Robot Motion Planning I.

Jan Faigl

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

Lecture 9

A4M36PAH - Planning and Games

Robot Motio Planning I.

Introduction

Notation and Terminology

Sampling-based Motion Planning

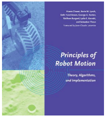
Randomized Sampling-Based Methods

Optimal Motion Planners

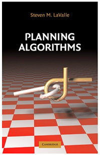
Literature

- Robot Motion Planning, *Jean-Claude Latombe*, Kluwer Academic Publishers, Boston, MA, 1991.

- Principles of Robot Motion: Theory, Algorithms, and Implementations, *H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun*, MIT Press, Boston, 2005.

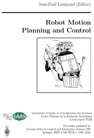


http://www.cs.cmu.edu/~biorobotics/book



- Planning Algorithms, *Steven M. LaValle*, Cambridge University Press, May 29, 2006.

http://planning.cs.uiuc.edu



- Robot Motion Planning and Control, *Jean-Paul Laumond*, Lectures Notes in Control and Information Sciences, 2009.

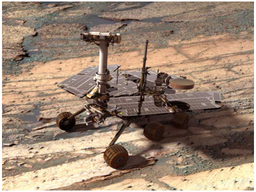
http://homepages.laas.fr/jpl/book.html

Robot Motion Planning – Motivational problem

- How to transform high-level task specification (provided by humans) into a low-level description suitable for controlling the actuators?

To develop algorithms *for such a transformation.*

The motion planning algorithms provide transformations how to move a robot (object) considering all operational constraints.

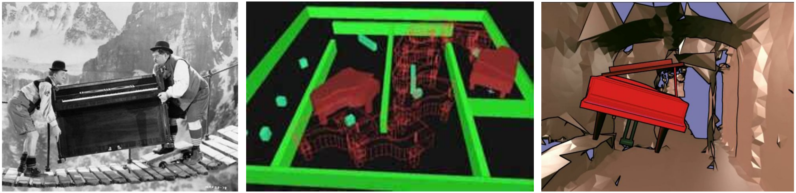


It encompasses several disciples, e.g., mathematics,

Piano Mover's Problem

A classical motion planning problem

Having a CAD model of the piano, model of the environment, the problem is how to move the piano from one place to another without hitting anything.

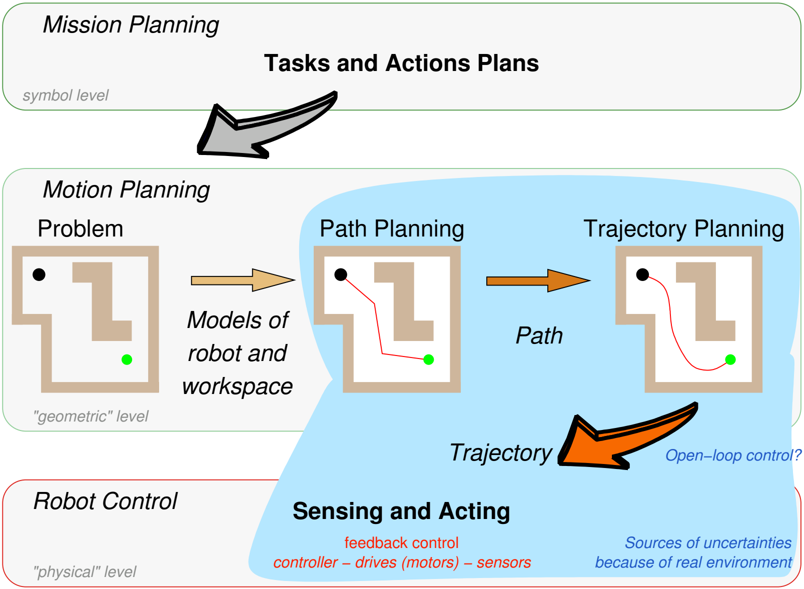


Basic motion planning algorithms are focused primarily on rotations and translations.

- We need notion of model representations and formal definition of the problem.
- Moreover, we also need a context about the problem and realistic assumptions.

The plans have to be admissible and feasible.

Robotic Planning Context



Real Mobile Robots

In a real deployment, the problem is a more complex.

- The world is changing
- Robots update the knowledge about the environment

  localization, mapping and navigation
- New decisions have to made
- A feedback from the environment

  Motion planning is a part of the mission replanning loop.

Josef Štrunc, Bachelor thesis, CTU, 2009.

An example of robotic mission:

Multi-robot exploration of unknown environment

How to deal with real-world complexity?

Relaxing constraints and considering realistic assumptions.

Notation

- $\mathcal{W}$ – **World model** describes the robot workspace and its boundary determines the obstacles $\mathcal{O}_i$.

  2D world, $\mathcal{W} = \mathbb{R}^2$

- A **Robot** is defined by its geometry, parameters (kinematics) and it is controllable by the motion plan.

- $\mathcal{C}$ – **Configuration space** ($\mathcal{C}$-space)

  A concept to describe possible configurations of the robot. The robot's configuration completely specify the robot location in $\mathcal{W}$ including specification of all degrees of freedom.

  E.g., a robot with rigid body in a plane $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times S^1$.

  - Let $\mathcal{A}$ be a subset of $\mathcal{W}$ occupied by the robot, $\mathcal{A} = \mathcal{A}(q)$.
  - A subset of $\mathcal{C}$ occupied by obstacles is

    $$\mathcal{C}_{obs} = \{q \in \mathcal{C} : \mathcal{A}(q) \cap \mathcal{O}_i, \forall i\}$$

  - Collision-free configurations are

    $$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$

Path / Motion Planning Problem

- **Path** is a continuous mapping in $\mathcal{C}$-space such that
$$\pi : [0,1] \rightarrow \mathcal{C}_{free}, \text{ with } \pi(0) = q_0, \text{ and } \pi(1) = q_f,$$
Only geometric considerations

- **Trajectory** is a path with explicate parametrization of time, e.g., accompanied by a description of the motion laws ($\gamma : [0,1] \rightarrow \mathcal{U}$, where $\mathcal{U}$ is robot's action space).

