Sampling-based Motion Planning (Pre Lecture)

Dr. Neil T. Dantam

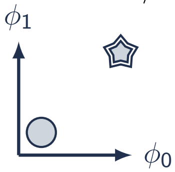
CSCI-534, Colorado School of Mines

Spring 2020

Overview

Overview

1. Start from initial configuration
2. Sample new configurations
3. **Construct** and **Add** valid states to tree/graph
4. Terminate when:
   - We find a path to the goal
   - We exhaust a timeout / max samples



Outcomes

- Know key abstractions in sampling-based motion planning
  - Robot Configuration Space
  - Metric Spaces
- Apply / Implement **Rapidly-Exploring Random Trees** (RRT)
- Apply / Implement **Probabilistic Roadmaps** (PRM)

Outline

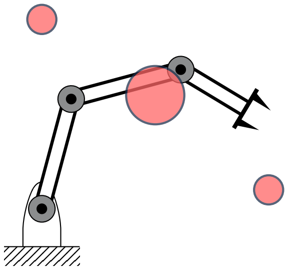
Sampling-Based Motion Planning

Metric Spaces

Rapidly-Exploring Random Trees (RRT)

Probabilistic Roadmaps (PRM)

Configuration Space



- Configuration Space $\mathcal{Q}$
- Obstacle Region: $\mathcal{Q}_{\text{obs}} = \{q \in \mathcal{Q} \mid \neg\texttt{is-valid}(q)\}$
- Free Space: $\mathcal{Q}_{\text{free}} = \{q \in \mathcal{Q} \mid \texttt{is-valid}(q)\}$
- Total Space:
  - $\mathcal{Q}_{\text{obs}} \cap \mathcal{Q}_{\text{free}} = \emptyset$
  - $\mathcal{Q}_{\text{obs}} \cup \mathcal{Q}_{\text{free}} = \mathcal{Q}$

Generally: no explicit representation of $\mathcal{Q}_{\text{obs}}$, $\mathcal{Q}_{\text{free}}$

Piano Mover's Problem

Definition

Piano Mover's Problem

- **Given:** Environment, Robot, Start and Goal configurations
  - **World** $\mathcal{W}$ in $\mathbb{R}^2$ or $\mathbb{R}^3$
  - **Robot** in $\mathcal{W}$: either a single or collection of rigid bodies
  - **Configuration space** $\mathcal{Q}$ for the robot
  - **Initial configuration** $q_0 \in \mathcal{Q}_{\mathrm{free}}$
  - **Goal configuration** $q_G \in \mathcal{Q}_{\mathrm{free}}$
- **Find:** Valid path from q_0 to q_G.

What data structure for "world" and "robot?"

We can define flexible abstractions.

Fundamental Abstractions

Configuration Space: Set of possible configurations

- $\mathcal{Q}$

Configuration Sampler: Generates candidate configurations

- $() \mapsto \mathcal{Q}$

Validity Checker: Is some configuration in $\mathcal{Q}_{\text{free}}$?

- $\underbrace{\mathcal{Q}}_{\text{sample}} \mapsto \underbrace{\{0,1\}}_{\text{valid?}}$

Nearest Neighbors: Closest point in tree/graph to new sample:

- $\underbrace{\mathcal{P}(\mathcal{Q})}_{\text{tree}} \times \underbrace{\mathcal{Q}}_{\text{new sample}} \mapsto \underbrace{\mathcal{Q}}_{\text{nearest neighbor}}$
- **Distance metric**: $\mathcal{Q} \times \mathcal{Q} \mapsto \mathbb{R}$

Sampling-based motion planning generalizes beyond robotics.

Motion Planning Problem, Redux

Definition

Piano Mover's Problem

- **Given:** Environment, Robot, Start and Goal configurations
  - **World** $\mathcal{W}$ in $\mathbb{R}^2$ or $\mathbb{R}^3$
  - **Robot** in $\mathcal{W}$: either a single or collection of rigid bodies
  - **Configuration space** $\mathcal{Q}$ for the robot
  - **Initial configuration** $q_0 \in \mathcal{Q}_{\mathrm{free}}$
  - **Goal configuration** $q_G \in \mathcal{Q}_{\mathrm{free}}$
- **Find:** Valid path from q_0 to q_G.

