Randomized Sampling-based Motion Planning Methods

Jan Faigl

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

Lecture 05

B4M36UIR – Artificial Intelligence in Robotics

Overview of the Lecture

- Part 1 – Randomized Sampling-based Motion Planning Methods
  - Sampling-Based Methods
  - Probabilistic Road Map (PRM)
  - Characteristics
  - Rapidly Exploring Random Tree (RRT)

Part I

Part 1 – Roadmap-based Planning Methods

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

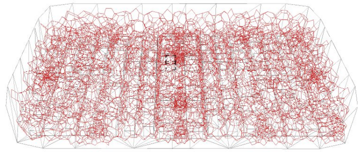
  Probabilistic complete algorithms: with increasing number of samples an admissible solution would be found (if exists)

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

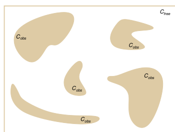
   2.2 Use the graph search to find the path

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.

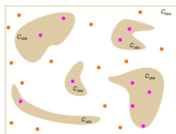
First planner that demonstrates ability to solve general planning problems in more than 4-5 dimensions.

PRM Construction

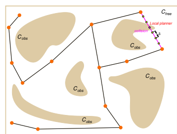
#1 Given problem domain



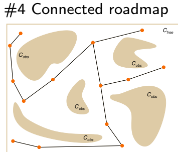
#2 Random configuration



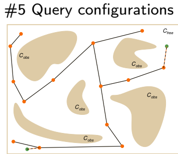
#3 Connecting samples



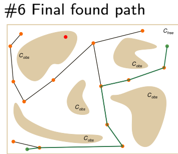
#4 Connected roadmap



#5 Query configurations

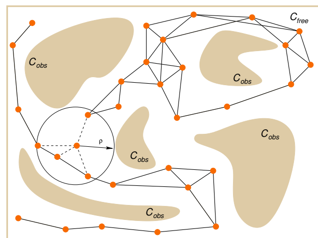


#6 Final found path



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
$$\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal}),$$
  - $\mathcal{C}_{free} = \text{cl}(\mathcal{C} \setminus \mathcal{C}_{obs})$, $\mathcal{C} = (0,1)^d$, 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 $\pi : [0,1] \to \mathbb{R}^d$ 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 $\pi(1) \in \text{cl}(\mathcal{Q}_{goal})$.
- A function π with the total variation $\text{TV}(\pi) < \infty$ is said to have bounded variation, where $\text{TV}(\pi)$ is the total variation
$$\text{TV}(\pi) = \sup_{\{n\in\mathbb{N}, 0=\tau_0<\tau_1<\ldots<\tau_n=s\}} \sum_{i=1}^{n} |\pi(\tau_i) - \pi(\tau_{i-1})|$$
- The total variation $\text{TV}(\pi)$ is de facto a path length.

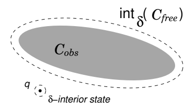
Path Planning Problem

- **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 $\pi(1) \in \text{cl}(\mathcal{Q}_{goal})$, 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 $c(\pi^*) = \min\{c(\pi) : \pi \text{ is feasible}\}$.
  - 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 $c(\pi) \leq k_c\, \text{TV}(\pi)$.

Probabilistic Completeness 1/2

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

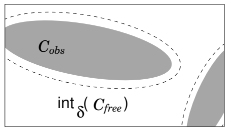


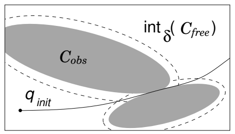
- $q \in \mathcal{C}_{free}$ is *δ-interior state of $\mathcal{C}_{free}$* if the closed ball of radius δ centered at q lies entirely inside $\mathcal{C}_{free}$.
- *δ-interior* of $\mathcal{C}_{free}$ is $\text{int}_\delta(\mathcal{C}_{free}) = \{q \in \mathcal{C}_{free} | \mathcal{B}_{/,\delta} \subseteq \mathcal{C}_{free}\}$.
  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

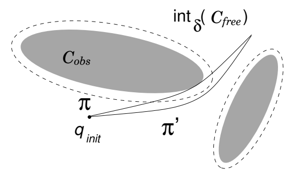
- Function $\psi : [0,1] \to \mathcal{C}_{free}$ is called **homotopy**, if $\psi(0) = \pi_1$ and $\psi(1) = \pi_2$ and $\psi(\tau)$ is collision-free path for all $\tau \in [0,1]$.
- A collision-free path π_1 is **homotopic** to π_2 if there exists homotopy function ψ.

