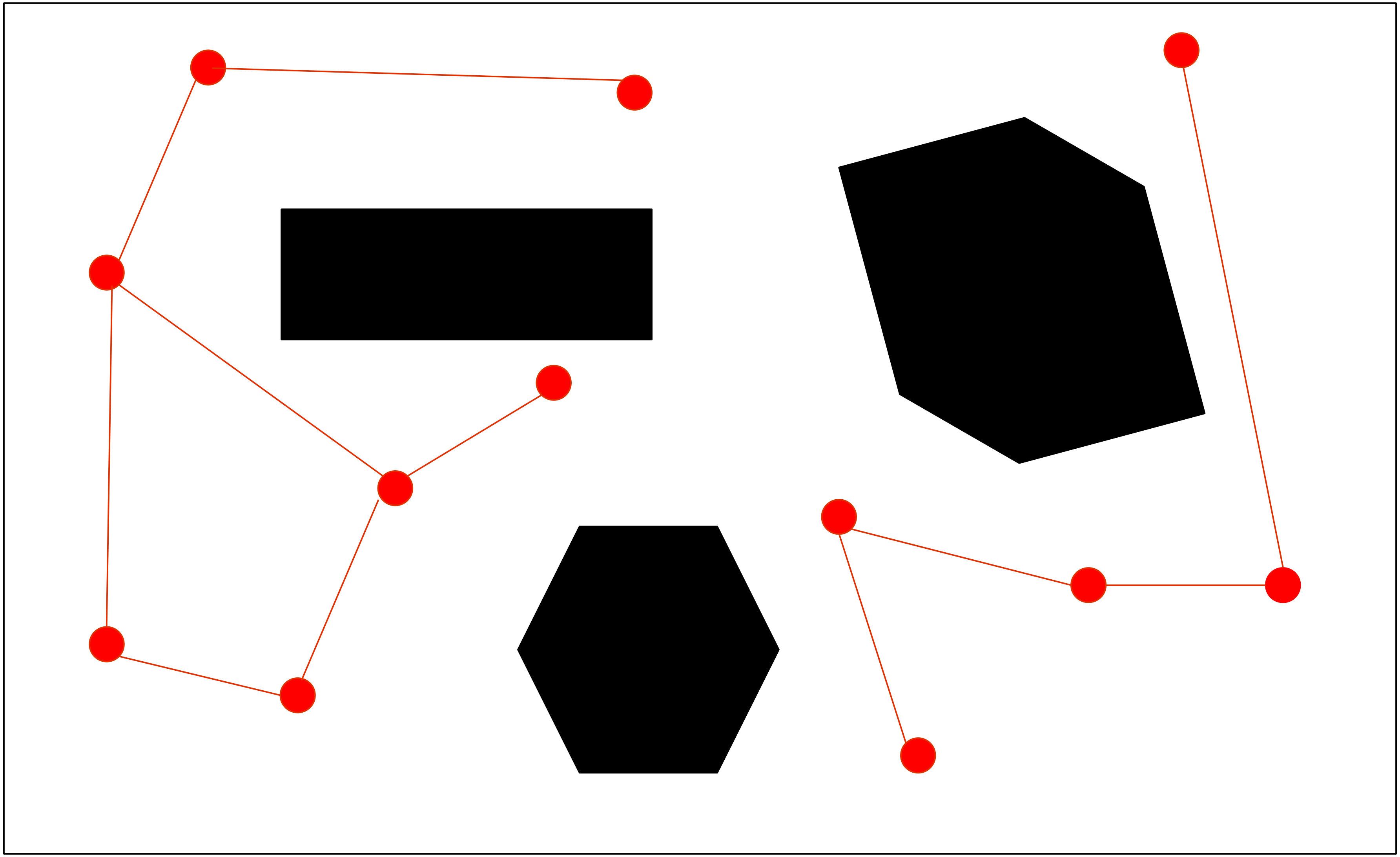
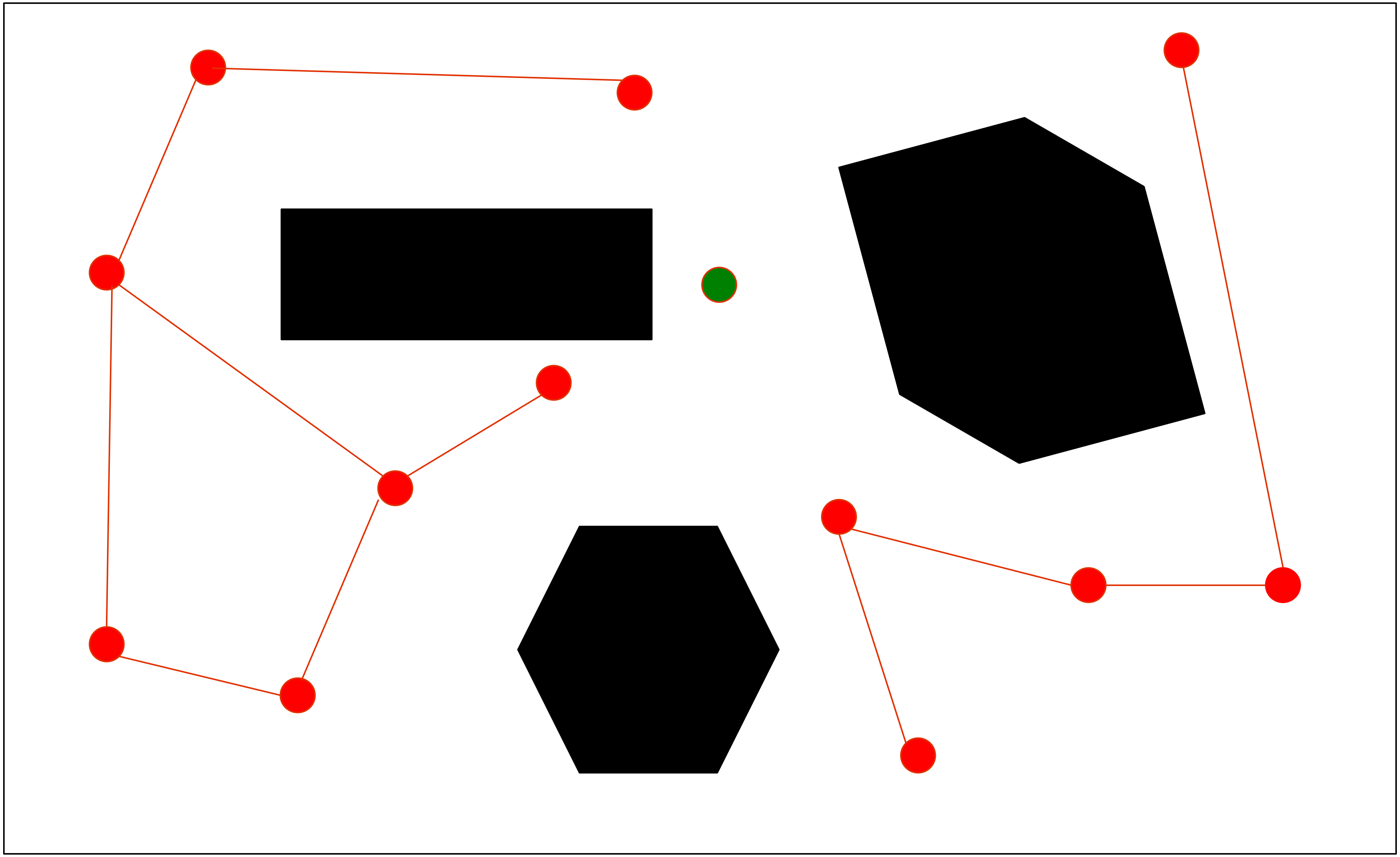
Video 11.1