It includes dynamics.

$$[T_0, T_f] \ni t \rightsquigarrow \tau \in [0,1] : q(t) = \pi(\tau) \in \mathcal{C}_{free}$$

The planning problem is determination of the function $\pi(\cdot)$.

Additional requirements can be given:

- Smoothness of the path
- Kinodynamic constraints

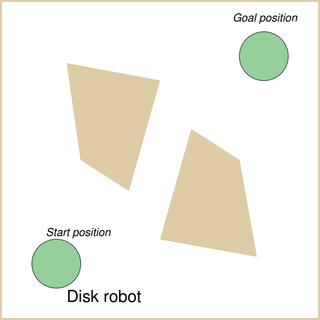
E.g., considering friction forces

- Optimality criterion

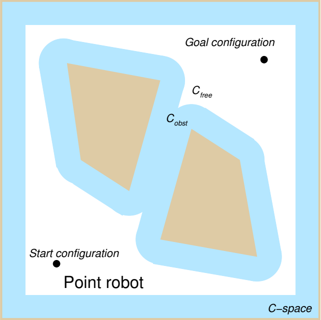
shortest vs fastest (length vs curvature)

Planning in $\mathcal{C}$-space

Robot motion planning robot for a disk robot with a radius ρ.



Motion planning problem in geometrical representation of $\mathcal{W}$

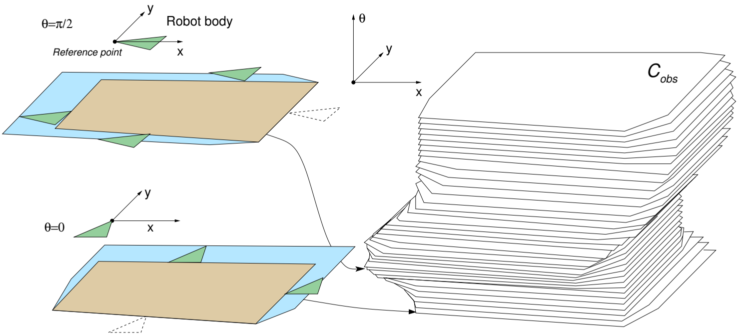


Motion planning problem in $\mathcal{C}$-space representation

$\mathcal{C}$-space has been obtained by enlarging obstacles by the disk $\mathcal{A}$ with the radius ρ.

By applying Minkowski sum: $\mathcal{O} \oplus \mathcal{A} = \{x + y \mid x \in \mathcal{O}, y \in \mathcal{A}\}$.

Example of $\mathcal{C}_{obs}$ for a Robot with Rotation



A simple 2D obstacle → has a complicated $\mathcal{C}_{obs}$

- Deterministic algorithms exist

  Requires exponential time in $\mathcal{C}$ dimension,

  J. Canny, PAMI, 8(2):200–209, 1986

- Explicit representation of $\mathcal{C}_{free}$ is impractical to compute.

Representation of $\mathcal{C}$-space

How to deal with continuous representation of $\mathcal{C}$-space?

Continuous Representation of $\mathcal{C}$-space

$\downarrow$

Discretization
processing critical geometric events, (random) sampling
roadmaps, cell decomposition, potential field

$\downarrow$

Graph Search Techniques
BFS, Gradient Search, A*

Planning Methods - Overview
(selected approaches)

- **Roadmap based methods**

  Create a connectivity graph of the free space.

  - Visibility graph

    (complete but impractical)

  - Cell decomposition
  - Voronoi diagram
- Discretization into a grid-based (or lattice-based) representation

  (resolution complete)

- Potential field methods *(complete only for a "navigation function", which is hard to compute in general)*

Classic path planning algorithms

- **Randomized sampling-based methods**
  - Creates a roadmap from connected random samples in $\mathcal{C}_{free}$
  - Probabilistic roadmaps

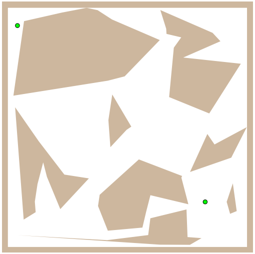
    samples are drawn from some distribution

  - Very successful in practice

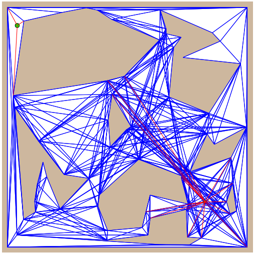
Visibility Graph

1. Compute visibility graph
2. Find the shortest path

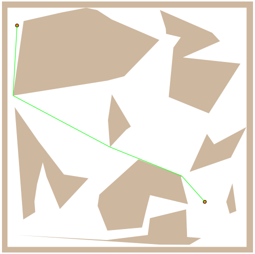
E.g., by Dijkstra's algorithm



Problem



Visibility graph



Found shortest path

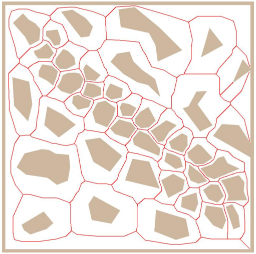
Constructions of the visibility graph:

- Naïve – all segments between n vertices of the map $O(n^3)$
- Using rotation trees for a set of segments – $O(n^2)$

