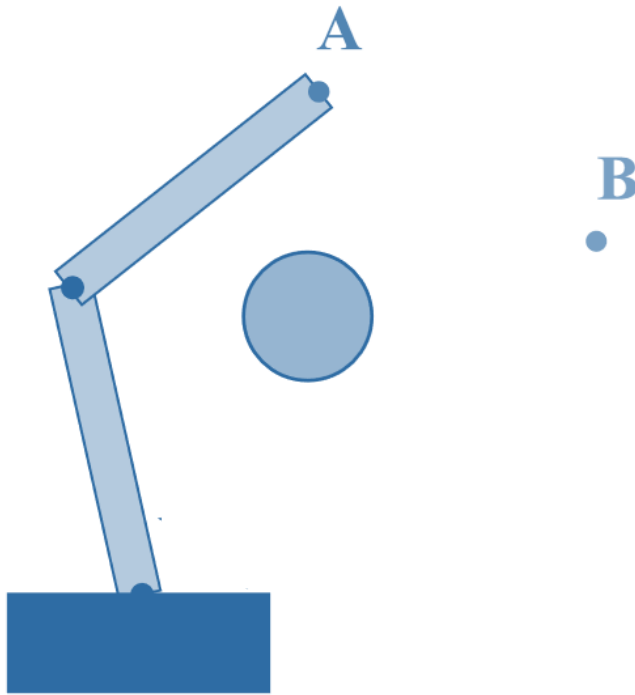
What is “go”? -> move the whole robot ?

what are “points” A and B? -> points of the 3D space ?

“what” needs to go (subject of the verb)? -> point on the robot ?

Motion planning problem

How to go from point A to point B?



What is “go”? -> move the whole robot ?

what are “points” A and B? -> points of the 3D space ?

“what” needs to go (subject of the verb)? -> point on the robot ?

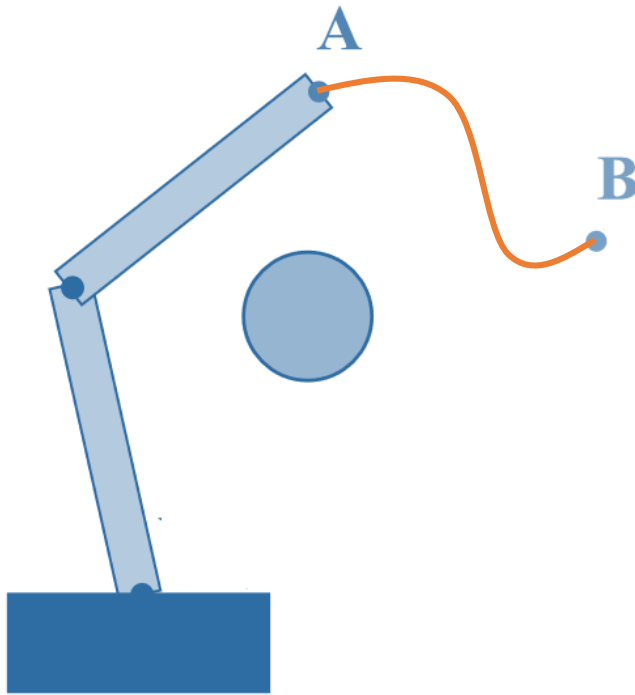
Motion planning problem

How to go from point A to point B?

What is “go”? -> move the whole robot ?

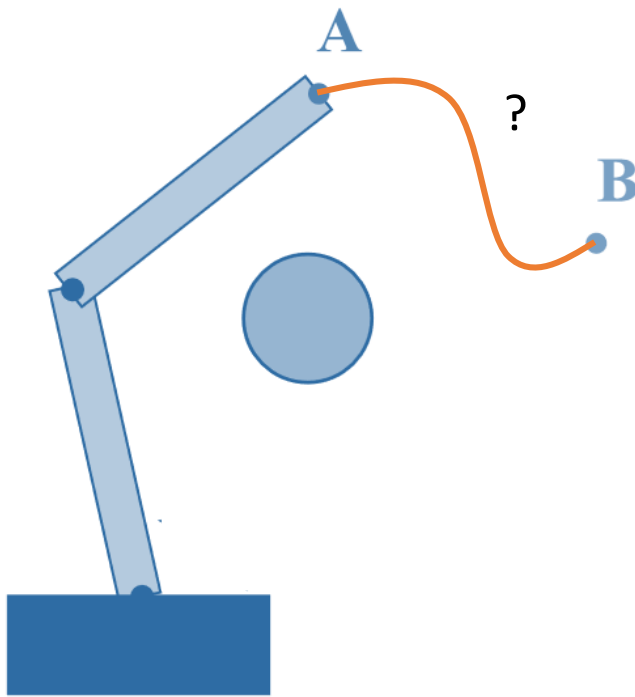
what are “points” A and B? -> points of the 3D space ?

“what” needs to go (subject of the verb)? -> point on the robot ?



Motion planning problem

How to go from point A to point B?



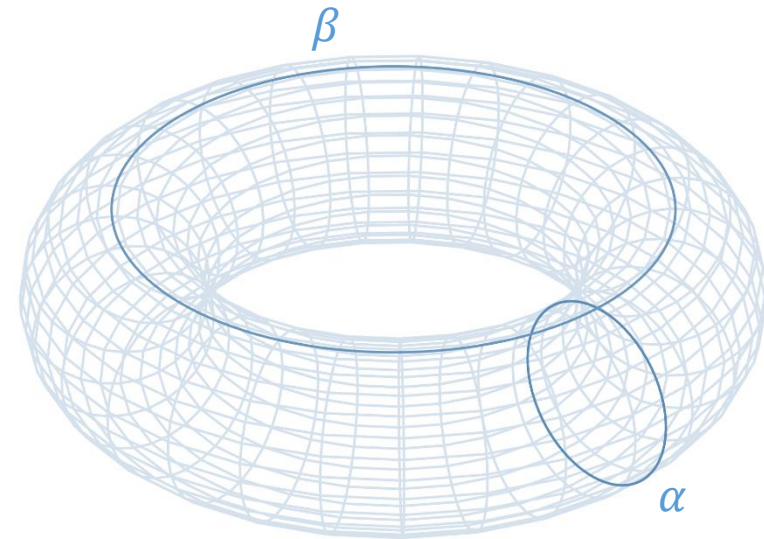
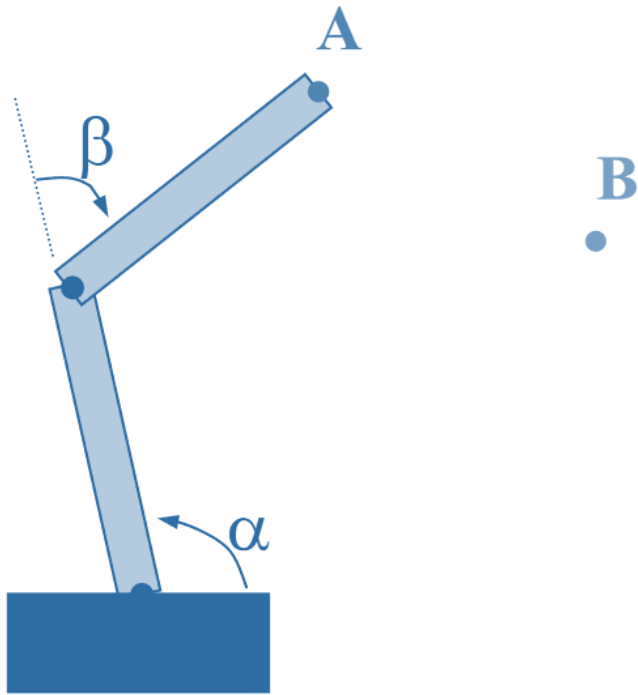
What is “go”? -> move the whole robot ?

what are “points” A and B? -> points of the 3D space ?

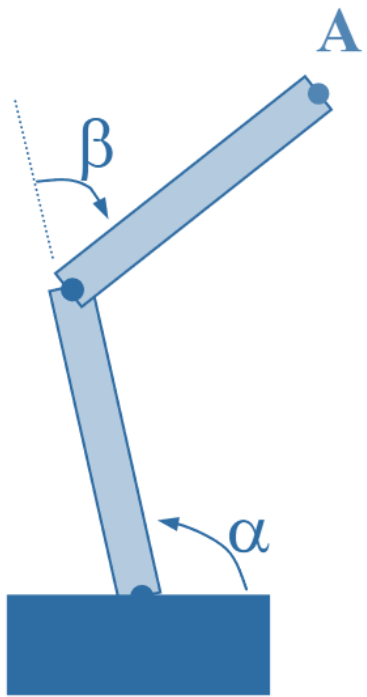
“what” needs to go (subject of the verb)? -> point on the robot ?

Definitely not a path planning problem from A to B

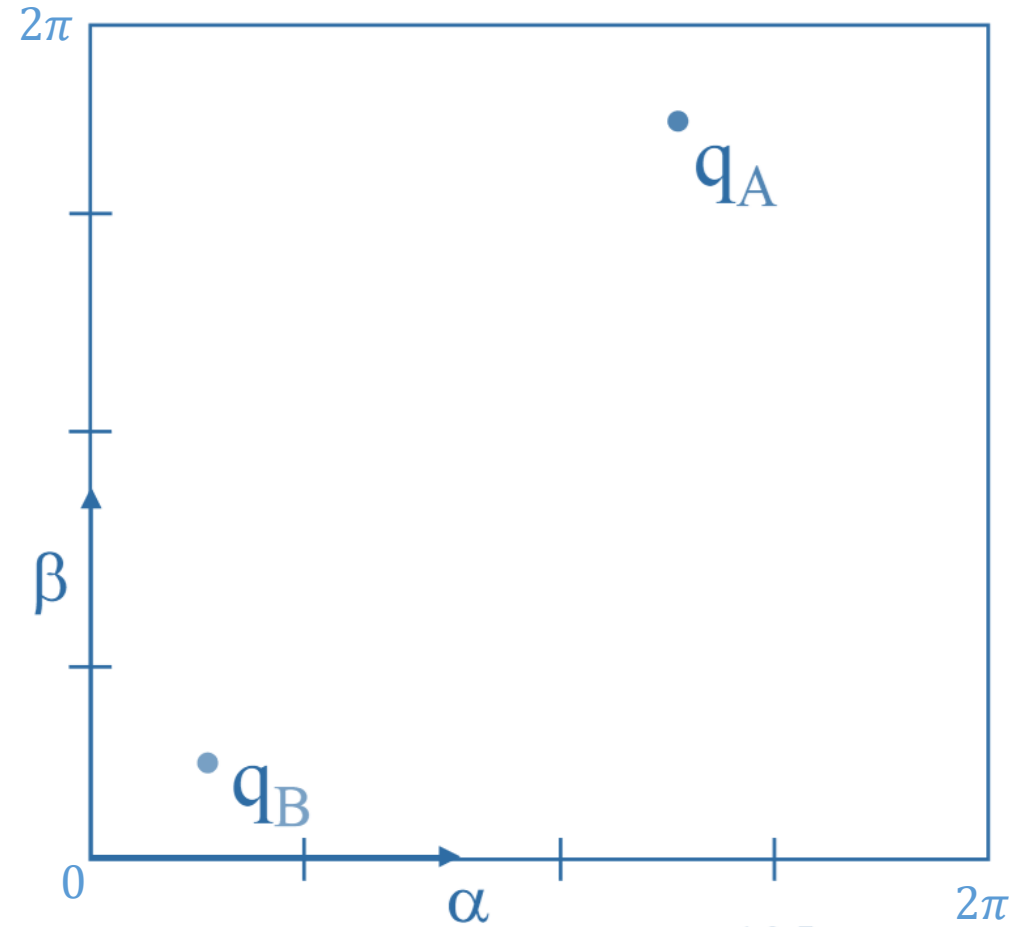
Motion planning problem



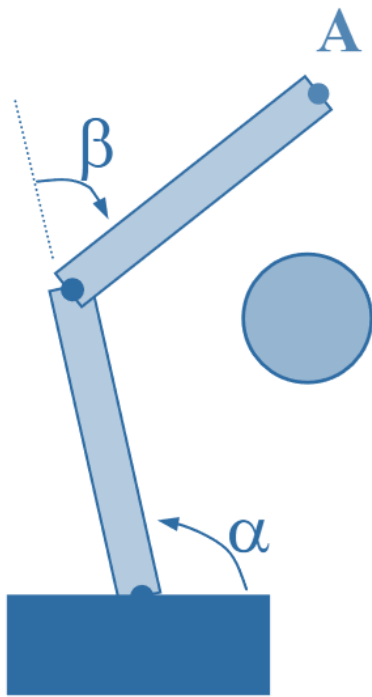
Motion planning problem



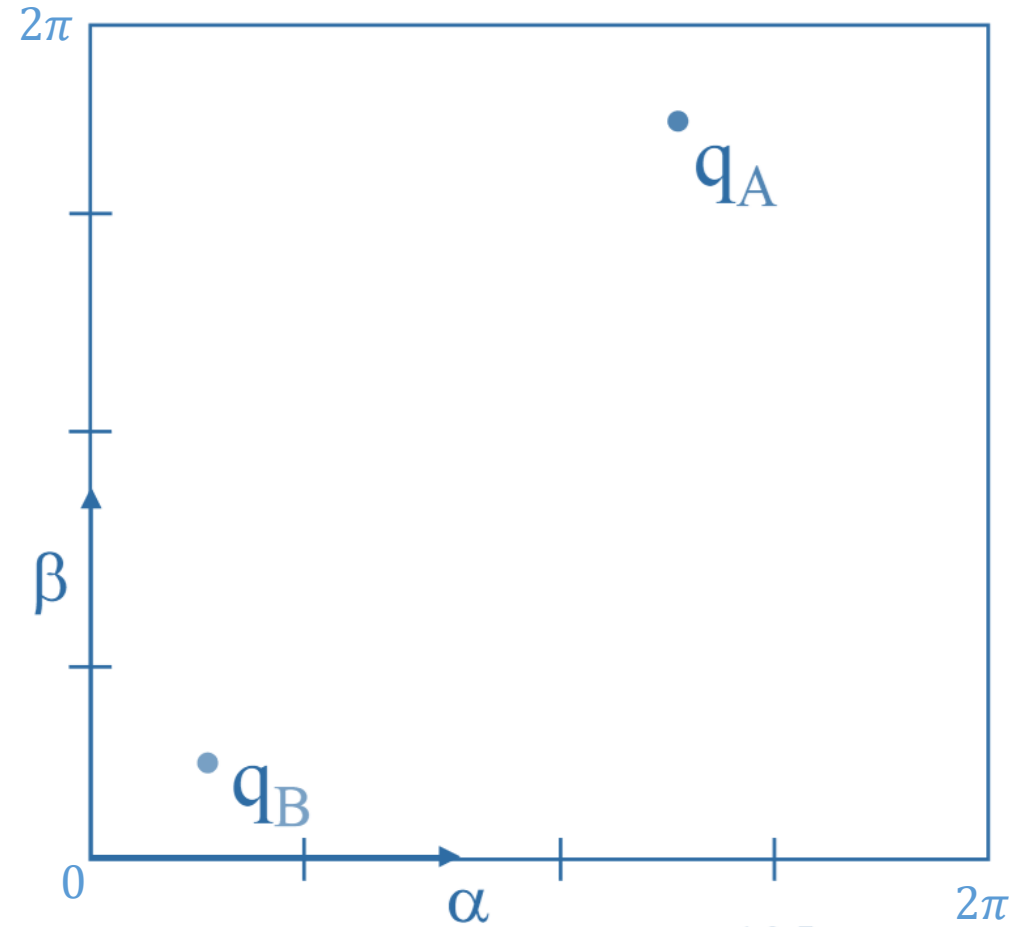
B

A single black dot representing point B.

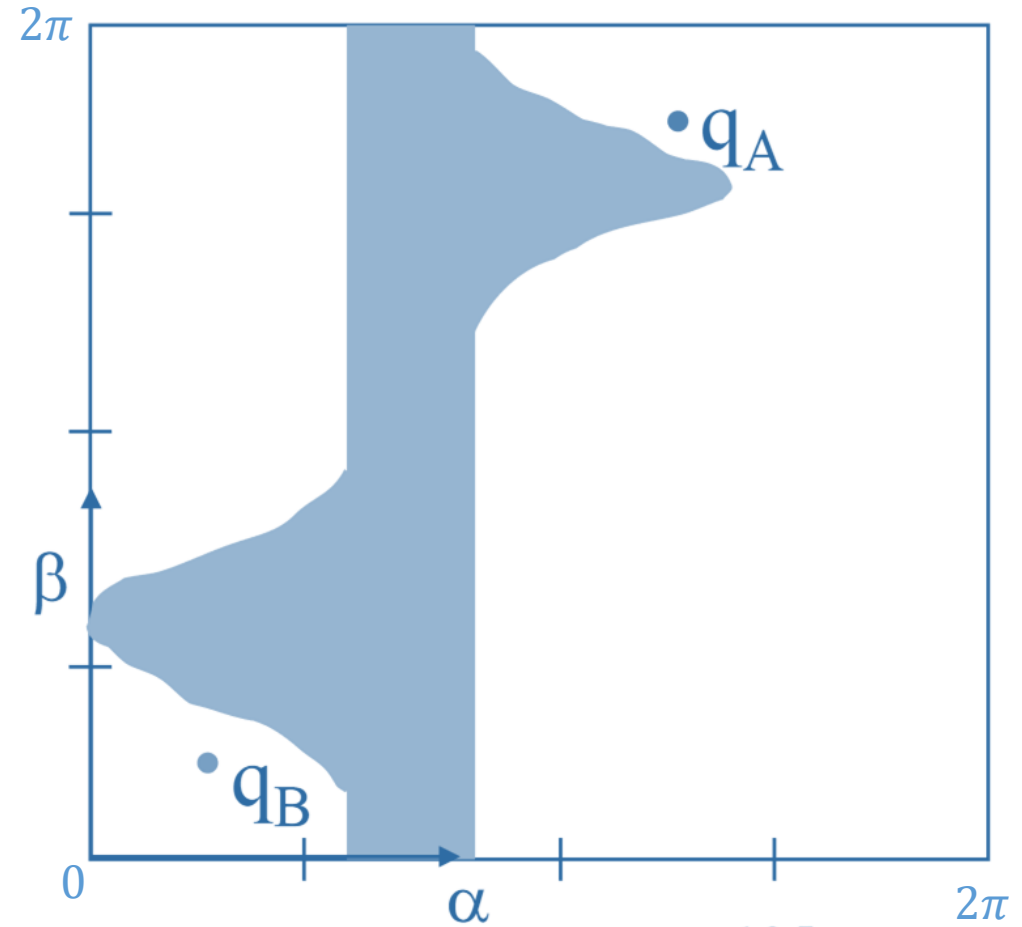
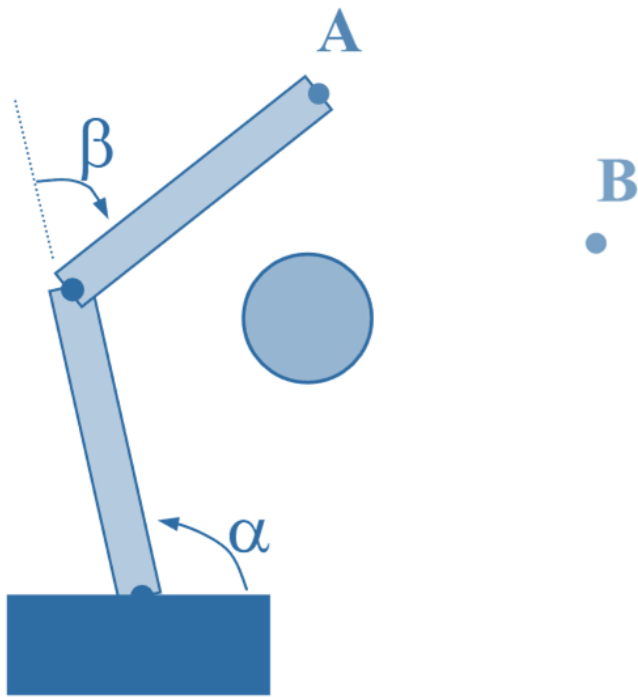
Motion planning problem



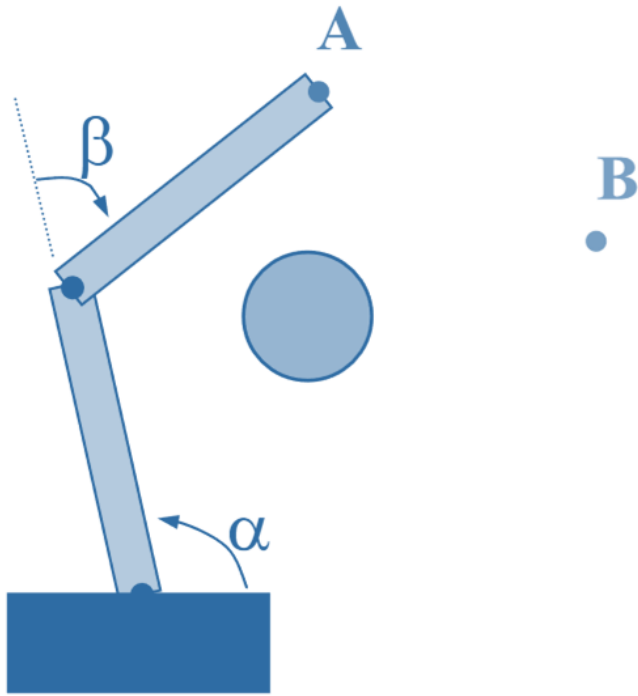
B

A small grey dot representing point B.

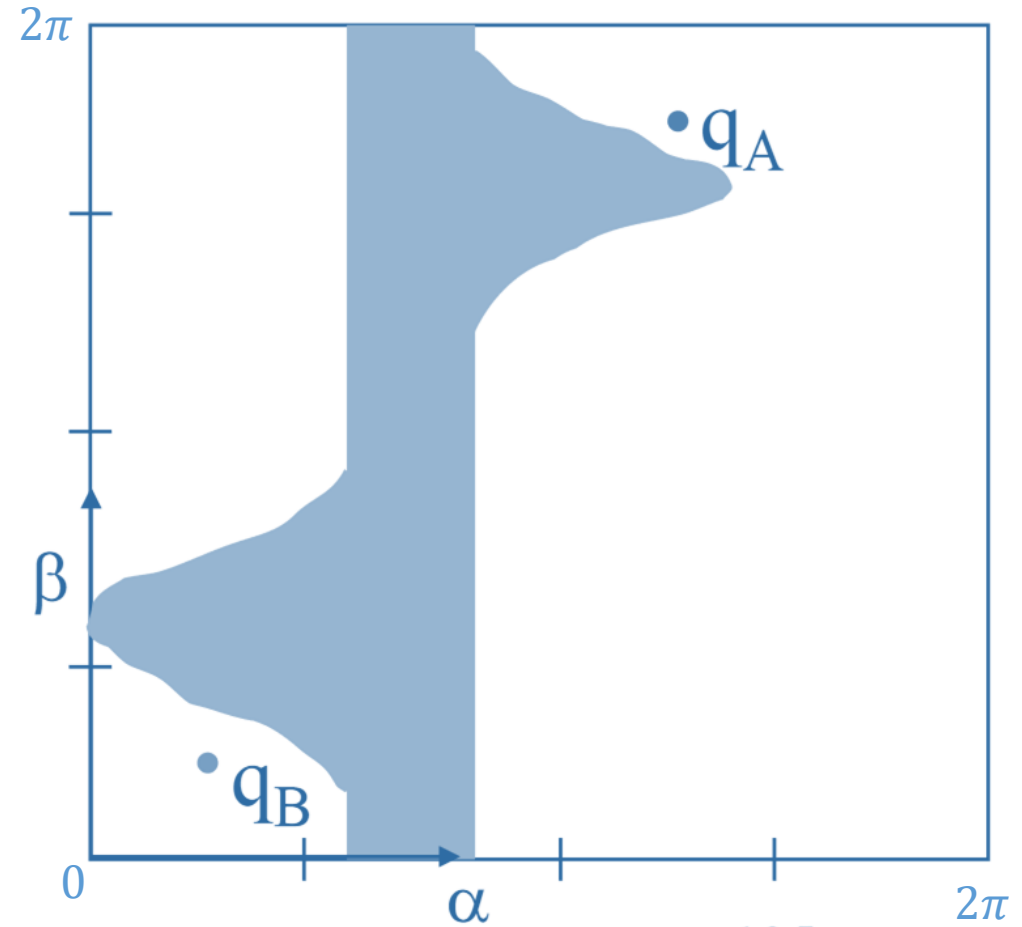
Motion planning problem



Motion planning problem

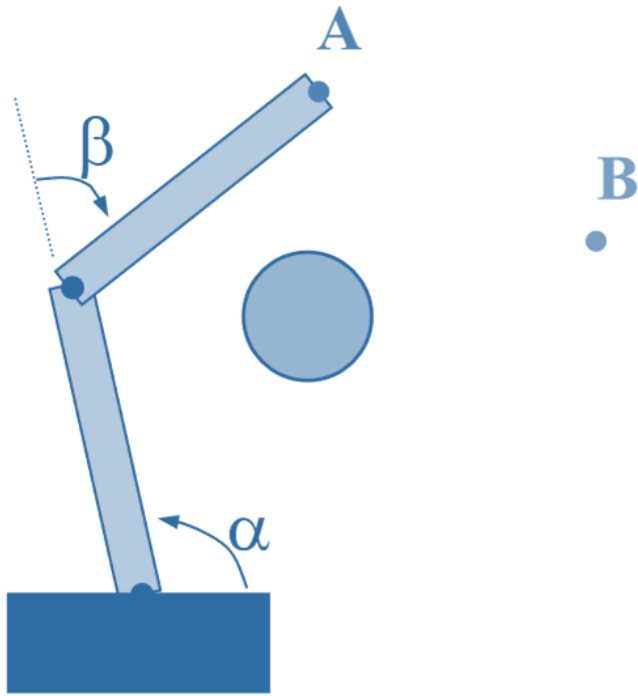


Motion planning problem

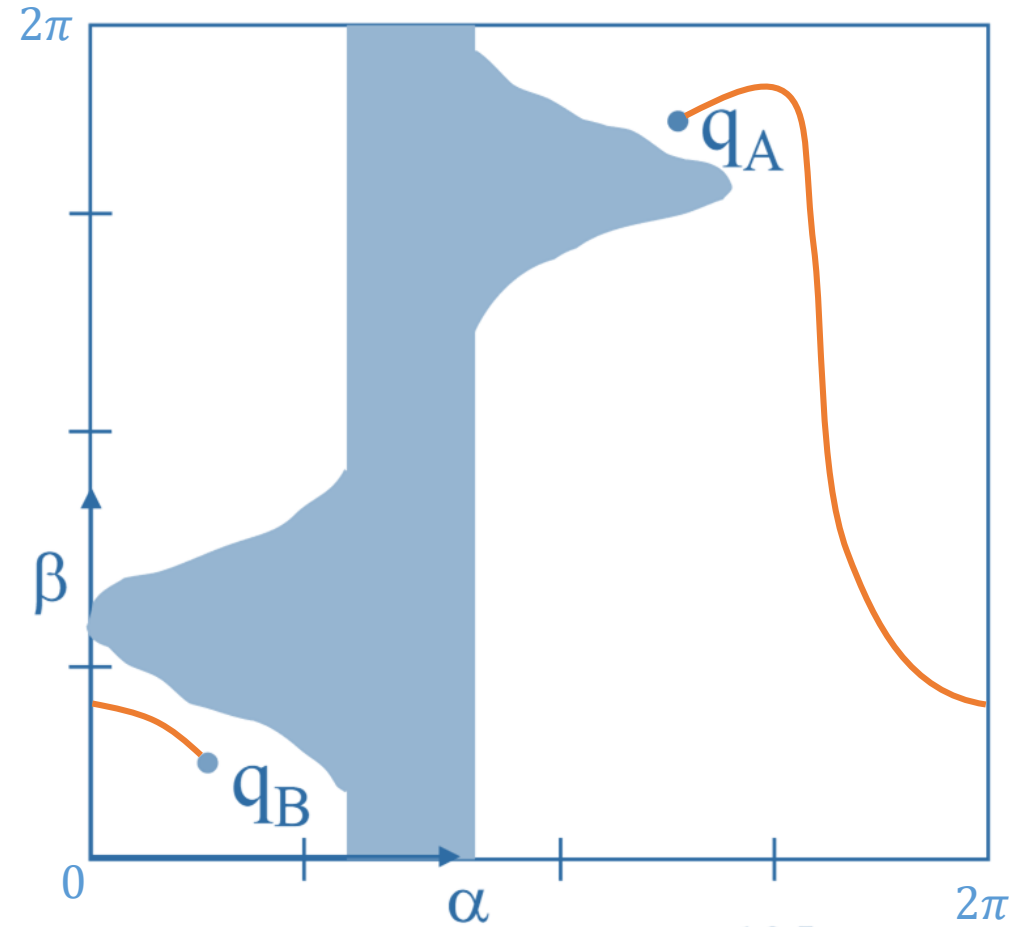


Path planning problem

Motion planning problem



Motion planning problem



Path planning problem

Configuration space \mathcal{C}

- point in \mathbb{R}^n : \mathbb{R}^n
- solid shape in 2D: $SE(2) = \mathbb{R}^2 \times SO(2) = \mathbb{R}^2 \times \mathbb{S}^1$
- revolute joint: \mathbb{S}^1 (circle)
- spherical joint: $SO(3) = \mathbb{RP}^3$ (real projective space)
= unit quaternion hemisphere with antipodal identification
- solid shape in 3D: $SE(3) = \mathbb{R}^3 \times SO(3)$
- n -joint arm: $\mathbb{T}^n = (\mathbb{S}^1)^n$ (torus)

Formulation of the motion planning problem (aka the piano mover problem)

- A physical world $\mathcal{W} = \mathbb{R}^2$ or \mathbb{R}^3
- A robot $\mathcal{A} \subset \mathcal{W}$ as a collection of links with joint configuration $q \in \mathcal{C}$, where \mathcal{C} is a n -dimensional manifold
- a mapping $g: \mathcal{C} \rightarrow 2^{\mathcal{W}}$ (forward kinematics mapping) that puts the robot in a given configuration, for simplicity we denote $g(q)$ as $\mathcal{A}(q)$
- A compact obstacle region $\mathcal{O} \subset \mathcal{W}$
- The motion planning problem for \mathcal{A} in \mathcal{W} amounts to a path planning problem in $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$ where $\mathcal{C}_{obs} = \{q \mid \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}$

Computational algebraic geometry solution to the problem

- *Lemma:* If we suppose that \mathcal{A} and \mathcal{O} are defined as semi-algebraic subsets of \mathcal{W} (ie defined using finite unions and intersections of regions delimited by polynomial equations with rational coefficients, including polygons/polyhedra, circles/spheres, ellipses/ellipsoids, etc), then it is possible to demonstrate that \mathcal{C}_{obs} and \mathcal{C}_{free} are also semi-algebraic subsets of \mathbb{R}^n for all types of robots with configuration spaces defined as finite Cartesian products of the previously listed manifolds.
- Example: polyhedral robots in polyhedral worlds

Computational algebraic geometry solution to the problem

- Semi-algebraic regions are defined as Tarski sentences (logical predicates on polynomial expressions on manifold coordinates with quantifiers and free variables).
- The cylindrical algebraic decomposition (aka Collins decomposition) used for quantifier elimination in Tarski sentences (and for deciding satisfiability of Tarski sentences) yields in fact a cell decomposition of \mathcal{C}_{free} (similar to the vertical cell decomposition in the 2D polygonal case)
- Hence it solves the motion planning problem (Schwartz and Sharir, 1990)
- Complexity doubly exponential in the dimension n

So, are path planning algorithms applicable to the motion planning problem?

- Problem 1: many of the presented path planning algorithms make assumptions on the nature of the obstacle region (e.g. polygons). Eventhough we can make such assumptions on \mathcal{O} , it is difficult to say anything about the shape of \mathcal{C}_{obs}
- Problem 2: many of the presented path planning algorithms require explicit computation of the obstacle region. Again, it is difficult to compute explicitly \mathcal{C}_{obs}
- Problem 3: supposing 1 and 2 are solved, many of the presented path planning algorithms don't scale well beyond 2 or 3 dimensions (typical robot arm has 6 dof)

Sampling-based approaches

Probabilistic roadmap (PRM)

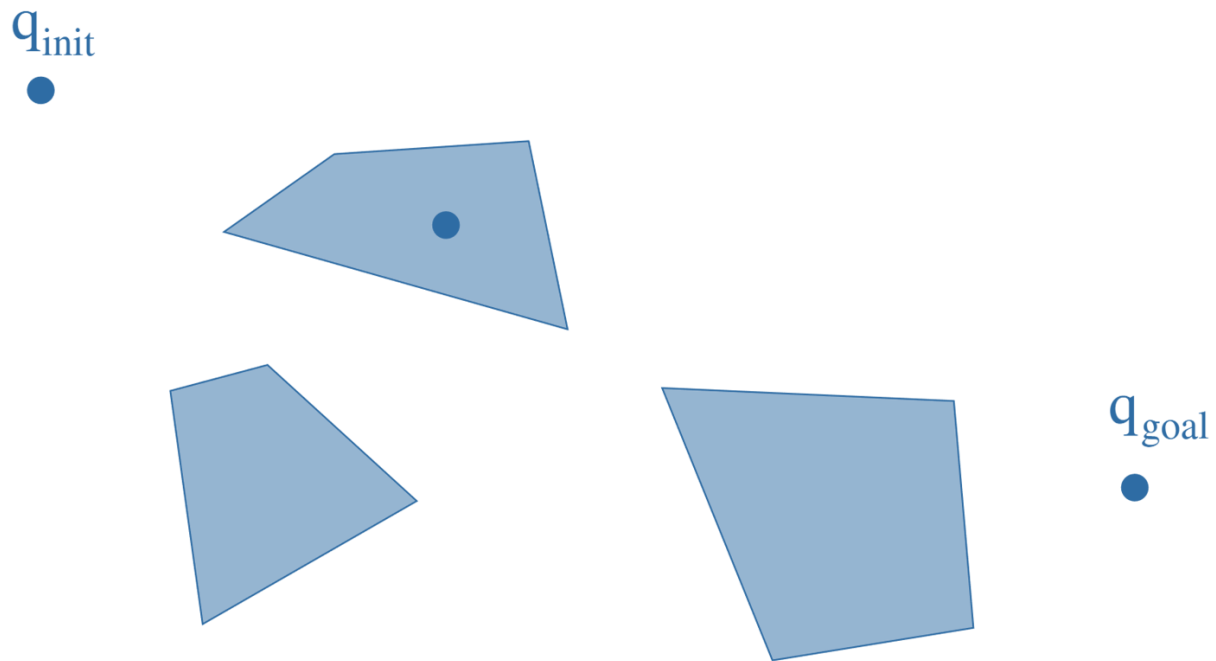
q_{init}



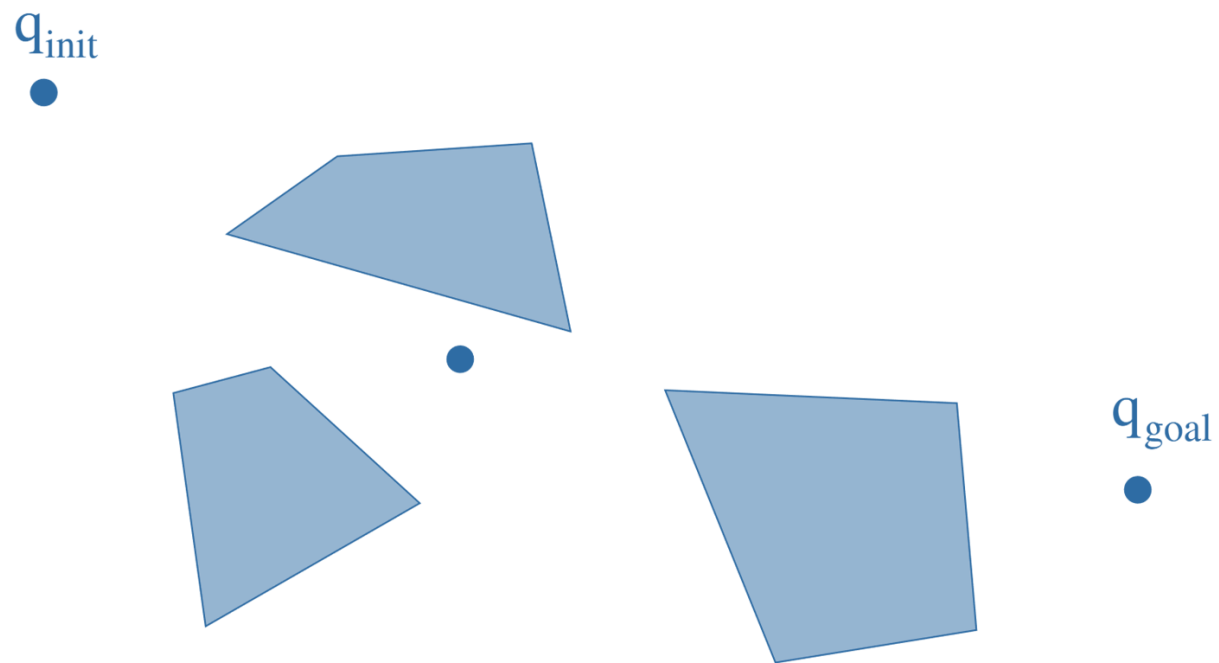
The diagram illustrates a 2D environment for a Probabilistic Roadmap (PRM) algorithm. It features a white background with three light blue polygonal obstacles. The first obstacle is a quadrilateral in the upper-middle section. The second is a triangle in the lower-left section. The third is a larger quadrilateral in the lower-right section. A dark blue dot labeled q_{init} is located in the upper-left area, and another dark blue dot labeled q_{goal} is located in the middle-right area, to the right of the third obstacle.

q_{goal}

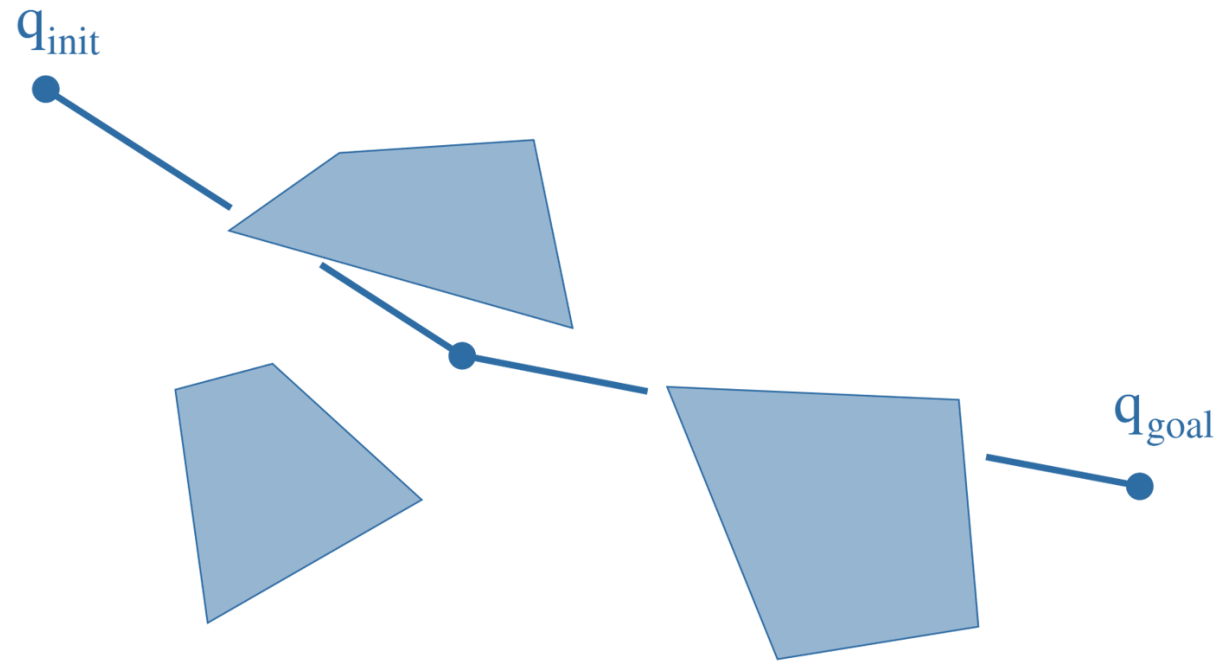
Probabilistic roadmap (PRM)