Definition

Sampling-based Motion Planning

- **Given:** Metric state space, validity checker, start, goal
  - $\mathcal{Q}$, state space
    - Metric function: $\mathcal{Q} \times \mathcal{Q} \mapsto \mathbb{R}$
    - (Uniform) Sampler: $q \sim \mathcal{Q}$
  - $v : \mathcal{Q} \mapsto \{0, 1\}$, validity checker
  - $q_0 \in \mathcal{Q}$, initial state
  - $q_g \sim G$, goal sampler
- **Find:** Valid path from q_0 to a q_g.

Outline

Definition: Metric Space

Definition: Metric Space

A **metric space** is a space $\mathcal{Q}$ equipped with a function $\rho : \mathcal{Q} \times \mathcal{Q} \mapsto \mathbb{R}$ that has the following properties for any $a, b, c \in \mathcal{Q}$

Nonnegative: $\rho(a, b) \geq 0$
(Distances are always greater than zero)

Reflexive: $(\rho(a, b) = 0) \iff (a = b)$
(Distance is zero only for identical elements)

Symmetric: $\rho(a, b) = \rho(b, a)$
(Distance from a to b is the same as from b to a)

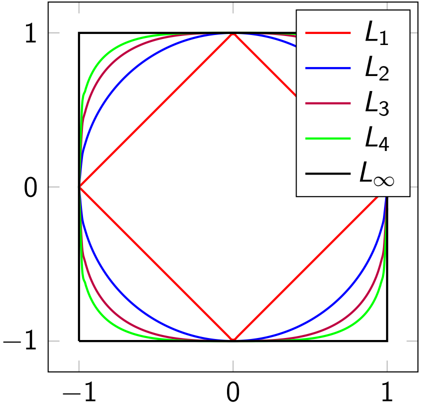
Triangle: $\rho(a, b) + \rho(b, c) \geq \rho(a, c)$
(a to b to c cannot be shorter than directly from a to c)

Example: Robot Joint Space and L_p metrics

$$\rho(x, x') = \left(\sum_{i=0}^{n-1} \left| x_i - x'_i \right|^p \right)^{\frac{1}{p}}$$

- **Space:**
  $\mathcal{Q} \subseteq \mathbb{R}^n$, the set of robot joint positions
- **Euclidean Distance:**
  $L_2(x, x') = \sqrt{\sum_{i=0}^{n-1} \left(x_i - x'_i\right)^2}$
- **Manhattan Distance:**
  $L_1(x, x') = \sum_{i=0}^{n-1} \left| x_i - x'_i \right|$
- **L_∞:**
  $L_\infty(x, x') = \max_{0 \le i \le n-1} \left(\left| x_i - x'_i \right| \right)$

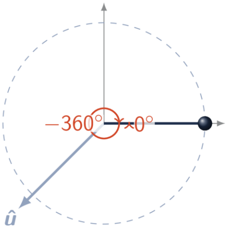
Unit "Circle"



Example: Rotation Metric

Quaternion Double-Cover

Axis-Angle



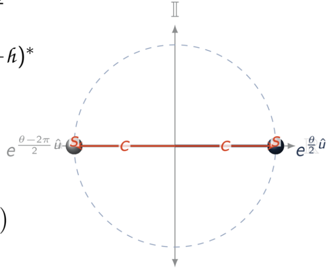
$$\exp\left(\tfrac{\theta}{2}\hat{u}\right) = \sin\tfrac{\theta}{2}\hat{u} + \cos\tfrac{\theta}{2}$$

$$h \otimes \vec{p} \otimes h^* = (-h) \otimes \vec{p} \otimes (-h)^*$$

$$\sin\tfrac{\theta}{2} = -\sin\tfrac{\theta - 2\pi}{2}$$
$$\cos\tfrac{\theta}{2} = -\cos\tfrac{\theta - 2\pi}{2}$$

$$\exp\left(\tfrac{\theta}{2}\hat{u}\right) = -\exp\left(\tfrac{\theta-2\pi}{2}\hat{u}\right)$$
$$\exp\left(\tfrac{\theta}{2}\hat{u}\right) \sim \exp\left(\tfrac{\theta-2\pi}{2}\hat{u}\right)$$

