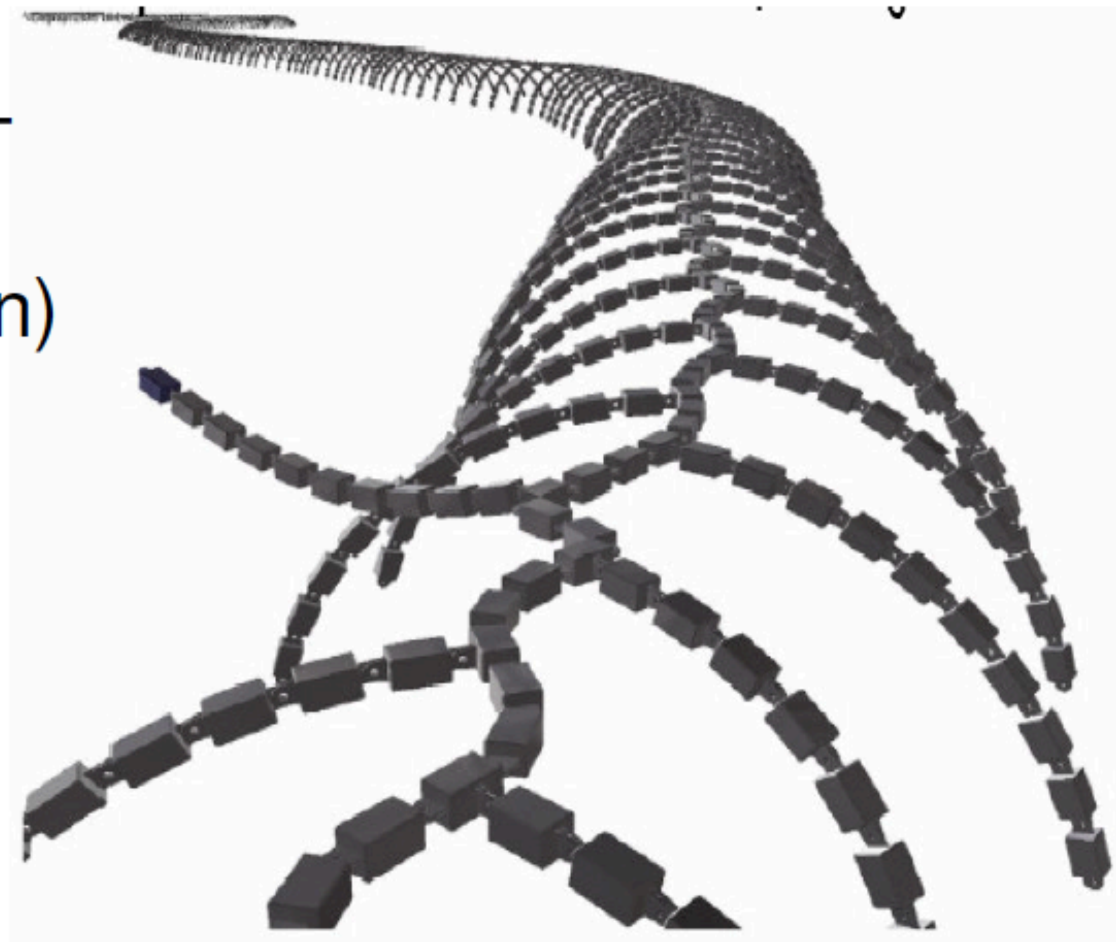
Motion Planning II

D.A. Forsyth
(with a lot of H. Choset, and some J. Li)

Large C-Space Dimension

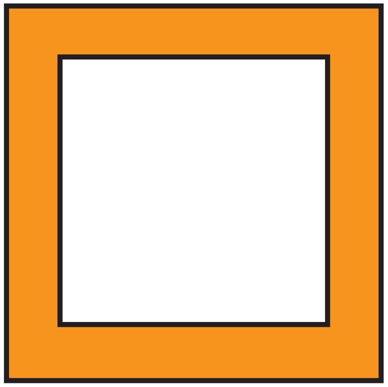
Millipede-like robot (S. Redon)



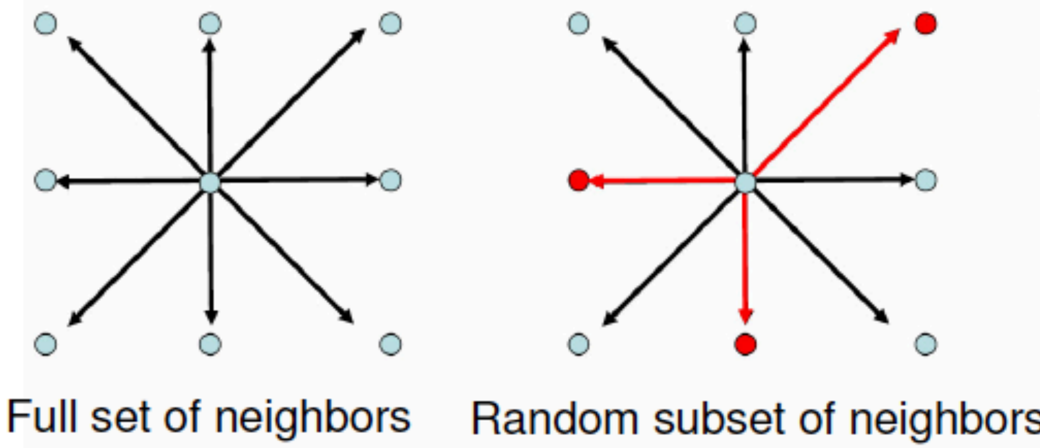
~13,000 DOFs !!!

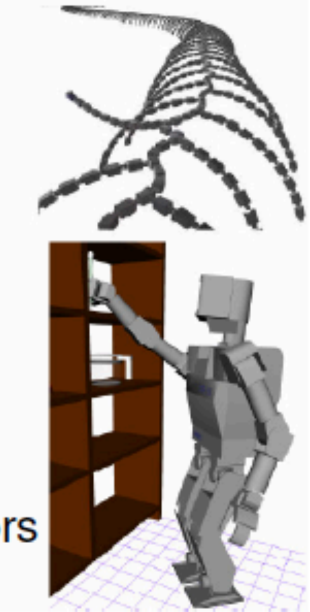
Dimension and its nuisances

- Counting:
  - A d-dimensional cube has 2^d vertices
- Volume:
  - your intuitions about volume are wrong in high dimension
  - consider cubical "orange" in high d
    - skin depth e/2
    - pulp (1-e)
    - volume of pulp:
      - (1-e)^d
    - volume of skin:
      - 1-(1-e)^d
    - IT'S ALL SKIN!
  - Almost all the volume of high d objects is very close to surface