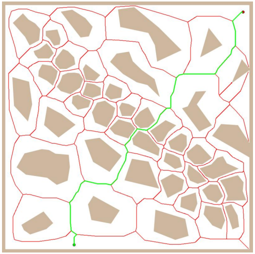
M. H. Overmars and E. Welzl, 1988

Voronoi Diagram

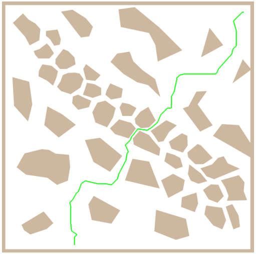
1. Roadmap is Voronoi diagram that maximizes clearance from the obstacles
2. Start and goal positions are connected to the graph
3. Path is found using a graph search algorithm



Voronoi diagram



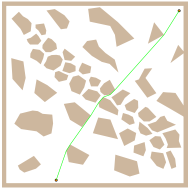
Path in graph



Found path

Visibility Graph vs Voronoi Diagram

Visibility graph

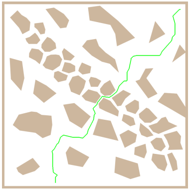


- Shortest path, but it is close to obstacles. We have to consider safety of the path.

  An error in plan execution can lead to a collision.

- Complicated in higher dimensions

Voronoi diagram



- It maximize clearance, which can provide conservative paths
- Small changes in obstacles can lead to large changes in the diagram
- Complicated in higher dimensions

A combination is called Visibility-Voronoi – R. Wein, J. P. van den Berg, D. Halperin, 2004

For higher dimensions we need other roadmaps.

Cell Decomposition

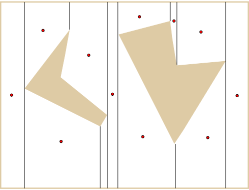
1. Decompose free space into parts.

   Any two points in a convex region can be directly connected by a segment.

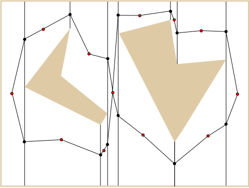
2. Create an adjacency graph representing the connectivity of the free space.

3. Find a path in the graph.

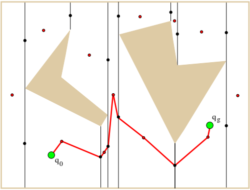
Trapezoidal decomposition



Centroids represent cells



Connect adjacency cells



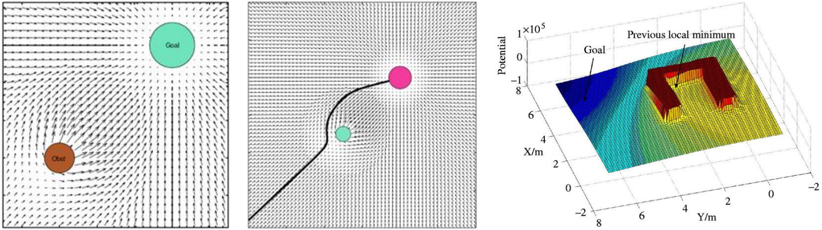
Find path in the adjacency graph

Other decomposition (e.g., triangulation) are possible.

Artificial Potential Field Method

- The idea is to create a function f that will provide a direction towards the goal for any configuration of the robot.
- Such a function is called navigation function and $-\nabla f(q)$ points to the goal.
- Create a potential field that will attract robot towards the goal q_f while obstacles will generate repulsive potential repelling the robot away from the obstacles.

The navigation function is a sum of potentials.



Such a potential function can have several local minima.

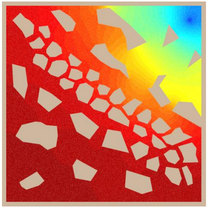
Avoiding Local Minima in Artificial Potential Field

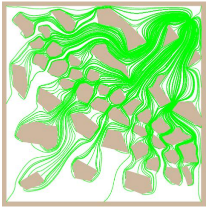
- Consider harmonic functions that have only one extremum

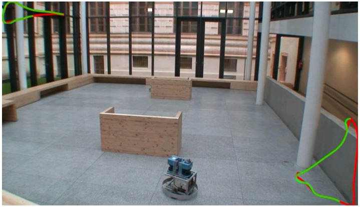
$$\nabla^2 f(q) = 0$$

- Finite element method

Dirichlet and Neumann boundary conditions







J. Mačák, Master thesis, CTU, 2009

Sampling-based Motion Planning

- Avoids explicit representation of the obstacles in $\mathcal{C}$-*space*
  - A "black-box" function is used to evaluate a configuration q is a collision free

    (E.g., based on geometrical models and testing collisions of the models)
- It creates a discrete representation of $\mathcal{C}_{free}$
- Configurations in $\mathcal{C}_{free}$ are sampled randomly and connected to a roadmap (**probabilistic roadmap**)
- Rather than full completeness they provides **probabilistic completeness** or resolution completeness

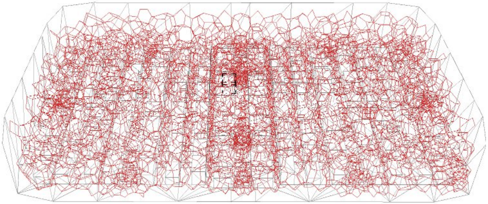
  Probabilistic complete algorithms: with increasing number of samples an admissible solution would be found (if exists)

Probabilistic Roadmaps

A discrete representation of the continuous $\mathcal{C}$-space generated by randomly sampled configurations in $\mathcal{C}_{free}$ that are connected into a graph.

- **Nodes** of the graph represent admissible configuration of the robot.
- **Edges** represent a feasible path (trajectory) between the particular configurations.

Probabilistic complete algorithms: with increasing number of samples an admissible solution would be found (if exists)



Having the graph, the final path (trajectory) is found by a graph search technique.

