16-782
Planning & Decision-making in Robotics

Planning Representations:
Probabilistic Roadmaps for Continuous Spaces

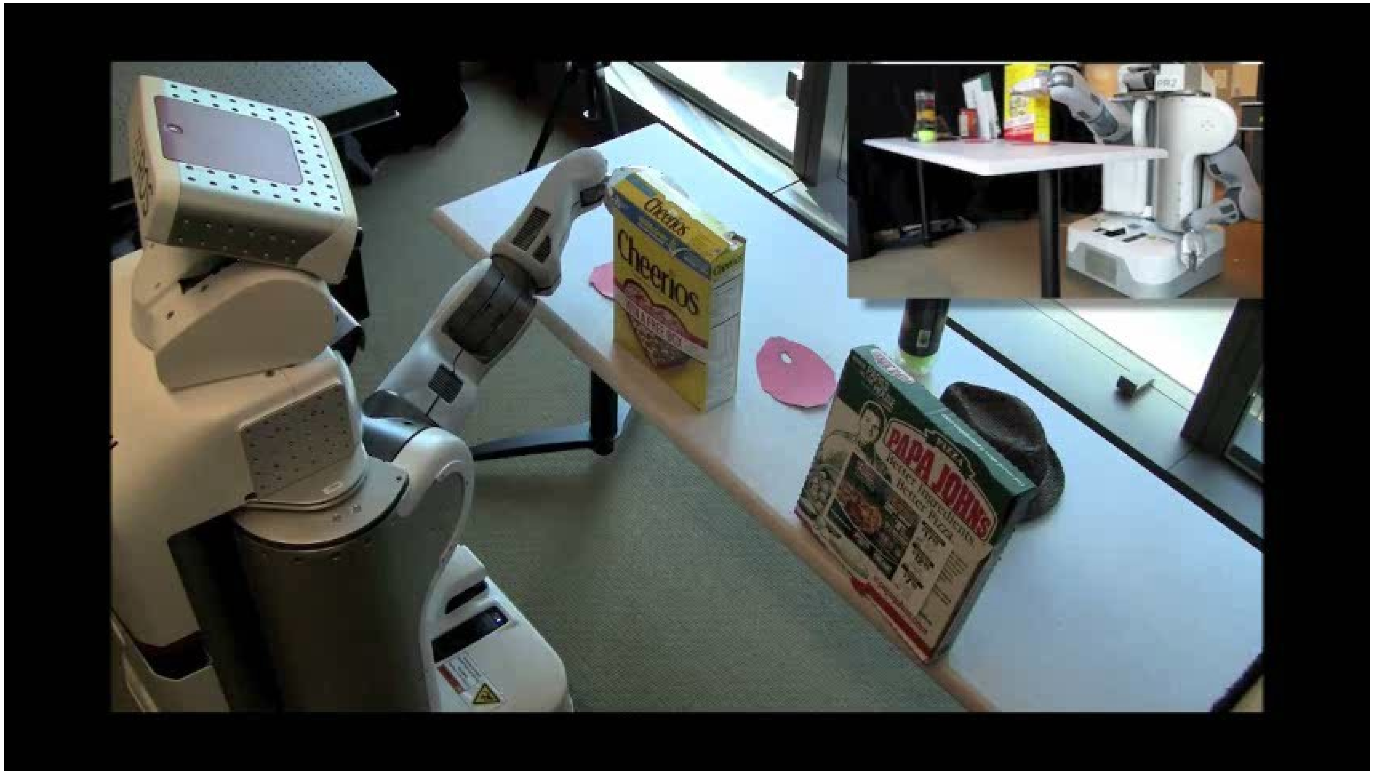
Maxim Likhachev

Robotics Institute

Carnegie Mellon University

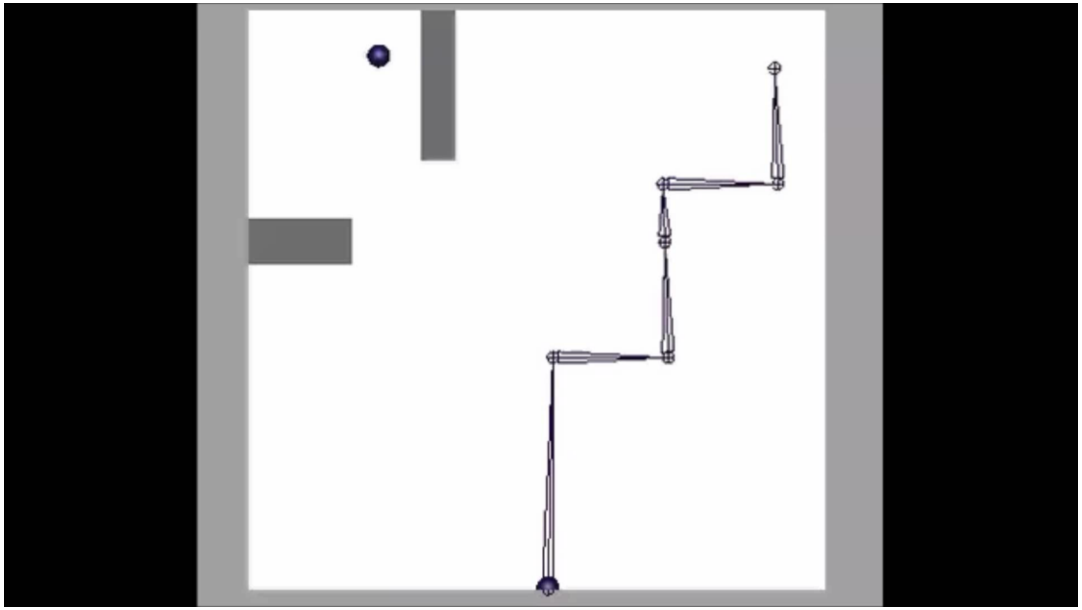
Example

- Planning for manipulation



Example

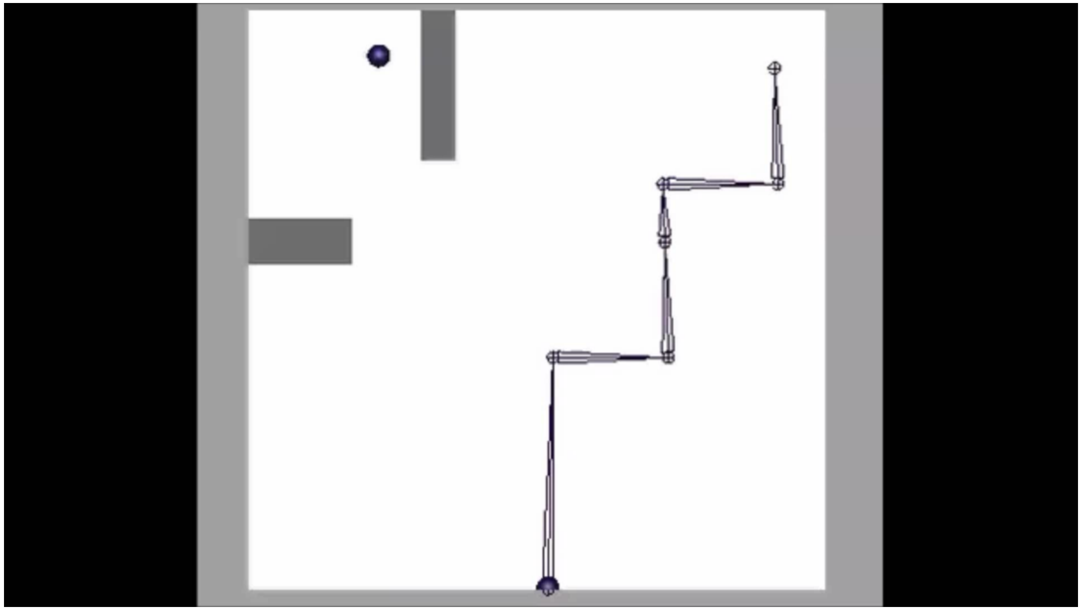
- Planning for manipulation



Example

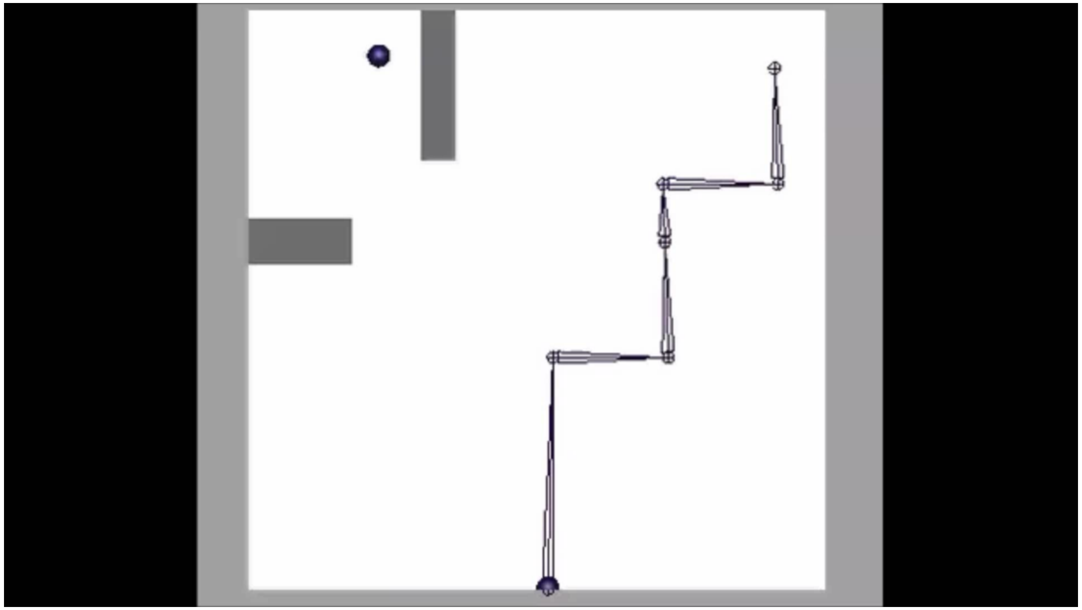
- Planning for manipulation
  - robot state is defined by joint angles $Q = \{q_1, \ldots, q_6\}$