Dealing with C-Space Dimension



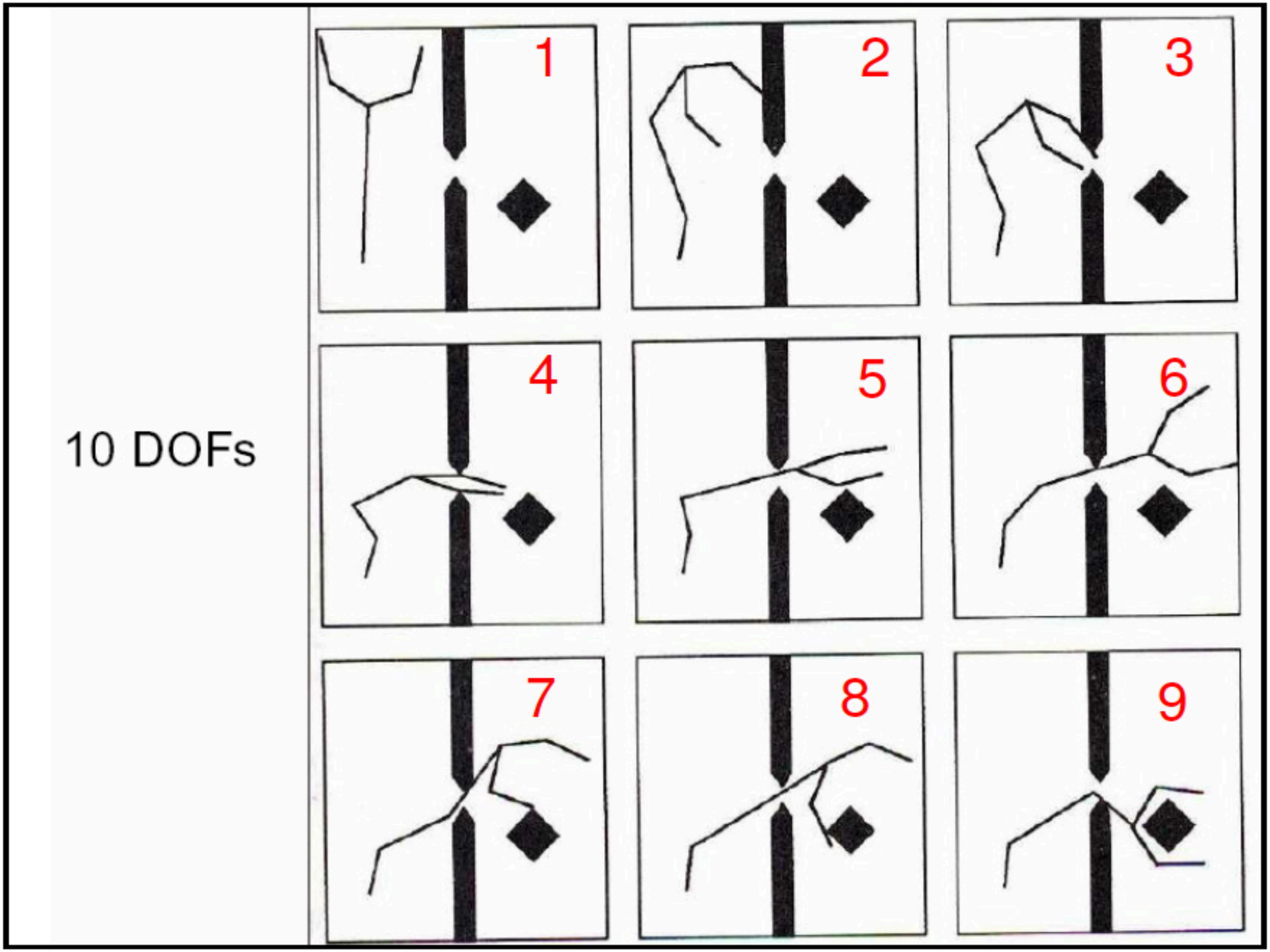


- We should evaluate all the neighbors of the current state, but:
- Size of neighborhood grows exponentially with dimension
- Very expensive in high dimension

Solution:

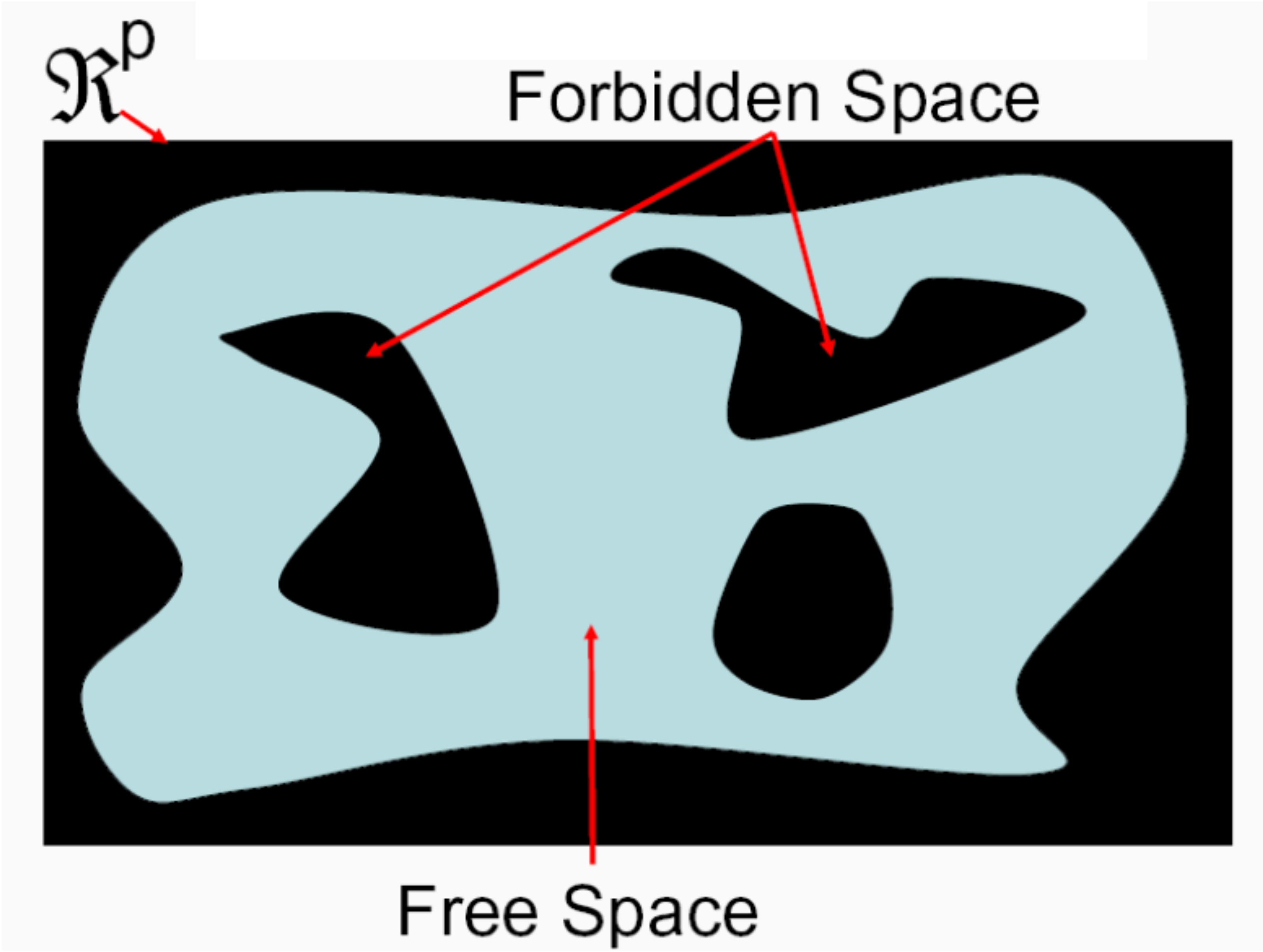
- Evaluate only a random subset of K of the neighbors
- Move to the lowest potential neighbor

