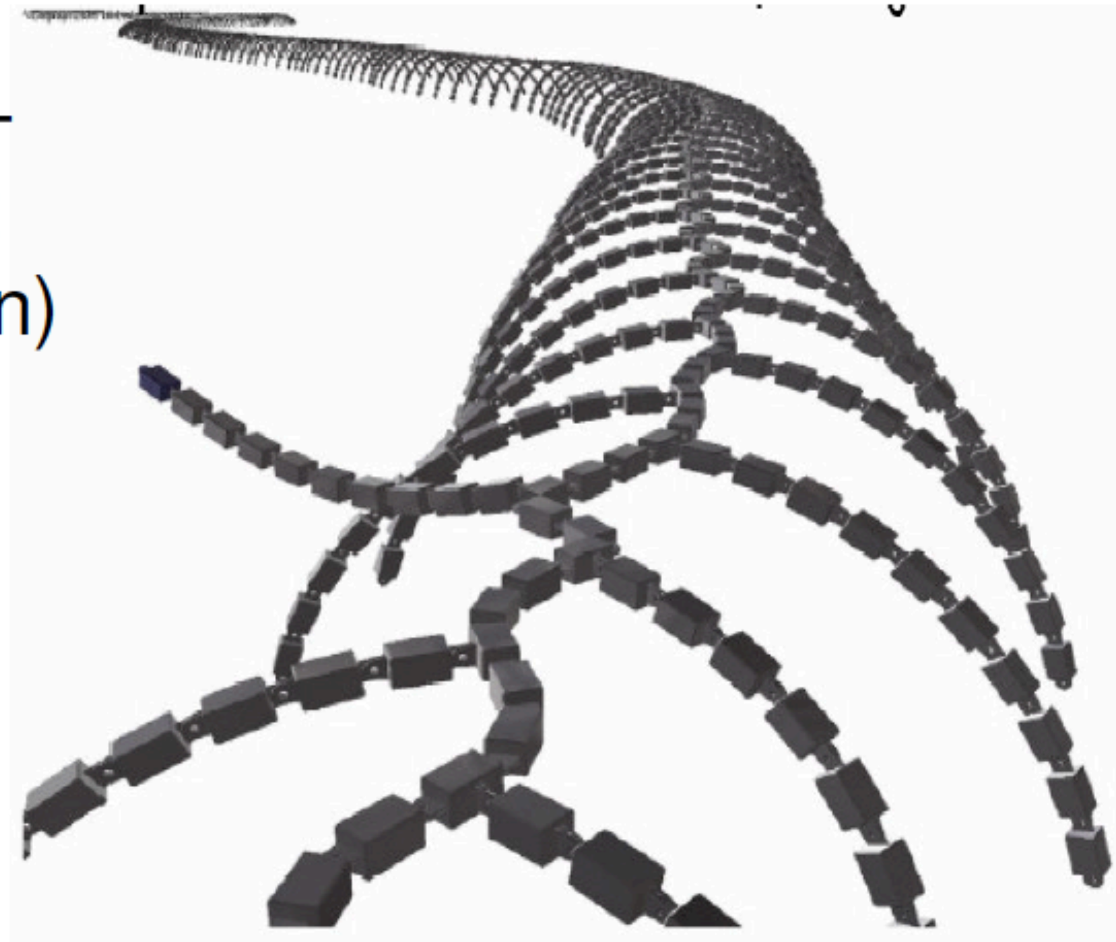
Motion Planning II

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

Large C-Space Dimension

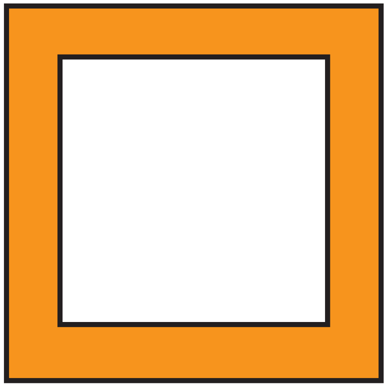
Millipede-like robot (S. Redon)



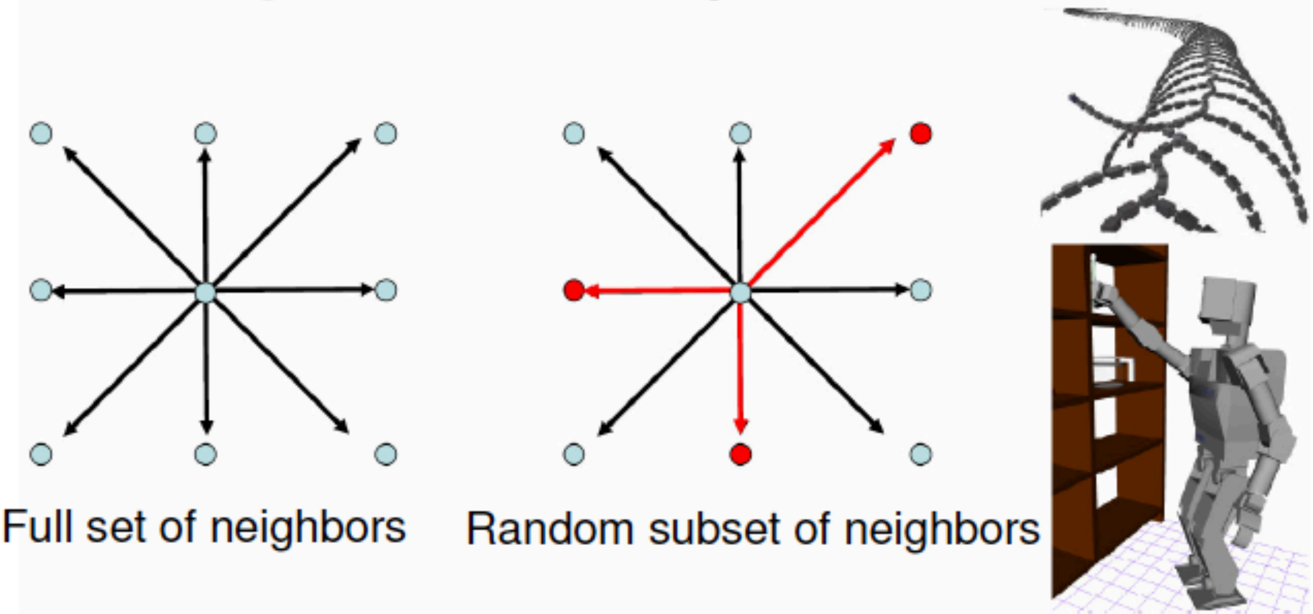
~13,000 DOFs !!!

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

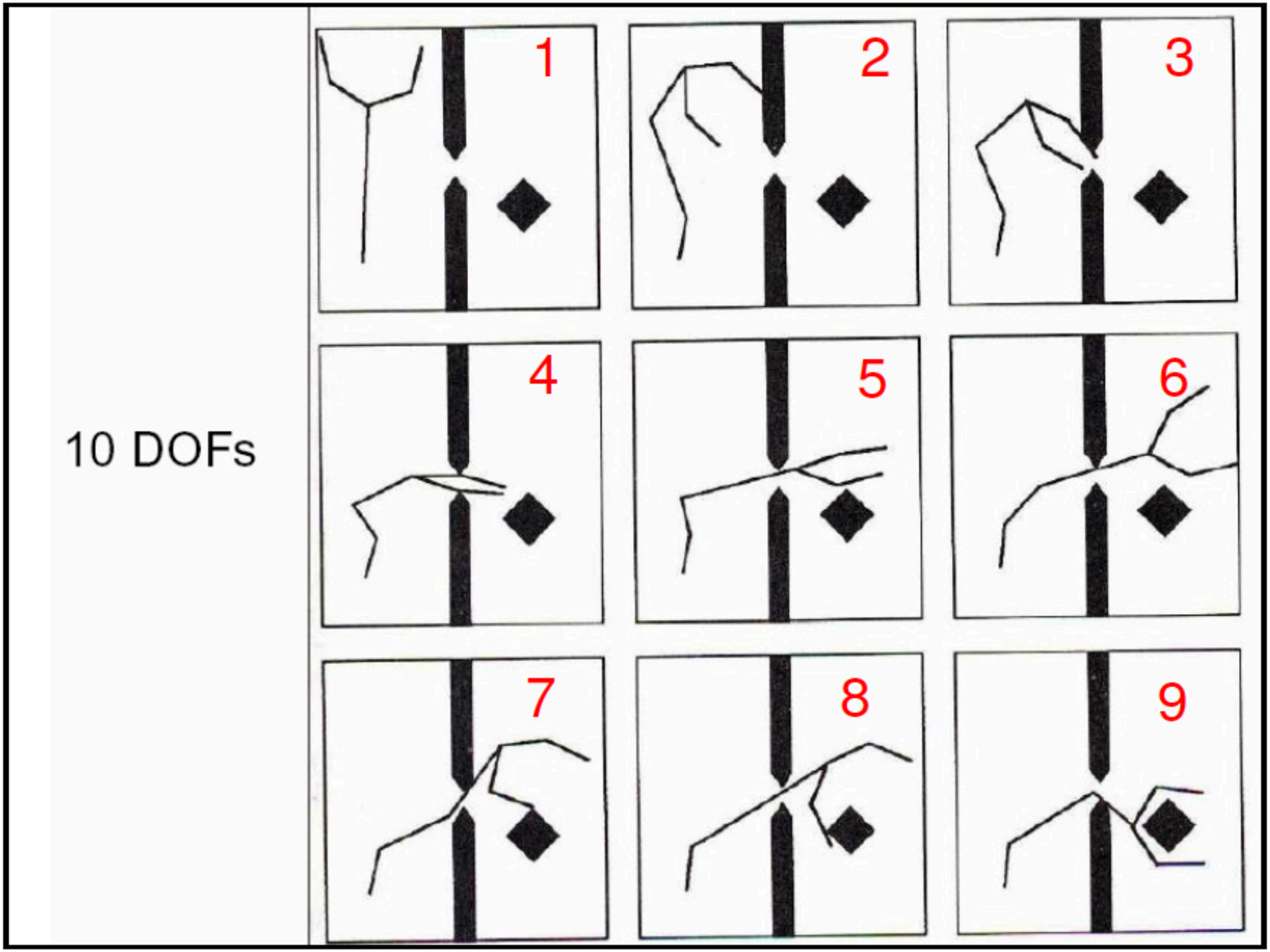


- We should evaluate all the neighbors of the current state, but:
- Size of neighborhood grows exponentially with dimension
- Very expensive in high dimension

Solution:

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

Choset slides



Choset slides

Why do we care?

- Our configuration space may be inconveniently large
  - even 3D is much harder than 2D
- We need the ideas to talk about dynamics
  - planning with dynamics is very different from kinematic planning

Dynamics make planning harder

- Dynamics introduce differential constraints

$$\dot{x} = f(x, u)$$

Derivative of state

State

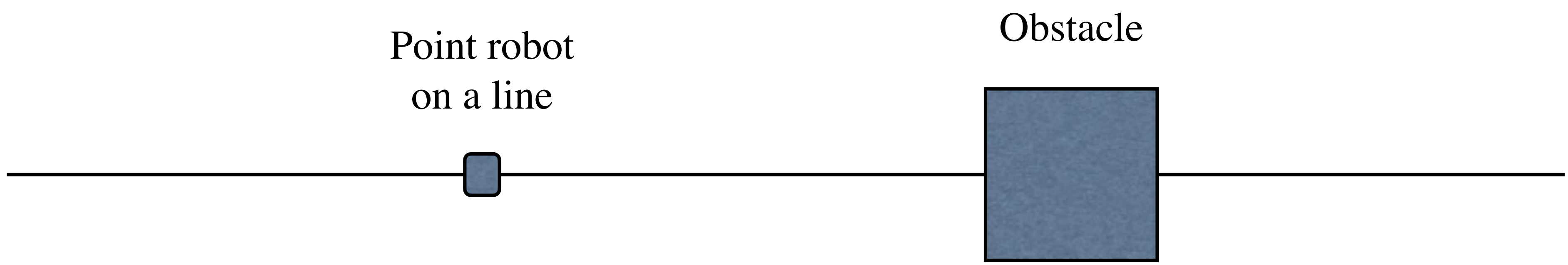
Control input - there might be constraints on this, too

Quite possibly nasty

Phase space

- Configuration space + all relevant derivatives
- For us, very likely:
  - position+orientation+velocity+ang.velocity

Simple example



- Configuration space: x
  - (with complications created by obstacle)
- State is:
  - (x, v)
  - so phase space is 2D
- Dynamics are:

$$\begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v \\ u \end{pmatrix}$$

Simplest case - extend obstacles

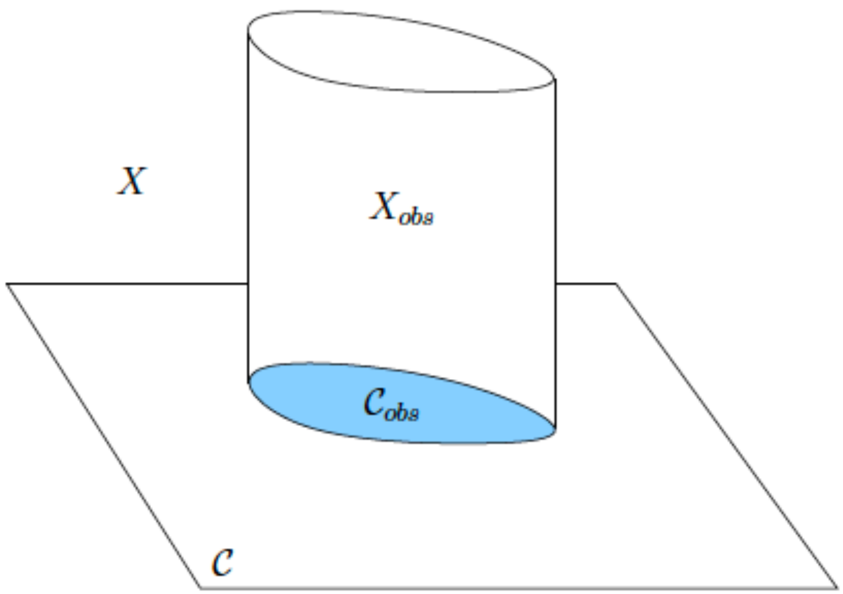
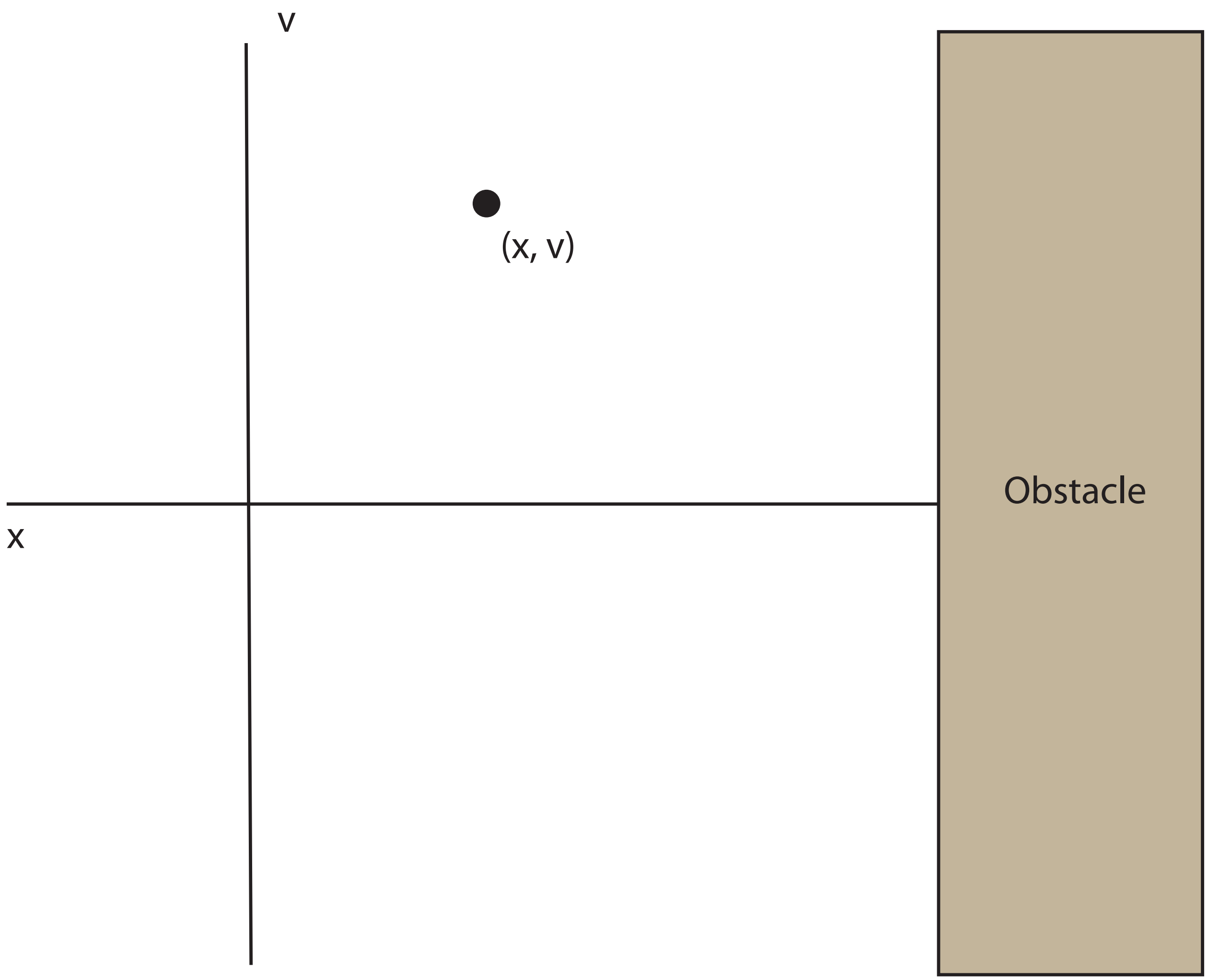


Figure 14.1: An obstacle region $\mathcal{C}_{obs} \subset \mathcal{C}$ generates a cylindrical obstacle region $X_{obs} \subset X$ with respect to the phase variables.