Quaternion



$$\begin{aligned} \theta &= 0.000 \\ \theta - 2\pi &= -6.280 \end{aligned}$$

$$\begin{array}{rrcrr} & 0.000 & \hat{\boldsymbol{u}} & + & 1.000 \\ \sim & 0.000 & \hat{\boldsymbol{u}} & + & -1.000 \end{array}$$

Example: Rotation Metric

Distance Functions

$$\exp\left(\frac{\theta}{2}\hat{u}\right) = \sin\frac{\theta}{2}\hat{u} + \cos\frac{\theta}{2} \qquad \text{and} \qquad \exp\left(\frac{\theta}{2}\hat{u}\right) \sim \exp\left(\frac{\theta - 2\pi}{2}\hat{u}\right)$$

Quaternion Space

- L_p Norm:
  - $\min\left\{\|p-q\|, \|p+q\|\right\}$
  - Length of 4D line segment
- 4D angle:
  - $\min\left\{\cos^{-1}\left(q \cdot p\right), \cos^{-1}\left(-q \cdot p\right)\right\}$
  - Angle between 4D quaternions

Log Space

$$\min\left\{\left\|\ln\left(q \otimes p^*\right)\right\|, \left\|\ln\left(-q \otimes p^*\right)\right\|\right\}$$

- $p \otimes h_r = q \quad \rightsquigarrow \quad h_r = q \otimes p^*$
- Matrix: $\left\|\ln\left(\mathbf{PQ}^{-1}\right)\right\|$

Exercise: Transformation Metric

Dual Quaternion: $\rho\left(h_1 + d_1\varepsilon, h_2 + d_2\varepsilon\right)$

Transformation Matrix: $\rho\left(\begin{bmatrix}\mathbf{R}_1 & \mathbf{v}_1 \\ \mathbf{0} & \mathbf{1}\end{bmatrix}, \begin{bmatrix}\mathbf{R}_2 & \mathbf{v}_2 \\ \mathbf{0} & \mathbf{1}\end{bmatrix}\right)$

Outline

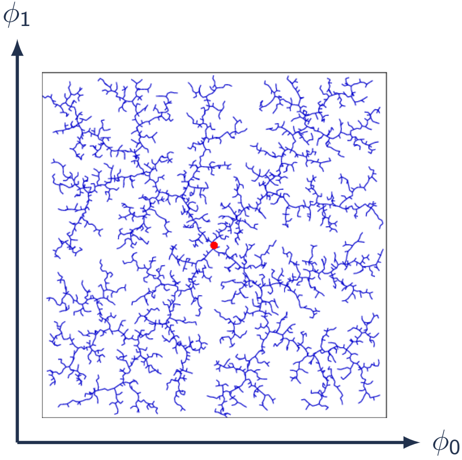
Sampling-Based Motion Planning

Metric Spaces

Rapidly-Exploring Random Trees (RRT)

Probabilistic Roadmaps (PRM)

