Probabilistic Roadmaps

Dinesh Manocha

Based on Slides from Hsu and Latombe

Free-Space and C-Space Obstacle

- How do we know whether a configuration is in the free space?
- Computing an explicit representation of the free-space boundary is very hard in practice?
  - High theoretical complexity
  - Issues in robust implementation

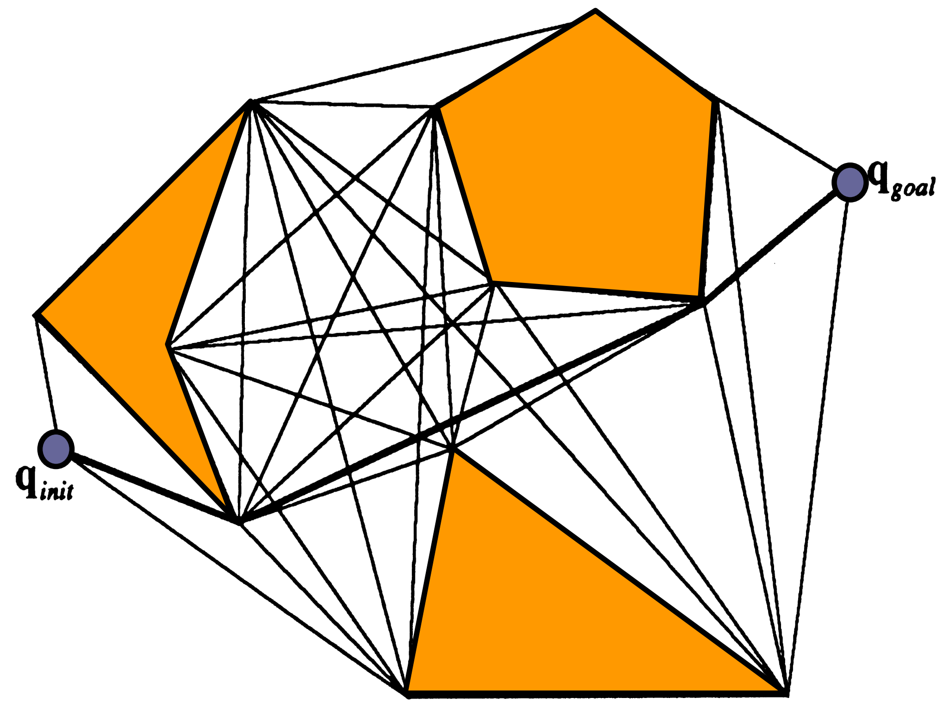
Free-Space and C-Space Obstacle

- How do we know whether a configuration is in the free space?
- Computing an explicit representation of the free-space is very hard in practice?
- Solution: Compute the position of the robot at that configuration in the workspace. Explicitly check for collisions with any obstacle at that position:
  - If colliding, the configuration is within C-space obstacle
  - Otherwise, it is in the free space
- Performing collision checks is relative simple

Two geometric primitives in configuration space

- **CLEAR**(q)
  Is configuration q collision free or not?

- **LINK**(q, q')
  Is the straight-line path between q and q' collision-free?
  - **Proximity**(q, q')
    Are two configuration q and q' close to each other?



Difficulty with classic approaches

- Running time increases exponentially with the dimension of the configuration space.
  - For a d-dimension grid with 10 grid points on each dimension, how many grid cells are there?