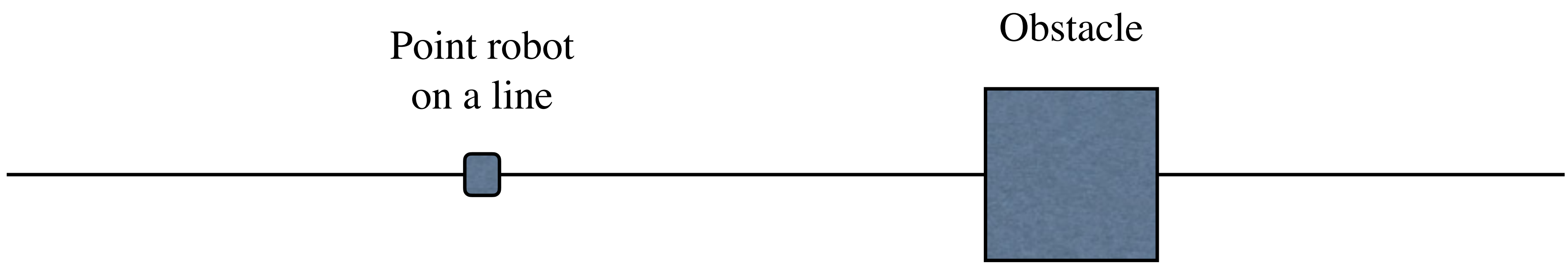
= derivatives

Lavalle, Ch 14

The phase space…

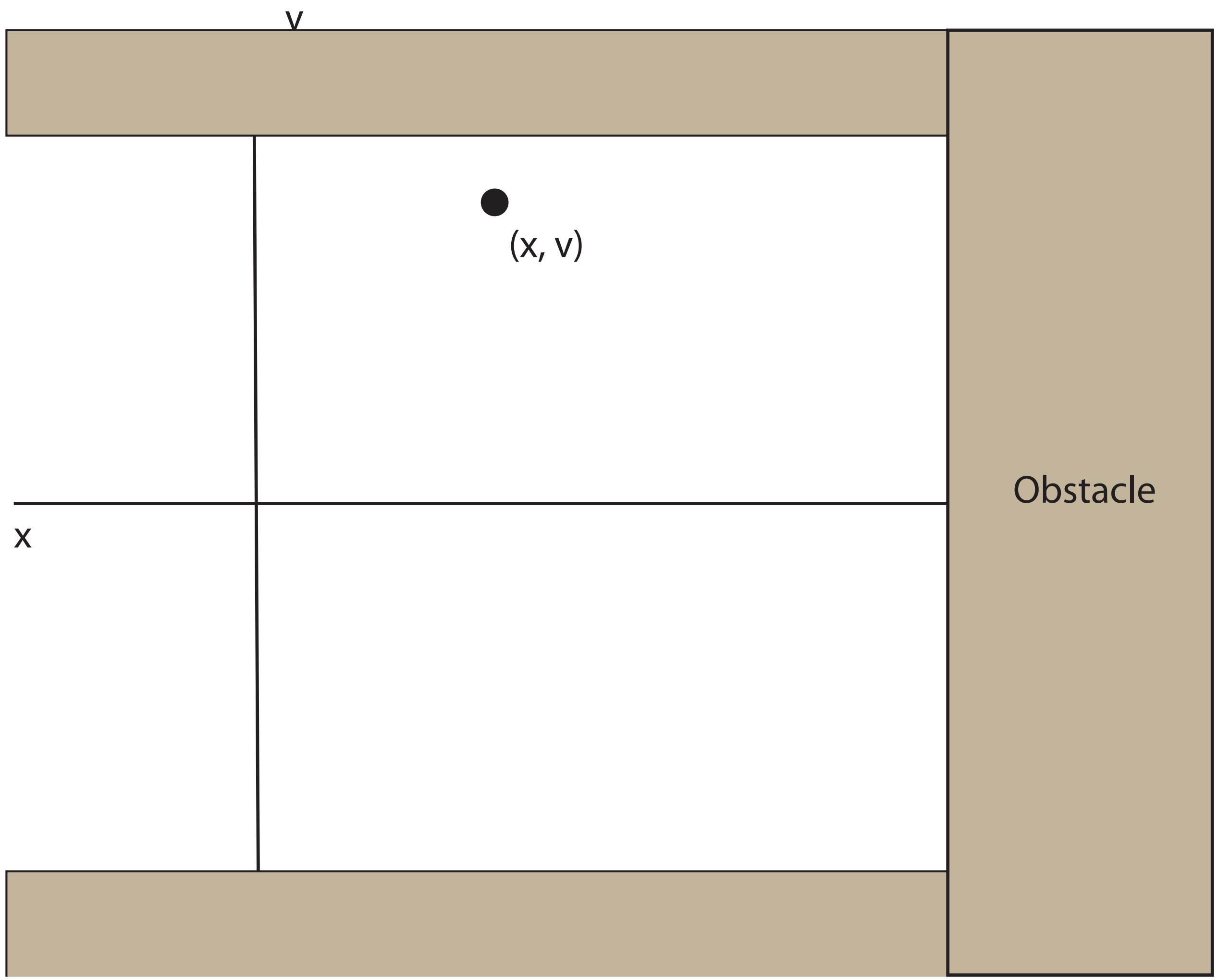


Limits on phase variables



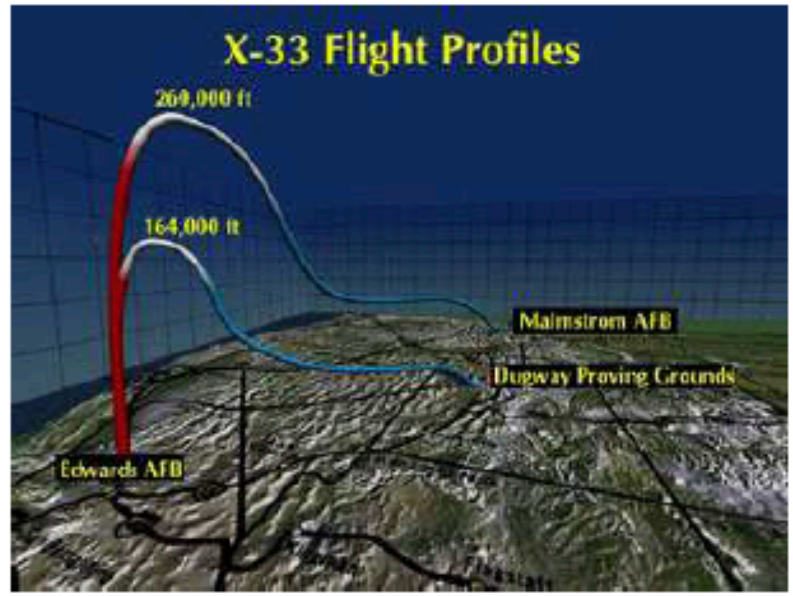
- Configuration space: x
  - (with complications created by obstacle)
- State is:
  - (x, v)
  - so phase space is 2D
- Velocity limits are: $-1<v<1$
  - draw phase space with obstacles

Phase space is now



Constraints on phase variables

NASA/Lockheed Martin X-33

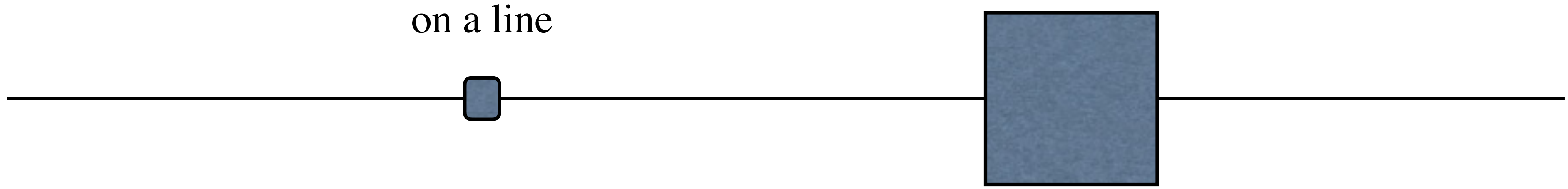


Re-entry trajectory

Figure 14.2: In the NASA/Lockheed Martin X-33 re-entry problem, there are complicated constraints on the phase variables, which avoid states that cause the craft to overheat or vibrate uncontrollably. (Courtesy of NASA)

Lavalle, Ch 14

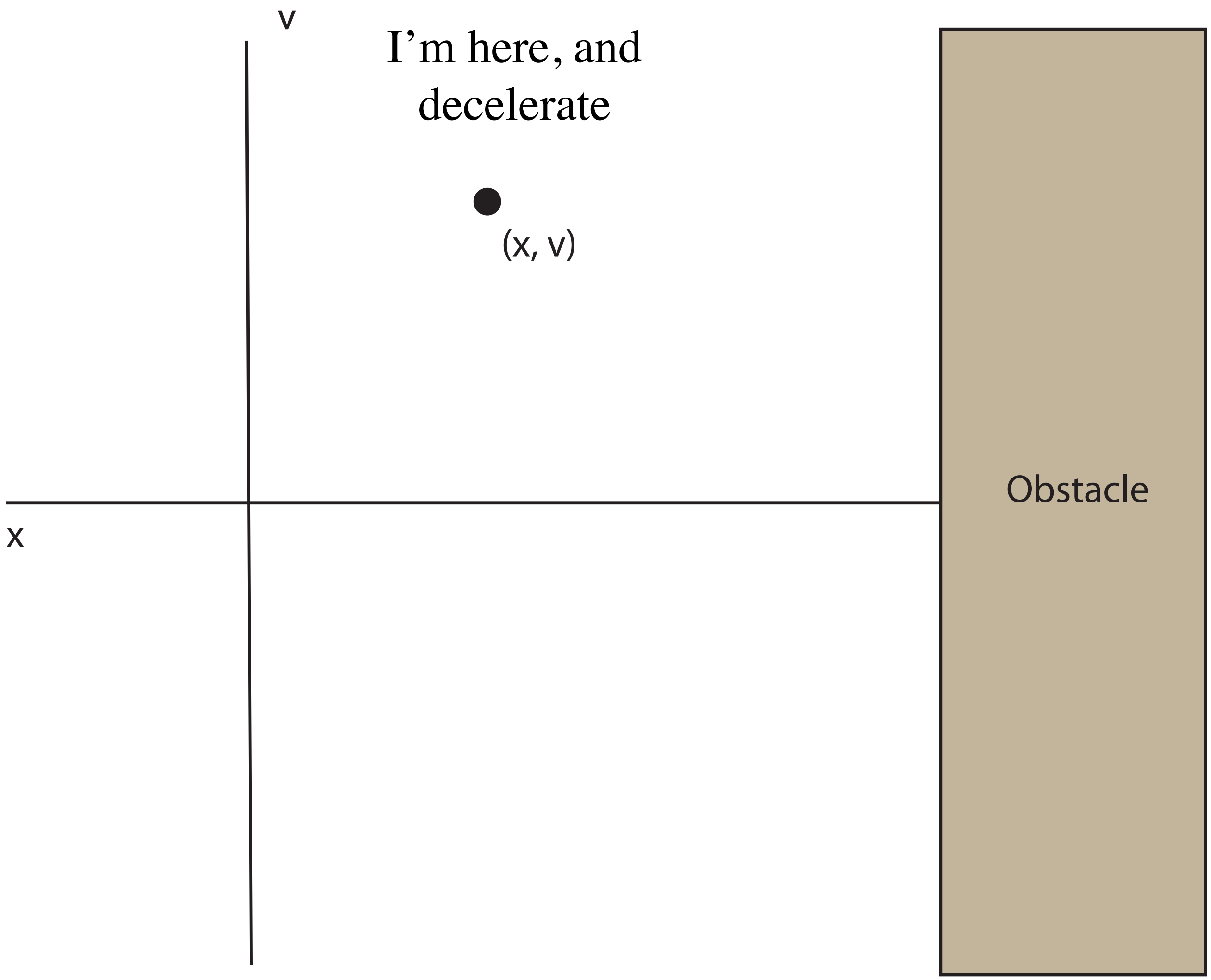
Control limits create nasty obstacles



- State is: (x, v)
- differential constraints:

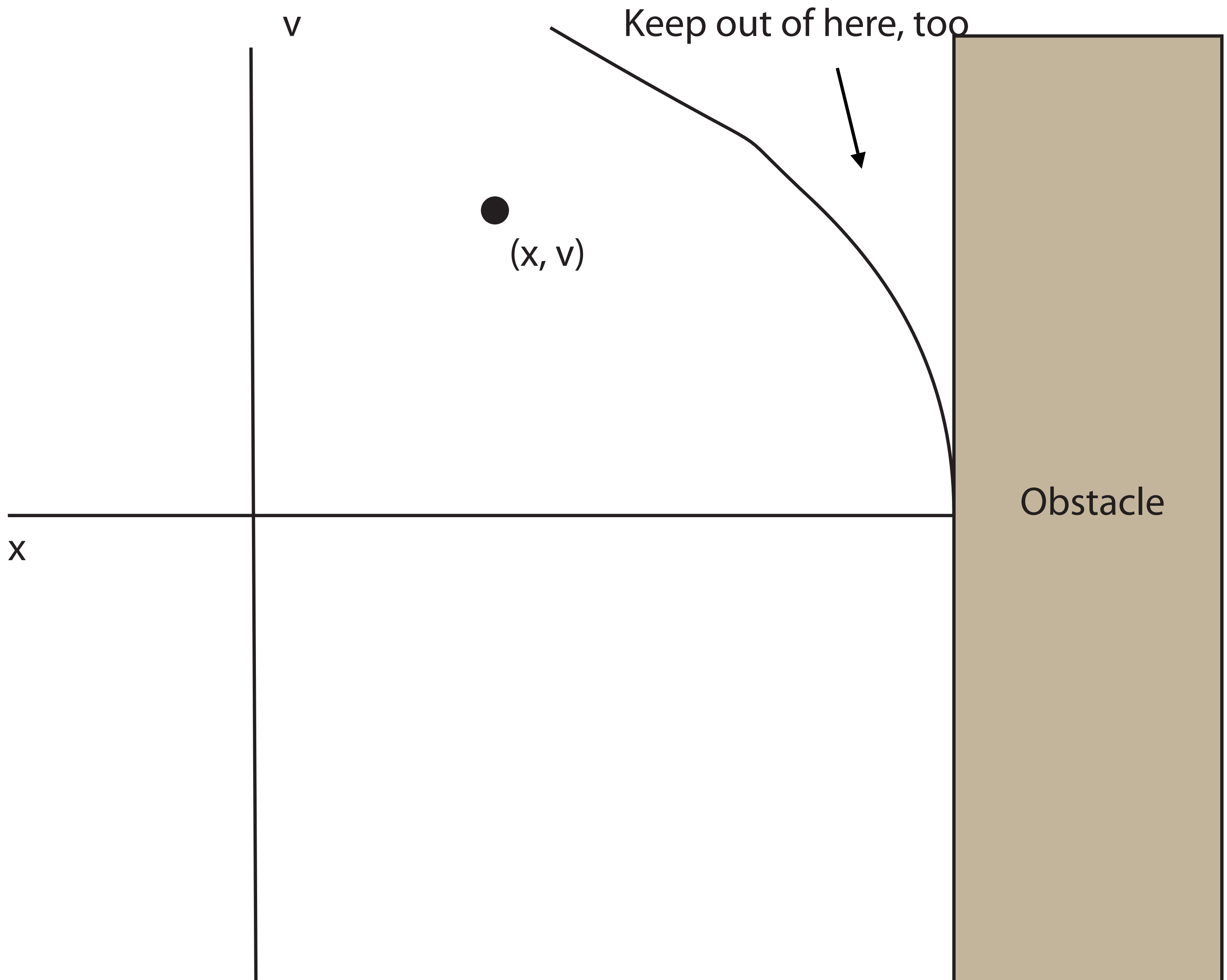
$$\begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v \\ u \end{pmatrix}$$