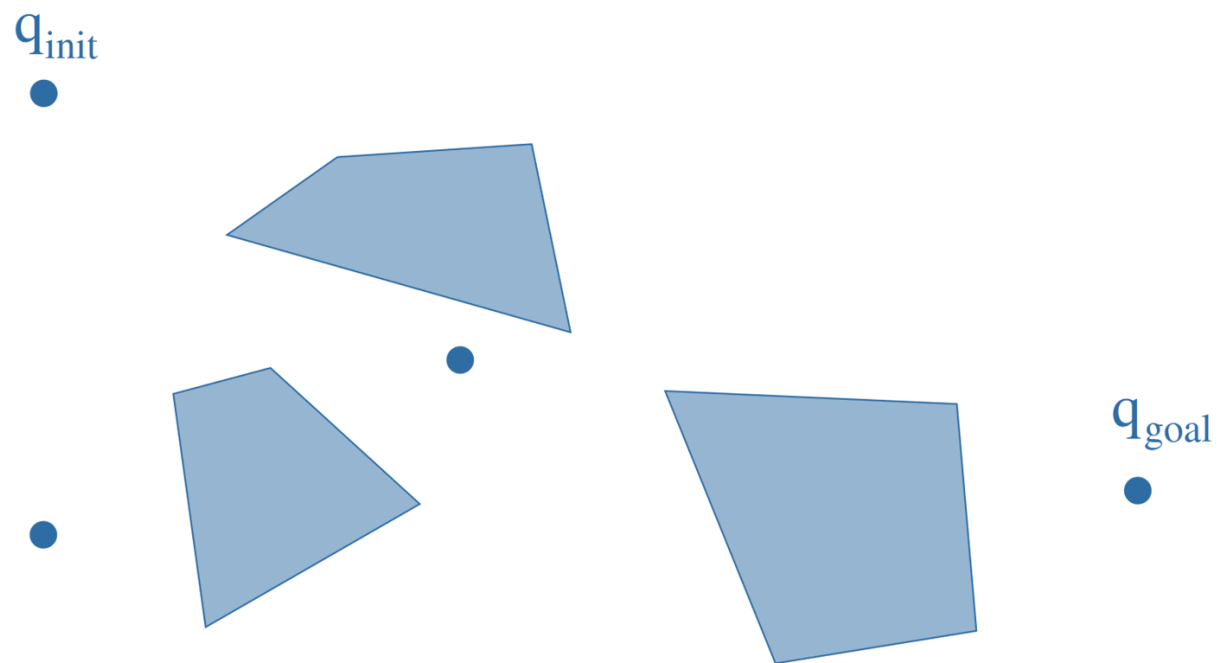
Probabilistic roadmap (PRM)



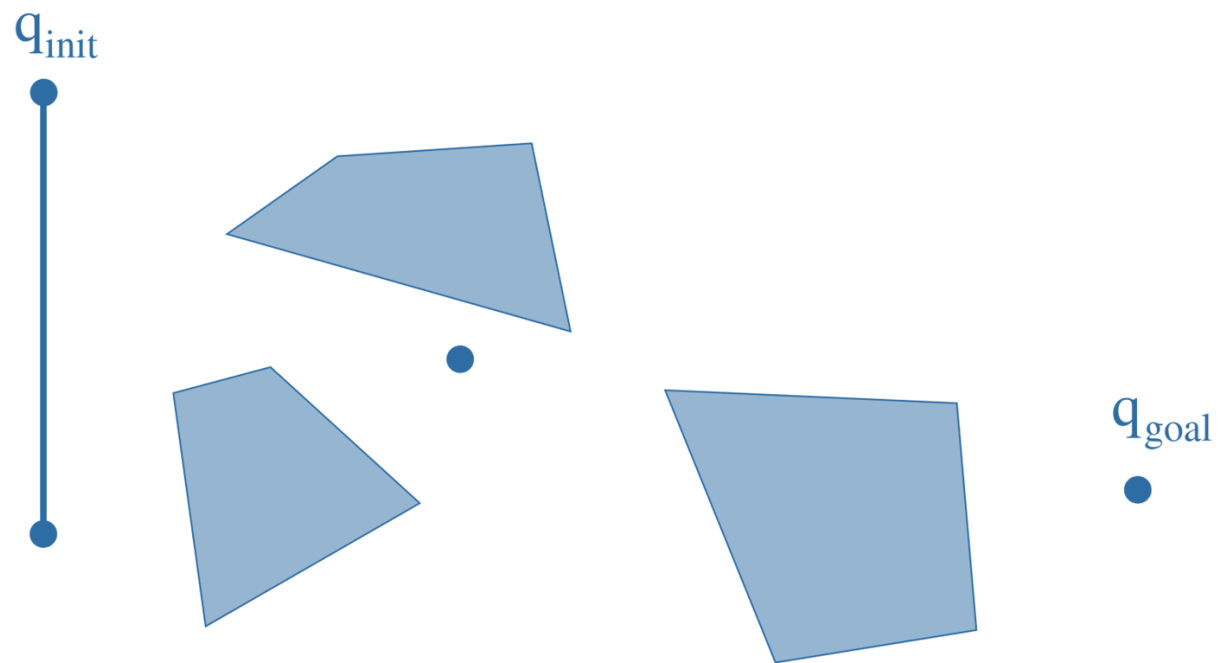
Probabilistic roadmap (PRM)



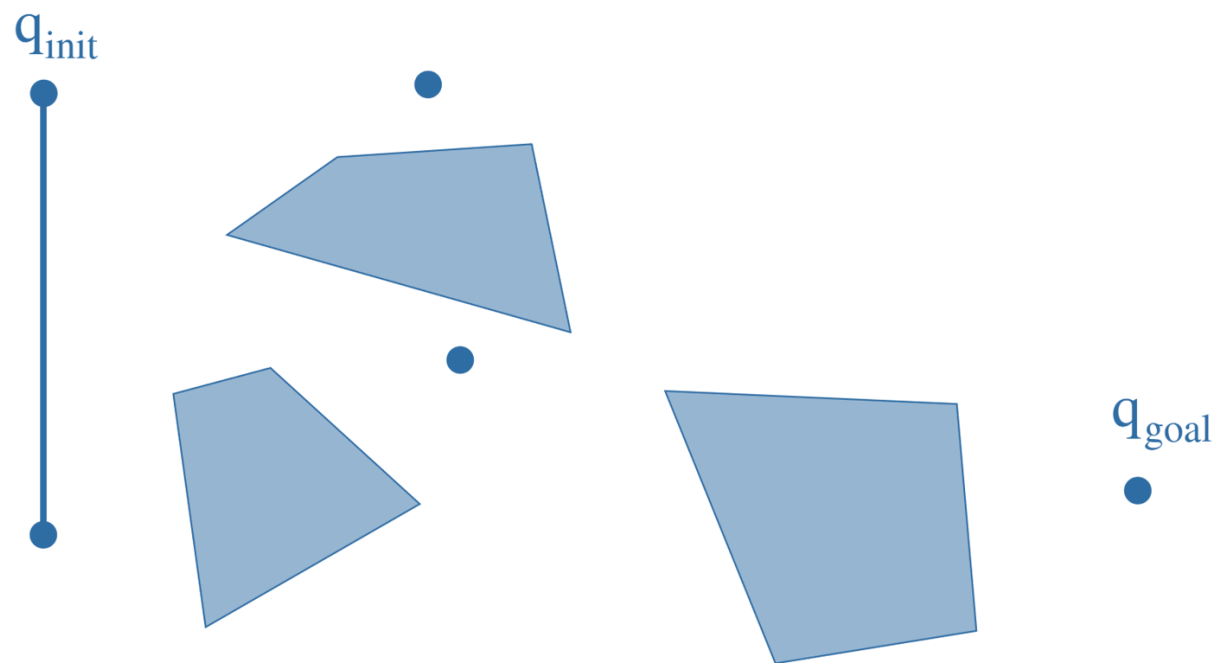
Probabilistic roadmap (PRM)



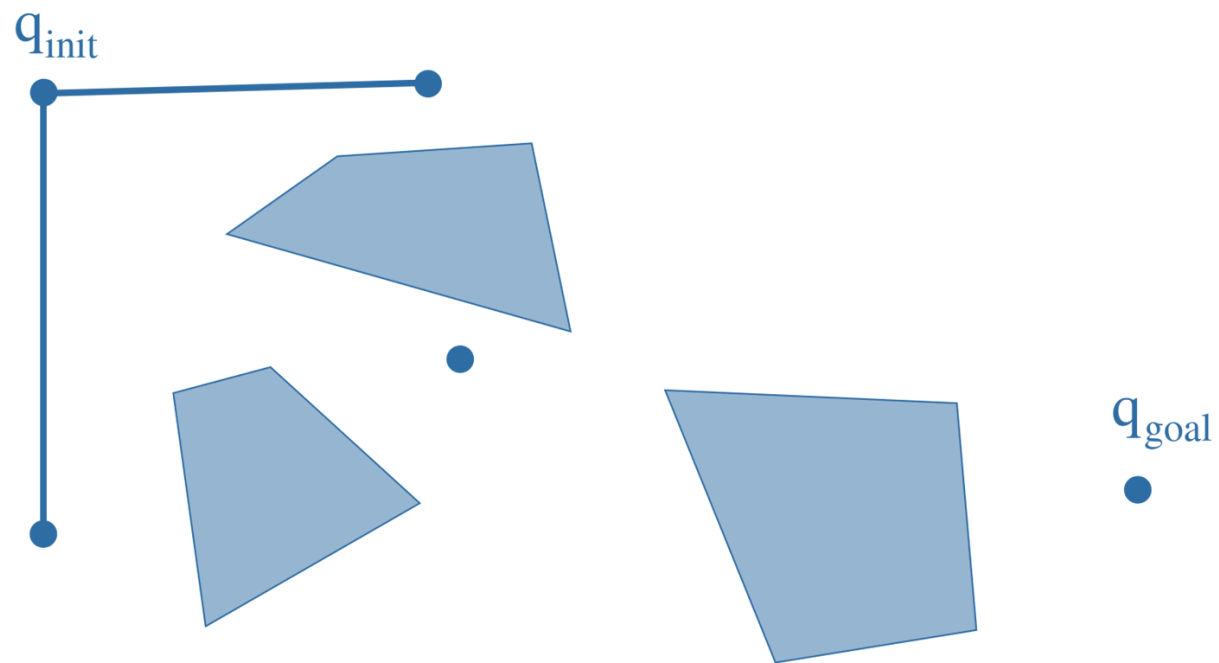
Probabilistic roadmap (PRM)



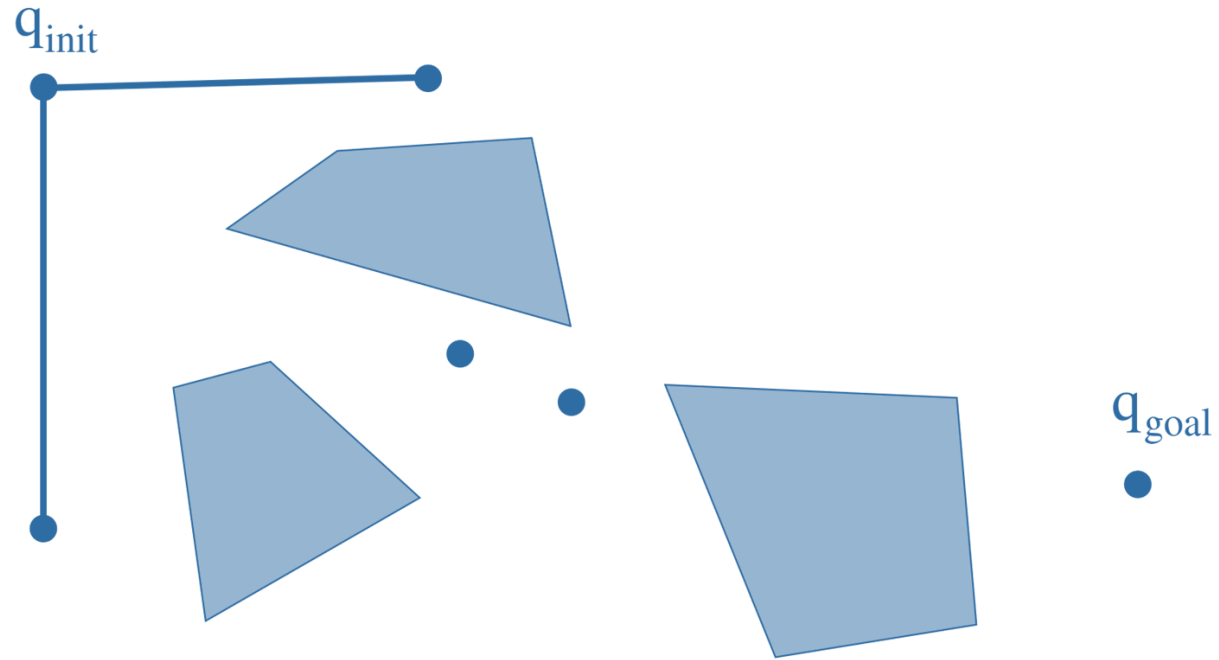
Probabilistic roadmap (PRM)



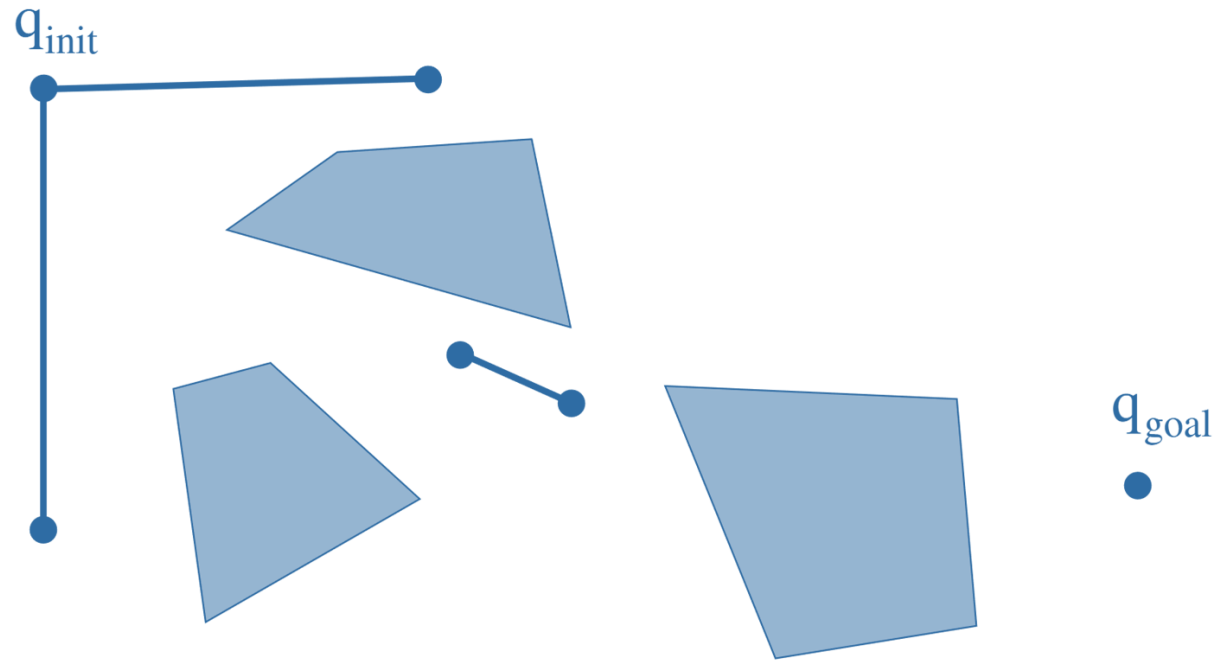
Probabilistic roadmap (PRM)



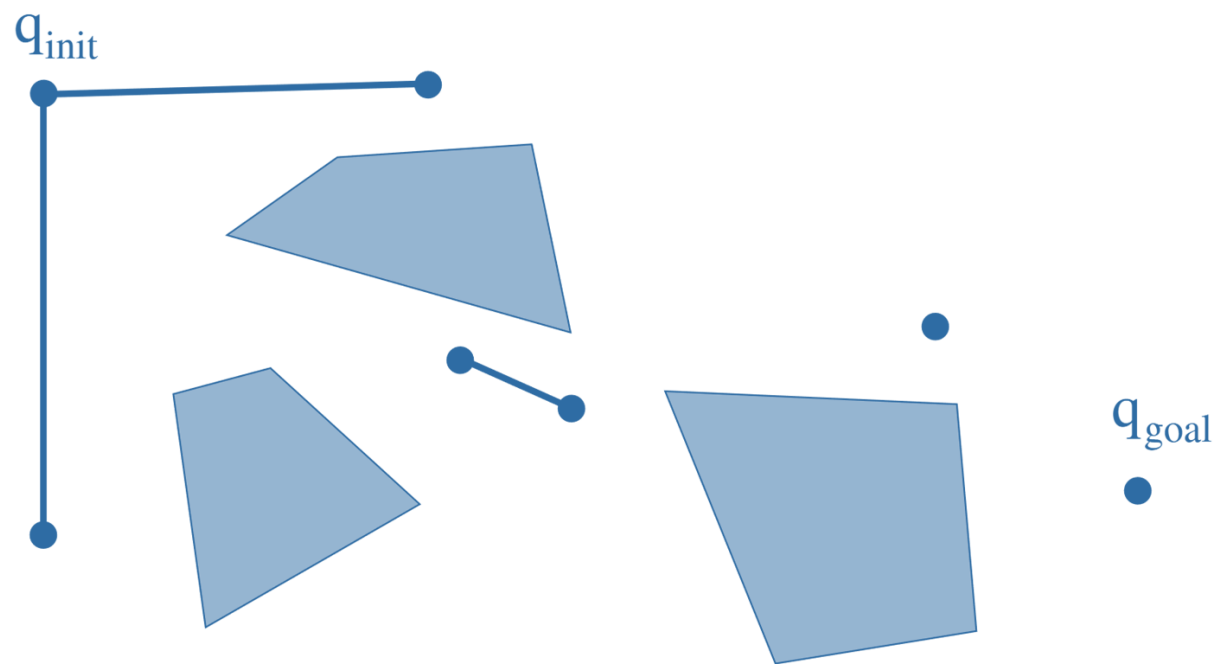
Probabilistic roadmap (PRM)



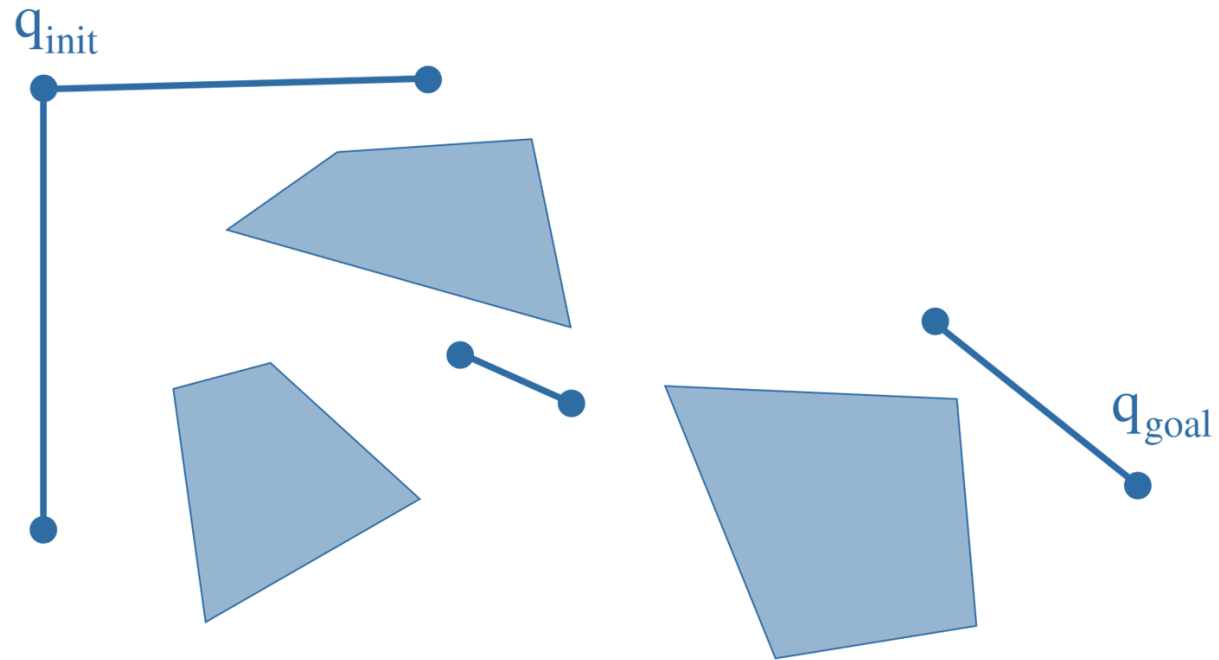
Probabilistic roadmap (PRM)



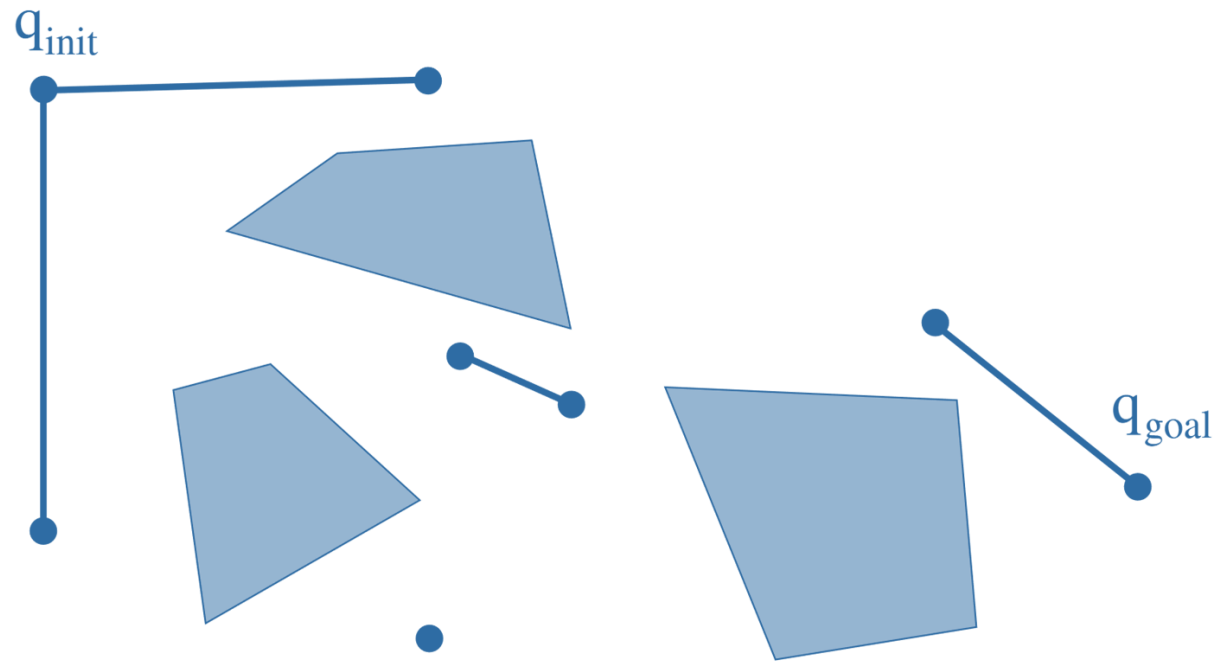
Probabilistic roadmap (PRM)



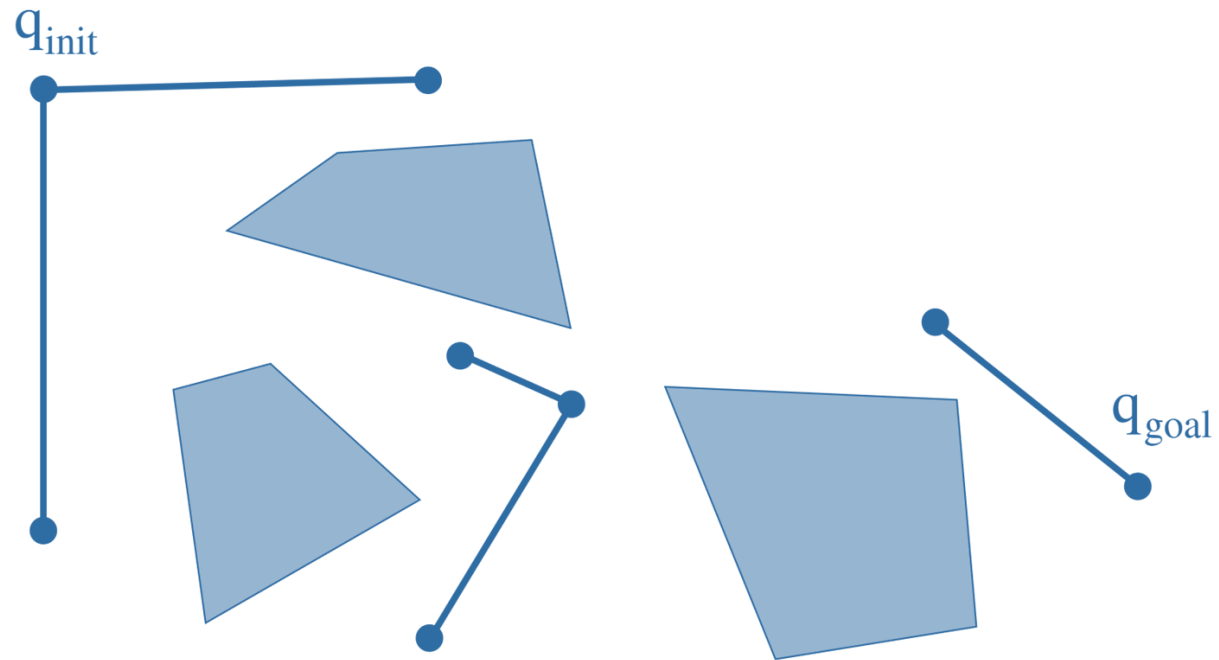
Probabilistic roadmap (PRM)



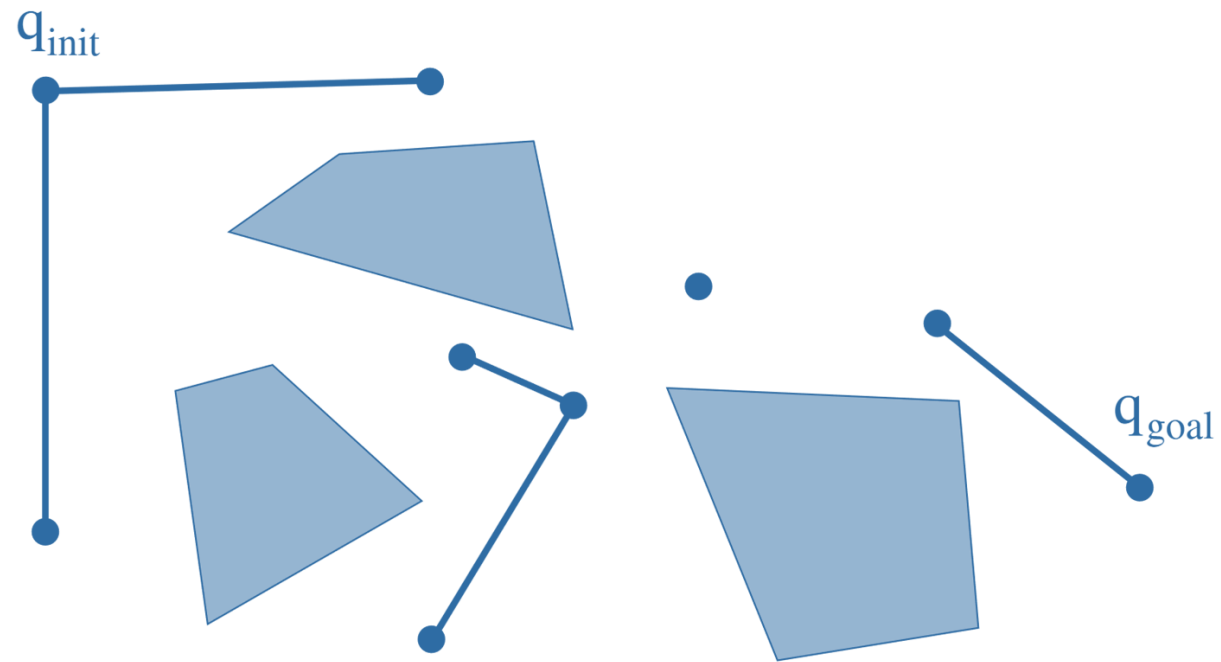
Probabilistic roadmap (PRM)



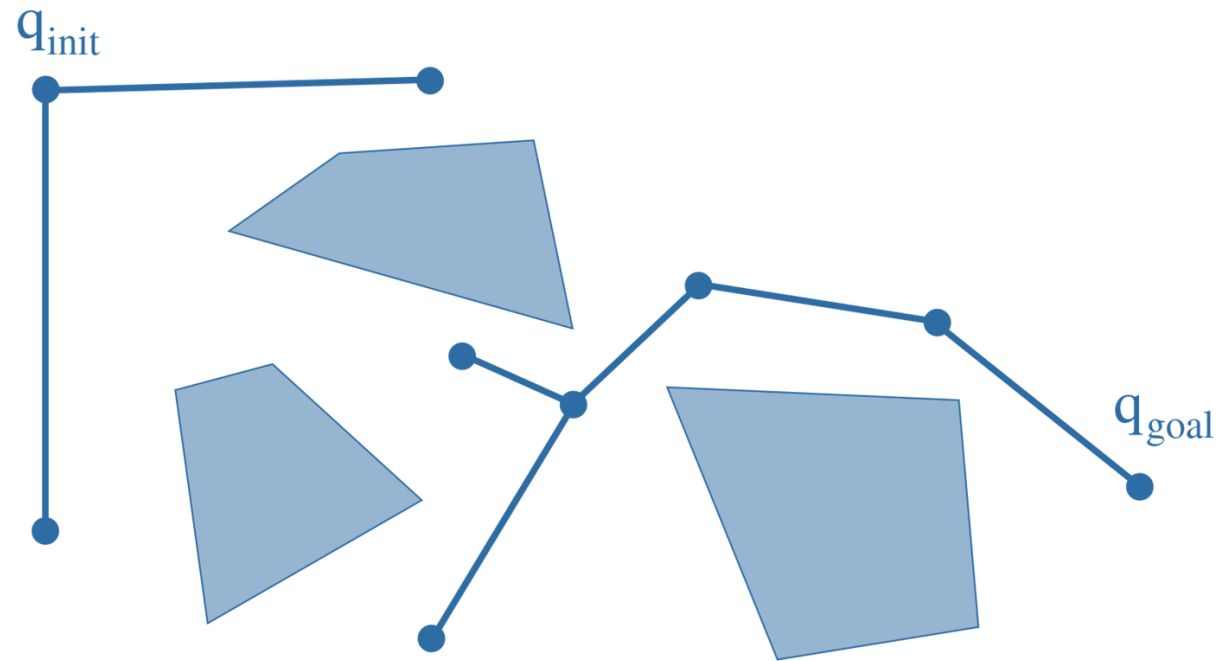
Probabilistic roadmap (PRM)



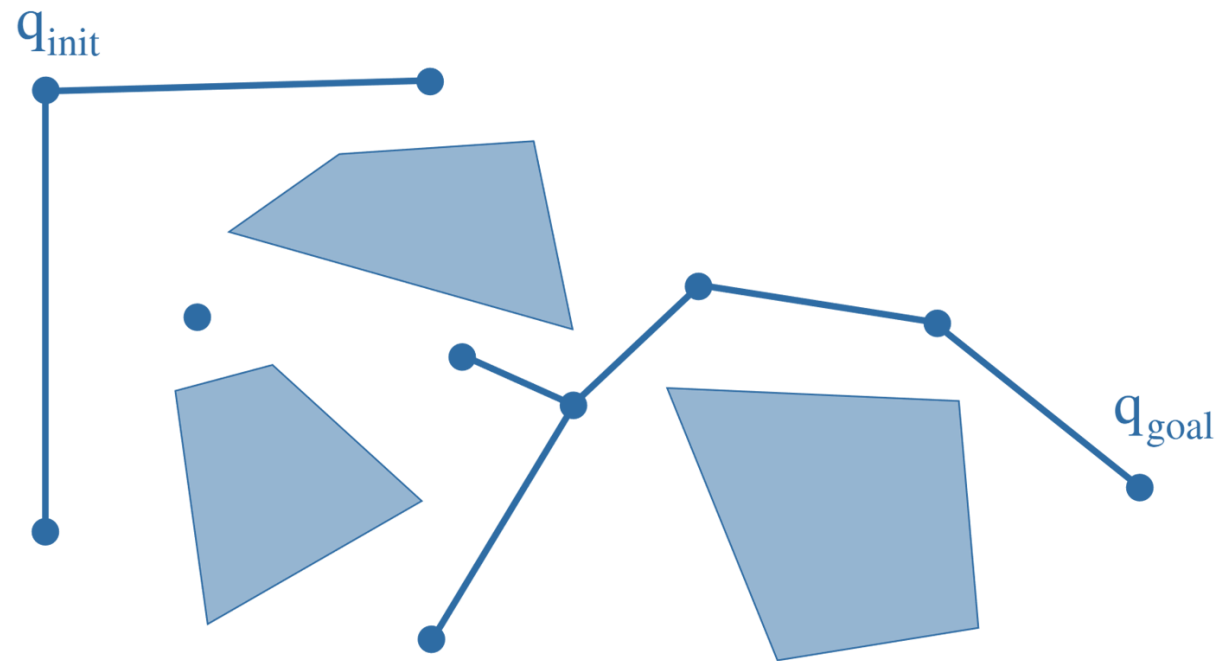
Probabilistic roadmap (PRM)



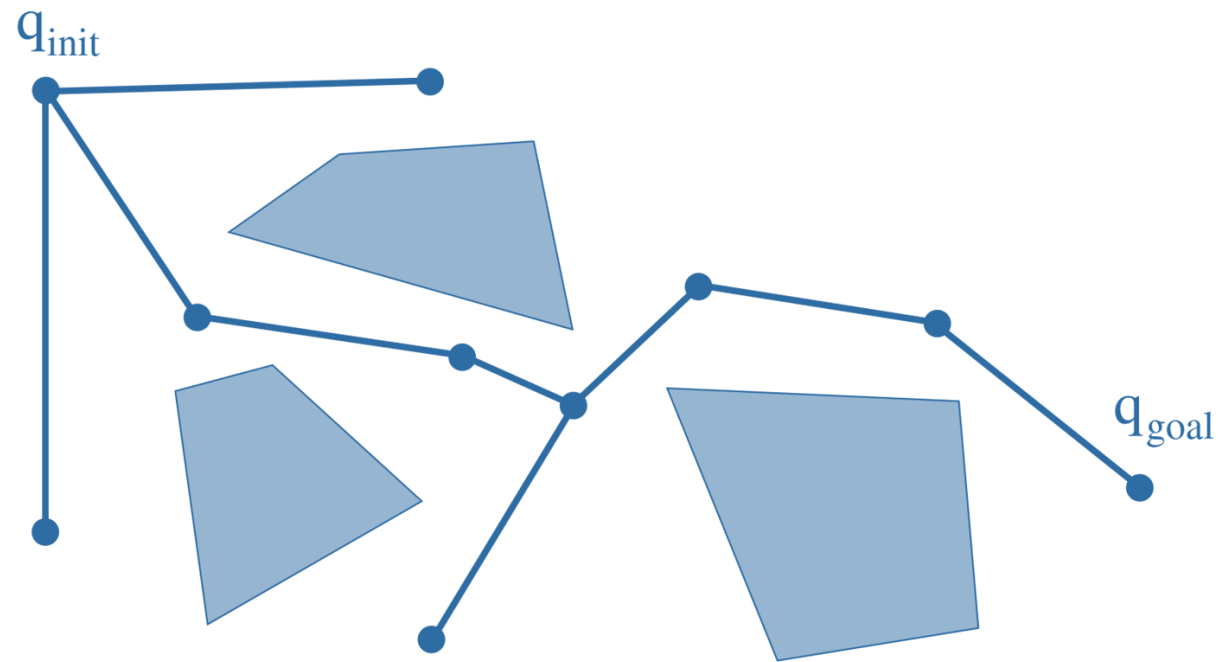
Probabilistic roadmap (PRM)



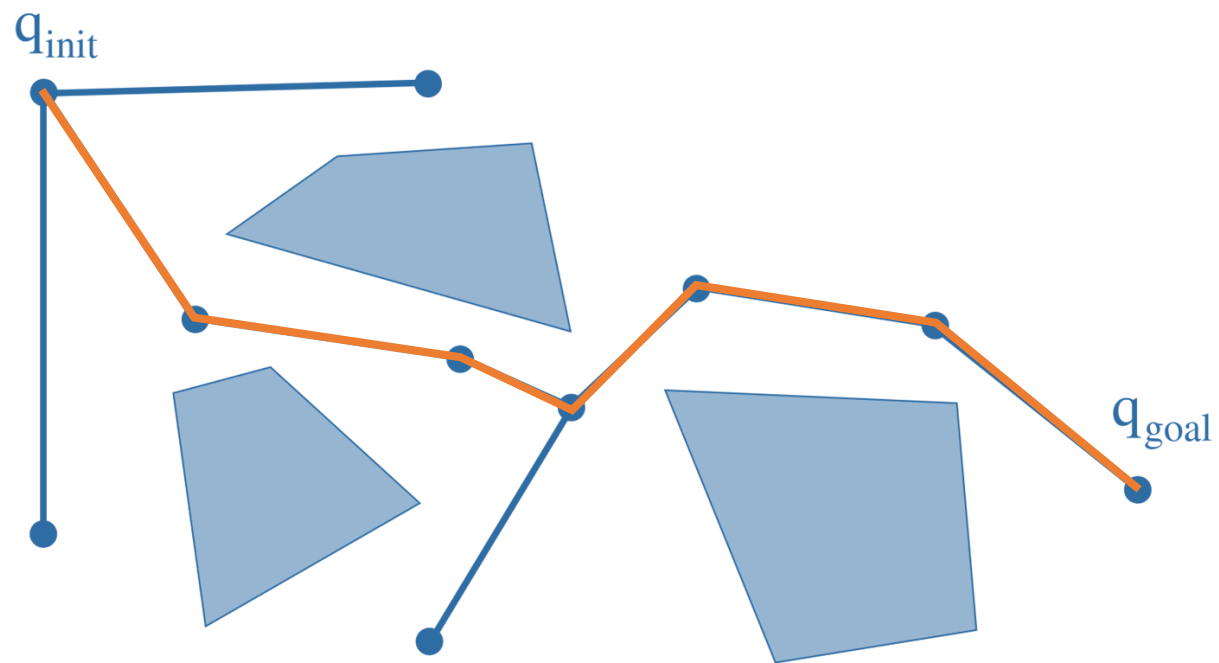
Probabilistic roadmap (PRM)



Probabilistic roadmap (PRM)