Probabilistic Roadmap Strategies

Multi-Query

- Generate a single roadmap that is then used for planning queries several times.
- An representative technique is **Probabilistic RoadMap (PRM)**

Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces
Lydia E. Kavraki and Petr Svestka and Jean-Claude Latombe and Mark H. Overmars,
IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

Single-Query

- For each planning problem constructs a new roadmap to characterize the subspace of $\mathcal{C}$-space that is relevant to the problem.
  - Rapidly-exploring Random Tree – RRT *LaValle, 1998*
  - Expansive-Space Tree – EST *Hsu et al., 1997*
  - Sampling-based Roadmap of Trees – SRT *(combination of multiple–query and single–query approaches)* *Plaku et al., 2005*

Multi-Query Strategy

Build a roadmap (graph) representing the environment

1. Learning phase
   1.1 Sample n points in $\mathcal{C}_{free}$
   1.2 Connect the random configurations using a local planner
2. Query phase
   2.1 Connect start and goal configurations with the PRM

   E.g., using a local planner

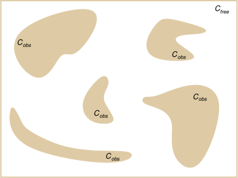
   2.2 Use the graph search to find the path

Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces
Lydia E. Kavraki and Petr Svestka and Jean-Claude Latombe and Mark H. Overmars,
IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

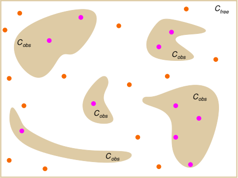
First planner that demonstrates ability to solve general planning problems in more than 4-5 dimensions.

PRM Construction

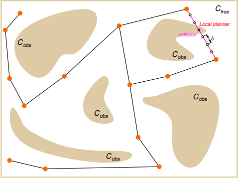
#1 Given problem domain



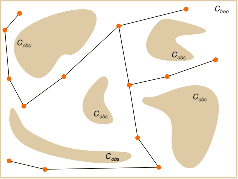
#2 Random configuration



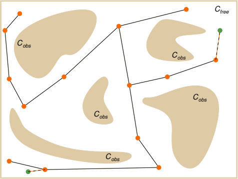
#3 Connecting samples



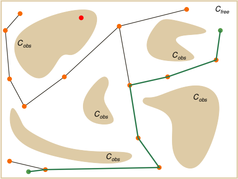
#4 Connected roadmap



#5 Query configurations

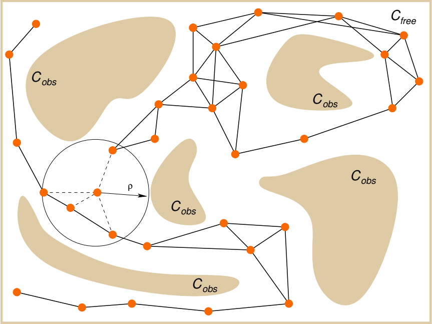


#6 Final found path



Practical PRM

- Incremental construction
- Connect nodes in a radius ρ
- Local planner tests collisions up to selected resolution δ
- Path can be found by Dijkstra's algorithm



What are the properties of the PRM algorithm?

We need a couple of more formalism.

Path Planning Problem Formulation

- Path planning problem is defined by a triplet
$$\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal}),$$
  - $\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs})$, $\mathcal{C} = (0,1)^d$, for $d \in \mathbb{N}$, $d \geq 2$
  - $q_{init} \in \mathcal{C}_{free}$ is the initial configuration (condition)
  - $\mathcal{G}_{goal}$ is the goal region defined as an open subspace of $\mathcal{C}_{free}$
- Function $\pi : [0,1] \rightarrow \mathbb{R}^d$ of *bounded variation* is called :
  - **path** if it is continuous;
  - **collision-free path** if it is path and $\pi(\tau) \in \mathcal{C}_{free}$ for $\tau \in [0,1]$;
  - **feasible** if it is collision-free path, and $\pi(0) = q_{init}$ and $\pi(1) \in \mathrm{cl}(\mathcal{Q}_{goal})$.
- A function π with the total variation $\mathrm{TV}(\pi) < \infty$ is said to have bounded variation, where $\mathrm{TV}(\pi)$ is the total variation
$$\mathrm{TV}(\pi) = \sup_{\{n \in \mathbb{N}, 0=\tau_0<\tau_1<\ldots<\tau_n=s\}} \sum_{i=1}^{n} |\pi(\tau_i) - \pi(\tau_{i-1})|$$
- The total variation $\mathrm{TV}(\pi)$ is de facto a path length.

Path Planning Problem

- **Feasible path planning**:

  For a path planning problem $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$
  - Find a feasible path $\pi : [0,1] \to \mathcal{C}_{free}$ such that $\pi(0) = q_{init}$ and $\pi(1) \in \mathrm{cl}(\mathcal{Q}_{goal})$, if such path exists.
  - Report failure if no such path exists.

- **Optimal path planning**:

  The optimality problem ask for a feasible path with the minimum cost.

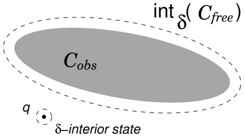
  For $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ and a cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$
  - Find a feasible path π^* such that $c(\pi^*) = \min\{c(\pi) : \pi \textit{ is feasible}\}$.
  - Report failure if no such path exists.

  The cost function is assumed to be monotonic and bounded, i.e., there exists k_c such that $c(\pi) \leq k_c \, \mathrm{TV}(\pi)$.

Probabilistic Completeness 1/2

First, we need **robustly feasible path planning problem** $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$.

- $q \in \mathcal{C}_{free}$ is *δ-interior state* **of** $\mathcal{C}_{free}$ if the closed ball of radius δ centered at q lies entirely inside $\mathcal{C}_{free}$.



- *δ-interior* of $\mathcal{C}_{free}$ is $\text{int}_\delta(\mathcal{C}_{free}) = \{q \in \mathcal{C}_{free} | \mathcal{B}_{/,\delta} \subseteq \mathcal{C}_{free}\}$.

  A collection of all δ-interior states.