Control constraint: $-1 \leq u \leq 1$

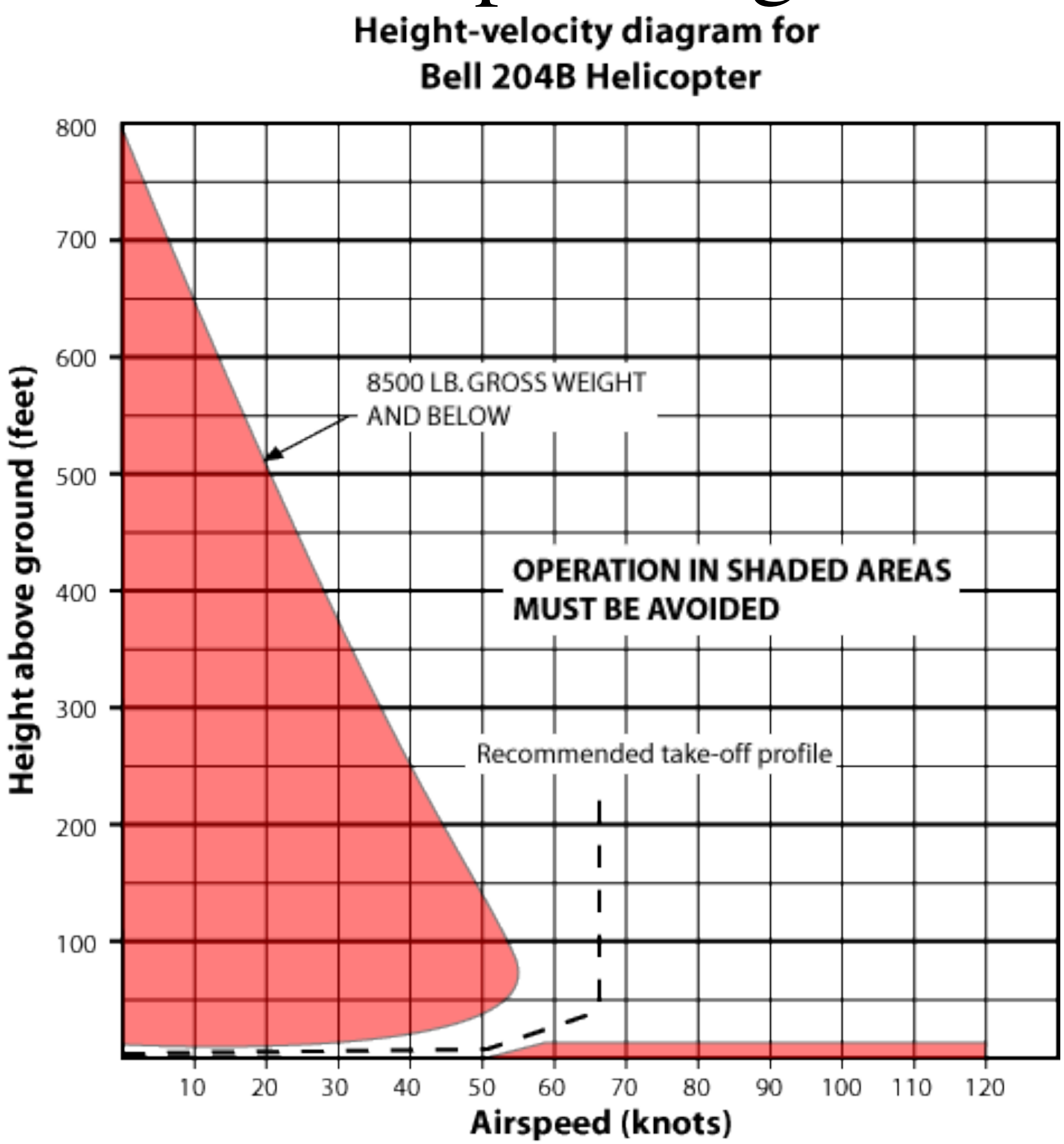


Distance travelled
until stationary
at acceleration = -1

$$\Delta x = \frac{v^2}{2}$$


v
Keep out of here, too
(x, v)
Obstacle
x

Helicopter height-velocity diagram



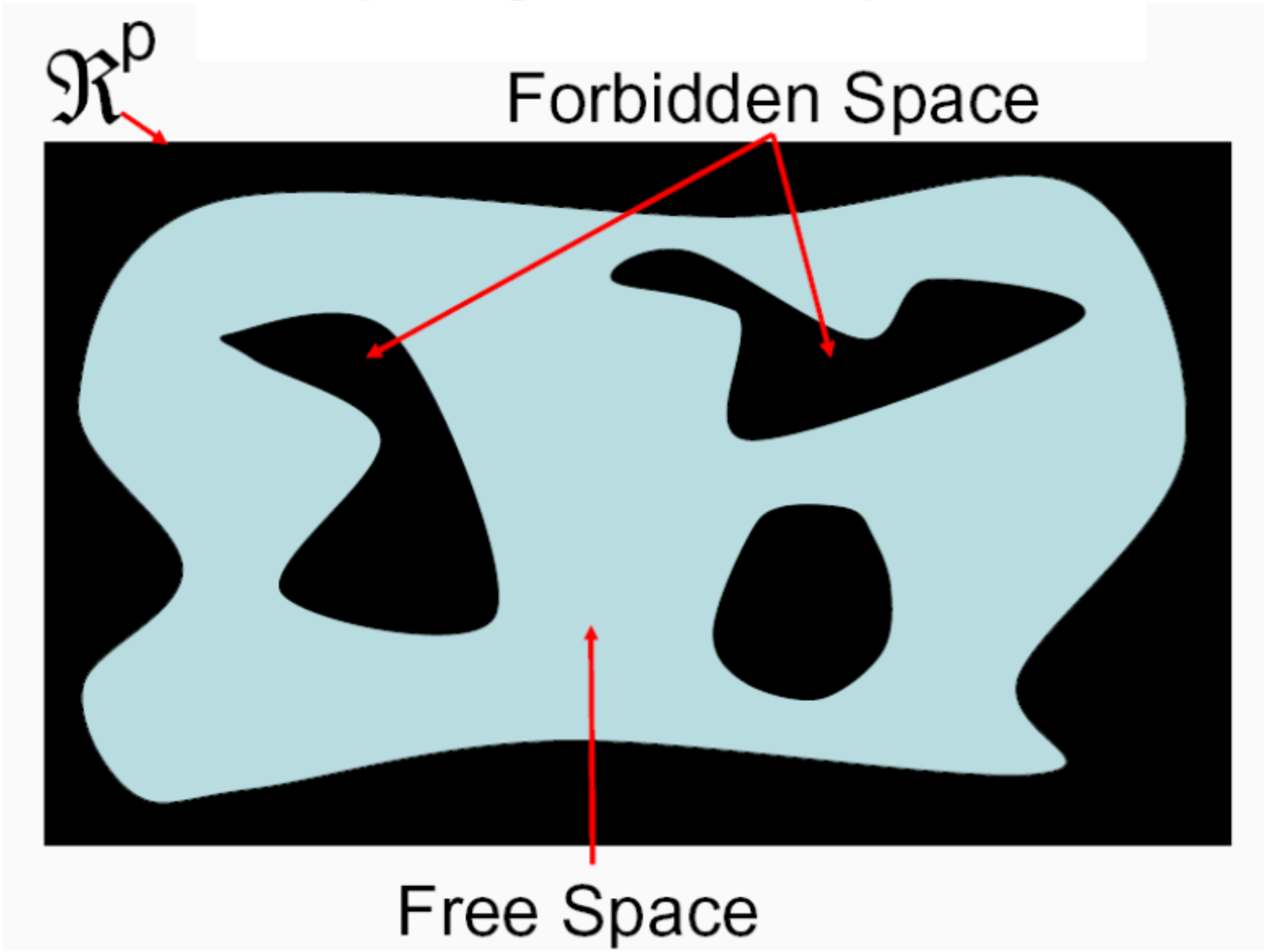
(colloquially, dead man's curve; from wikipedia; there are all sorts of operating limits to helicopters)

Random roadmaps

- Problems:
  - We can’t construct
    - visibility complexes
    - voronoi diagrams
    - grids
  - cause the space is “too big” (too many neighbors/faces/etc)
  - Potential functions may have nasty behaviors, too

- Idea:
  - draw random samples in configuration space, and join up
  - we might get a road map like this, and samples are relatively easy to draw