Probabilistic roadmap (PRM)



Visibility-based PRM

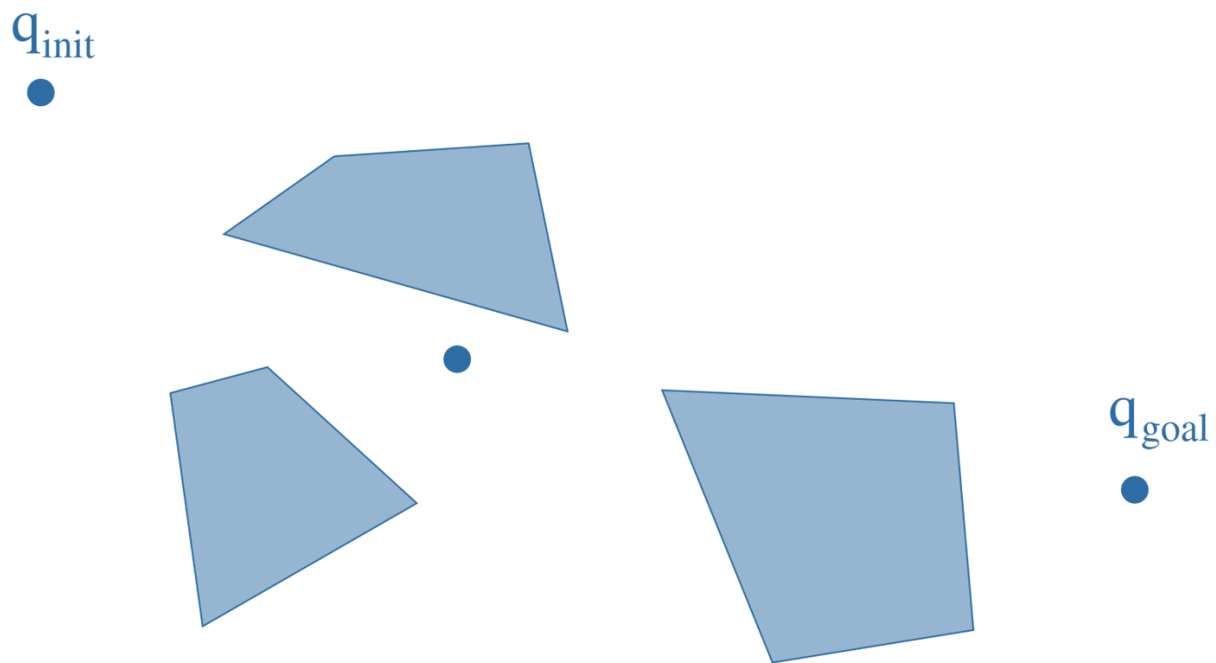
q_{init}



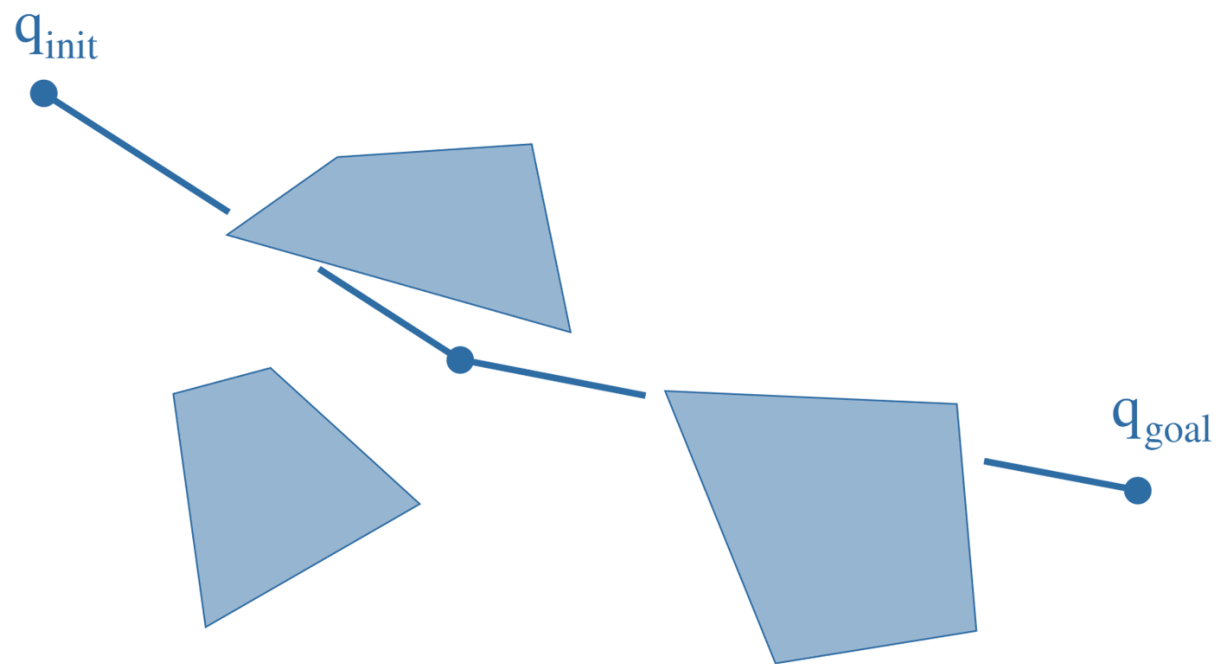
The diagram illustrates a visibility-based Probabilistic Roadmap (PRM) problem. It features a white background with three blue polygonal obstacles. The first obstacle is a quadrilateral in the upper-middle section. The second is a quadrilateral in the lower-left section. The third is a larger quadrilateral in the lower-right section. An initial configuration, labeled q_{init} , is represented by a blue dot in the upper-left area. A goal configuration, labeled q_{goal} , is represented by a blue dot in the middle-right area. The path from q_{init} to q_{goal} must avoid the obstacles, requiring a detour through the narrow passage between the two lower obstacles.