  - need to find a (least-cost) motion that connects Q_{start} to Q_{goal}

Constraints?



Example

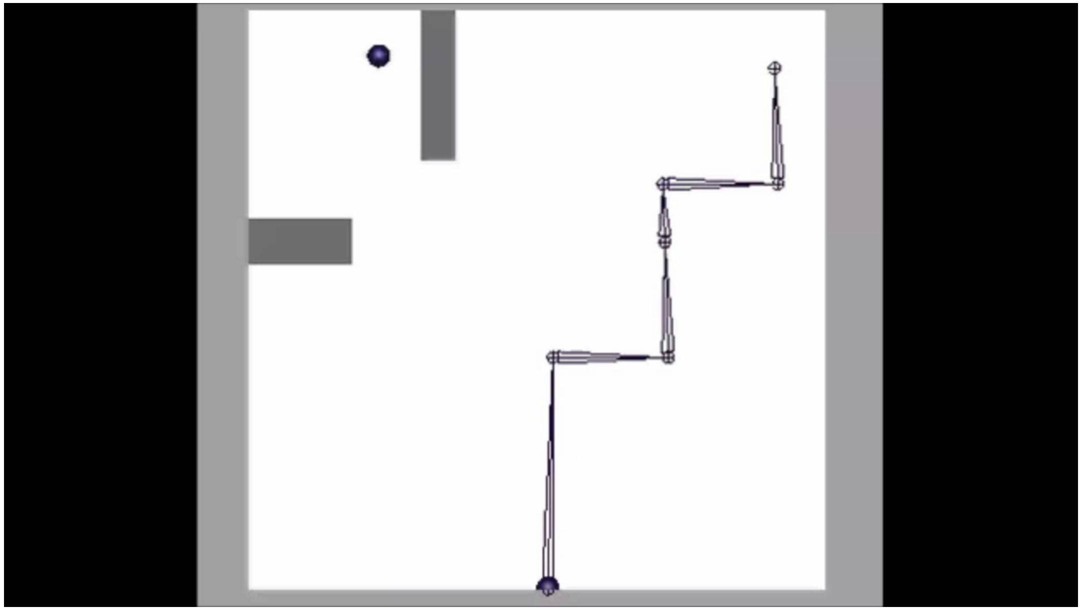
- Planning for manipulation
  - robot state is defined by joint angles $Q = \{q_1, \ldots, q_6\}$
  - need to find a (least-cost) motion that connects Q_{start} to Q_{goal}
  - Constraints:
    - All joint angles should be within corresponding joint limits
    - No collisions with obstacles and no self-collisions



Exampl

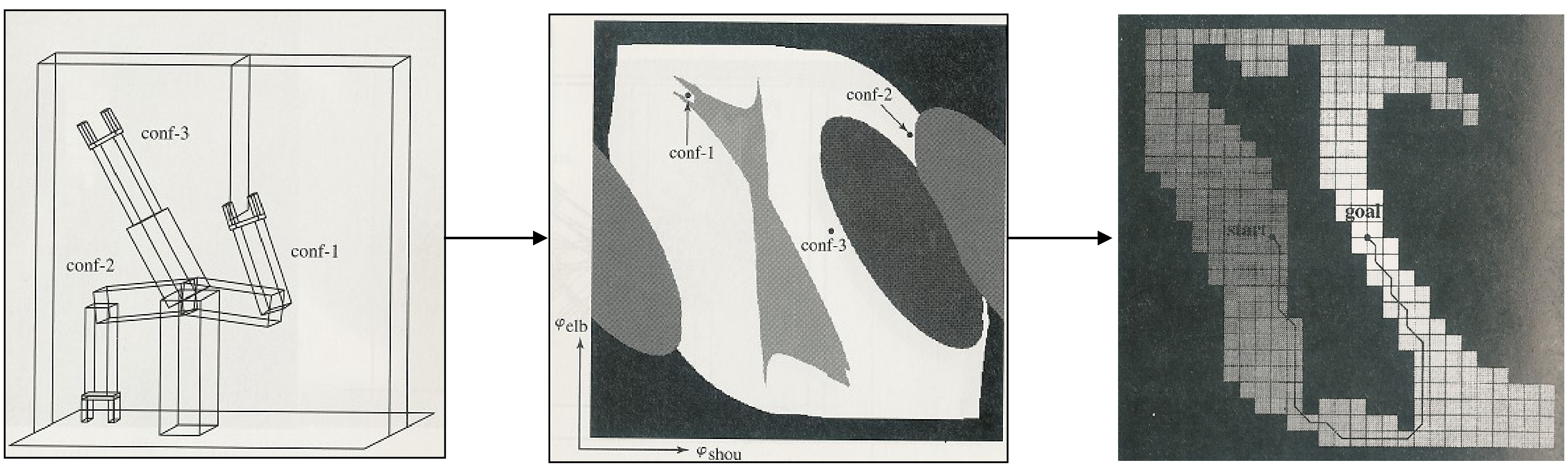
Can we use a grid-based representation for planning?