CJ Taylor

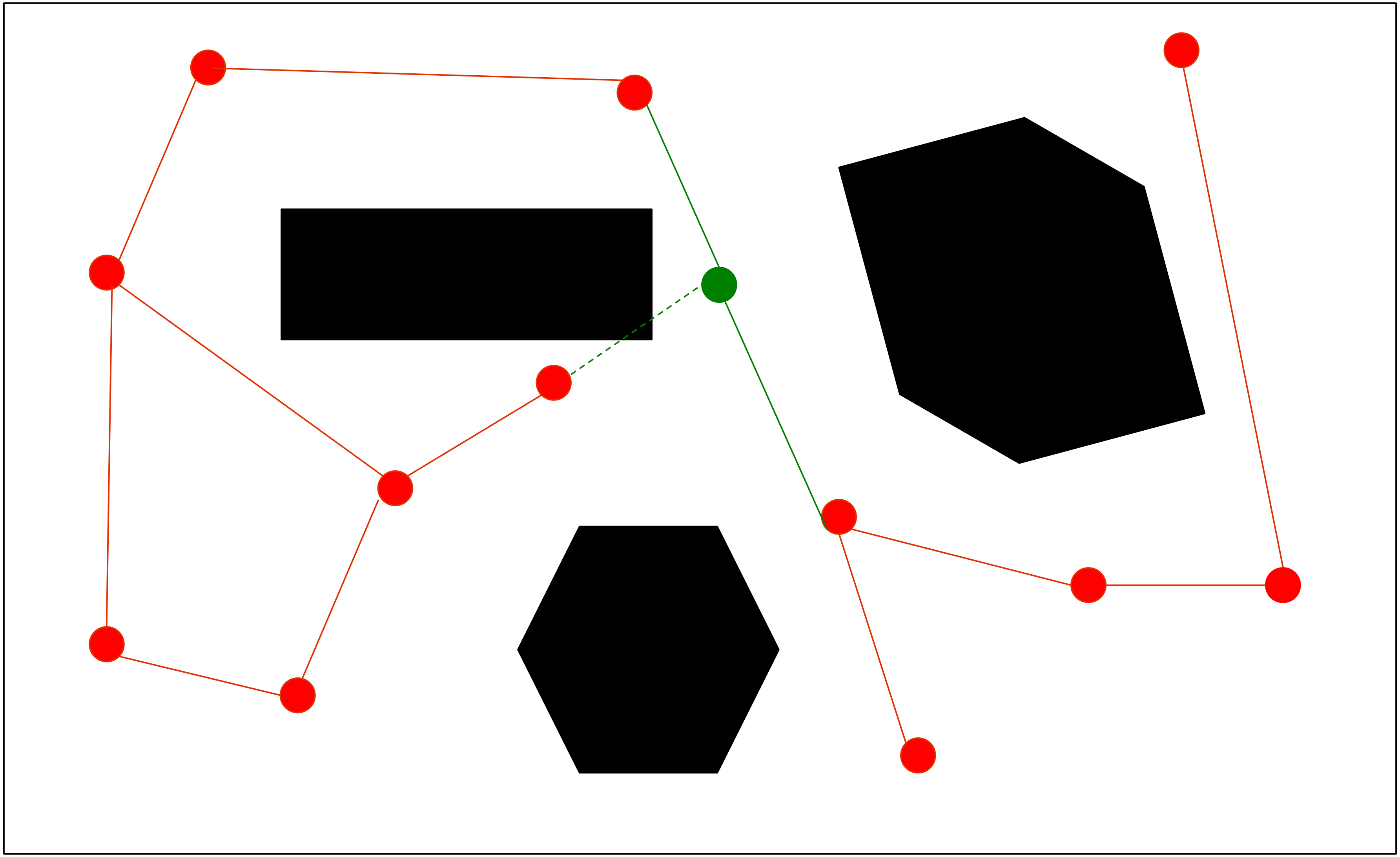
Random Graph Construction



Random Graph Construction



Random Graph Construction



Probablistic Road Map Pseudocode

- Repeat n times
  - Generate a random point in configuration space, **x**
  - If **x** is in freespace
    - Find the k closest points in the roadmap to **x** according to the Dist function
    - Try to connect the new random sample to each of the k neighbors using the LocalPlanner procedure. Each successful connection forms a new edge in the graph.

