Computational Geometry and Geometric Computing

Robot Motion Planning

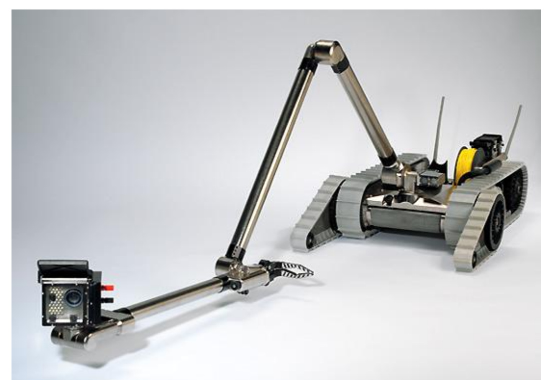
Steven M. LaValle

Presented by
Vera Bazhenova 2010

Overview

– Introduction
– Motion Planning
– Configuration Space
– Sampling-Based Motion Planning
– Comparaison of related Algorithms

Introduction



Introduction

– Robot motion planning encompasses several different disciplines

– Most notably robotics, computer science, control theory and mathematics

– This volume presents an interdisciplinary account of recent developments in the field

Introduction

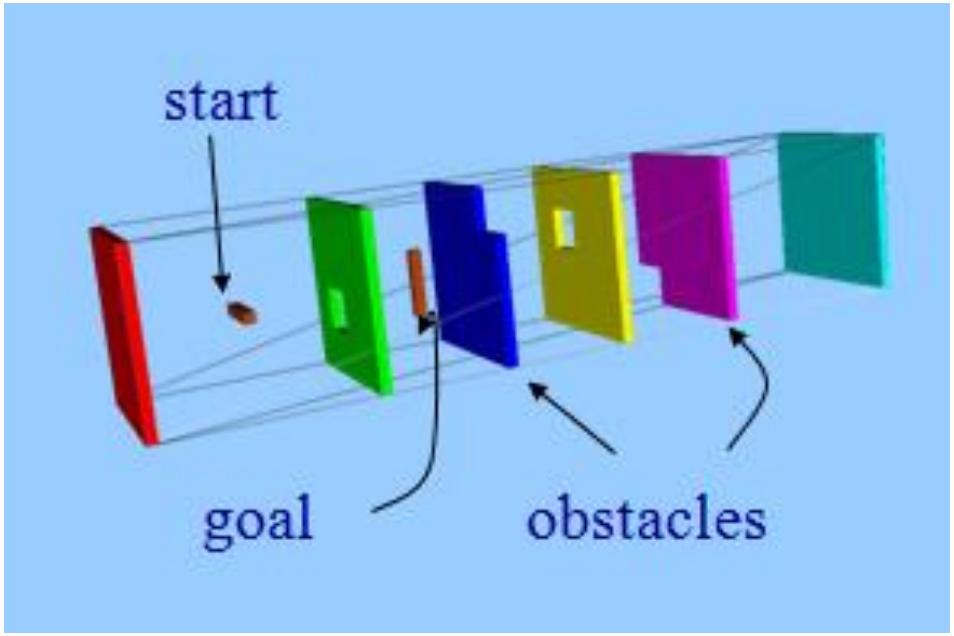
– Most important in robotics is to have algorithms that convert high-level specifications of tasks from humans into low-level descriptions of how to move

– **Motion planning** and **trajectory planning** are often used for these kinds of problems

Introduction

Motion Planning

– Given a robot, find a sequence of valid configurations that moves the robot from the source to destination



Introduction

Piano Mover's Motion Planning

– Assume: A computer-aided design (CAD) model of a house and a piano as input to an algorithm

– Required: Move the piano from one room to another in the house without hitting anything

Introduction

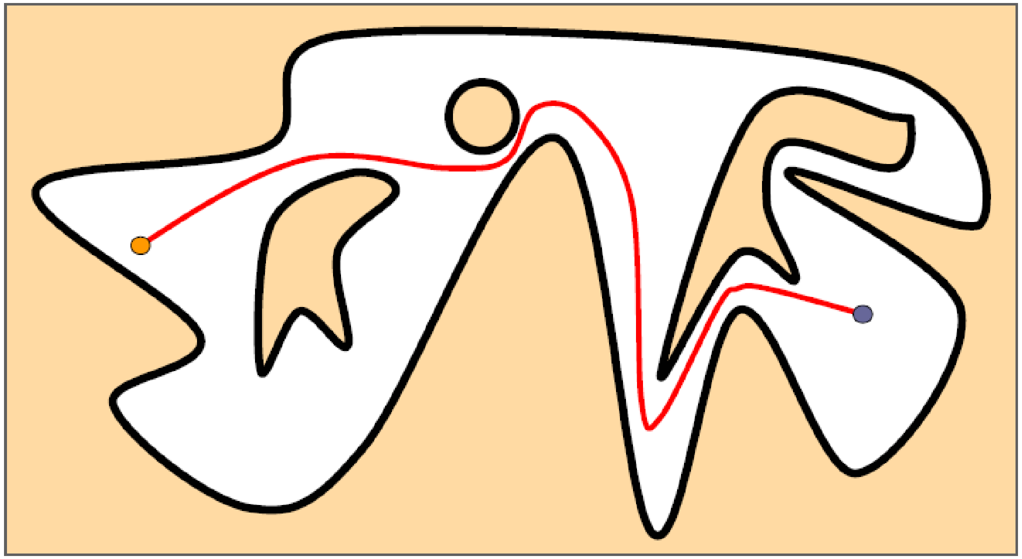
Piano Mover's Motion Planning

Robot motion planning usually ignores dynamics and other differential constraints and focuses primarily on the translations and rotations required to move the piano

Introduction

Trajectory Planning

– By Trajectory Planning we are using robot coordinates because it's easier, but we loose visualization



Introduction

Piano Mover's Trajectory planning

– Taking the solution from a robot motion planning algorithm

– Determining how to move along the solution in a way that respects the mechanical limitations of the robot

Introduction

– We consider planning as a branch of algorithms

– The focus here includes numerous concepts that are not necessarily algorithmic but aid in modeling, solving, and analyzing planning problems

Introduction

Planning involves:

– **State Space:** captures all possible situations that could arise (Position, Orientation)

– **Time:** Sequence of decisions that must be applied over time

– **Action:** Manipulate the state and it must be specified how the state changes when action are applied

Introduction

– **Initial and goal states:** Starting in some initial state and trying to arrive at a specified goal state

– **A criterion:** Encodes the desired outcome of a plan in terms of the state and actions that are executed and consist of two types, Feasibility and Optimality

– **A plan**: imposes a specific strategy or behavior on a decision maker

Introduction

Compute motion strategies:

– Geometric paths

– Time-parameterized trajectories

– Sequence of sensor-based motion commands

Achieve high-level goals:

– Go to the door and do not collide with obstacles

– Build a map of the hallway

– Find and track the target

Overview

– Introduction
– Motion Planning
– Configuration Space
– Sampling-Based Motion Planning
– Comparaison of related Algorithms

Overview

- Introduction
- Motion Planning
  - Geometric Representation and Transformation
  - Configuration Space
- Configuration Space
- Comparaison of related Algorithms

Motion Planning

– **Work Space:** Environment in which robot operates

– **Obstacles:** Already occupied spaces of the world.

– **Free Space:** Unoccupied space of the world

Motion Planning

Continuous representation

↓

Discretization

↓

Graph searching

(blind, best-first, A*)

Motion Planning

Geometric Representation and Transformation

Geometric modeling consist on two alternatives:

- A Boundary Representation
- A Solid Representation

Motion Planning

Geometric Representation and Transformation

Let's define the world $\mathcal{W}$ for which there are two possible choices:

1) a 2D world, in which $\mathcal{W} = \mathbb{R}^2$
2) a 3D world, in which $\mathcal{W} = \mathbb{R}^3$

Motion Planning

Geometric Representation and Transformation

The world W contains two kinds of entities:

- **Obstacles:** Portions of the world that are "permanently" occupied
- **Robots:** Bodies that are modeled geometrically and are controllable via a motion plan

Both are considered as Subset of $\mathcal{W}$

Motion Planning

Geometric Representation and Transformation

– The obstacle region $\mathcal{O}$ denote the set of all points in W that lie in one or more obstacles: Hence, $\mathcal{O} \subseteq \mathcal{W}$

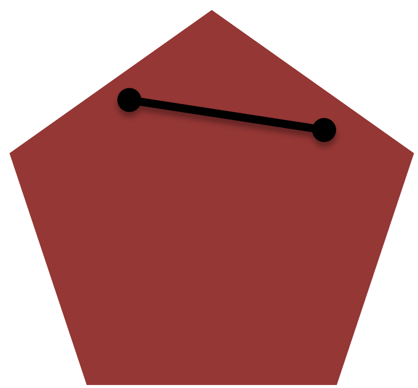
– $\mathcal{O}$ is a combination of Boolean primitives H

– Each primitives H is easy to represent and manipulate

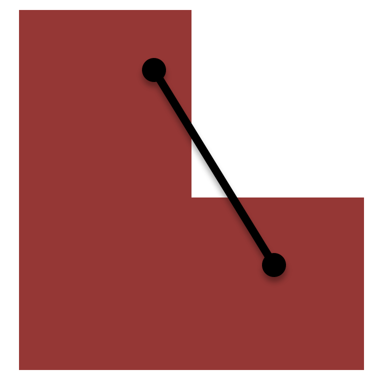
Motion Planning

Geometric Representation and Transformation

– Let's consider obstacle $\mathcal{O}$ as **Convex Polygons**



Convex



Non Convex

Case of Non convex is returned to the convex representation

$$\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \cdots \cup \mathcal{O}_n$$

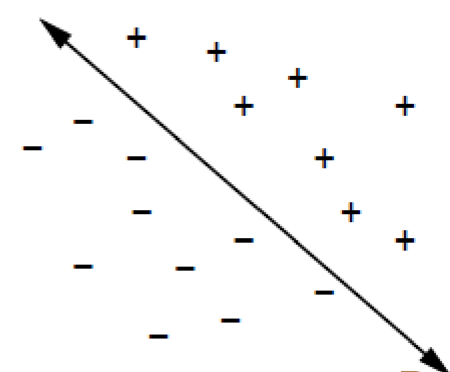
Motion Planning

Geometric Representation and Transformation

- Primitives H are presented by a set of two points (x,y)
- Each two points describe a line

$$ax + by + c = 0$$

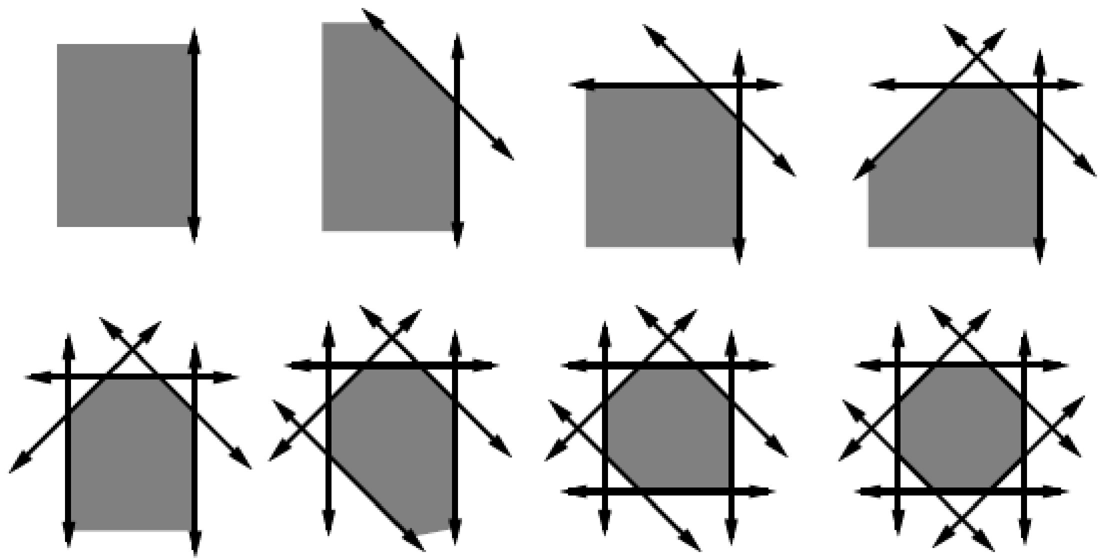
$f(x,y) = ax + by + c$ is positive from one side and negative from the other side