Choset slides



Choset slides

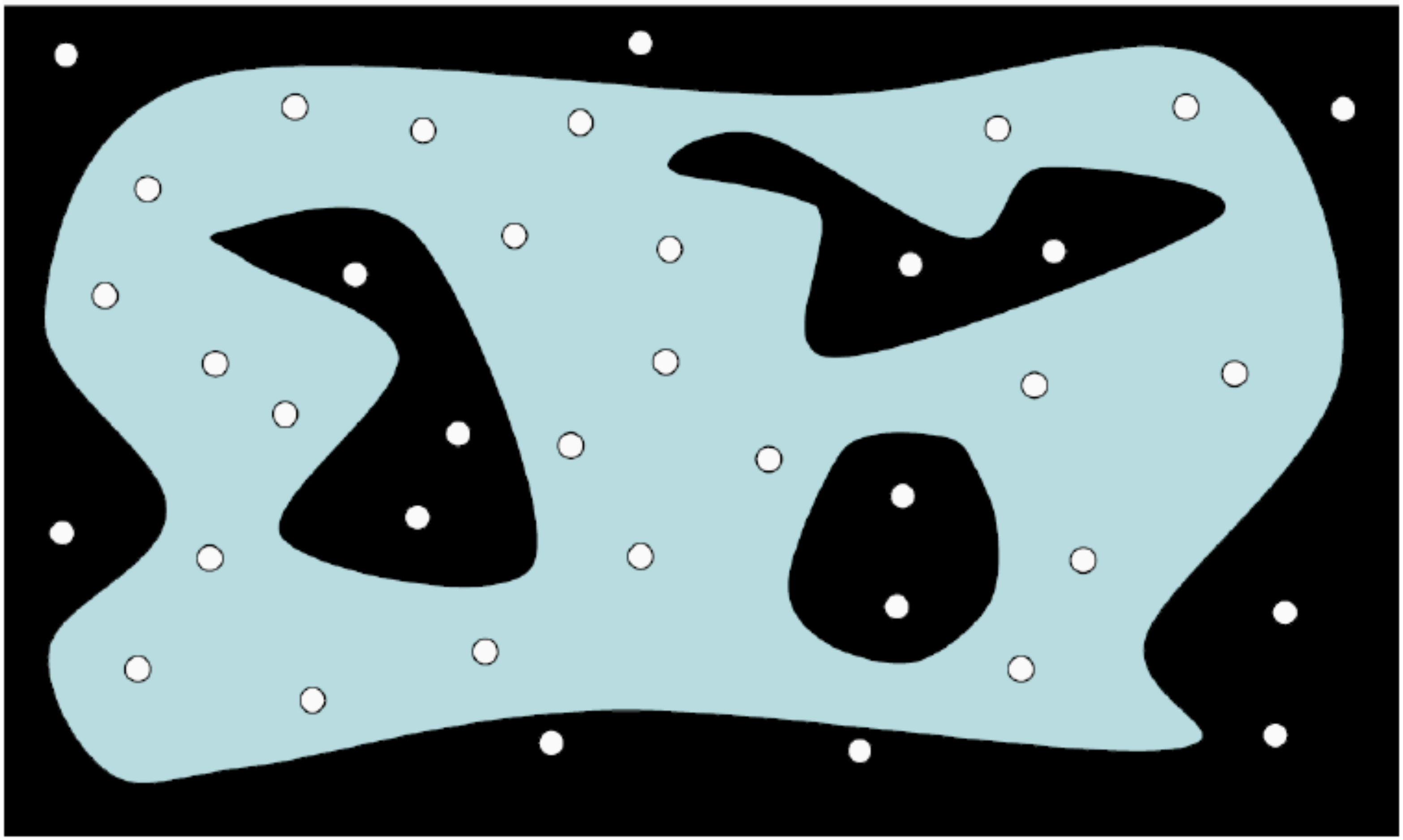
Sampling Techniques



Choset slides

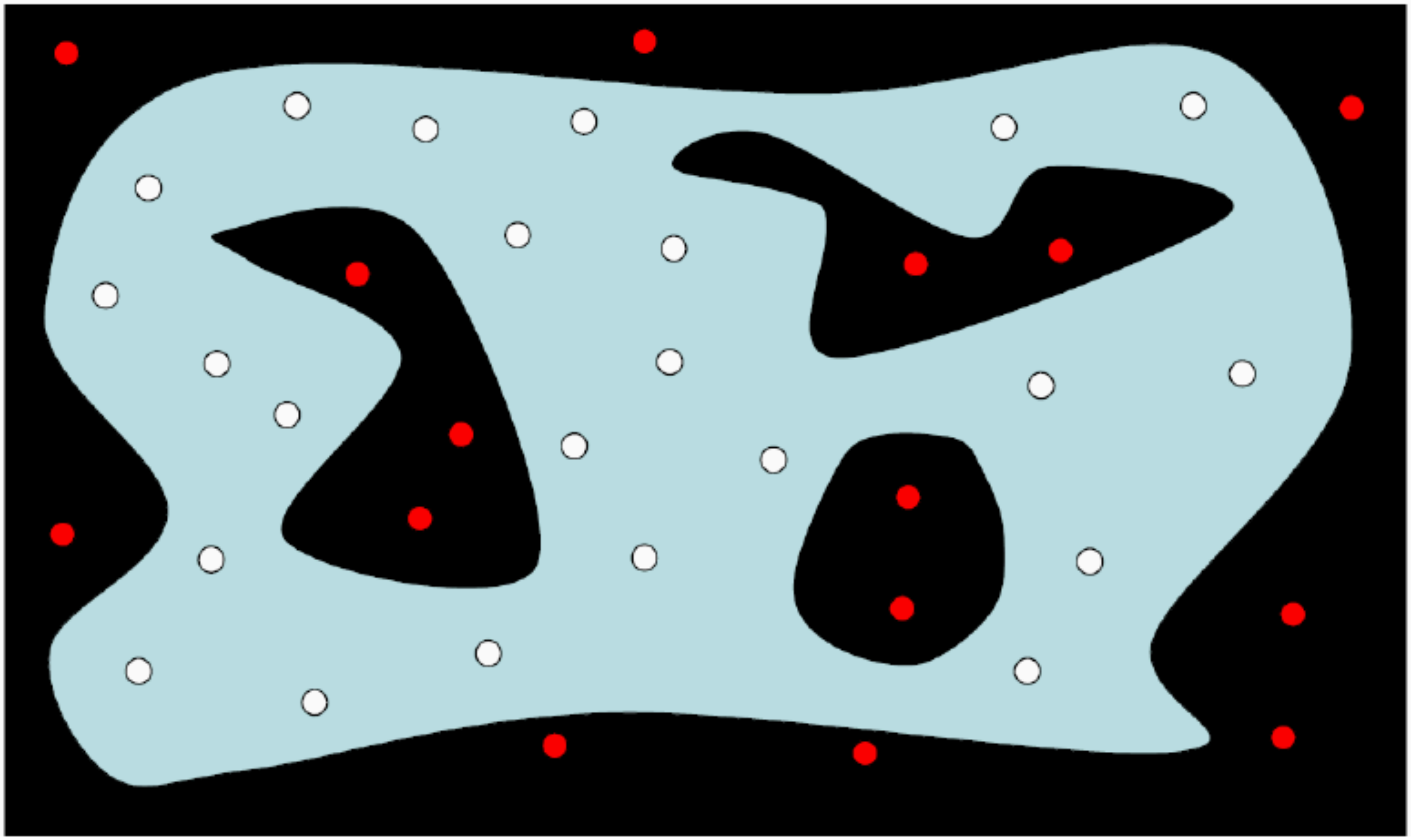
Sampling Techniques

Sample random locations



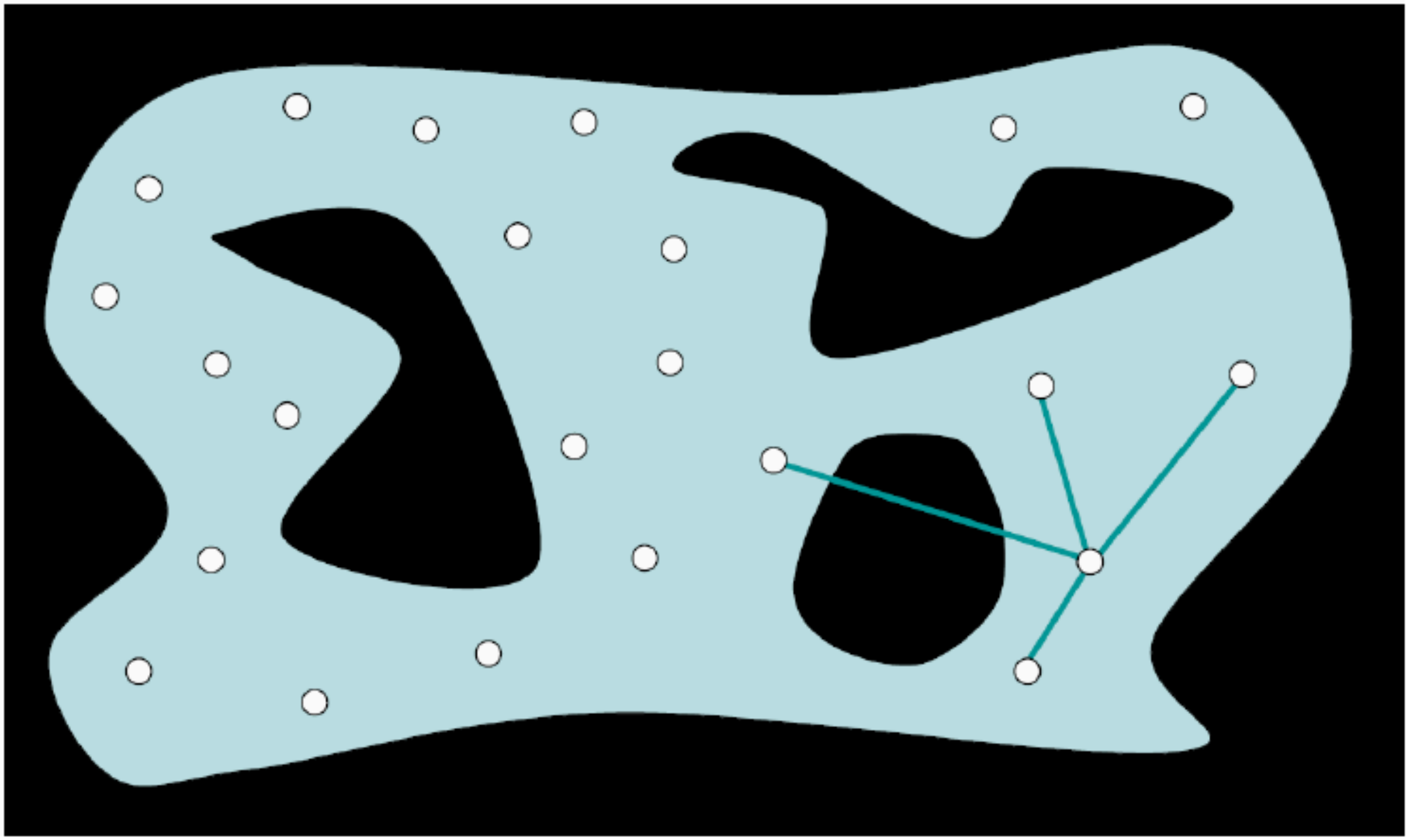
Sampling Techniques