- Planning for manipulation
  - robot state is defined by joint angles $Q = \{q_1, \ldots, q_6\}$
  - need to find a (least-cost) motion that connects Q_{start} to Q_{goal}
  - Constraints:
    - All joint angles should be within corresponding joint limits
    - No collisions with obstacles and no self-collisions



Resolution Complete vs. Sampling-based Planning

- Resolution complete planning (e.g. Grid-based):
  - generate a systematic (uniform) representation (graph) of a free C-space (C_{free})
  - search the generated representation for a solution guaranteeing to find it if one exists (completeness)
  - can interleave the construction of the representation with the search (i.e., construct only what is necessary)

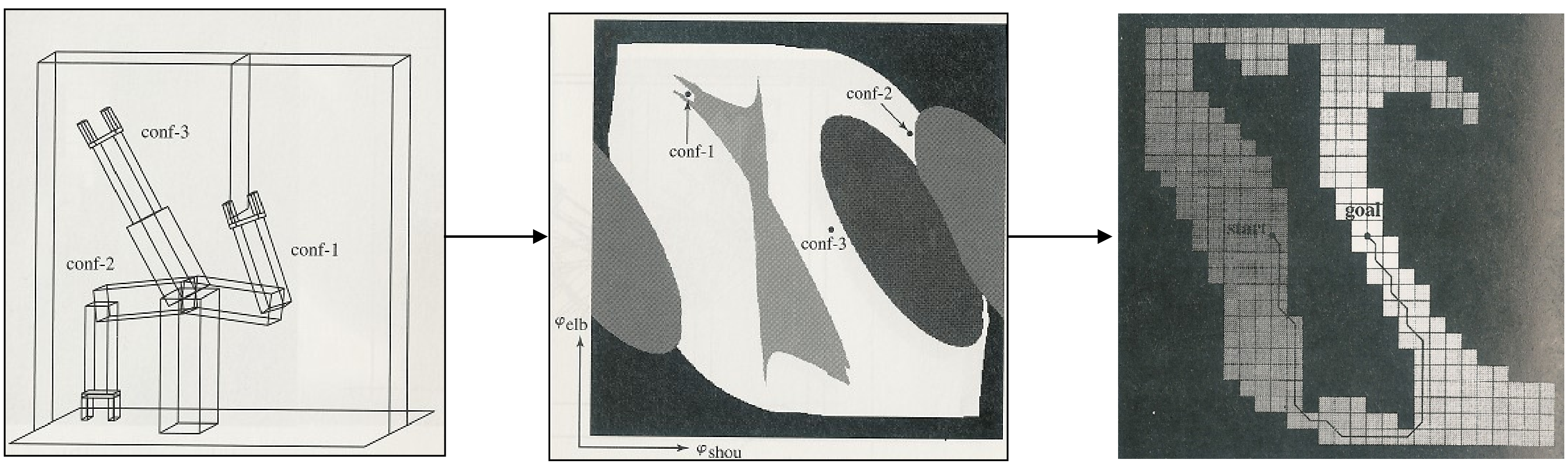


the example above is borrowed from "AI: A Modern Approach" by S. RussellL & P. Norvig

Resolution Complete vs. Sampling-based Planning

- Resolution complete planning (e.g. Grid-based):
  - complete and provide sub-optimality bounds on the solution

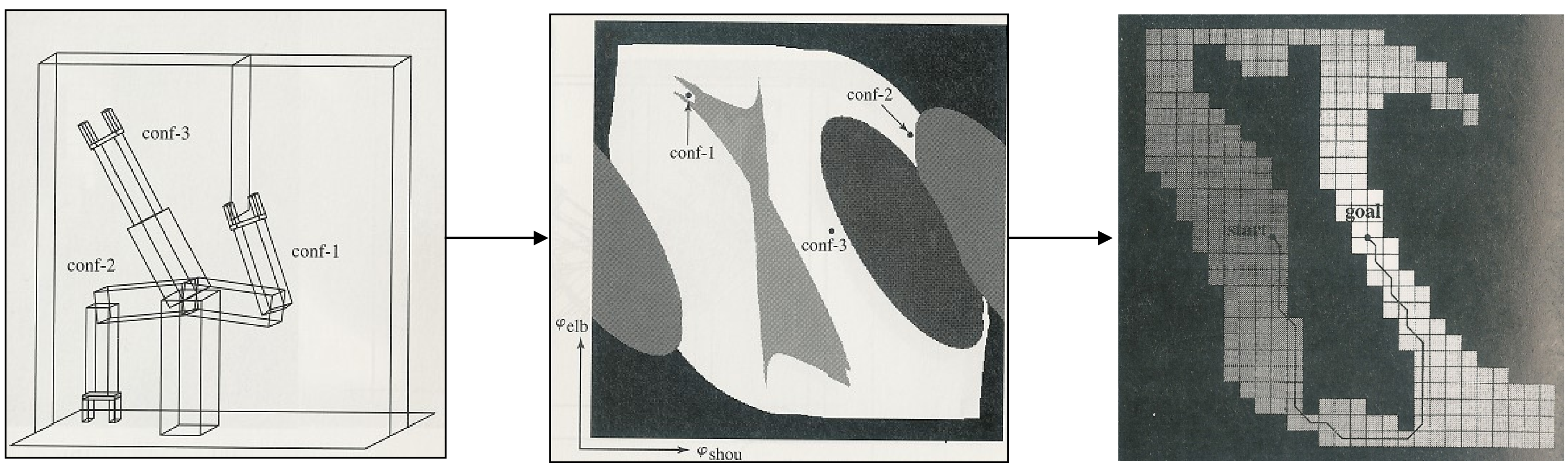
Great. Any issues?



the example above is borrowed from "AI: A Modern Approach" by S. RusselL & P. Norvig

Resolution Complete vs. Sampling-based Planning

- Resolution complete planning (e.g. Grid-based):
  - complete and provide sub-optimality bounds on the solution
  - can get computationally very expensive, especially in high-D