RRT Illustration



`http://msl.cs.uiuc.edu/rrt/`

RRT Algorithm

Algorithm 1: Rapidly-Exploring Random Tree

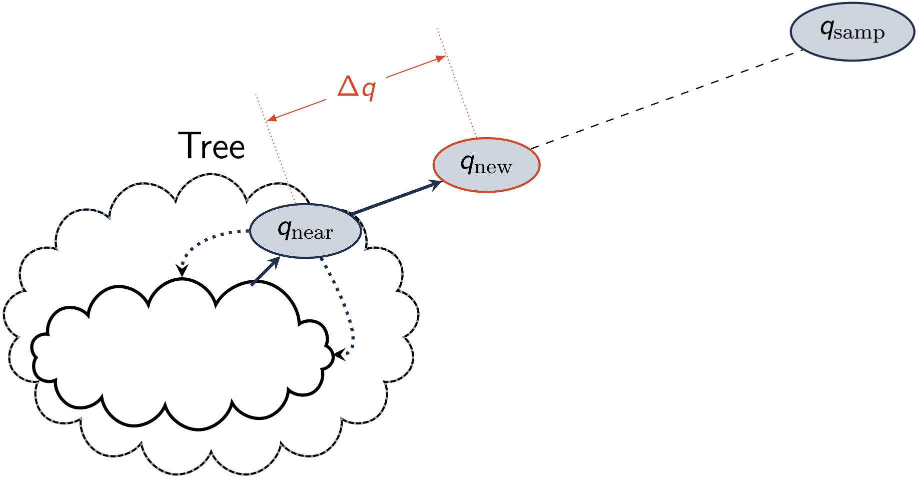
Input: $\phi_0 \in \mathcal{Q}$: start
Input: $\Phi_g \subseteq \mathcal{Q}$: goal set

$V \leftarrow \{\phi_0\}$; `// Tree nodes`
$E \leftarrow \emptyset$; `// Tree edges`
for $k \leftarrow 0$ **to** `LIMIT` **do**
    $q_{\mathrm{samp}} \leftarrow \texttt{sample}()$;
    $q_{\mathrm{near}} \leftarrow \texttt{nearest-neighbor}(V, q_{\mathrm{samp}})$;
    $q_{\mathrm{new}} \leftarrow \texttt{new-conf}(q_{\mathrm{near}}, q_{\mathrm{samp}})$;
    if $\texttt{valid}(q_{\mathrm{new}})$ **then**
        if $\exists q \in \Phi_g,\ \texttt{dist}(q, q_{\mathrm{new}}) < \epsilon$ **then**
            return path from q_0 to q_{new};
        $V \leftarrow V \cup \{q_{\mathrm{new}}\}$;
        $E \leftarrow E \cup \{q_{\mathrm{near}} \rightarrow q_{\mathrm{new}}\}$;
return `TIMEOUT`;

Subroutines

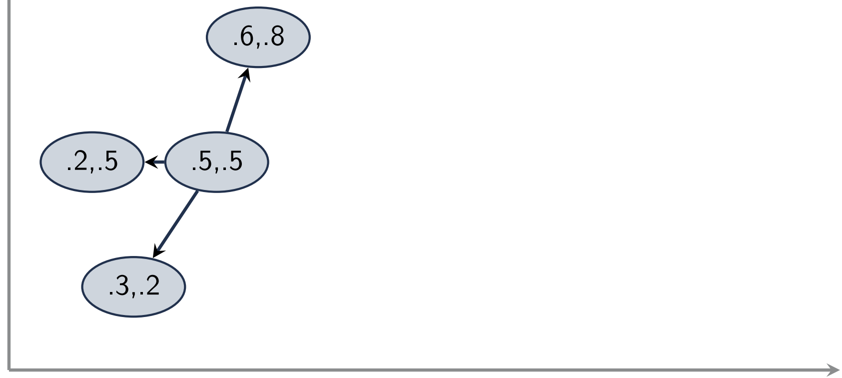
- `sample`
- `nearest-neighbor`
- `new-conf`
- `valid`
- `dist`