Motion Planning

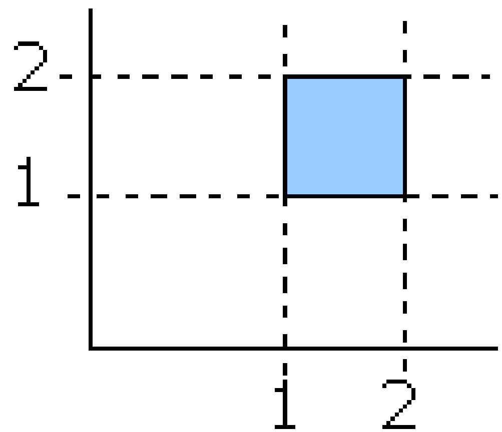
Geometric Representation and Transformation

- Obstacle is defined by a set of vertices (corner) or lines (edges) described by pairs of two points



$$H_i = \{(x, y) \in \mathcal{W} \mid f_i(x, y) \leq 0\}$$

Example:



$f_1(x,y) = x-2$
$f_2(x,y) = y-2$
$f_3(x,y) = 1-x$
$f_4(x,y) = 1-y$

$$H_i = \{(x,y) \in \mathcal{W} \mid f_i(x,y) \leq 0\}$$

i = 1.. 4

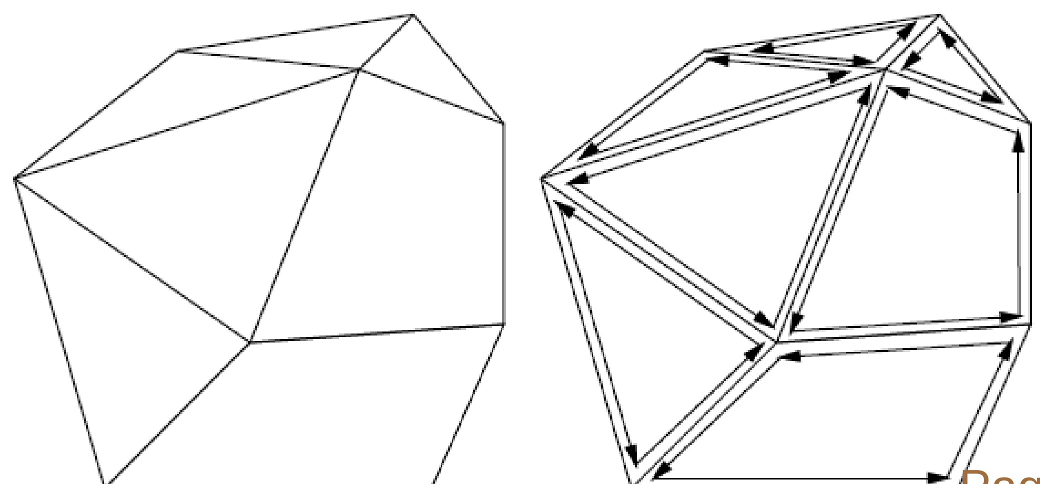
$$\mathcal{O} = H_1 \cap H_2 \cap \cdots \cap H_m$$

m = 1.. 4

Motion Planning

Geometric Representation and Transformation

- In 3D, the representation with primitives is similar to 2D
- Instead of Polygons, we use Polyhedral
- The lines became planes

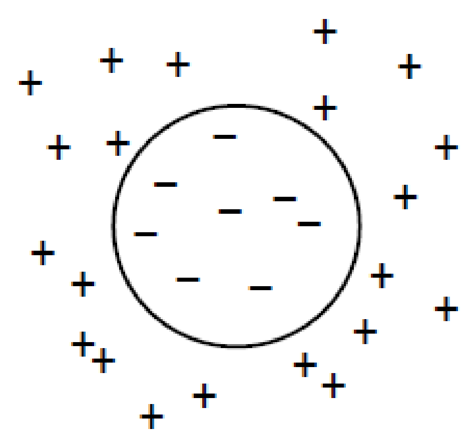


Semi-Algebraic Models:

$$H = \{(x, y) \in \mathcal{W} \mid f(x, y) \leq 0\}$$

- f can be any polynomial
- For example let's take:

$$f = x^2 + y^2 - 4$$



Motion Planning

Geometric Representation and Transformation

Other representation Models:

- **3D triangles:** Represent complex geometric shapes as a union of triangles

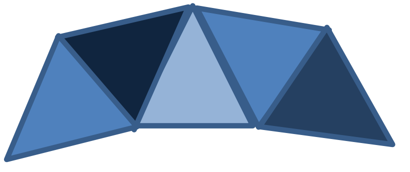
- **Nonuniform rational B-splines:** find an Interpolation over a set of points

Motion Planning

Geometric Representation and Transformation

Other representation Models:

– **3D triangles:**



– **Nonuniform rational B-splines:**

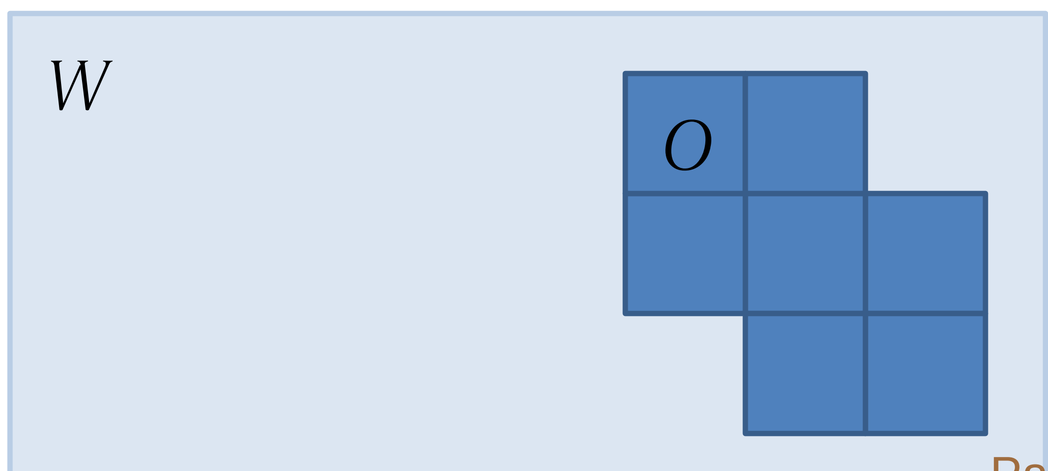
$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i N_{i,k}(u)}$$

Motion Planning

Geometric Representation and Transformation

Other representation Models:

– **Bitmaps:** Discretize a bounded portion of the world into rectangular cells



Motion Planning

Geometric Representation and Transformation

Translation:

$$h(\mathcal{A}) = \{h(a) \in \mathcal{W} \mid a \in \mathcal{A}\}$$

- A is the set of Robot points
- h is the transformation applied to each point of A
- Transformation is defined as below:

$$h(x, y) = (x + x_t, y + y_t)$$

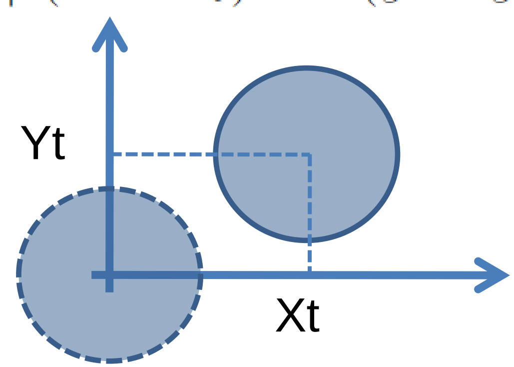
Translation: Example

Appling h to the Representation H of a disc:

$$H_i = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 - 1 \leq 0\}$$

$$\Downarrow$$

$$h(H_i) = \{(x,y) \in \mathcal{W} \mid (x - x_t)^2 + (y - y_t)^2 - 1 \leq 0\}$$



Motion Planning

Geometric Representation and Transformation

Rotation:

Given the following Rotation Matrix:

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

the set of Robot points get rotated to:

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

Motion Planning

Geometric Representation and Transformation

A kinematic chain

– Is the assembly of several kinematic pairs connecting rigid body segments

Motion Planning

Geometric Representation and Transformation

2D Kinematic Chain:

In the case of two unattached rigid bodies $A1$ and $A2$, there is 6 degrees of Freedom, two Rotations and four Translations:

$$x_1,\ y_1,\ \theta_1,\ x_2,\ y_2,\ \text{and}\ \theta_2$$

By attaching bodies, degrees of freedom are removed

Motion Planning

Geometric Representation and Transformation

2D Kinematic Chain: *Attaching bodies*

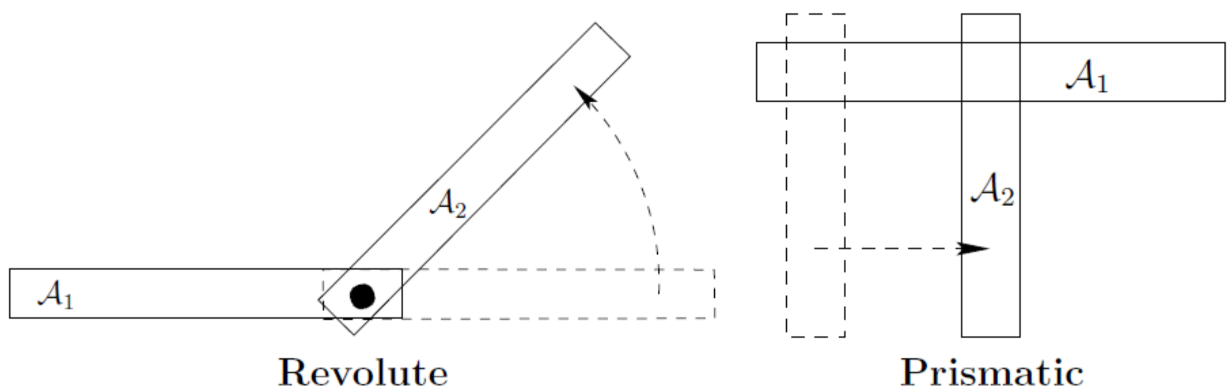
- The place at which the links are attached is called a joint
- Two types of 2D joints: Prismatic and Revolute

Motion Planning

Geometric Representation and Transformation

2D Kinematic Chain: *Attaching bodies*

– **Revolute Joint:** allows one link to rotate with respect to the other

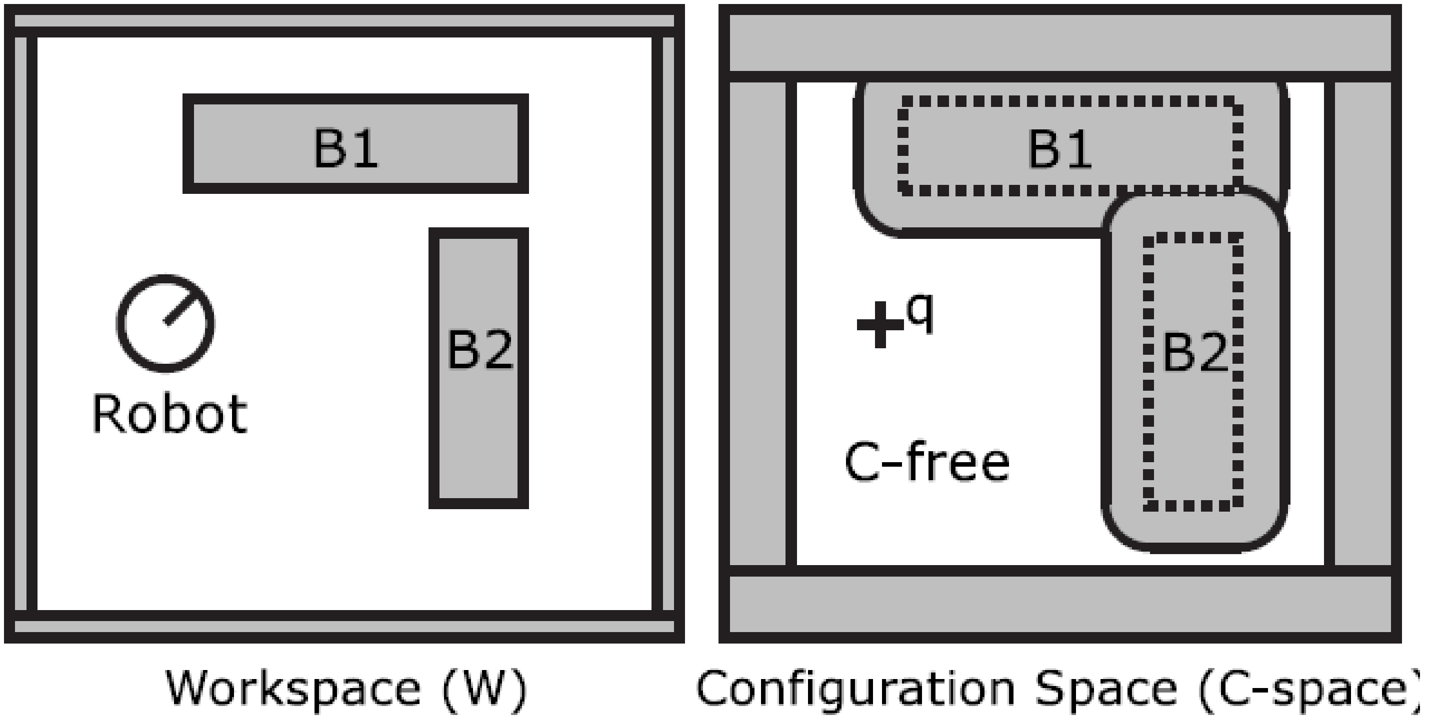


– **Prismatic Joint:** allows one link to translate with respect to the other

Overview

– Introduction
– Motion Planning
– Configuration Space
– Sampling-Based Motion Planning
– Comparaison of related Algorithms