Remove the samples in the forbidden regions



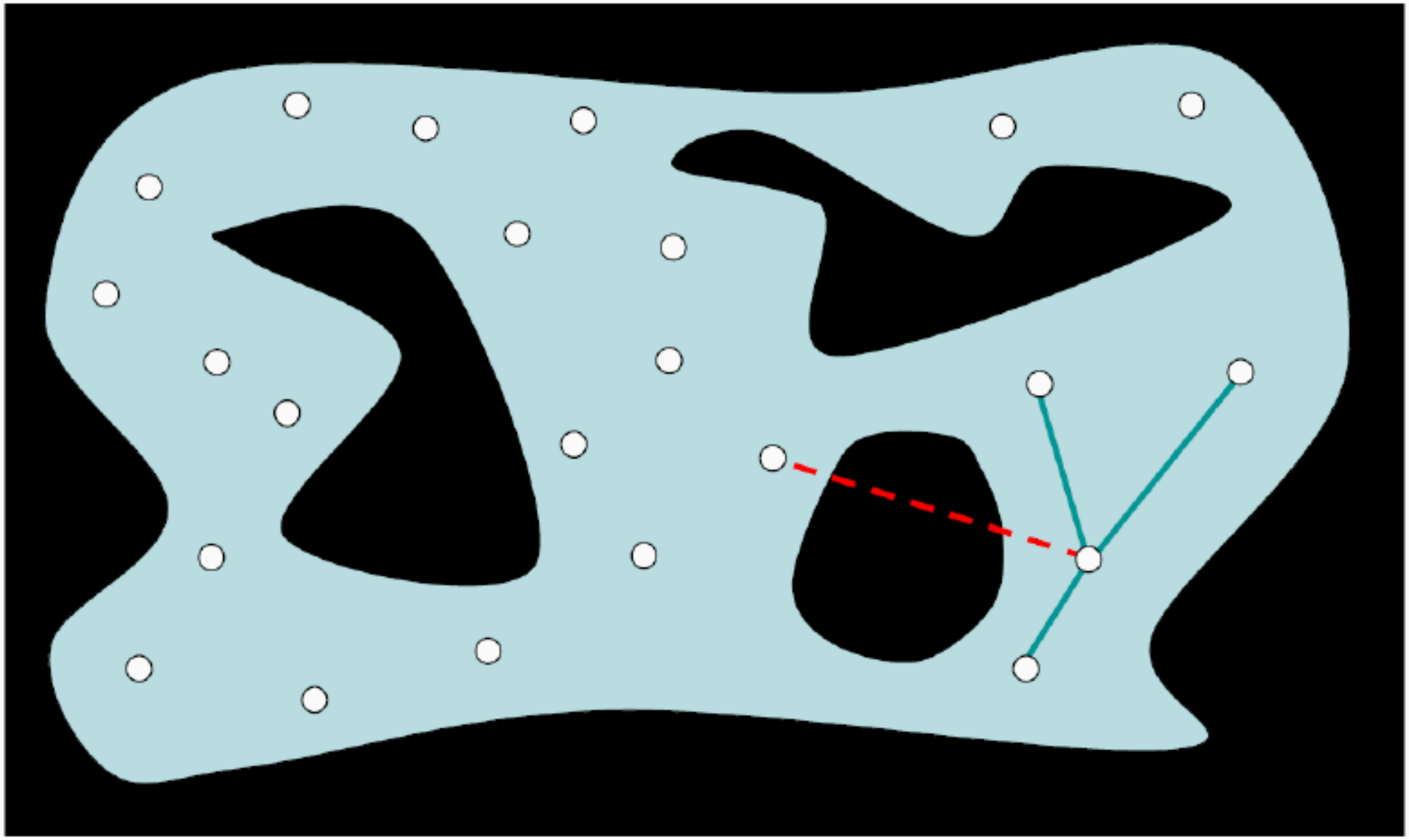
Sampling Techniques

Link each sample to its *K* nearest neighbors



Sampling Techniques

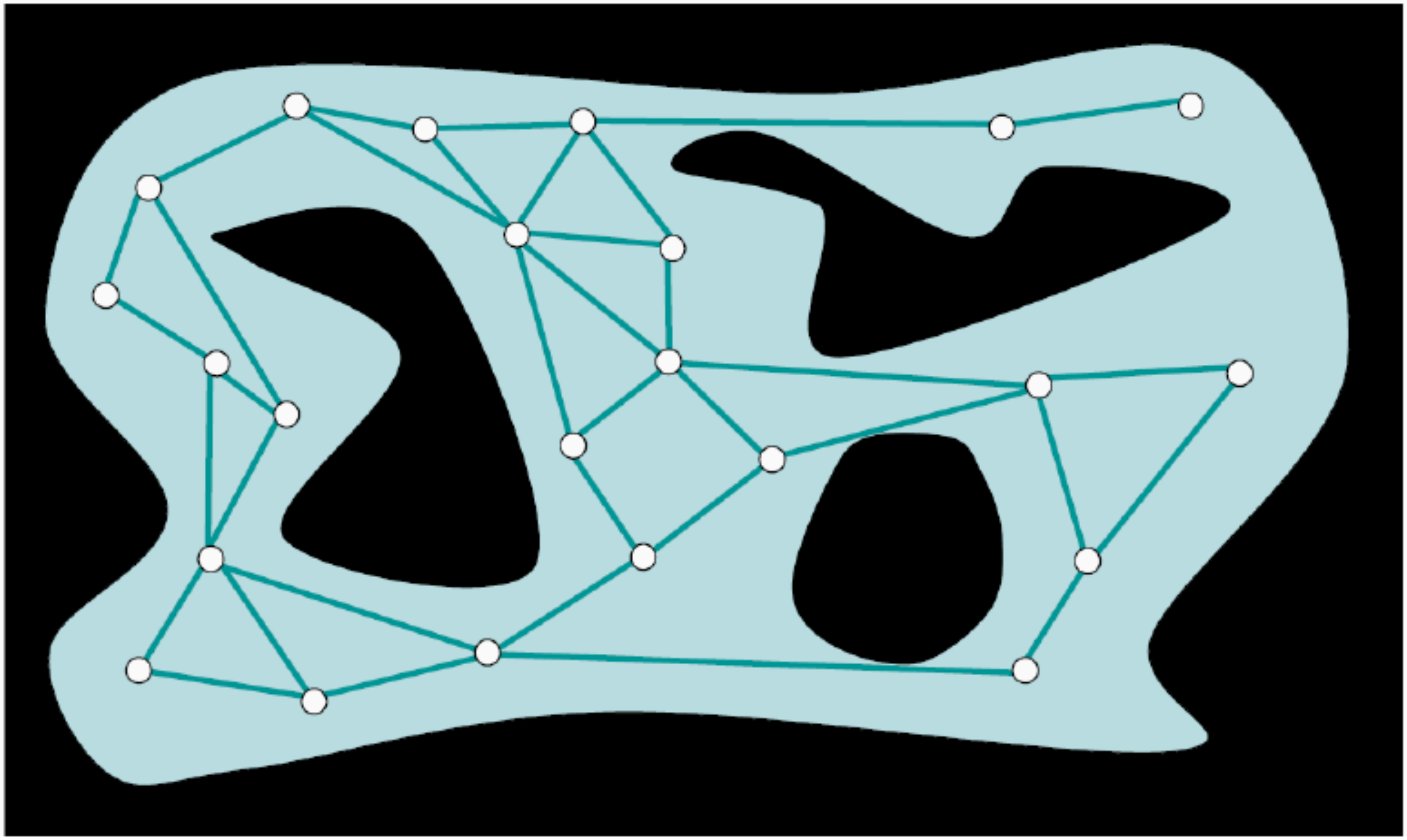
Remove the links that cross forbidden regions