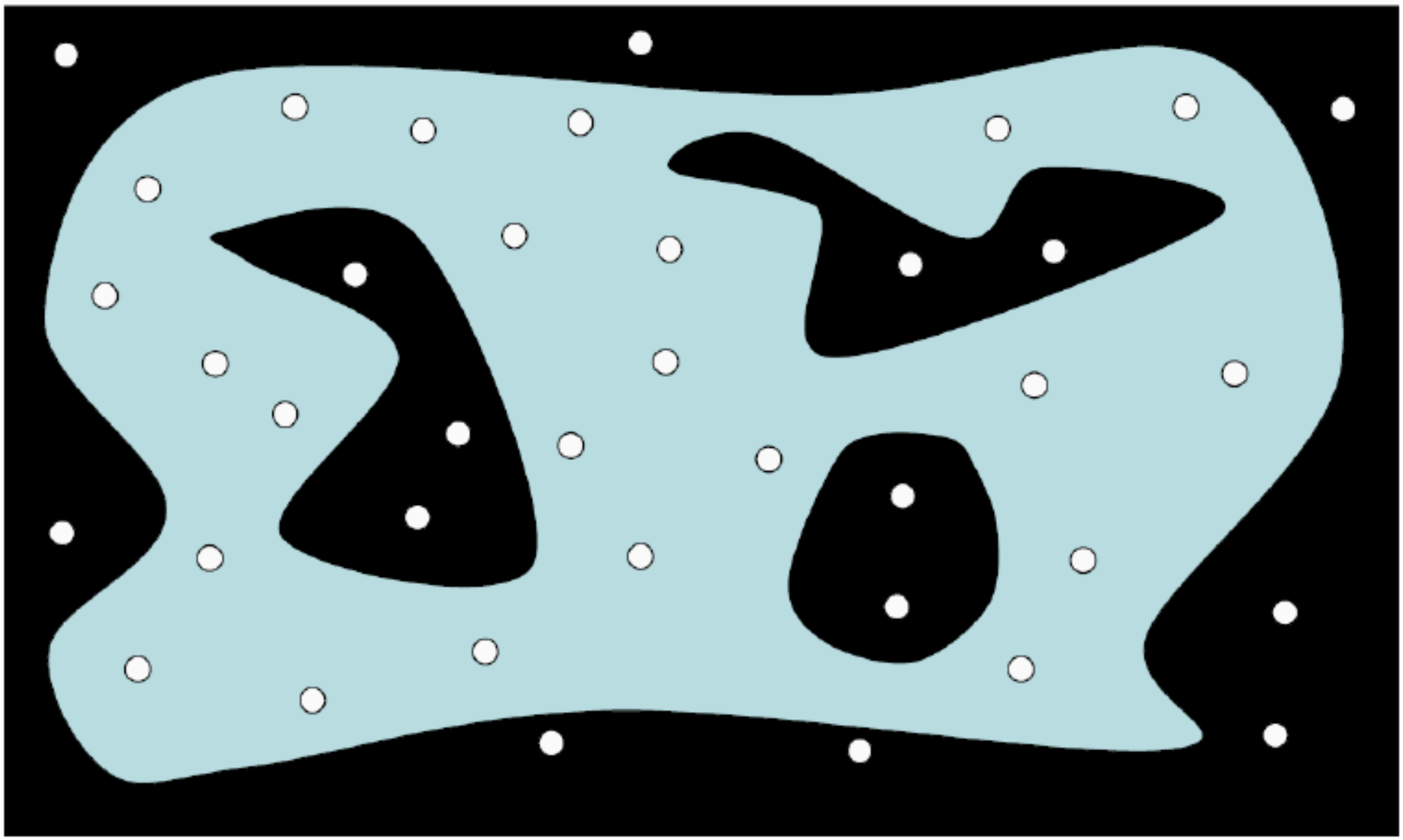
Sampling Techniques



Choset slides

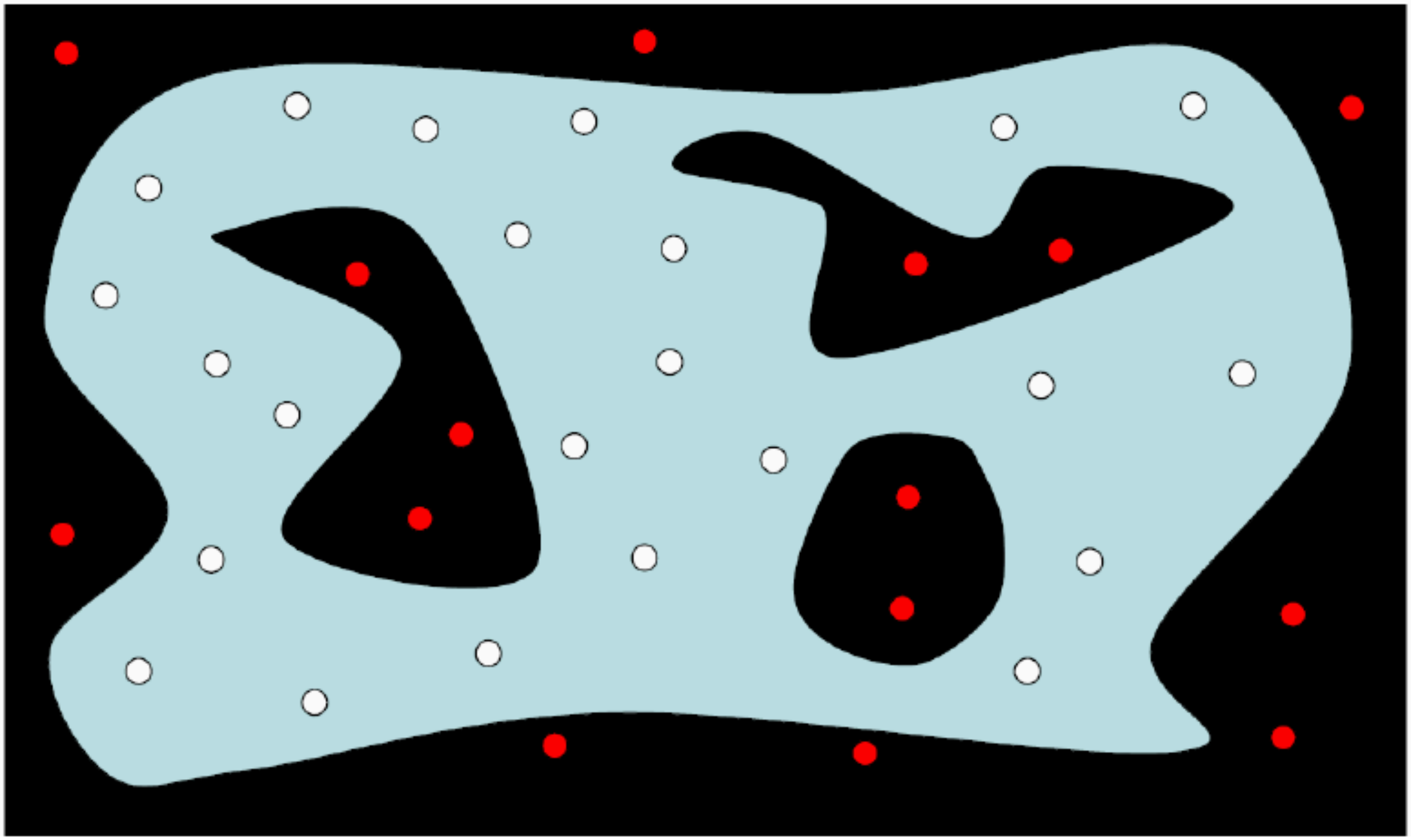
Sampling Techniques

Sample random locations



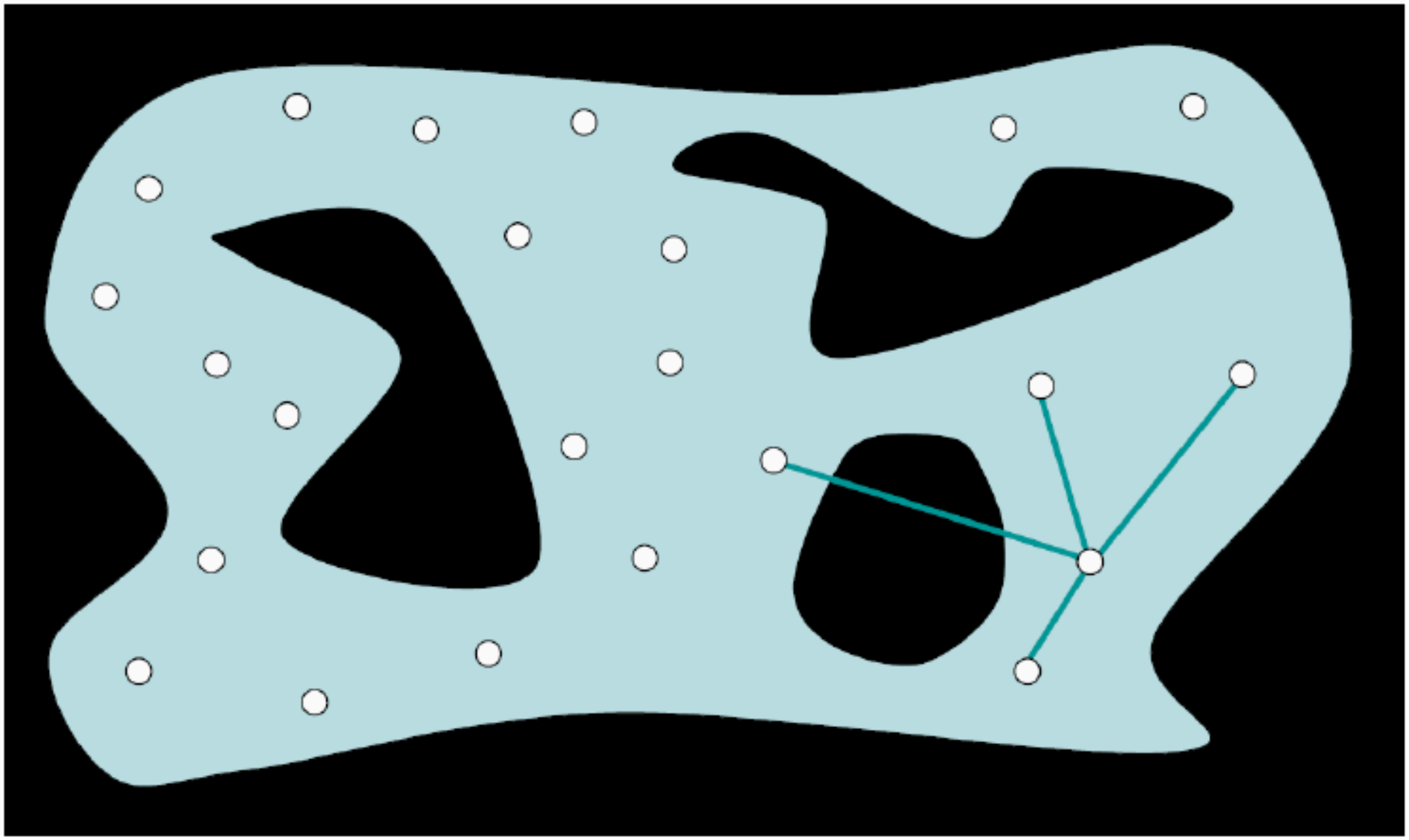
Sampling Techniques

Remove the samples in the forbidden regions



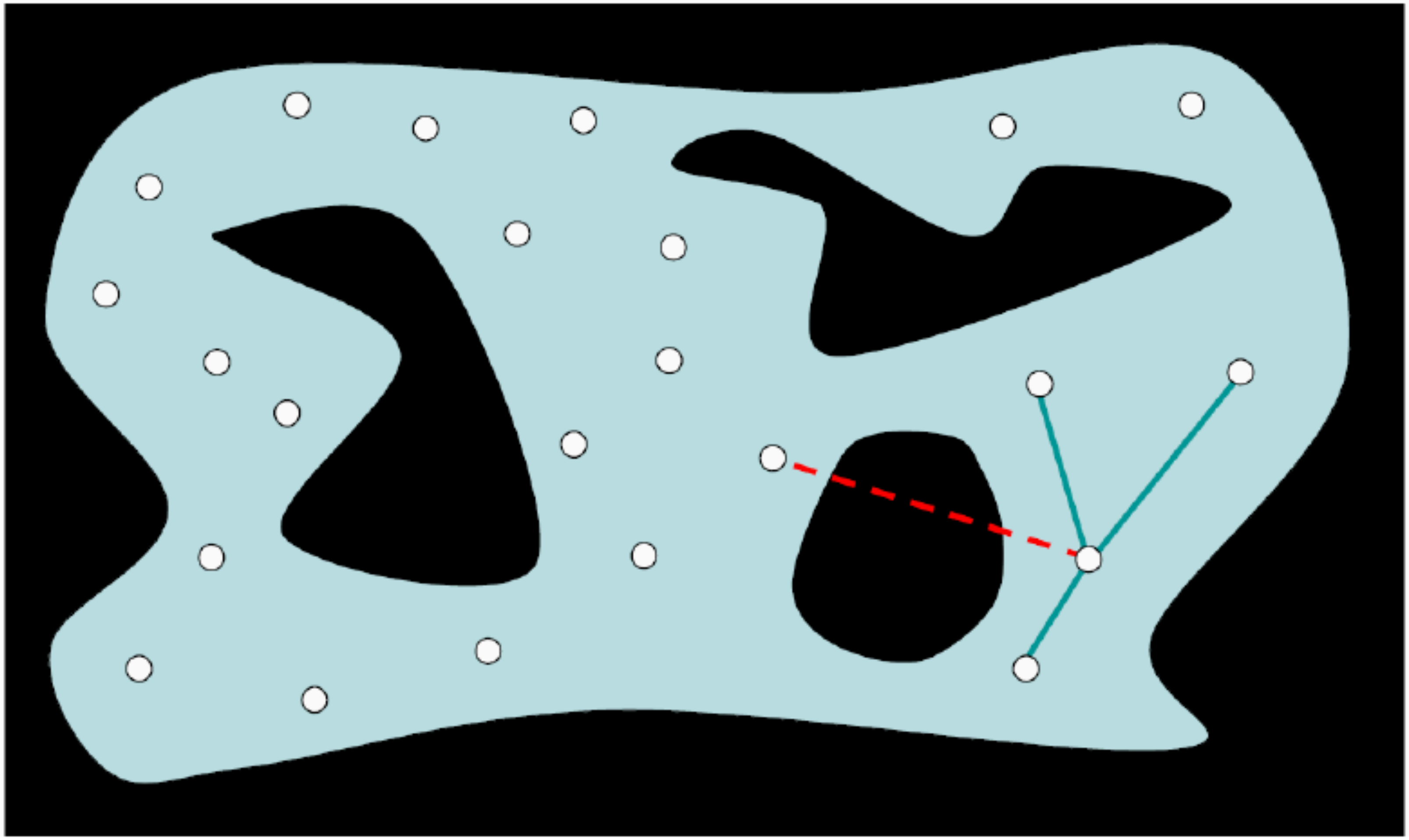
Sampling Techniques

Link each sample to its *K* nearest neighbors



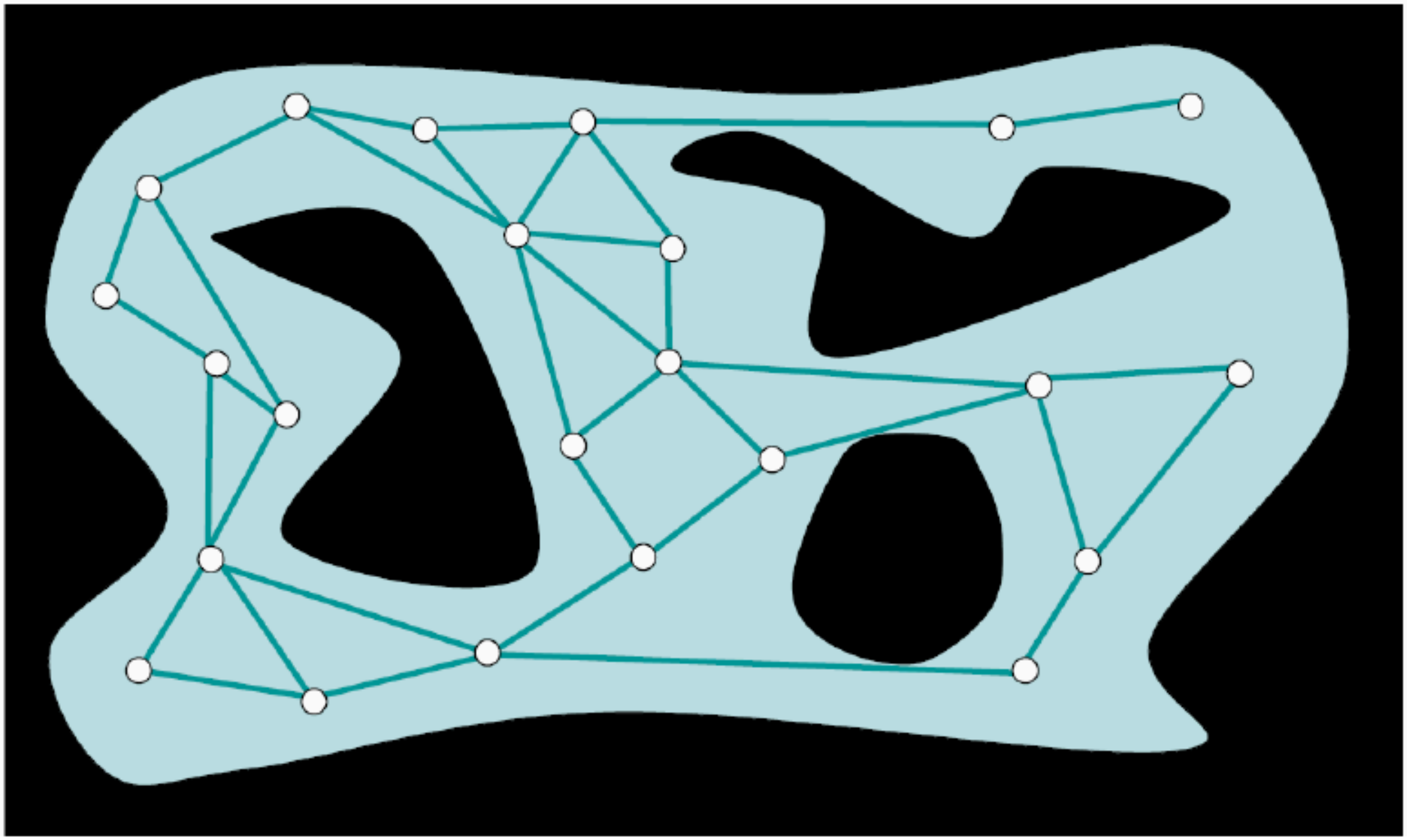
Sampling Techniques

Remove the links that cross forbidden regions



Sampling Techniques

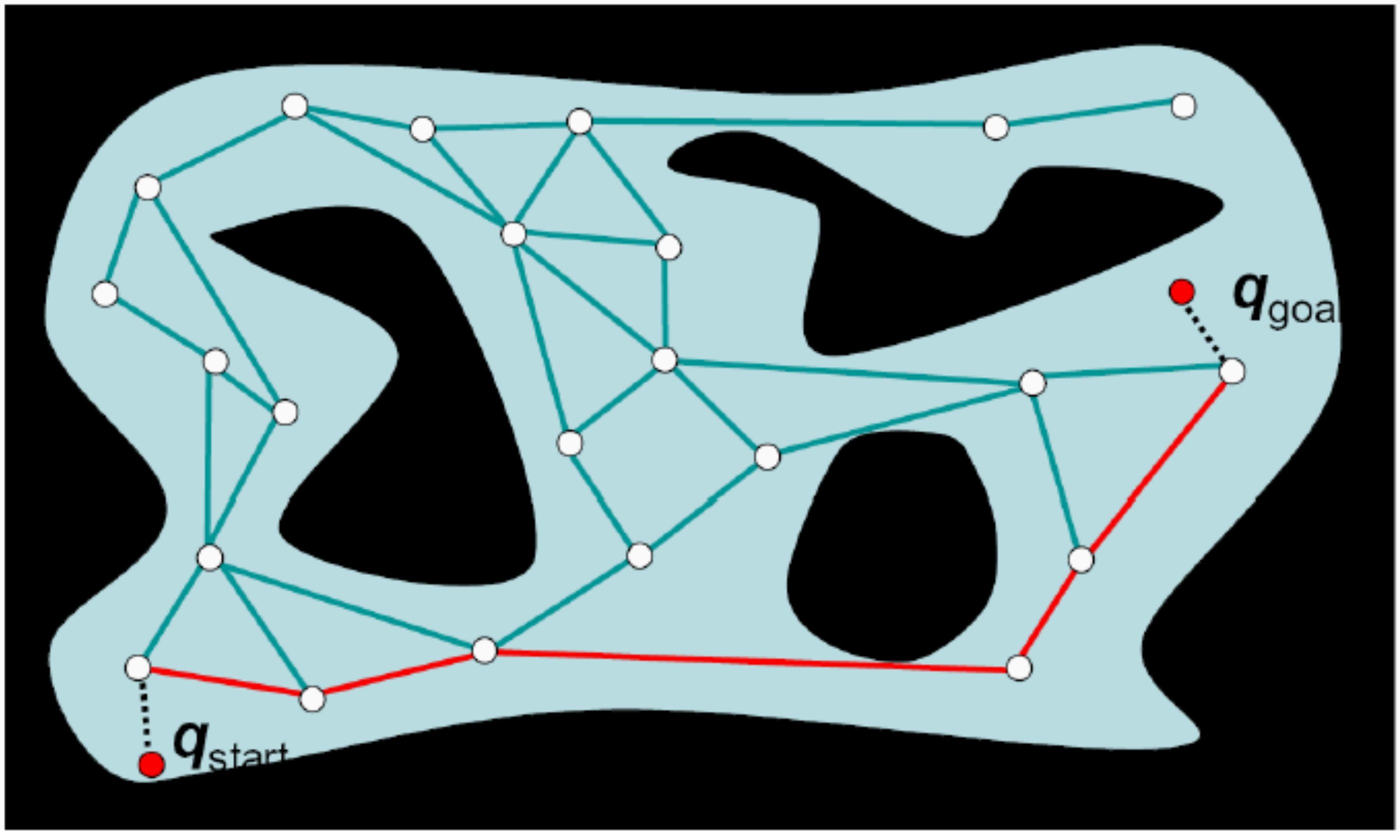
Remove the links that cross forbidden regions



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

Sampling Techniques

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



Choset slides

Sampling Techniques

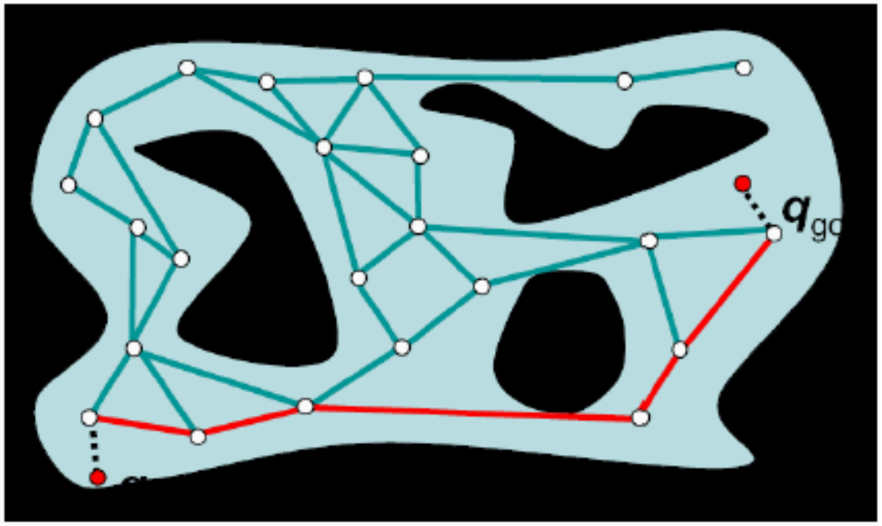
Continuous Space

↓

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

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

Random trees

- Notice how randomized roadmap is for “any plan”
  - but we may not need that
  - plan for a specific start, a specific goal
- For the moment, focus on start
  - grow a tree with start at root
  - join tree to goal
    - perhaps by growing backward from goal, and linking
- Q: how to grow the tree?