The Configuration Space



A vector $\boldsymbol{q}$ including all degree of freedom is called a configuration

The Configuration Space

Configuration space ($\mathcal{C}_{space}$)

= set of all configurations

Free space ($\mathcal{C}_{free}$)

= set of allowed (feasible) configurations

Obstacle space ($\mathcal{C}_{obstacle}$)

= set of disallowed configurations

$$\mathcal{C}_{space} = \mathcal{C}_{free} + \mathcal{C}_{obstacle}$$

The Configuration Space

Path of an object =

= curve in the configuration space

represented either by:

Mathematical expression, or

Sequence of points

Trajectory

= **Path** + assignment of time to points along the path

The Configuration Space

Is the set of legal configurations of the robot

It also defines the topology of continuous motion

For rigid-object robots (no joints) there exists:

– a transformation to the robot and obstacles that turns the robot into a single point.

The Configuration Space

The C-Space Transform

– Turn Robot to a Point:
Make the Obstacle bigger by applying the **Minkowski** addition

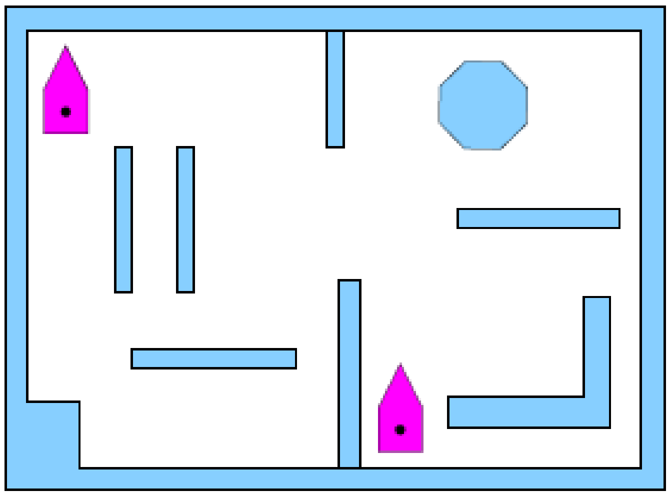
$$S1 \oplus S2 = \{p+q \mid p \in S1,\ q \in S2\}$$

$$\text{mit } S1, S2 \in \mathbb{R}^n$$

– The configuration space is the Minkowski sum of the set of obstacles and the robot placed at the origin

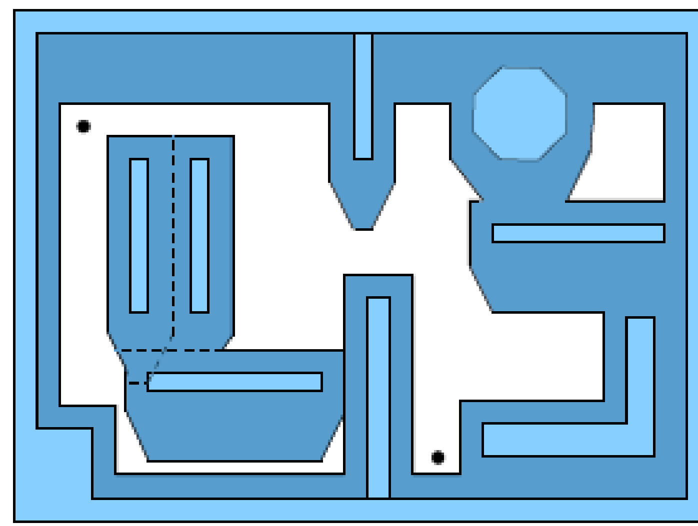
The Configuration Space

The C-Space Transform



1. Robot motion planning Problem

2. Robot motion planning Problem after C-Space Transformation



Overview

– Introduction
– Motion Planning
– Configuration Space
– Sampling-Based Motion Planning
– Comparaison of related Algorithms

Sampling-Based Motion Planning

The motion planning problem consists of the following:

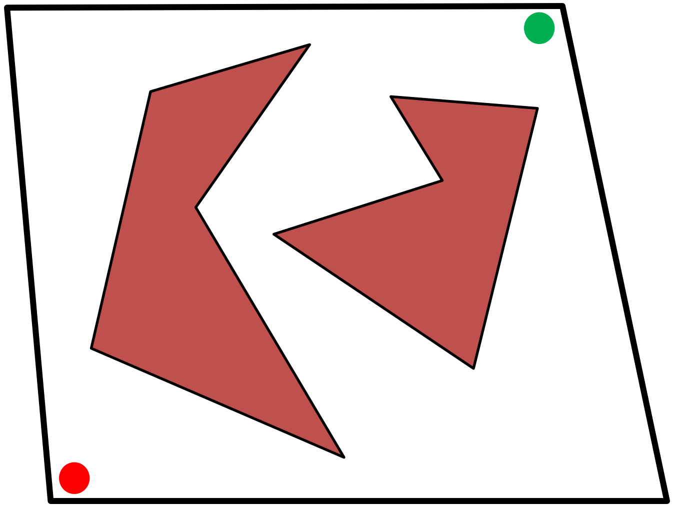
Input

– geometric descriptions of a robot and its obstacles

– initial and goal configurations

Output

– a path from start to finish (or the Recognition that none exists



Sampling-Based Motion Planning

Classic Path Planning Approaches

– Roadmap

– Cell decomposition

– Potential field

Sampling-Based Motion Planning

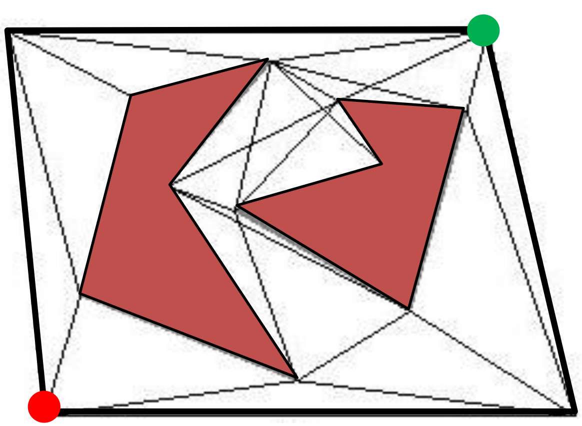
Roadmap

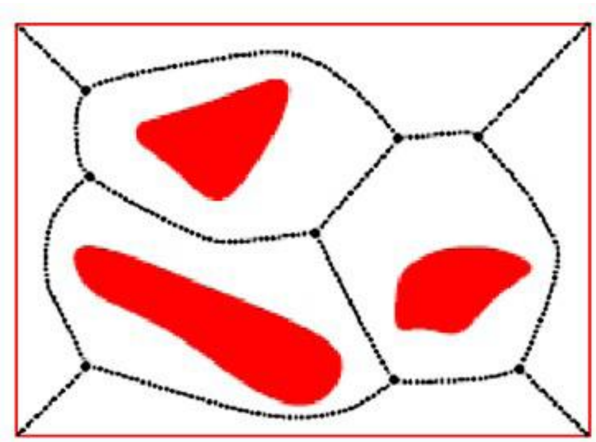
Cell decomp.

Potential field

Roadmap

Represent the connectivity of the free space by a network of 1-D curves, as in *Visibility* or *Voronoi* Diagrams





Sampling-Based Motion Planning

Roadmap

Cell decomp.

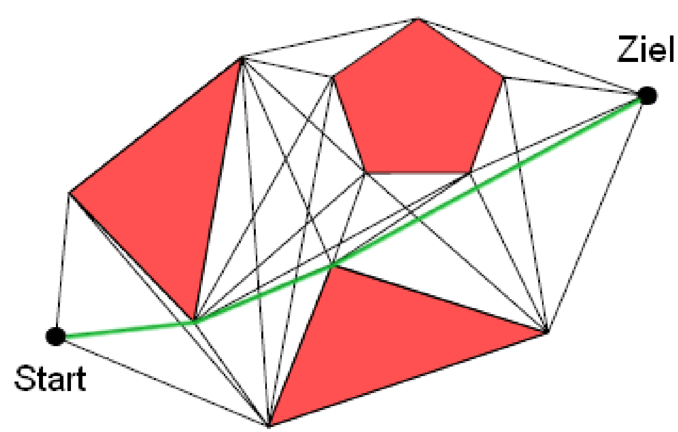
Potential field

Visibility Diagram

Construct all of the line segments that connect vertices to one another

From $\mathcal{C}_{free}$, a graph is defined

Converts the problem into graph search



Sampling-Based Motion Planning

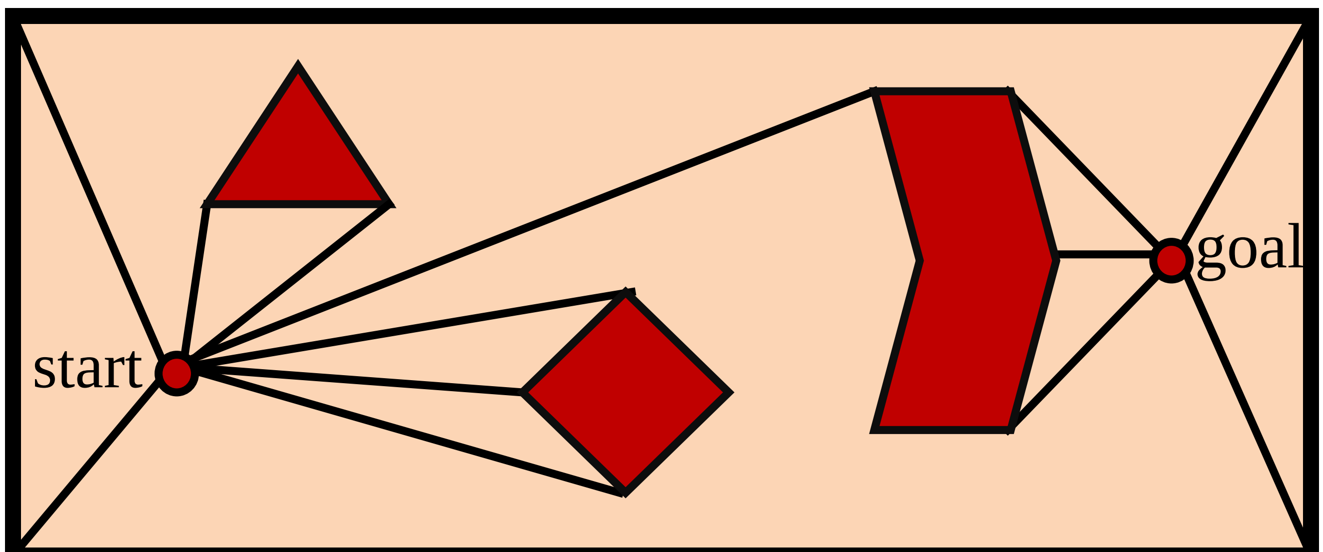
Roadmap

Cell decomp.