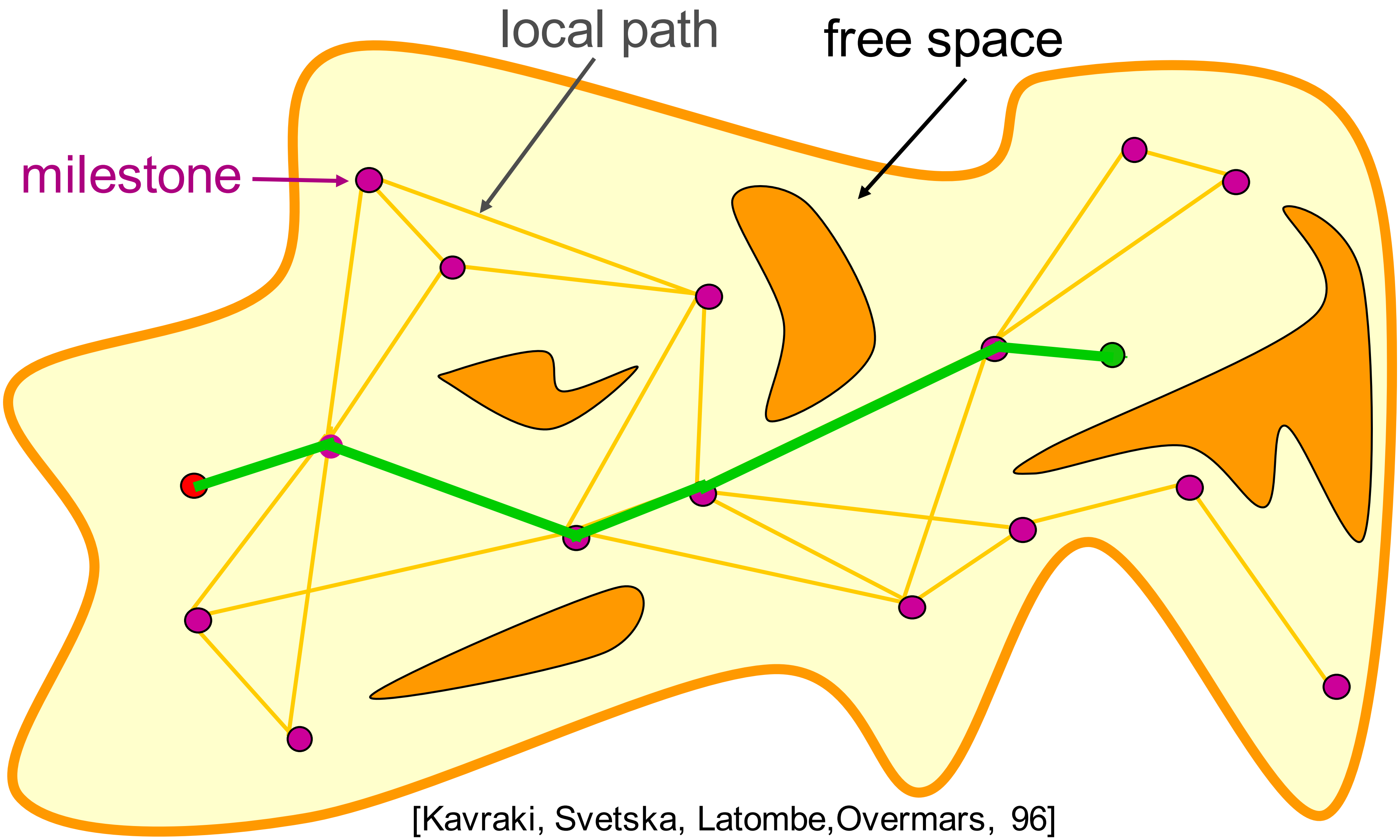
$$10^d$$

- Several variants of the path planning problem have been proven to be PSPACE-hard.

Completeness

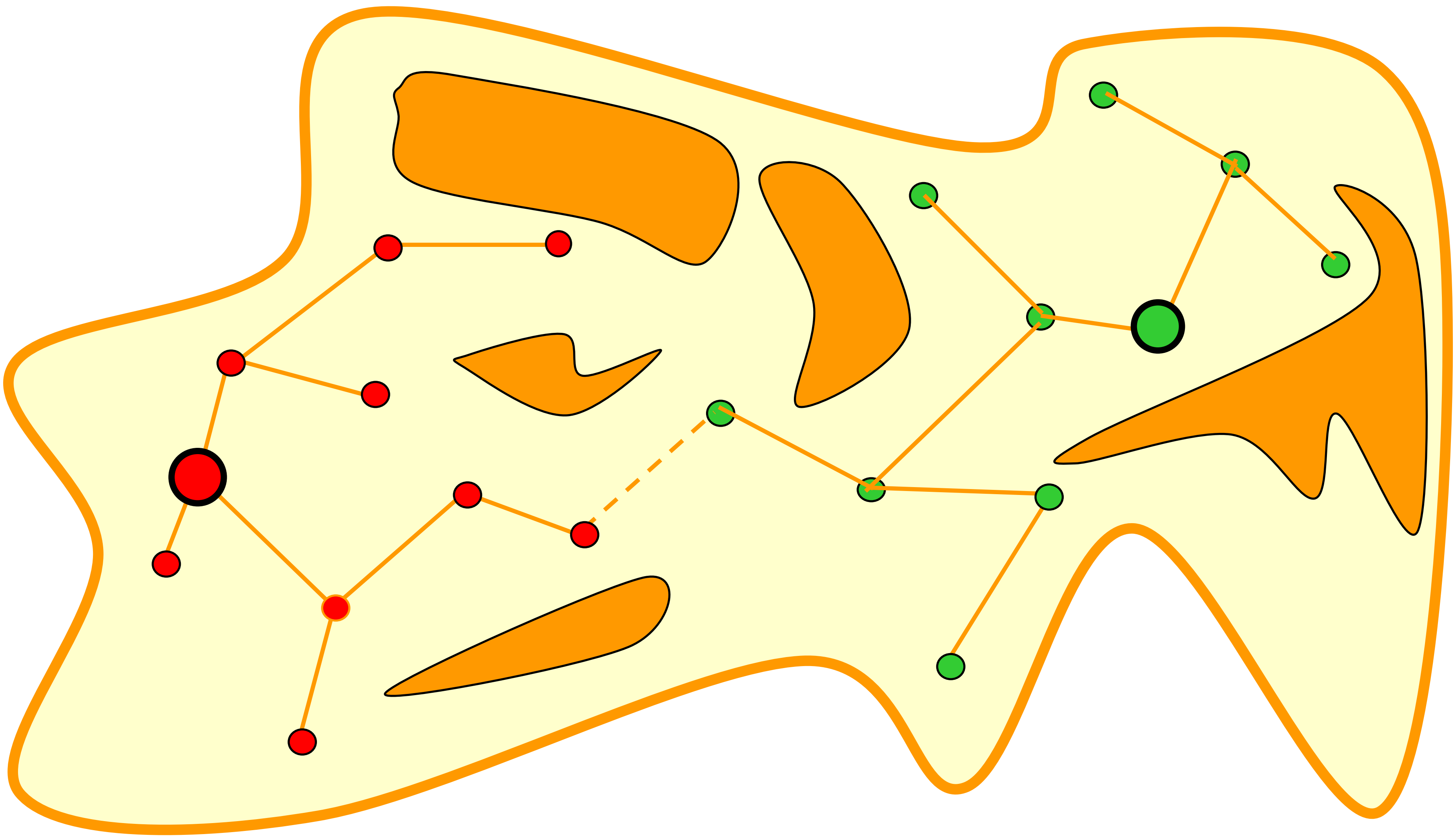
- Complete algorithm → Slow
  - A **complete** algorithm finds a path if one exists and reports no otherwise.
  - Example: Canny's roadmap method
- Heuristic algorithm → Unreliable
  - Example: potential field
- **Probabilistic completeness**
  - Intuition: If there is a solution path, the algorithm will find it with high probability.