The Dist function

- The PRM procedure relies upon a distance function, Dist,that can be used to gauge the distance between two points in configuration space. This function takes as input the coordinates of the two points and returns a real number:

$$Dist(\mathbf{x}, \mathbf{y}) \in \mathbb{R}$$

- Common choices for distance functions include:
  - The L1 distance : $Dist_1 = \sum_i |\mathbf{x}_i - \mathbf{y}_i|$
  - The L2 distance : $Dist_2 = \sqrt{(\sum_i (\mathbf{x}_i - \mathbf{y}_i)^2)}$

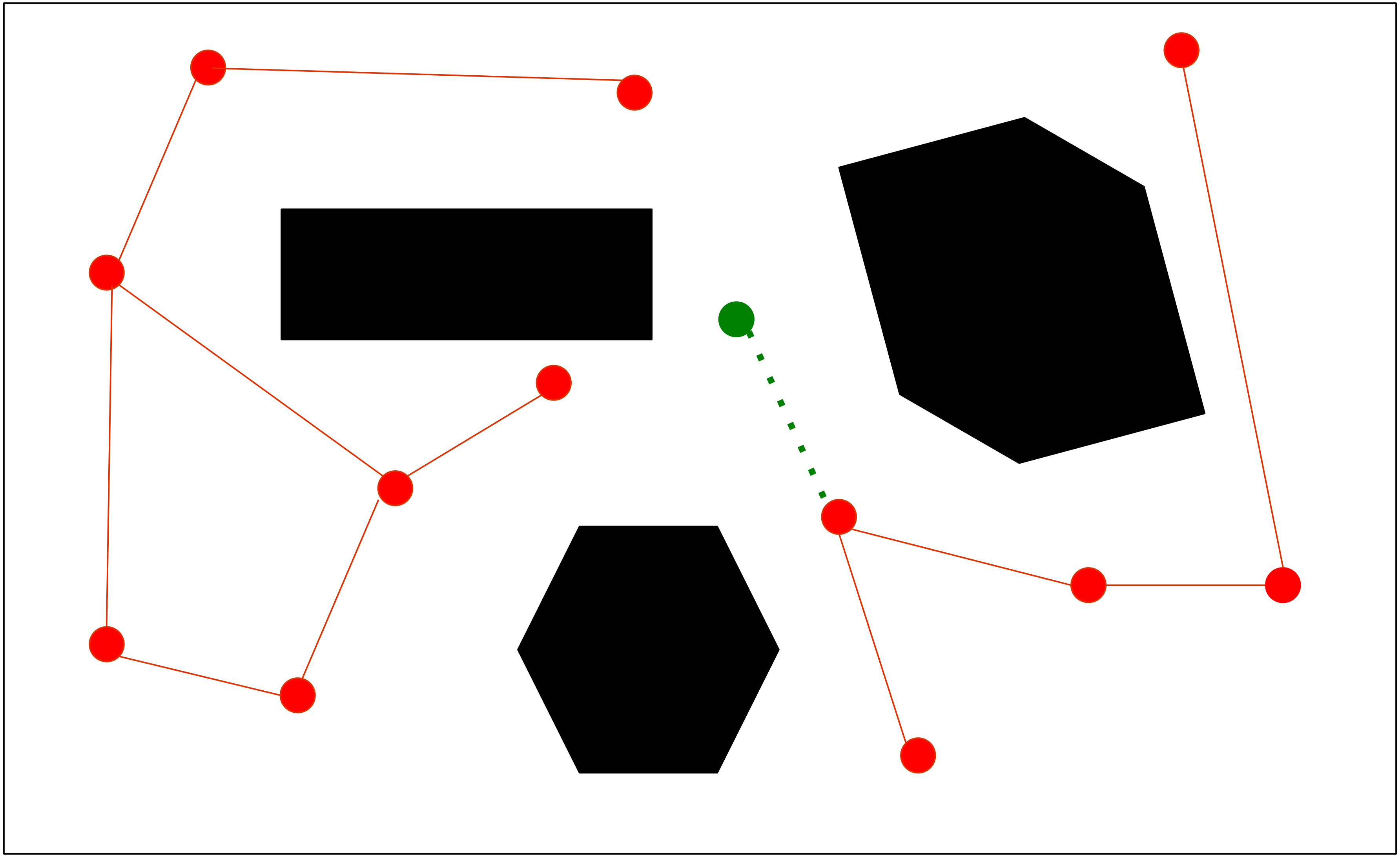
Handling angular displacements

- There are often cases where some of the coordinates of the configuration space correspond to angular rotations. In these situations care must be taken to ensure that the $Dist$ function correctly reflects distances in the presence of wraparound.

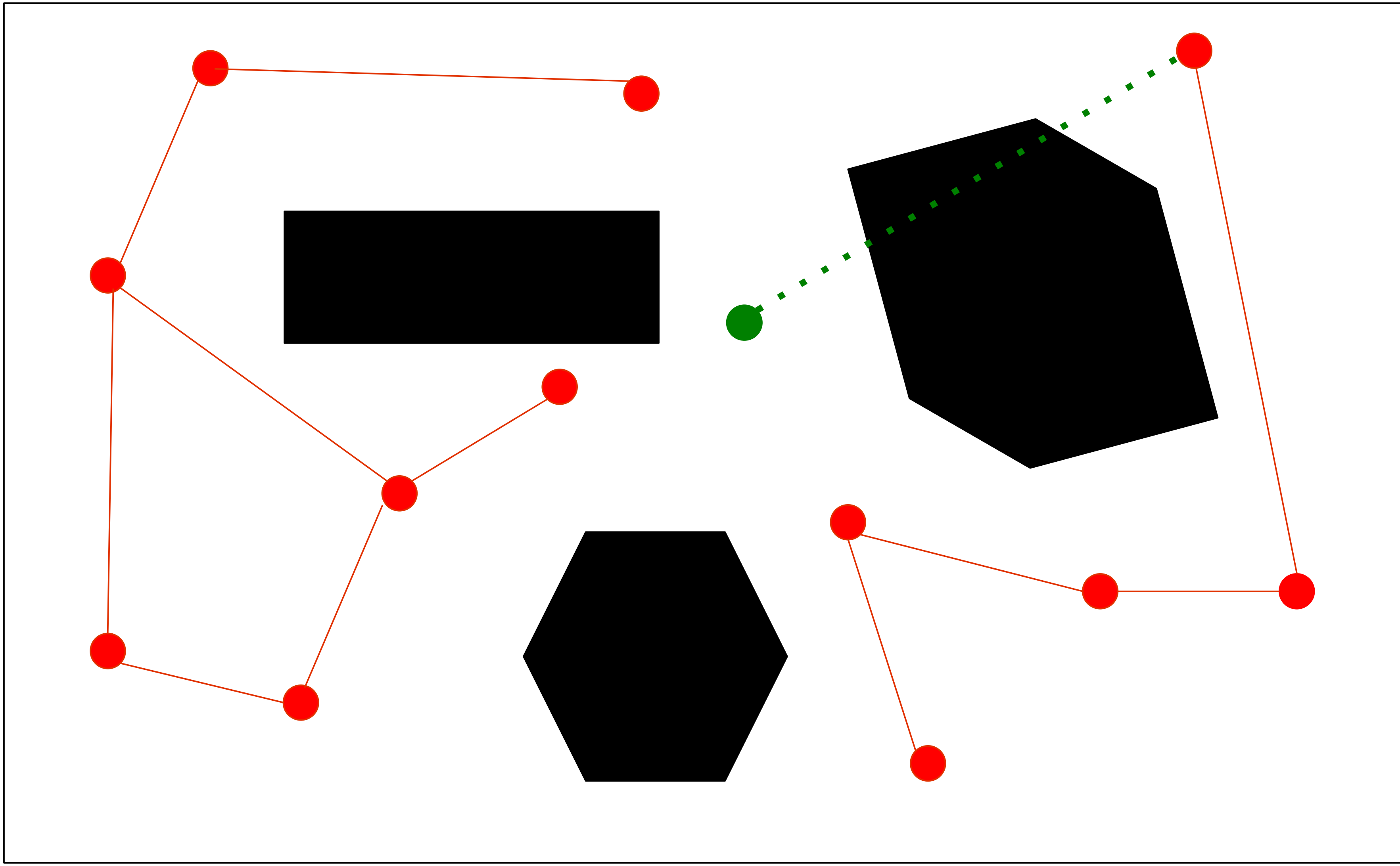
- For example if θ_1 and θ_2 denote two angles between 0 and 360 degrees the expression below can be used to capture the angular displacement between them.