Naïve Random Tree

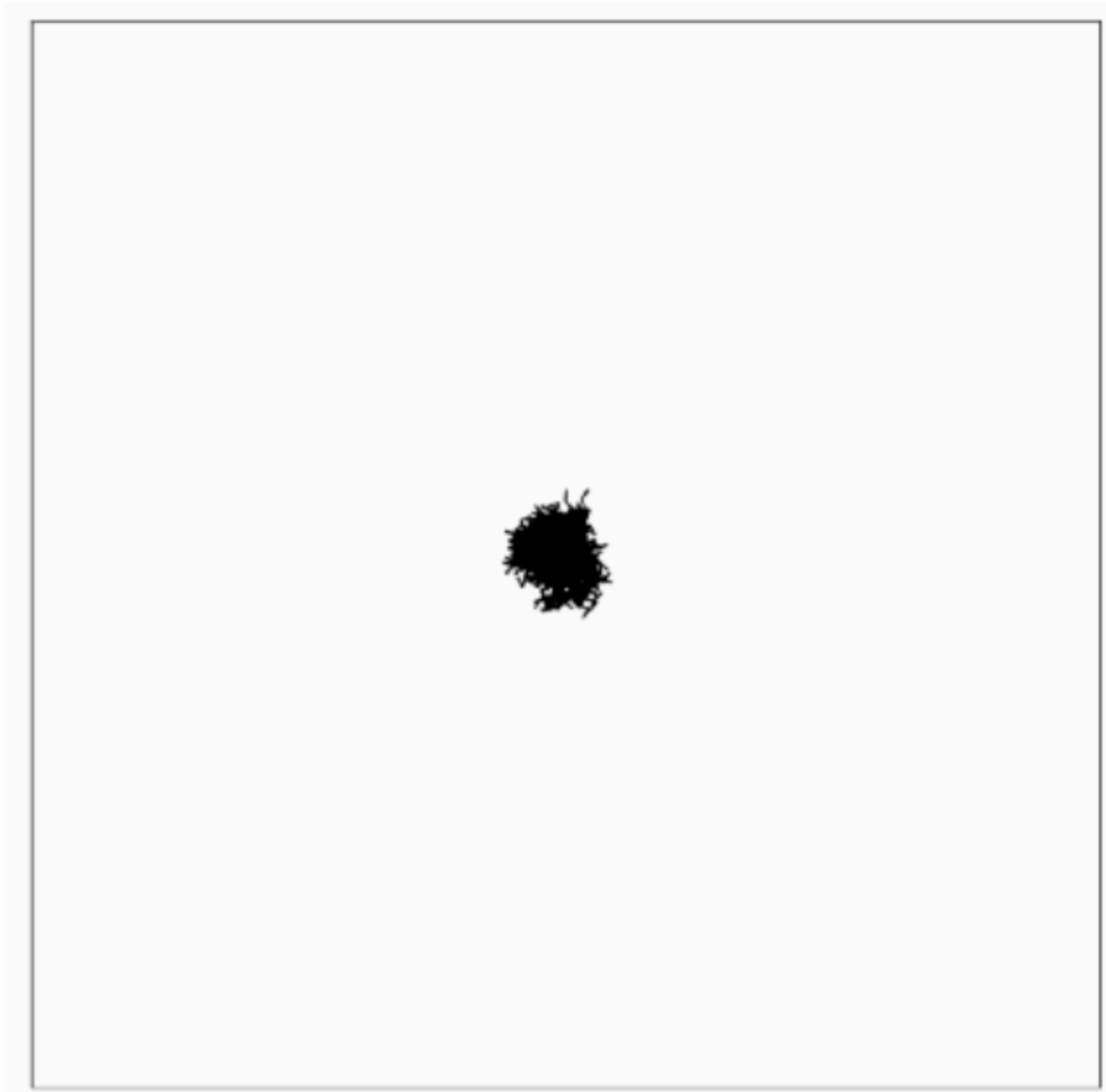
Start with middle

Sample near this node

Then pick a node at random in tree

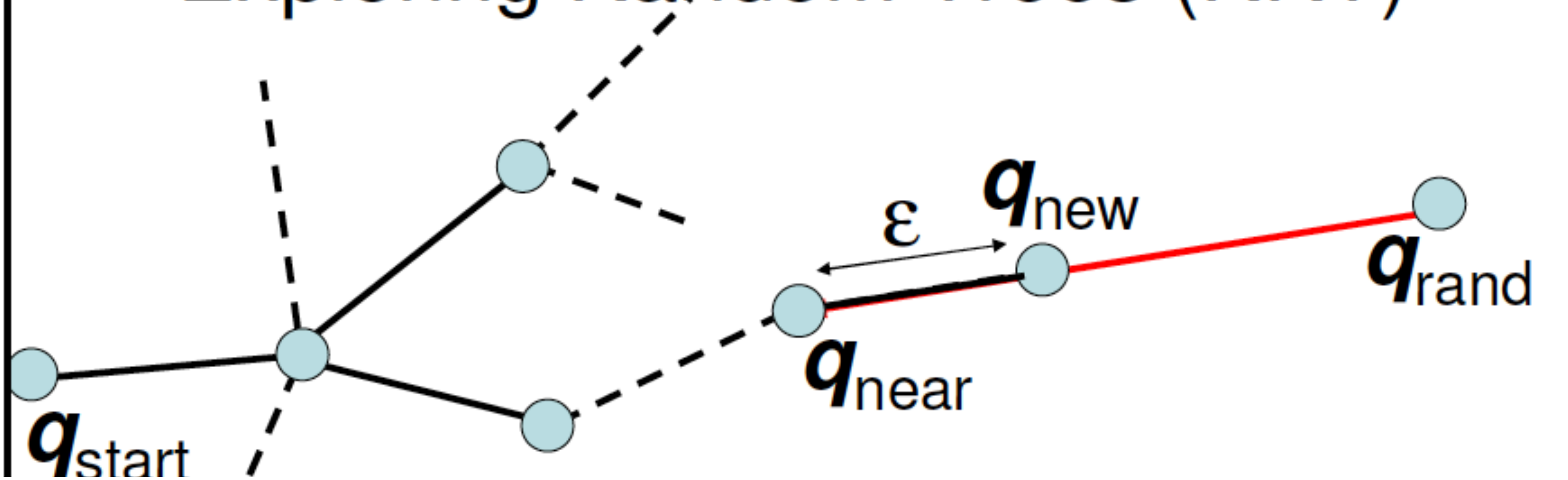
Sample near it

End up Staying in middle



Kosecka slides

Even More Radical: Rapidly Exploring Random Trees (RRT)

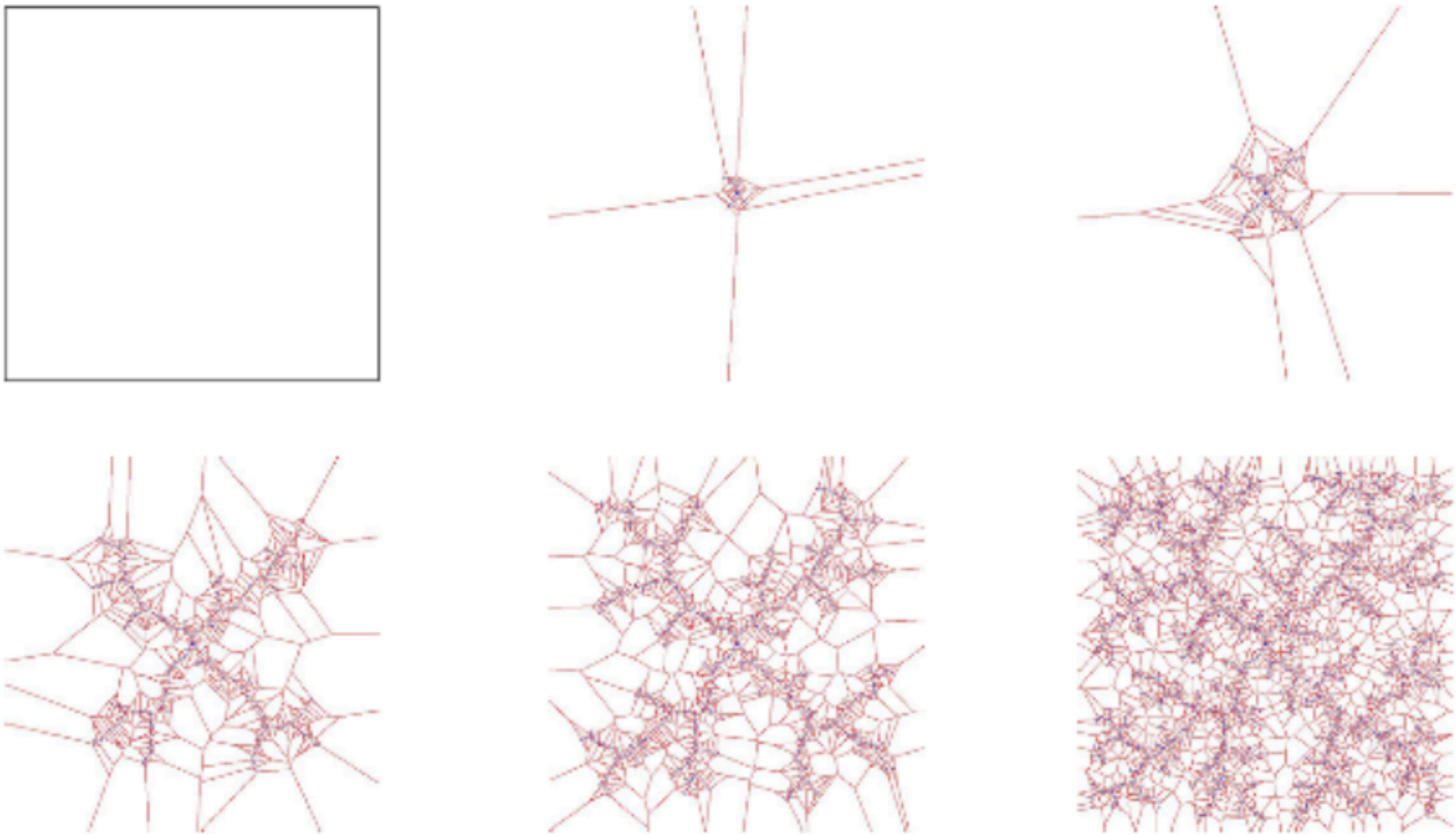



```
Algorithm BuildRRT
    Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
    Output: RRT graph G

    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

RRT's are biased towards large Voronoi cells



The nodes most likely to be closest to a randomly chosen point in state space are those with the largest Voronoi regions. The largest Voronoi regions belong to nodes along the frontier of the tree, so these frontier nodes are automatically favored when choosing which node to expand.

Kosecka slides

RRT's expand (another way)

- The nodes of the tree are
  - mostly on the boundary
  - of a "blob" of nodes
    - because that's where the volume is
- Draw a sample in c-space
  - if the blob is spread out in c-space, it's "inside", but we're OK
  - otherwise, sample is likely "outside"
    - so nearest node is very likely on boundary

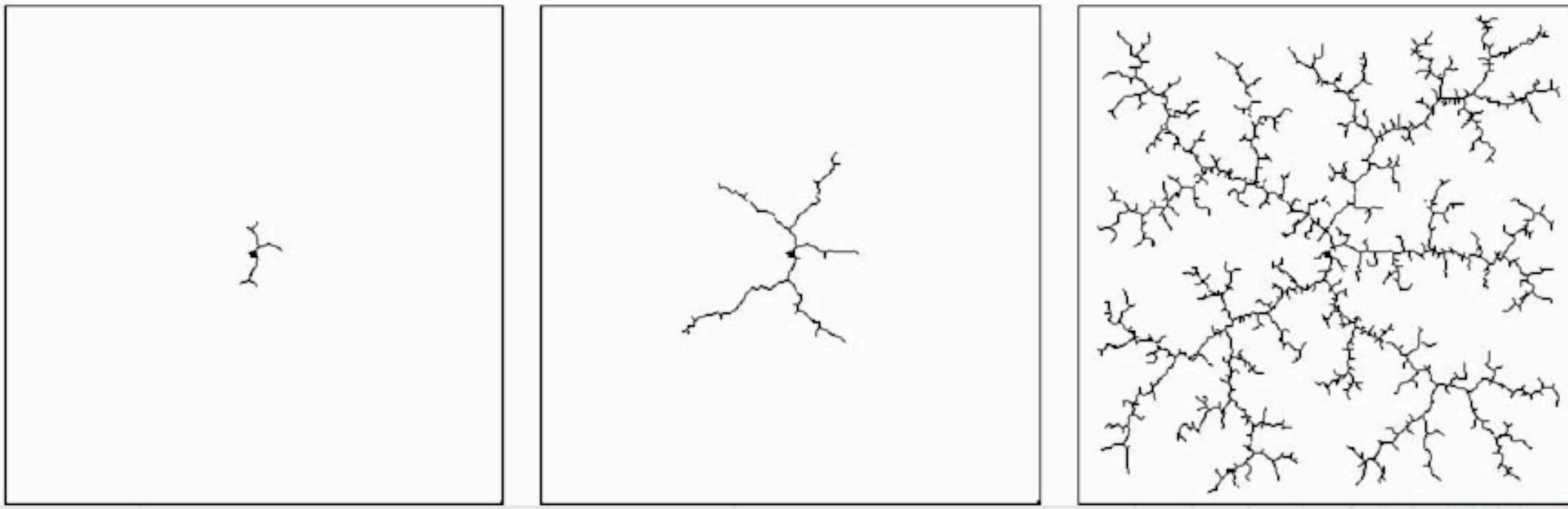
```
Algorithm BuildRRT
    Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
    Output: RRT graph G

    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
  - this could be tricky
- Notice
  - tree builds out into free space quickly
  - in different applications, one uses different epsilon
    - sometimes even add whole edge