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

Asymptotic Optimality 2/4

- A collision-free path $\pi : [0,s] \to \mathcal{C}_{free}$ 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 $\text{int}_\delta(\mathcal{C}_{free})$ 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

Vstup: q_{init}, number of samples n, radius ρ
Výstup: 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

Vstup: q_{init}, number of samples n, radius ρ
Výstup: 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

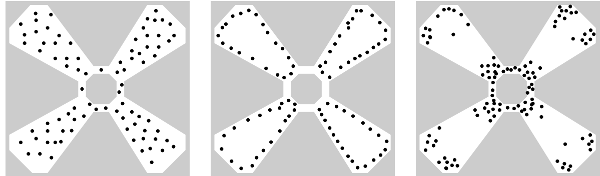
Based on analysis of Karaman and Frazzoli

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

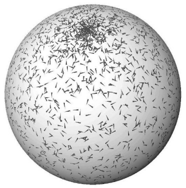
Comments about Random Sampling 2/2

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

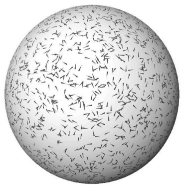
Do you know the Oraculum? (from Alice in Wonderland)

- 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.

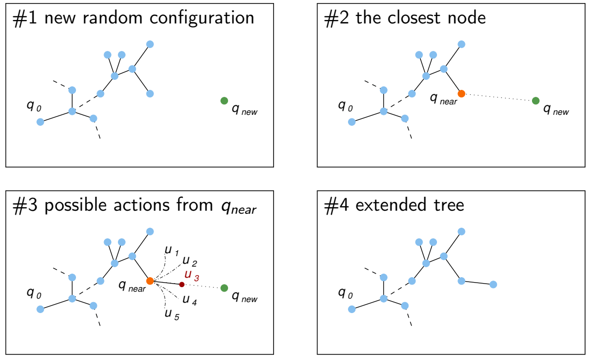
1. Start with the initial configuration q_0, which is a root of the constructed graph (tree)
2. Generate a new random configuration q_{new} in $\mathcal{C}_{free}$
3. Find the closest node q_{near} to q_{new} in the tree

   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

Vstup: q_{init}, number of samples n
Výstup: 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

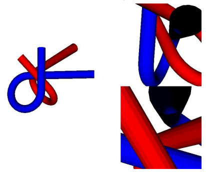
- Rapidly explores the space

  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".

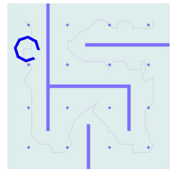
  E.g., RAPID, Bullet libraries.
- Similarly to PRM, RRT algorithms have poor performance in narrow passage problems.
- RRT algorithms provides feasible paths.

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

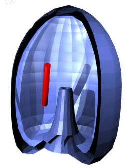
RRT – Examples 1/2



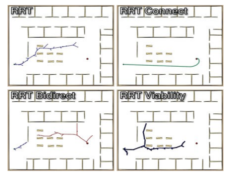
Alpha puzzle benchmark



Apply rotations to reach the goal



Bugtrap benchmark

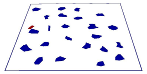


Variants of RRT algorithms

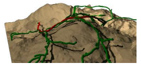
Courtesy of V. Vonásek and P. Vaněk

RRT – Examples 2/2

- Planning for a car-like robot

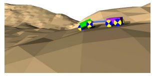


- Planning on a 3D surface



- Planning with dynamics

(friction forces)



Courtesy of V. Vonásek and P. Vaněk

Car-Like Robot

- Configuration

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

position and orientation

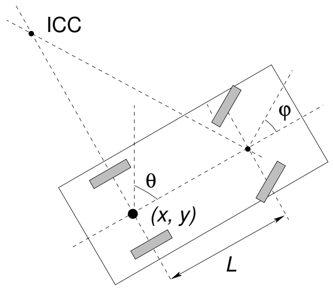
- Controls

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

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 $\dim(\vec{u}) < \dim(\vec{x})$

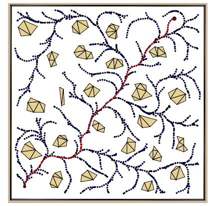
Differential constraints on possible $\dot{q}$*:*

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

Control-Based Sampling

- Select a configuration q from the tree T of the current configurations
- Pick a control input $\vec{u} = (v, \varphi)$ 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* $k\delta t = dt$.

Summary of the Lecture

Topics Discussed

- Randomized Sampling-based Methods
- Probabilistic Road Map (PRM)
- Characteristics of path planning problems
- Random sampling
- Rapidly Exploring Random Tree (RRT)

- Next: Improved randomized sampling-based methods