$$Dist(\theta_1, \theta_2) = \min(|\theta_1 - \theta_2|, (360 - |\theta_1 - \theta_2|)) \tag{1}$$

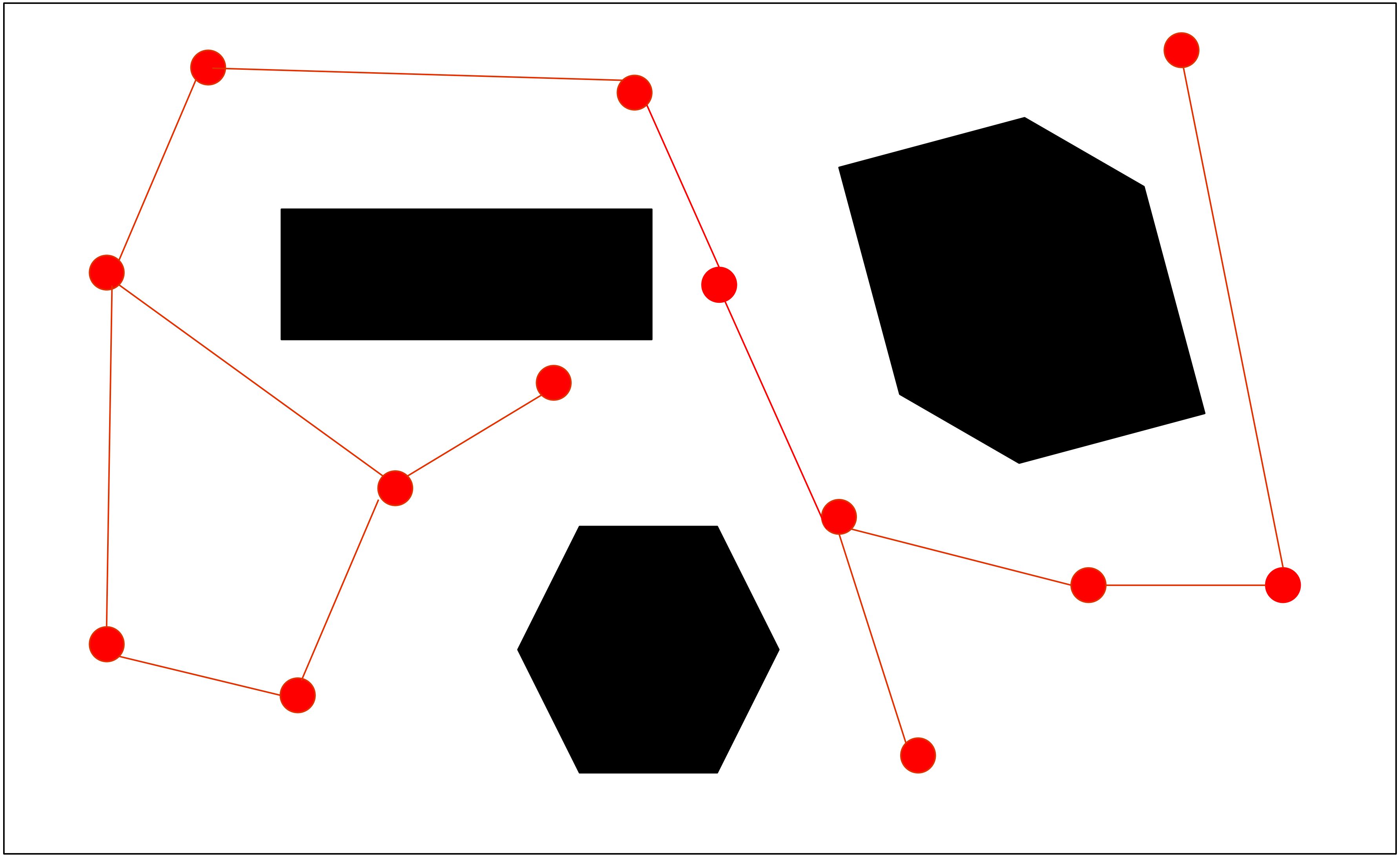
Checking for collision along a path



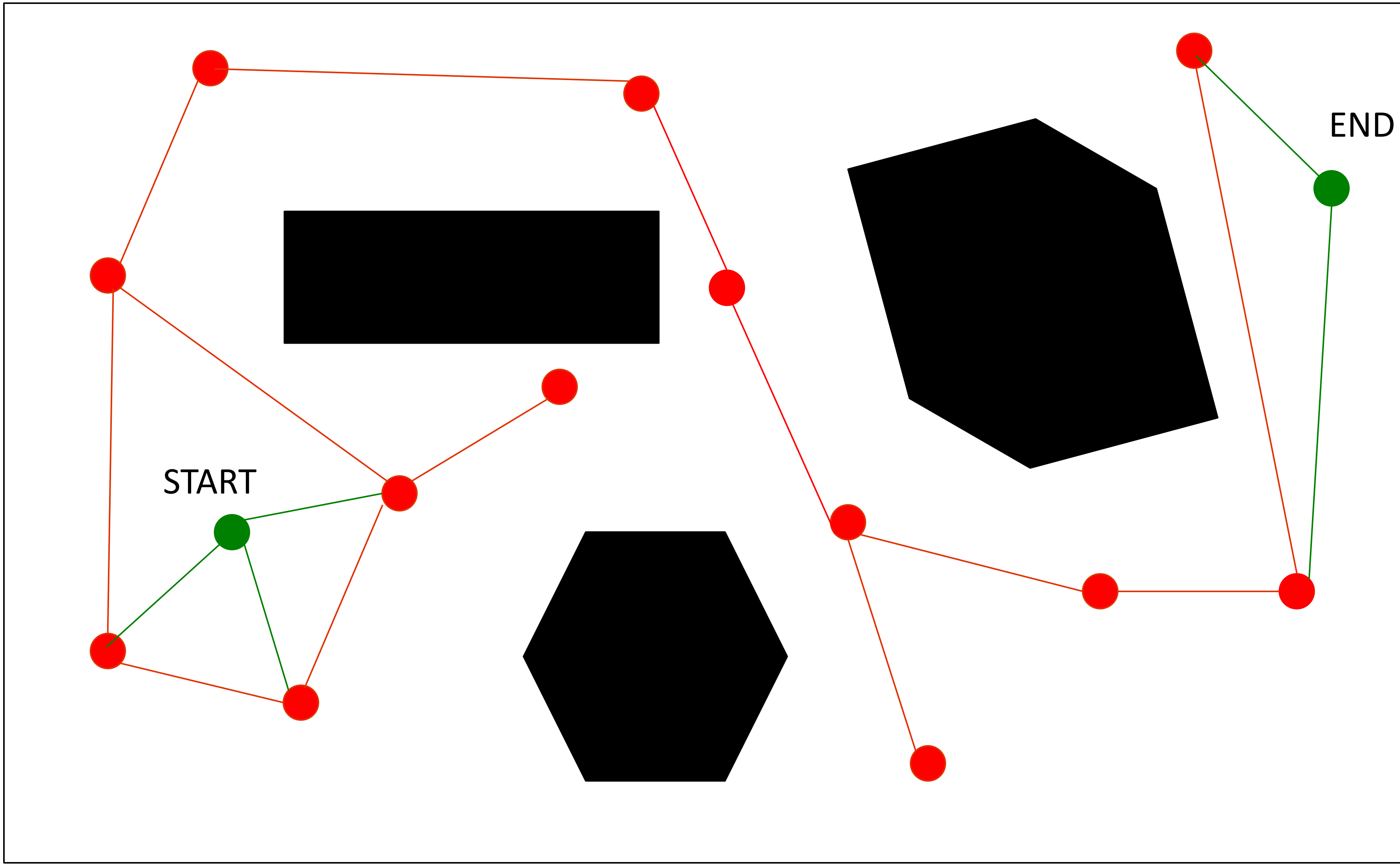
Checking for collision along a path