Probabilistic Roadmap (PRM): multiple queries



[Kavraki, Svetska, Latombe,Overmars, 96]

Probabilistic Roadmap (PRM): single query



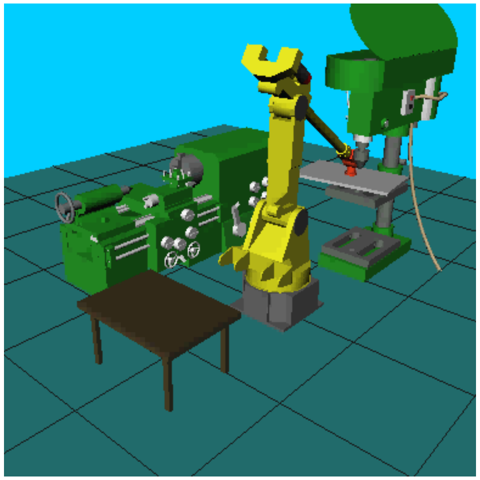
Multiple-Query PRM

Classic multiple-query PRM

- *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces,* L. Kavraki *et al*., 1996.

Assumptions

- Static obstacles
- Many queries to be processed in the same environment
- Examples
  - Navigation in static virtual environments
  - Robot manipulator arm in a workcell



Overview

- Precomputation: roadmap construction
  - Uniform sampling
  - Resampling (expansion)
- Query processing

Uniform sampling

Input: geometry of the moving object & obstacles

Output: roadmap G = (V, E)

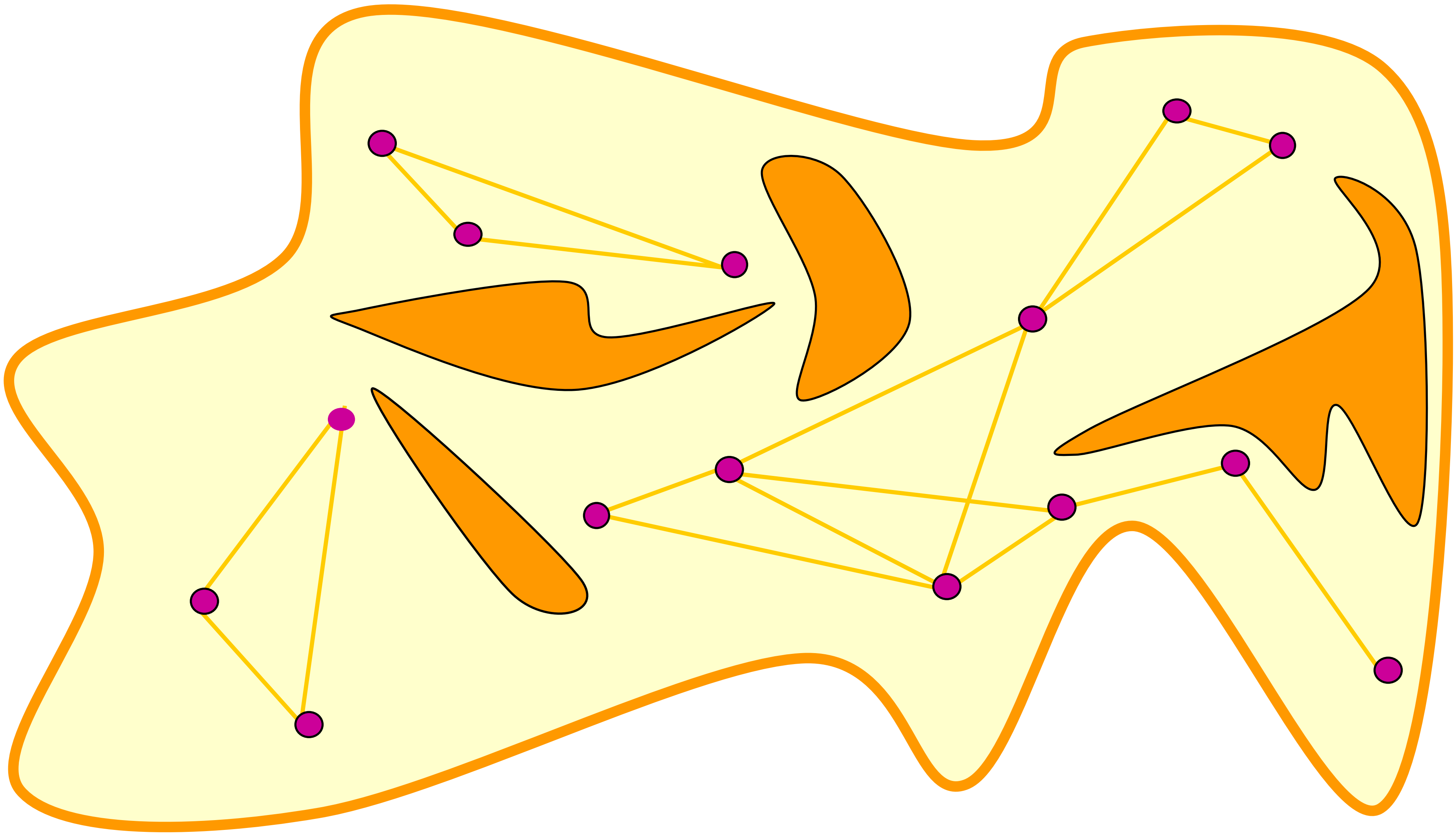
V ← ∅ and E ← ∅.
repeat
  q ← a configuration sampled uniformly at random from C.
  if CLEAR(q) **then**
    Add q to V.
    N_q ← a set of nodes in V that are close to q.
    for each q′ ∈ N_q, in order of increasing d(q,q′)
      if LINK(q′,q) **then**
        Add an edge between q and q′ to E.

Some terminology

- The graph G is called a **probabilistic roadmap**.
- The nodes in G are called **milestones**.