q_{goal}

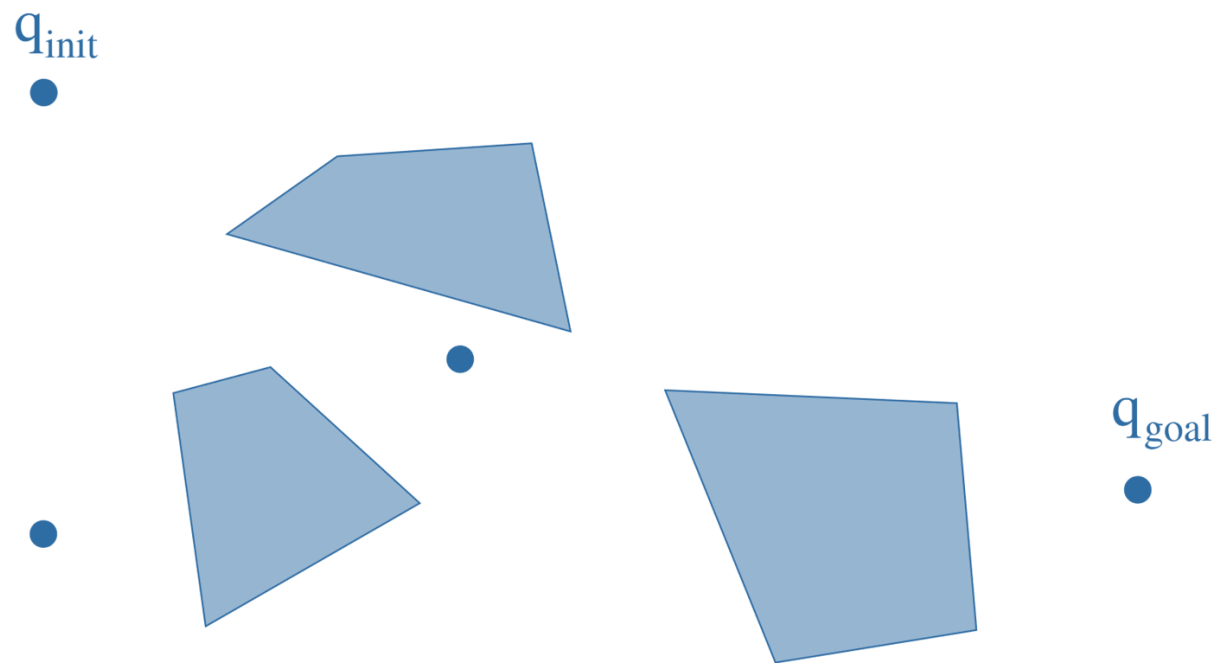
Visibility-based PRM



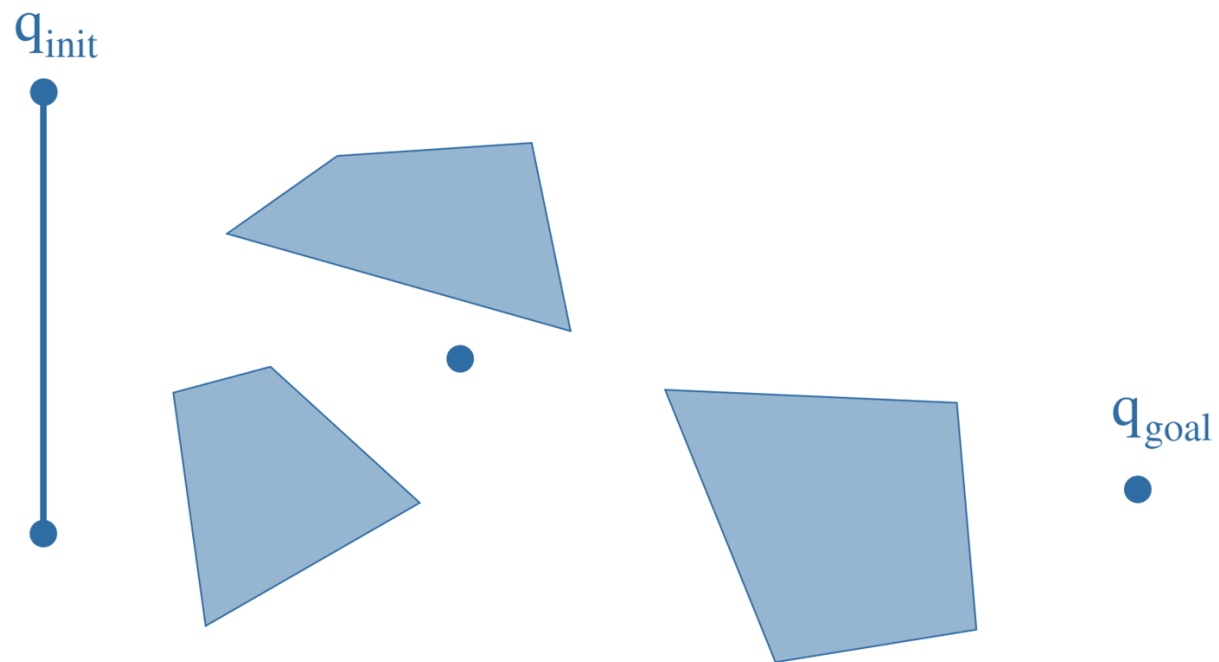
Visibility-based PRM



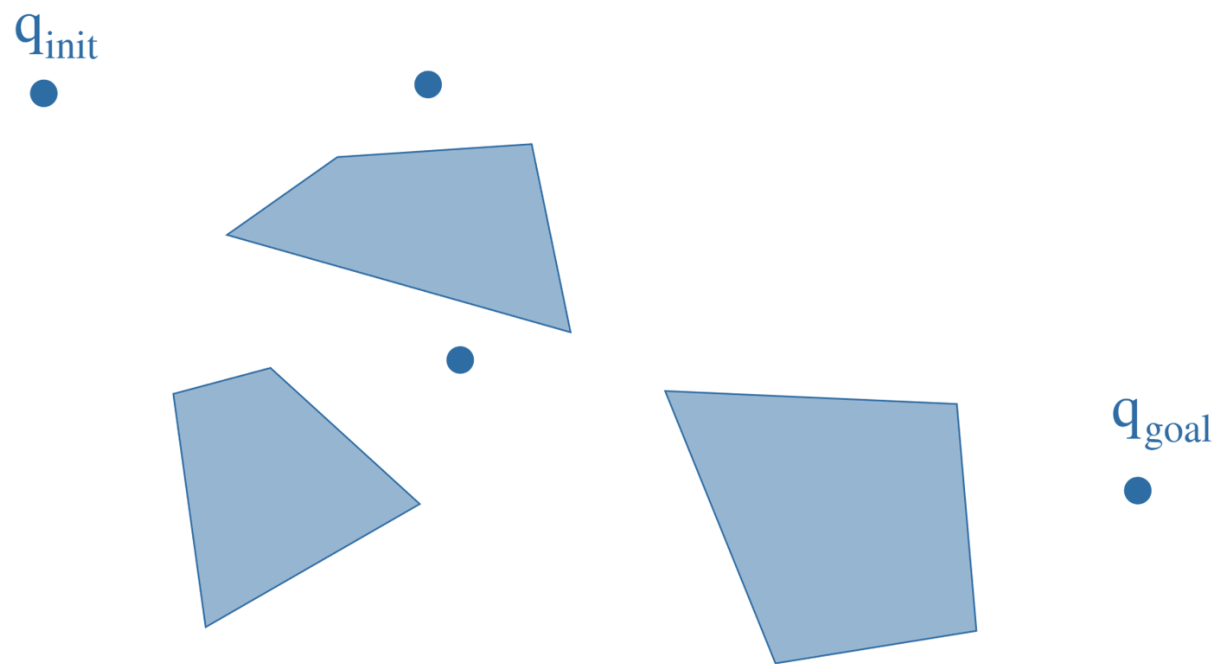
Visibility-based PRM



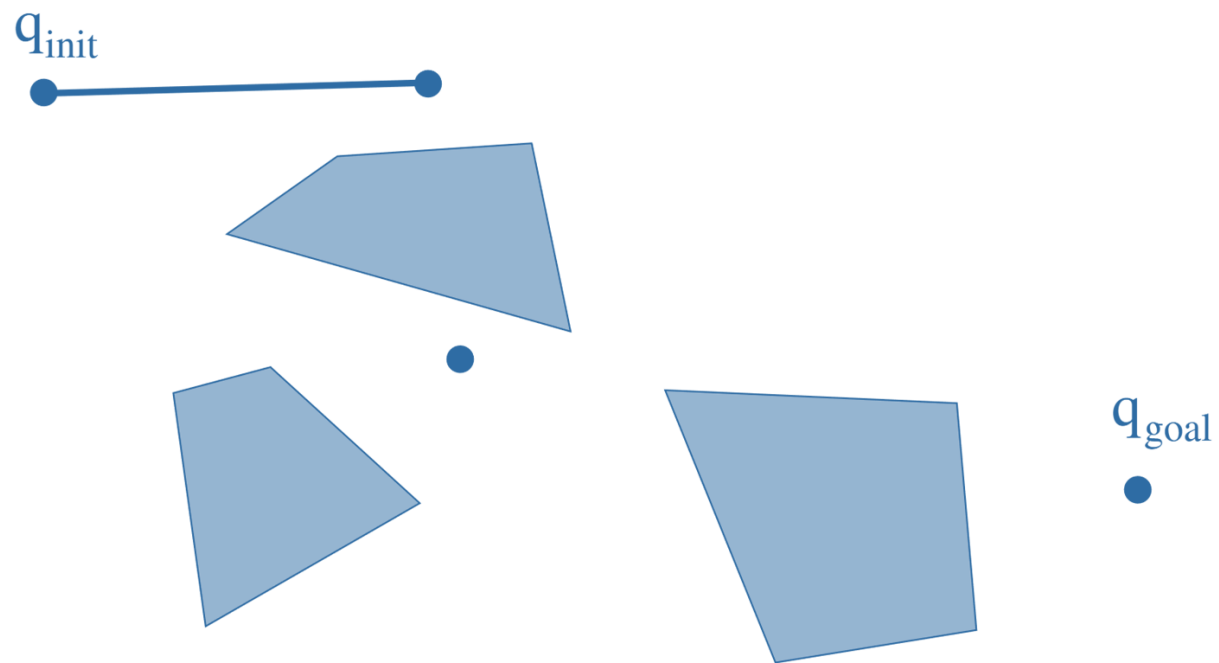
Visibility-based PRM



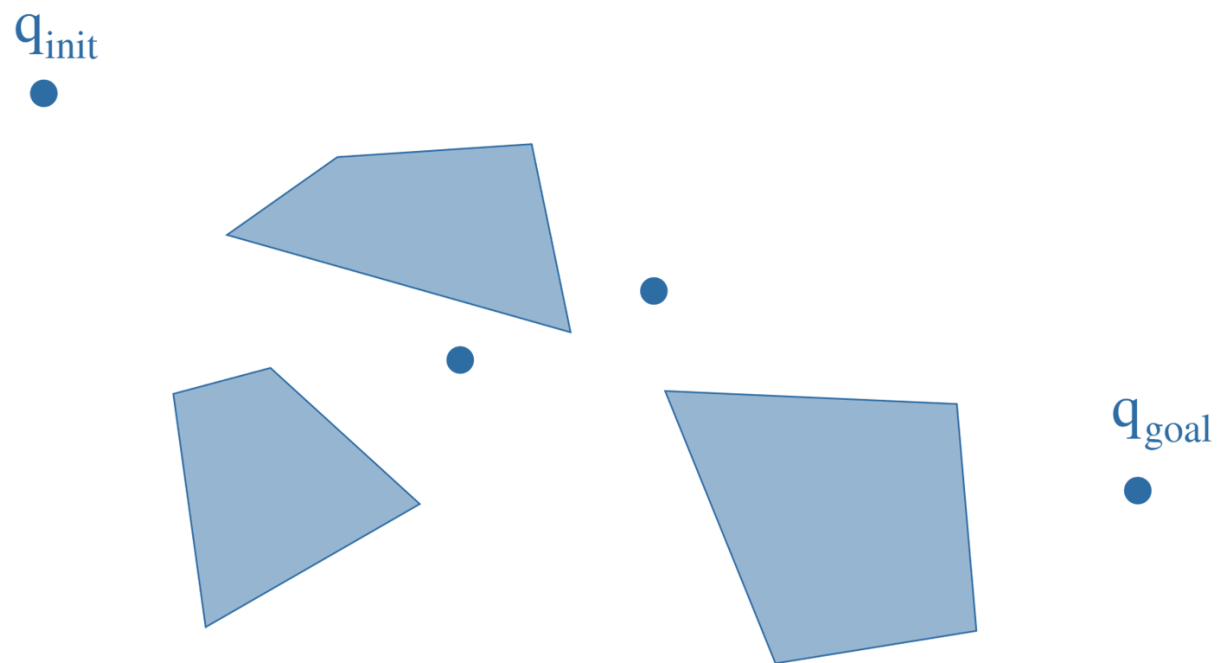
Visibility-based PRM



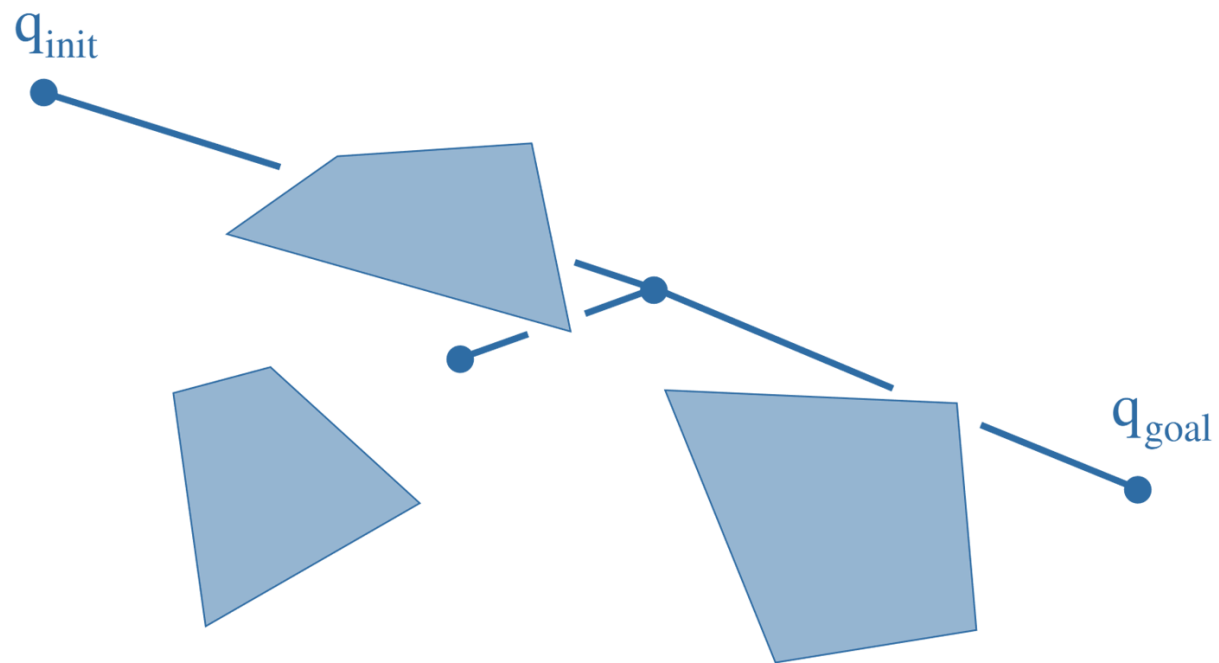
Visibility-based PRM



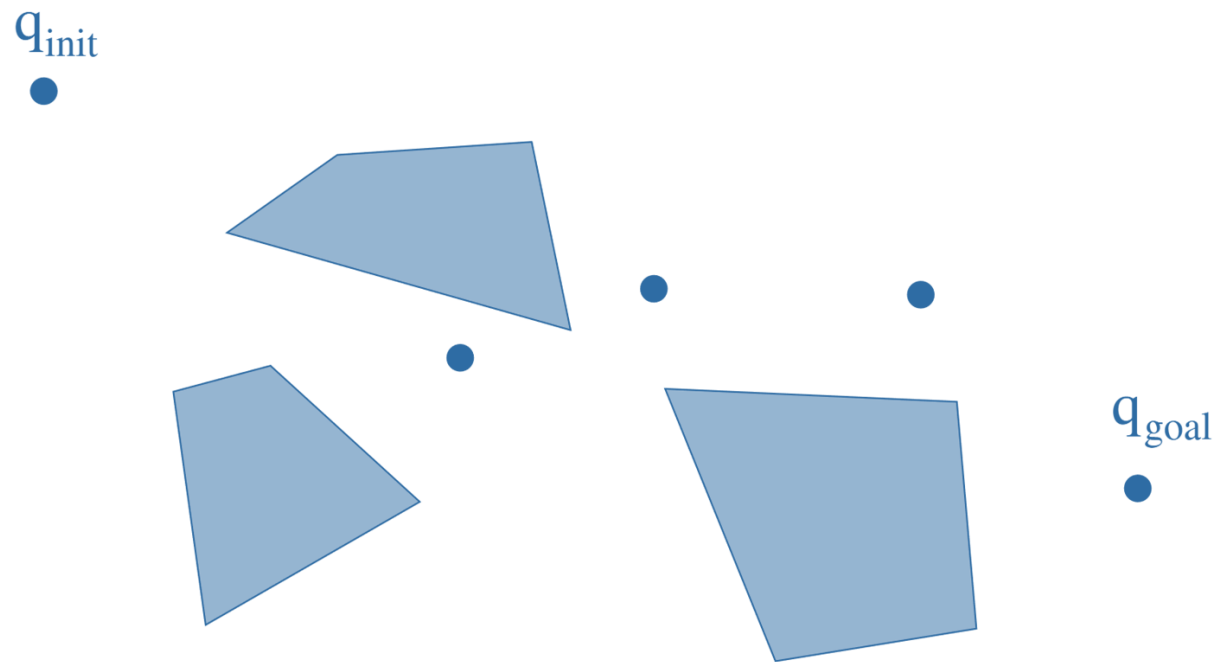
Visibility-based PRM



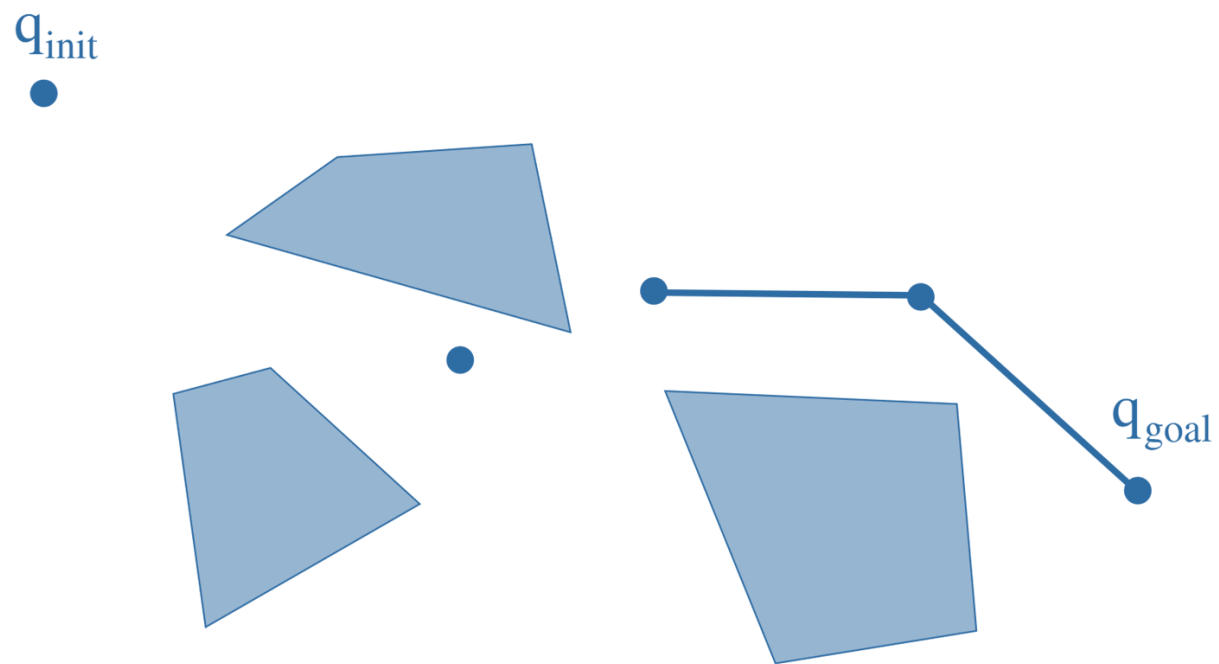
Visibility-based PRM



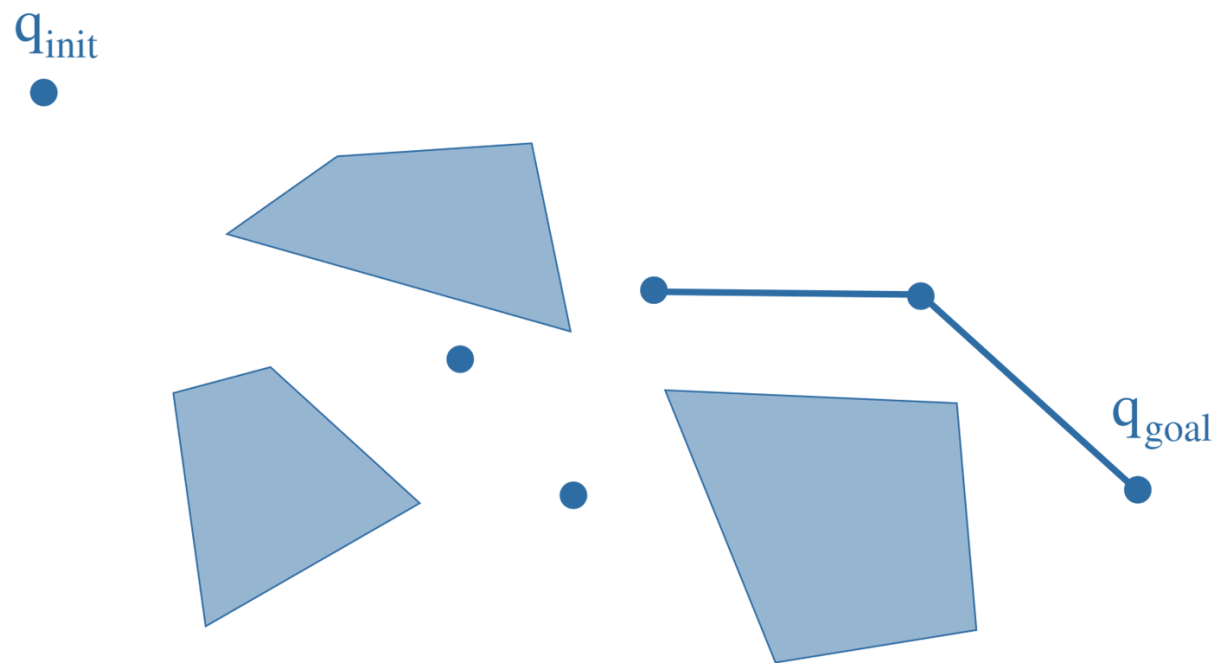
Visibility-based PRM



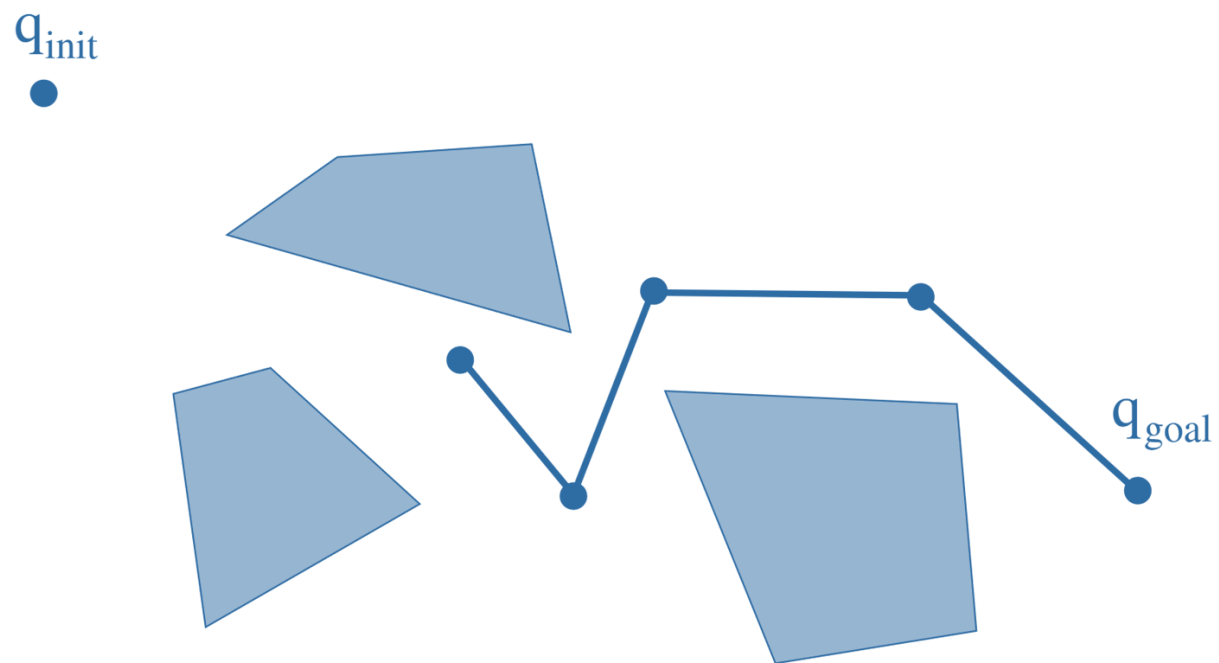
Visibility-based PRM



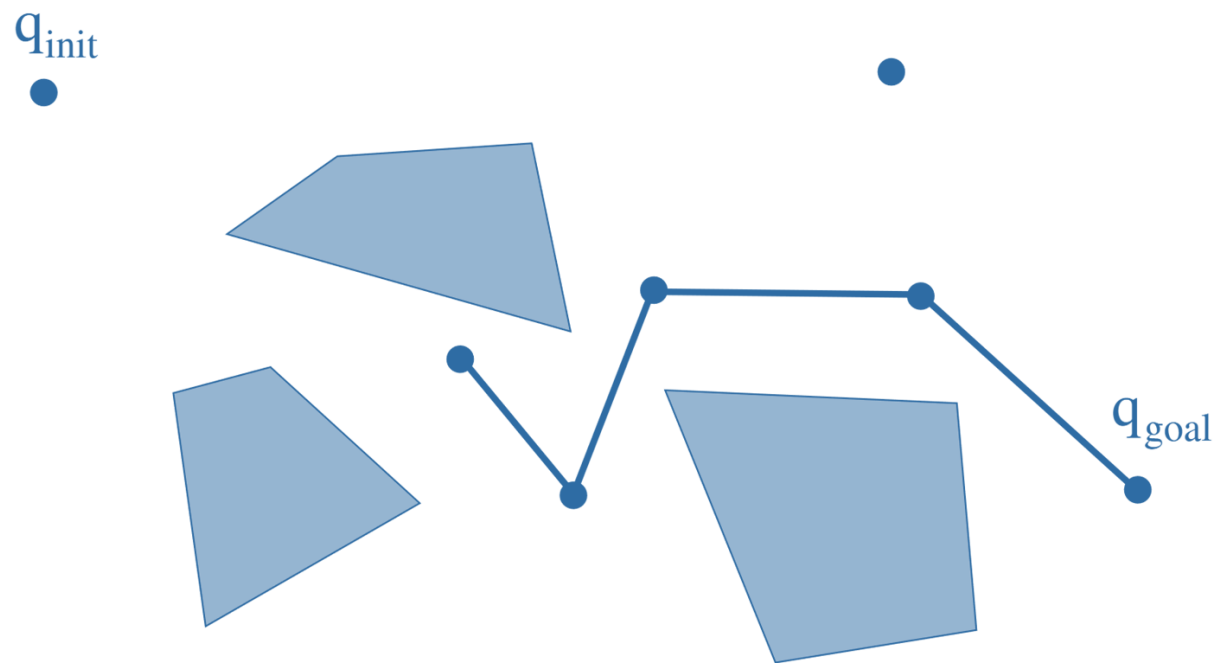
Visibility-based PRM



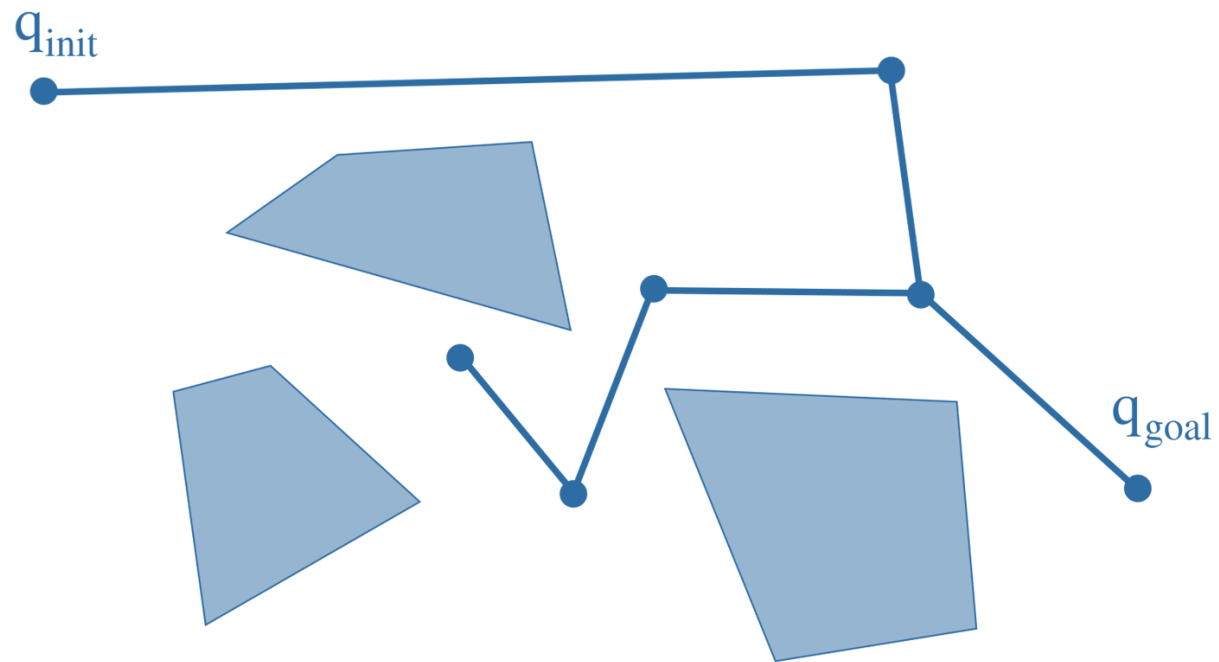
Visibility-based PRM



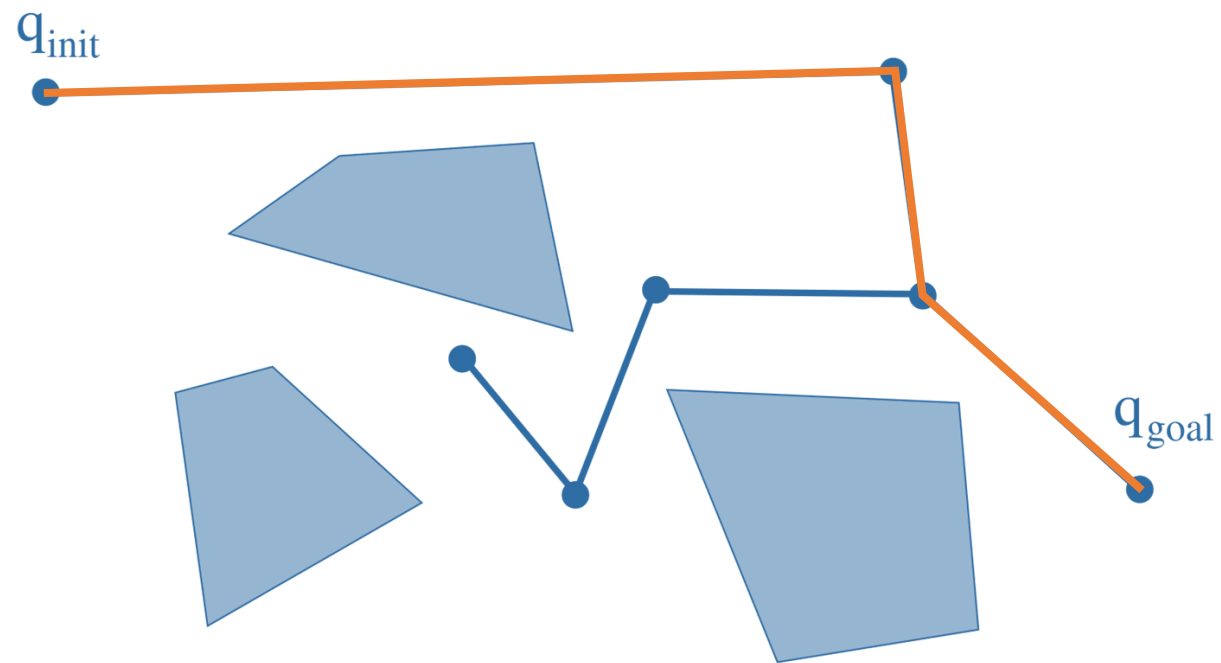
Visibility-based PRM



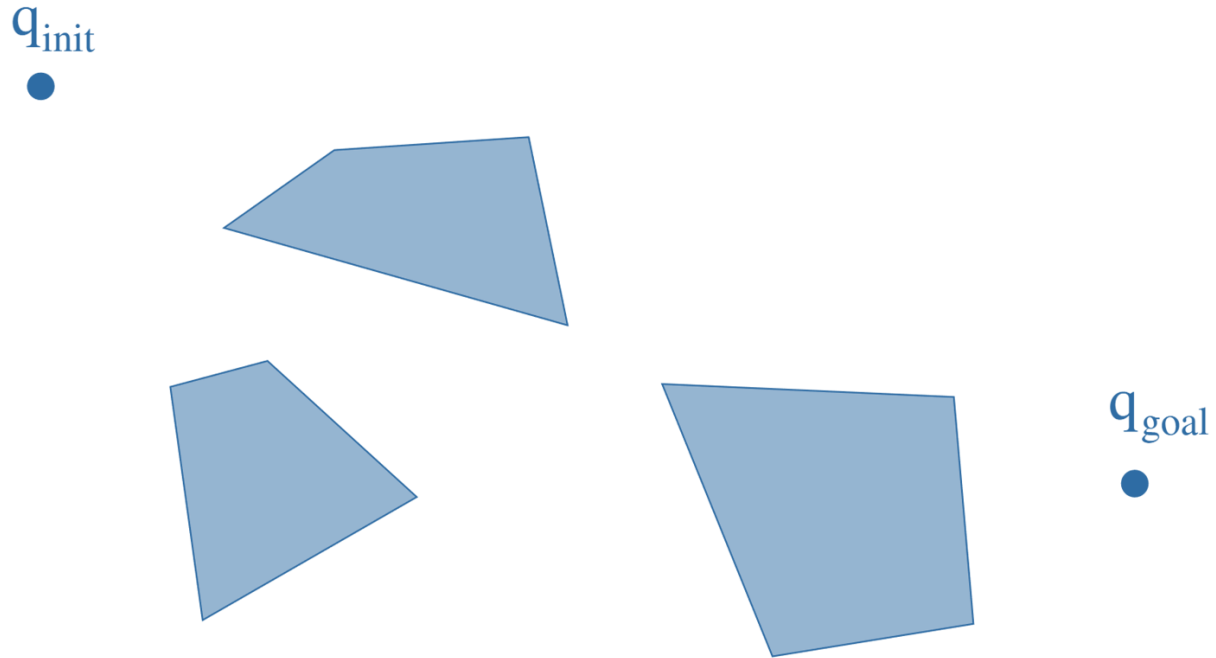
Visibility-based PRM



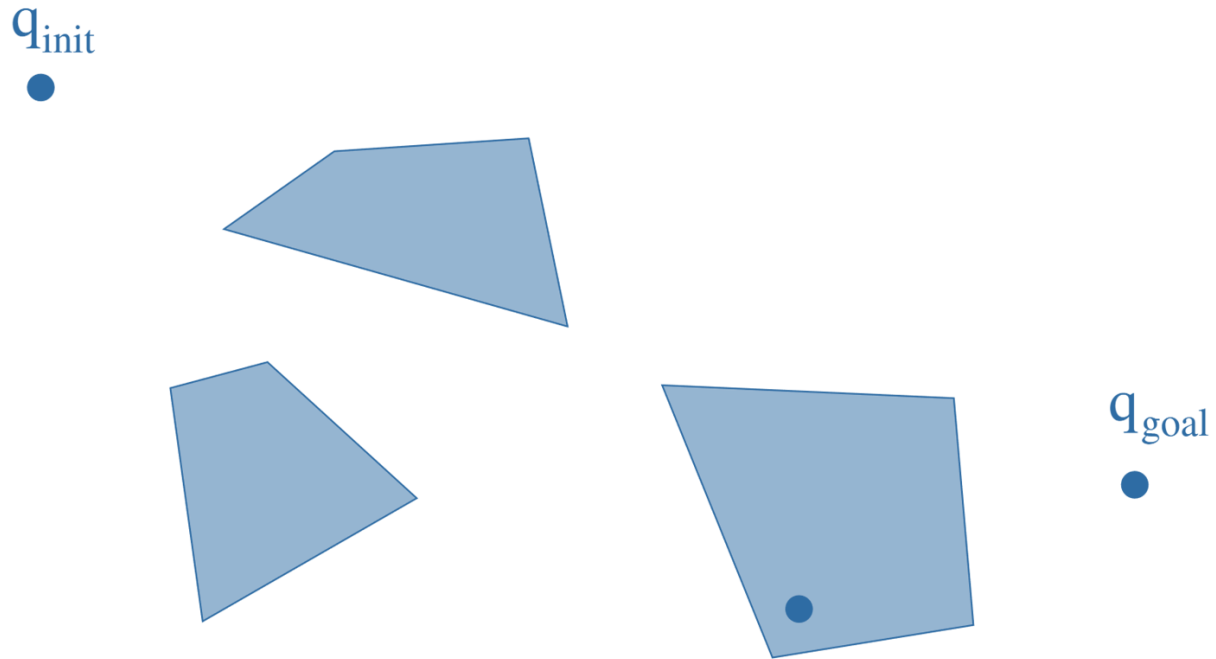
Visibility-based PRM



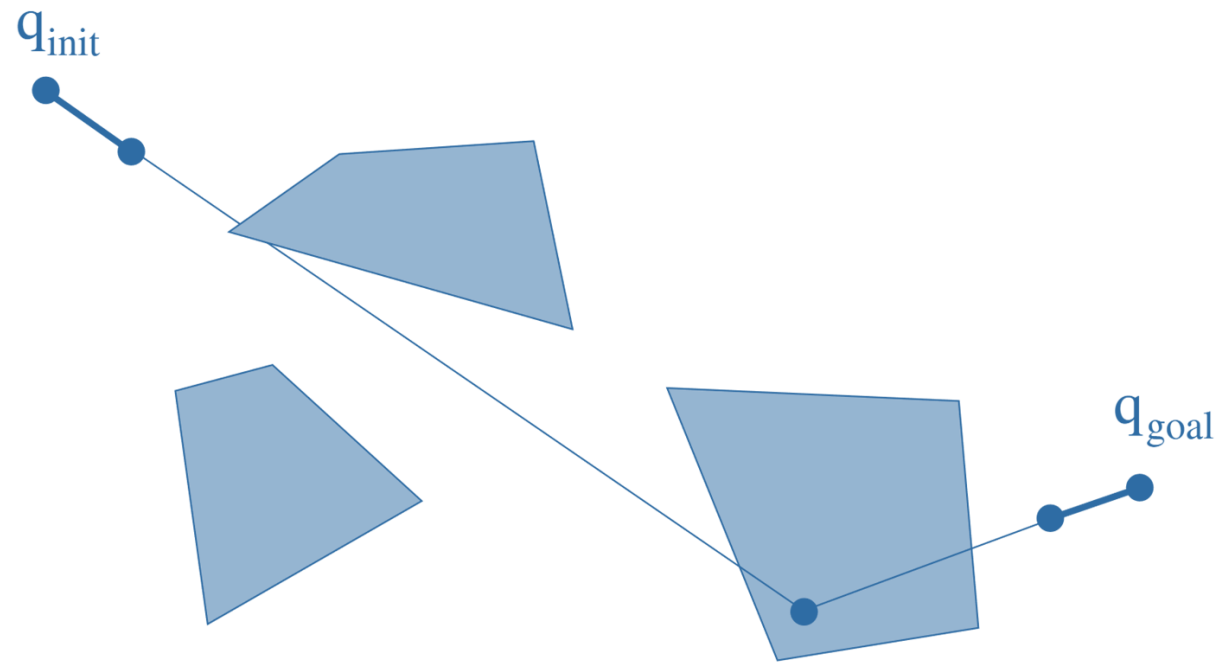
Rapidly exploring Random Tree (RRT)



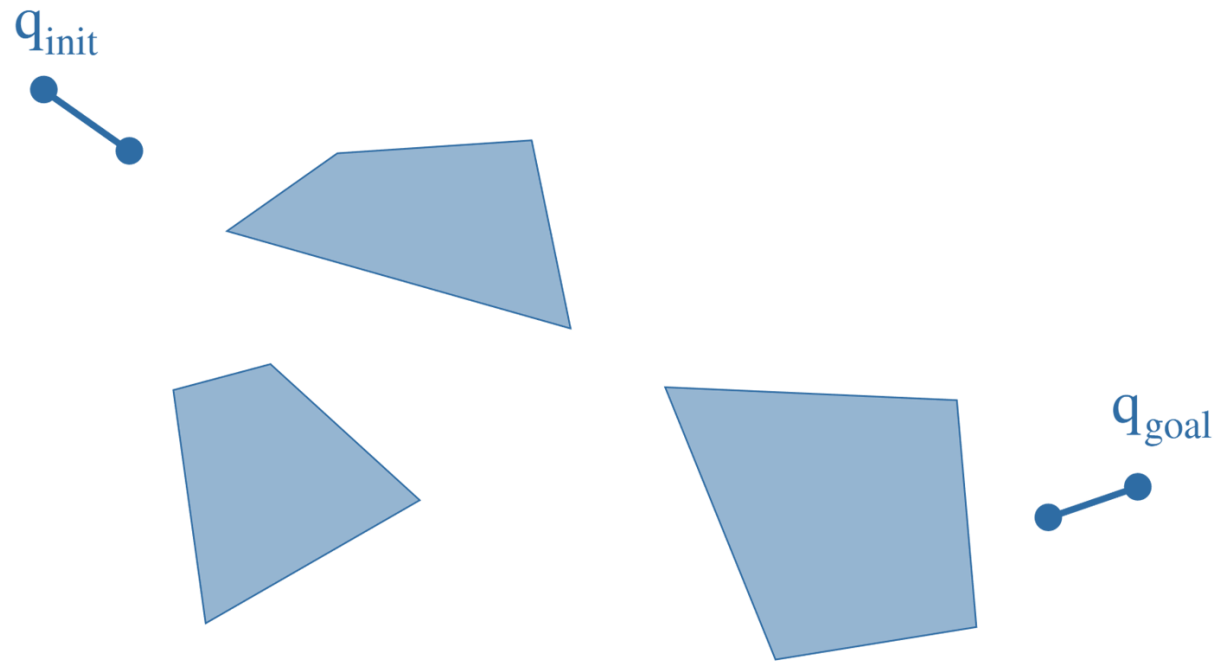
Rapidly exploring Random Tree (RRT)



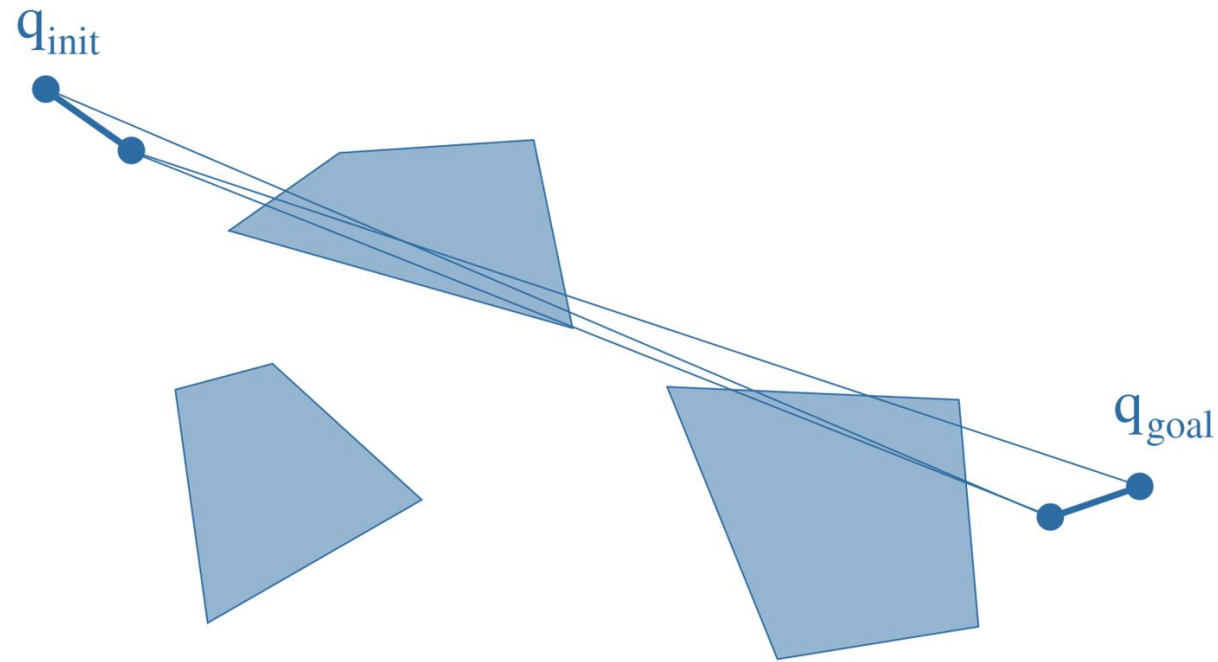
Rapidly exploring Random Tree (RRT)



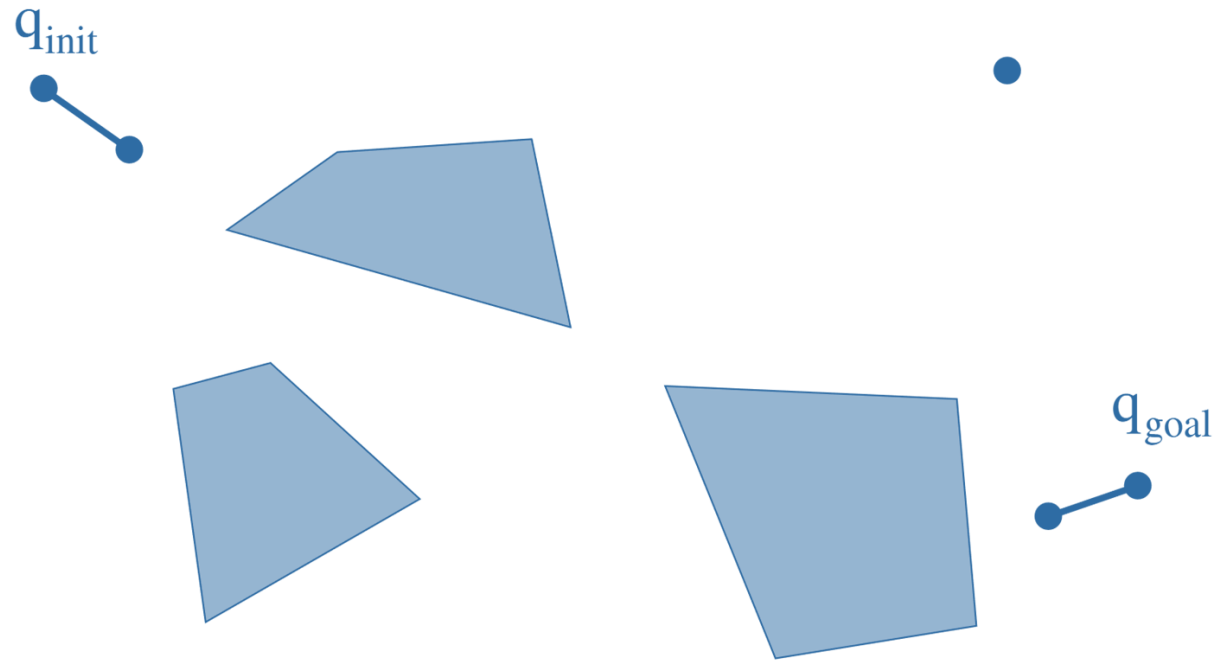
Rapidly exploring Random Tree (RRT)



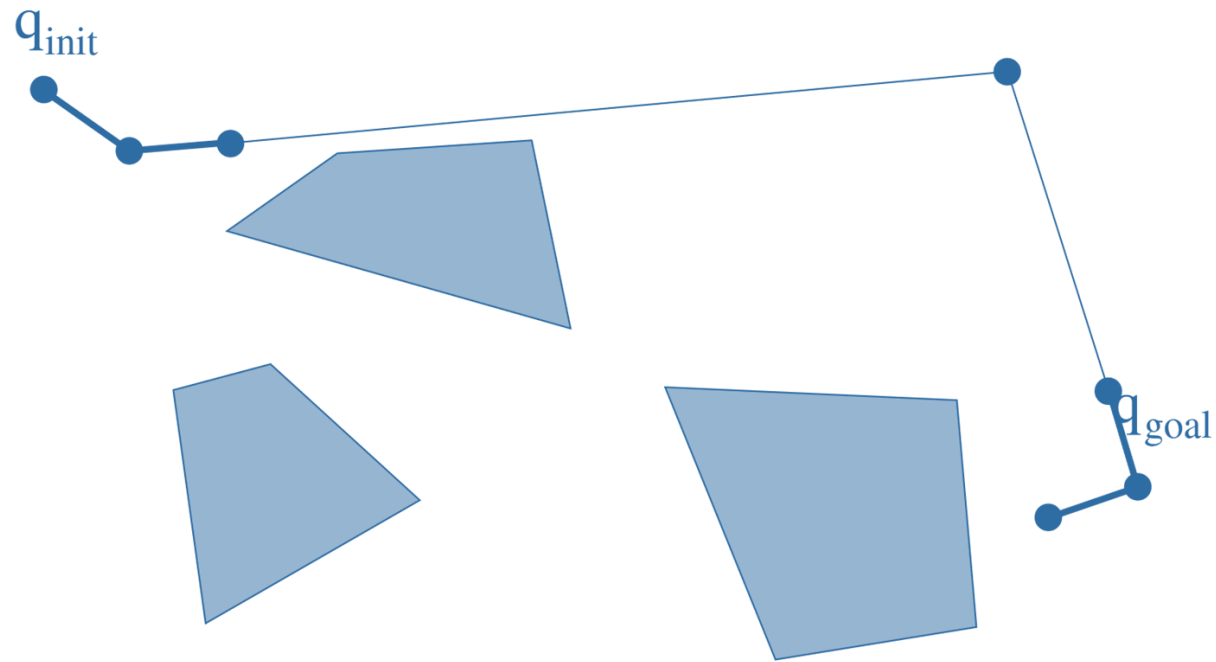
Rapidly exploring Random Tree (RRT)



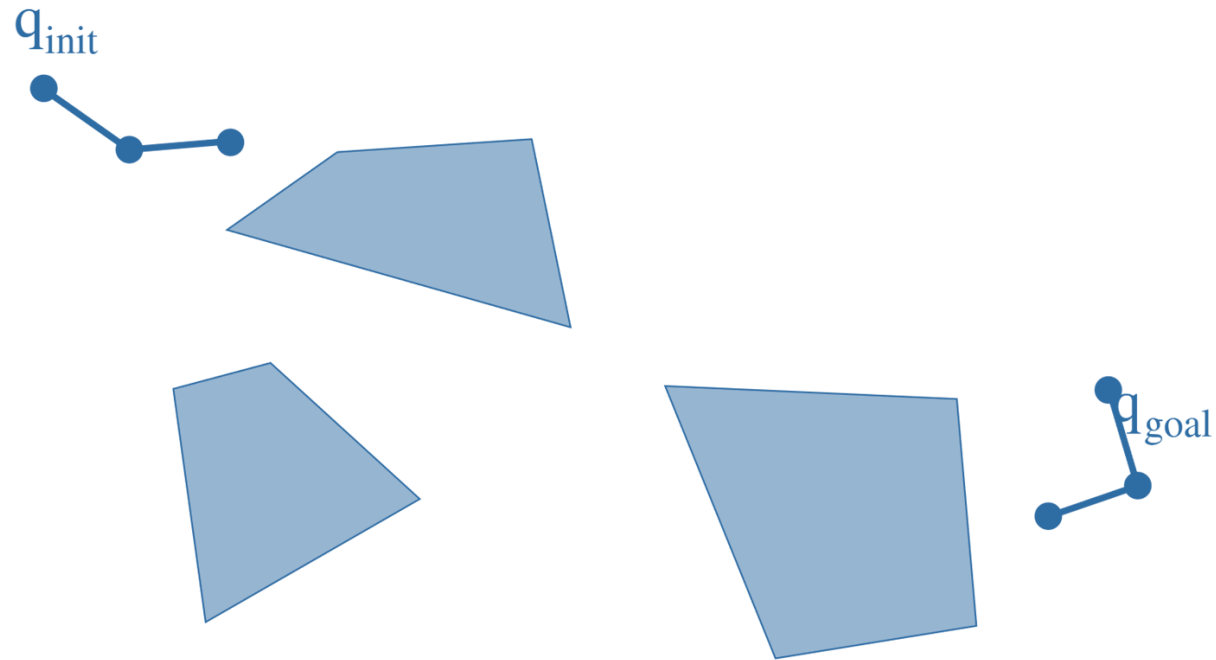
Rapidly exploring Random Tree (RRT)



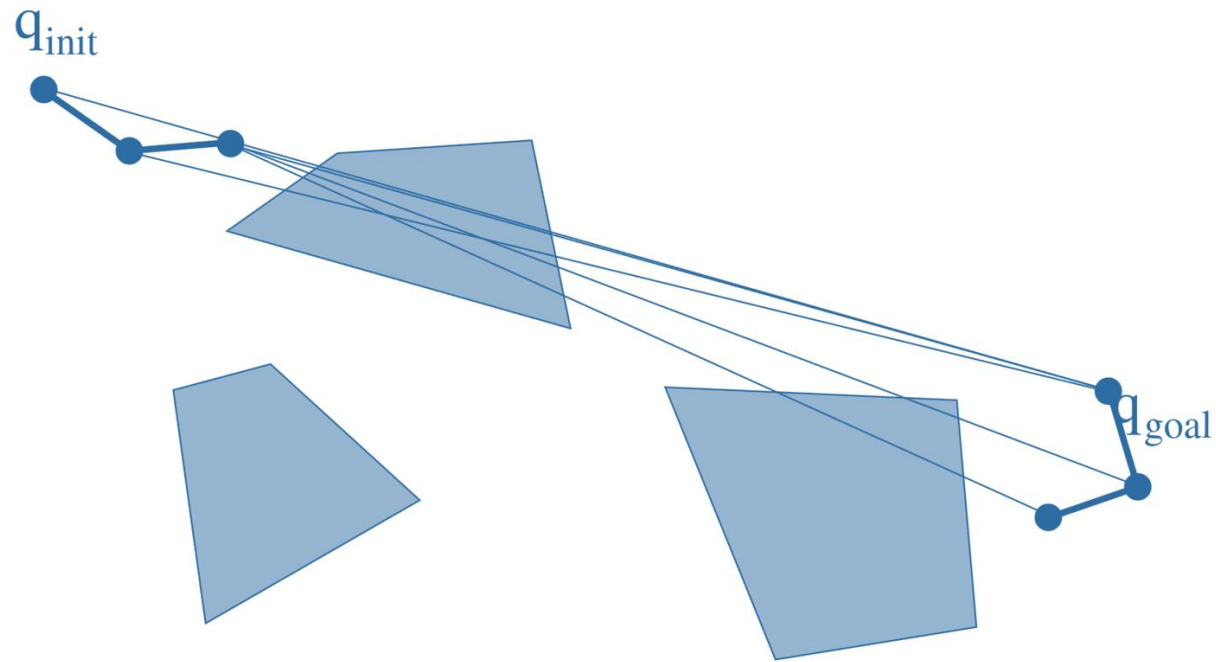
Rapidly exploring Random Tree (RRT)



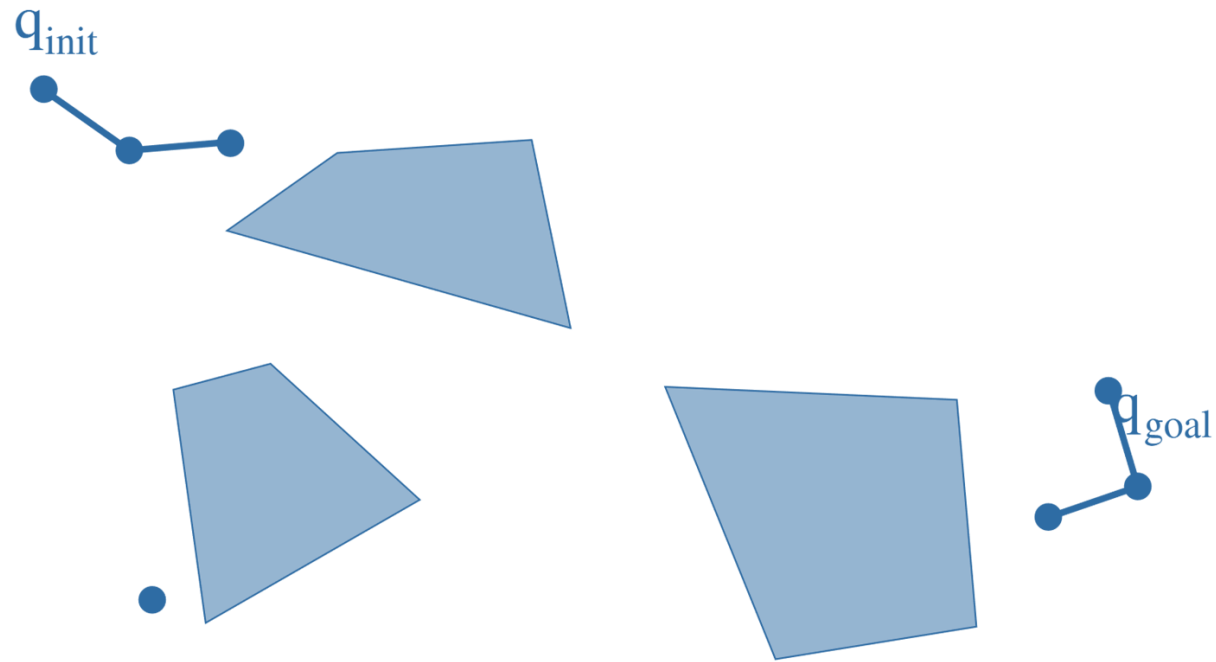
Rapidly exploring Random Tree (RRT)



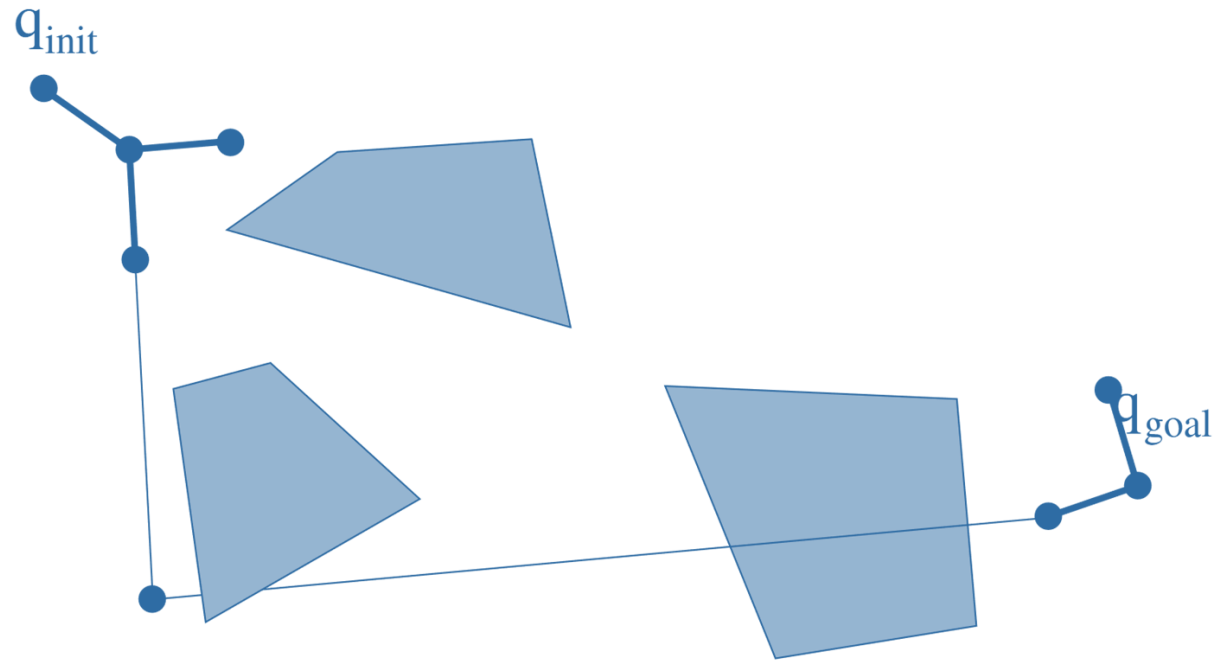
Rapidly exploring Random Tree (RRT)



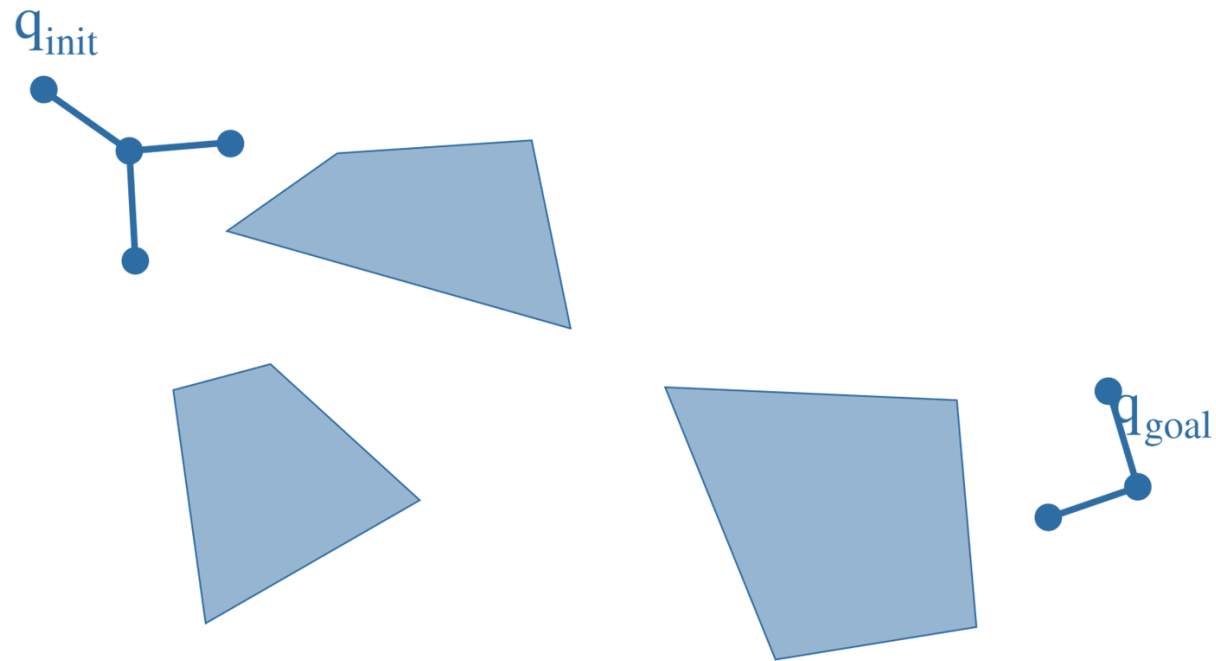
Rapidly exploring Random Tree (RRT)



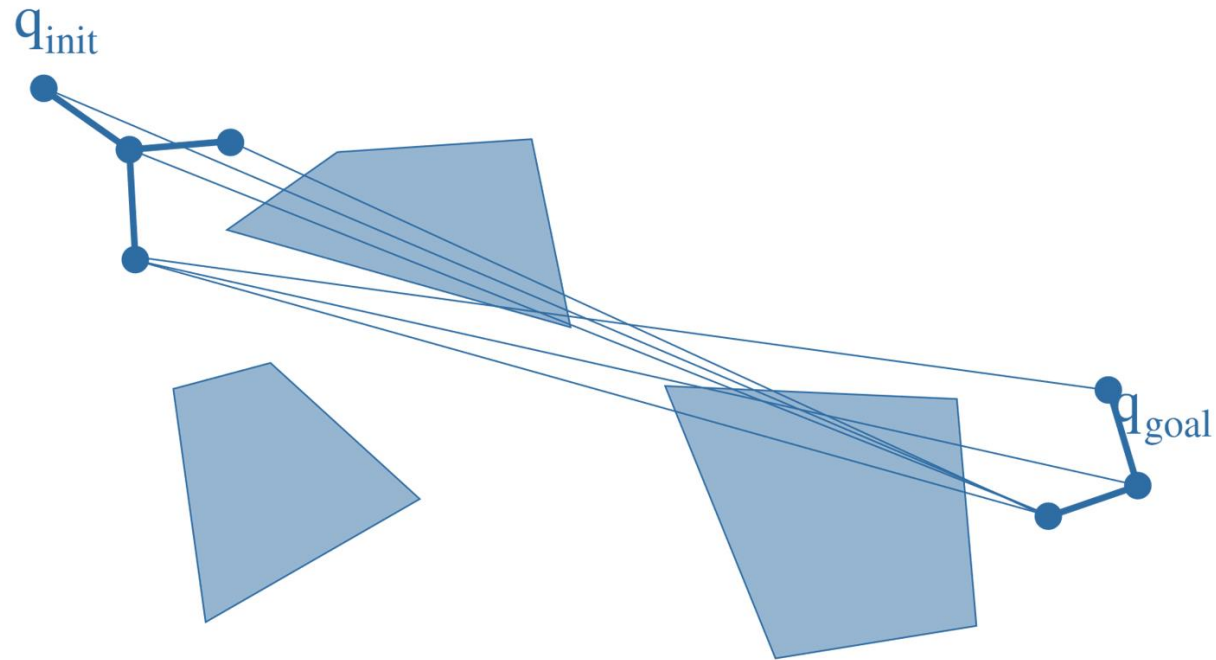
Rapidly exploring Random Tree (RRT)



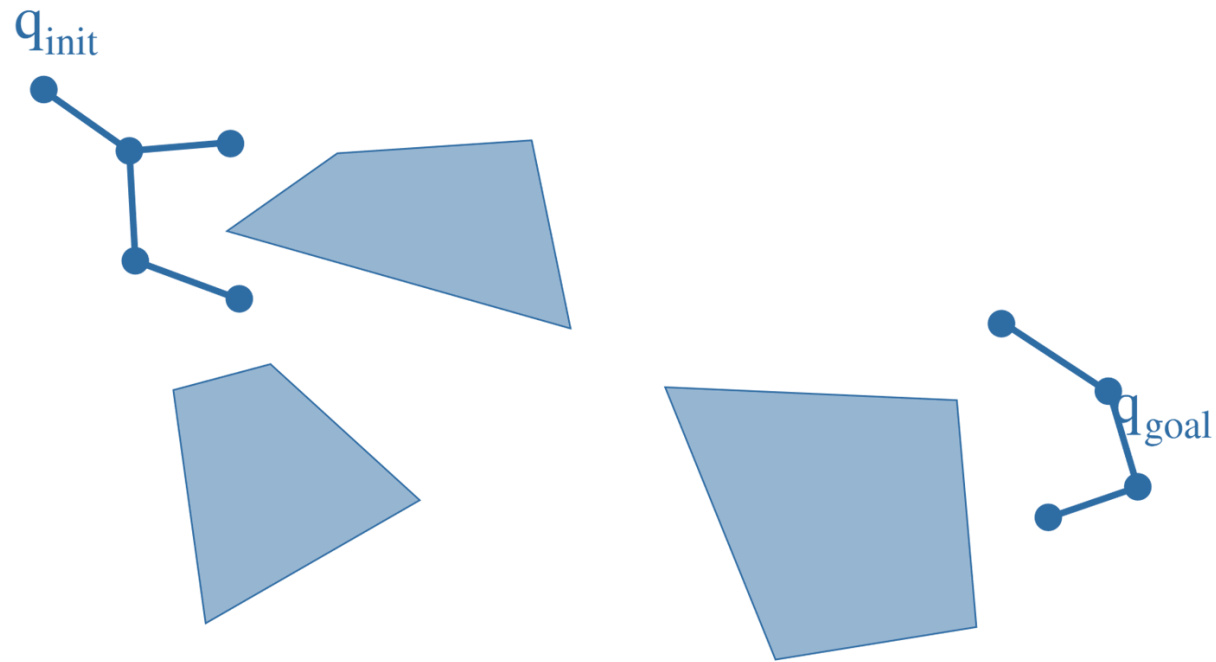
Rapidly exploring Random Tree (RRT)



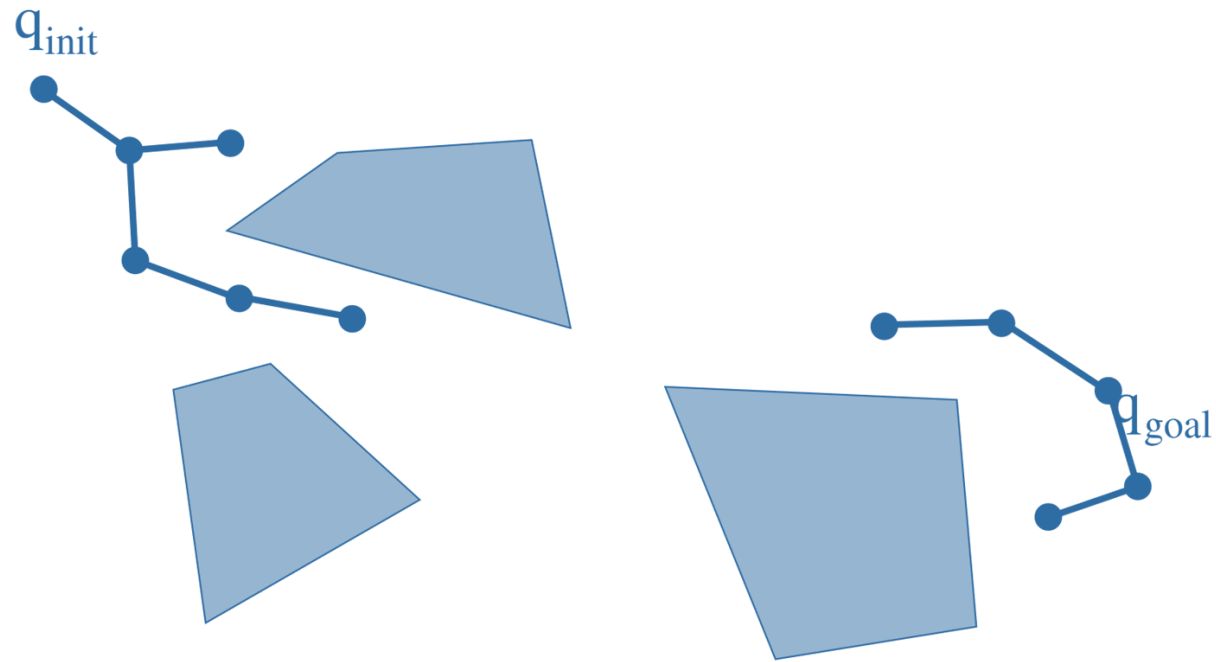
Rapidly exploring Random Tree (RRT)