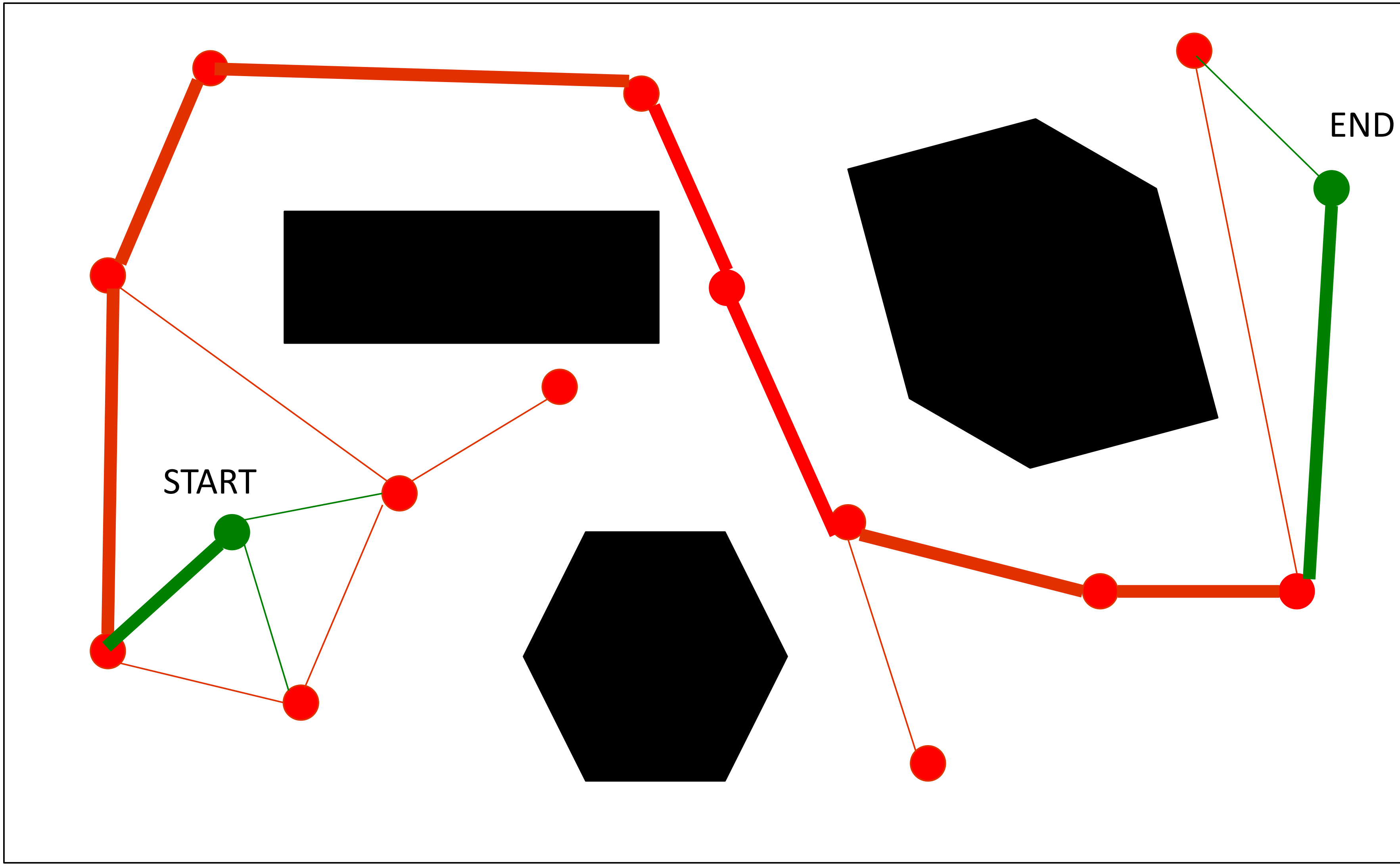
Initial Road Map



Road Map with Start and End added



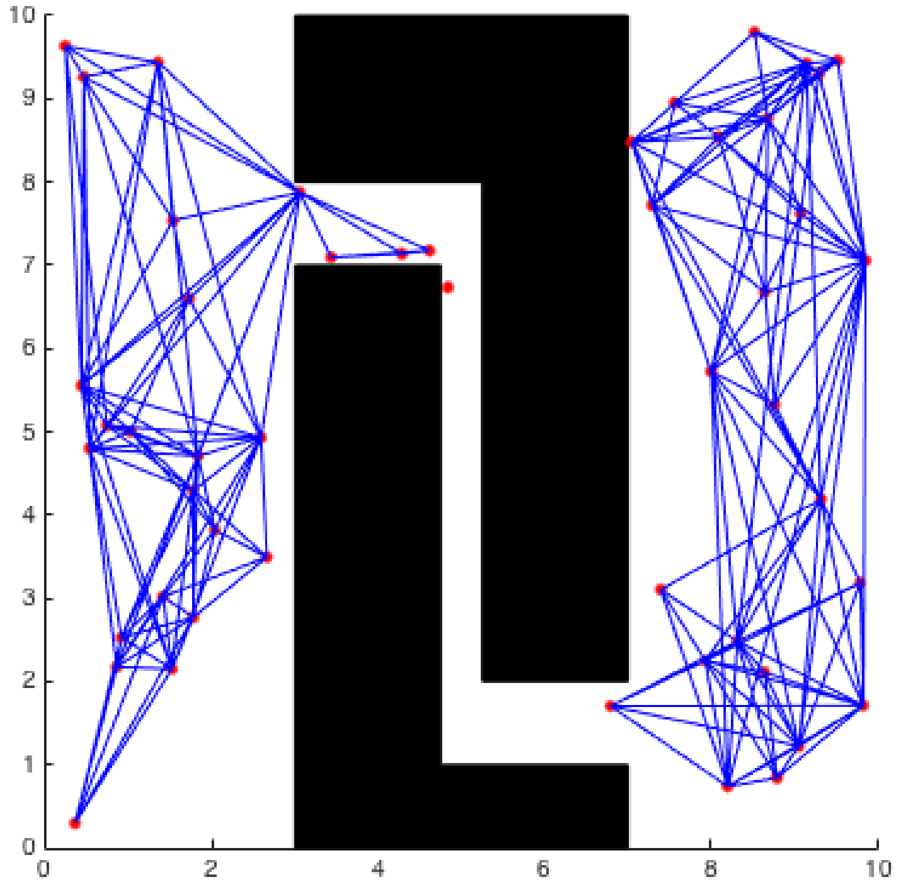
Final Route



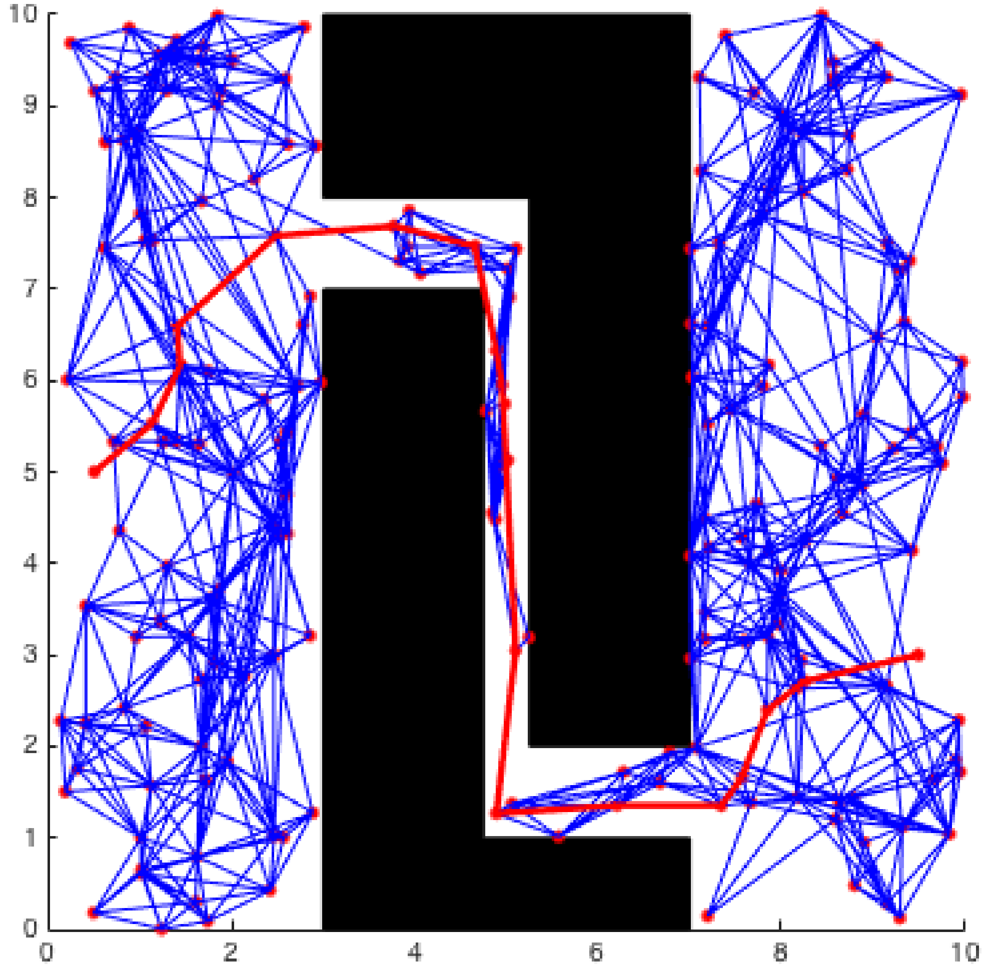
Video 11.2

CJ Taylor

Twisty Passageway – Failure Case