Difficulty

- Many small connected components



Resampling (expansion)

- Failure rate

$$r(q) = \frac{\text{no. failed LINK}}{\text{no. LINK}}$$

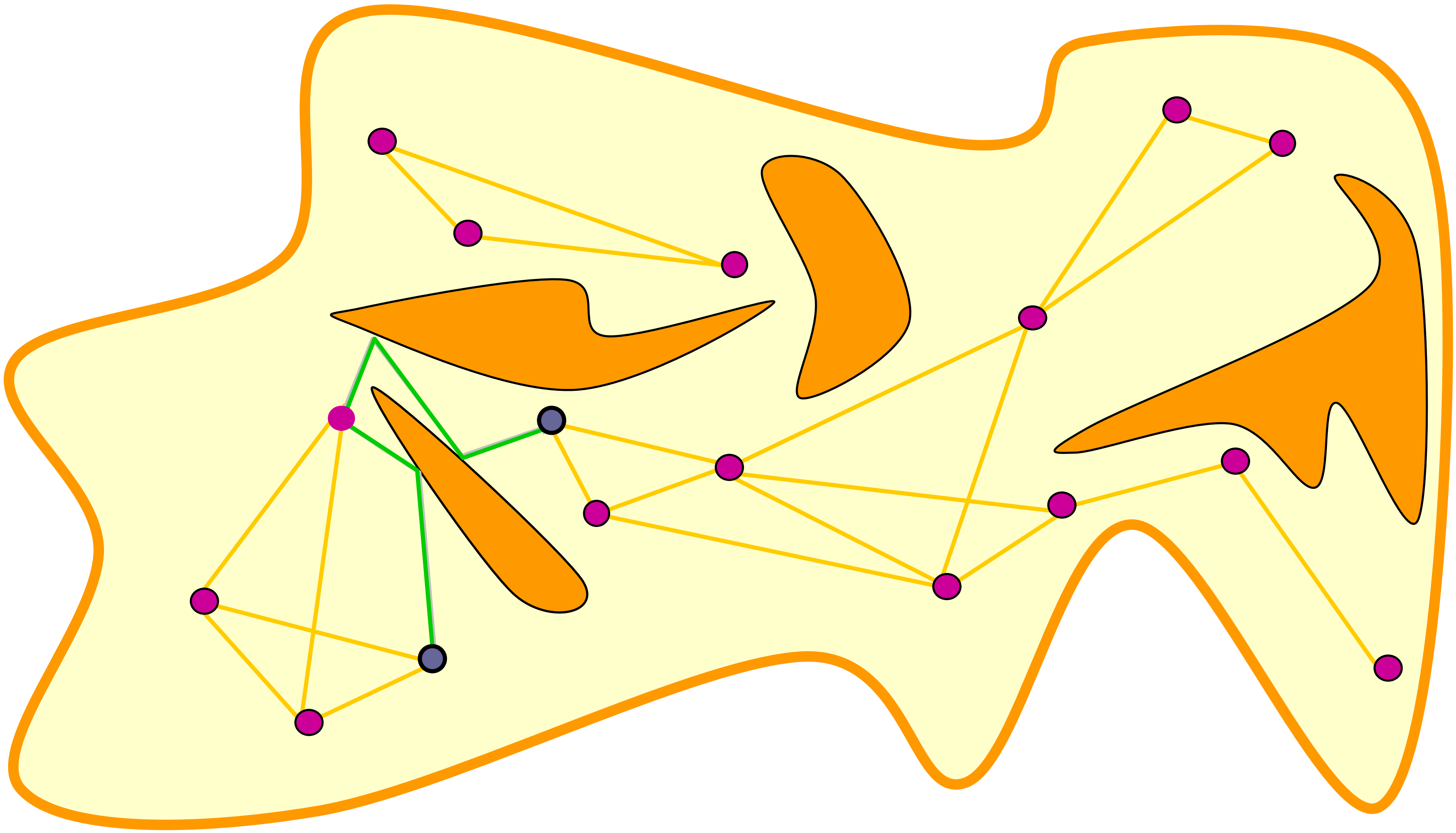
- Weight

$$w(q) = \frac{r(q)}{\sum_p r(p)}$$

- Resampling probability

$$\Pr(q) = w(q)$$

Resampling (expansion)



Query processing

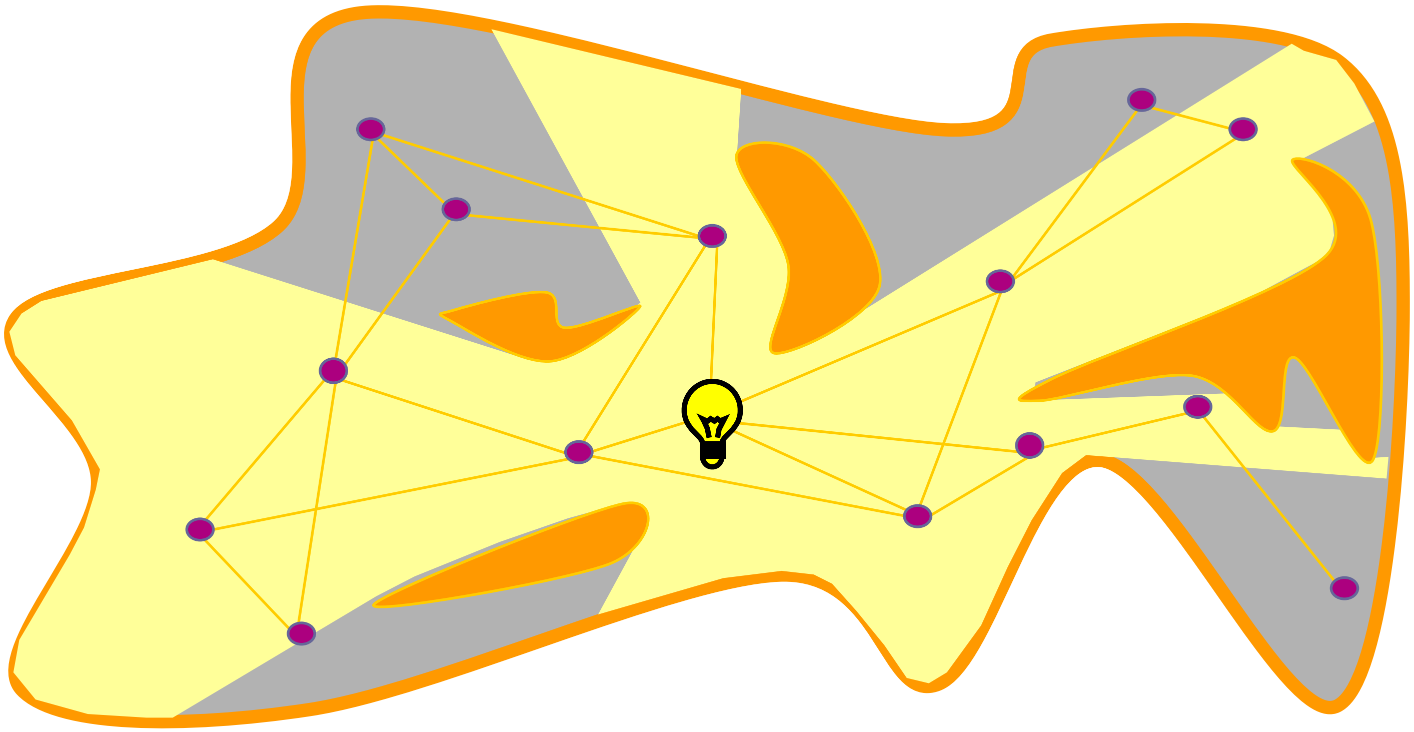
- Connect q_{init} and q_{goal} to the roadmap
- Start at q_{init} and q_{goal}, perform a random walk, and try to connect with one of the milestones nearby
- Try multiple times

