Randomized Sampling-based Motion Planning Methods

Jan Faigl

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

Lecture 06

B4M36UIR – Artificial Intelligence in Robotics

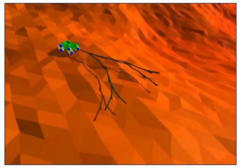
Overview of the Lecture

Part I

Part 1 – Sampling-based Motion Planning

(Randomized) Sampling-based Motion Planning

- It uses an 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
  - In 2D or 3D shape of the robot and environment can be represented as sets of triangles, i.e., tesselated models
  - Collision test – an intersection of triangles



E.g., using RAPID library http://gamma.cs.unc.edu/OBB/

- 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 provide **probabilistic completeness** or resolution completeness

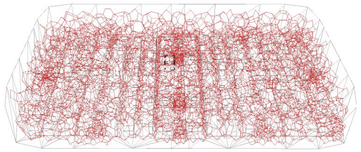
Probabilistic complete algorithms: with increasing number of samples an admissible solution would be found (if exists)

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

Incremental Sampling and Searching

- Single query sampling-based algorithms incrementally create a search graph (roadmap)
  1. **Initialization** – $G(V, E)$ an undirected search graph, V may contain q_{start}, q_{goal} and/or other points in $\mathcal{C}_{free}$
  2. **Vertex selection method** – choose a vertex $q_{cur} \in V$ for expansion
  3. **Local planning method** – for some $q_{new} \in \mathcal{C}_{free}$, attempt to construct a path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ such that $\tau(0) = q_{cur}$ and $\tau(1) = q_{new}$, τ must be checked to ensure it is collision free
     - If τ is not a collision-free, go to Step 2
  4. **Insert an edge in the graph** – Insert τ into E as an edge from q_{cur} to q_{new} and insert q_{new} to V if $q_{new} \notin V$
  5. **Check for a solution** – Determine if G encodes a solution, e.g., single search tree or graph search
  6. **Repeat to Step 2** – iterate unless a solution has been found or a termination condition is satisfied

LaValle, S. M.: Planning Algorithms (2006), Chapter 5.4

Probabilistic Roadmap Strategies

Multi-Query – roadmap based

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

Kavraki, L., Svestka, P., Latombe, J.-C., Overmars, M. H.B (1996): Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces. T-RO.

Single-Query – incremental

- 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

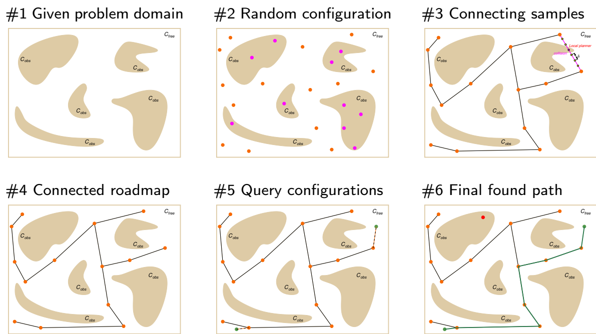
Build a roadmap (graph) representing the environment

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.

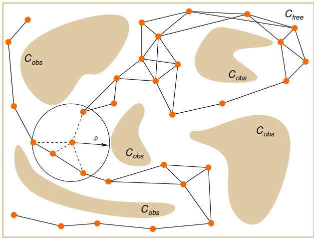
First planner that demonstrates ability to solve general planning problems in more than 4-5 dimensions.

PRM Construction



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 formalisms.

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)
  - $\mathcal{Q}_{goal}$ 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 $\pi(1) \in \mathrm{cl}(\mathcal{Q}_{goal})$.
- 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

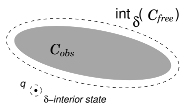
- **Feasible path planning**:
  For a path planning problem $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$
  - Find a feasible path $\pi : [0,1] \to \mathcal{C}_{free}$ such that $\pi(0) = q_{init}$ and $\pi(1) \in \mathrm{cl}(\mathcal{Q}_{goal})$, if such path exists
  - Report failure if no such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



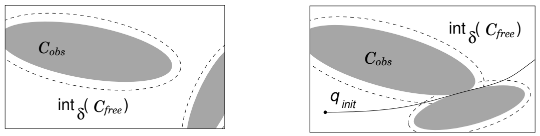
- *δ-interior* of $\mathcal{C}_{free}$ is $\mathrm{int}_\delta(\mathcal{C}_{free}) = \{q \in \mathcal{C}_{free} | \mathcal{B}_{/,\delta} \subseteq \mathcal{C}_{free}\}$
  A collection of all δ-interior states
- A collision free path π has **strong δ-clearance**, if π lies entirely inside $\mathrm{int}_\delta(\mathcal{C}_{free})$
- $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ 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