RRT Step Illustration



Exercise: RRT-Step

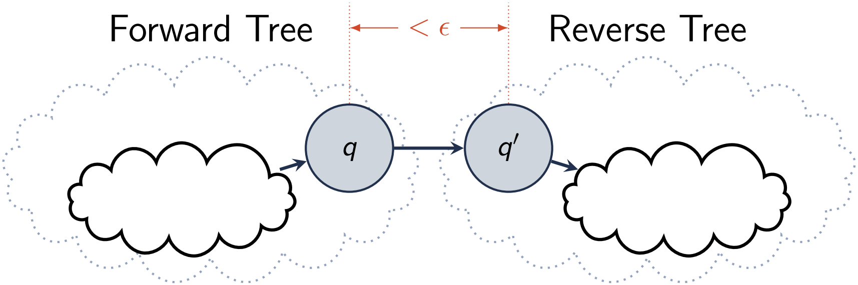
Next rands:
(1.2,1.2)
(1.5,0.3)



RRT-Connect

Bi-directional RRT

1. Construct one RRT, rooted at the start
2. Construct second RRT, rooted at the goal
3. Terminate when the two trees connect



Outline

Sampling-Based Motion Planning

Metric Spaces

Rapidly-Exploring Random Trees (RRT)

Probabilistic Roadmaps (PRM)

PRM Overview

Preprocessing: Construct the Roadmap

1. Sample new configurations
2. Connect to "neighboring" configurations in roadmap

Query: Search the Roadmap

1. Find path in roadmap via discrete search (e.g., A*)
2. Connect subsequent configurations with "local planner" (interpolate)

PRM Construction

Algorithm

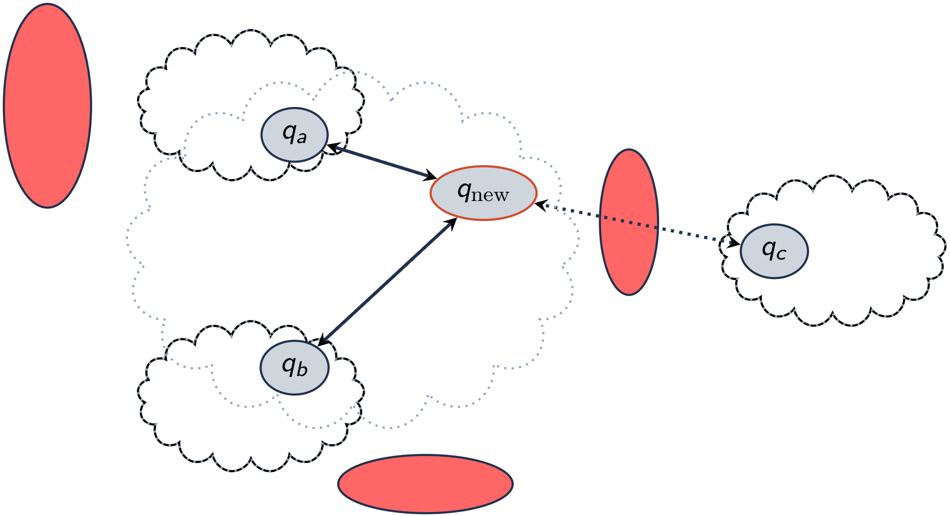
Algorithm 2: Construct Probabilistic Roadmap

```
V ← ∅; // Roadmap nodes
E ← ∅; // Roadmap Edges
k ← 0; // Iteration count
while k < LIMIT do
    q_new ← sample();
    if valid(q_new) then
        V ← V ∪ {q_new};
        k ← k + 1;
        foreach q ∈ neighborhood(q_new) do
            if ¬connected(q, q_new) ∧ connect(q, q_new) then
                E ← E ∪ {q → q_new};
return (V, E);
```

- Neighbors:
  - K-nearest neighbors (KNN)
  - KNN from connected-components of V
  - Radius (within ϵ distance)
- Connected Components:
  - Optional: do we want different paths?
  - Efficiency: Track as edges are added
- Connect: valid path exists
  - E.g., linearly interpolate and check validity

PRM Construction

Illustration



Practical Considerations

- Nearest Neighbors: Kd-tree (partitioning) or brute force (for high-dof)
- Collision Checking: Bounding-boxes then decompose into triangles
- Paths: Smooth and Shortcut
- Hundreds of variations on RRT, PRM, and similar methods
- RRT-Connect usually works pretty well