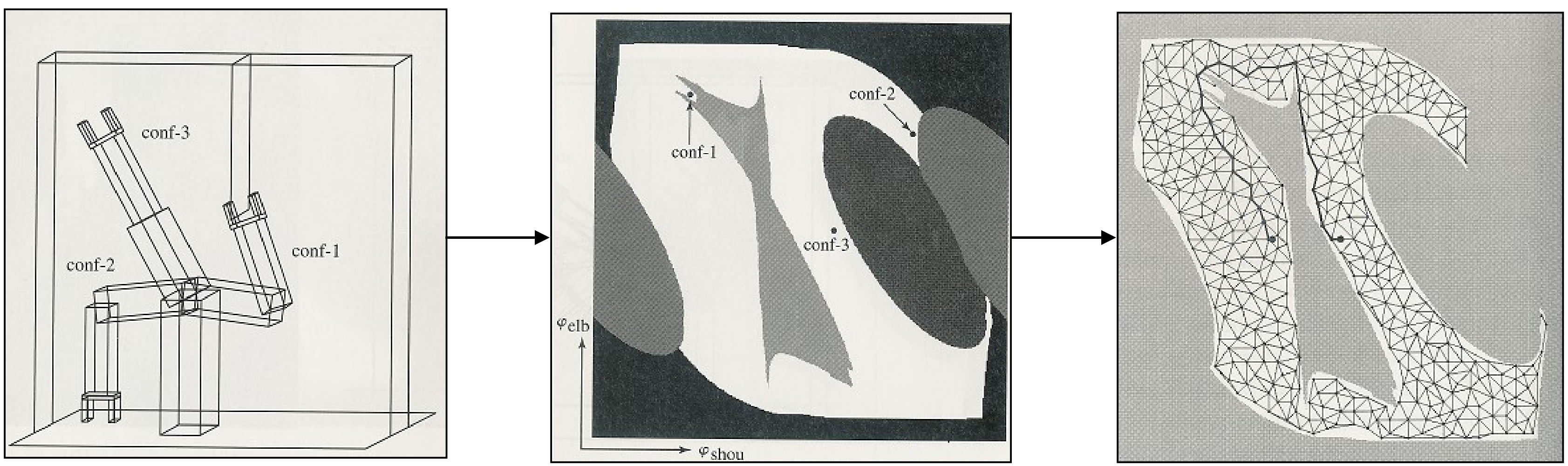


the example above is borrowed from "AI: A Modern Approach" by S. RussellL & P. Norvig

Resolution Complete vs. Sampling-based Planning

- Sampling-based planning:

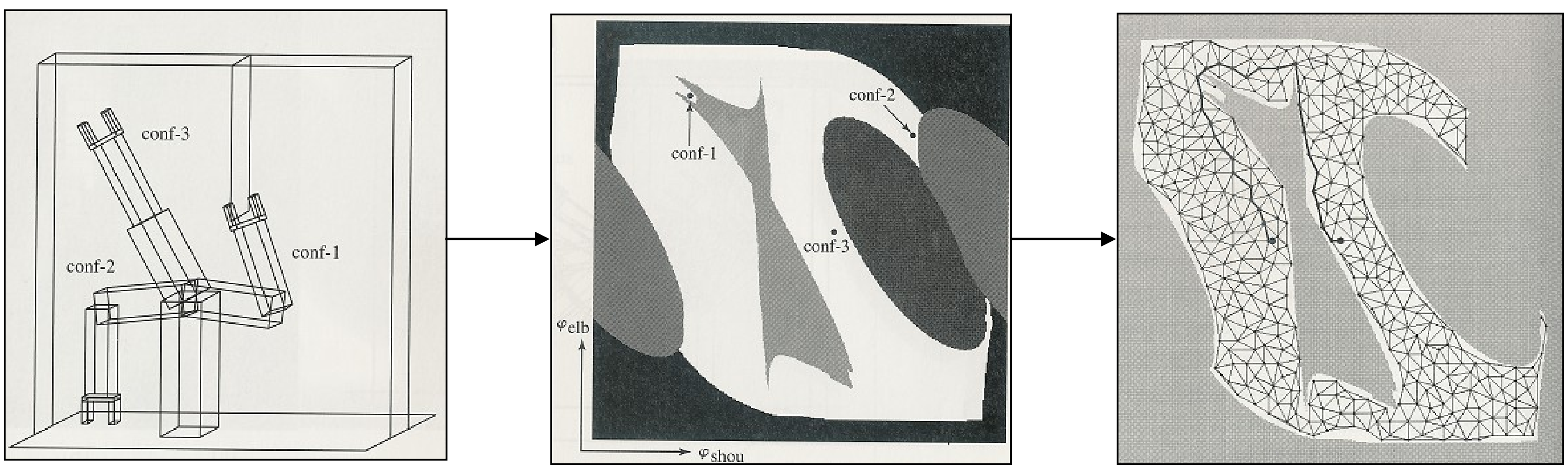
Main observation:
The space is continuous and rather benign!



the example above is borrowed from "AI: A Modern Approach" by S. RusselL & P. Norvig

Resolution Complete vs. Sampling-based Planning

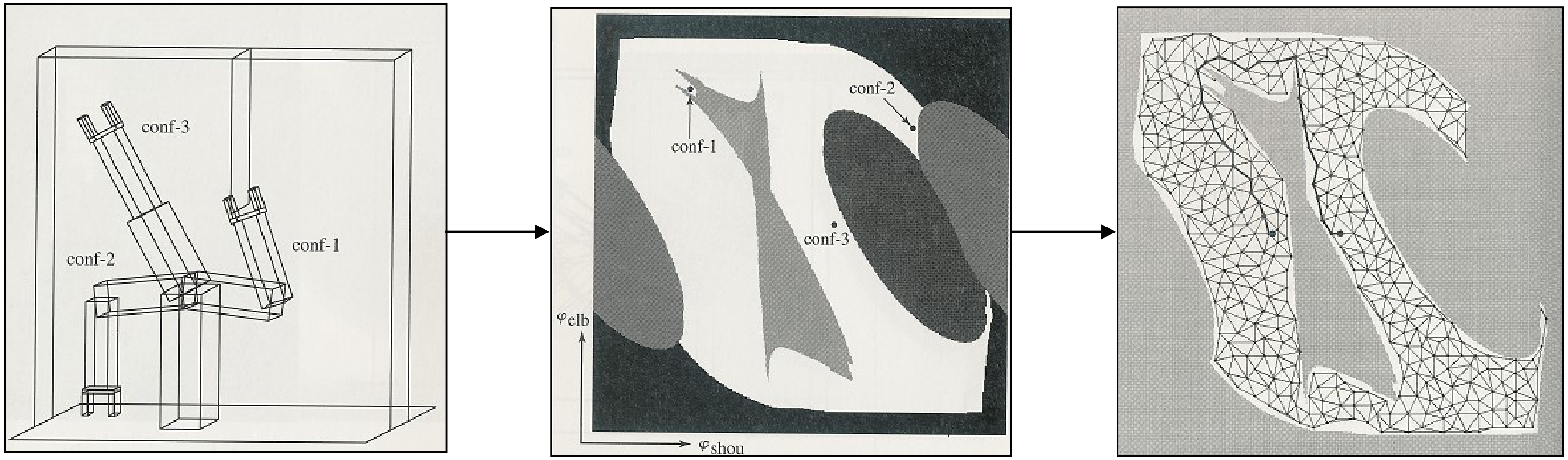
- Sampling-based planning:
  - generate a sparse (sample-based) representation (graph) of a free C-space (C_{free})
  - search the generated representation for a solution



the example above is borrowed from "AI: A Modern Approach" by S. RusselL & P. Norvig

Resolution Complete vs. Sampling-based Planning

- Sampling-based planning:
  - provide **probabilistic** completeness guarantees
    - guaranteed to find a solution, if one exists, but only in the limit of the number of samples (that is, only as the number of samples approaches infinity)
  - well-suited for high-dimensional planning



the example above is borrowed from "AI: A Modern Approach" by S. RussellL & P. Norvig