Error

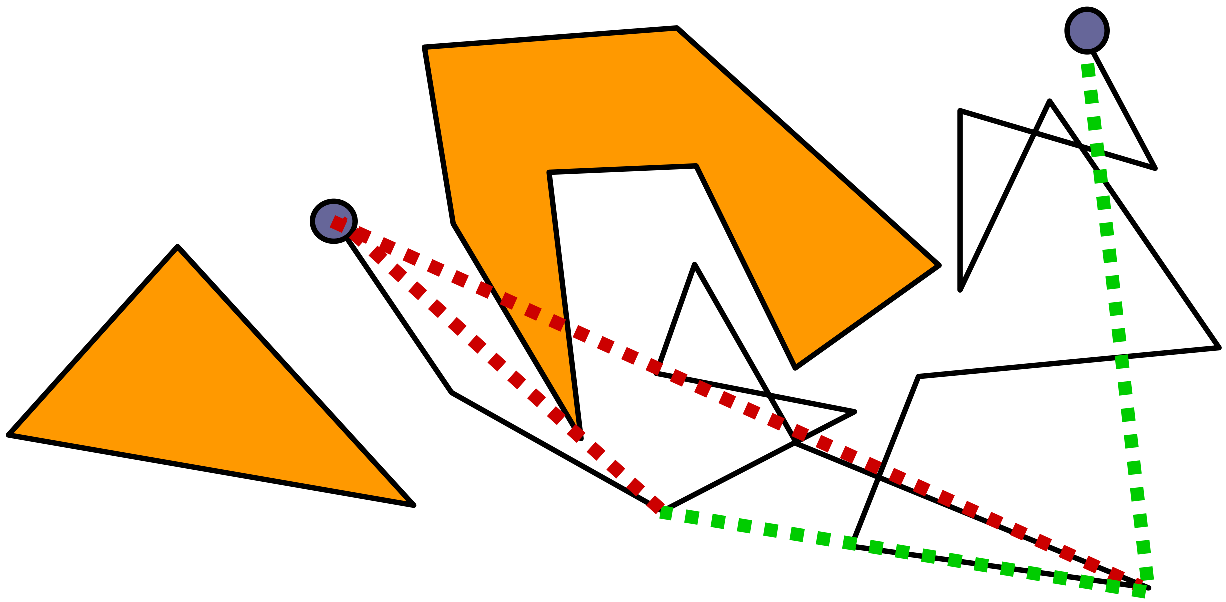
- If a path is returned, the answer is always correct.
- If no path is found, the answer may or may not be correct. We hope it is correct with high probability.

Why does it work? Intuition

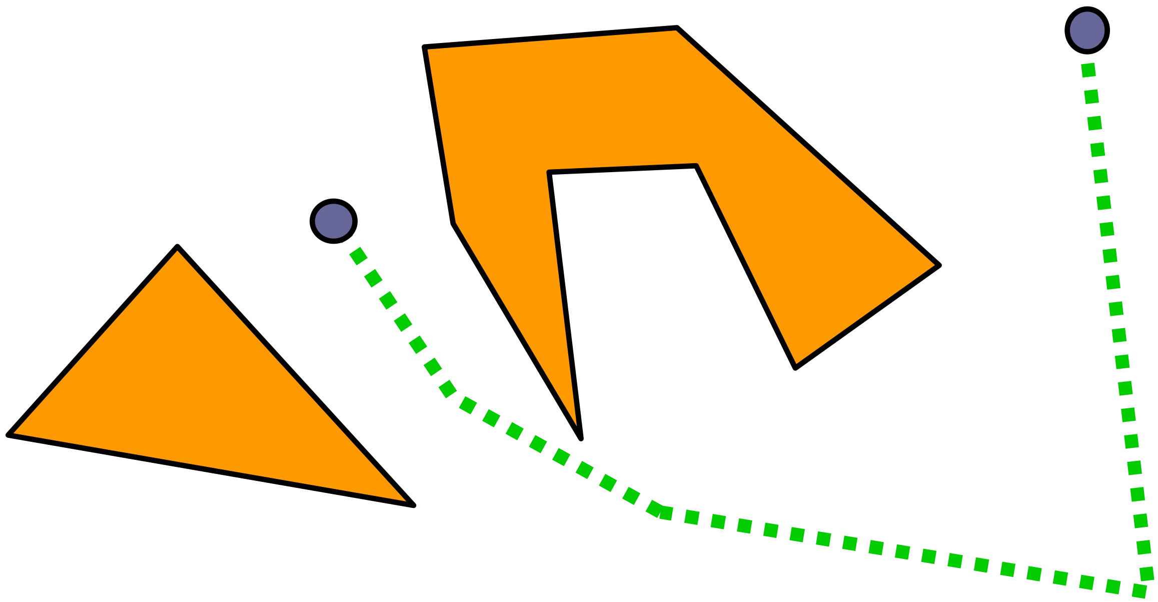
- A small number of milestones **almost** “cover” the **entire** configuration space.