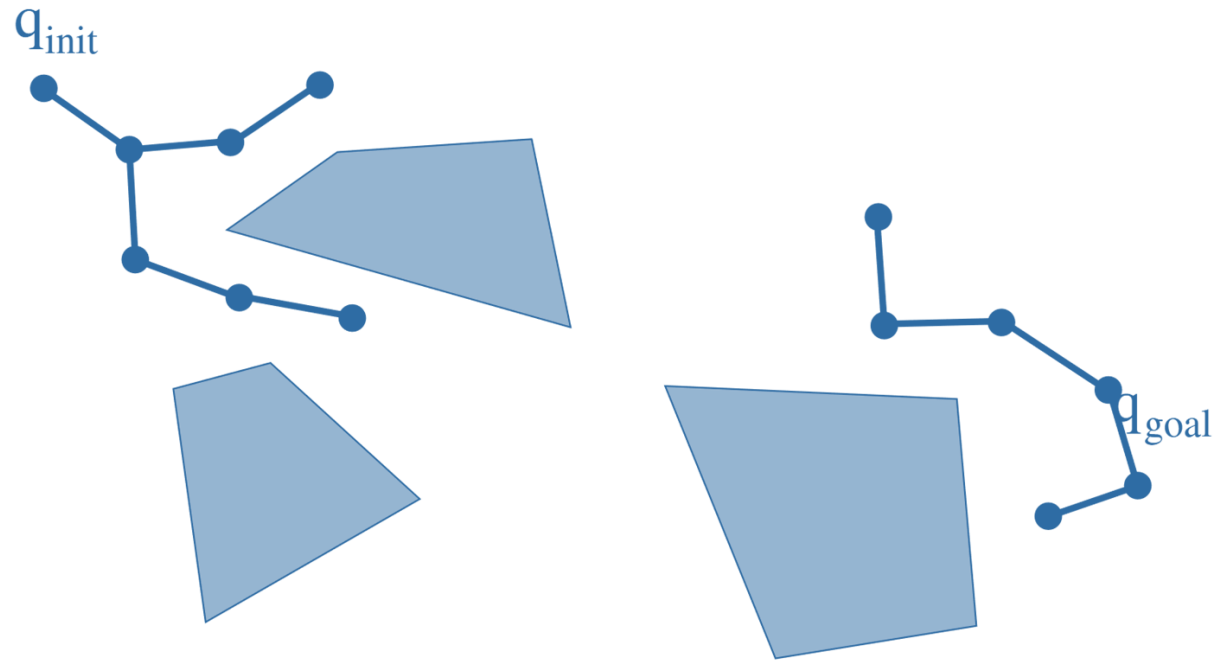
Rapidly exploring Random Tree (RRT)



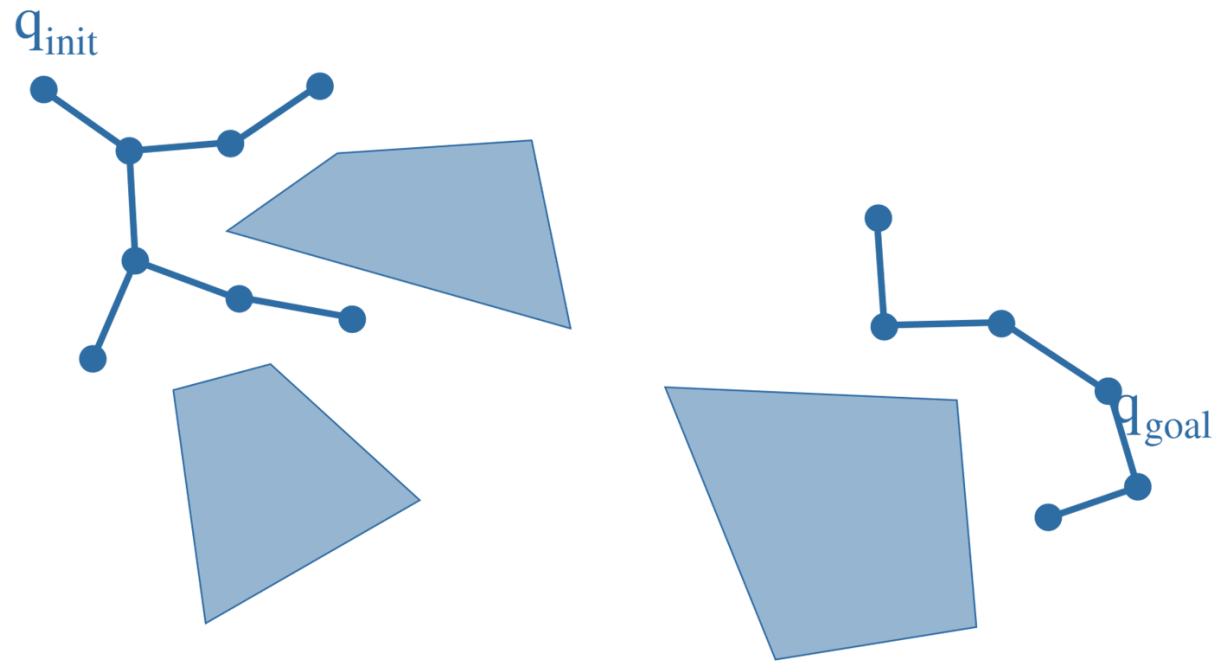
Rapidly exploring Random Tree (RRT)



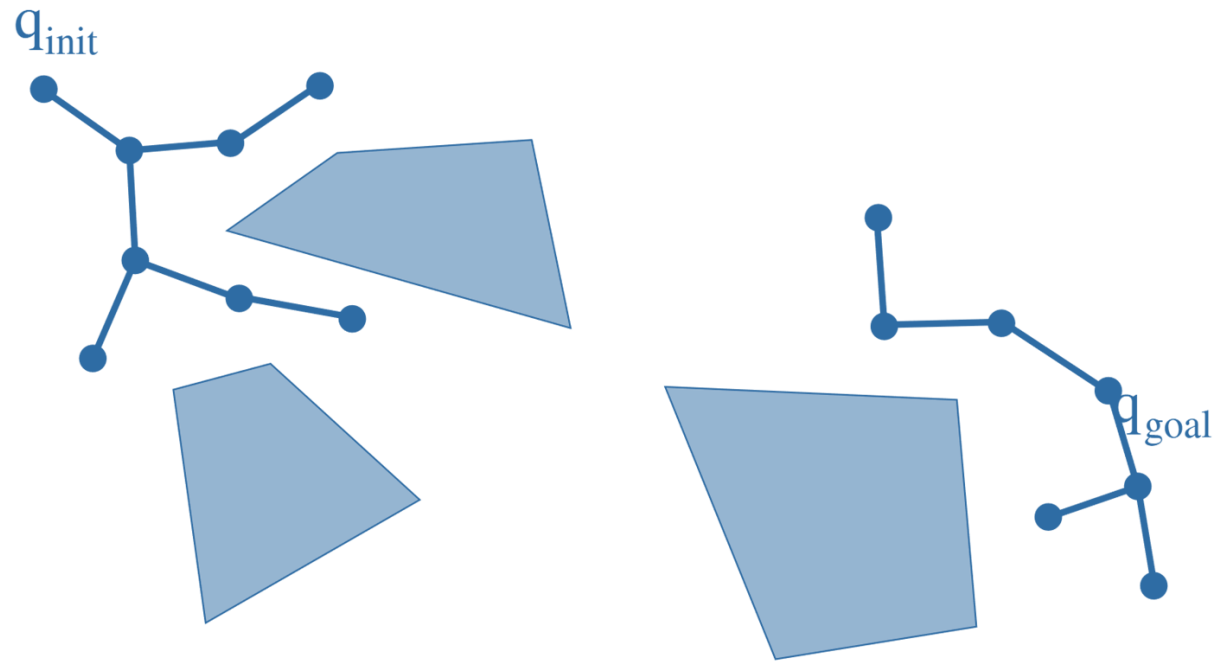
Rapidly exploring Random Tree (RRT)



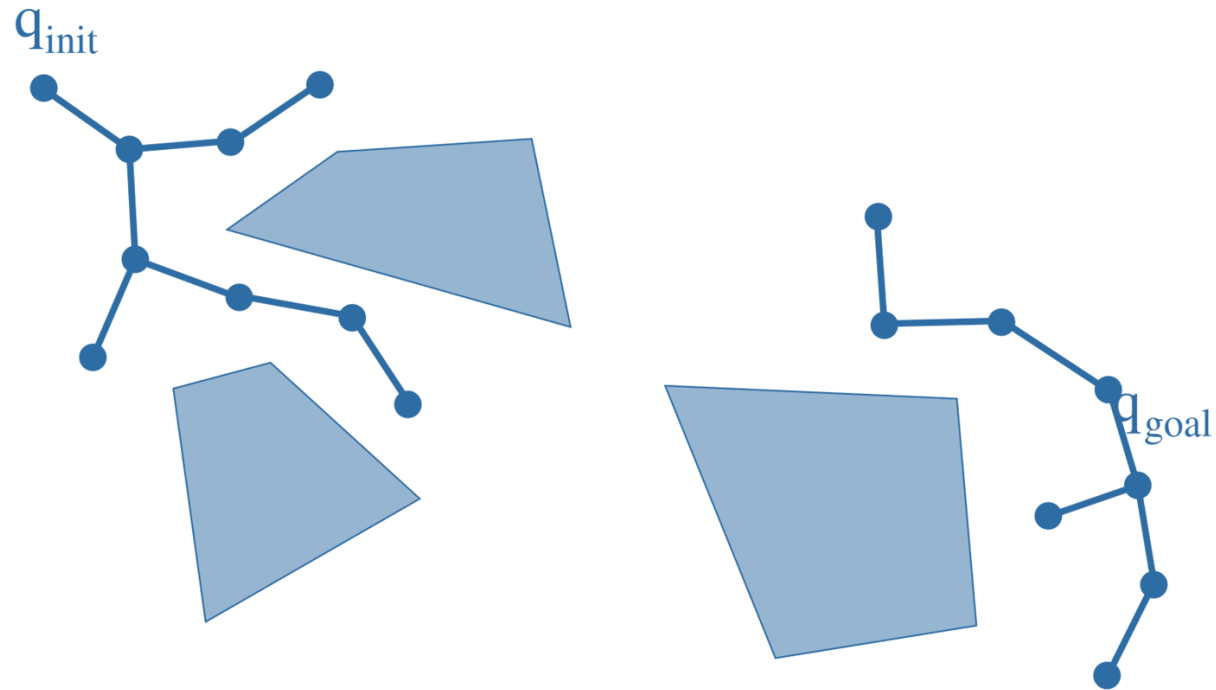
Rapidly exploring Random Tree (RRT)



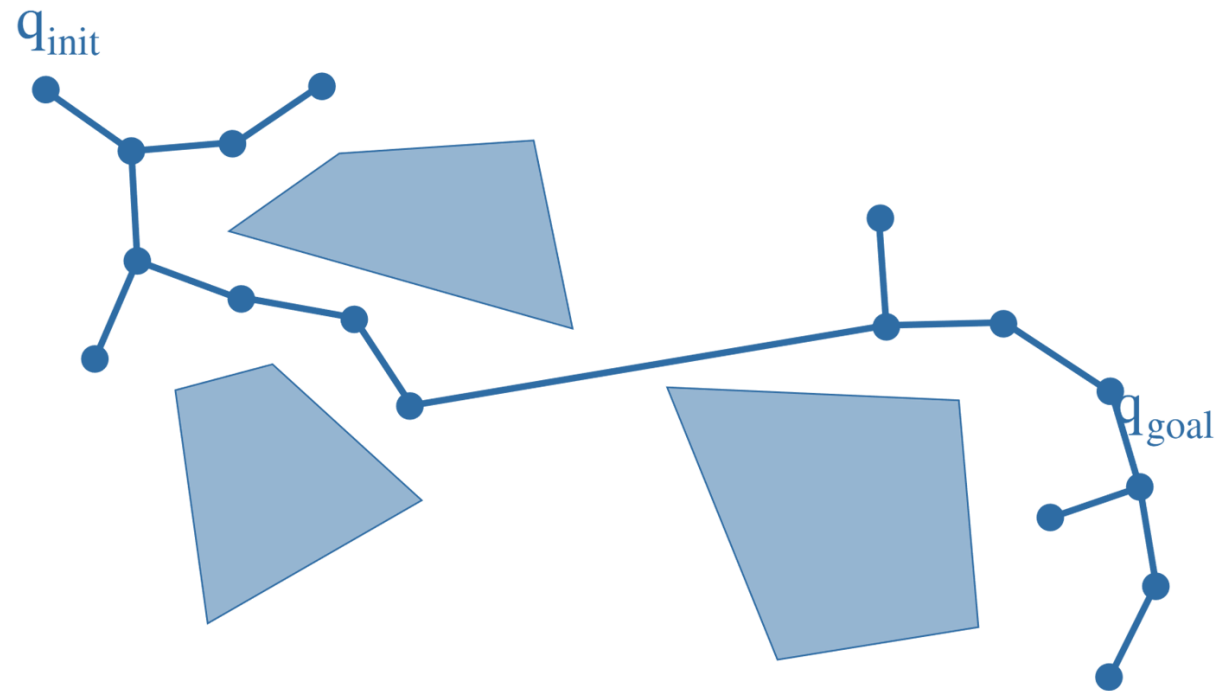
Rapidly exploring Random Tree (RRT)



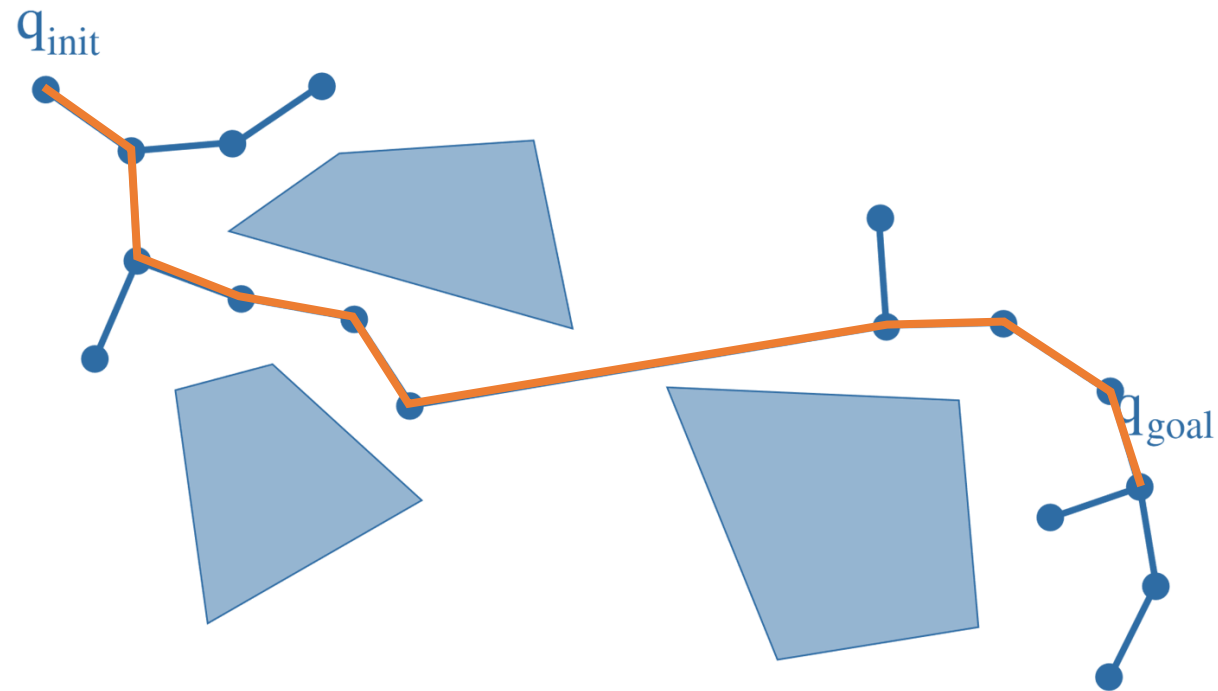
Rapidly exploring Random Tree (RRT)



Rapidly exploring Random Tree (RRT)



Rapidly exploring Random Tree (RRT)



Correctness and complexity

Complexity

- The piano mover problem is PSPACE-hard (Reif, 1979)
- Complexity of the cylindrical cell decomposition: running time bounded by $(md)^{O(1)^n}$ m number of Polynoms, d maximum degree of polynoms, n dimension of the C-space
- Canny's algorithm : $m^n (\log(d))^{O(n^4)}$

Correctness

- Completeness
 - The algorithm can find in finite time a solution or report in finite time the absence of solution
- Resolution completeness
 - If the sampling is deterministic following a dense sequence, then the algorithm will find in finite time a solution if it exists
 - The algorithm cannot report in finite time the absence of solution
- Probablistic completeness
 - If a solution exists, then the probability of the algorithm of find the solution tends to 1 as the number of samples/expansions tend to infinity
 - The algorithm cannot report in finite time the absence of solution

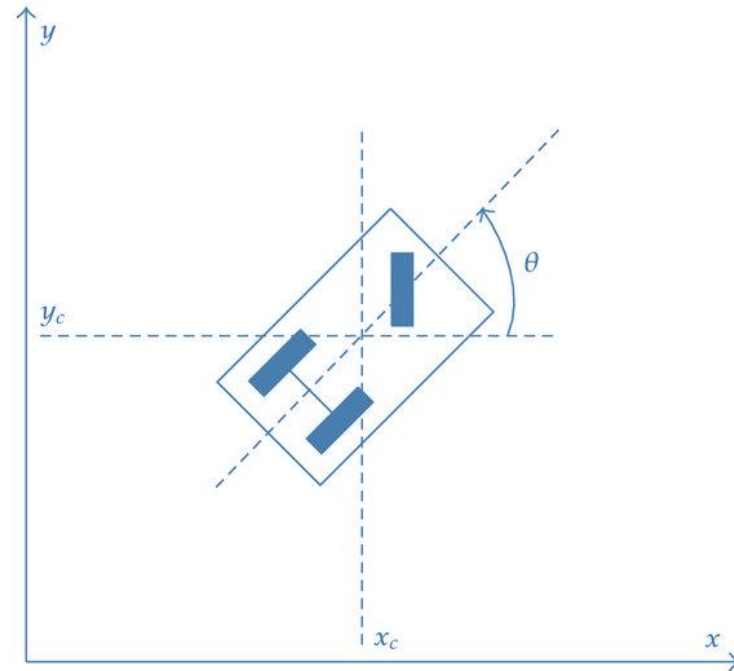
Correctness

- Exact roadmap building and cell decomposition methods are complete (certical cell decomposition, cylindrical cell decomposition)
- Sampling-based methods (PRM, RRT)
 - are probabilistically complete if random sampling is used
 - are resolution complete if a deterministic dense sampling sequence is used

Path/motion planning for non
holonomic systems

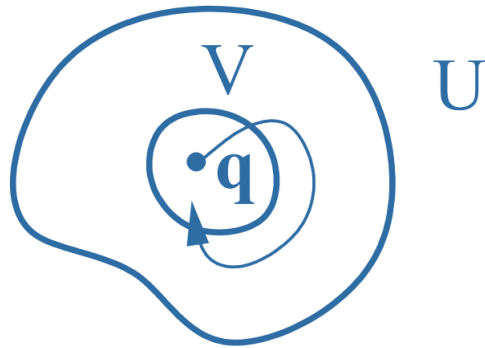
Nonholonomic systems

- Nonholonomic systems involve differential constraints
- e.g. a mobile robot (car) only has two kinematic controls
 - steering wheel
 - gaz/brake
- but evolves in $SE(2)$ (3-d)
- Notion of admissible paths



Controllability of nonholonomic systems

- Nonholonomic systems are locally controllable iff for any q in \mathcal{C} and any neighborhood U of q there exists a neighborhood V of q completely reachable from q by admissible paths included in U

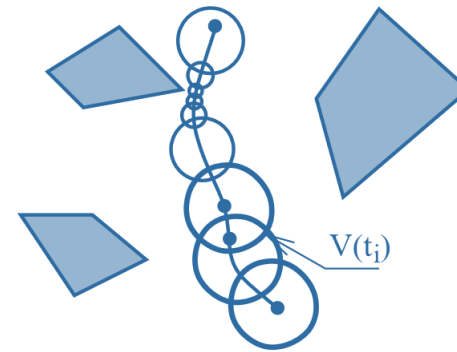
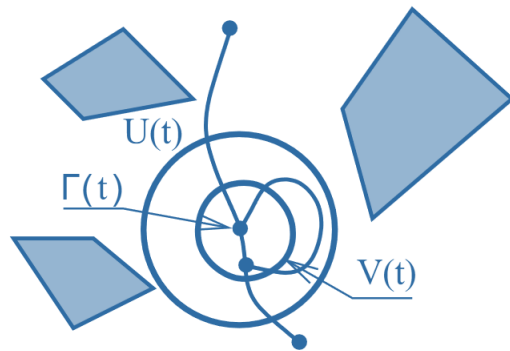


Controllability of nonholonomic systems

- It can be demonstrated that a nonholonomic system is controllable iff the dimension of the control vector field Lie Algebra of the system is n
- Lie brackets of control field add the missing dimensions
- E.g. for a mobile robot, or a car, the Lie bracket of the two control vector field correspond to the parallel parking maneuver (créneau) that makes the system move sideways (which was the missing dimension in the control)

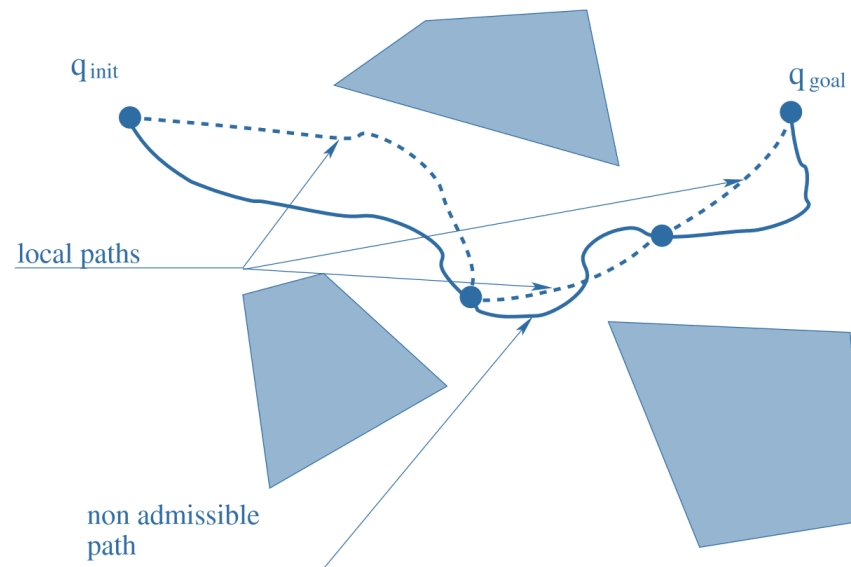
Reduction property

- For a locally controllable system, the existence of a collision-free path between two points of the uncontrained system in $SE(2)$ is equivalent to the existence of a collision free adimissible path for the constrained nonholonomic system



Path planning for nonholonomic systems

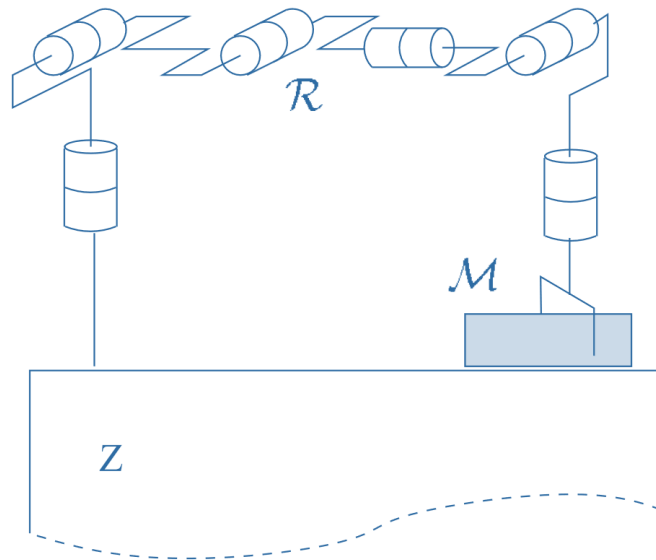
- Path planning for nonholonomic systems is based on path planning of unconstrained system and deformation by dichotomy of non admissible paths applying a local steering method



Manipulation planning

Manipulation planning

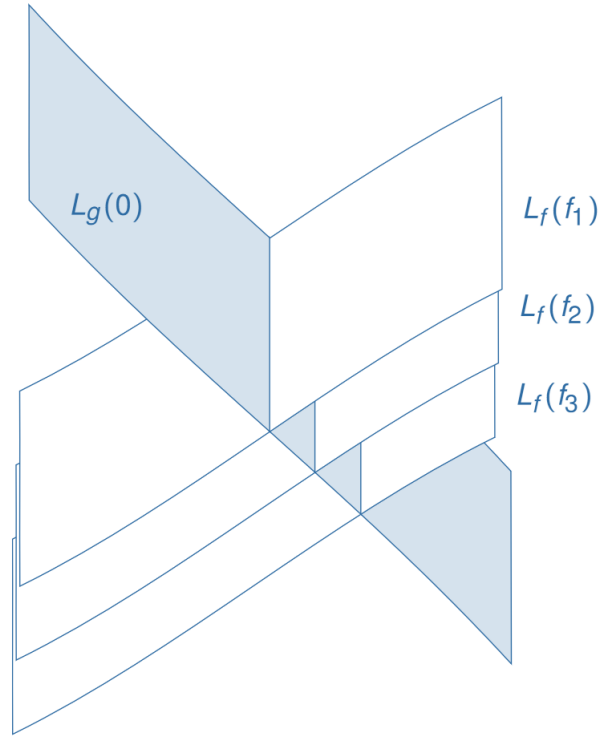
- Motion planning in $SE(3)$ for a manipulated object \mathcal{M} that cannot “move on its own”
- Can only move indirectly through interaction with robot \mathcal{R} or lie at rest on environment \mathcal{Z}



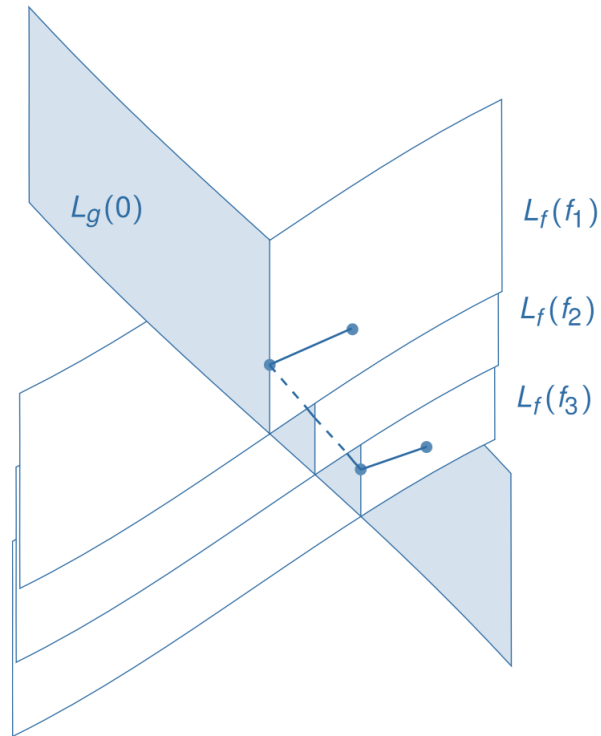
Stratification and foliation of the configuration space

- The configuration space $\mathcal{C} = \mathcal{C}_{\mathcal{R}} \times \mathcal{C}_{\mathcal{M}}$ is stratified in two strata:
 - Set of configurations in which the robot is grasping the object
 - Set of configurations in which the object is lying at rest
- Each stratum is foliated
 - the foliation of the grasp stratum is induced by the relative position of the robot and the grasp (infinitely many)
 - the foliation of rest stratum is induced by the position at which the object rest (infinitely many)
- The system can only move through a given leaf of a foliation, it cannot freely move across foliation of the same stratum
- The only way to move from one leaf to another in the same stratum is by going through a connecting leaf on the other stratum (respectively called transit paths and transfer paths)

Manipulation planning



Manipulation planning



Reduction property 2

- If we consider the intersection of the two strata endowed with both foliations (called the bottom stratum), then we can demonstrate that the existence of an admissible manipulation path between two configurations, that is, a finite sequence of transit and transfer paths, reduces to the existence a collision free path that ignores the foliation.
- Same kind of reasoning as for the nonholonomic system

Enter Humanoids (finally)

A path planning or a motion planning problem?

- Both problems are interesting on their own
- The frontier between the two notions was already “blurred” in the case of a mobile robot on the 2D plane: typically your vacuum cleaner robot
- Depends on the scale and level of the problem
- A humanoid wanting to go from point A to point B on Google map is an instance of a pure path planning problem
- At a finer grain a humanoid wanting to move its foot to do one step along the long Google maps path is an instance of a pure motion planning problem
- There are instances that lie between the two levels, we call them multi-contact motion planning problems

Definition of a humanoid robot

- $SE(3) \times \mathbb{R}^n$.

Definition of a humanoid robot

- That's a lot of dimensions to play with
 - $(6 + 6)$ legs + $(7+7)$ arm + 2 torso + 2 neck + 6 of $SE(3)$
 - Up to 46-d, without even accounting for dexterous fingers
- Stratified by each type of contact configuration on the space /
Foliated in each stratification
- And on top of that: The balance constraint which is a huge issue
 - probability of any random configuration or any random motion to make a manipulator arm or wheeled robot or multi-legged robot “fall down”: 0
 - probability of any random configuration or any random motion to make a bipedal humanoid robot fall down: 1
 - A humanoid is a inherently unstable system, it is an inverted pendulum always on its unstable equilibrium

Approaches to studying humanoids and make them move

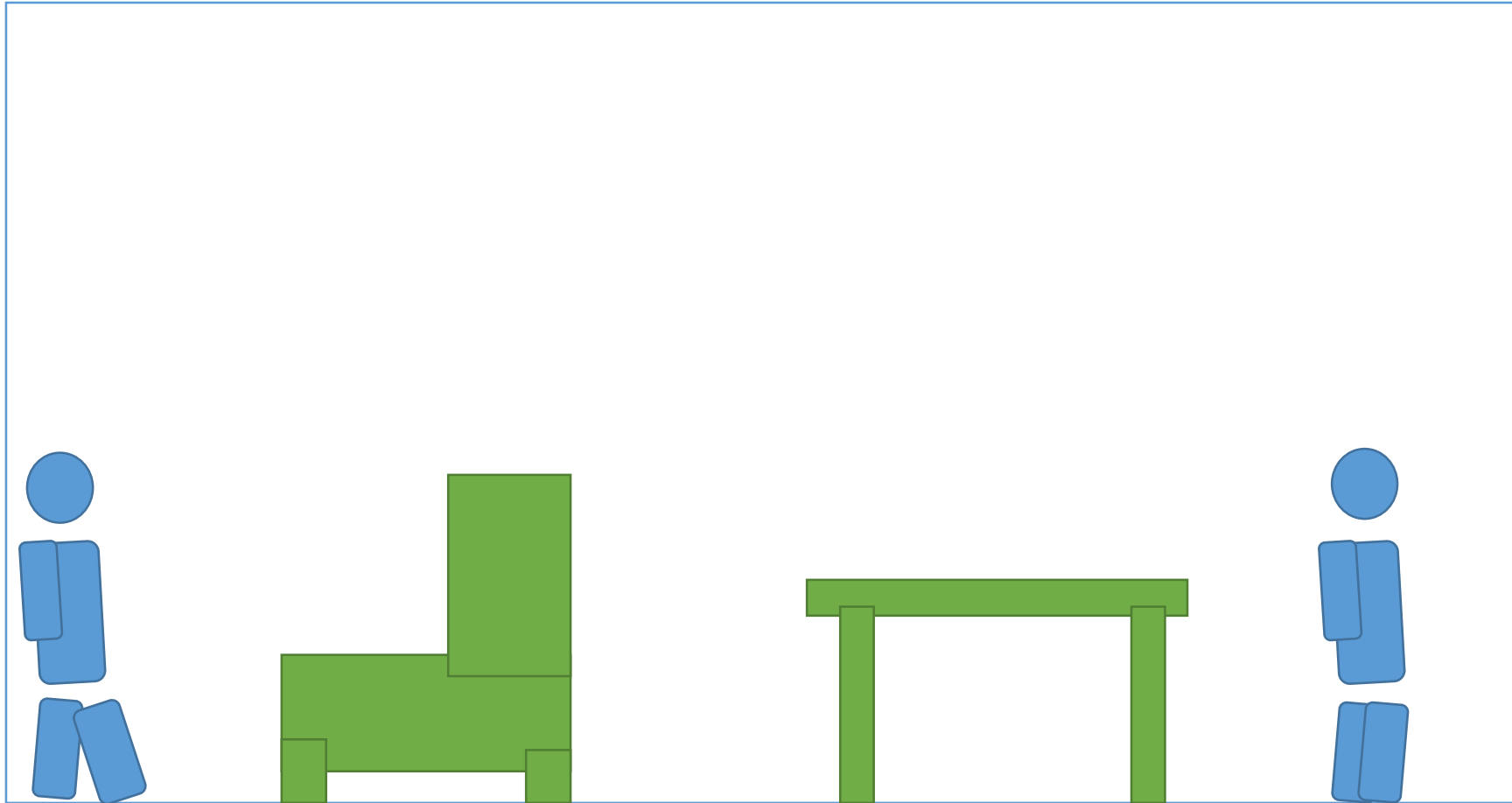
- Decoupling : upper body for manipulation / lower body for locomotion and navigation; global planner / local planner
- Interleaving : motion planning and motion control
- Lower body controllers
 - aka walking controllers
 - Based on the physics of inverted pendulums (aka Zero Moment Point (ZMP) or Center of Pressure (COP) control, Capture point control) Kajita et al 2003
 - used on Honda's Asimo
 - “Stabilizers” are lower level control loop that take in charge balance one or two feet
 - We can usually perform path-planning with these controllers
- Upper-body controllers
 - Don't care about balance, leave it to the stabilizer
 - See robot as fancy dual arm manipulator (Pepper-like robot)

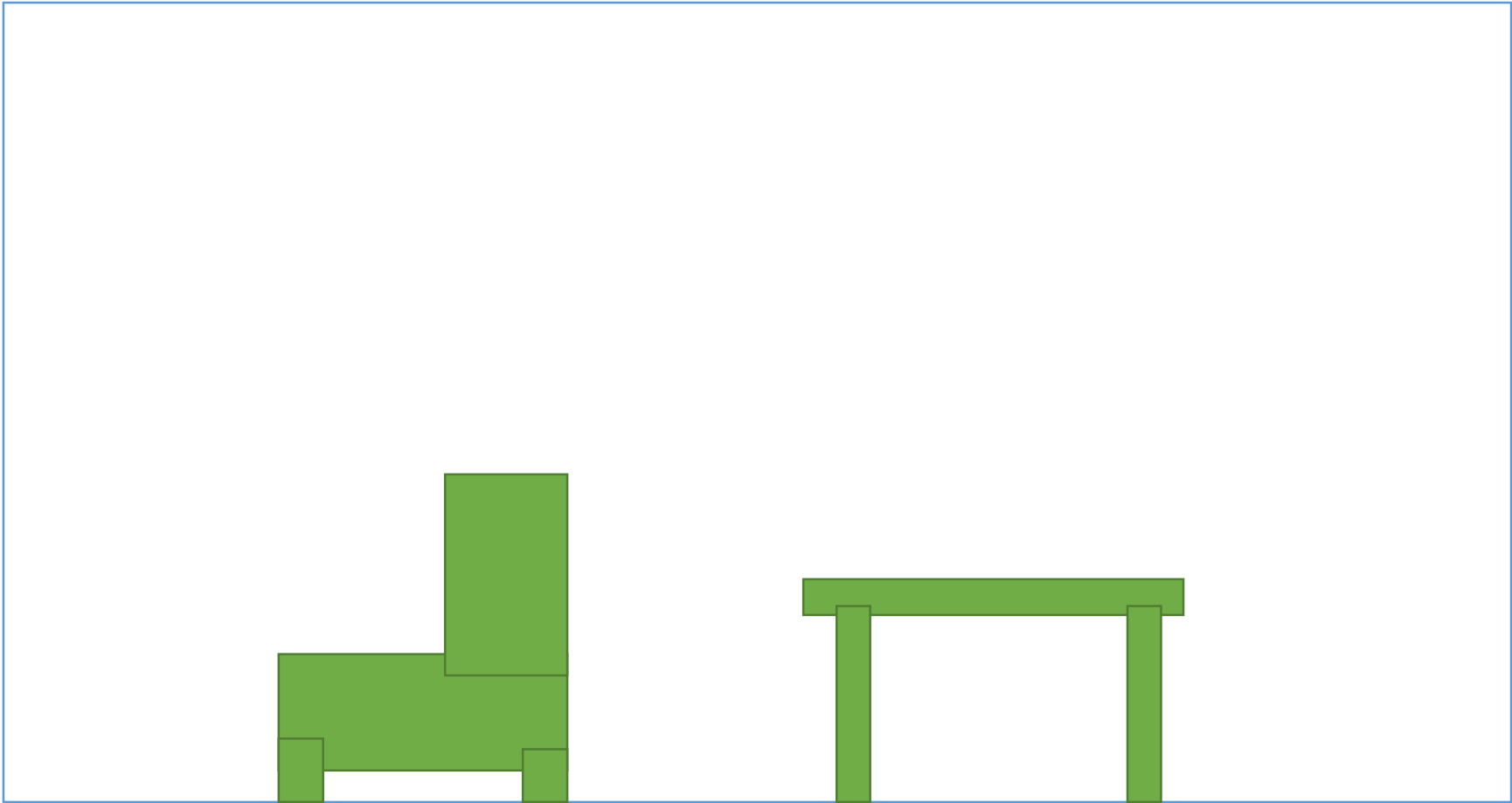
Approaches to studying humanoids and make them move

- Our approach: multi-contact planning and whole-body control
 - See the robot as whole
 - Can use hands for crawling (marcher à 4 pattes), for climbing ladders, staircases, for walking on hands (why not), can use legs for manipulation (pushing object lying on the floor, kicking a ball)
 - It's just a robot that lives in its 46-d C-space and happens to have a humanoid shape, no functional decoupling
 - Our philosophy: walking should “emerge” naturally from this approach, not be hard-encoded as a separate control
- Of course, only an ideal objective, as of now, we still encode domain-knowledge based heuristics to prune search trees (such as “prefer feet on ground”, “prefer hands on table”, etc)