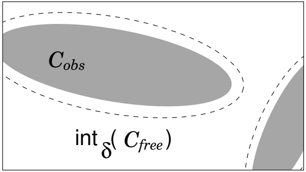
- A collision free path π has **strong δ-clearance**, if π lies entirely inside $\text{int}_\delta(\mathcal{C}_{free})$.
- $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ is *robustly feasible* if a solution exists and it is a feasible path with *strong δ-clearance*, for δ>0.

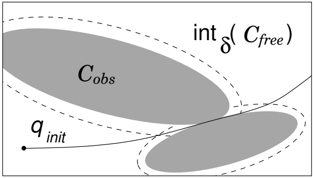
Probabilistic Completeness 2/2

An algorithm $\mathcal{ALG}$ is **probabilistically complete** if, for any *robustly feasible path planning problem* $\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$

$$\lim_{n \to 0} Pr(\mathcal{ALG} \text{ returns a solution to } \mathcal{P}) = 1.$$

- It is a "*relaxed*" notion of completeness
- Applicable only to problems with a **robust solution**.





We need some space, where random configurations can be sampled

Asymptotic Optimality 1/4

Asymptotic optimality relies on a notion of **weak δ-clearance**

Notice, we use strong δ-clearance for probabilistic completeness

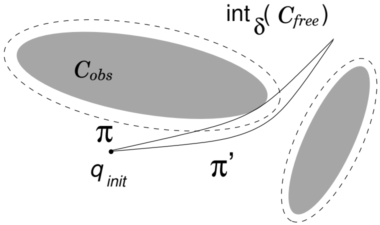
- Function $\psi : [0, 1] \rightarrow \mathcal{C}_{free}$ is called **homotopy**, if $\psi(0) = \pi_1$ and $\psi(1) = \pi_2$ and $\psi(\tau)$ is collision-free path for all $\tau \in [0, 1]$.
- A collision-free path π_1 is **homotopic** to π_2 if there exists homotopy function ψ.

A path homotopic to π can be continuously transformed to π through $\mathcal{C}_{free}$.

Asymptotic Optimality 2/4

- A collision-free path $\pi : [0, s] \rightarrow \mathcal{C}_{free}$ has **weak δ-clearance** if there exists a path π' that has **strong δ-clearance** and homotopy ψ with $\psi(0) = \pi$, $\psi(1) = \pi'$, and for all $\alpha \in (0, 1]$ there exists $\delta_\alpha > 0$ such that $\psi(\alpha)$ has strong δ-clearance.

Weak δ-clearance does not require points along a path to be at least a distance δ away from obstacles.



- A path π with a weak δ-clearance
- π' lies in $\mathrm{int}_\delta(\mathcal{C}_{free})$ and it is the same homotopy class as π

Asymptotic Optimality 3/4

- It is applicable with a **robust optimal solution** that can be obtained as a limit of robust (non-optimal) solutions.
- A collision-free path π^* is **robustly optimal solution** if it has *weak δ-clearance* and for any sequence of collision free paths $\{\pi_n\}_{n\in\mathbb{N}}$, $\pi_n \in \mathcal{C}_{free}$ such that $\lim_{n\to\infty} \pi_n = \pi^*$,

$$\lim_{n\to\infty} c(\pi_n) = c(\pi^*).$$

There exists a path with strong δ-clearance, and π^* is homotopic to such path and π^* is of **the lower cost**.

- Weak δ-clearance implies robustly feasible solution problem

(thus, probabilistic completeness)

Asymptotic Optimality 4/4

An algorithm $\mathcal{ALG}$ is **asymptotically optimal** if, for any path planning problem $\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ and cost function c that admit a robust optimal solution with the finite cost c^*

$$Pr\left(\left\{\lim_{i\to\infty} Y_i^{\mathcal{ALG}} = c^*\right\}\right) = 1.$$

- $Y_i^{\mathcal{ALG}}$ is the extended random variable corresponding to the minimum-cost solution included in the graph returned by $\mathcal{ALG}$ at the end of iteration i.

Properties of the PRM Algorithm

- Completeness for the standard PRM has not been provided when it was introduced
- A simplified version of the PRM (called sPRM) has been mostly studied
- sPRM is probabilistically complete

What are the differences between PRM and sPRM?

PRM vs simplified PRM (sPRM)

PRM

Input: q_{init}, number of samples n, radius ρ
Output: PRM – $G = (V, E)$

$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
for $i = 0, \ldots, n$ **do**
    $q_{rand} \leftarrow$ SampleFree;
    $U \leftarrow$ Near$(G = (V, E), q_{rand}, \rho)$;
    $V \leftarrow V \cup \{q_{rand}\}$;
    foreach $u \in U$, *with increasing* $||u - q_r||$ **do**
        if q_{rand} *and* u *are not in the same connected component of* $G = (V, E)$ **then**
            if CollisionFree(q_{rand}, u) **then**
                $E \leftarrow E \cup \{(q_{rand}, u), (u, q_{rand})\}$;

return $G = (V, E)$;

sPRM Algorithm

Input: q_{init}, number of samples n, radius ρ
Output: PRM – $G = (V, E)$

$V \leftarrow \{q_{init}\} \cup \{\text{SampleFree}_i\}_{i=1,\ldots,n-1}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
    $U \leftarrow$Near$(G = (V, E), v, \rho) \setminus \{v\}$;
    foreach $u \in U$ **do**
        if CollisionFree(v, u) **then**
            $E \leftarrow E \cup \{(v, u), (u, v)\}$;

return $G = (V, E)$;

There are several ways for the set U of vertices to connect them

- k-nearest neighbors to v
- variable connection radius ρ as a function of n

PRM – Properties

- sPRM (simplified PRM)
  - **Probabilistically complete and asymptotically optimal**
  - Processing complexity $O(n^2)$
  - Query complexity $O(n^2)$
  - Space complexity $O(n^2)$
- Heuristics practically used are usually not probabilistic complete
  - k-nearest sPRM is not probabilistically complete
  - variable radius sPRM is not probabilistically complete