Choset slides

Sampling Techniques

Remove the links that cross forbidden regions

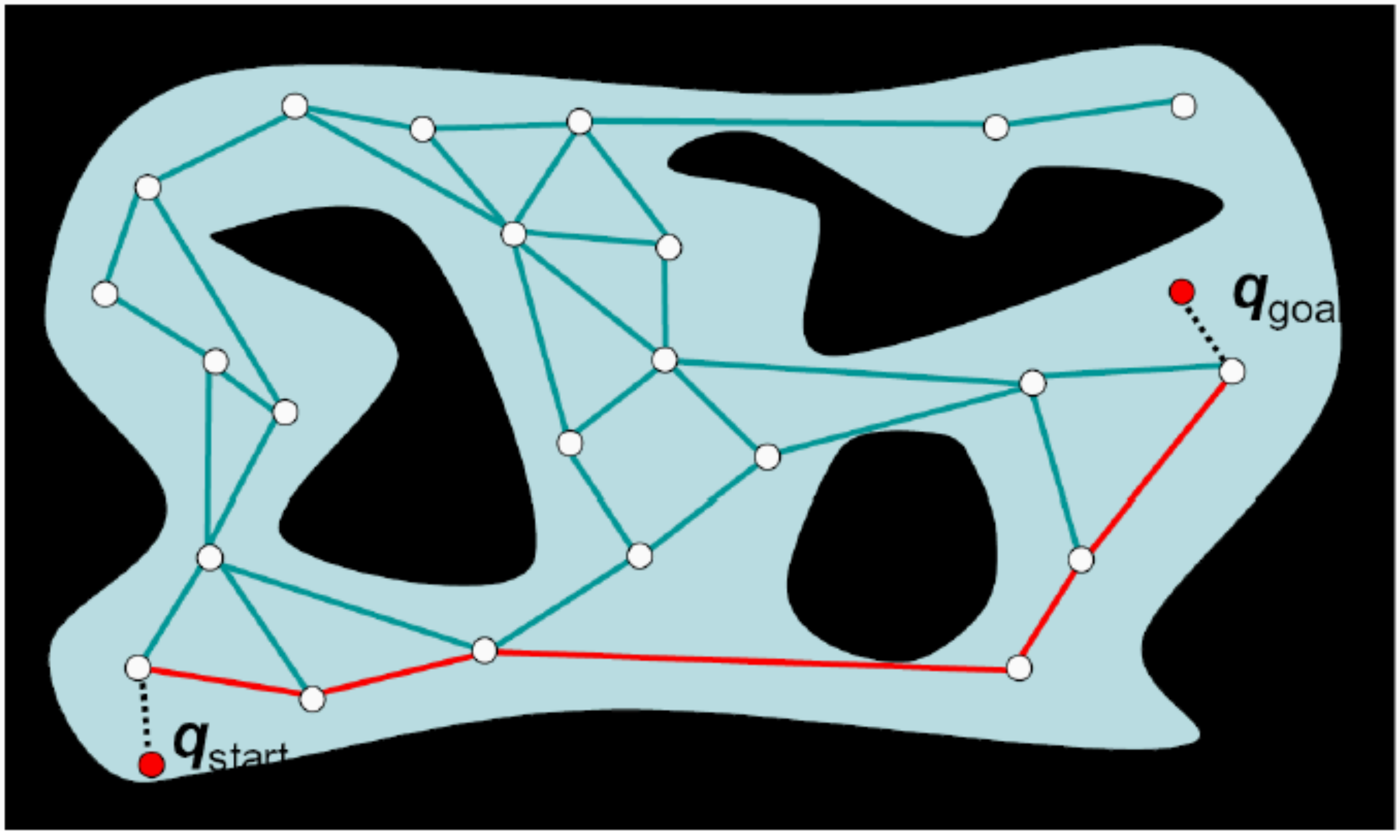


The resulting graph is a *probabilistic roadmap (PRM)*

Choset slides

Sampling Techniques

Link the start and goal to the PRM and search using A*



Choset slides

Sampling Techniques

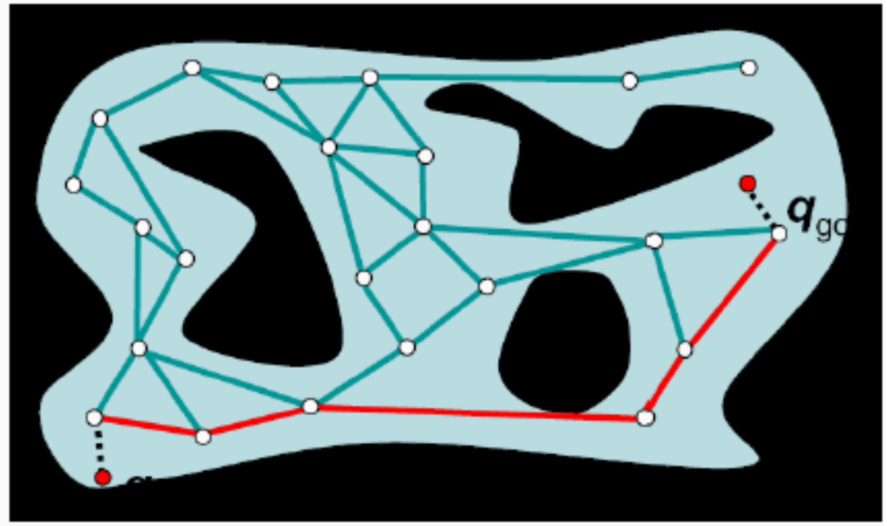
Continuous Space

↓

Discretization

↓

A* Search



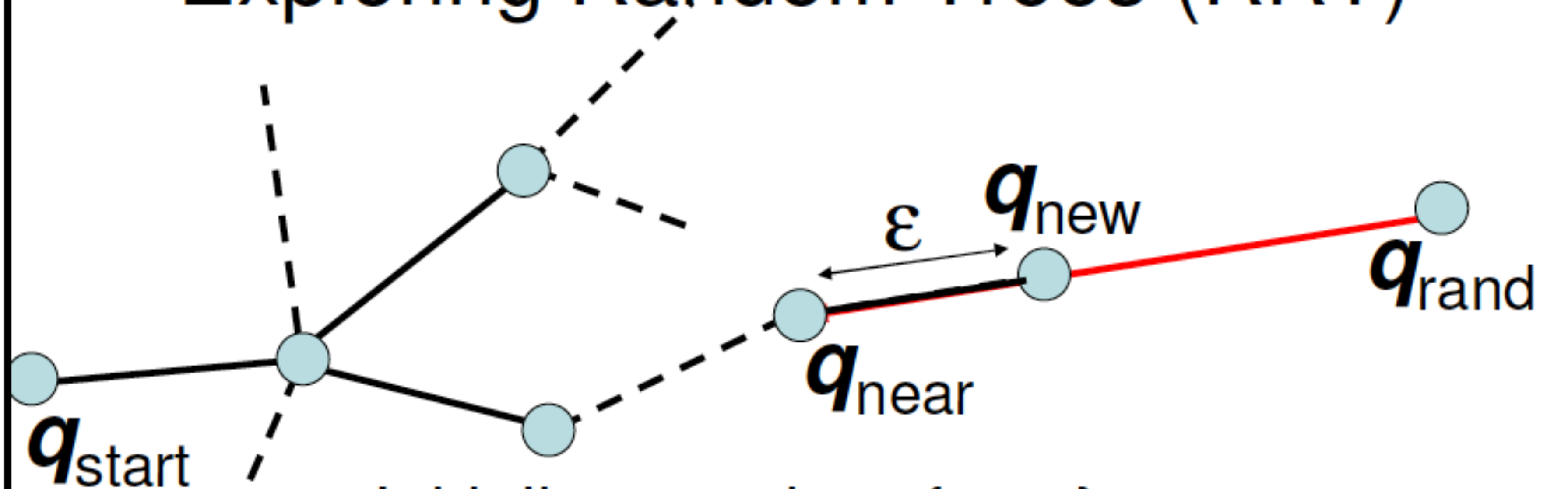
- “Good” sampling strategies are important:
  - Uniform sampling
  - Sample more near points with few neighbors
  - Sample more close to the obstacles
  - Use pre-computed sequence of samples

Choset slides

Sampling Techniques

- Remarkably, we can find a solution by using *relatively few randomly* sampled points.
- In most problems, a relatively small number of samples is sufficient to cover most of the feasible space with probability 1
- For a large class of problems:
  - Prob(finding a path) $\rightarrow$ 1 exponentially with the number of samples
- *But*, cannot detect that a path does not exist

Even More Radical: Rapidly Exploring Random Trees (RRT)




```
Algorithm BuildRRT
    Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
    Output: RRT graph G

    G.init(q_init)
    for k = 1 to K do
        q_rand ← RAND_CONF()
        q_near ← NEAREST_VERTEX(q_rand, G)
        q_new ← NEW_CONF(q_near, q_rand, Δq)
        G.add_vertex(q_new)
        G.add_edge(q_near, q_new)
    return G
```

- "←" denotes assignment. For instance, "*largest* ← *item*" means that the value of *largest* changes to the value of *item*.
- "**return**" terminates the algorithm and outputs the following value.

Choset slides

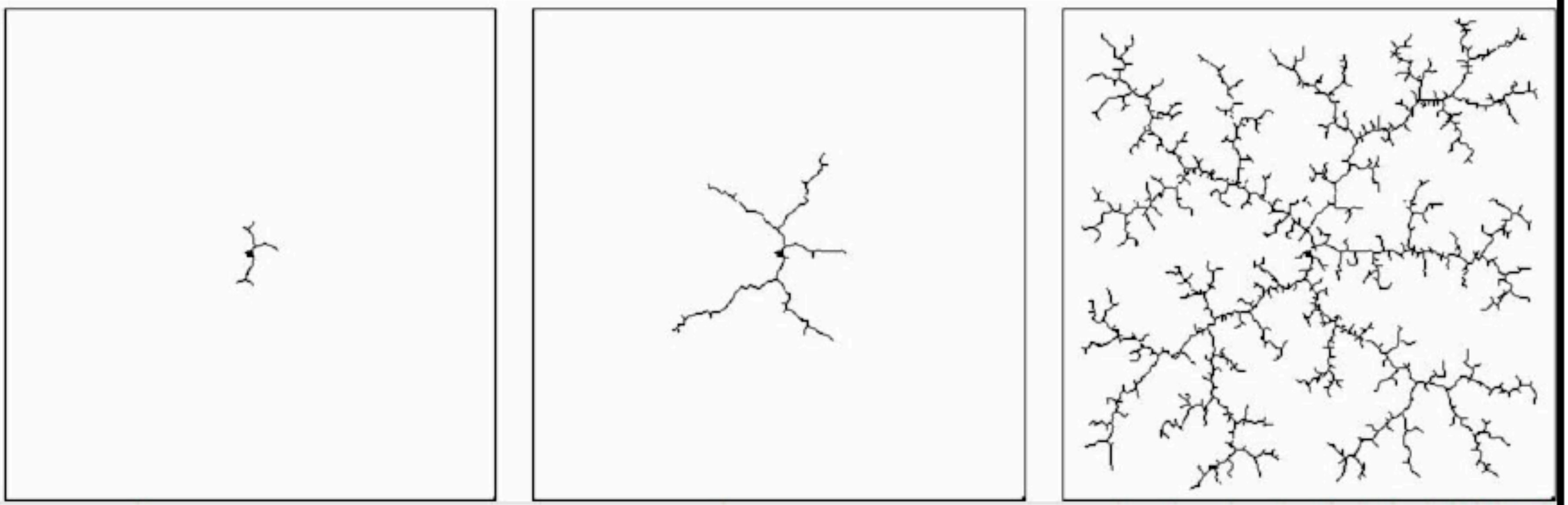
```
Algorithm BuildRRT
    Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
    Output: RRT graph G

    G.init(q_init)
    for k = 1 to K do
        q_rand ← RAND_CONF()
        q_near ← NEAREST_VERTEX(q_rand, G)
        q_new ← NEW_CONF(q_near, q_rand, Δq)
        G.add_vertex(q_new)
        G.add_edge(q_near, q_new)
    return G
```

- "←" denotes assignment. For instance, "*largest* ← *item*" means that the value of *largest* changes to the value of *item*.
- "**return**" terminates the algorithm and outputs the following value.

