Sampling-based Planning 02

Jane Li

Assistant Professor
Mechanical Engineering Department, Robotic Engineering Program
Worcester Polytechnic Institute
Quiz (10 pts)

• (3 pts) Explain at high-level how to plan a path using PRM?

• (3 pts) What are the two popular ways to find nearest neighbor in PRM? And how to speed up your search?

• (4 pts) What heuristics can be used to guide expansion? List two, and explain why
Two-phase solution

- Environment remains unchanged
- Reuse roadmap for multi-query
Finding Nearest Neighbors (NN)

- Two popular ways to do NN in PRM
  - Find k nearest neighbors (even if they are distant)
  - Find all nearest neighbors within a certain distance

- Naive NN computation can be slow with thousands of nodes
  - use *kd-tree* to store nodes and do NN queries
Possible Heuristics

• # of Nodes nearby
  • For a node $c$, count the # of nodes $N$ within a predefined distance
  • $N$ is small $\rightarrow$ obstacle region may occupy large portion of $c$’s neighborhood

• Use Heuristics $= 1/N$ to guide random sampling
**Possible Heuristics**

• Distance to nearest reachable neighbor
  
  • For a node $c$, find the distance $d$ to the nearest connected component that doesn’t contain this node
  
  • $d$ is small $\rightarrow c$ lies in the region where two components fail to connect
  
• Heuristics $= 1/d$
Possible Heuristics

• Others?

• Behavior of local planner?
  • Always fail to connect $\rightarrow$ difficult region
Advanced Roadmaps
Overview

- Sampling strategies
- Hierarchical roadmap
Motivating problem

• A mobile robot navigates in an unknown or partially known home environment
  • 2D navigation
  • Focus on efficient expansion and reconstruction of roadmap
How to represent the traversable areas?

- Cell decomposition (e.g., grid) – commonly used

- Pros
  - Complete coverage
  - Regular space division
  - Can search for a path as new traversable area is explored

- Cons
  - Sensitive to dimensionality
How to represent the traversable areas?

- **Roadmap**
  - Construct map based on samples in traversable areas

- **Pros**
  - Reduced computational complexity

- **Cons**
  - Unknown or partially known environment → Need to expand or reconstruct map
  - Random sampling → fail to cover narrow passages.
How to fix the problem?

- Random sampling fails to cover the whole traversable areas?
  - Open area
  - Narrow passages

Other sampling strategies?
Obstacle-based PRM

- To navigate a narrow passage
  - Need more sample points for where planning is hard
  - Sample near C-obstacles?

Can you explicitly construct C-obstacles?
• How to find points on C-obstacles?
  • Find a point in the C-obstacles – a collision configuration
  • Select a random direction in C-space
  • Find a free point in that direction
  • Find the boundary point between then using binary search
PRM VS OBPRM

**PRM**
- 328 nodes
- 4 major CCs

**OBPRM**
- 161 nodes
- 2 major CCs
Gaussian Sampling [1]

- Gaussian sampler
  - Find a $q_1$
  - Pick a $q_2$ from a Gaussian distribution centered at $q_1$
  - If: both are in collision or collision-free, discard them
  - Else: keep the collision-free one

Sampling distribution for varying Gaussian width (width decreasing from left to right) [1]
Milestones (13,000) created by uniform sampling before the narrow passage was adequately sampled

Milestones (150) created by Gaussian sampling

The gain is not in sampling fewer milestones, but in connecting fewer pairs of milestones
Bridge sampling [2]

- Bridge sampler
  - Sample a $q_1$ that is in collision

  - Sample a $q_2$ in neighborhood of $q_1$ using some probability distribution (e.g. Gaussian)

  - If $q_2$ in collision, get the midpoint of $(q_1, q_2)$

  - Check if midpoint is in collision, if not, add it as a node
Bridge sampling
Bridge vs Gaussian

Gaussian

Bridge test
Bridge sampling

Bridge Sampling performs well in narrow passages
Deterministic Sampling

- Random sampling (biased or not) can be unpredictable and irregular
  - Each time your run your algorithm you get a different sequence of samples, so performance varies

- In the limit, space will be sampled well, but in finite time result may be irregular
What do we care about?

- Dispersion

\[ \delta(P) = \sup_{x \in X} \left\{ \min_{p \in P} \{ \rho(x, p) \} \right\}. \]

- What does it mean?
  - Intuitively, the dispersion quantifies how well a space is covered by a set of points \( S \) in terms of the largest open Euclidean ball that touches none of the points.

\( P \) is a finite set of points, \((X, \rho)\) is a metric space (\( \rho \) is a distance metric), which is the radius of the largest empty ball.
Quasi-random sampling

- Use **quasi-random** to replace random sampling
  - Deterministic sequence of **equivalent in dispersion**
  - Consistent performance

- E.g., Van der Corput sequence (for base = 10)

\[
g_b(n) = \sum_{k=0}^{L-1} d_k(n)b^{-k-1}
\]
Adaptive sampling [3]

• Region-Sensitive Adaptive Motion Planner (RESAMPL)

• Main idea
  • Classify regions based on the entropy of the samples in it
  • Uses the classification to further refine the sampling
A brief introdEntropy
Region Construction

- Region construction

(a) C-space
(b) Initial Sampling
(c) Region Construction
Region construction

Algorithm 3.1 Region Construction

Require: Model $\mathcal{M}$, initial samples $S$, and $k$.