Smoothing the path



Smoothing the path



Summary

- What probability distribution should be used for sampling milestones?
- How should milestones be connected?
- A path generated by a randomized algorithm is usually jerky. How can a path be smoothed?

Single-Query PRM

Lazy PRM

- *Path Planning Using Lazy PRM*, R. Bohlin & L. Kavraki, 2000.

Precomputation: roadmap construction

- Nodes
  - Randomly chosen configurations, which may or may **not** be collision-free
  - No call to **CLEAR**
- Edges
  - an edge between two nodes if the corresponding configurations are close according to a suitable metric
  - no call to **LINK**

Query processing: overview

1. Find a shortest path in the roadmap
2. Check whether the nodes and edges in the path are collision.
3. If yes, then done. Otherwise, remove the nodes or edges in violation. Go to (1).

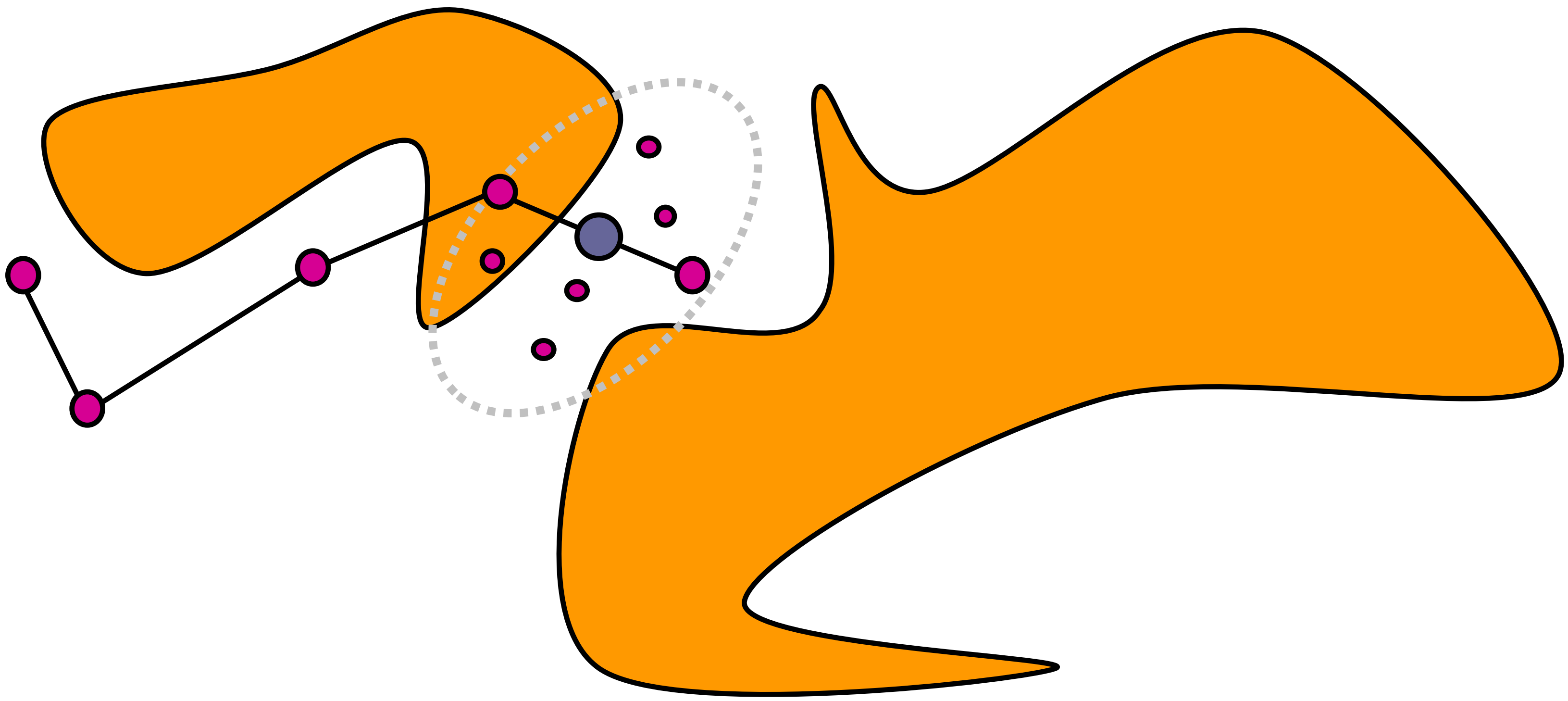
We either find a collision-free path, or exhaust all paths in the roadmap and declare failure.