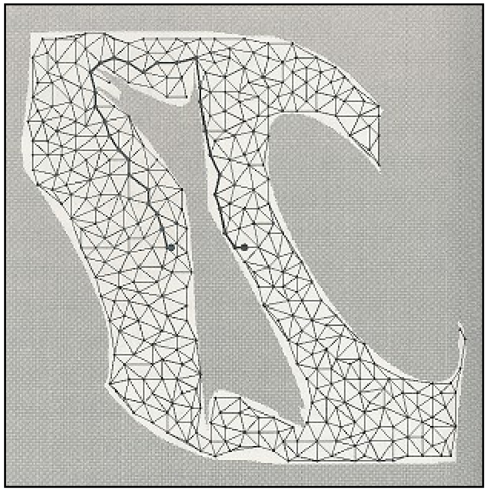
Main Questions in Sampling-based Planning

- How to select samples to construct a "good" graph
- How to search the graph
- Can we interleave these steps

Probabilistic Roadmaps (PRMs)

Step 1. Preprocessing Phase: Build a roadmap (graph) $\mathcal{G}$ which, hopefully, should be accessible from any point in C_{free}

Step 2. Query Phase: Given a start configuration q_I and goal configuration q_G, connect them to the roadmap $\mathcal{G}$ using a local planner, and then search the augmented roadmap for a shortest path from q_I to q_G

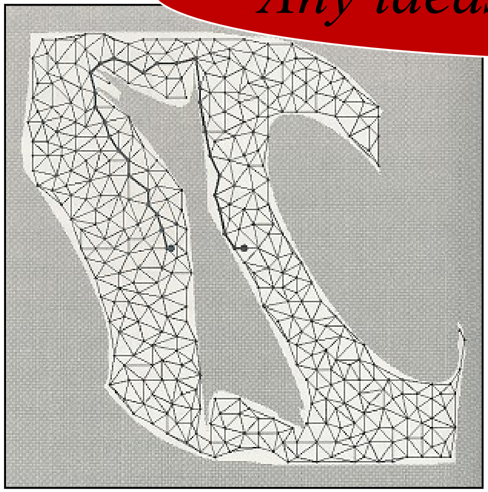


Probabilistic Roadmaps (PRMs)

Step 1. Preprocessing Phase: Build a roadmap (graph) $\mathcal{G}$ which, hopefully, should be accessible from any point in C_{free}

Step 2. Query Phase: Given a start configuration q_I and goal configuration q_G, connect them to the roadmap $\mathcal{G}$ using a local planner, and then search the augmented roadmap for a shortest path from q_I to q_G

Any ideas for the local planner?

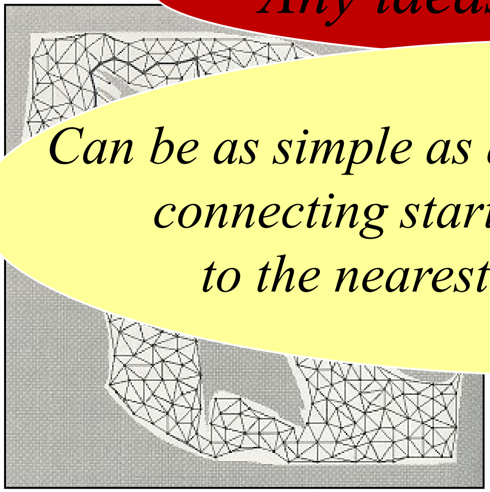


Probabilistic Roadmaps (PRMs)

Step 1. Preprocessing Phase: Build a roadmap (graph) $\mathcal{G}$ which, hopefully, should be accessible from any point in C_{free}

Step 2. Query Phase: Given a start configuration q_I and goal configuration q_G, connect them to the roadmap $\mathcal{G}$ using a local planner, and then search the augmented roadmap for a shortest path from q_I to q_G

Any ideas for the local planner?



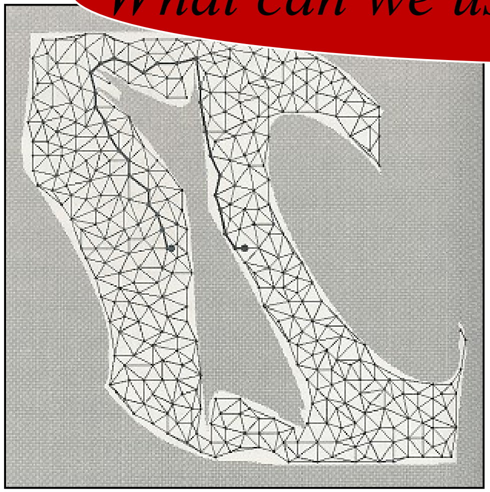
Can be as simple as a straight line (interpolation) connecting start (or goal) configuration to the nearest vertex in the roadmap

Probabilistic Roadmaps (PRMs)

Step 1. Preprocessing Phase: Build a roadmap (graph) $\mathcal{G}$ which, hopefully, should be accessible from any point in C_{free}

Step 2. Query Phase: Given a start configuration q_I and goal configuration q_G, connect them to the roadmap $\mathcal{G}$ using a local planner, and then search the augmented roadmap for a shortest path from q_I to q_G

What can we use to search the roadmap?



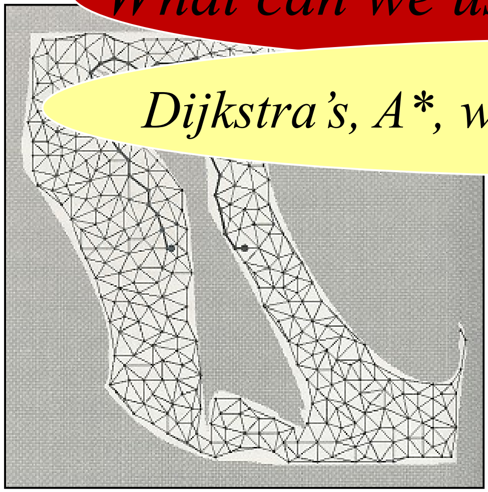
Probabilistic Roadmaps (PRMs)

Step 1. Preprocessing Phase: Build a roadmap (graph) $\mathcal{G}$ which, hopefully, should be accessible from any point in C_{free}

Step 2. Query Phase: Given a start configuration q_I and goal configuration q_G, connect them to the roadmap $\mathcal{G}$ using a local planner, and then search the augmented roadmap for a shortest path from q_I to q_G

What can we use to search the roadmap?