Potential field

Visibility Diagram

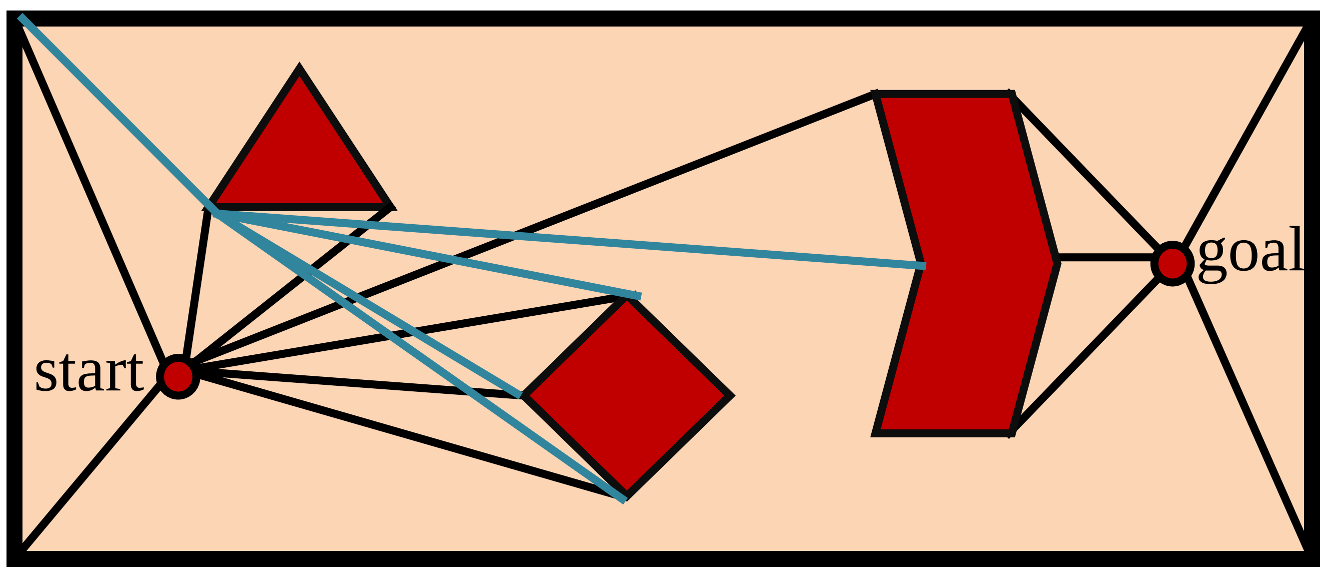
We start by drawing lines of sight from the start and goal to all “visible” vertices and corners of the Configuration Space



Sampling-Based Motion Planning

Visibility Diagram

Than draw the lines of sight from every vertex of every obstacle like before



Sampling-Based Motion Planning

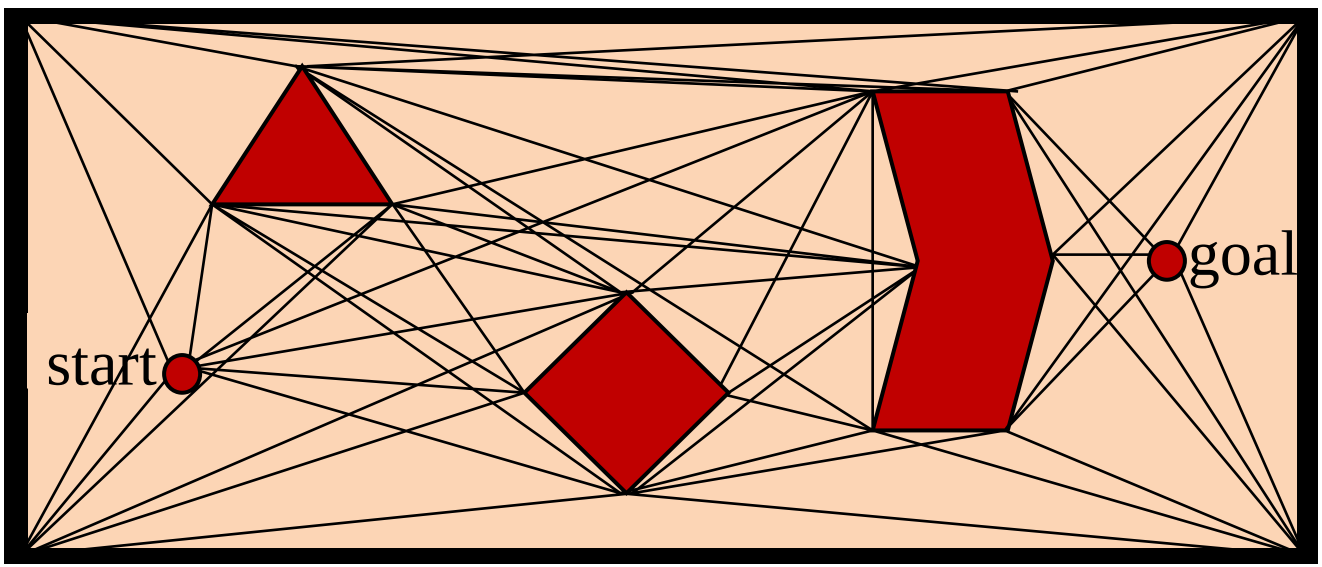
Roadmap

Cell decomp.

Potential field

Visibility Diagram

After connecting all vertices of our obstacles, we obtain following graph



Sampling-Based Motion Planning

Roadmap

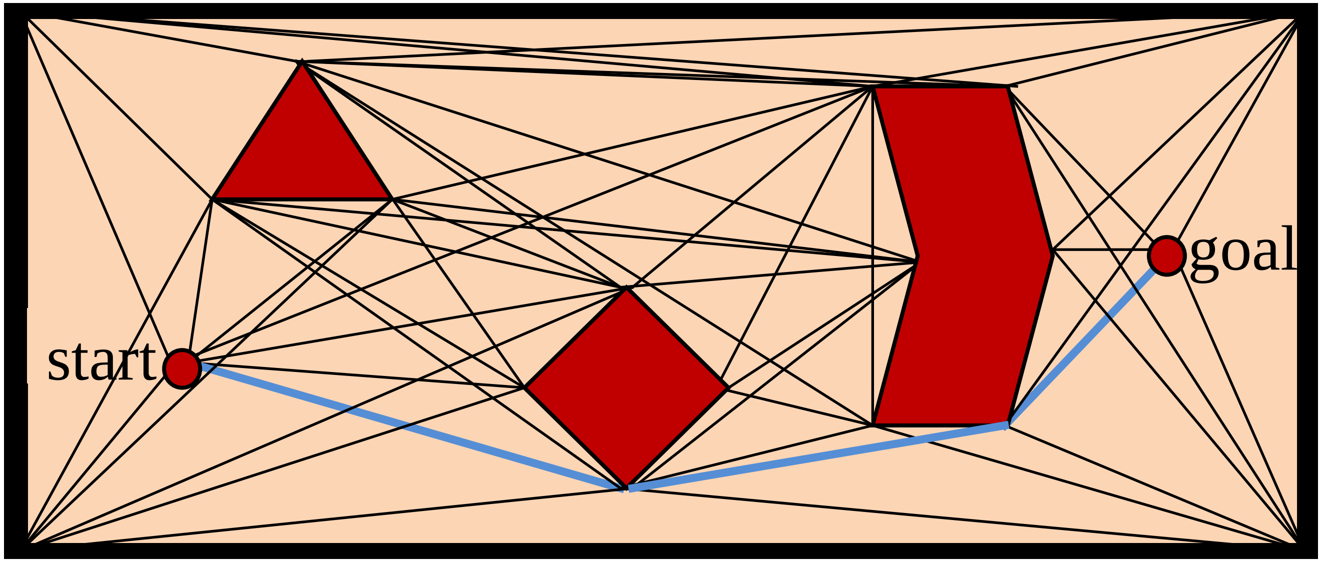
Cell decomp.

Potential field

Visibility Diagram

+ Through the founded Graph we could find the most shorter path

- But the trajectory is to close to the obstacles



If there are n vertices, the easy algorithm is $O(n^3)$. Slightly tougher $O(n^2 logn)$. $O(n^2)$ in theory.

Sampling-Based Motion Planning

Roadmap

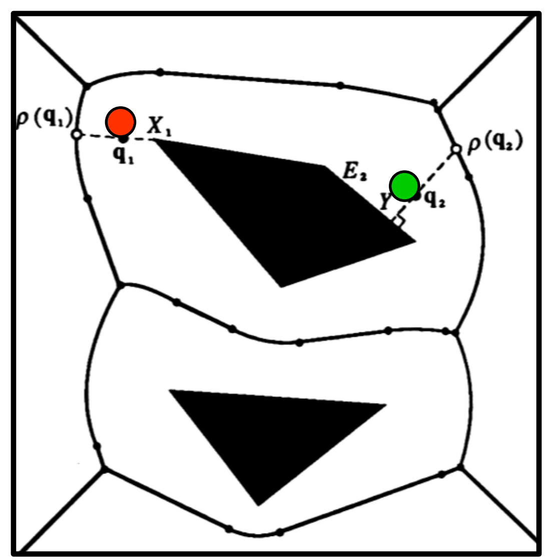
Cell decomp.

Potential field

Voronoi Diagram

Set of points equidistant from the closest two or more obstacle boundaries

Maximizing the clearance between the points and obstacles



Sampling-Based Motion Planning

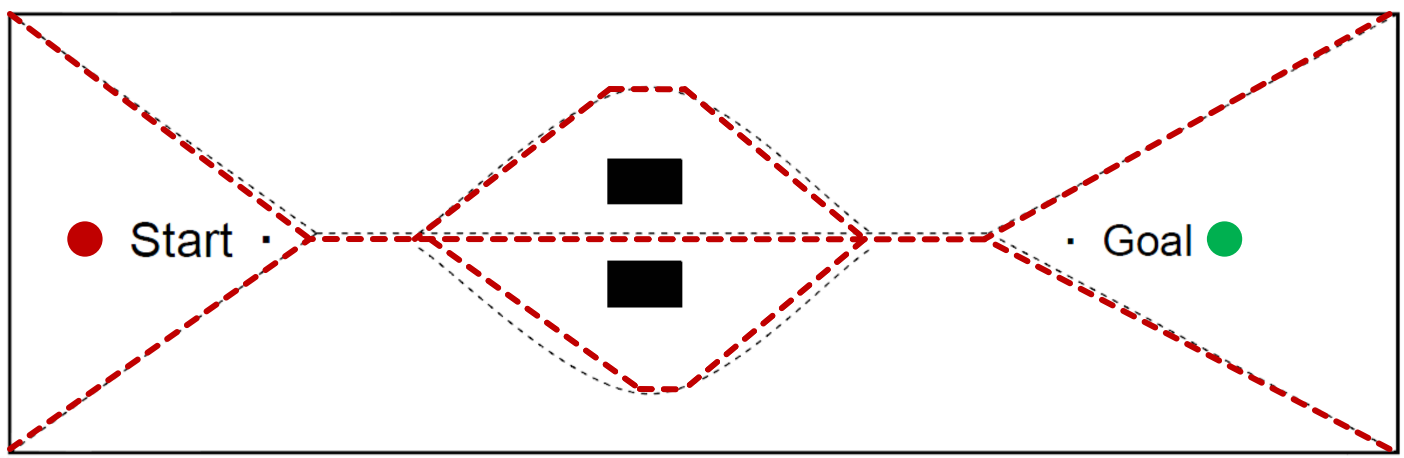
Roadmap

Cell decomp.