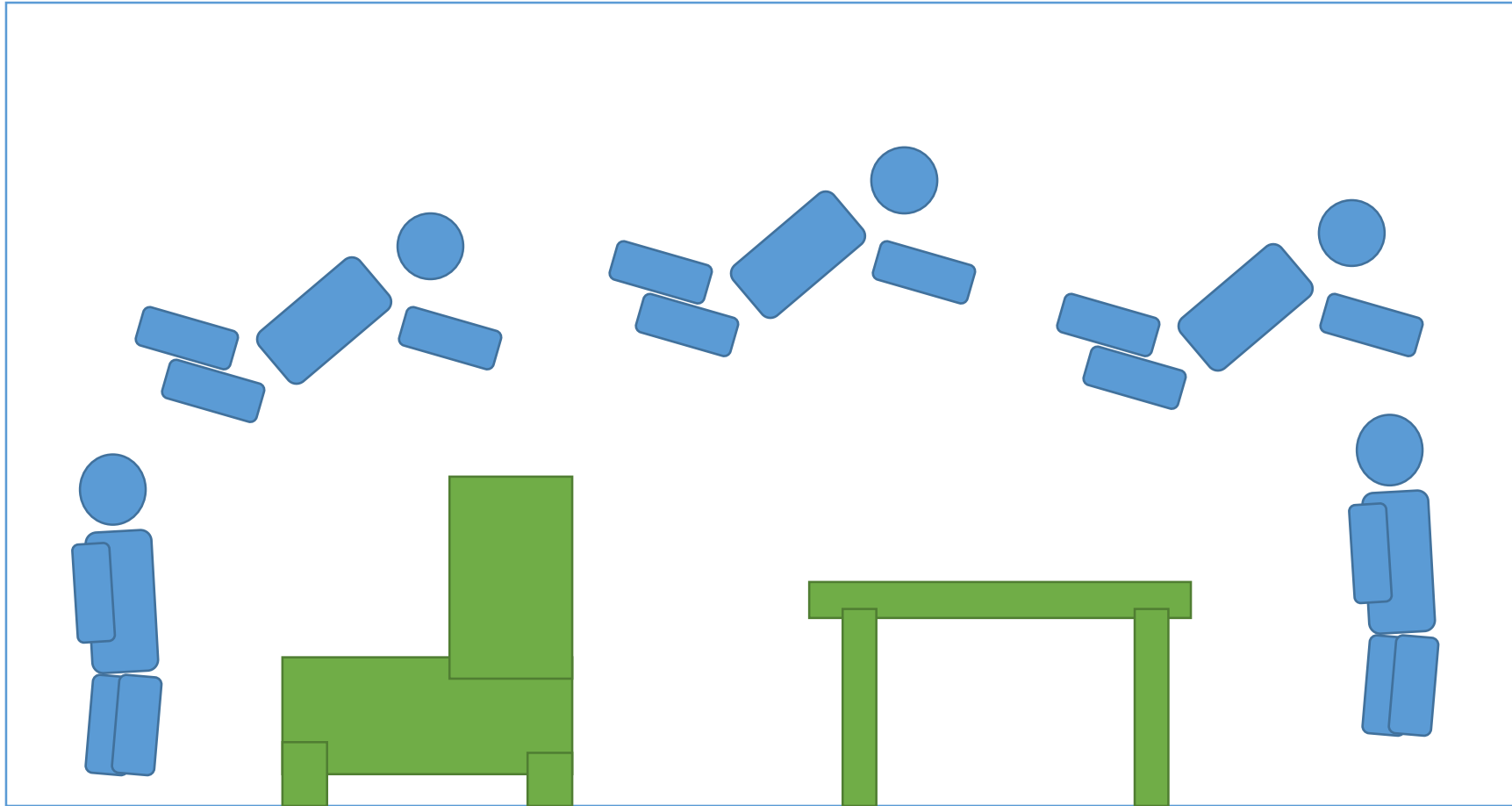
Humanoid multi-contact planning

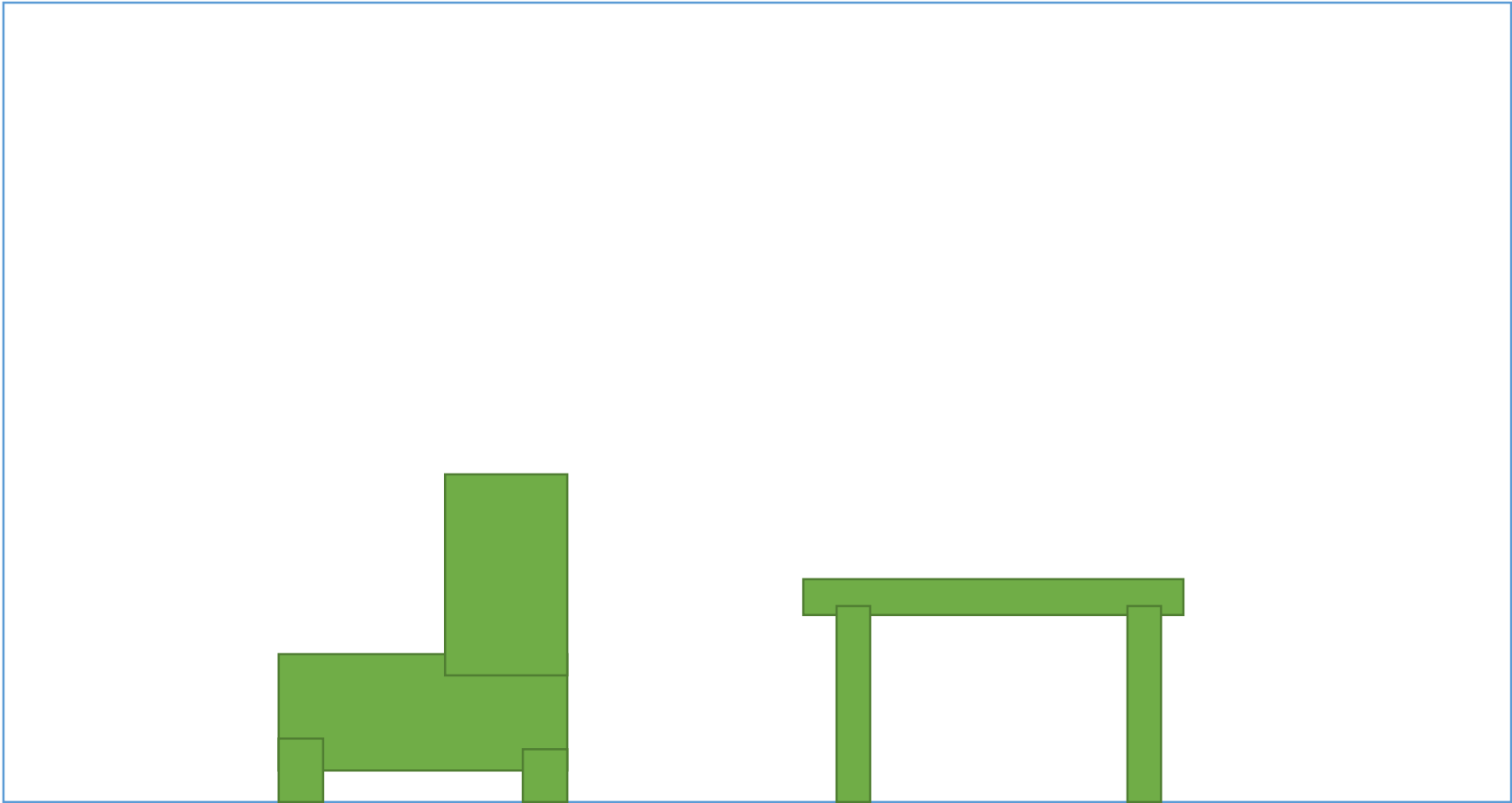
A multi-contact planning instance



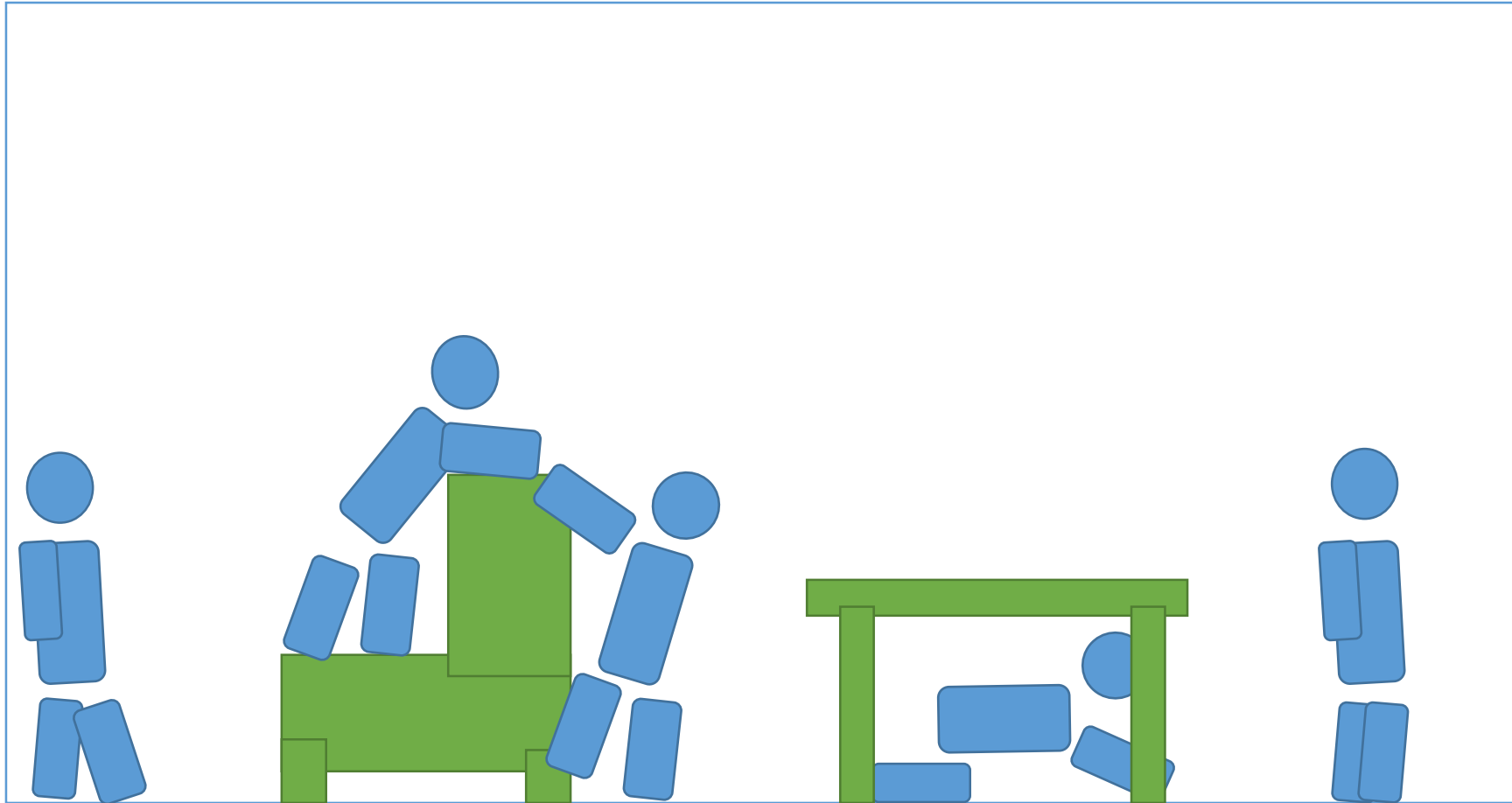


Collision-free path planning/motion planning

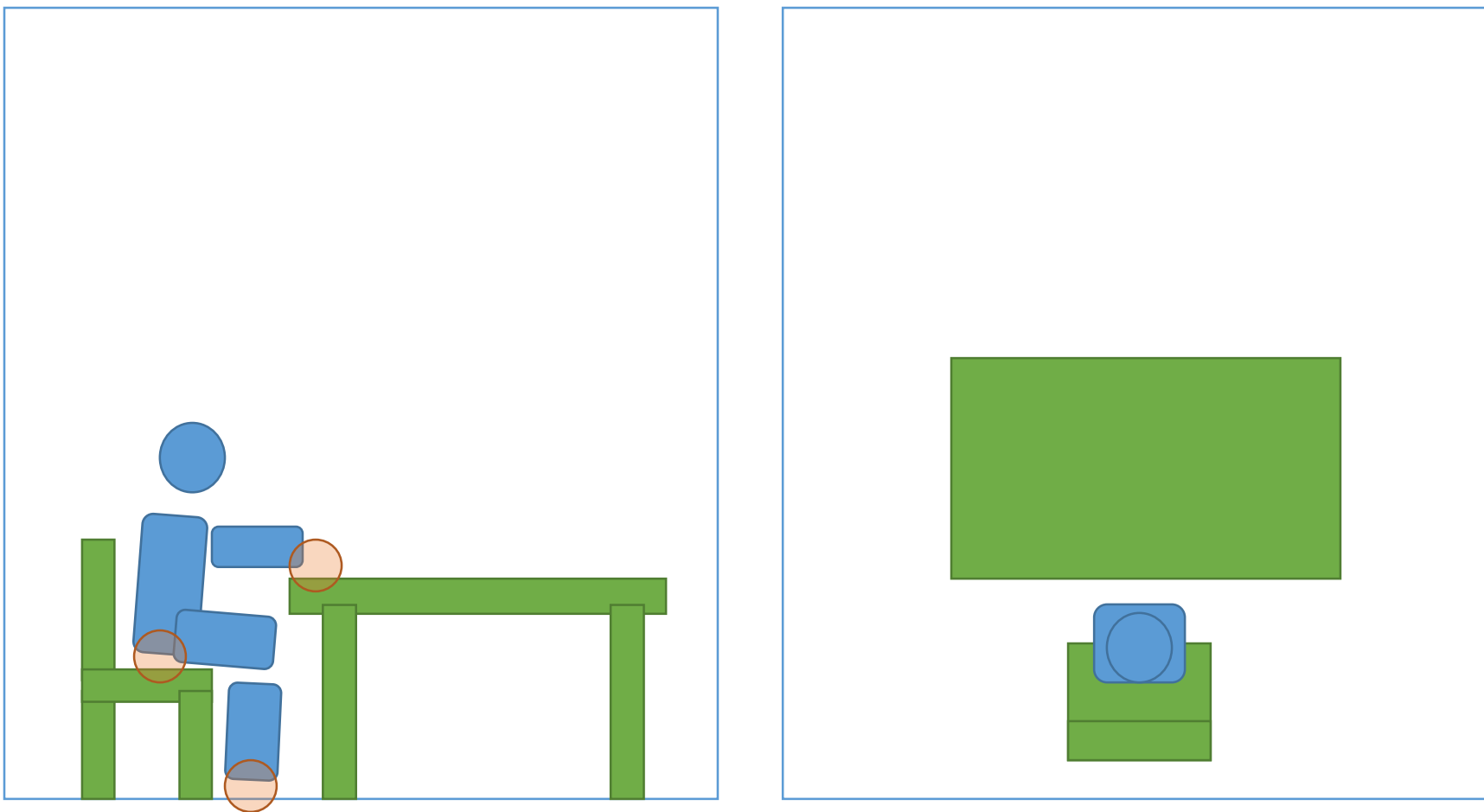




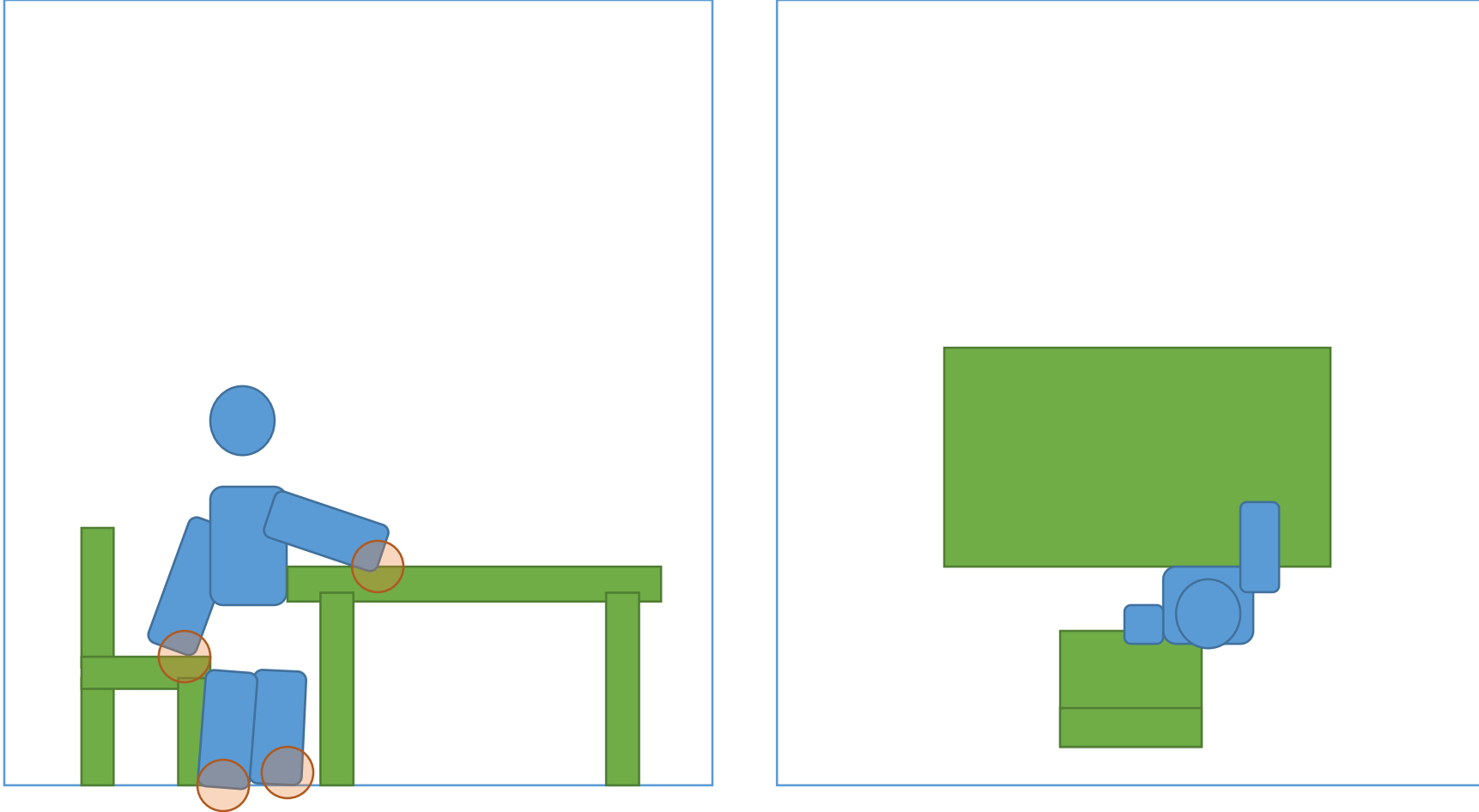
Multi-contact planning



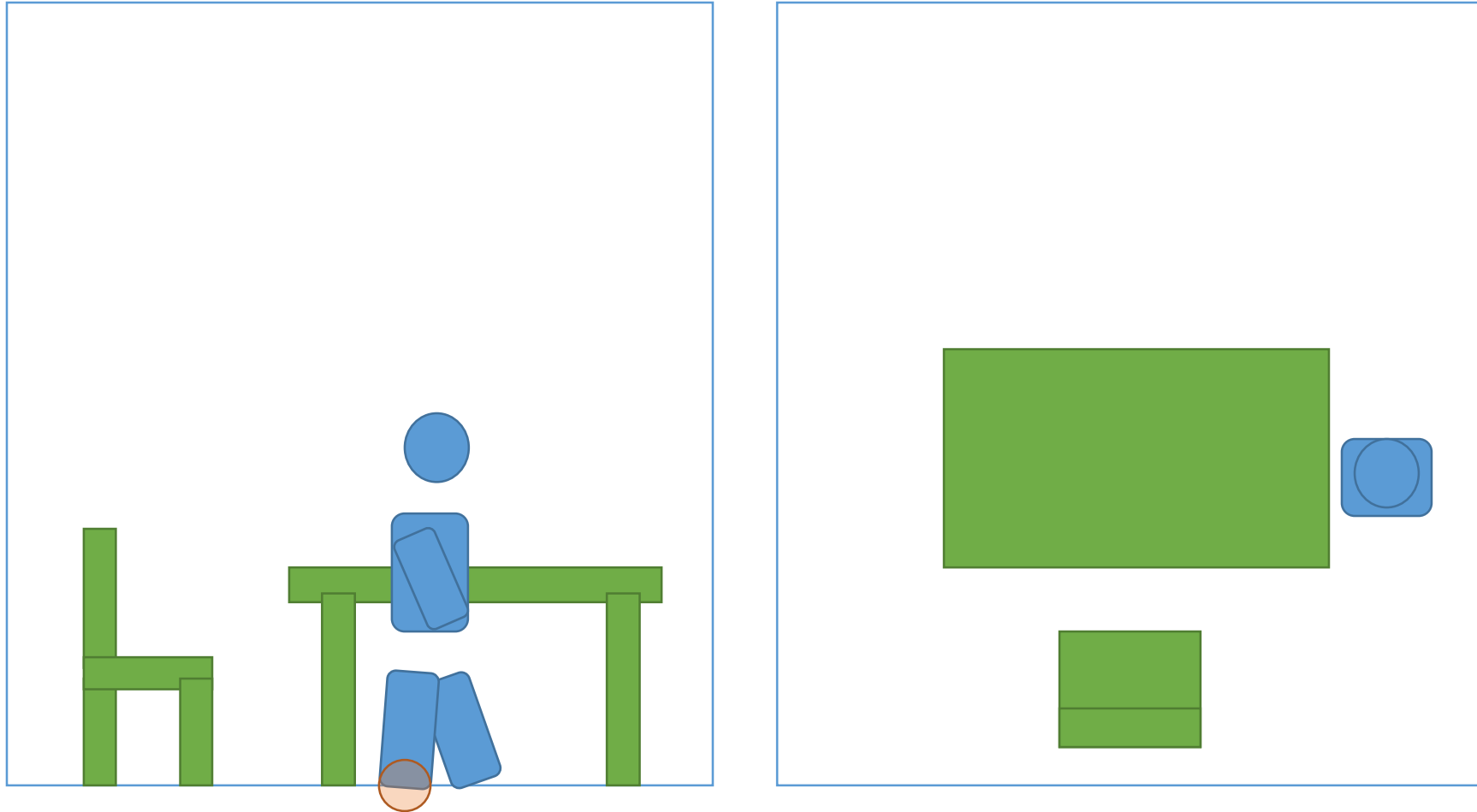
Multi-contact planning w/ path planning



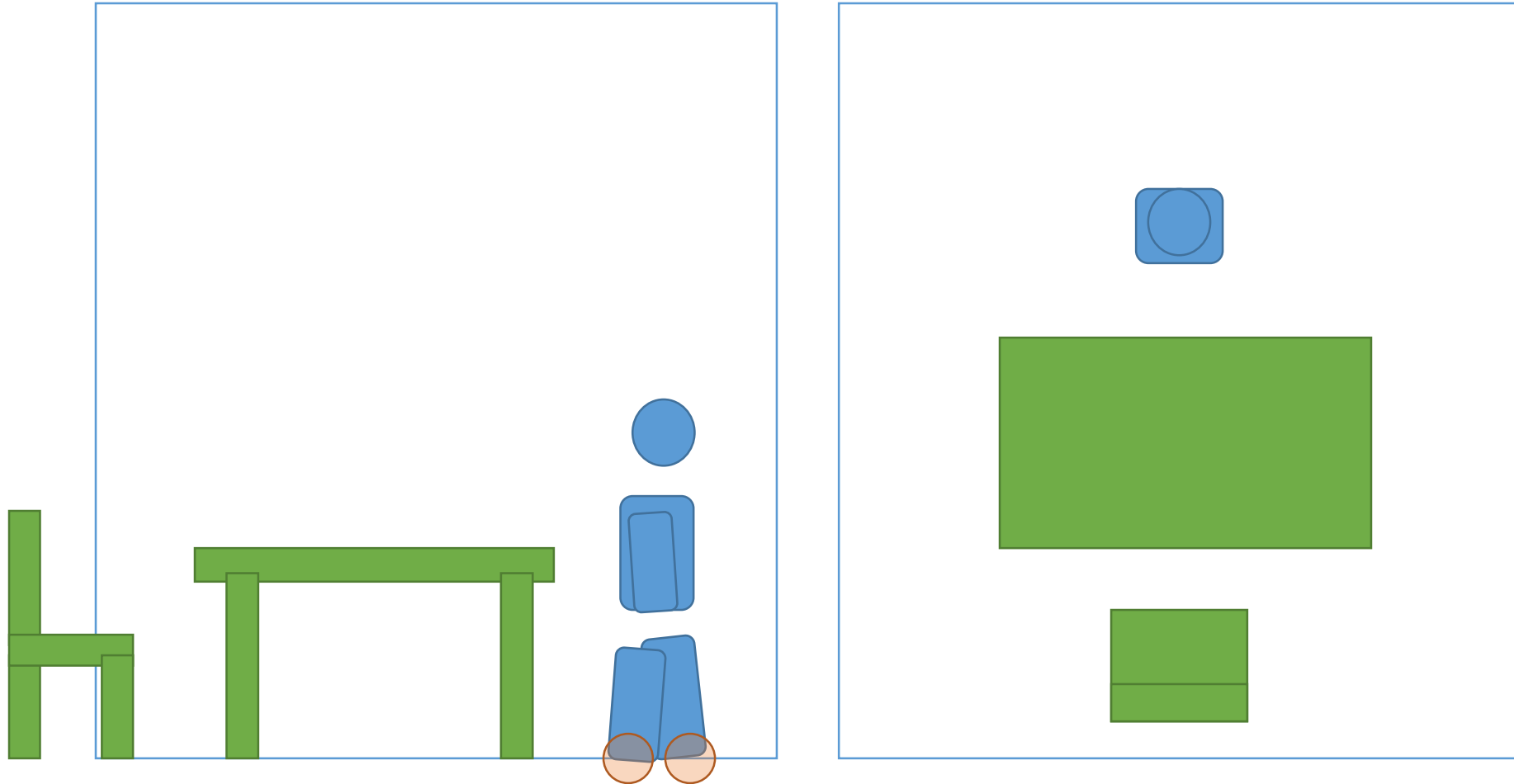
Multi-contact planning w/ path planning



Multi-contact planning w/ path planning



Multi-contact planning w/ path planning



Humanoid multi-contact planning

- Induces the same stratification / foliation structure as the manipulation planning problem
- Each leaf of a stratum corresponds to a stance σ
- a stance σ is a set of contacts
- a contact c is defined as an element of $E = \mathbb{N}^4 \times SE(2)$
- the set of all stances is denoted $\Sigma \subset 2^E$
- Two stances σ and σ' are adjacent if they differ by exactly one contact
 $\exists c \in E \ \sigma = \sigma' \cup \{c\}$ or $\sigma' = \sigma \cup \{c\}$

Humanoid multi-contact planning

- each configuration of the robot q is mapped to a unique stance σ through a forward kinematics function $\sigma = \gamma(q)$
- conversely with each sigma is associated a submanifold of the C-space $Q_\sigma = \gamma^{-1}(\{\sigma\})$
- We are interested in a subset $\mathcal{F}_\sigma \subset Q_\sigma$ made of physically admissible configurations (existence of admissible contact forces at the contacts of the stance)
- An admissible sequence of stances (“path” in Σ) is a sequence $(\sigma_i)_{i \in \{1, \dots, k\}}$ such that $\forall i \in \{1, \dots, k-1\}$ σ_i is adjacent to σ_{i+1} and $\mathcal{F}_{\sigma_i} \cap \mathcal{F}_{\sigma_{i+1}} \neq \emptyset$

Humanoid motion planning and control framework

Overview of the Framework

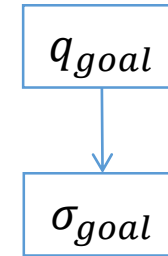
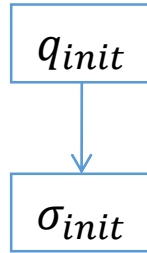
Locomotion query: q_{init}, q_{goal}

q_{init}

q_{goal}

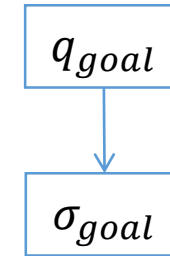
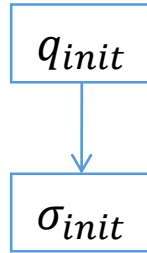
Overview of the Framework

Locomotion query: q_{init}, q_{goal}



Overview of the Framework

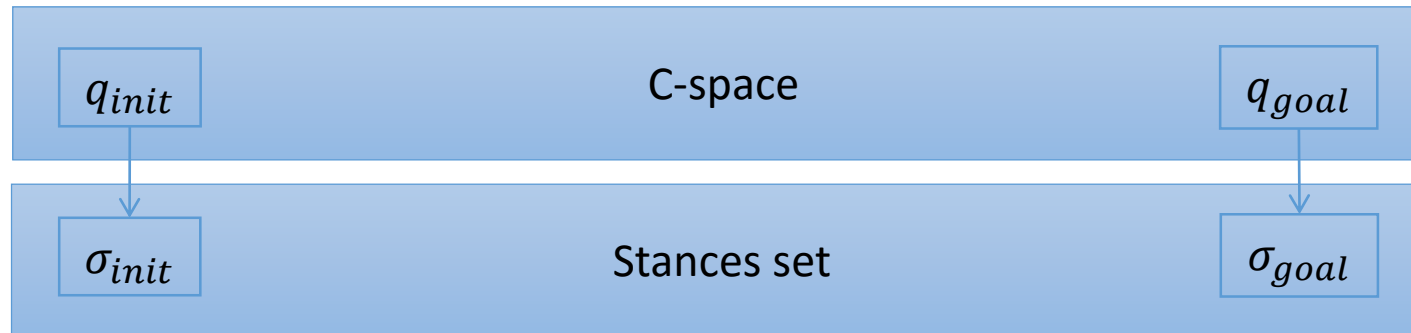
Locomotion query: q_{init}, q_{goal}



a stance σ = a set of contacts = a contact state

Overview of the Framework

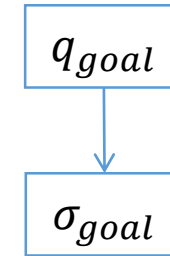
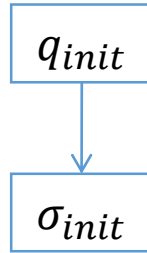
Locomotion query: q_{init}, q_{goal}



a stance σ = a set of contacts = a contact state

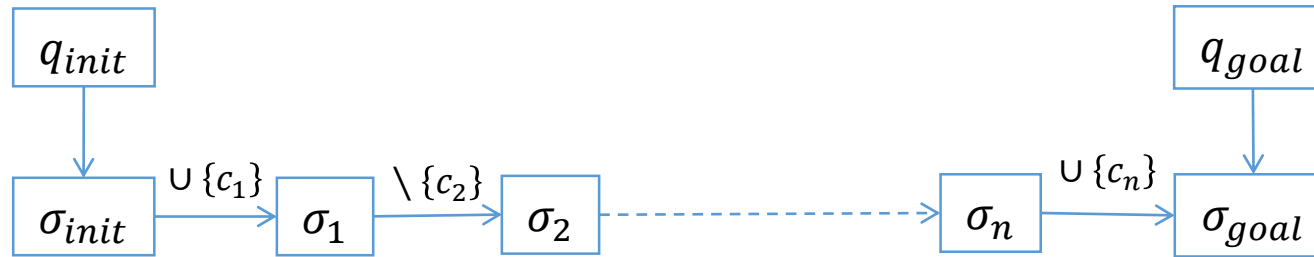
Overview of the Framework

Locomotion query: q_{init}, q_{goal}



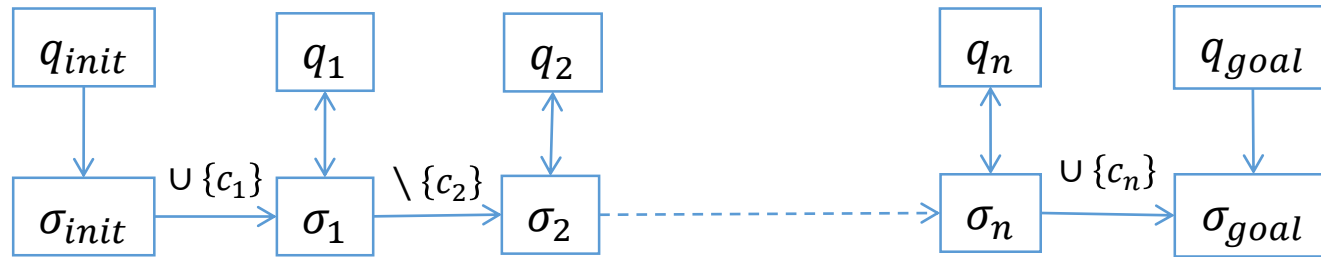
a stance σ = a set of contacts = a contact state

Overview of the Framework



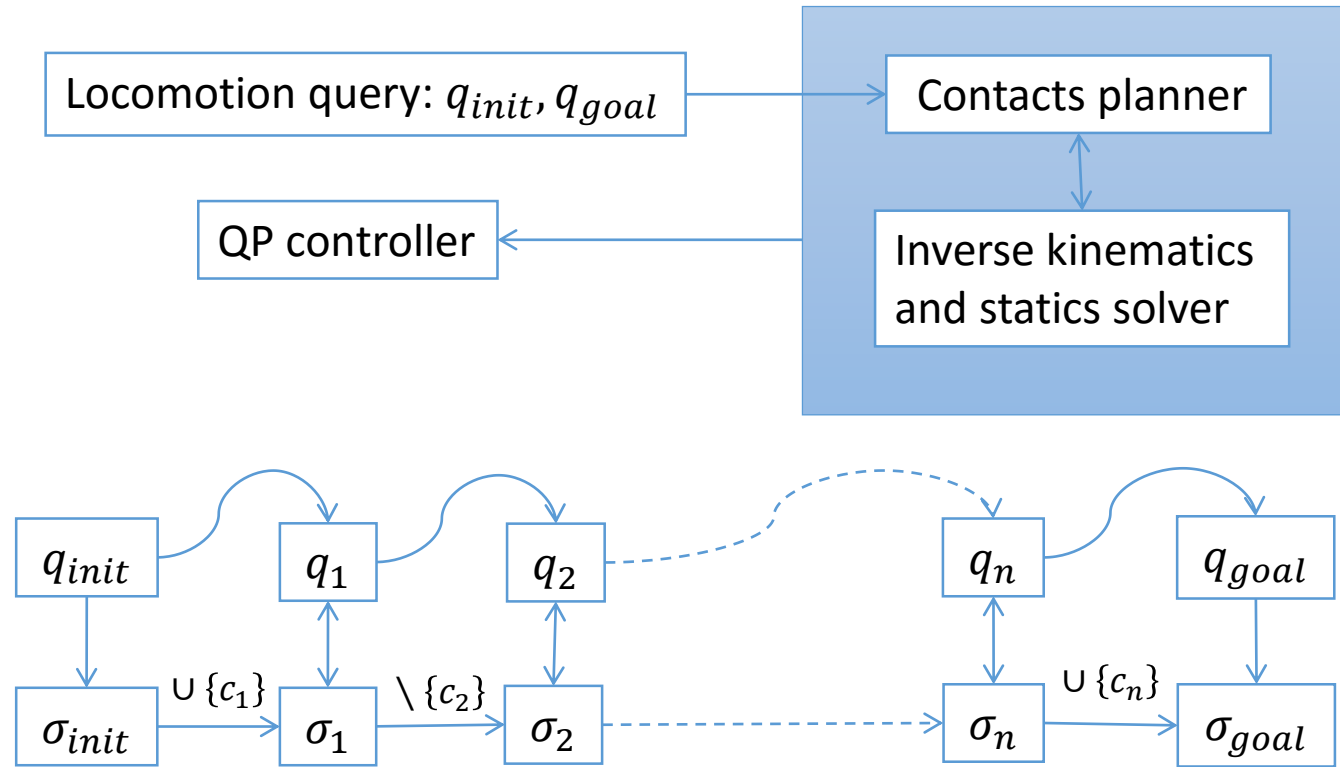
a stance σ = a set of contacts = a contact state

Overview of the Framework



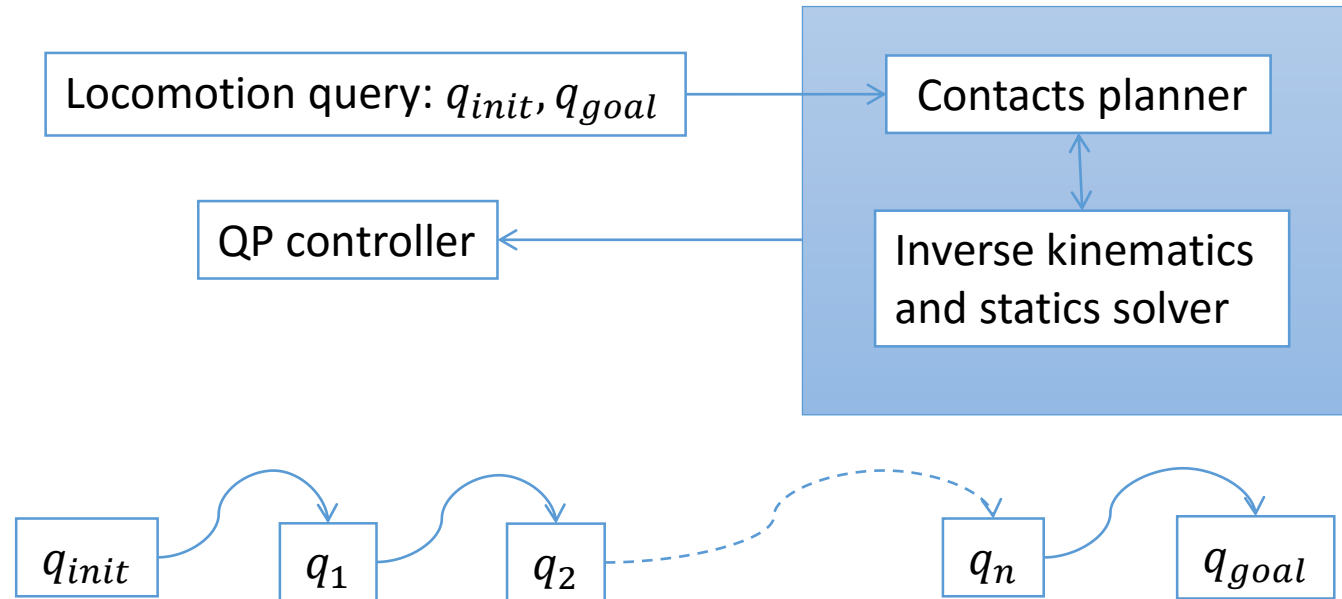
a stance σ = a set of contacts = a contact state