Dijkstra's, A, wA*, and any other variant…*

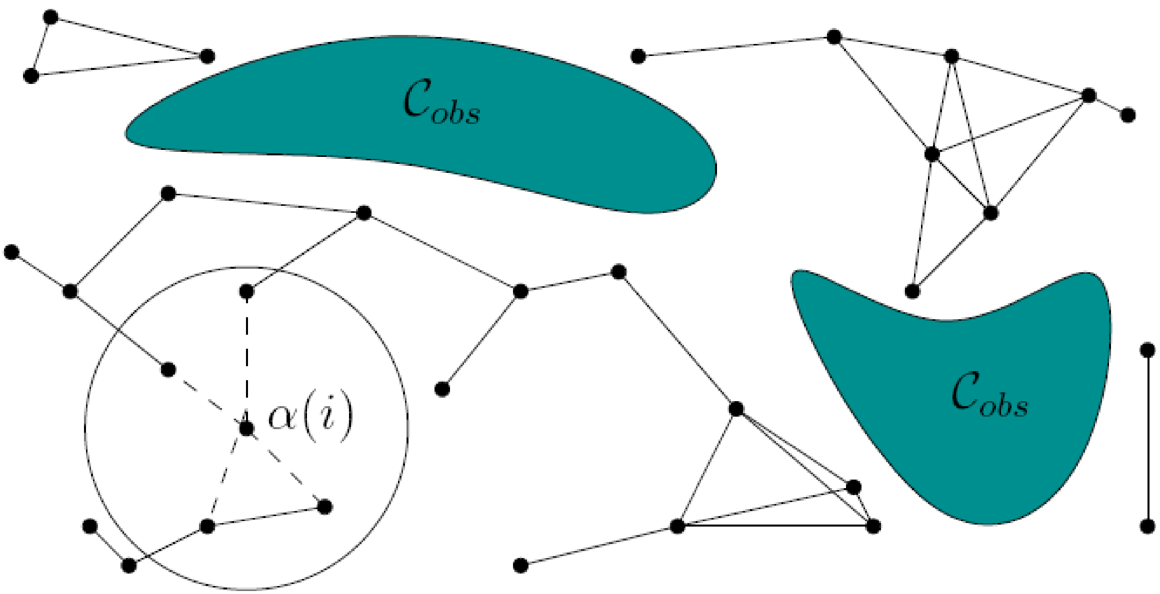


Probabilistic Roadmaps (PRMs)

Step 1: Preprocessing Phase.

BUILD_ROADMAP

$\mathcal{G}$.init(); $i \leftarrow 0$;
while $i < N$
    if $\alpha(i) \in \mathcal{C}_{free}$ **then**
        $\mathcal{G}$.add_vertex($\alpha(i)$); $i \leftarrow i + 1$;
        for each $q \in$ NEIGHBORHOOD($\alpha(i)$,$\mathcal{G}$)
            if ((**not** $\mathcal{G}$.same_component($\alpha(i), q$)) **and** CONNECT($\alpha(i), q$)) **then**
                $\mathcal{G}$.add_edge($\alpha(i), q$);



borrowed from "Planning Algorithms" by S. LaValle

Probabilistic Roadmaps (PRMs)

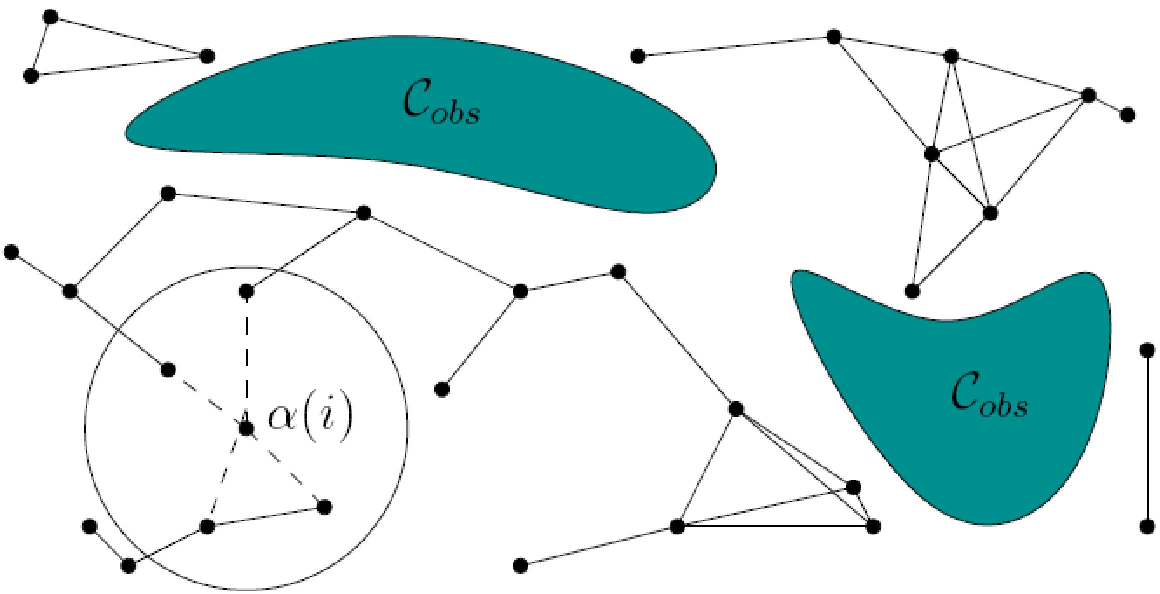
Step 1: Preprocessing Phase.

BUILD_ROADMAP

$\mathcal{G}$.init(); $i \leftarrow 0$;
while $i < N$
  if $\alpha(i) \in \mathcal{C}_{free}$ **then**
    $\mathcal{G}$.add_vertex($\alpha(i)$)
    for each $q \in$ NEIG
      if ((**not** $\mathcal{G}$.same_com
        $\mathcal{G}$.add_edge($\alpha(i), q$);

new i^{th} configuration sample

α(i) is an ith sample in the configuration space.
Each sample can be drawn uniformly
(or more intelligently as described later)



borrowed from "Planning Algorithms" by S. LaValle

Probabilistic Roadmaps (PRMs)

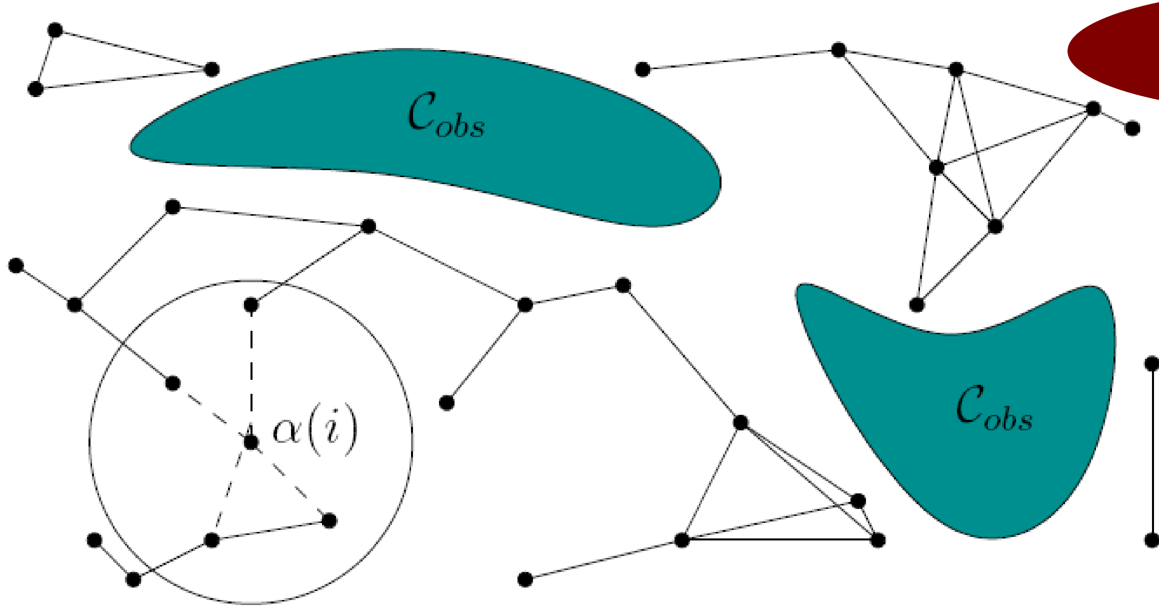
Step 1: Preprocessing Phase.