Potential field

Voronoi Diagram

– Compute the Voronoi Diagram of C-space



Sampling-Based Motion Planning

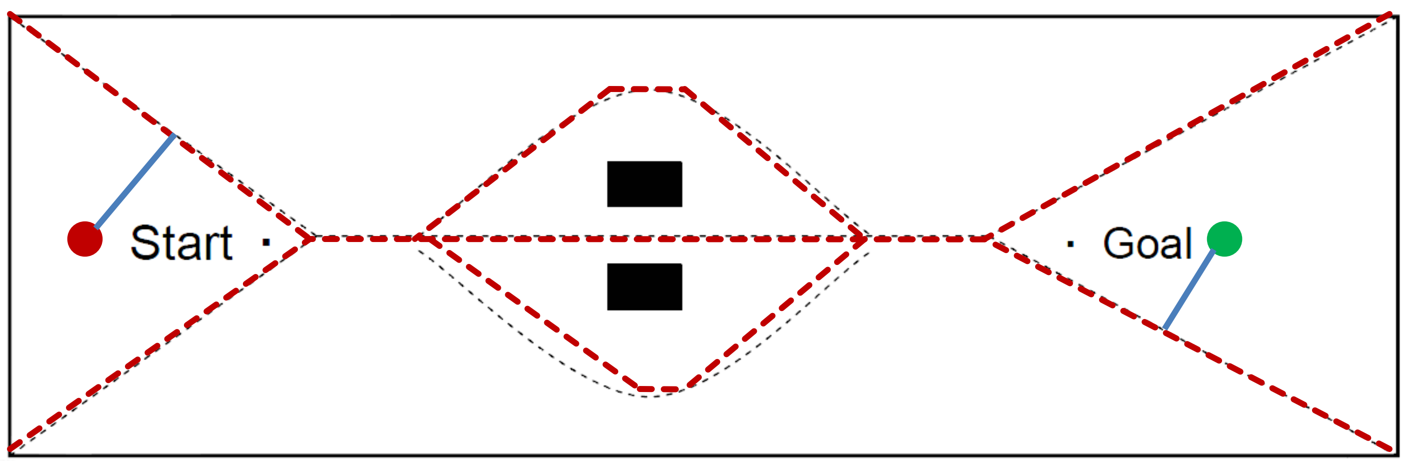
Roadmap

Cell decomp.

Potential field

Voronoi Diagram

– Compute shortest straightline path from start and Goal to closest point on Voronoi Diagram



Sampling-Based Motion Planning

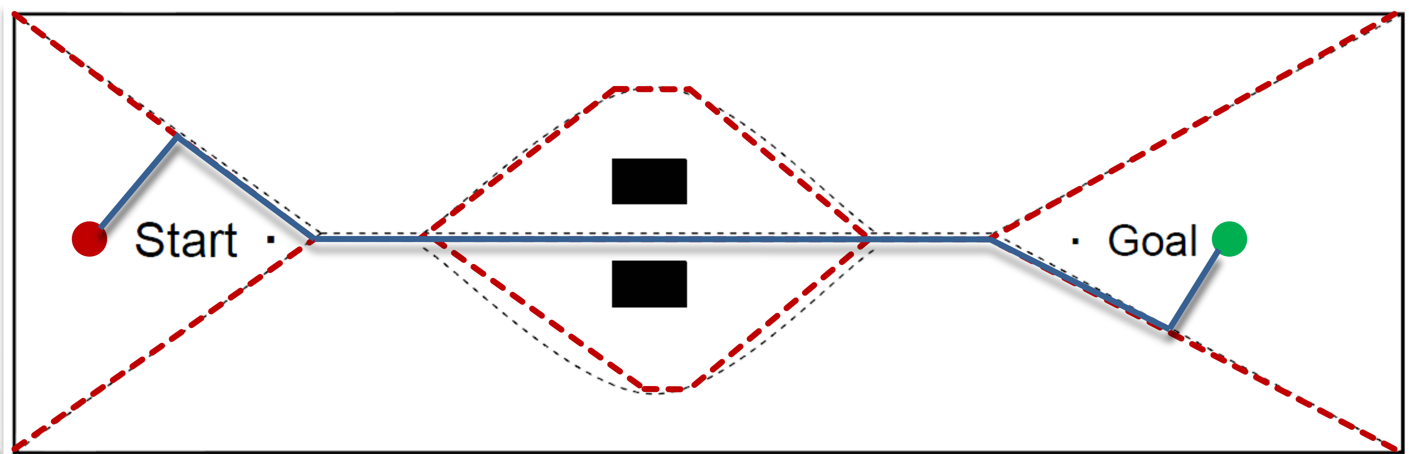
Roadmap

Cell decomp.

Potential field

Voronoi Diagram

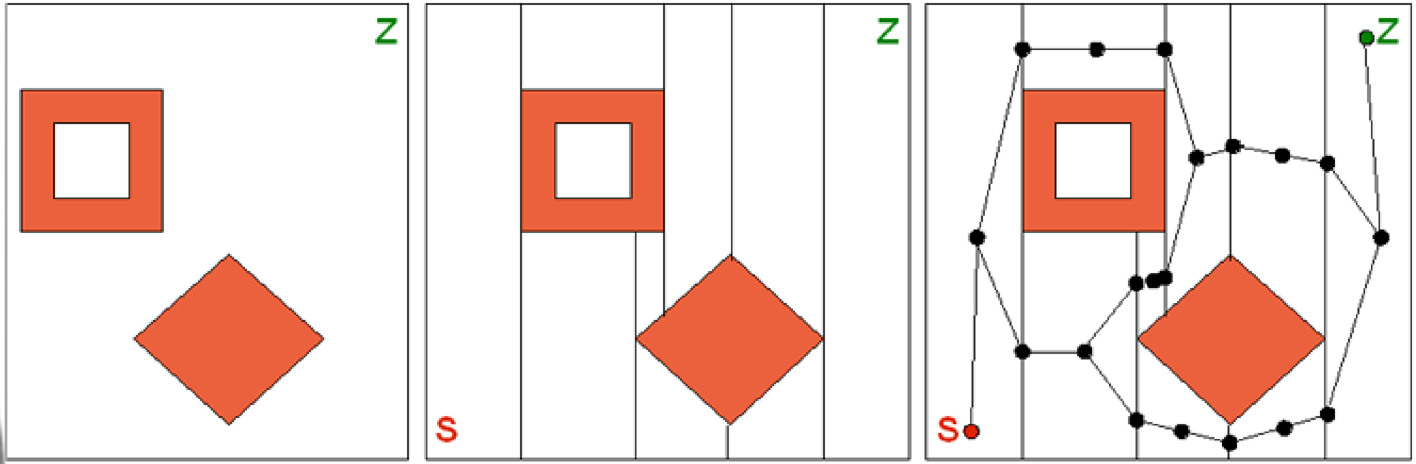
– Compute shortest path from start to goal along Voronoi Diagram



Sampling-Based Motion Planning

Cell decomposition

Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells



Sampling-Based Motion Planning

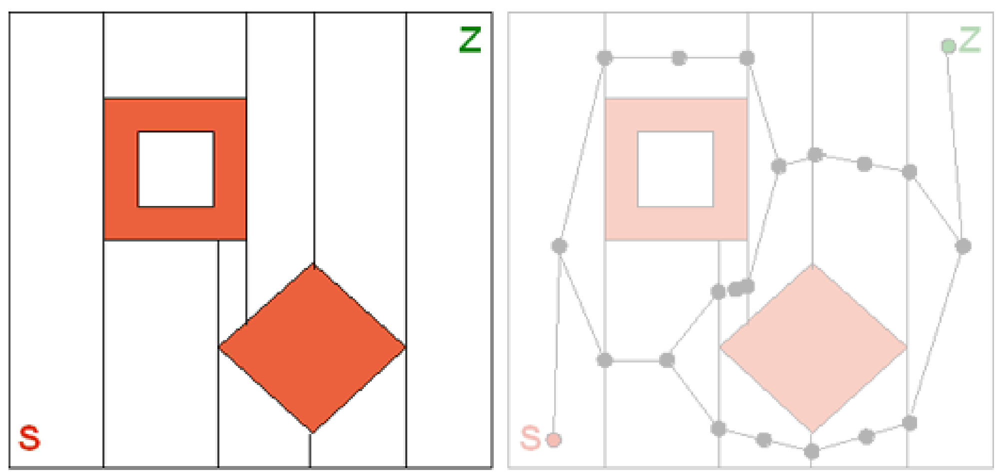
Roadmap

Cell decomp.

Potential field

Cell decomposition:
Exact decomposition (Trapezoidal)

– Decompose the free space with vertical lines through the vertices without intersecting with the forbidden space



Sampling-Based Motion Planning

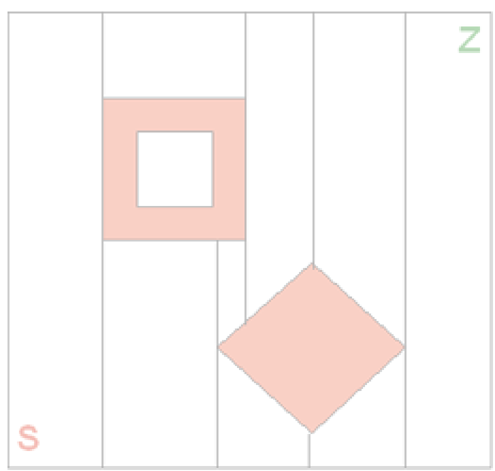
Roadmap

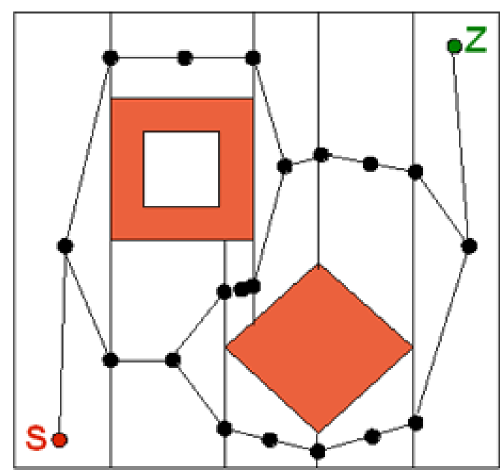
Cell decomp.

Potential field

Cell decomposition:
Exact decomposition (Trapezoidal)

– Add to the center of each segment and trapezoid a graph node





Sampling-Based Motion Planning

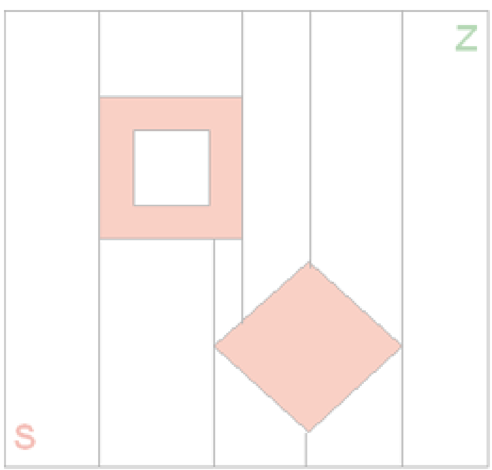
Roadmap

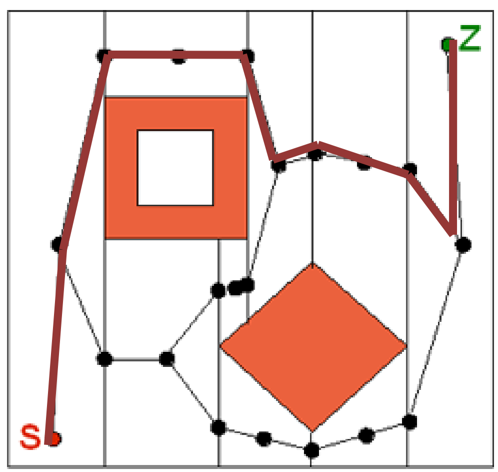
Cell decomp.

Potential field

Cell decomposition:
Exact decomposition (Trapezoidal)

– Find the shortest path through the obtained graph with a graph search algorithm





Sampling-Based Motion Planning

Roadmap

Cell decomp.