- The sample qrand is drawn UAR from configuration space
  - or reject if inside obstacle
- Notice
  - node with big voronoi region of free space more likely to get expanded
    - the nearest neighbor step
  - so tree builds out into free space quickly
  - in different applications, one uses different epsilon
    - sometimes even add whole edge

Choset slides

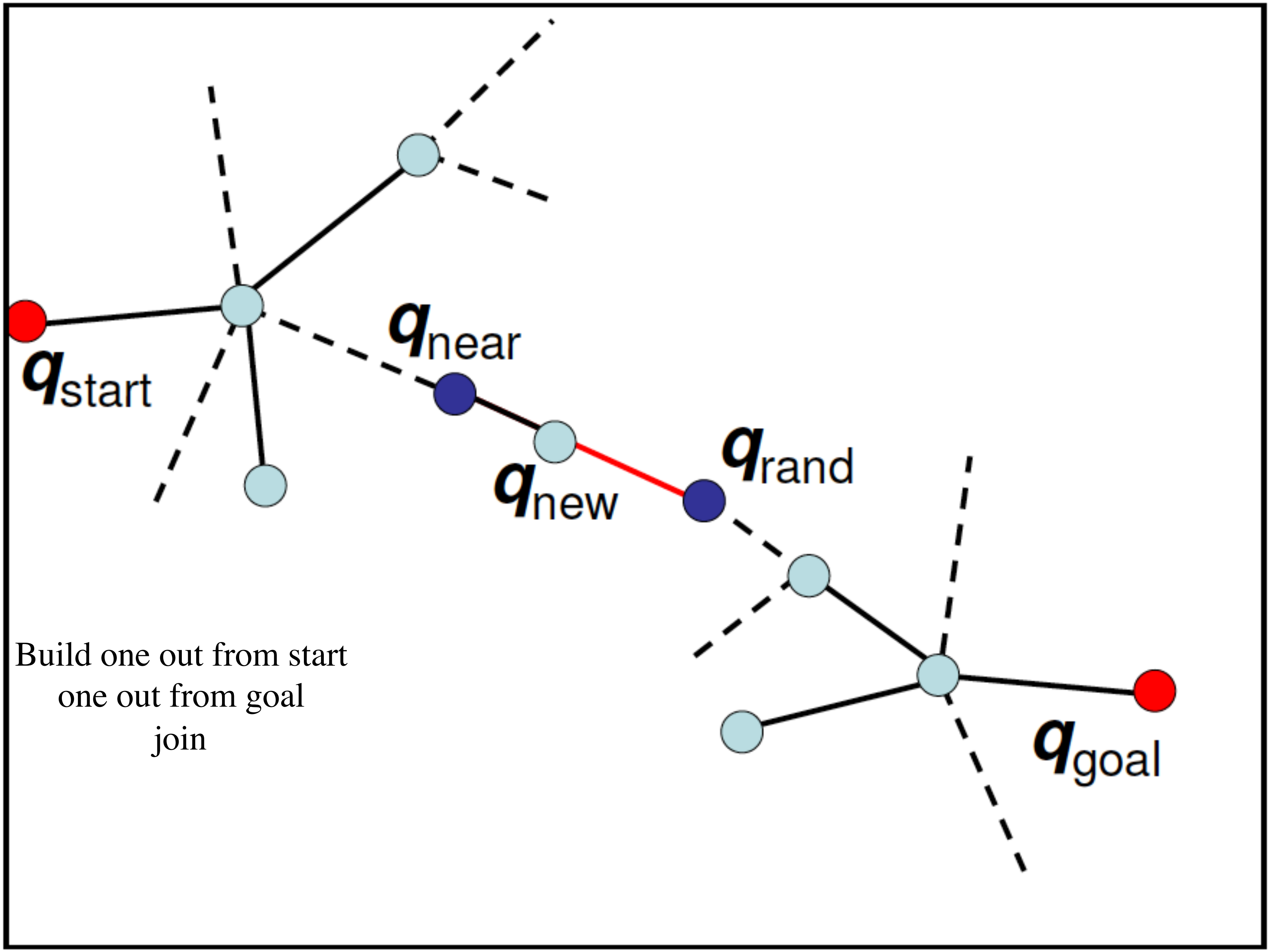
Properties



- Tends to explore the space rapidly in all directions
- Does not require extensive pre-processing
- Single query/multiple query problems
- Needs only collision detection test → No need to represent/pre-compute the entire C-space

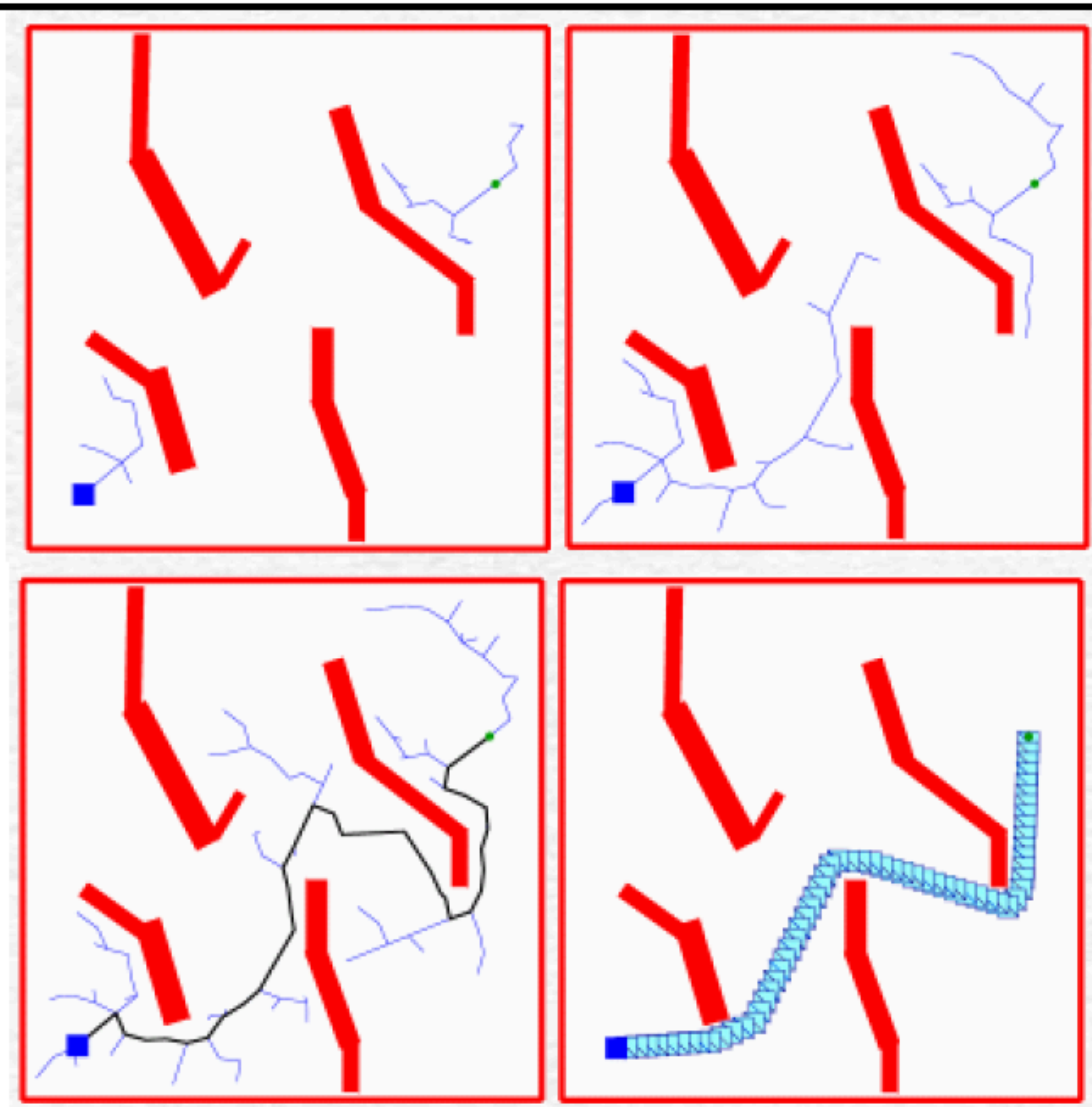
You have to be able to draw the samples - this can get tricky