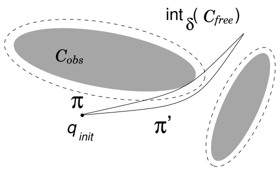
- Function $\psi : [0,1] \to \mathcal{C}_{free}$ is called **homotopy**, if $\psi(0) = \pi_1$ and $\psi(1) = \pi_2$ and $\psi(\tau)$ is collision-free path for all $\tau \in [0,1]$
- A collision-free path π_1 is **homotopic** to π_2 if there exists homotopy function ψ

  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 $\mathrm{int}_\delta(\mathcal{C}_{free})$ 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)

Algorithm 1: PRM

Vstup: q_{init}, number of samples n, radius ρ
Výstup: PRM – $G = (V, E)$
$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
for $i = 0, \dots, n$ **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 $G = (V, E)$;

Algorithm 2: sPRM

Vstup: q_{init}, number of samples n, radius ρ
Výstup: PRM – $G = (V, E)$
$V \leftarrow \{q_{init}\} \cup \{\text{SampleFree}_i\}_{i=1,\dots,n-1}$; $E \leftarrow \emptyset$;
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

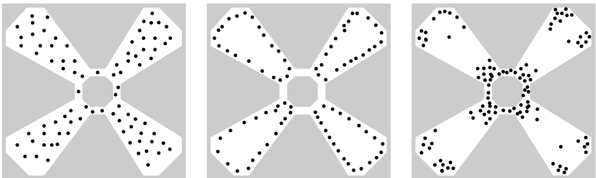
- sPRM (simplified PRM)
  - **Probabilistically complete and asymptotically optimal**
  - Processing complexity $O(n^2)$
  - Query complexity $O(n^2)$
  - Space complexity $O(n^2)$
- Heuristics practically used are usually not probabilistic complete
  - k-nearest sPRM is not probabilistically complete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

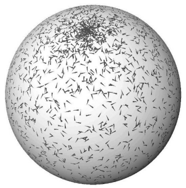
Comments about Random Sampling 2/2

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

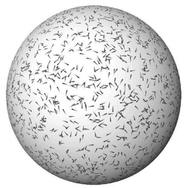
Do you know the Oraculum? (from Alice in Wonderland)

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

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



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.

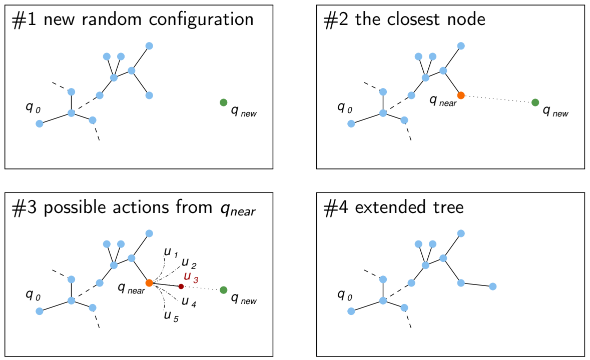
1. Start with the initial configuration q_0, which is a root of the constructed graph (tree)
2. Generate a new random configuration q_{new} in $\mathcal{C}_{free}$
3. Find the closest node q_{near} to q_{new} in the tree

   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

Algorithm 3: Rapidly Exploring Random Tree (RRT)

Vstup: q_{init}, number of samples n
Výstup: Roadmap $G = (V, E)$
$V \leftarrow \{q_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1, \dots, n$ **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

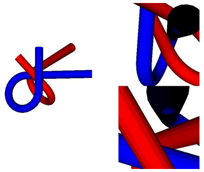
- Rapidly explores the space

  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"

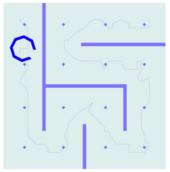
  E.g., RAPID, Bullet libraries
- Similarly to PRM, RRT algorithms have poor performance in narrow passage problems
- RRT algorithms provides feasible paths

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

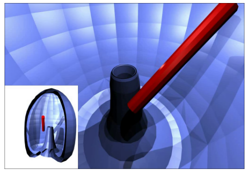
RRT – Examples 1/2



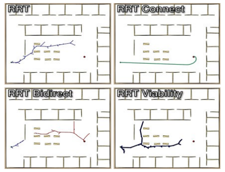
Alpha puzzle benchmark



Apply rotations to reach the goal



Bugtrap benchmark

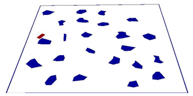


Variants of RRT algorithms

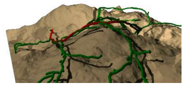
Courtesy of V. Vonásek and P. Vaněk

RRT – Examples 2/2

- Planning for a car-like robot

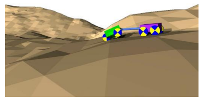


- Planning on a 3D surface



- Planning with dynamics

(friction forces)