Potential field

Cell decomposition:
Approximate decomposition

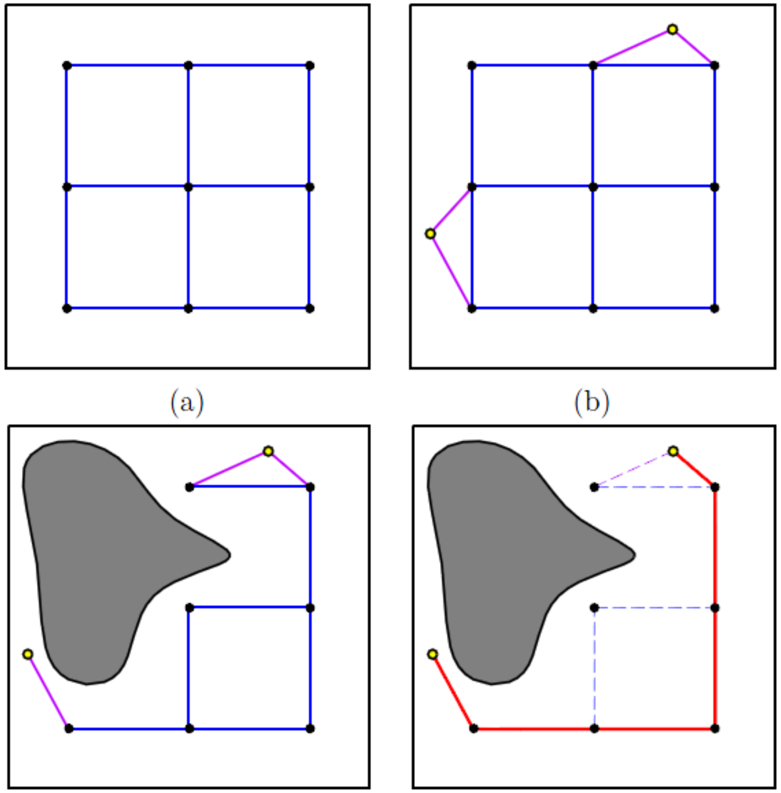
– One of the most convenient way to make sampling-based planning algorithms is to define a **grid** over C and conduct a discrete search algorithm

– **Neighborhoods:** For each grid point q we need to define the set of nearby grid points for which an edge may be constructed

Sampling-Based Motion Planning

Cell decomposition:
Approximate decomposition

– Once the grid and neighborhoods have been defined, a discrete planning problem is obtained



Sampling-Based Motion Planning

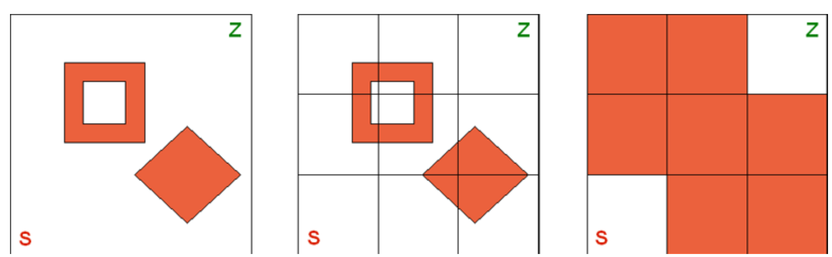
Roadmap

Cell decomp.

Potential field

Cell decomposition:
Approximate decomposition

– Decompose the C-Space with a start resolution to cell grid
– Each cell that intersect with obstacles is forbidden
– Is there some path existing?



Sampling-Based Motion Planning

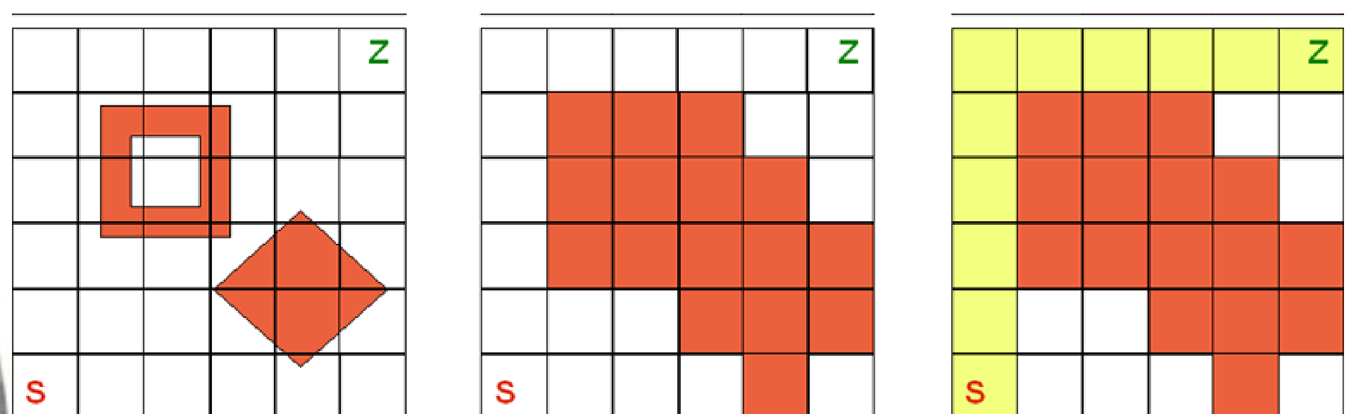
Roadmap

Cell decomp.

Potential field

Cell decomposition:
Approximate decomposition

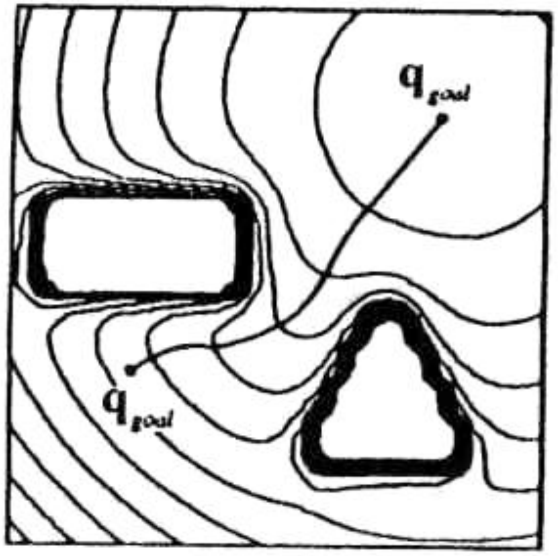
– If no Path is existing than refine the resolution until a solution is found



Sampling-Based Motion Planning

Potential field

Define a potential function over the free space that has a global minimum at the goal and follow the steepest descent of the potential function



Sampling-Based Motion Planning

Roadmap

Cell decomp.

Potential field

Potential field

–The goal location generates an **attractive potential** – pulling the robot towards the goal

– The obstacles generate a **repulsive potential** – pushing the robot far away from the obstacles

– The **negative gradient of the total potential** is treated as an artificial force applied to the robot

Sampling-Based Motion Planning

Potential field

The sum of the forces control of the robot

Artificial **Potential**

$$U(q) = \underbrace{U_{goal}(q)}_{\text{attractive potential}} + \underbrace{\sum U_{obstacles}(q)}_{\text{repulsive potential}}$$

Artificial **Force Field**

$$F(q) = -\nabla U(q)$$

Negative gradient

Sampling-Based Motion Planning

- Roadmap
- Cell decomp.
- Potential field

Potential field

Sampling-Based Motion Planning

Potential field

Pros

– Spatial paths are not preplanned and can be generated in real time

– Planning and controlling are merged into one function

Cons

–Trapped in local minima in the potential field

– Because of this limitation, commonly used for local path planning

Sampling-Based Motion Planning

Checking Path Segment

– Collision detection algorithms determine whether a configuration lies in $\mathcal{C}_{\text{free}}$

– Motion planning algorithms require that an entire path maps into $\mathcal{C}_{\text{free}}$

– The interface between the planner and collision detection usually involves validation of a path segment

Sampling-Based Motion Planning

Checking Path Segment

– For a Path $\tau_s : [0, 1] \rightarrow \mathcal{C}_{free}$ a sampling for the interval [0, 1] is calculated

– The collision checker is called only on the samples

Problem:

- How a Resolution can be found?
- How to guarantee that the places where the path is not sampled are collision-free?

Path Segment
Incremental Search
SBPA
RRT
PRM

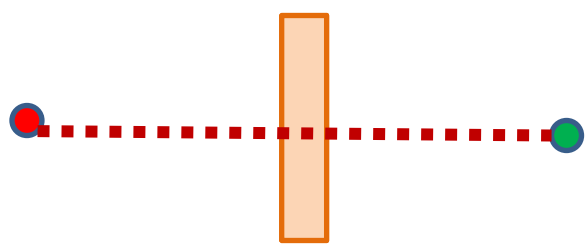
Checking Path Segment

– A fixed $\Delta q > 0$ is often chosen as the C-space step size
– Points $t_1, t_2 \in [0,1]$ are chosen close enough together to ensure that $\rho(\tau(t_1), \tau(t_2)) \leq \Delta q$, ρ is a metric on $\mathcal{C}$

Sampling-Based Motion Planning

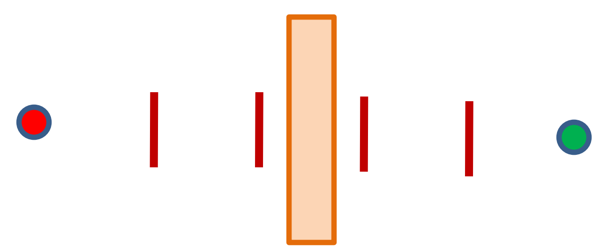
Checking Path Segment

– If Δq is too small, considerable time is wasted on collision checking (1)
– If Δq is too large, then there is a chance that the robot could jump through a thin obstacle (2)



(1)

(2)



Sampling-Based Motion Planning

Checking Path Segment

- Suppose that for a configuration $q(x_t,y_t,\theta)$ the collision detection algorithm indicates that $A(q)$ is at least d units away from collision

- Suppose that the next candidate configuration to be checked along τ is $q'(x't,y't,\theta')$

- If no point on A travels more than distance d when moving from q to q' along τ , then q' and all configurations between q and q' must be collision-free

Sampling-Based Motion Planning

Checking Path Segment

The bounds d can generally be used to set a step size Δq for collision checking that guarantees the intermediate points lie in $\mathcal{C}_{\text{free}}$

$$\{(x'_t, y'_t, \theta') \in \mathcal{C} \mid$$

$$|x_t - x'_t| + |y_t - y'_t| + r|\theta - \theta'| < d\}$$

Sampling-Based Motion Planning

Incremental Sampling and Searching

Most sample-based planning algorithms consisting of single-query model, witch means (q_I , q_G) is given only once per robot and obstacle set, following this template:

1. Initialization
2. Vertex Selection Method (VSM)
3. Local Planning Method (LPM)
4. Insert an Edge in the Graph
5. Check for a Solution
6. Return to Step 2

Sampling-Based Motion Planning

Incremental Sampling and Searching

1. **Initialization:**

   Let $\mathcal{G}(V,E)$ represent an undirected *search graph, for which*
   V contains at least one vertex and E contains no edges. Typically, *V* contains q_I , q_G, or both. In general, other points in $\mathcal{C}_{free}$ may be included

2. **Vertex Selection Method:**

   Choose a vertex $q_{cur} \in V$ for expansion

Sampling-Based Motion Planning

Incremental Sampling and Searching

3. **Local Planning Method (LPM):**

For some $q_{new} \in \mathcal{C}_{free}$ that may or may not be represented by a vertex in V attempt to construct a path $\tau_s : [0, 1] \rightarrow \mathcal{C}_{free}$ such that $\tau(0) = q_{cur}$ and $\tau(1) = q_{new}$. Using the methods of Slides 72-77 τ_s must be checked to ensure that it does not cause a collision. If this step fails to produce a collision-free path segment, then go to step 2.