Choset slides

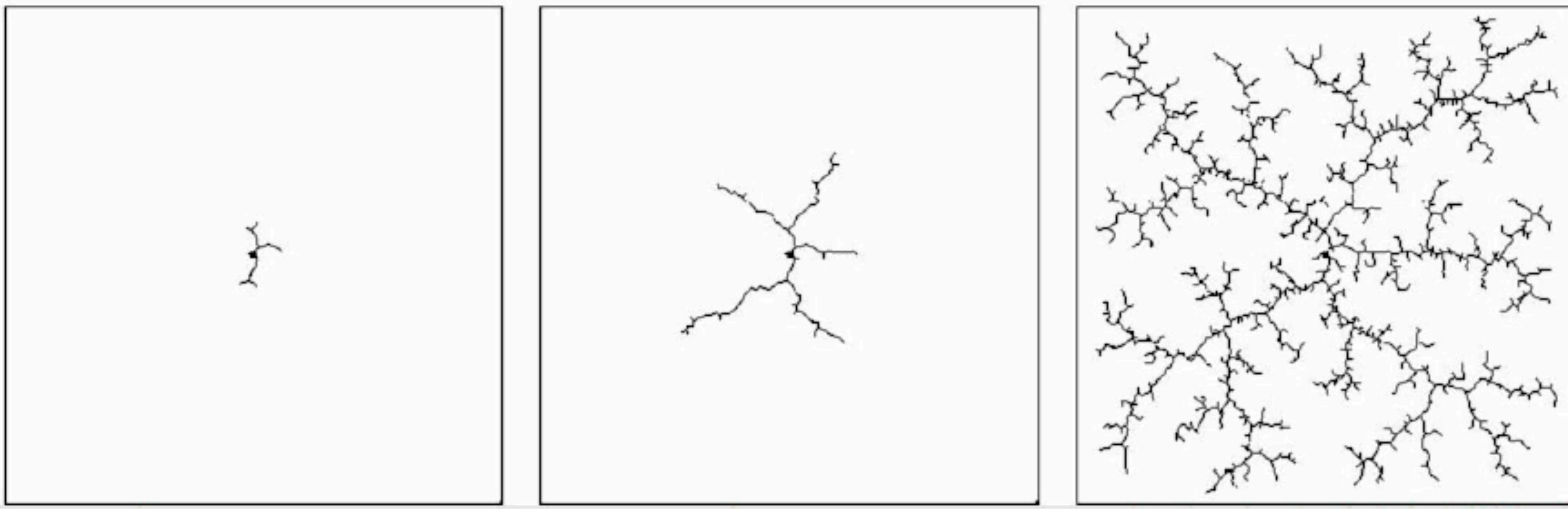
Properties



- 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

Choset slides

Properties



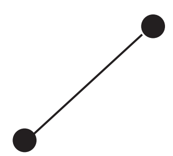
- Notice
  - Drawing the sample could get tricky
  - You need to be able to do the collision detection for the edge
    - BUT
      - many/most edges should be easy if there is a lot of free space

A quick conservative test - I

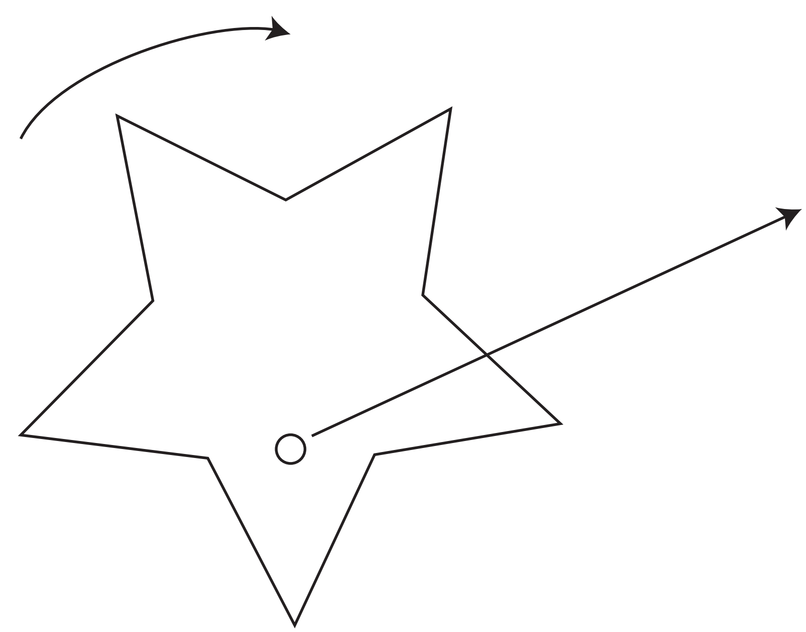
- Construct an axis aligned bounding box in 3-space
  - containing all configurations on the edge segment
  - how? below
- Test this box against objects
  - no intersection? edge is OK
  - intersection? more detailed test

A quick conservative test - II

- Building a box for robot rotating and translating
  - Robot rotates about origin in its own coordinate system
    - this origin translates



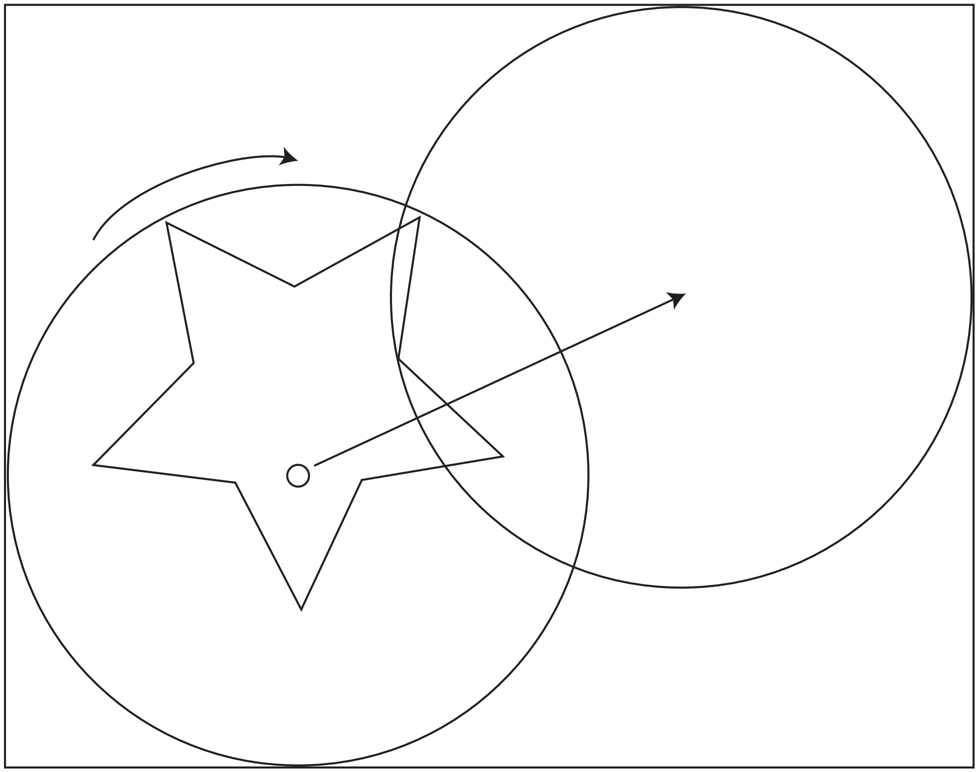
In c-space



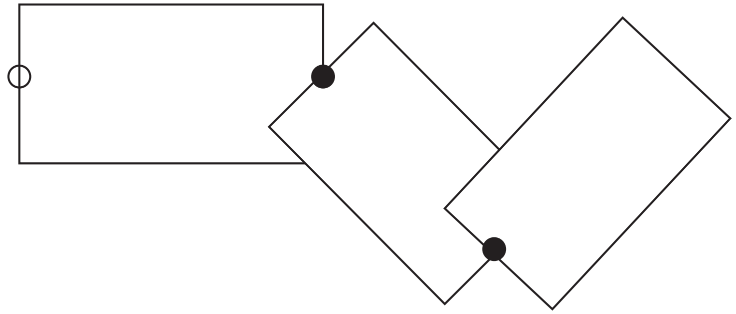
In 3D

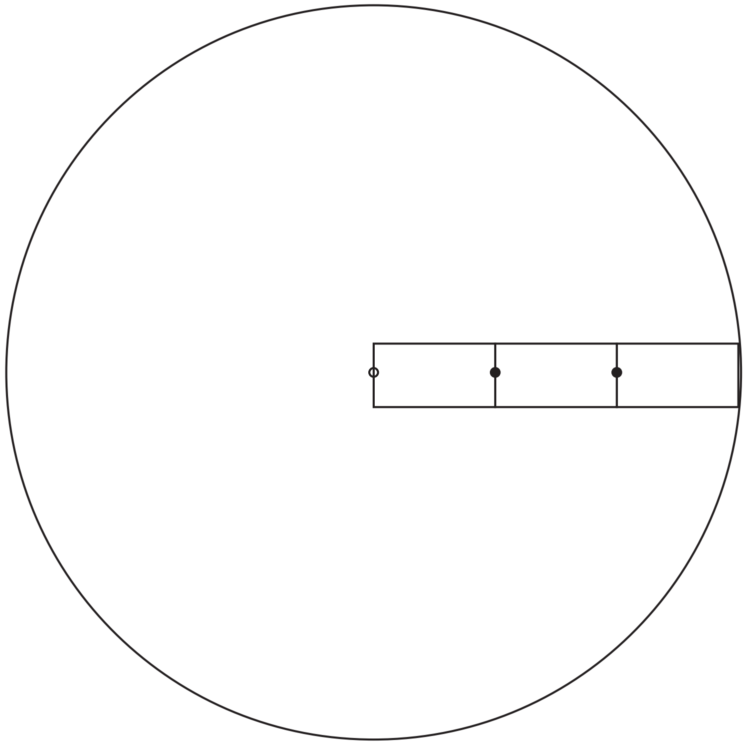
A quick conservative test - II

- Building a box for robot rotating and translating
  - Robot rotates about origin in its own coordinate system
    - this origin translates
  - Build bounding sphere, centered on origin, in advance
    - Translate this sphere's center - yields box
- Loose, quick bound
  - Loose
    - if segment intersects by this test
      - subdivide and go again



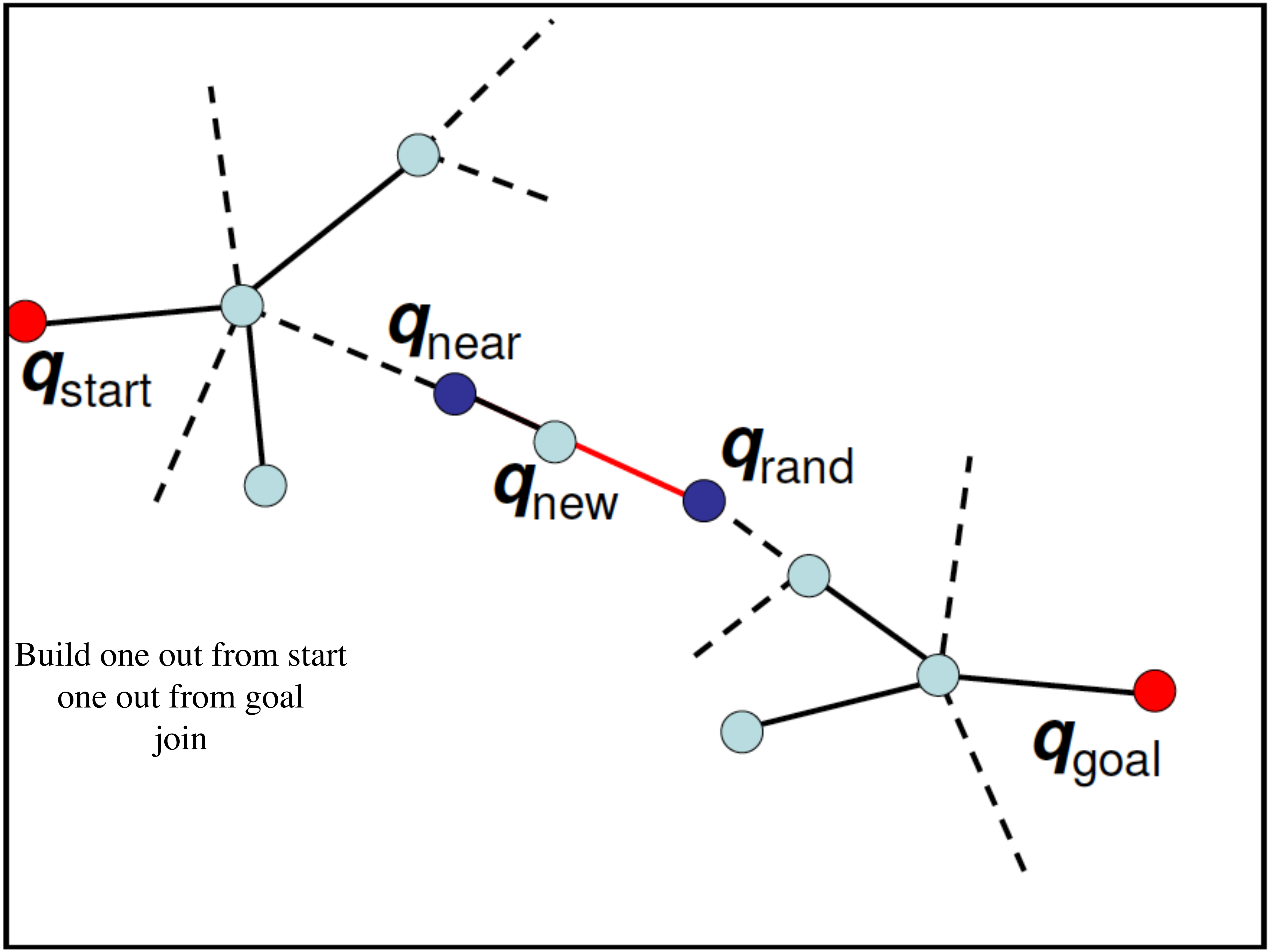
Bad for kinematic chains





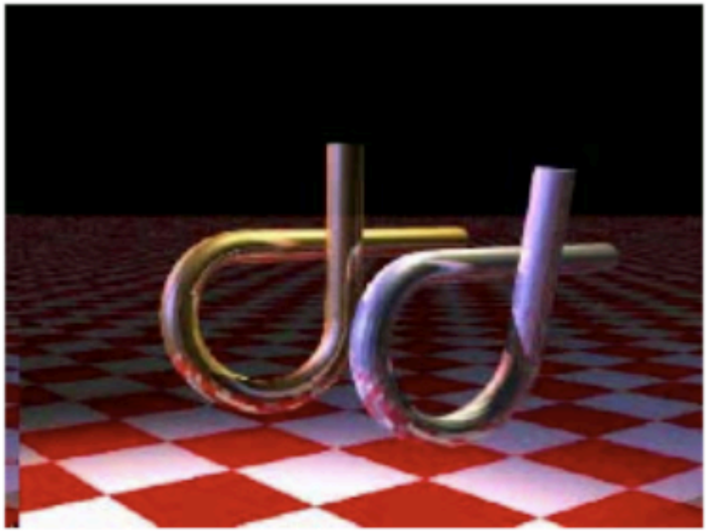
Bad for kinematic chains - II

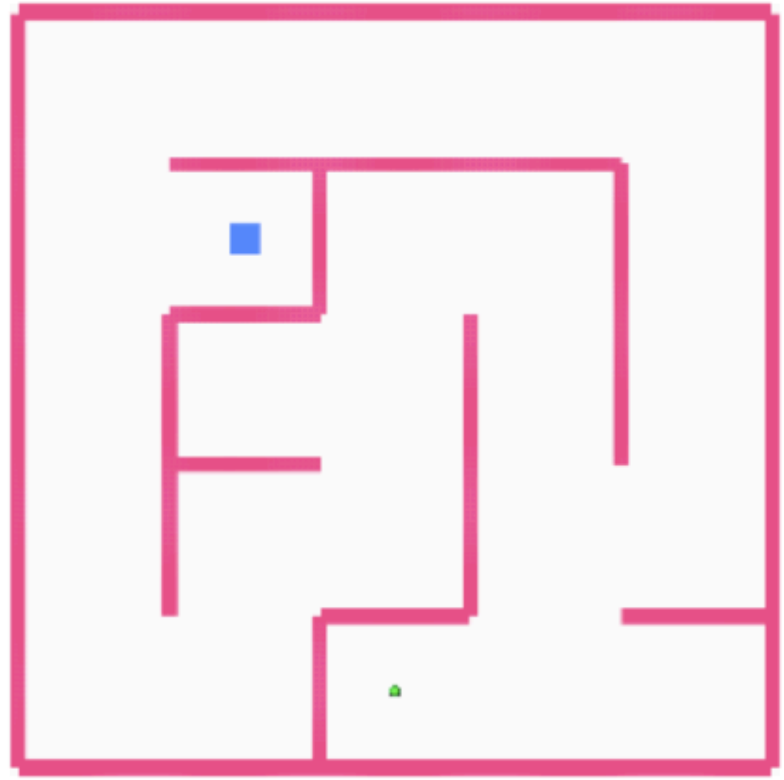
- Specialized techniques
  - typically per segment bounds
  - see Lavalle chapter, on website