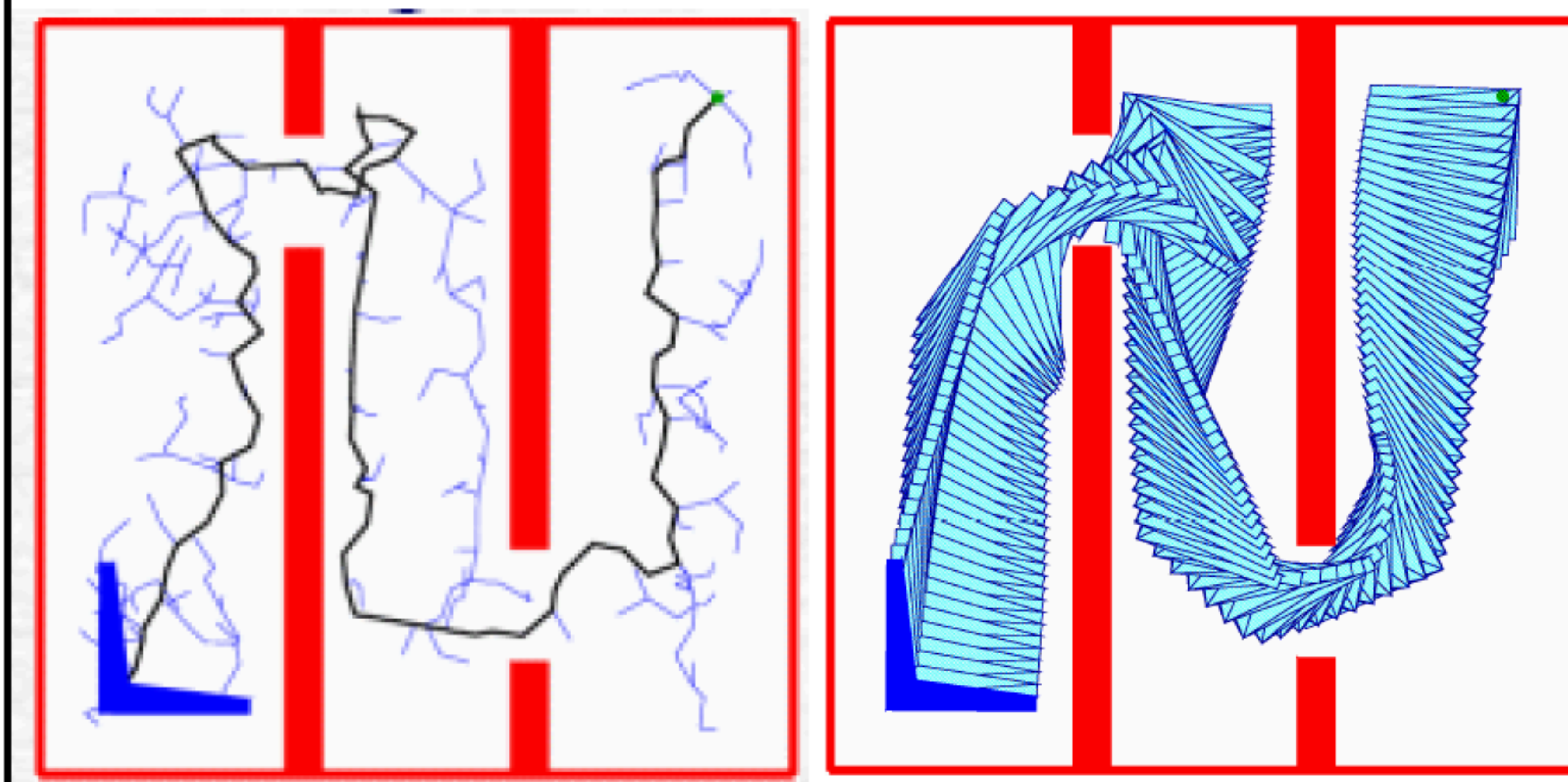
Choset slides



Build one out from start
one out from goal
join

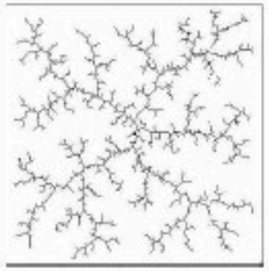
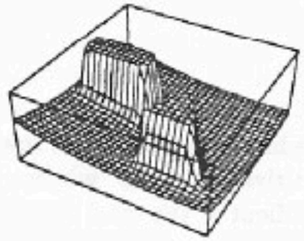
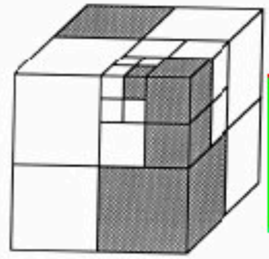
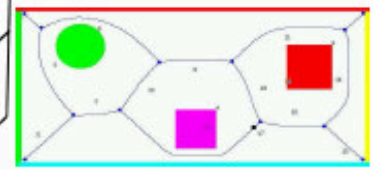
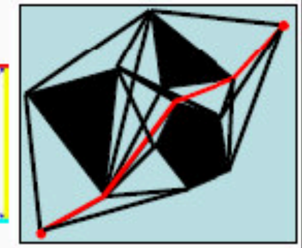
Choset slides


Choset slides



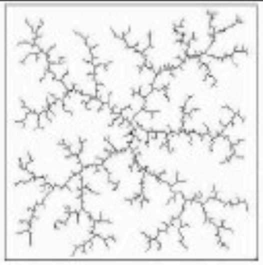
From Kuffner et al.

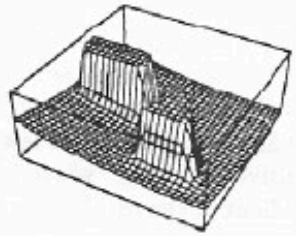
Choset slides

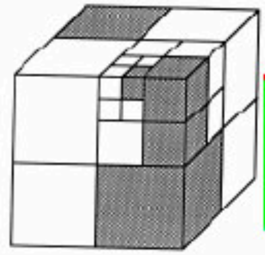






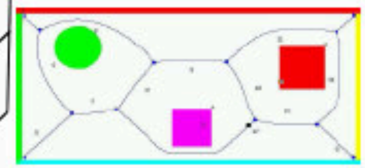
	Sampling	Potential Fields	Approx. Cell Decomposition	Voronoi	Visibility
Practical in ~2-D or 3-D	Y	Y	Y	Y	Y
Practical in >> 2-D or 3-D	Y	Y (using randomized version)	??	N	N
Fast	Y	Y	Y	In low dim.	In 2-D
Online Extensions	Y	Y	??	??	N
Complete?	Probabilistically complete	Probabilistically-resolution complete	Resolution-Complete	Y	Y

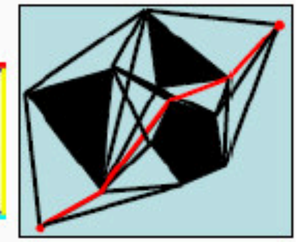
Choset slides











	Sampling	Potential Fields	Approx. Cell Decomposition	Voronoi	Visibility
Practical in ~2-D or 3-D	Y	Y	Y	Y	Y
Practical in >> 2-D or 3-D	Y	Y (using randomized version)	??	N	N
Fast	Y	Y	Y	In low dim.	In 2-D
Online Extensions					N
Complete?	Probabilistically complete	Probabilistically-resolution complete	Resolution-Complete	Y	Y

More exact/Complete

Faster/More practical in high dim.

Choset slides

- (Limited) background in Russell&Norvig Chapter 25
- Two main books:
  - J-C. Latombe. Robot Motion Planning. Kluwer. 1991.
  - S. Lavalle. Planning Algorithms. 2006. http://msl.cs.uiuc.edu/planning/
  - H. Choset et al., Principles of Robot Motion: Theory, Algorithms, and Implementations. 2006.
- Other demos/examples:
  - http://voronoi.sbp.ri.cmu.edu/~choset/
  - http://www.kuffner.org/james/research.html
  - http://msl.cs.uiuc.edu/rrt/