Based on analysis of Karaman and Frazzoli

PRM algorithm:

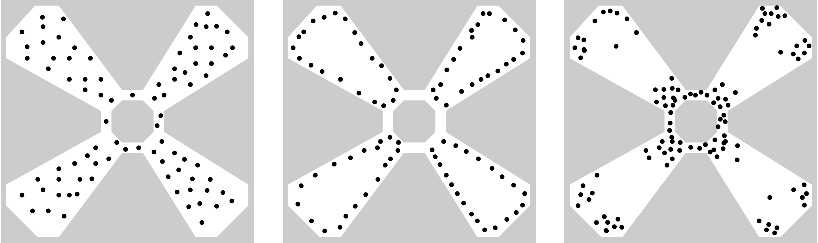
\+ Has very simple implementation

\+ Completeness (for sPRM)

− Differential constraints (car-like vehicles) are not straightforward

Comments about Random Sampling 1/2

- Different sampling strategies (distributions) may be applied



- Notice, one of the main issue of the randomized sampling-based approaches is the narrow passage
- Several modifications of sampling based strategies have been proposed in the last decades

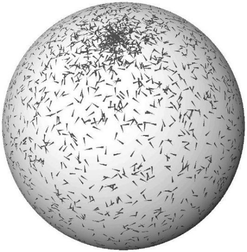
Comments about Random Sampling 2/2

- A solution can be found using only a few samples.

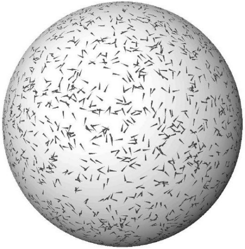
Do you know the Oraculum? (from Alice in Wonderland)

- Sampling strategies are important
  - Near obstacles
  - Narrow passages
  - Grid-based
  - Uniform sampling must be carefully considered.

James J. Kuffner, Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning, ICRA, 2004.



Naïve sampling



Uniform sampling of SO(3) using Euler angles

Rapidly Exploring Random Tree (RRT)

Single–Query algorithm

- It incrementally builds a graph (tree) towards the goal area.

It does not guarantee precise path to the goal configuration.

1. Start with the initial configuration q_0, which is a root of the constructed graph (tree)
2. Generate a new random configuration q_{new} in $\mathcal{C}_{free}$
3. Find the closest node q_{near} to q_{new} in the tree

   E.g., using KD-tree implementation like ANN or FLANN libraries

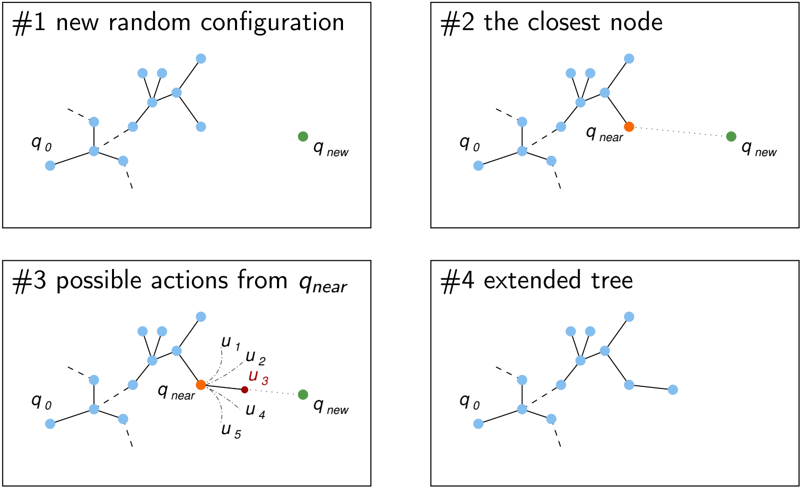
4. Extend q_{near} towards q_{new}

   Extend the tree by a small step, but often a direct control $u \in \mathcal{U}$ that will move robot the position closest to q_{new} is selected (applied for δt).

5. Go to Step 2, until the tree is within a sufficient distance from the goal configuration

   Or terminates after dedicated running time.

RRT Construction



RRT Algorithm

- Motivation is a single query and *control-based* path finding
- It incrementally builds a graph (tree) towards the goal area.

RRT Algorithm

Input: q_{init}, number of samples n

Output: Roadmap $G = (V, E)$

$V \leftarrow \{q_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $q_{rand} \leftarrow$ SampleFree;
  $q_{nearest} \leftarrow$ Nearest($G = (V, E), q_{rand}$);
  $q_{new} \leftarrow$ Steer($q_{nearest}, q_{rand}$);
  if CollisionFree($q_{nearest}, q_{new}$) **then**
    $V \leftarrow V \cup \{x_{new}\}$; $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$;

return $G = (V, E)$;

Extend the tree by a small step, but often a direct control $u \in \mathcal{U}$ that will move robot to the position closest to q_{new} is selected (applied for dt).

Rapidly-exploring random trees: A new tool for path planning
S. M. LaValle,
Technical Report 98-11, Computer Science Dept., Iowa State University, 1998

Properties of RRT Algorithms

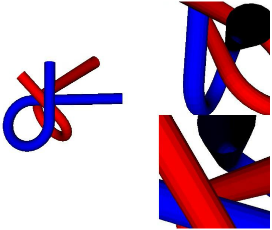
- Rapidly explores the space

  q_{new} will more likely be generated in large not yet covered parts.
- Allows considering kinodynamic/dynamic constraints (during the expansion).
- Can provide trajectory or a sequence of direct control commands for robot controllers.
- A collision detection test is usually used as a "black-box".