Courtesy of V. Vonásek and P. Vaněk

Car-Like Robot

- Configuration

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

position and orientation

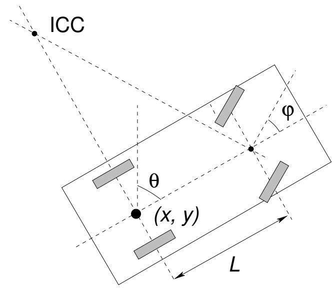
- Controls

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

forward velocity, steering angle

- System equation

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



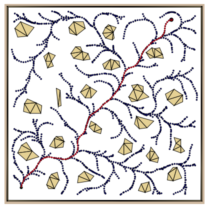
Kinematic constraints $\dim(\overrightarrow{u}) < \dim(\overrightarrow{x})$

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* $k\delta t = dt$

Part II

Part 2 – Optimal Sampling-based Motion Planning Methods

Sampling-Based Motion Planning

- PRM and RRT are theoretically probabilistic complete
- They provide a feasible solution without quality guarantee

  Despite that, they are successfully used in many practical applications
- In 2011, a systematical study of the asymptotic behavior 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
- Based on the study, new algorithms have been proposed: RRG and optimal RRT (RRT*)

  Karaman, S., Frazzoli, E. (2011):Sampling-based algorithms for optimal motion planning. IJRR.

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 satisfy 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)

Algorithm 4: Rapidly-exploring Random Graph (RRG)

Vstup: q_{init}, number of samples n

Výstup: $G = (V, E)$

```
V ← ∅; E ← ∅;
for i = 0, ..., n do
    q_rand ← SampleFree;
    q_nearest ← Nearest(G = (V, E), q_rand);
    q_new ← Steer(q_nearest, q_rand);
    if CollisionFree(q_nearest, q_new) then
        Q_near ← Near(G =
        (V, E), q_new, min{γ_RRG(log(card(V))/ card(V))^{1/d}, η});
        V ← V ∪ {q_new};
        E ← E ∪ {(q_nearest, q_new), (q_new, q_nearest)};
        foreach q_near ∈ Q_near do
            if CollisionFree(q_near, q_new) then
                E ← E ∪ {(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 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

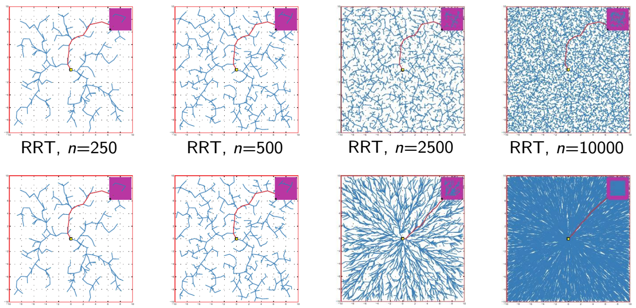
- PRM* – it follows standard PRM algorithm where connections are attempted between roadmap vertices that are within connection radius r as a function of n

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

- 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 1/3

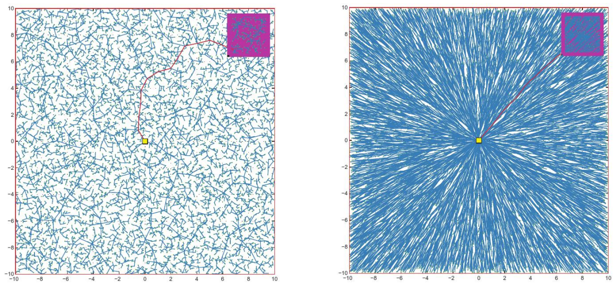


RRT, n=250 | RRT, n=500 | RRT, n=2500 | RRT, n=10000

RRT*, n=250 | RRT*, n=500 | RRT*, n=2500 | RRT*, n=10000

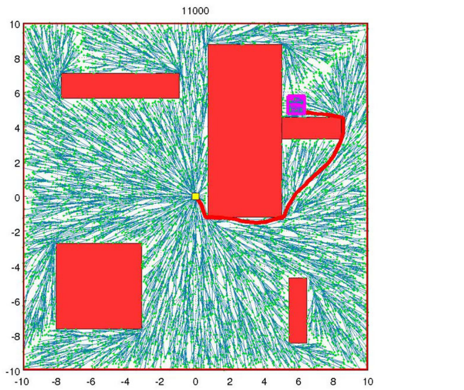
Karaman & Frazzoli, 2011

Example of Solution 2/3



RRT, n=20000 | RRT*, n=20000

Example of Solution 3/3



https://www.youtube.com/watch?v=YKiQTJpPFkA

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 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)
- Optimal sampling-based motion planning
- Rapidly-exploring Random Graph (RRG)

- Next: Multi-Goal Motion Planning and Multi-Goal Path Planning