BUILD_ROADMAP

$\mathcal{G}$.init(); $i \leftarrow 0$;
while $i < N$
  if $\alpha(i) \in \mathcal{C}_{free}$ **then**
    $\mathcal{G}$.add_vertex($\alpha(i)$); $i \leftarrow i + 1$;
    for each $q \in$ NEIGHBORHOOD($\alpha(i)$,$\mathcal{G}$)
      if ((**not** $\mathcal{G}$.same_component($\alpha(i), q$)) **and** CONNECT($\alpha(i), q$)) **then**
        $\mathcal{G}$.add_edge($\alpha(i), q$);

some region around the configuration sample

can be connected by a local planner



borrowed from "Planning Algorithms" by S. LaValle

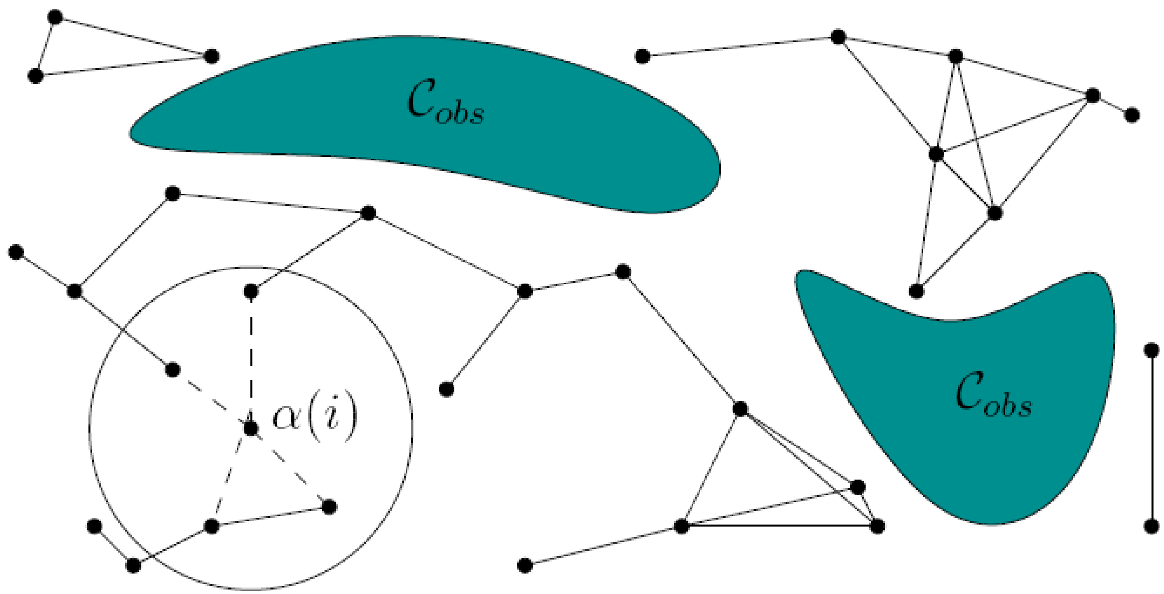
Probabilistic Roadmaps (PRMs)

Step 1: Preprocessing Phase.

BUILD_ROADMAP

```
G.init(); i ← 0;
while i < N
    if α(i) ∈ C_free then
        G.add_vertex(α(i)); i ← i + 1;
        for each q ∈ NEIGHBORHOOD(α(i),G)
            if ((not G.same_component(α(i), q)) and CONNECT(α(i), q)) then
                G.add_edge(α(i), q);
```

can be replaced with:
"number of successors of q < K"



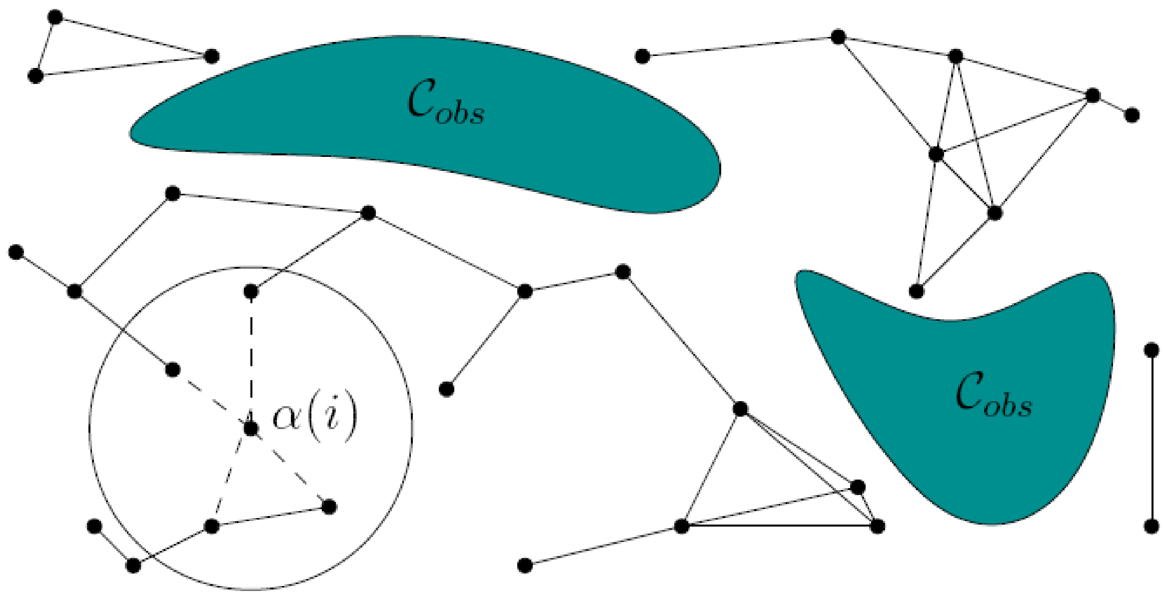
borrowed from "Planning Algorithms" by S. LaValle

Probabilistic Roadmaps (PRMs)

Step 1: Preprocessing Phase.

Efficient implementation of $q \in$ NEIGHBORHOOD$(\alpha(i), \mathcal{G})$

- select K vertices closest to $\alpha(i)$
- select K (often just 1) closest points from each of the components in $\mathcal{G}$
- select all vertices within radius r from $\alpha(i)$