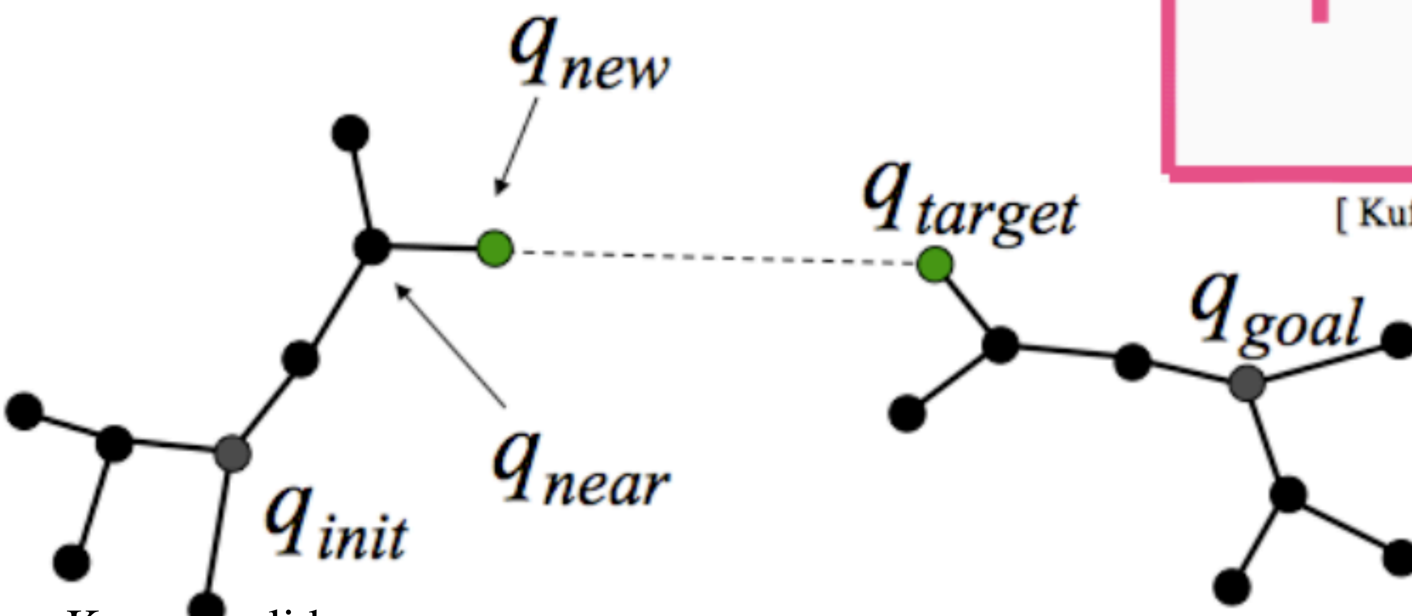
Choset slides

Grow two RRT's together





[Kuffner, LaValle ICRA '00]



Kosecka slides

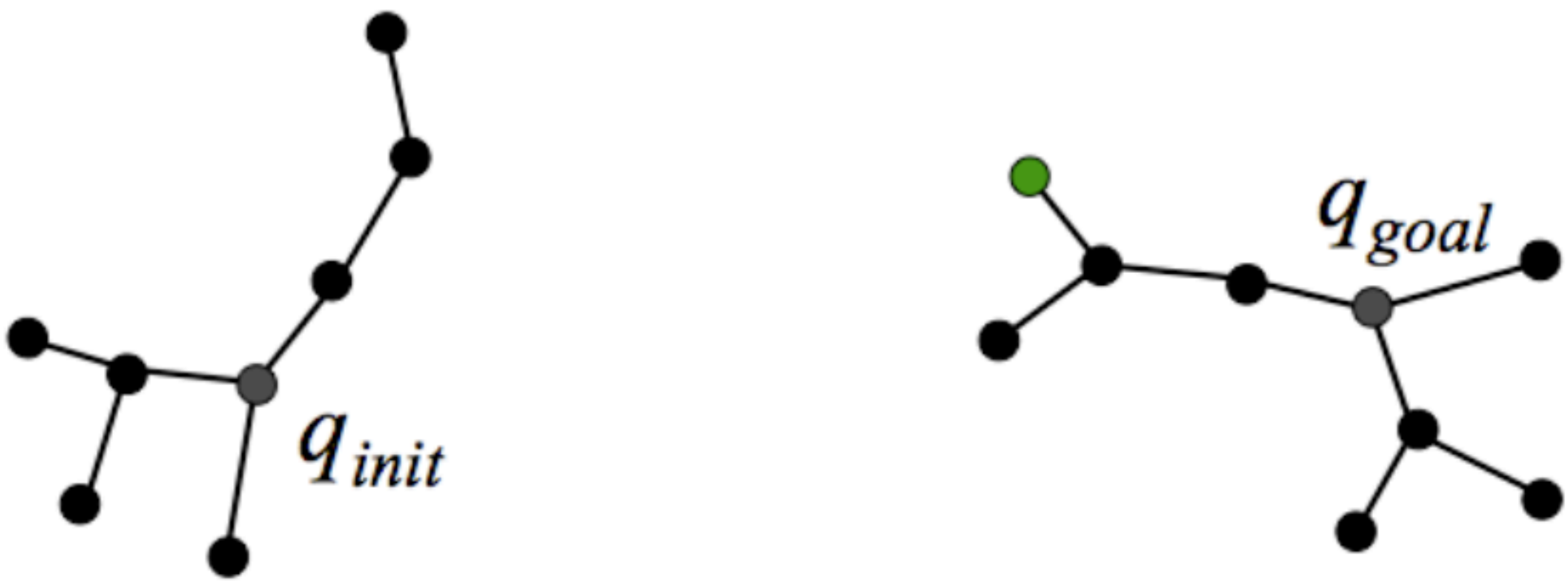
Two RRT's

A single RRT-Connect iteration...

Kosecka slides

Two RRT's

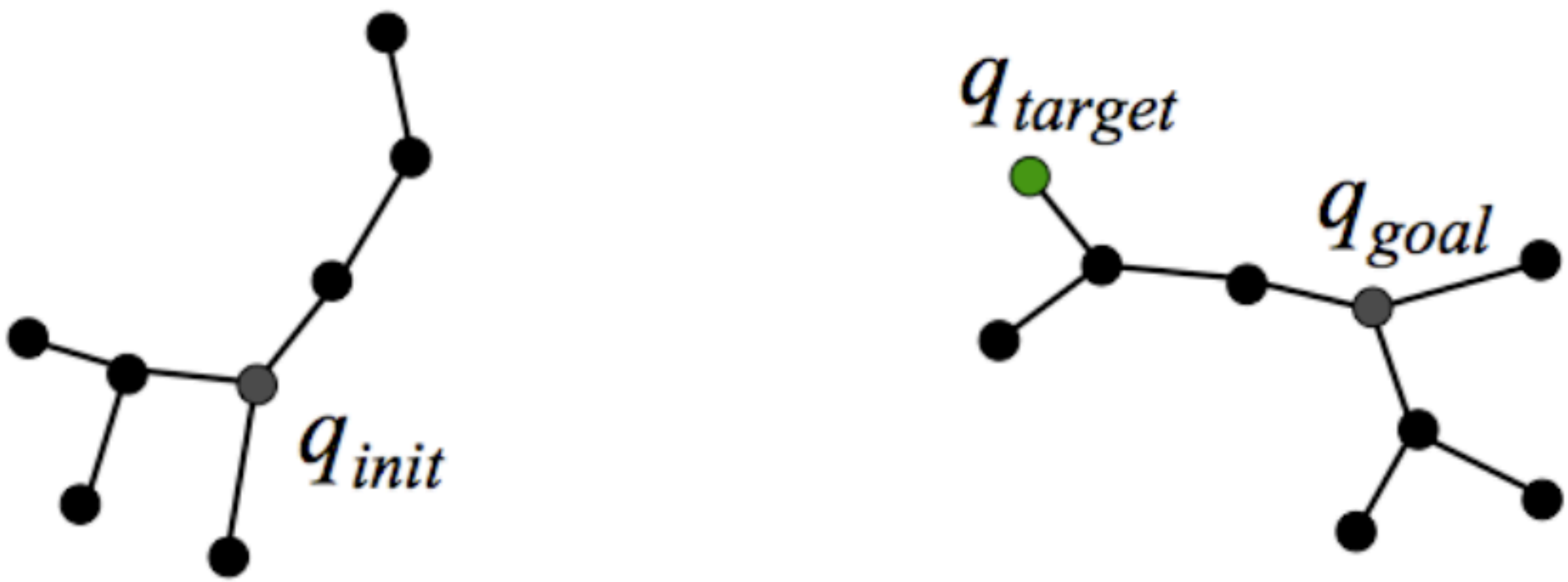
1) One tree grown using random target



Kosecka slides

Two RRT's

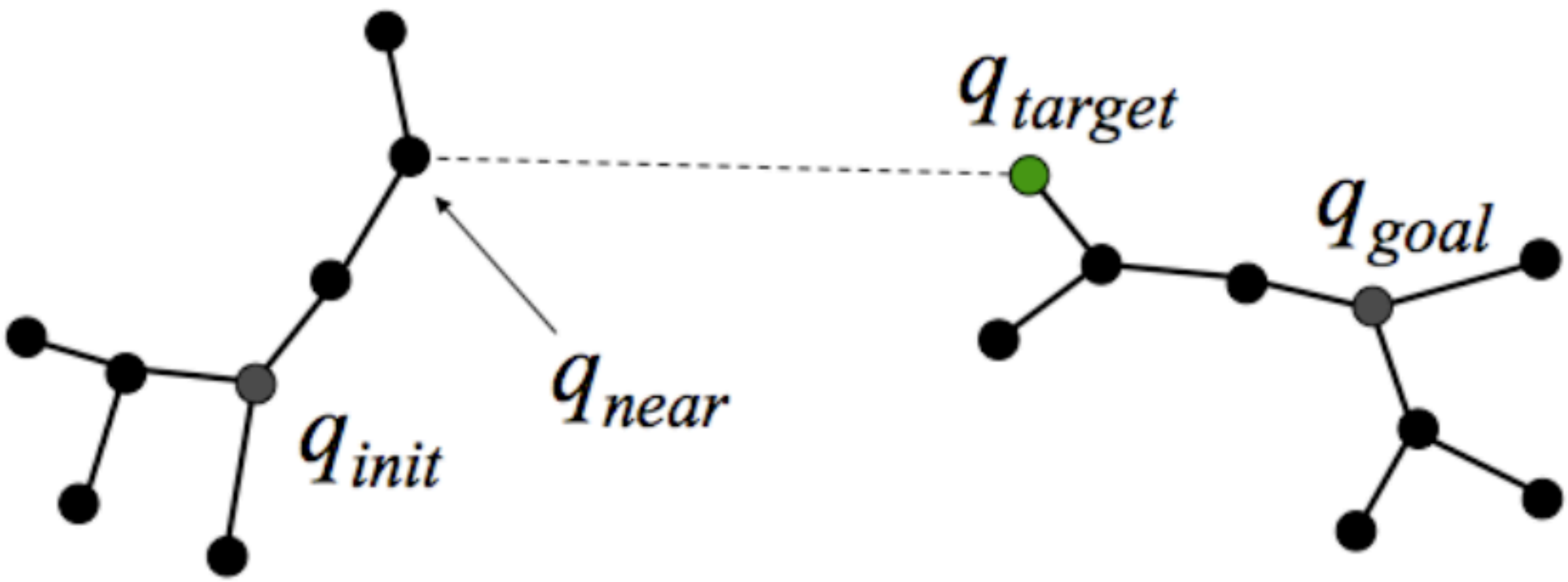
2) New node becomes target for other tree