1: while there exists an unmarked sample in $S$ do
2: Let $c$ be a randomly selected unmarked sample $\in S$.
3: Set $N = \{k$ nearest neighbors to $c\}$.
4: Set $R$ = a new region with center $c$ and neighbors $N$.
5: Add $R$ to $\mathcal{M}$.
6: Flag $c$ and $N$ as marked.
7: end while
8: return $\mathcal{M}$
Adaptive sampling

- Re-sampling based on region classification
  - Use entropy to measure the percentage of free(blocked) pointed
Region classification

(a) Region Construction

\[ H(X) = - \sum_{i=1}^{n} P(x_i) \log_b P(x_i) \]

(b) Free

(c) Surface

(d) Narrow

(e) Blocked
Region classification

- Free region
  - Percentage (or entropy) of blocked sample is low enough
Region classification

- **Blocked region**
  - Percentage of free samples in the region (or entropy) is low enough

- **Attention!**
  - Free nodes are discovered during the classification
  - Do not classify a region as blocked until several attempts have been made to classify and add additional samples
Region classification

- Surface region
  - High entropy region that can be partition to two low-entropy regions

- Process
  - Divide the region to 2 sub-region
  - Determine centroid of free_blocked space
  - If both sub-region has low entropy → surface region
Region classification

• Narrow region
  • High entropy region that cannot be partition to two low-entropy regions

• Attention
  • Difficult to classify
  • Do not attempt to classify a region as narrow until several attempts have been made to classify and add additional samples
Region classification

Algorithm 3.2 Region Classification

Require: A region $R$, threshold $e_{low}$, threshold $e_{high}$, number of attempts to classify $t$, and number of samples to add in each classification attempt $k$.

1: for $t$ attempts to classify $R$ do
2:   Let $e_R$ be the entropy of $R$ (% of blocked samples in $R$).
3:   if $e_R < e_{low}$ then
4:     return free
5:   end if
6:   Add $k$ additional samples to $R$ and recompute $e_R$.
7:   Partition $R$ into two subregions, $R_{free}$ and $R_{blocked}$.
8:   Let $e_{free}$ be the entropy of $R_{free}$ (% of blocked samples in $R_{free}$).
9:   Let $e_{blocked}$ be the entropy of $R_{blocked}$ (% of free samples in $R_{blocked}$).
10:  if $e_{free} < e_{low}$ and $e_{blocked} < e_{low}$ then
11:     return surface
12:   end if
13: end for
14: if $e_R == 1$ then
15:     return blocked
16: end if
17: if $e_R > e_{high}$ then
18:     return narrow
19: end if
20: return surface
Change the sampling strategy

- Generates additional nodes in less open area
- Improved coverage, yet the roadmap still has **irregularly** distributed nodes
- Navigates in unexplored area – **unsolved**
Hierarchical roadmap

• Main idea
  • Incrementally constructs a **hierarchical roadmap** using low cost sonar sensors

• Hierarchical roadmap
  • A multi-layered structure that abstract the traversable areas using the adequate number of nodes and edges

  • Nodes in the hierarchical roadmap are distributed regularly to cover
  • and divide the traversable areas
Hierarchical roadmap

Region roadmap

A → B

C

Region node B

Hub roadmap

Hub node

Gate node

Sub roadmap
Motion planning

• Step 1
  • Search a topological path on the region roadmap

• Step 2
  • Search metric local paths in the sub-regions
Region roadmap is constructed by dividing the entire environment into several sub-regions

- E.g., rooms in a house

- How to divide?
Region extraction

- **Step 1: Obtaining reliable region in the grid map**
  - High confidence of grid cell occupancy → practical issue of sonar

- **Step 2: Cell Decomposition for traversable area**
  - Recursively dividing area until every cell has only empty grid cells
Region extraction

- **Step 3: Normalized Graph Cut**
  - Tentatively divided into two sub-regions using normalized graph cut

- **Step 4: Extracting a New Sub-region**
  - Use a convexity criterion determine whether the reliable region can be regarded as one sub-region

\[
C_{1\text{cluster}} = \frac{\# \text{ of oce. grids } \in CH1}{\sum \text{ size of Cell}}
\]

\[
C_{2\text{clusters}} = \frac{\sum_{i=1}^{2} \# \text{ of oce. grids } \in CH2(i)}{\sum \text{ size of Cell}}
\]

\[
C_{1\text{cluster}} > c_t \quad \& \quad C_{2\text{clusters}} < 0.5 \times C_{1\text{cluster}}
\]

**New region extracted!**
Region node and gate node

• Region nodes
  • The extracted sub-regions become region nodes (RNs)
  • Generate edges based on the adjacency between two sub-regions

• Gate nodes
  • Providing paths to the neighbor RNs
  • Defined as the midpoint of the boundary between two RNs
Hierarchical roadmap in the home environment

(a) Topological path

(b) Region roadmap

(c) Region node A

(d) Region node B

(e) Goal point

(f) Local Metric path


Student talk
End