Twisty Passageway – Success via denser samples



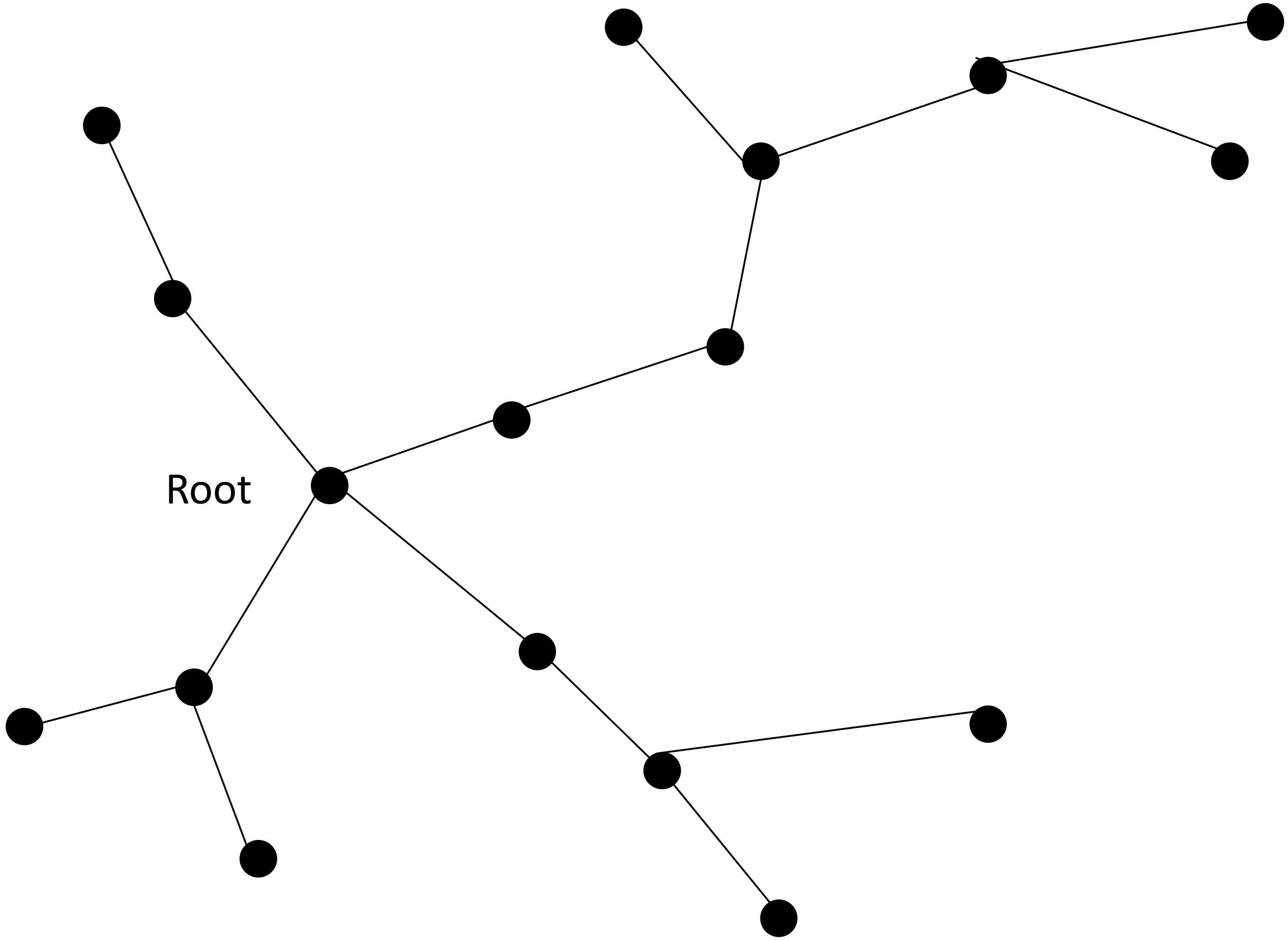
Video 11.3

CJ Taylor

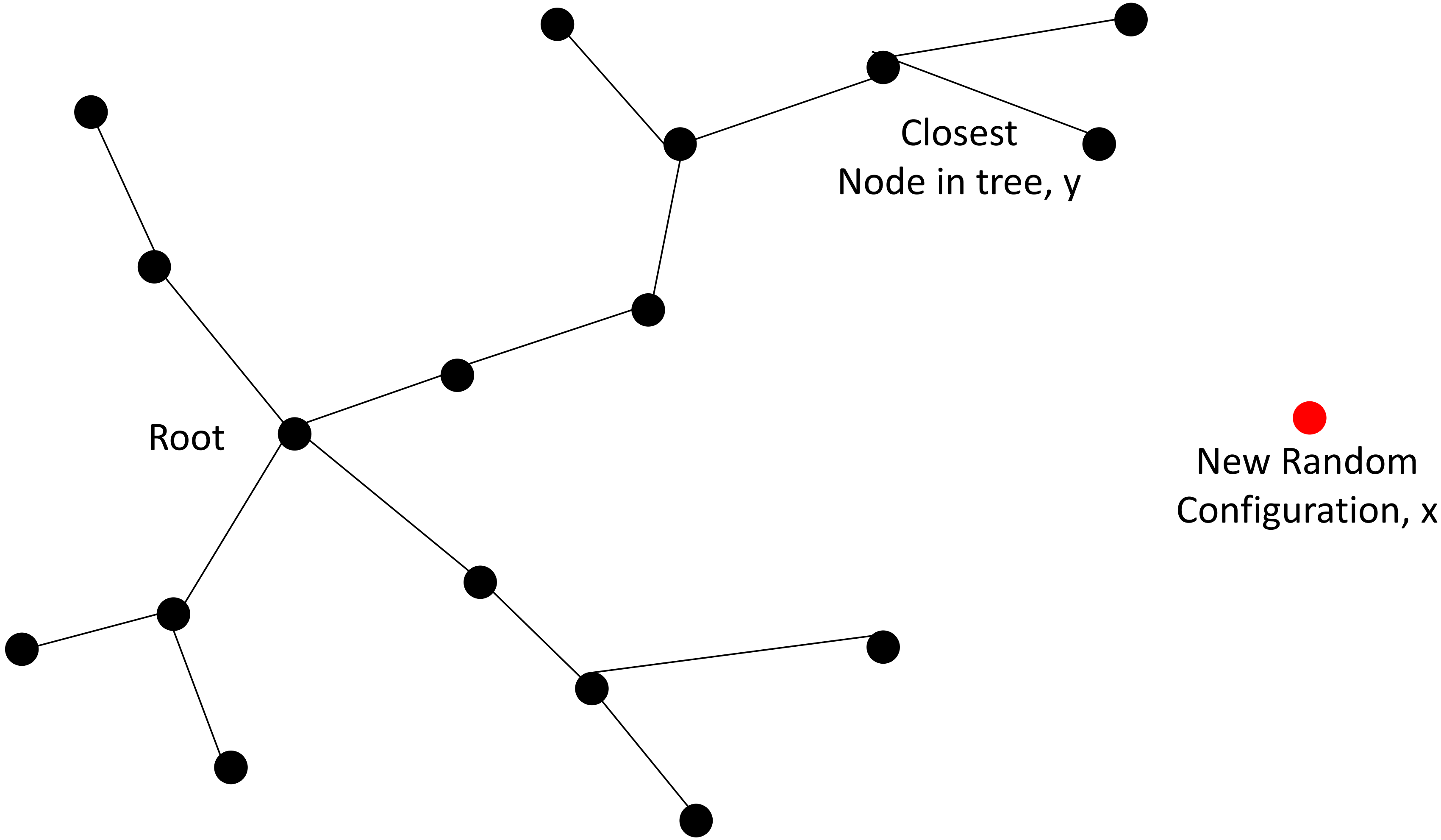
RRT Procedure

- Add start node to tree
- Repeat n times
  - Generate a random configuration, x