Sampling-Based Motion Planning

Incremental Sampling and Searching

4. **Insert an Edge in the Graph**

   Insert τ_s into E, as an edge from q_{cur} to q_{new}. If q_{new} is not already in V, then it is inserted

5. **Check for a Solution**

   Determine whether $\mathcal{G}$ encodes a solution path. As in the discrete case, if there is a single search tree, then this is trivial; otherwise, it can become complicated and expensive

Sampling-Based Motion Planning

Incremental Sampling and Searching

6. **Return to Step 2:**
   Iterate unless a solution has been found or some termination condition is satisfied, in which case the algorithm reports failure

Sampling-Based Motion Planning

Path Segment

Incremental Search

SBPA

RRT

PRM

Sampling based Planning Algorithm:

There are several classes of algorithms based on the number of search trees:

– **Unidirectional (single-tree) methods**

– **Bidirectional Methods**

– **Multi-directional (more than two trees) methods**

Sampling-Based Motion Planning

Path Segment
Incremental Search
SBPA
RRT
PRM

Sampling based Planning Algorithm:

Unidirectional (single-tree) methods: A* (A-Star)

– Is a single-tree Method

– A* traverses the graph and follows the path with the lowest cost

– Keeps a sorted priority queue of alternate path segments along the way

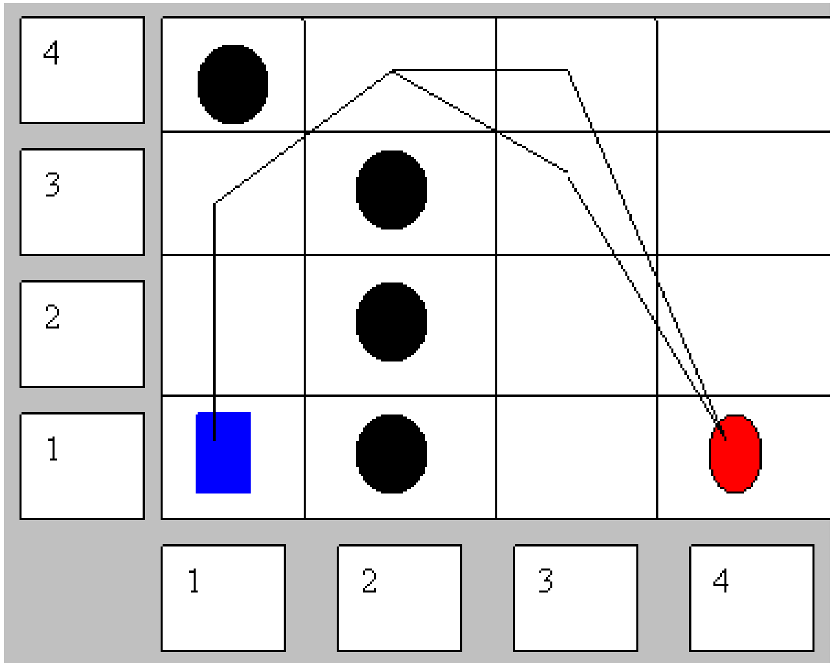
– If by adding a new Point, a segment of the path get a higher cost than another stored path segment, the lower-cost path segment will be followed

– The process continues until the goal is reached

Sampling-Based Motion Planning

- Path Segment
- Incremental Search
- SBPA
- RRT
- PRM

– The start position is (1, 1)
– The successive node (1, 2)
– There is no ambiguity, until the Robot reaches node (2, 4)
– The successor node can be determined by evaluating the cost to the target from both the nodes (3,4) and (3,3)



Sampling-Based Motion Planning

Sampling based Planning Algorithm:

– **Bidirectional Methods:** By a Bug-Trap or a challenging region problem, we better use a bidirectional approach

(a) (b)

Sampling-Based Motion Planning

Path Segment

Incremental Search

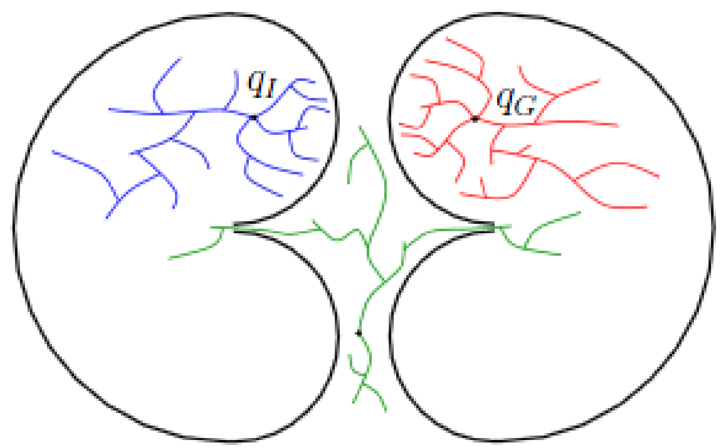
SBPA

RRT

PRM

Sampling based Planning Algorithm:

– **Multi-directional Methods:** For a double bug-trap, multi-directional search may be needed



Sampling-Based Motion Planning

Path Segment

Incremental Search

SBPA

RRT

PRM

Rapidly Exploring Dense Trees

– The idea is to incrementally construct a search tree that gradually improves the resolution but does not need to explicitly set any resolution parameters

– Instead of one long path, there are shorter paths that are organized into a tree

– If the sequence of samples is random, the resulting tree is called a *rapidly exploring random tree (RRT),* which indicate that a dense covering of the space is obtained

Rapidly Exploring Dense Trees

Basic **RRT** Algorithm:

1. Initially, start with the initial configuration as root of tree

2. Pick a random state in the configuration space

3. Find the closest node in the tree

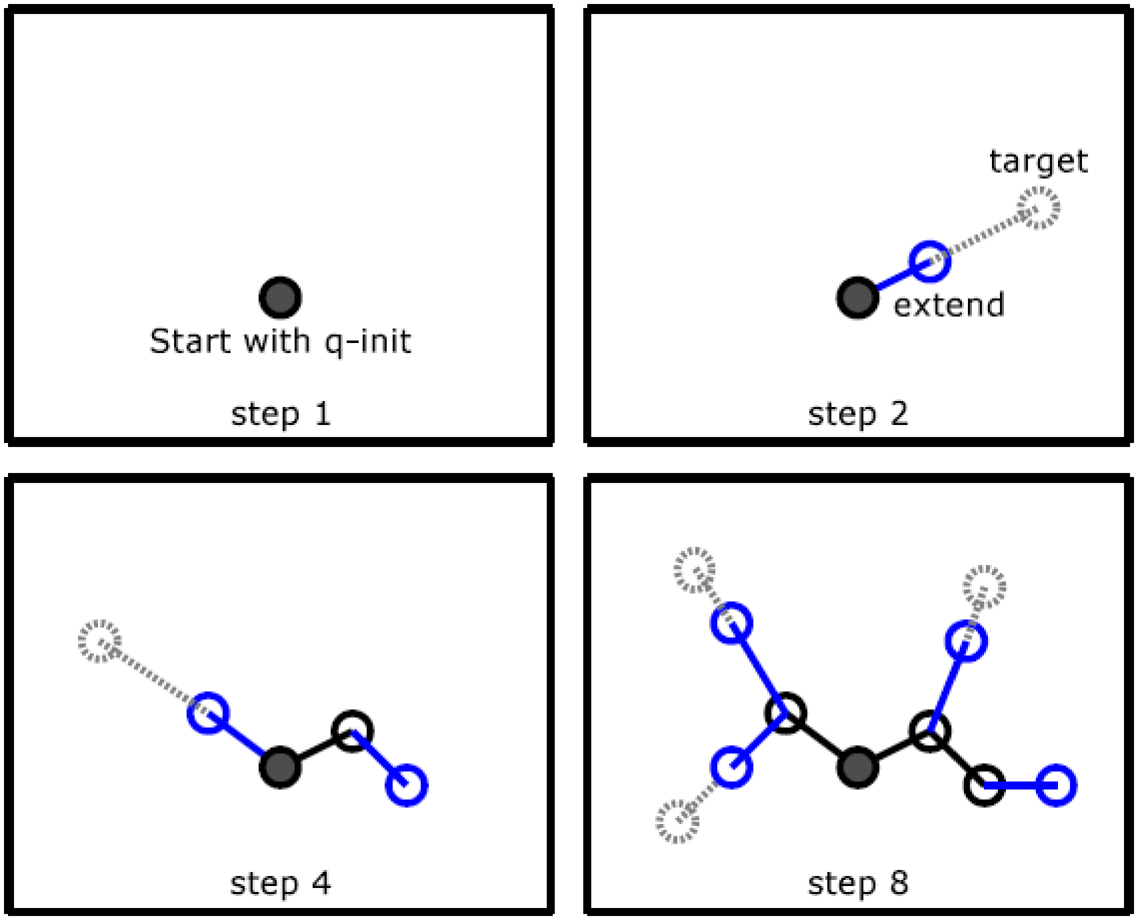
4. Extend that node toward the state if possible

5. Goto (2)

Sampling-Based Motion Planning

Rapidly Exploring Dense Trees

Basic **RRT** Algorithm:

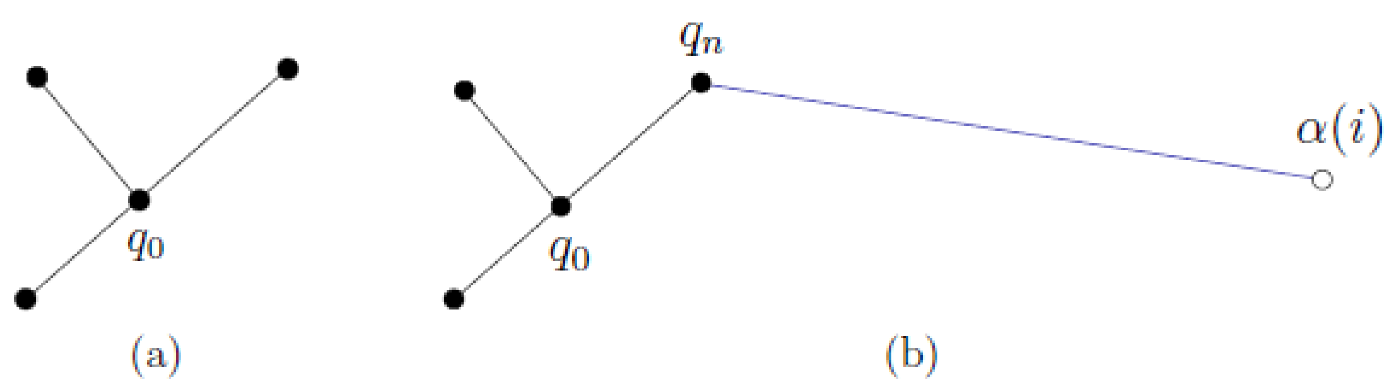


Sampling-Based Motion Planning

Rapidly Exploring Dense Trees

– The algorithm for constructing RDTs (which includes RRTs)

– It requires the availability of a dense sequence, α, and iteratively connects from α(i) to the nearest point among all those reached by $\mathcal{G}$



Sampling-Based Motion Planning

Path Segment

Incremental Search

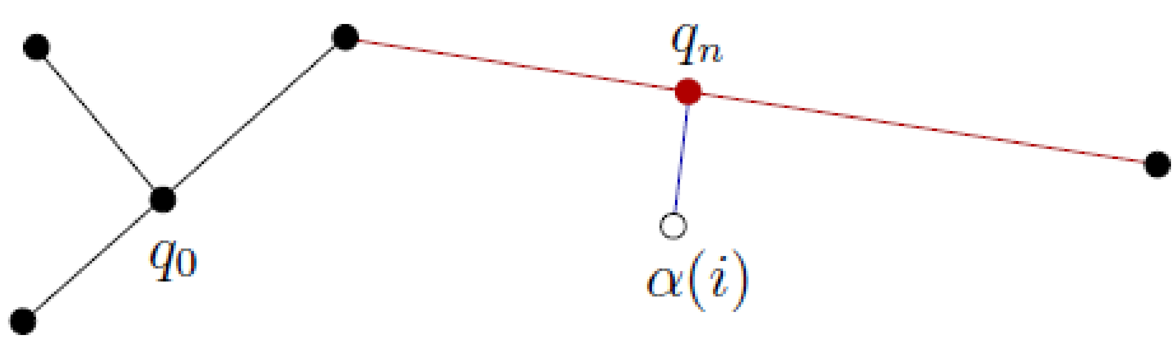
SBPA

RRT

PRM

Rapidly Exploring Dense Trees

If the nearest point in S lies in an edge (α), then the edge is split into two, and a new vertex is inserted into $\mathcal{G}$



Sampling-Based Motion Planning

Rapidly Exploring Dense Trees

– Several main branches are first constructed as it rapidly reaches the far corners of the space