borrowed from "Planning Algorithms" by S. LaValle

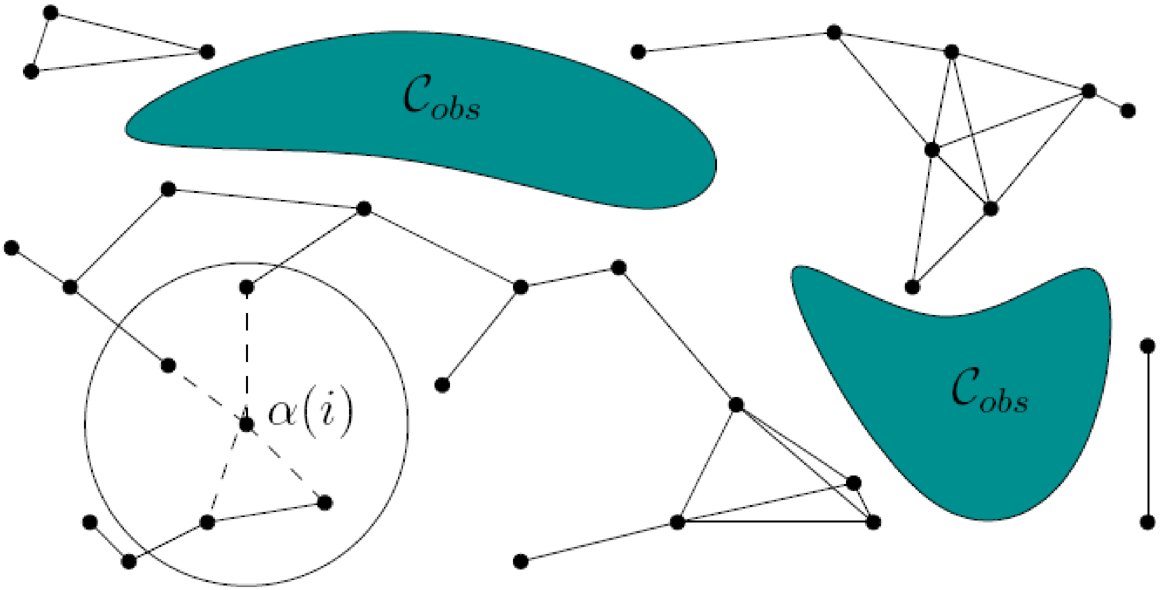
Probabilistic Roadmaps (PRMs)

Step 1: Preprocessing Phase

Why do we need anything better than uniform sampling?

Sampling strategies

- sample uniformly from C_{free}



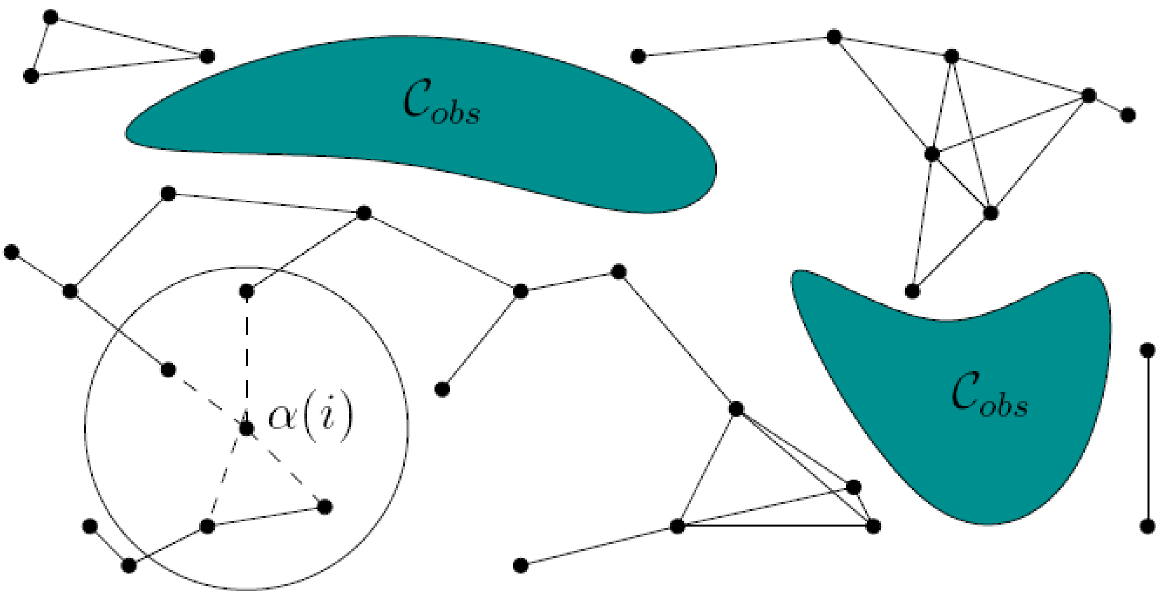
borrowed from "Planning Algorithms" by S. LaValle

Probabilistic Roadmaps (PRMs)

Step 1: Preprocessing Phase.

Sampling strategies

- sample uniformly from C_{free}
- select at random an existing vertex with a probability distribution inversely proportional to how well-connected a vertex is, and then generate a random motion from it to get a sample $\alpha(i)$
- bias sampling towards obstacle boundaries



borrowed from "Planning Algorithms" by S. LaValle

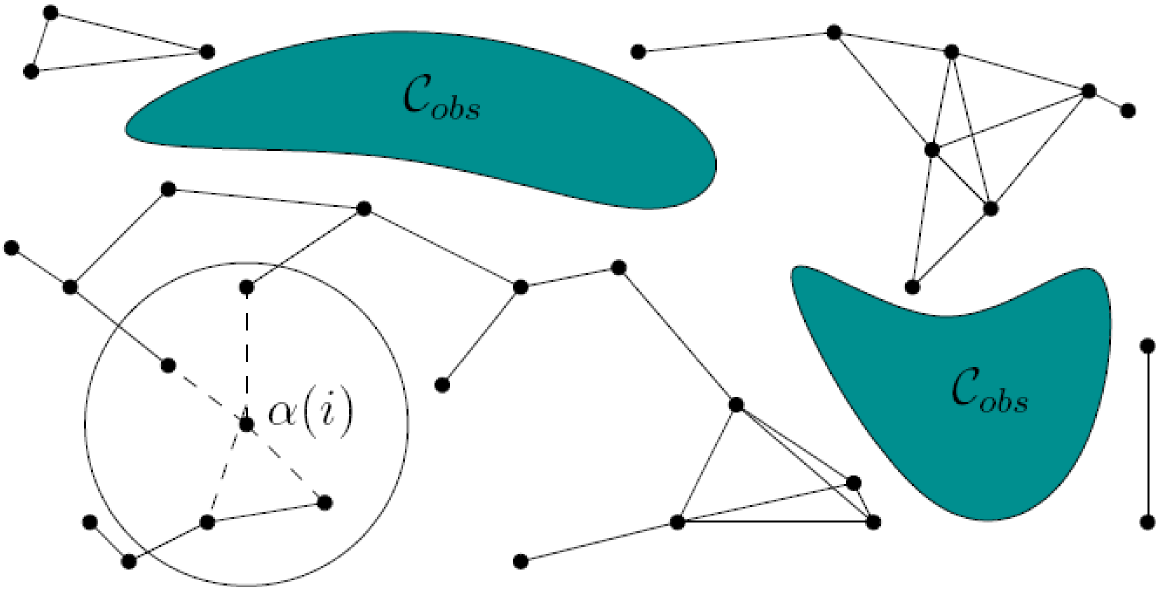
Step 1: Preprocessing Phase.

Sampling strategies

- sample q_1 and q_2 from Gaussian around q_1 and if either is in C_{obs}, then the other one is set as $\alpha(i)$

- sample q_1, q_2 , q_3 from Gaussian around q_2 and set q_2 as $\alpha(i)$ if q_2 is in C_{free}, and q_1 and q_3 are in C_{obs}

- bias sampling away from obstacles



borrowed from "Planning Algorithms" by S. LaValle

What You Should Know…

- Pros and Cons of Resolution-complete approaches (like Grid-based or Lattice-based graphs) vs. Sampling-based approaches
- What domains are more suitable for each
- How PRM works