Query processing: details

- Find the shortest path in the roadmap
  - A* algorithm
  - Dijkstra’s algorithm
- Check whether nodes and edges are collisions free
  - **CLEAR**(q)
  - **LINK**(q_0, q_1)

Node enhancement

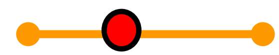
- Select nodes that close the boundary of F



Sampling a Point Uniformly at Random

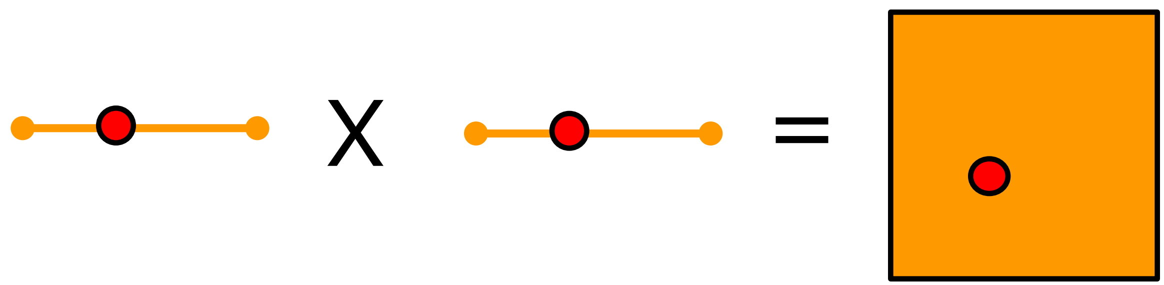
Positions

- Unit interval

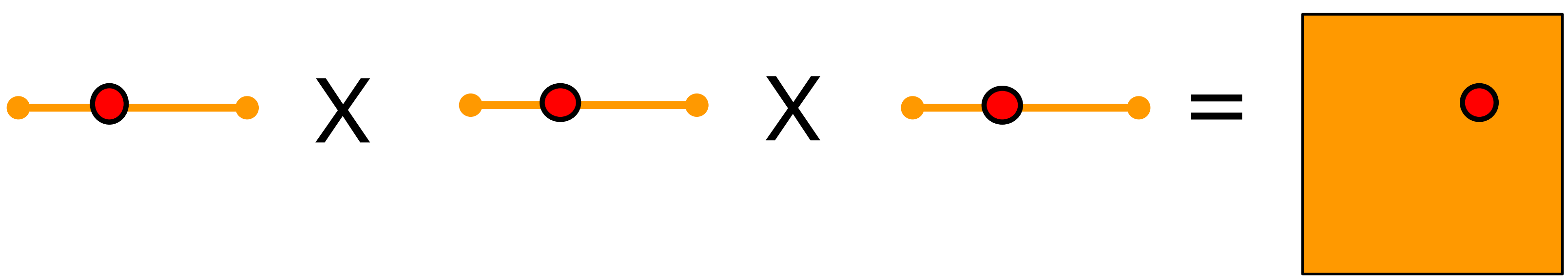


  Pick a random number from [0,1]

- Unit square

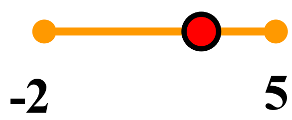


- Unit cube



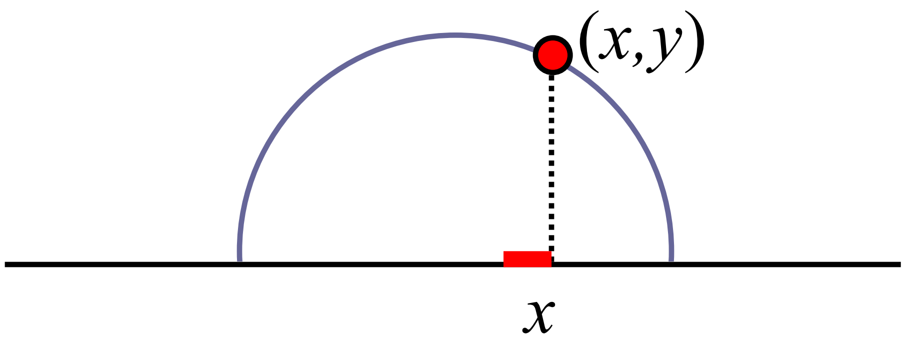
Intervals scaled & shifted

- What shall we do?



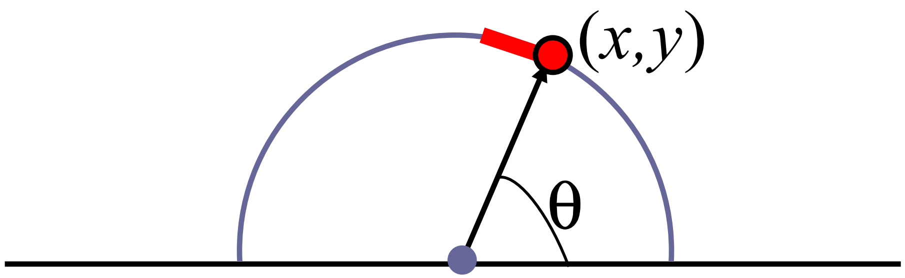
If x is a random number from $[0,1]$, then $7x$-2.

Orientations in 2-D



- Sampling
  1. Pick x uniform at random from [-1,1]
  2. Set $y = \sqrt{1 - x^2}$
- Intervals of same widths are sampled with equal probabilities

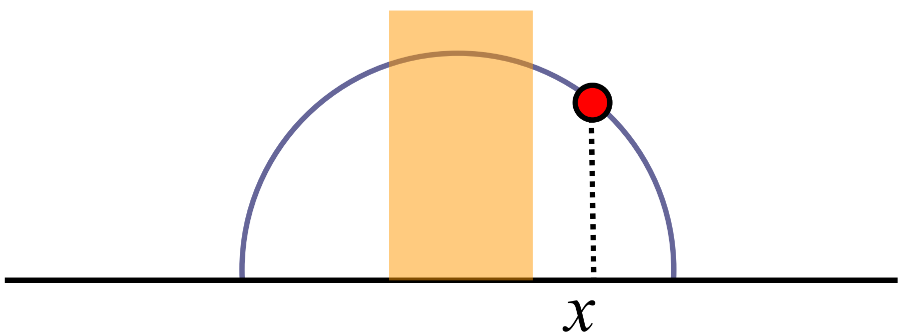
Orientations in 2-D



- Sampling
  1. Pick θ uniformly at random from $[0, 2\pi]$
  2. Set $x = \cos\theta$ and $y = \sin\theta$
- Circular arcs of same angles are sampled with equal probabilities.