Kosecka slides

Two RRT's

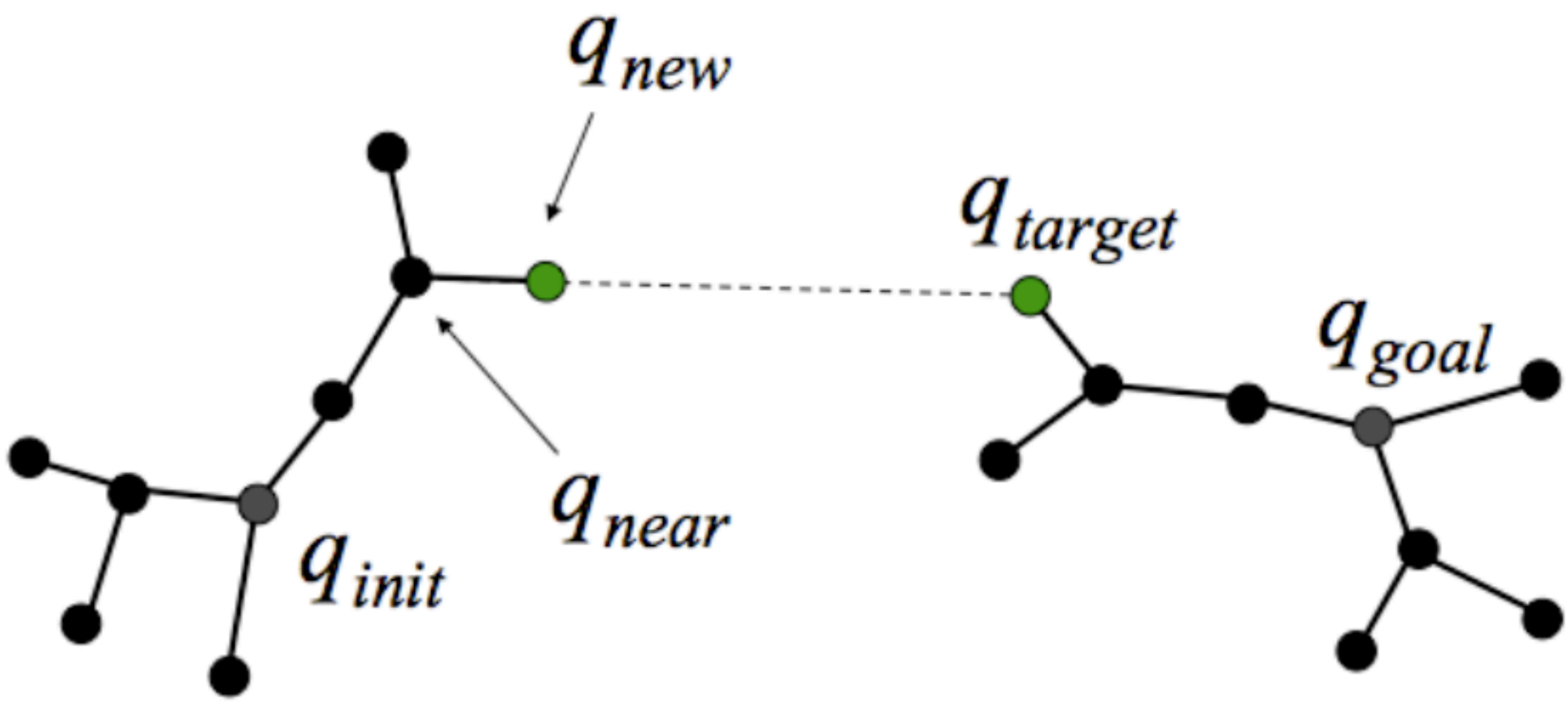
3) Calculate node "nearest" to target



Kosecka slides

Two RRT's

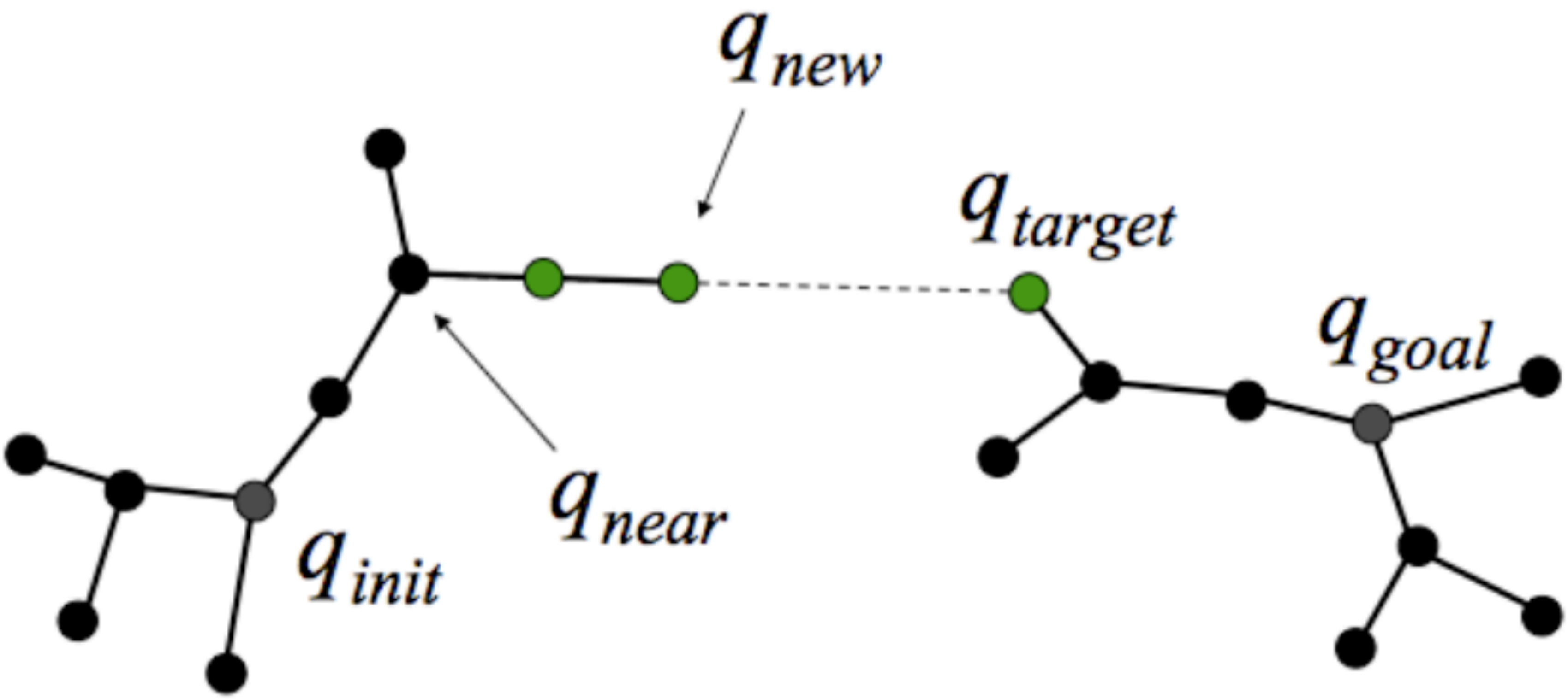
4) Try to add new collision-free branch



Kosecka slides

Two RRT's

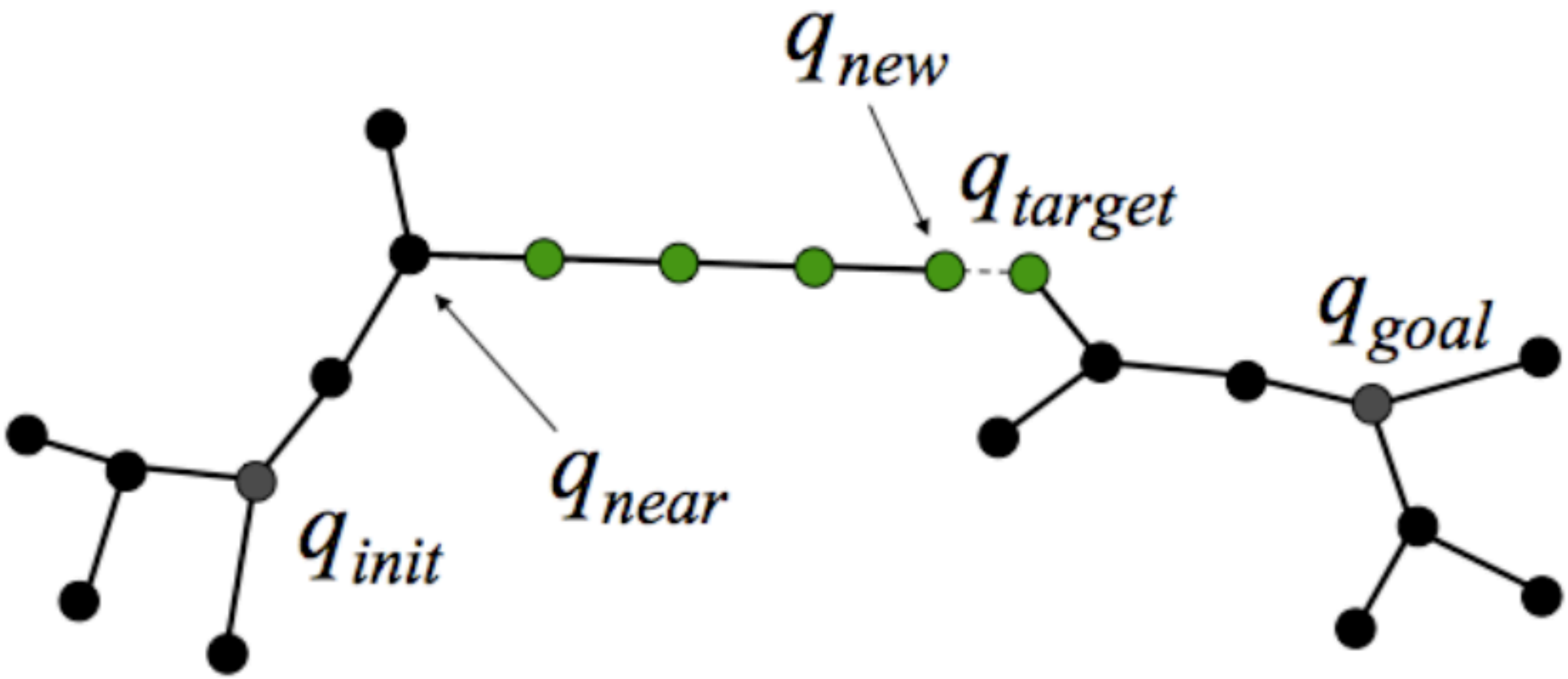
5) If successful, keep extending branch



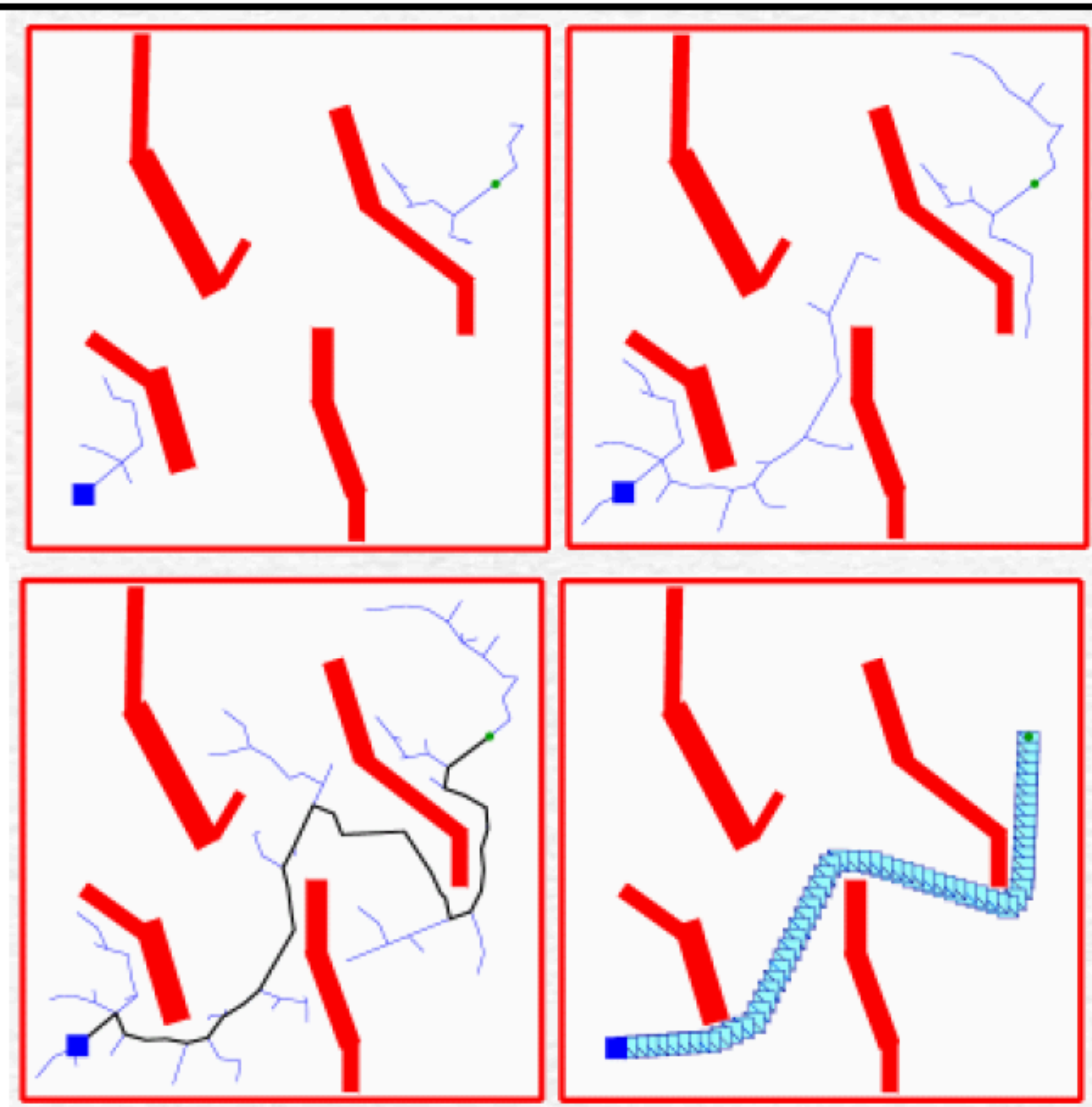
Kosecka slides

Two RRT's

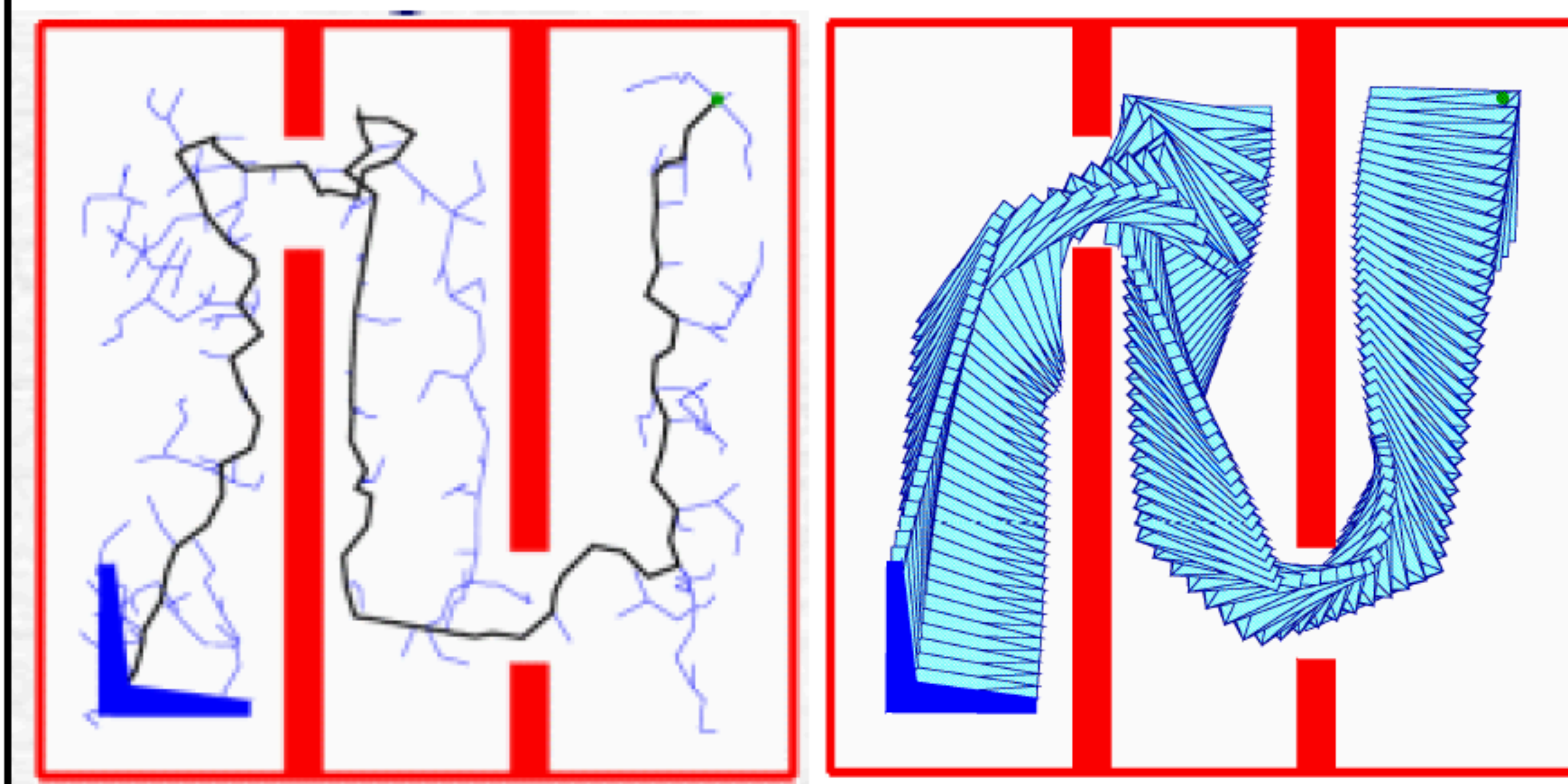
5) If successful, keep extending branch



Kosecka slides

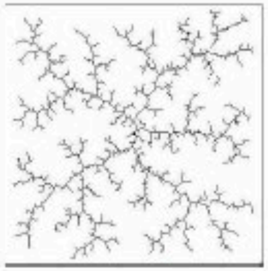
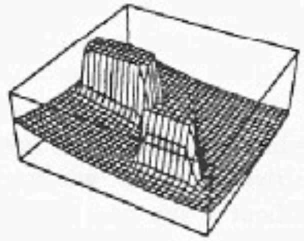
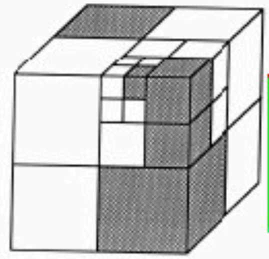
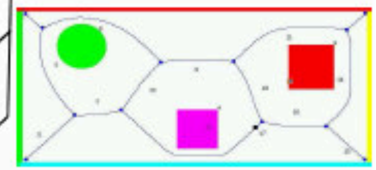
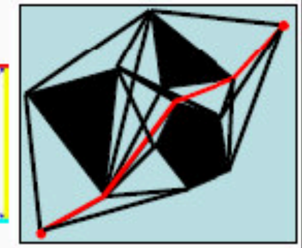


Choset slides



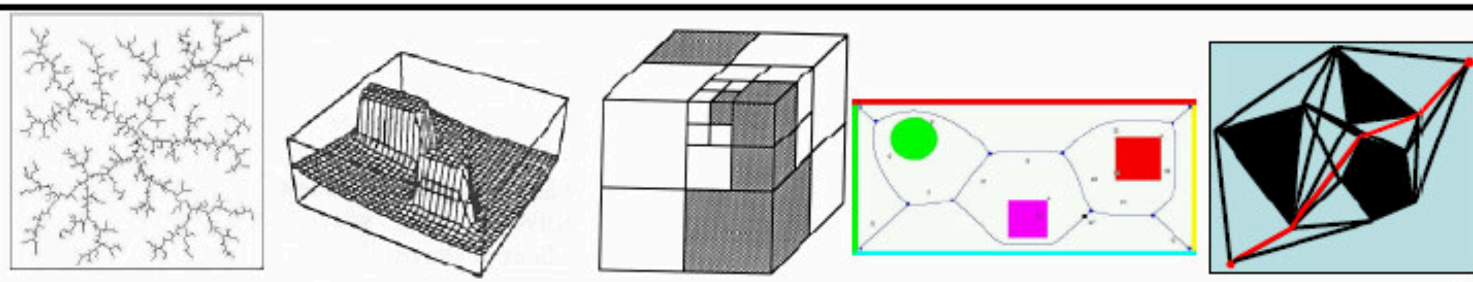
From Kuffner et al.

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/