Overview of the Framework

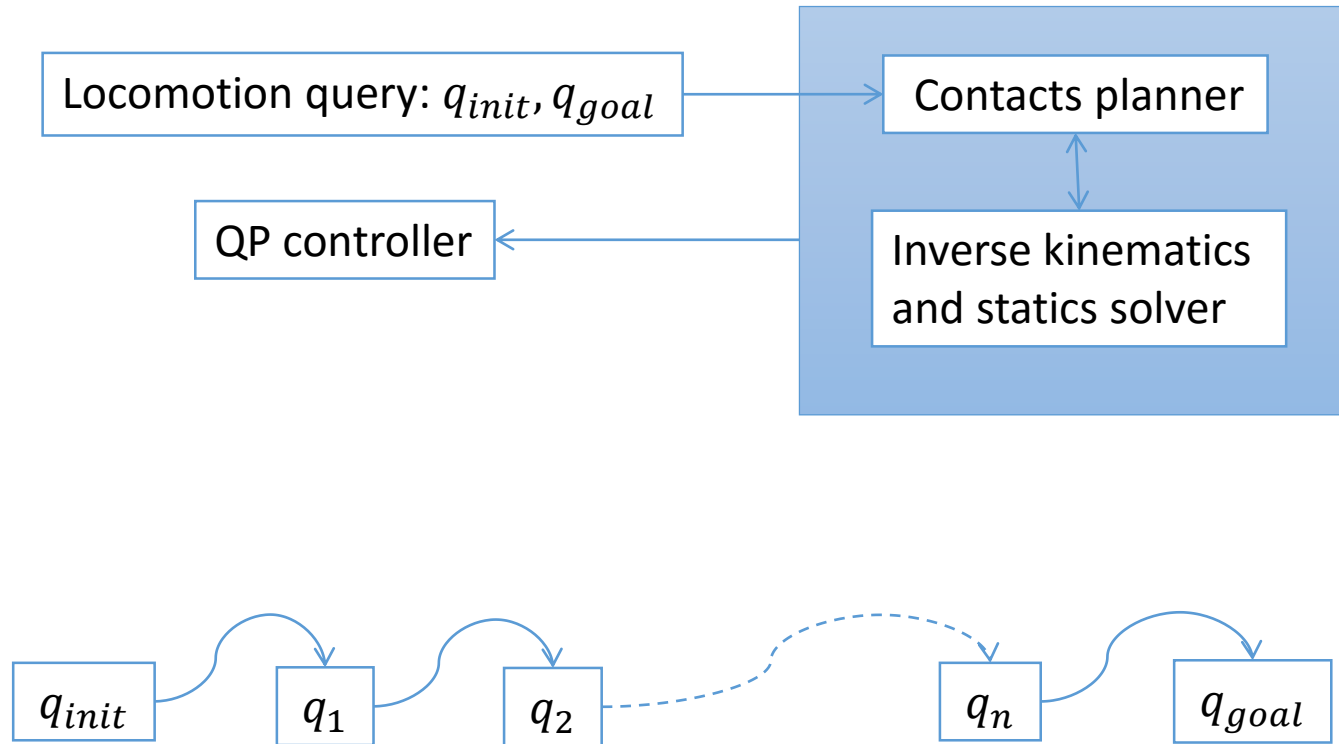


a stance σ = a set of contacts = a contact state

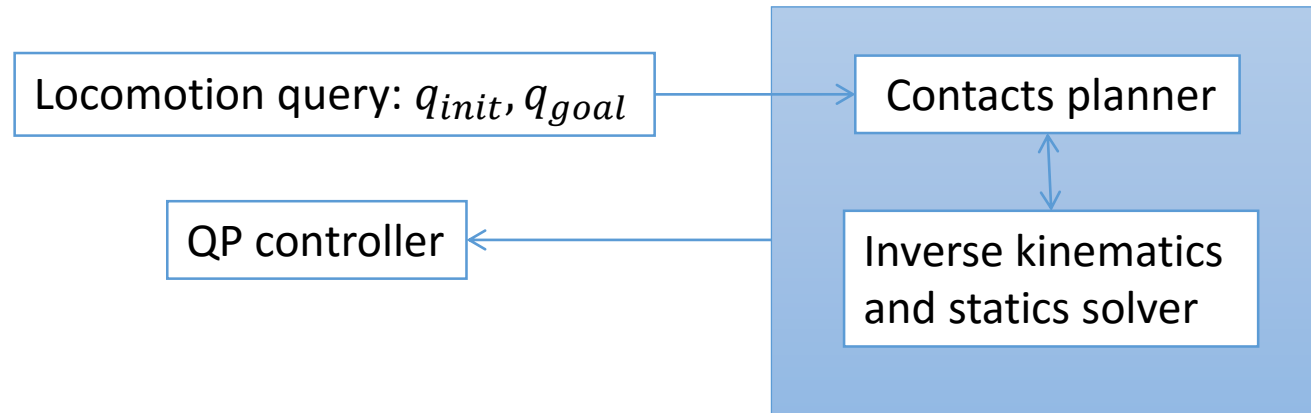
Overview of the Framework



Overview of the Framework



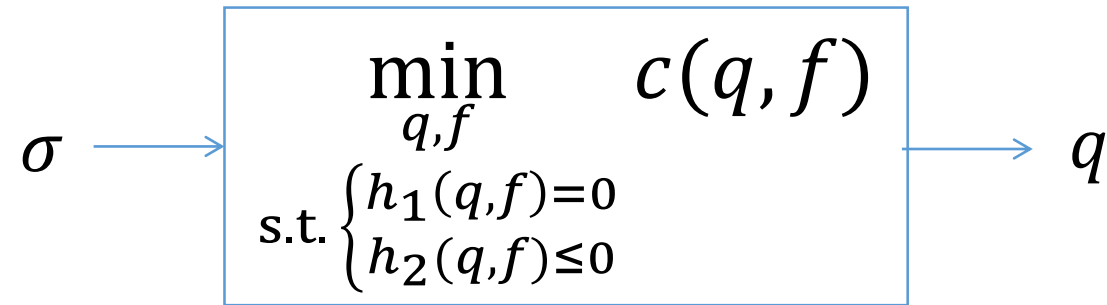
Overview of the Framework



Overview of the Framework

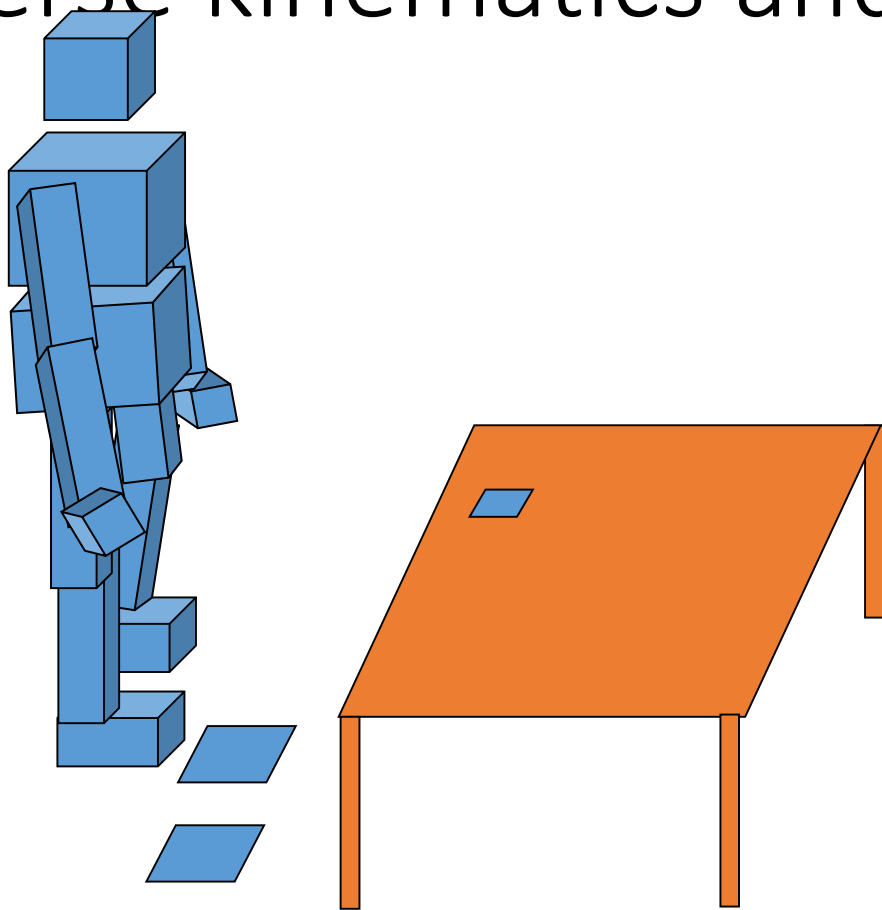
Inverse kinematics
and statics solver

Inverse kinematics and statics solver

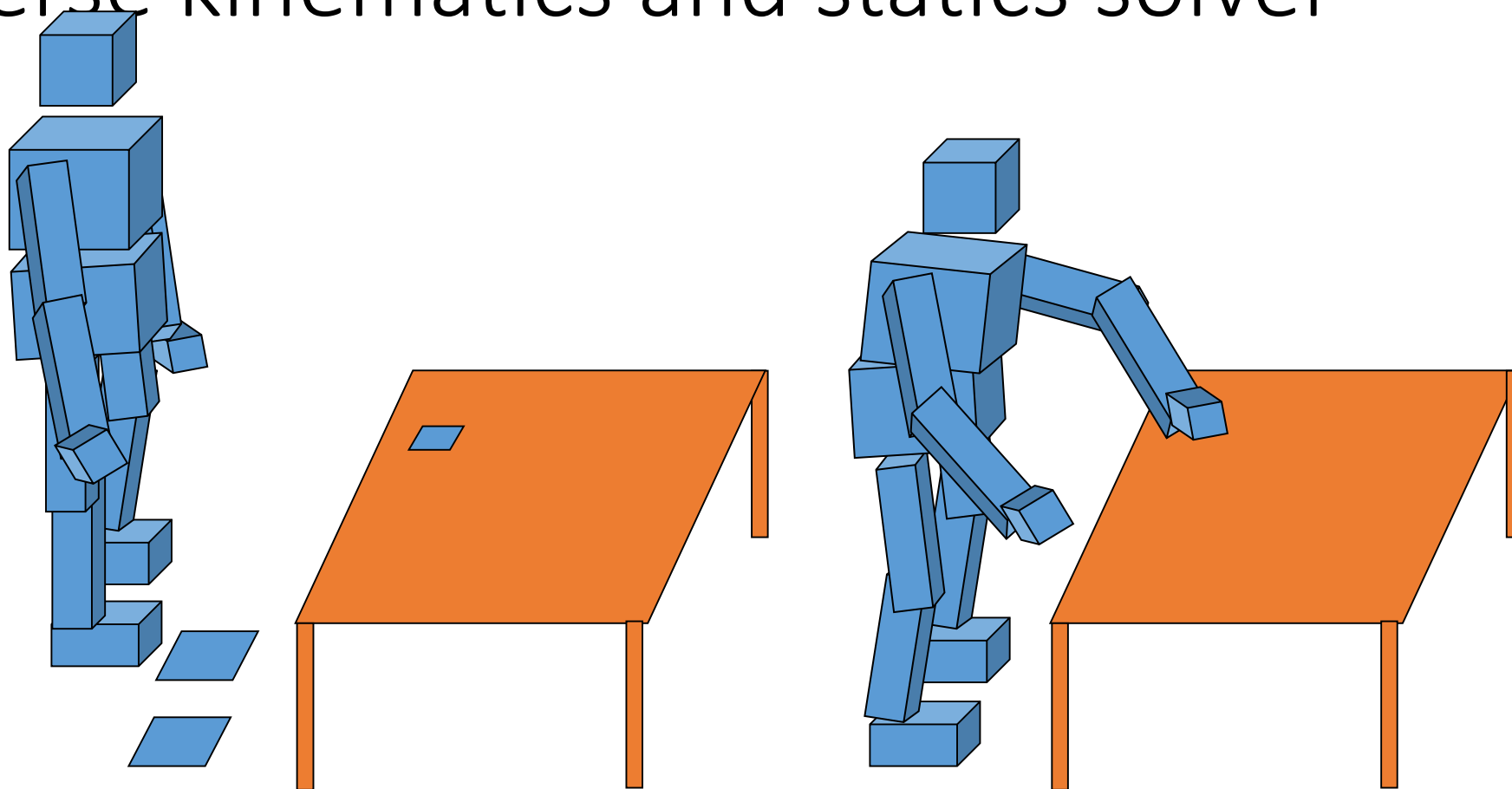


- $c(q, f)$ distance of q to a reference nominal posture q_{ref} + norm of f
- $h_1(q, f) = 0$ fixed contacts positions, static equilibrium equation (free-flyer part)
- $h_2(q, f) \leq 0$ floating contacts positions, joint limits, static equilibrium equation (actuated part, torque limits), friction cones, collision-avoidance

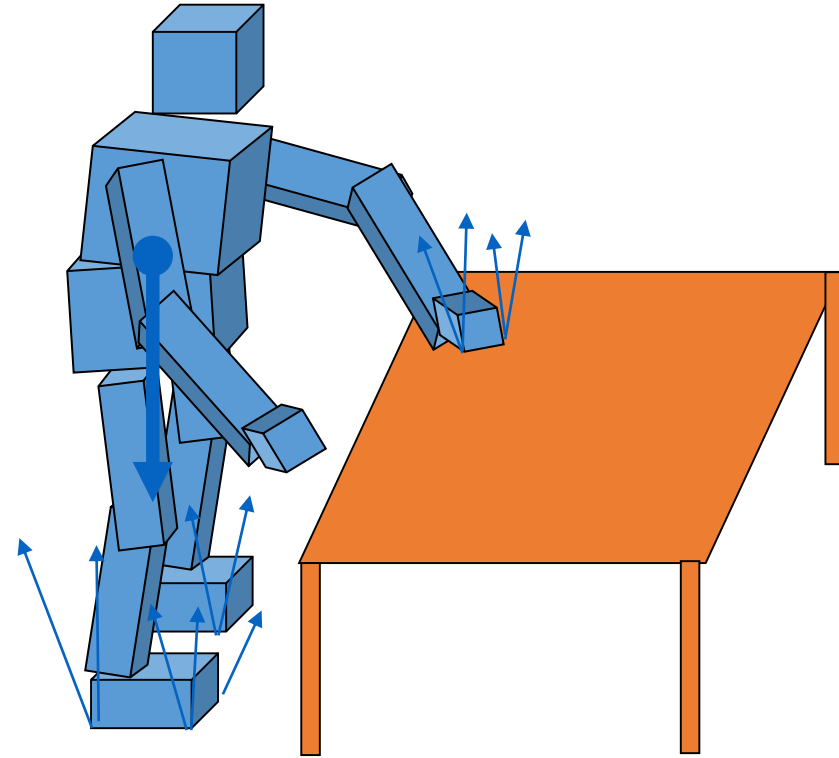
Inverse kinematics and statics solver



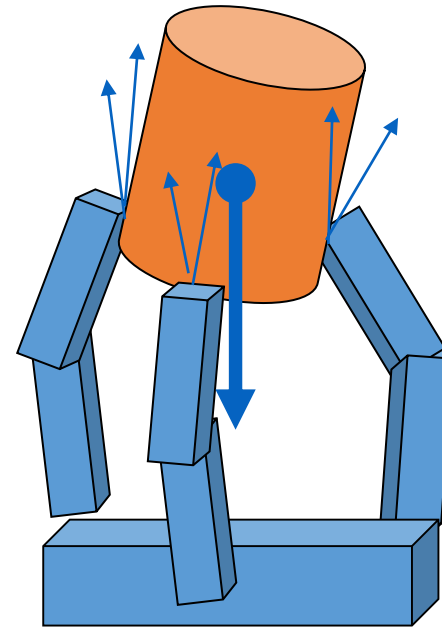
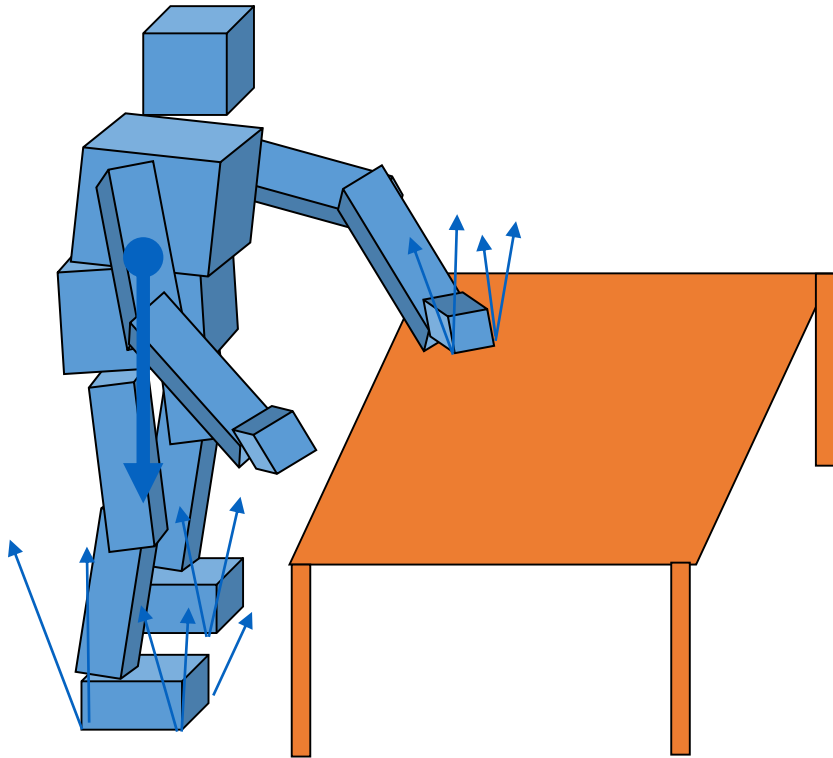
Inverse kinematics and statics solver



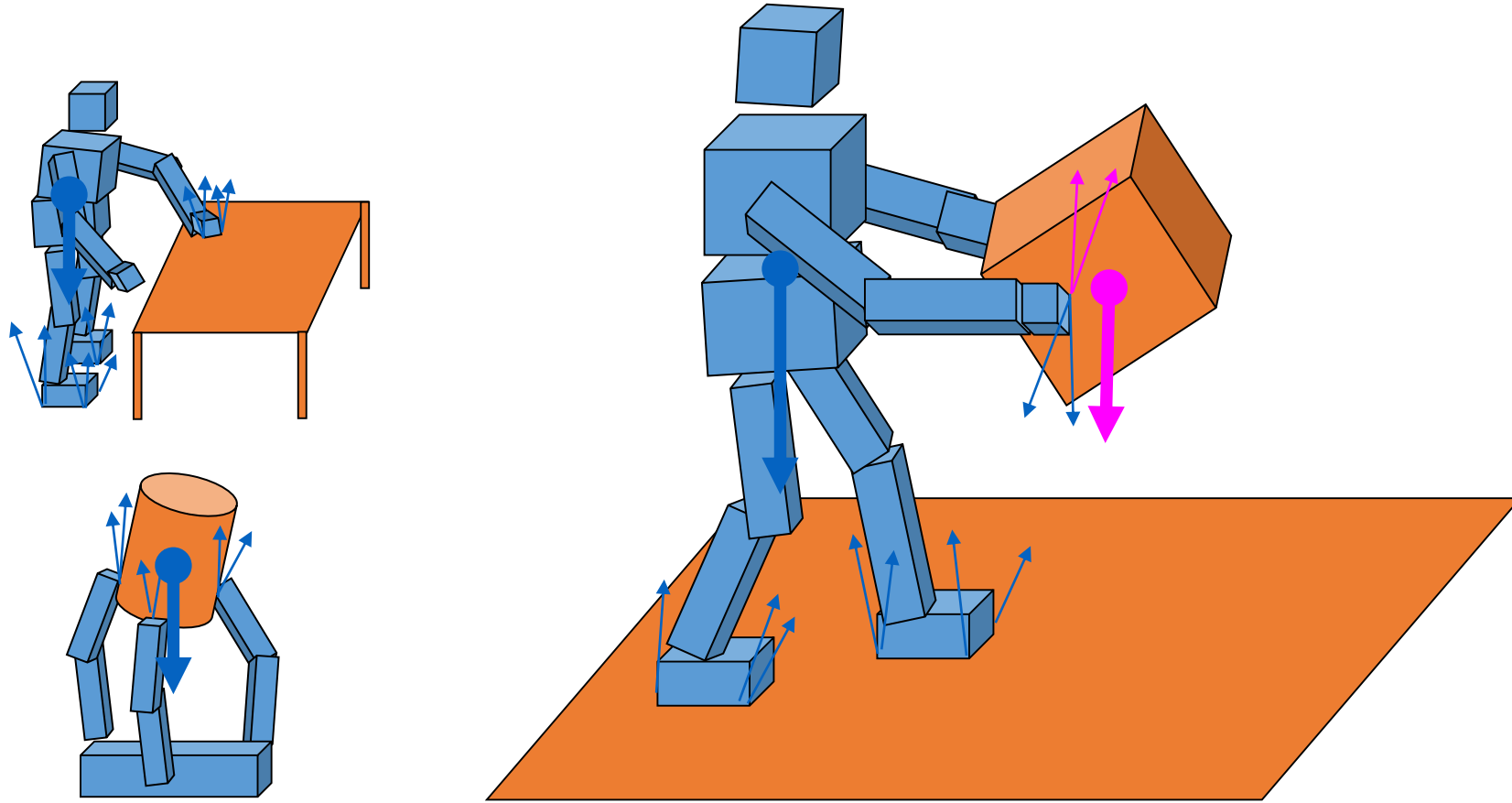
Inverse kinematics and statics solver



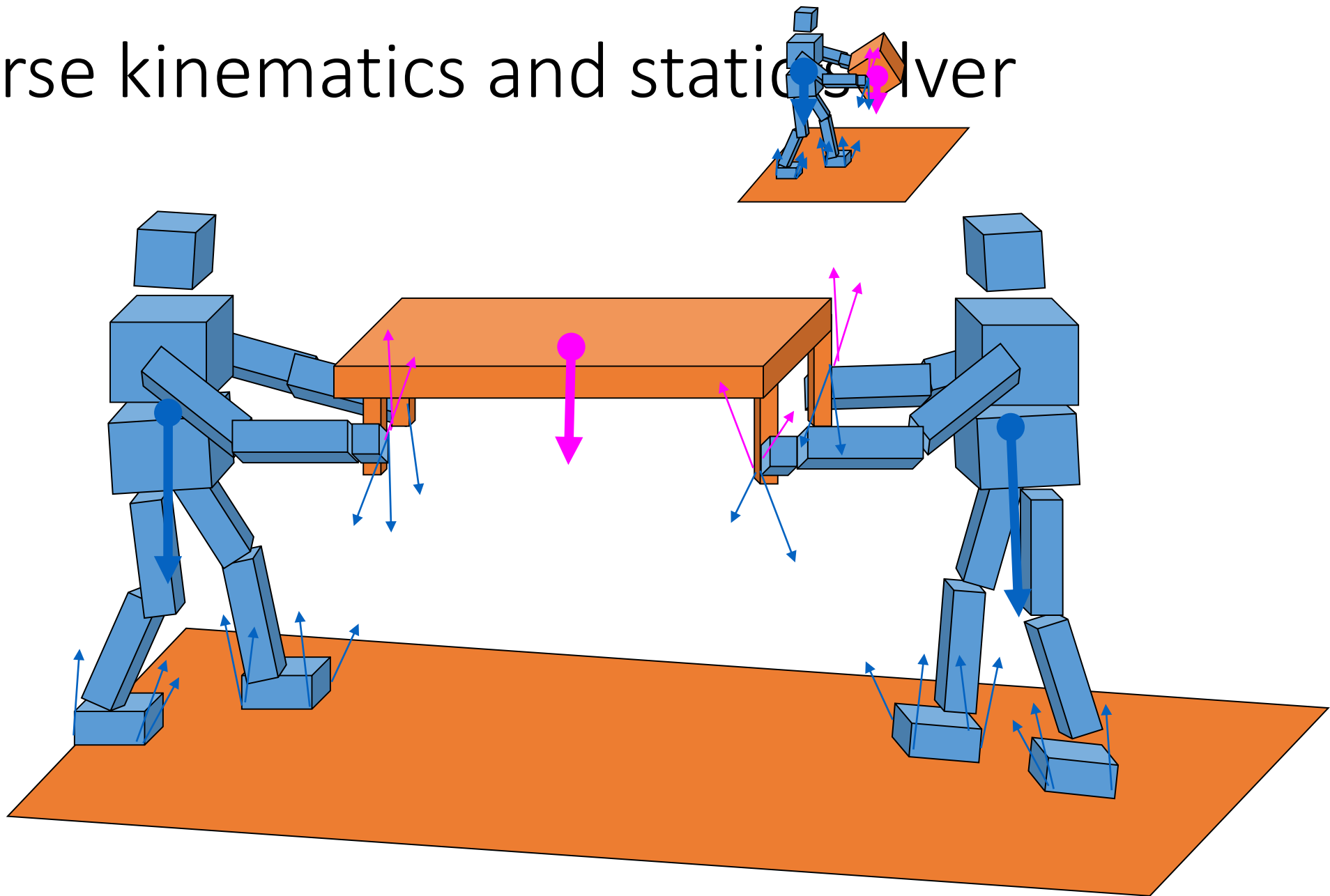
Inverse kinematics and statics solver



Inverse kinematics and static solver

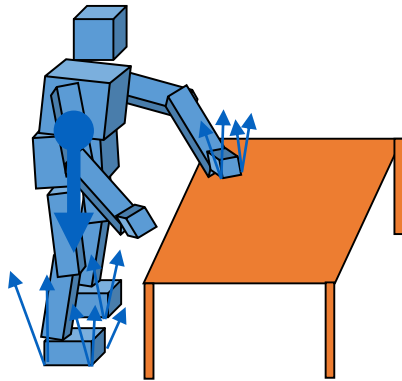
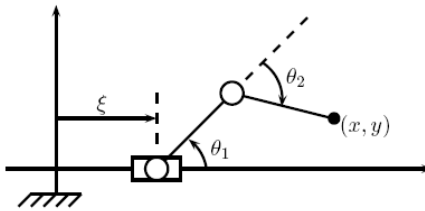


Inverse kinematics and statics solver

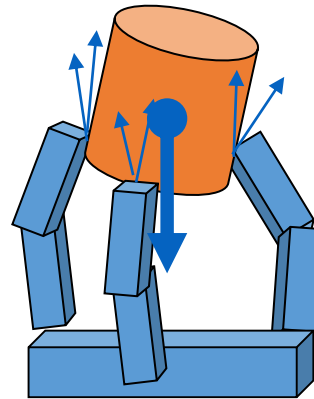
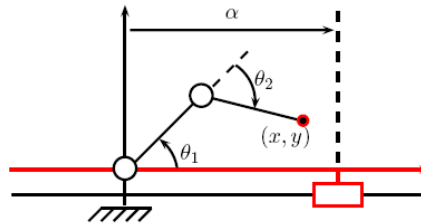


Inverse kinematics and static solver

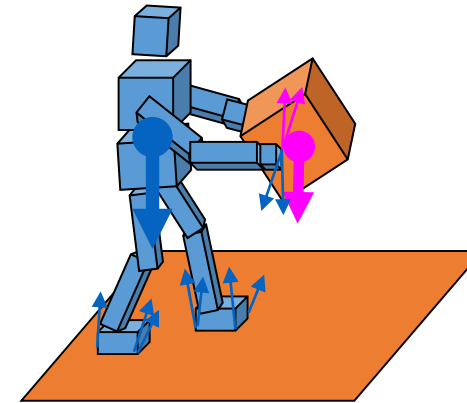
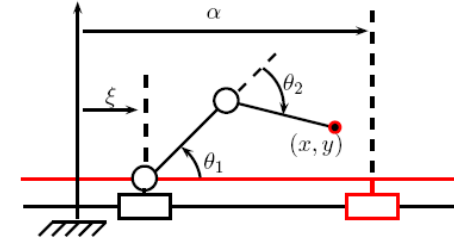
Locomotion



Manipulation



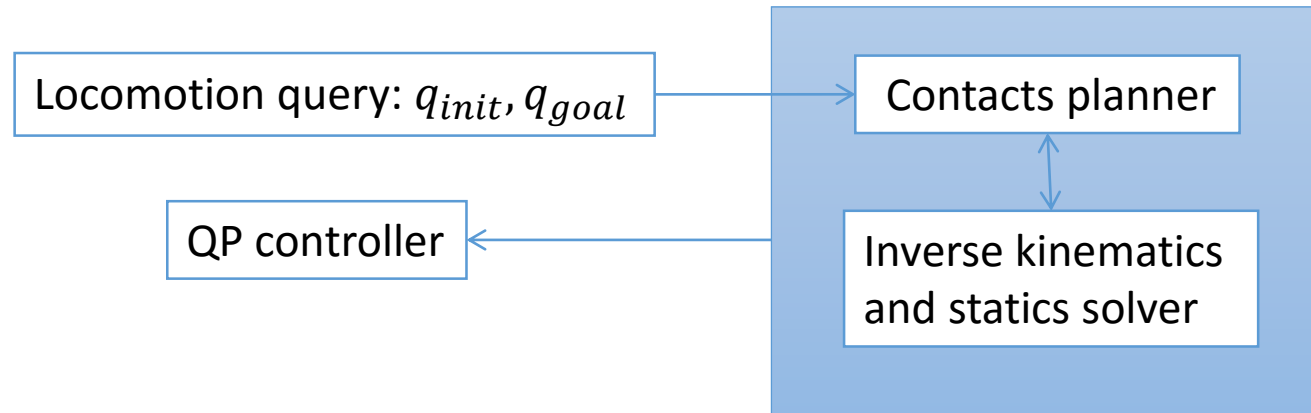
Locomotion-and-Manipulation



Inverse kinematics and statics solver



Overview of the Framework



Overview of the Framework

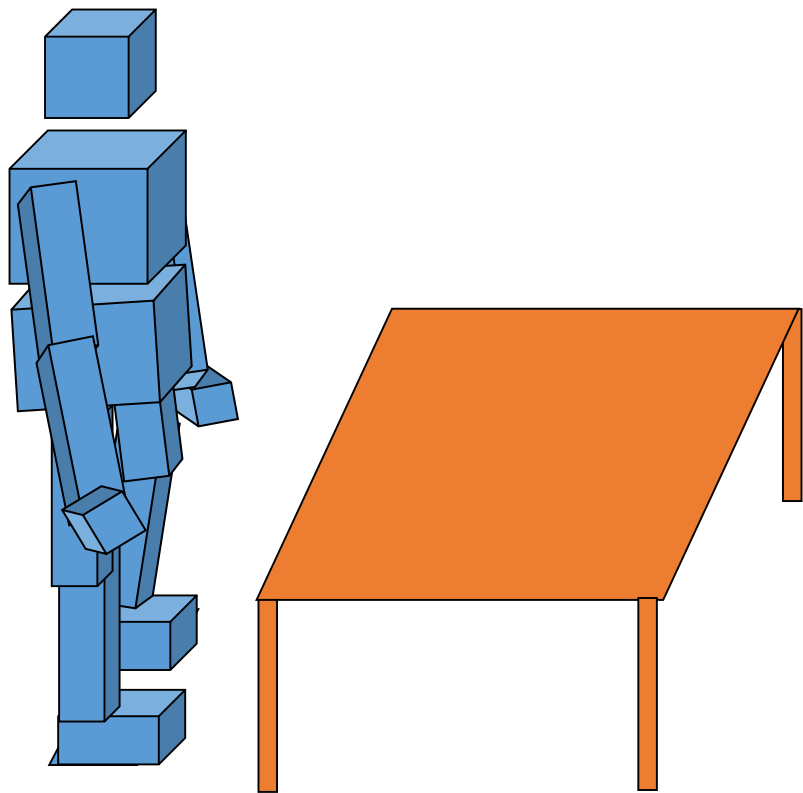
Contacts planner

Search Algorithm

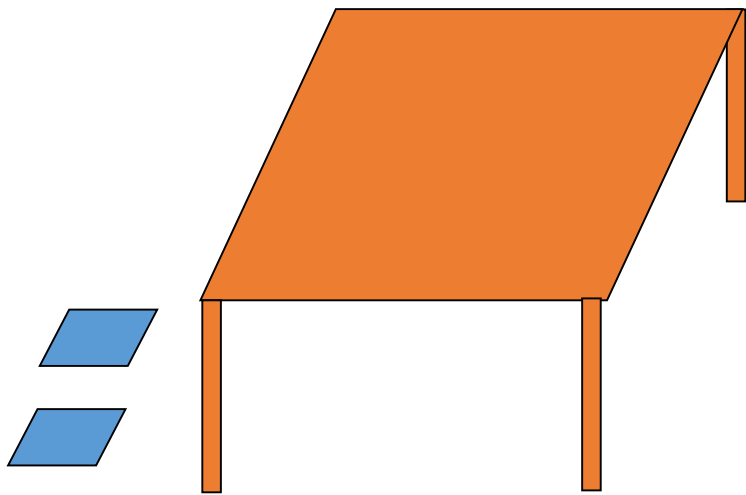
Best-First Search

- Initialize a search tree and a priority queue with the initial stance
- Loop:
 - Pop most promising stance from the priority queue
 - For all present contacts in this stance
 - Try to remove the contact and see if the robot keeps balance (call the IKS)
 - If so, push the new stance into the priority queue and add it as a leaf in the tree
 - For all unused bodies in this stance
 - Try to add the unused body to the stance and see if the robot can reach balance (call the IKS)
 - If so, push the new stance into the priority queue and add it as a leaf in the tree
- Until reaching the goal