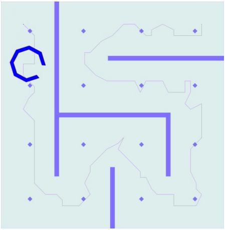
  E.g., RAPID, Bullet libraries.
- Similarly to PRM, RRT algorithms have poor performance in narrow passage problems.
- RRT algorithms provides feasible paths.

  It can be relatively far from optimal solution, e.g., according to the length of the path.
- Many variants of RRT have been proposed.

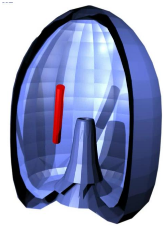
RRT – Examples 1/2



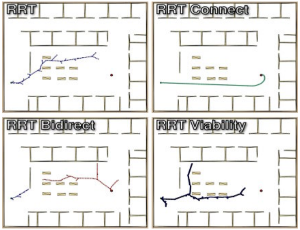
Alpha puzzle benchmark



Apply rotations to reach the goal



Bugtrap benchmark

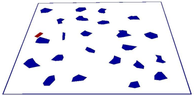


Variants of RRT algorithms

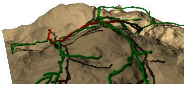
Courtesy of V. Vonásek and P. Vaněk

RRT – Examples 2/2

- Planning for a car-like robot

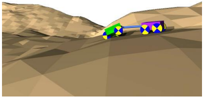


- Planning on a 3D surface



- Planning with dynamics

(friction forces)



Courtesy of V. Vonásek and P. Vaněk

Car-Like Robot

- Configuration

$$\vec{\boldsymbol{x}} = \begin{pmatrix} x \\ y \\ \phi \end{pmatrix}$$

position and orientation

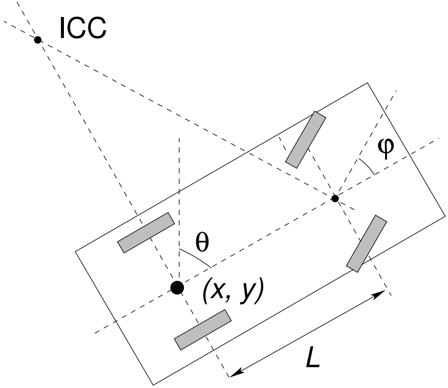
- Controls

$$\vec{\boldsymbol{u}} = \begin{pmatrix} v \\ \varphi \end{pmatrix}$$

forward velocity, steering angle

- System equation

$$\begin{aligned} \dot{x} &= v \cos\phi \\ \dot{y} &= v \sin\phi \\ \dot{\varphi} &= \frac{v}{L} \tan\varphi \end{aligned}$$



Kinematic constraints $\dim(\vec{\boldsymbol{u}}) < \dim(\vec{\boldsymbol{x}})$

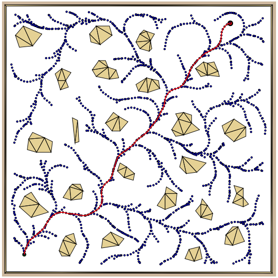
Differential constraints on possible $\dot{q}$*:*

$$\dot{x}\sin(\phi) - \dot{y}\cos(\phi) = 0$$

Control-Based Sampling

- Select a configuration q from the tree T of the current configurations
- Pick a control input $\overrightarrow{u} = (v, \varphi)$ and integrate system (motion) equation over a short period

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta \varphi \end{pmatrix} = \int_{t}^{t+\Delta t} \begin{pmatrix} v \cos \phi \\ v \sin \phi \\ \frac{v}{L} \tan \varphi \end{pmatrix} dt$$



- If the motion is collision-free, add the endpoint to the tree

E.g., considering k configurations for $k\delta t = dt$.

RRT and Quality of Solution

- RRT provides a feasible solution without quality guarantee
  Despite of that, it is successfully used in many practical applications
- In 2011, a systematical study of the asymptotic behaviour of randomized sampling-based planners has been published
  It shows, that in some cases, they converge to a non-optimal value with a probability 1.

Sampling-based algorithms for optimal motion planning
Sertac Karaman, Emilio Frazzoli
International Journal of Robotic Research, 30(7):846–894, 2011.

http://sertac.scripts.mit.edu/rrtstar

RRT and Quality of Solution 1/2

- Let Y_i^{RRT} be the cost of the best path in the RRT at the end of iteration i.
- Y_i^{RRT} converges to a random variable

$$\lim_{i \to \infty} Y_i^{RRT} = Y_\infty^{RRT}.$$

- The random variable Y_∞^{RRT} is sampled from a distribution with zero mass at the optimum, and

$$Pr[Y_\infty^{RRT} > c^*] = 1.$$

Karaman and Frazzoli, 2011

- The best path in the RRT converges to a sub-optimal solution almost surely.

RRT and Quality of Solution 2/2

- RRT does not satify a necessary condition for the asymptotic optimality
  - For $0 < R < \inf_{q \in \mathcal{Q}_{goal}} ||q - q_{init}||$, the event $\{\lim_{n\to\infty} Y_n^{RTT} = c^*\}$ occurs only if the k-th branch of the RRT contains vertices outside the R-ball centered at q_{init} for infinitely many k.

See Appendix B in Karaman&Frazzoli, 2011

- It is required the root node will have infinitely many subtrees that extend at least a distance ϵ away from q_{init}

The sub-optimality is caused by disallowing new better paths to be discovered.