What is the difference?

- Both are uniform in some sense.
- For sampling orientations in 2-D, the second method is usually more appropriate.



- The definition of uniform sampling depends on the task at hand and not on the mathematics.

Orientations in 3-D

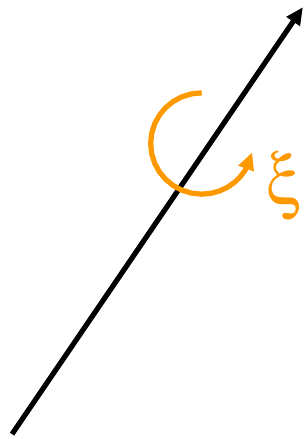
- Unit quaternion

  $(\cos\xi/2, n_x\sin\xi/2, n_y\sin\xi/2, n_z\sin\xi/2)$ with $n_x^2+n_y^2+n_z^2=1$.

- Sample $\mathbf{n}$ and θ separately

- Sample ξ from [0, 2π] uniformly at random

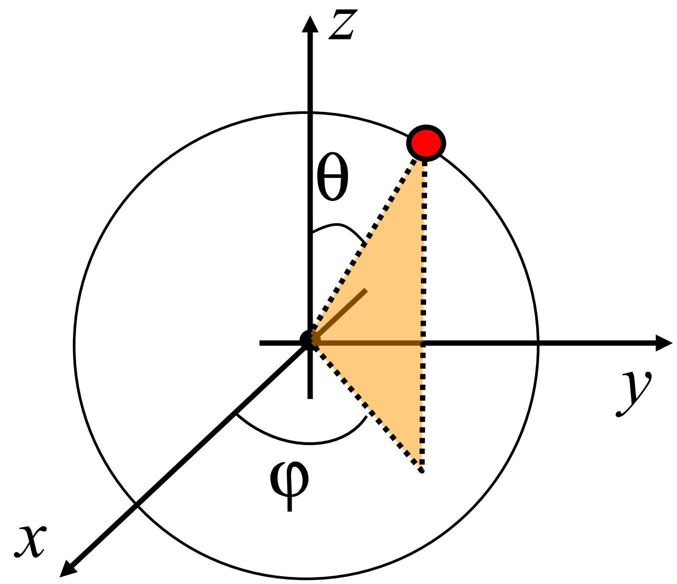
$\mathbf{n} = (n_x, n_y, n_z)$



Sampling a point on the unit sphere

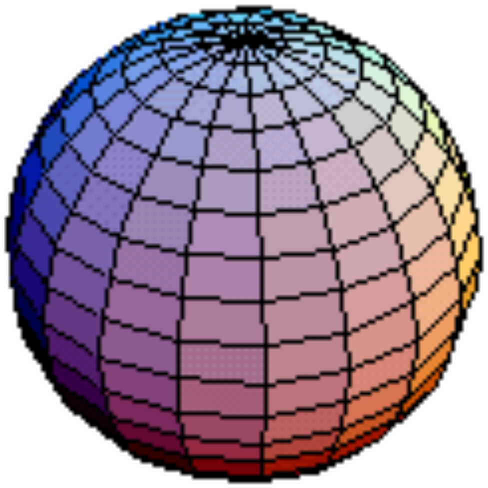
- Longitude and latitude

$$\begin{cases} n_x = \sin\theta\cos\varphi \\ n_y = \sin\theta\sin\varphi \\ n_z = \cos\theta \end{cases}$$



First attempt

- Choose θ and φ uniformly at random from $[0, 2\pi]$ and $[0, \pi]$, respectively.

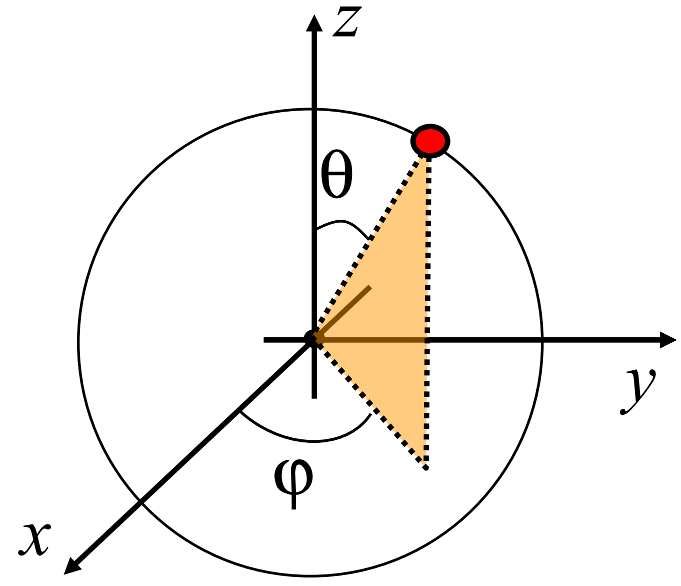


Better solution

- Spherical patches of same areas are sampled with equal probabilities.
- Suppose U_1 and U_2 are chosen uniformly at random from $[0,1]$.

$$\begin{cases} n_z = U_1 \\ n_x = R\cos(2\pi U_2) \\ n_y = R\sin(2\pi U_2) \end{cases}$$

where $R = \sqrt{1-U_1^2}$



Medial Axis based Planning

- Use medial axis based sampling
  - Medial axis: similar to internal Voronoi diagram; set of points that are equidistant from the obstacle
  - Compute approximate Voronoi boundaries using discrete computation
-

Medial Axis based Planning

- Sample the workspace by taking points on the medial axis
  - Medial axis of the workspace (works well for translation degrees of freedom)
  - How can we handle robots with rotational degrees of freedom?