Search Algorithm

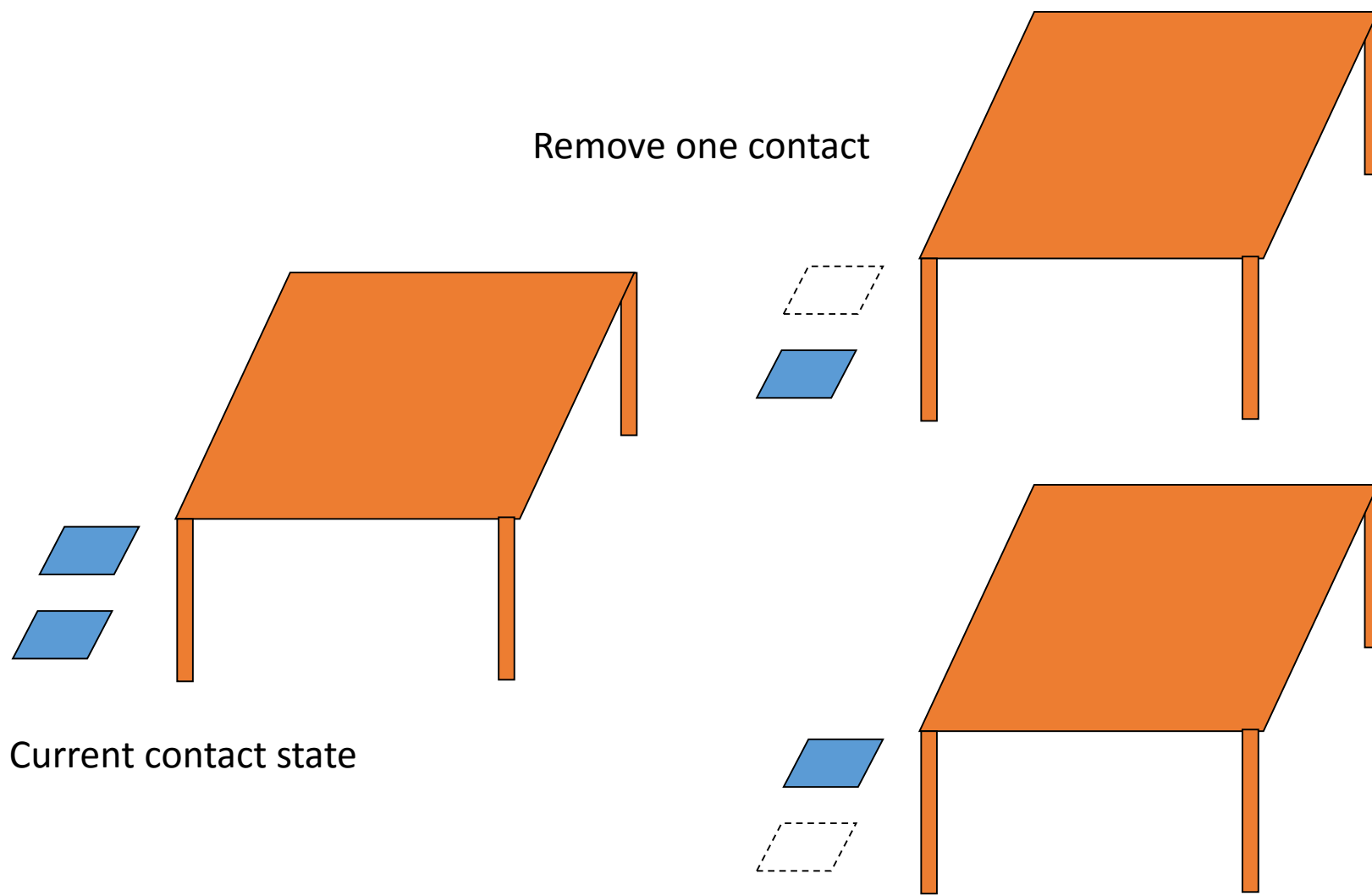


Search Algorithm

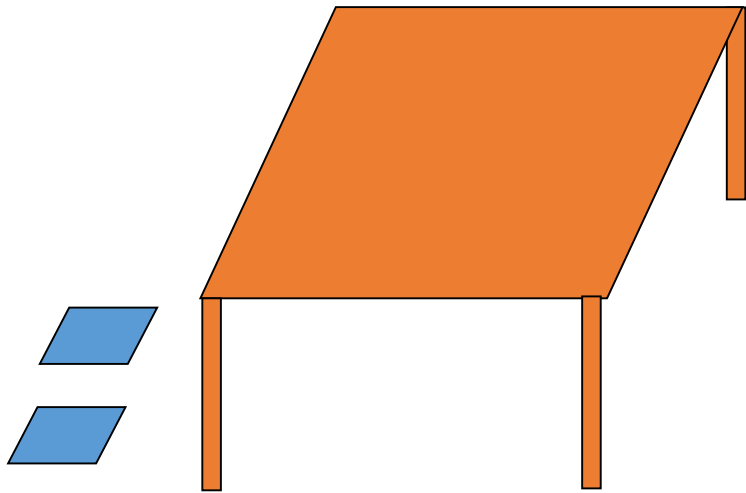


Current stance

Search Algorithm

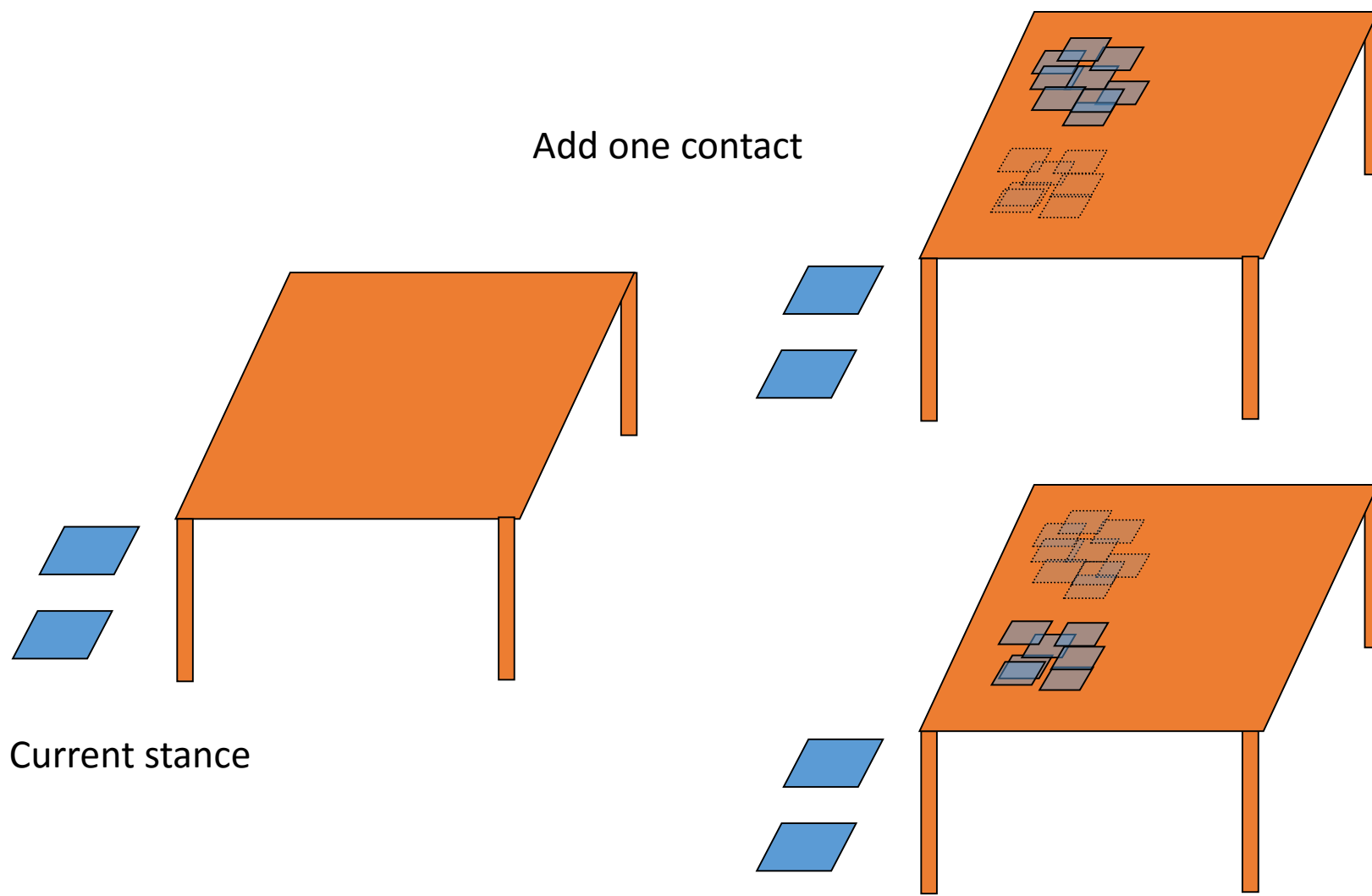


Search Algorithm

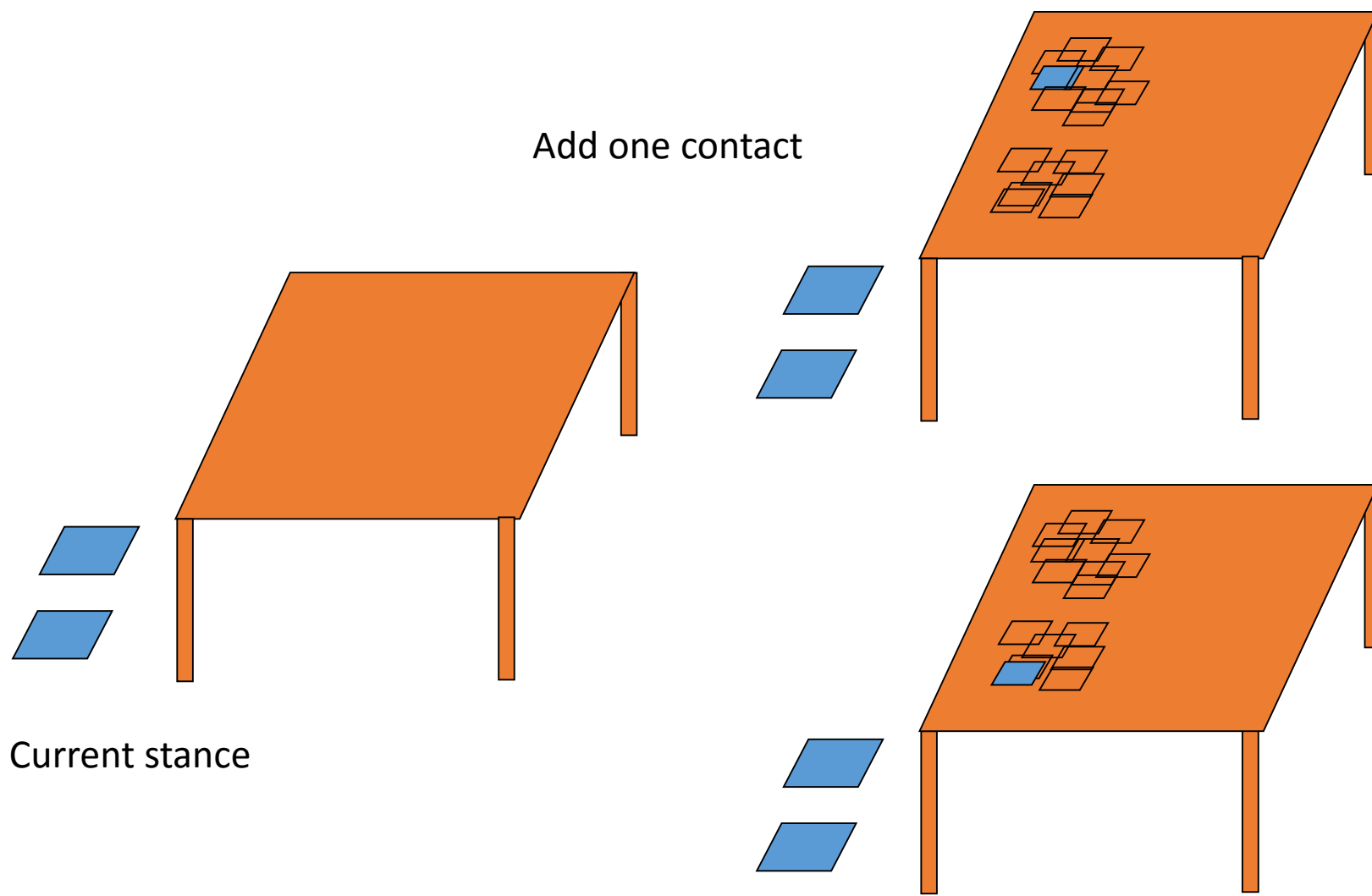


Current stance

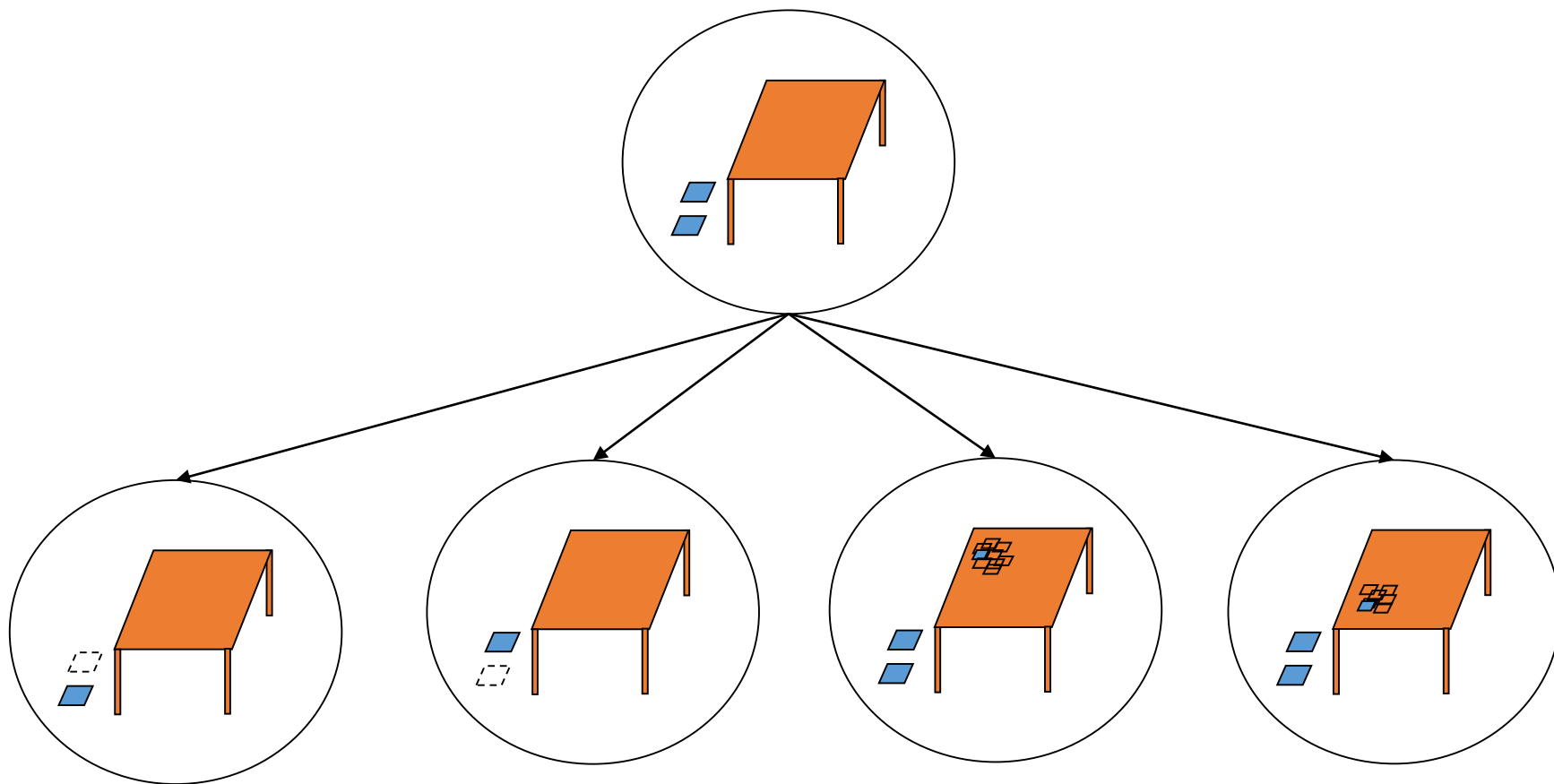
Search Algorithm



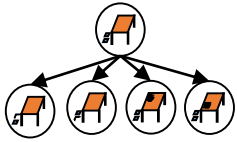
Search Algorithm



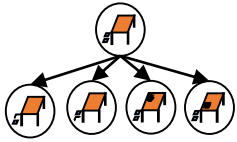
Search Algorithm



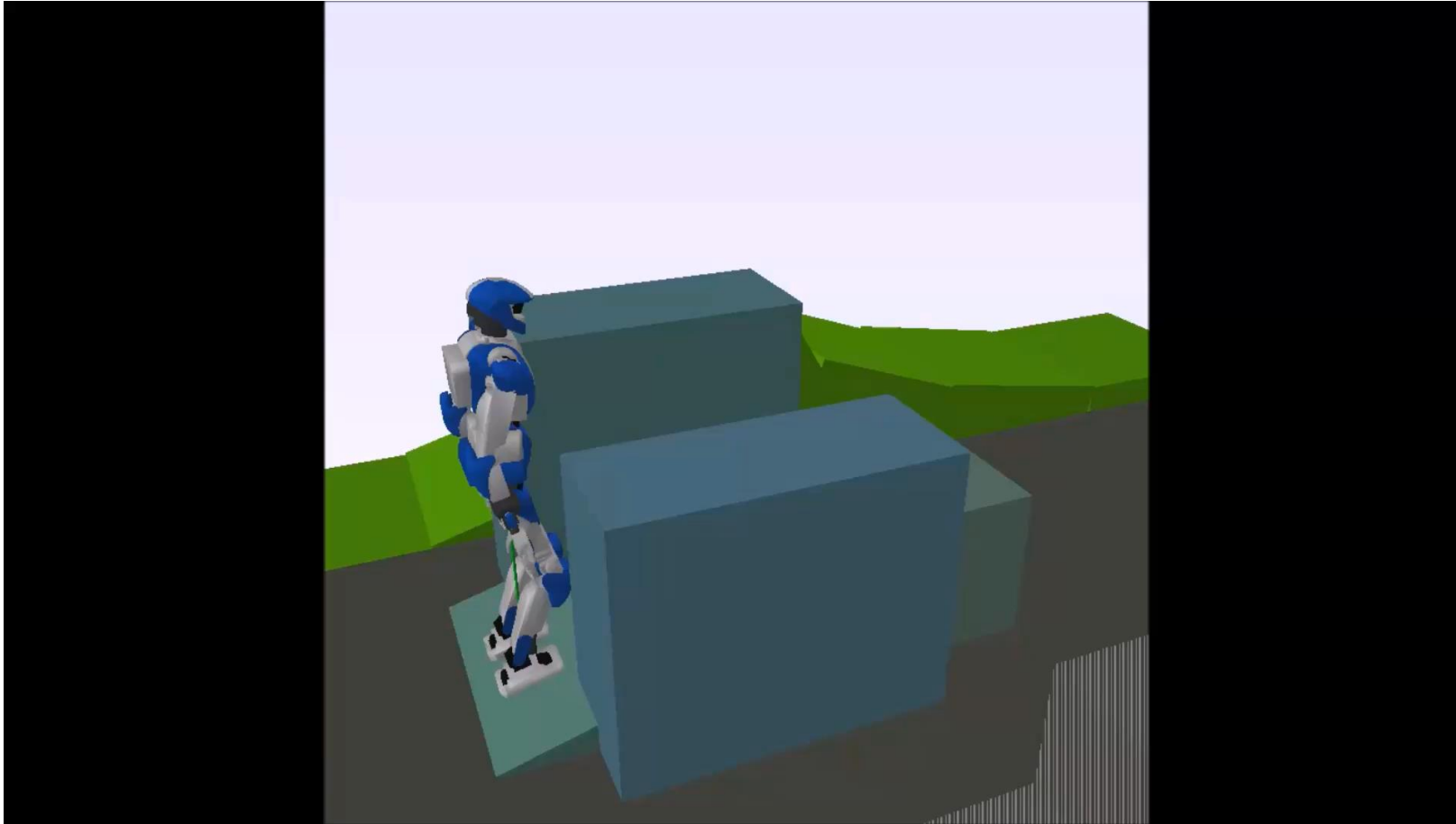
Search Algorithm



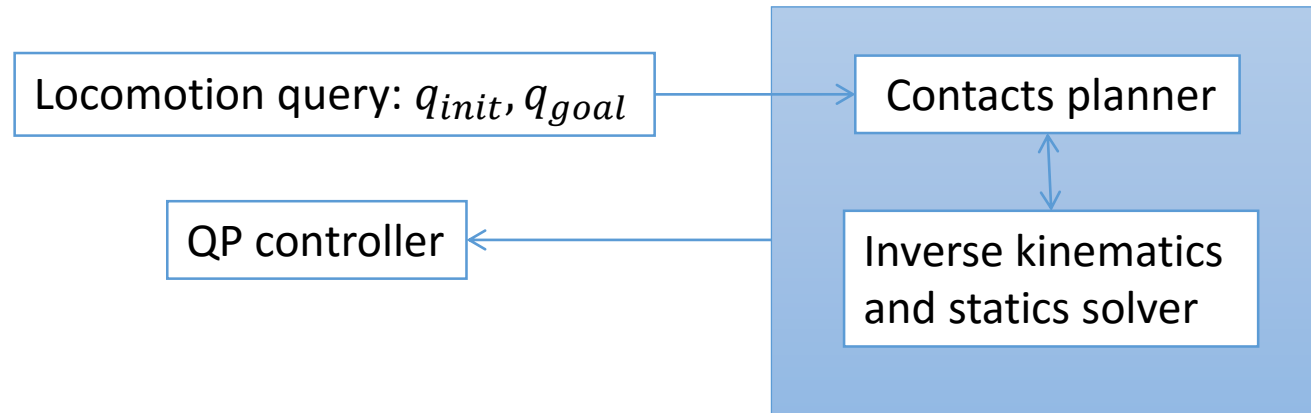
Search Algorithm



Search algorithm



Overview of the Framework



Overview of the Framework

QP controller

QP Control and Simulation

Basic idea: at a given q, \dot{q} , the dynamics equation of motion is linear in \ddot{q}, τ, f :

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

QP Control and Simulation

$$\begin{array}{l} \min_{\ddot{q}, f, \tau} \sum_k w_k \|\ddot{g}_k - \ddot{g}_k^d\|^2 \\ \left\{ \begin{array}{l} M(q)\ddot{q} + N(q, \dot{q}) = S\tau + J^T f \\ J\ddot{q} + \dot{J}\dot{q} = 0 \\ Kf \geq 0 \\ |\tau| \leq \tau_{max} \end{array} \right. \end{array}$$

QP Control and Simulation

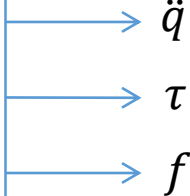
$$\begin{array}{l} \min_{\ddot{q}, f, \tau} \sum_k w_k \|\ddot{g}_k - \ddot{g}_k^d\|^2 \\ \left\{ \begin{array}{l} M(q)\ddot{q} + N(q, \dot{q}) = S\tau + J^T f \\ J\ddot{q} + \dot{J}\dot{q} = 0 \\ Kf \geq 0 \\ |\tau| \leq \tau_{max} \end{array} \right. \end{array}$$


Diagram illustrating the QP Control and Simulation problem. The optimization problem is defined by the objective function and constraints, and the resulting variables are \ddot{q} , τ , and f .

QP Control and Simulation

$$\begin{aligned} & \min_{\ddot{q}, f, \tau} \sum_k w_k \|\ddot{g}_k - \ddot{g}_k^d\|^2 \\ & \left\{ \begin{array}{l} M(q)\ddot{q} + N(q, \dot{q}) = S\tau + J^T f \\ J\ddot{q} + \dot{J}\dot{q} = 0 \\ Kf \geq 0 \\ |\tau| \leq \tau_{max} \end{array} \right. \end{aligned} \quad \rightarrow \quad \ddot{q} \xrightarrow{\int} \dot{q} \xrightarrow{\int} q$$

- Position control
- Simulation

QP Control and Simulation

$$\begin{array}{l} \min_{\ddot{q}, f, \tau} \sum_k w_k \|\ddot{g}_k - \ddot{g}_k^d\|^2 \\ \left\{ \begin{array}{l} M(q)\ddot{q} + N(q, \dot{q}) = S\tau + J^T f \\ J\ddot{q} + \dot{J}\dot{q} = 0 \\ Kf \geq 0 \\ |\tau| \leq \tau_{max} \end{array} \right. \end{array}$$

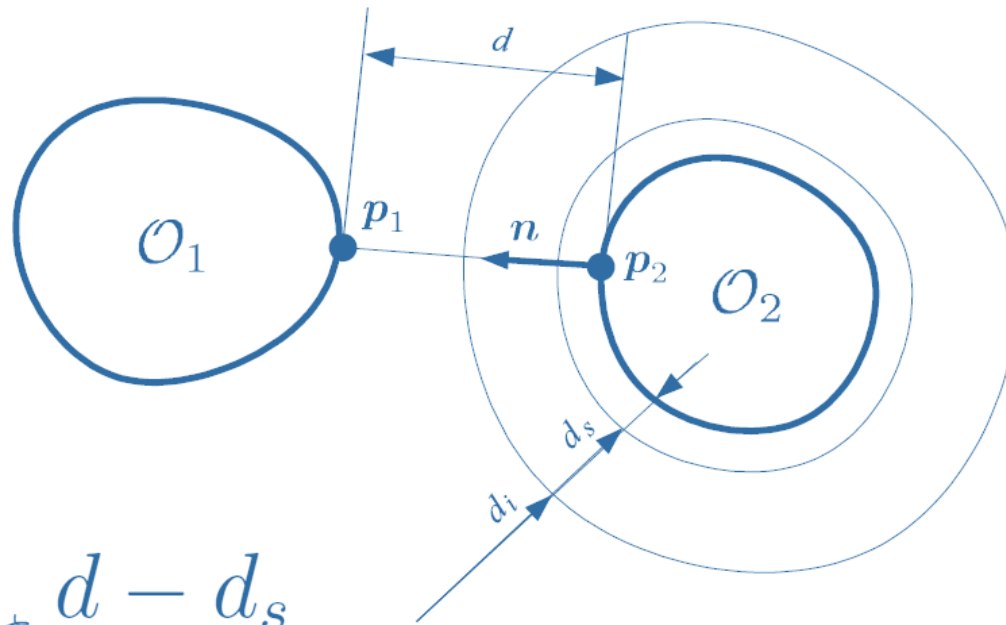
→ τ

Torque command

- to real robot
- to physics simulator

QP Control and Simulation

- Online collision avoidance linear constraint based on velocity-damper formulation (Kanehiro et al, RSS 2008)



$$\dot{d} + \bar{t} \ddot{d} \geq \xi \frac{d - d_s}{d_i - d_s}$$

QP Control and Simulation

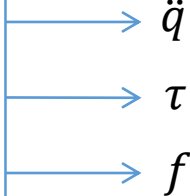
$$\begin{array}{l} \min_{\ddot{q}, f, \tau} \sum_k w_k \|\ddot{g}_k - \ddot{g}_k^d\|^2 \\ \left\{ \begin{array}{l} M(q)\ddot{q} + N(q, \dot{q}) = S\tau + J^T f \\ J\ddot{q} + \dot{J}\dot{q} = 0 \\ Kf \geq 0 \\ |\tau| \leq \tau_{max} \end{array} \right. \end{array}$$


Diagram illustrating the QP Control and Simulation problem. The optimization problem is defined by the objective function and constraints, and the resulting variables are \ddot{q} , τ , and f .

QP Control and Simulation

Stances, static postures sequence

Finite state machine

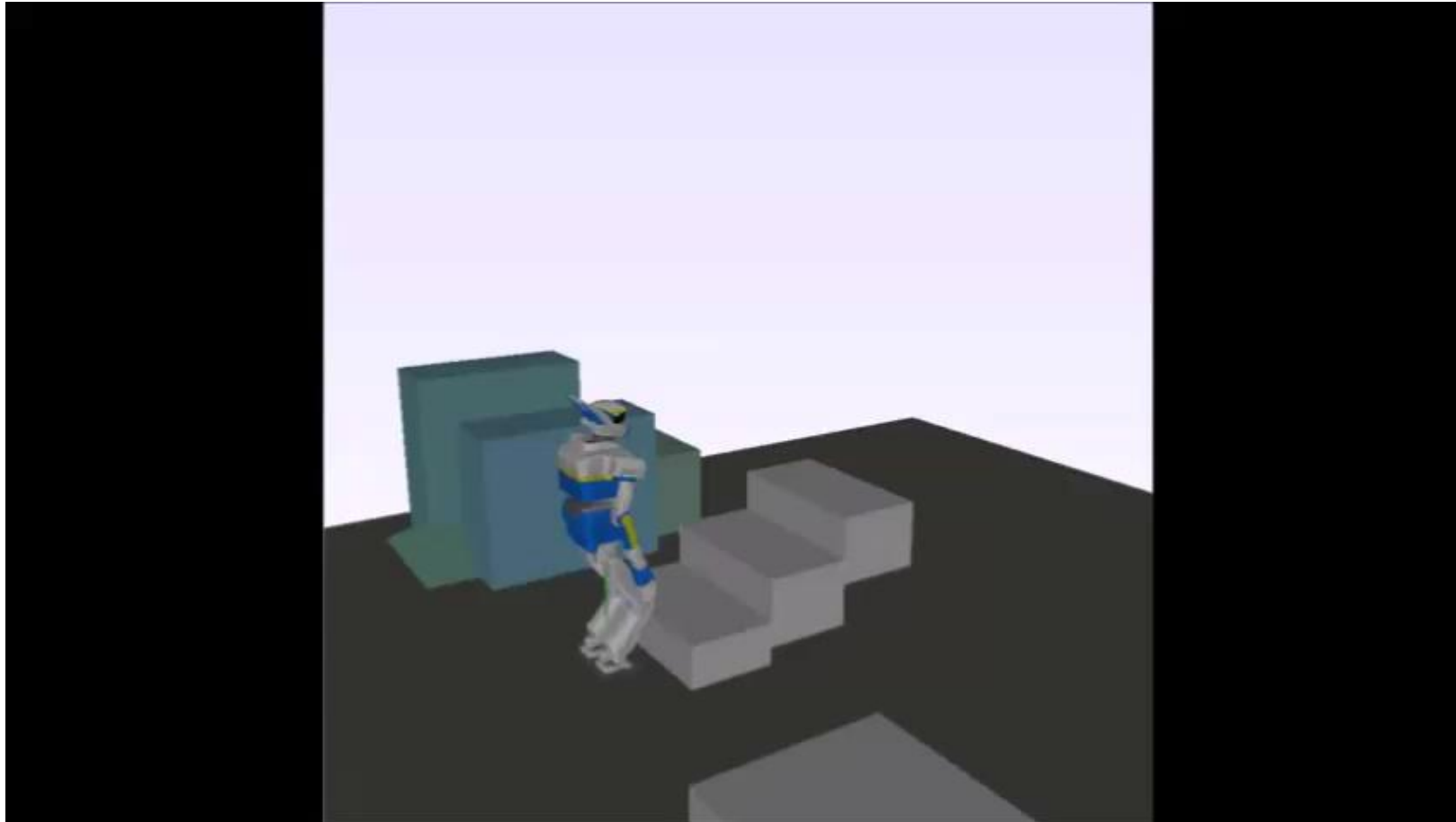
$$\begin{aligned} & \min_{\ddot{q}, f, \tau} \sum_k w_k \|\ddot{g}_k - \ddot{g}_k^d\|^2 \\ & \begin{cases} M(q)\ddot{q} + N(q, \dot{q}) = S\tau + J^T f \\ J\ddot{q} + \dot{J}\dot{q} = 0 \\ Kf \geq 0 \\ |\tau| \leq \tau_{max} \end{cases} \end{aligned}$$

\ddot{q}
 τ
 f

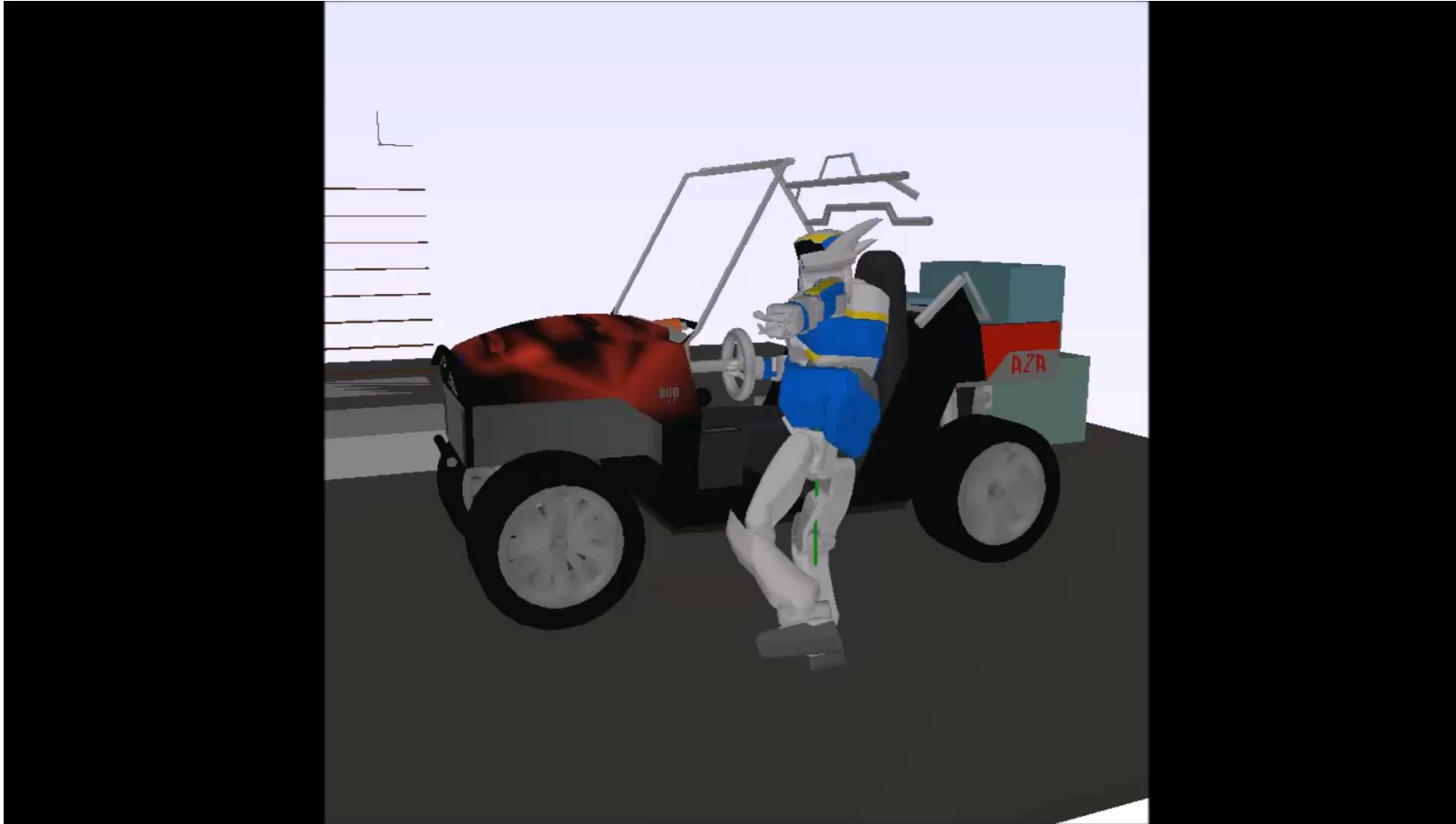
QP Control

- QP control has become golden standard in humanoid robotics
 - All DARPA Robotics Challenge teams used a version of QP control on their robots

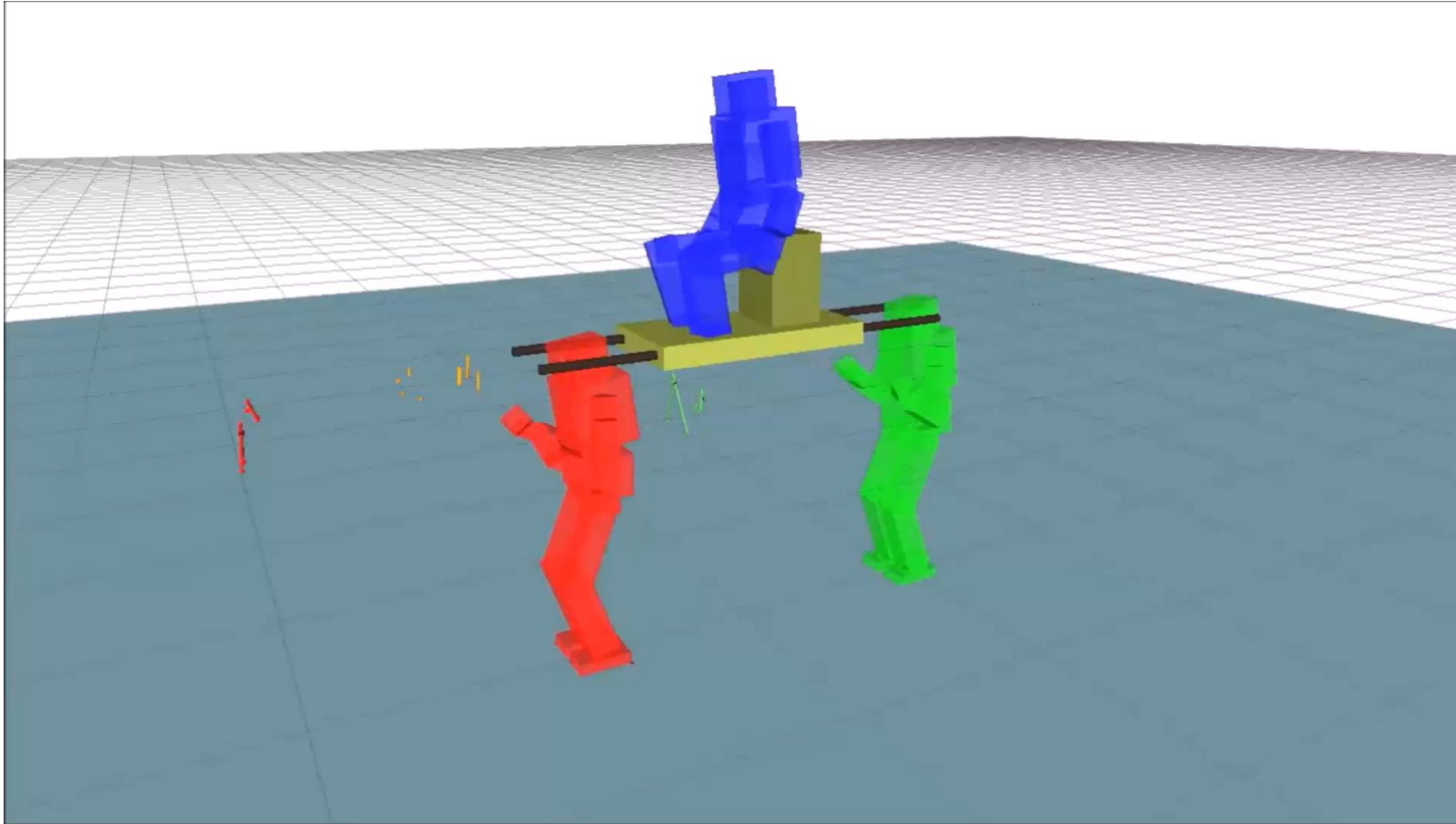
QP Control



QP Control



QP Control



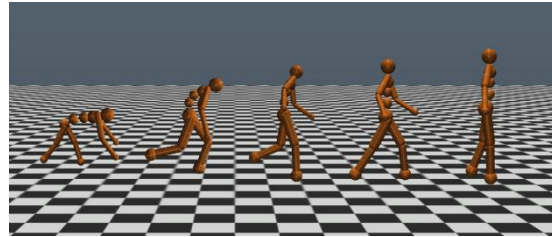
What's next

Many topics we didn't cover

- Perception
 - Humanoid mult-contact visual SLAM (Simultaneous Localization and Mapping)
- Stability of the control
- Robustness to modelling uncertainties
- Middleware control architectures
- Hardware

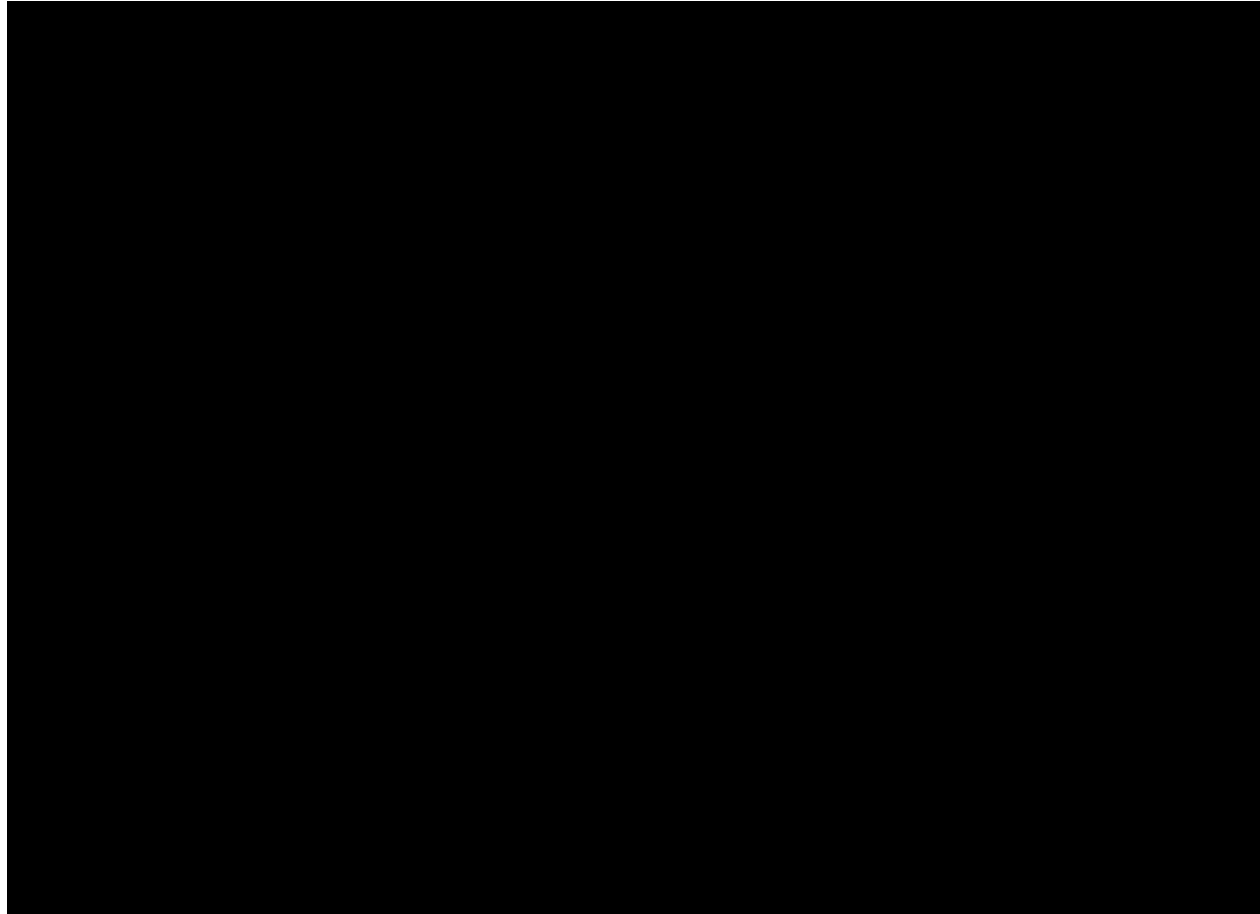
What's next in humanoid motion planning and control?

- Exciting new software developments, leveraging ML techniques, deep RL techniques, and evolutionary algorithms
 - Google Deepmind “Emergence of Locomotion Behaviours in Rich Environments”
 - Uber AI labs “Welcoming the Era of Deep Neuroevolution”

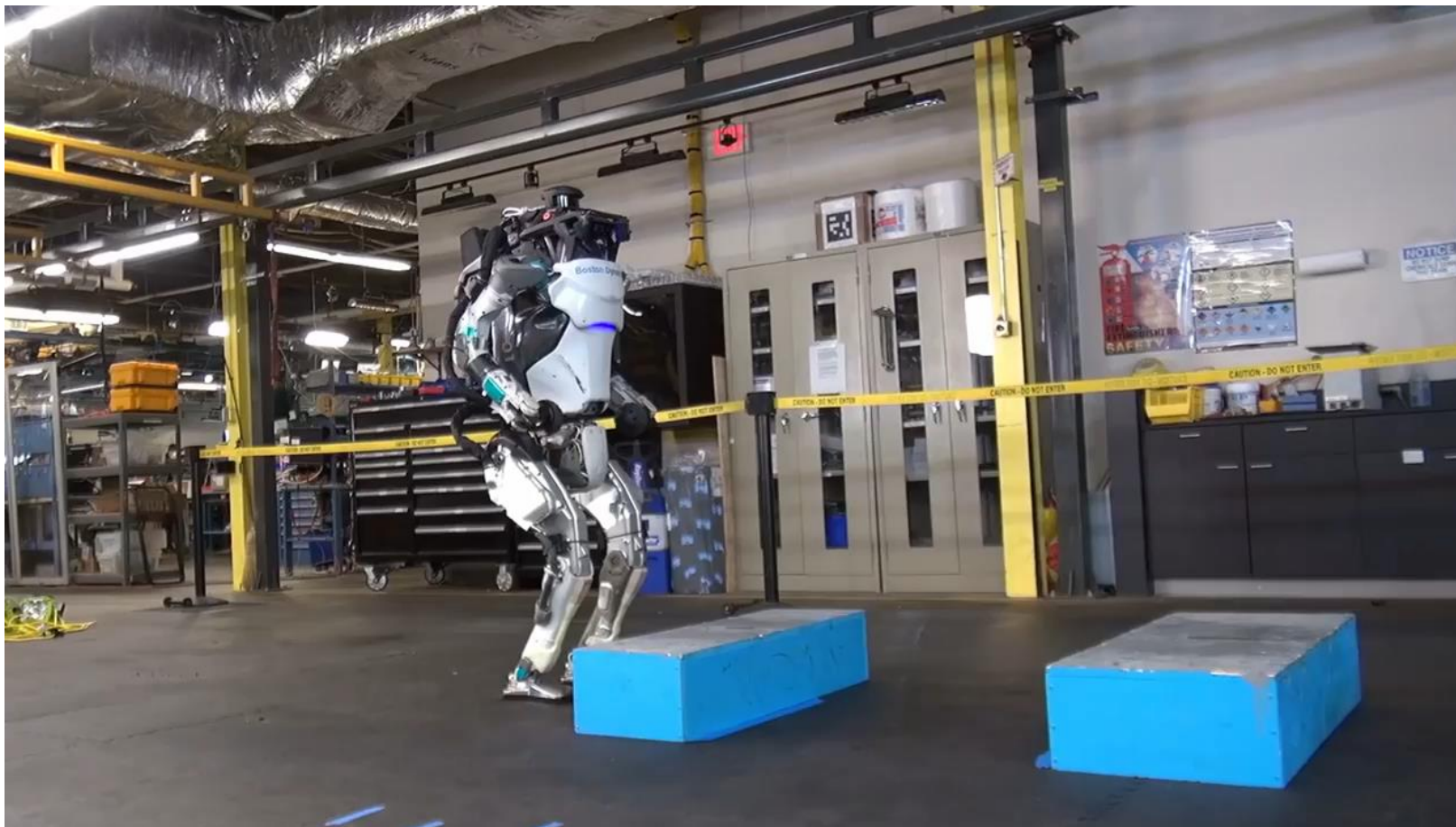


- We are currently working in our team on applying trial and error learning techniques based on evolutionary CMA-ES optimization in multi-contact QP control (Spitz et al, Humanoids 2017)
- Exciting new hardware developments
 - Hardware has always been a big limitation
 - Boston Dynamics’ backflip !
 - New hardware paradigms based on series elastic actuators, artificial muscles, highly resilient hardware capable of sustaining more extreme motion
 - Highly dynamic motions ahead

Deep Reinforcement Learning



Super hardware



Thank you for your attention