Rapidly-exploring Random Graph (RRG)

RRG Algorithm

Input: q_{init}, number of samples n

Output: $G = (V, E)$

$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
for $i = 0, \ldots, n$ **do**
    $q_{rand} \leftarrow \text{SampleFree}$;
    $q_{nearest} \leftarrow \text{Nearest}(G = (V, E), q_{rand})$;
    $q_{new} \leftarrow \text{Steer}(q_{nearest}, q_{rand})$;
    if $\text{CollisionFree}(q_{nearest}, q_{new})$ **then**
        $\mathcal{Q}_{near} \leftarrow \text{Near}(G = (V, E), q_{new}, \min\{\gamma_{RRG}(\log(\text{card}(V))/\,\text{card}(V))^{1/d}, \eta\})$;
        $V \leftarrow V \cup \{q_{new}\}$; $E \leftarrow E \cup \{(q_{nearest}, q_{new}), (q_{new}, q_{nearest})\}$;
        foreach $q_{near} \in \mathcal{Q}_{near}$ **do**
            if $\text{CollisionFree}(q_{near}, q_{new})$ **then**
                $E \leftarrow E \cup \{(q_{rand}, u), (u, q_{rand})\}$;

return $G = (V, E)$;

Proposed by Karaman and Frazzoli (2011). Theoretical results are related to properties of ***Random Geometric Graphs (RGG)*** *introduced by Gilbert (1961) and further studied by Penrose (1999).*

RRG Expansions

- At each iteration, RRG tries to connect new sample to the all vertices in the r_n ball centered at it.
- The ball of radius

$$r(\text{card}(V)) = \min\left\{\gamma_{RRG}\left(\frac{\log(\text{card}(V))}{\text{card}(V)}\right)^{1/d}, \eta\right\}$$

where

  - η is the constant of the local steering function
  - $\gamma_{RRG} > \gamma^*_{RRG} = 2(1 + 1/d)^{1/d}(\mu(\mathcal{C}_{free})/\xi_d)^{1/d}$
    - d – dimension of the space;
    - $\mu(\mathcal{C}_{free})$ – Lebesgue measure of the obstacle–free space;
    - ξ_d – volume of the unit ball in d-dimensional Euclidean space.
- The connection radius decreases with n
- The rate of decay ≈ the average number of connections attempted is proportional to $\log(n)$

RRG Properties

- Probabilistically complete
- Asymptotically optimal
- Complexity is $O(\log n)$

(per one sample)

- Computational efficiency and optimality
  - Attempt connection to $\Theta(\log n)$ nodes at each iteration;

    in average

    - Reduce volume of the "connection" ball as $\log(n)/n$;
    - Increase the number of connections as $\log(n)$.

Other Variants of the Optimal Motion Planning

- **PRM*** – it follows standard PRM algorithm where connections are attempted between roadmap vertices that are within connection radius r as a function of n

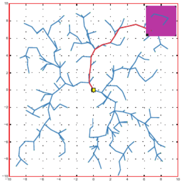
$$r(n) = \gamma_{PRM}(\log(n)/n)^{1/d}$$

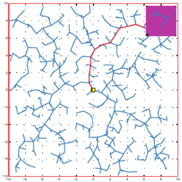
- **RRT*** – a modification of the RRG, where cycles are avoided

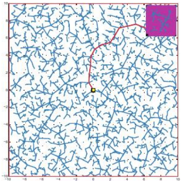
  A tree version of the RRG

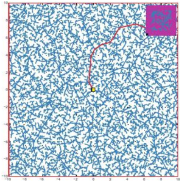
  - A tree roadmap allows to consider non-holonomic dynamics and kinodynamic constraints.
  - It is basically RRG with "rerouting" the tree when a better path is discovered.

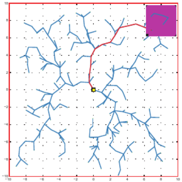
Example of Solution 1/2

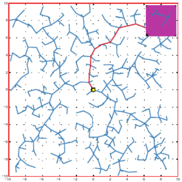

RRT, *n*=250

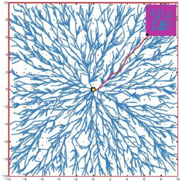

RRT, *n*=500

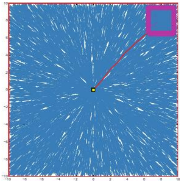

RRT, *n*=2500


RRT, *n*=10000


RRT*, *n*=250

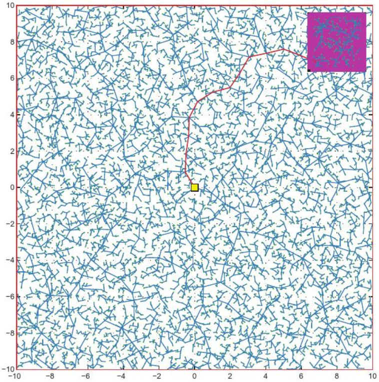

RRT*, *n*=500


RRT*, *n*=2500

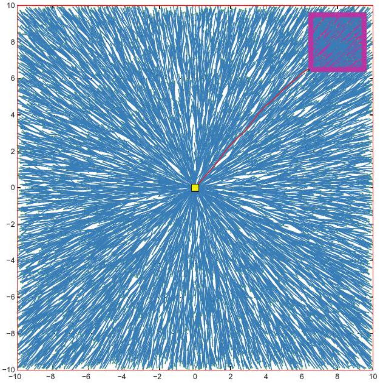

RRT*, *n*=10000

Karaman & Frazzoli, 2011

Example of Solution 2/2



RRT, n=20000



RRT*, n=20000

Overview of Randomized Sampling-based Algorithms

Algorithm	Probabilistic Completeness	Asymptotic Optimality
sPRM	✔	✘
k-nearest sPRM	✘	✘
RRT	✔	✘
RRG	✔	✔
PRM*	✔	✔
RRT*	✔	✔

Notice, k-nearest variants of RRG, PRM, and RRT* are complete and optimal as well.*

Summary

- Introduction to motion planning
- Overview of sampling-based planning methods
  - Basic roadmap methods
    - Visibility graph
    - Voronoi diagram
    - Cell decomposition
  - Artificial potential field method
- Randomized Sampling-based Methods and their properties (PRM, sPRM, RRT)
- Optimal Motion Planners (RRG, PRM*, RRT*)