– More and more area is filled in by smaller branches

– The tree gradually improves the resolution as the iterations continue

Sampling-Based Motion Planning

Path Segment

Incremental Search

SBPA

RRT

PRM

Rapidly Exploring Dense Trees

– This behavior turns out to be ideal for sampling-based motion planning

Sampling-Based Motion Planning

Path Segment

Incremental Search

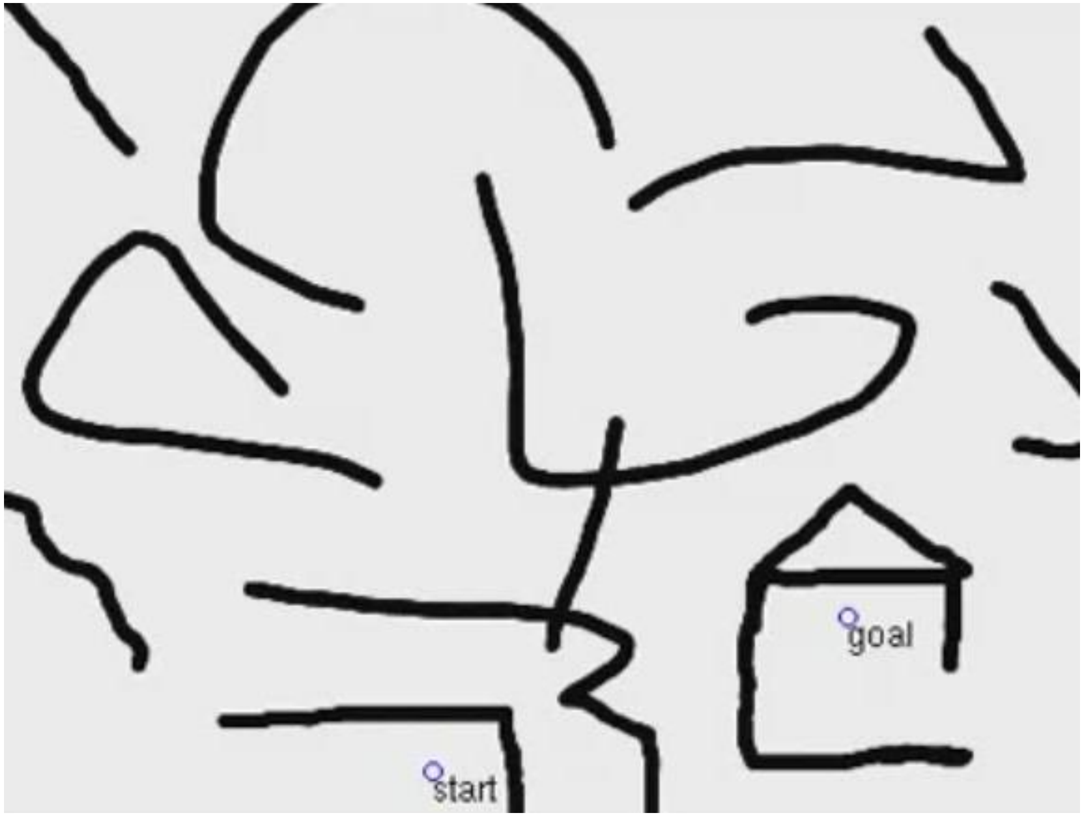
SBPA

RRT

PRM

Rapidly Exploring Dense Trees

– The RRT is dense in the limit (with probability one), which means that it gets arbitrarily to any point in the space



Sampling-Based Motion Planning

Path Segment
Incremental Search
SBPA
RRT
PRM

Probabilistic roadmaps (PRMs)

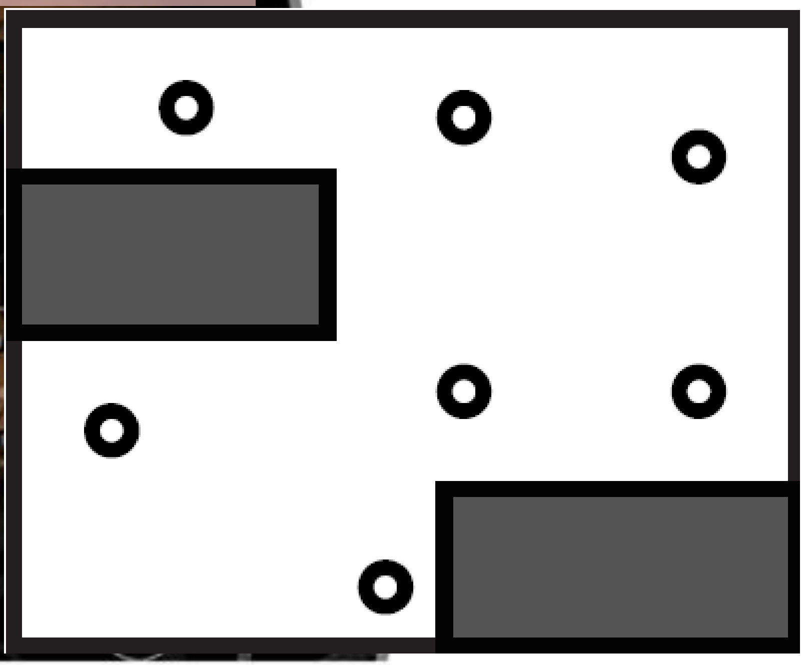
Separate planning into two stages:

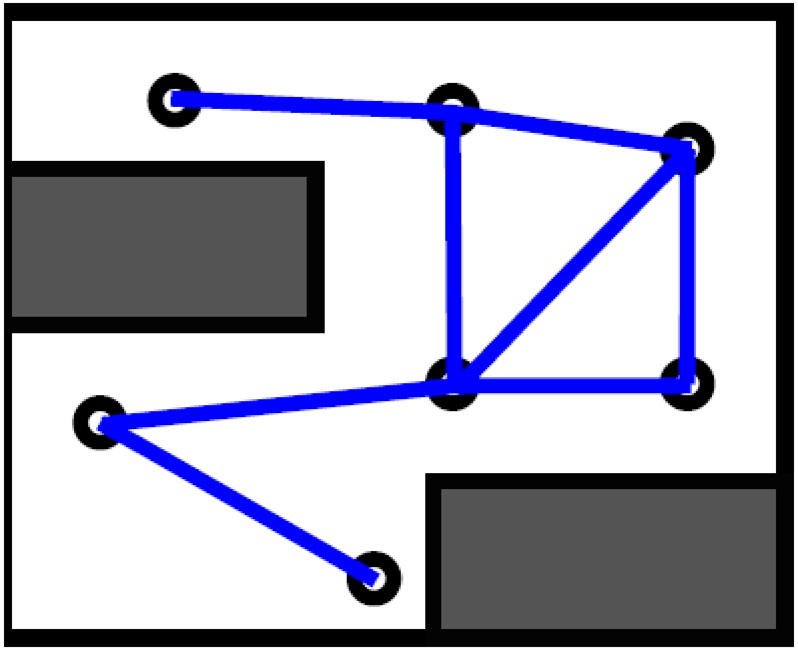
• *Learning Phase*
– find random sample of free configurations (vertices)
– attempt to connect pairs of nearby vertices with a local planner
– if a valid plan is found, add an edge to the graph

Sampling-Based Motion Planning

Probabilistic roadmaps (PRMs)

Learning Phase:





Sampling-Based Motion Planning

Probabilistic roadmaps (PRMs)

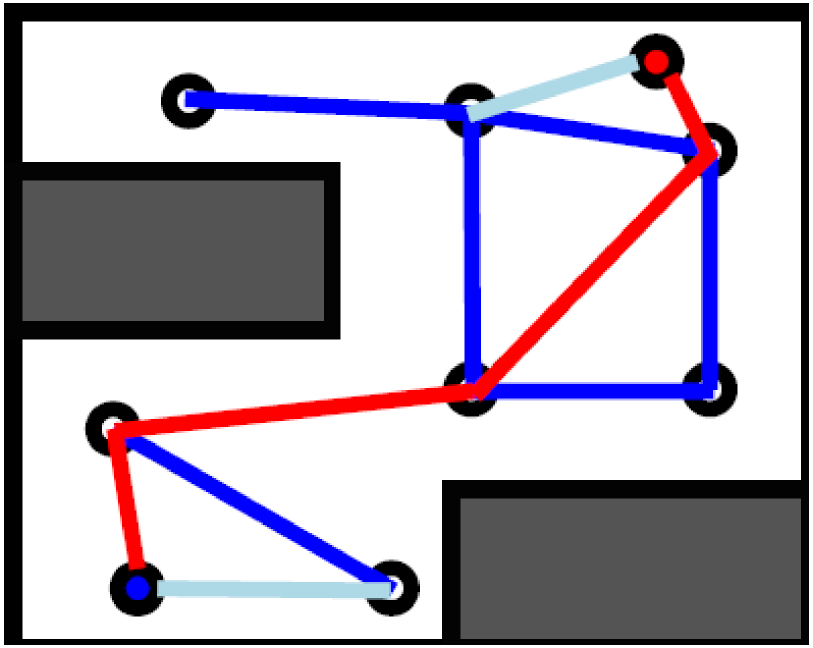
Separate planning into two stages:

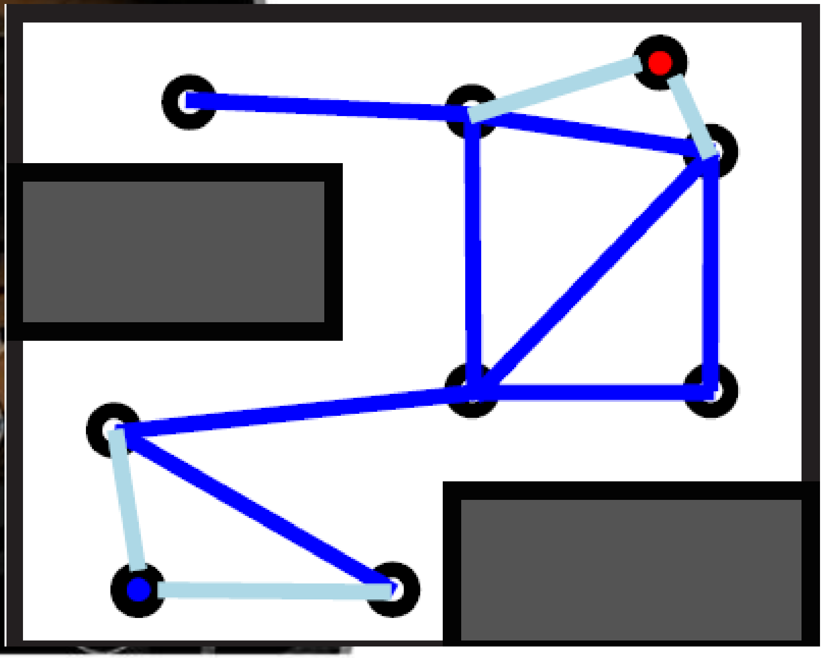
• *Query Phase*
– find local connections to graph from initial and goal positions
– search over roadmap graph using A* to find a plan

Path Segment
Incremental Search
SBPA
RRT
PRM

Probabilistic roadmaps (PRMs)

Query phase:





Overview

- Introduction
- Motion Planning
- Configuration Space
- Sampling-Based Motion Planning
- Comparaison of related Algorithms

Comparaison of related Algorithms

– **“Complete”:** If a plan exists, the path and the trajectory are found

– **Optimal:** The plan returned is optimal in reference to some metric

– **Efficient World Updates:** Can change world without recomputing everything

– **Efficient Query Updates:** Can change query without replanning from scratch

– **Good dof Scalability:** Scales well with increasing C-space dimensions

Comparaison of related Algorithms

Approach	Complete	Optimal	Efficient World Updates	Efficient Query Updates	Good DOF Scalability
A*	yes	grid	no	no	no
Visuality	yes	yes	no	no	no
Voronoi	yes	no	yes	yes	no
Potential Field	yes	no	no	no	yes
RRT	yes	no	semi	semi	yes
PRM	yes	graph	no	yes	yes

Sources

[1] D. Aarno, D. Kragic, and H. I. Christensen. Artificial potential biased probabilistic roadmap method. In *Proceedings IEEE International Conference on Robotics & Automation, 2004*

[2] R. Abgrall. Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes. *Communications on Pure and Applied Mathematics,* 49(12):1339–1373, December 1996.

[3] R. Abraham and J. Marsden. *Foundations of Mechanics. Addison-Wesley, Reading,*MA, 2002.

[4] PLANNING ALGORITHMS, Steven M. LaValle University of Illinois / Available for downloading at http://planning.cs.uiuc.edu/

[5] Wikipedia

Thank You for Your attention!