  - If x is in freespace using the CollisionCheck function
    - Find the closest node in the tree to the random configuration, y
    - If (Dist (x, y) < delta) – Check if x is too far from y
    - Find a configuration, z, that is along the path from x to y such that Dist(z,y) <= delta
    - x = z;
    - If (LocalPlanner (x,y)) – Check if you can get from x to y
    - Add x to the tree with y as its parent

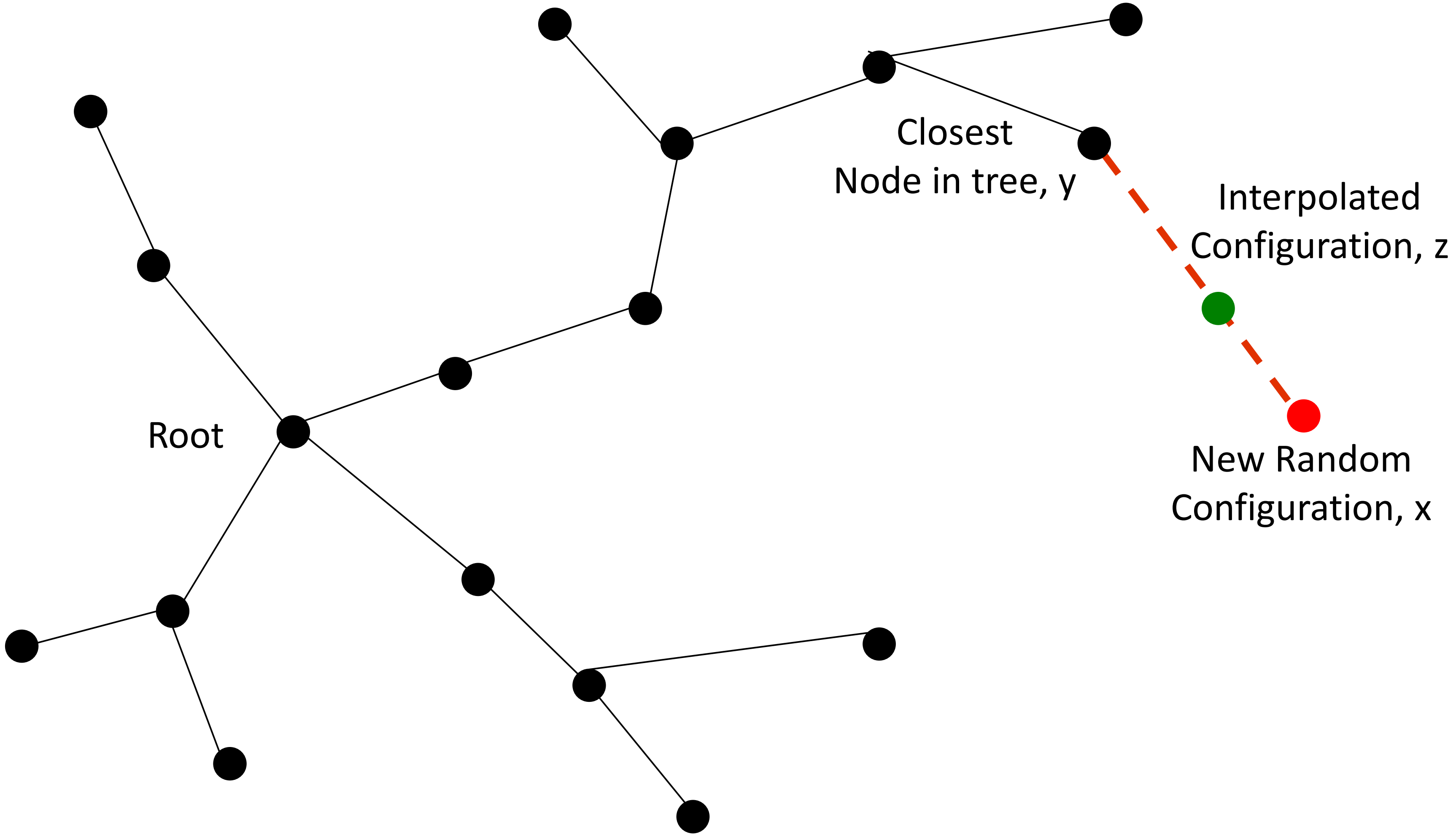
RRT Extension Procedure – Initial tree



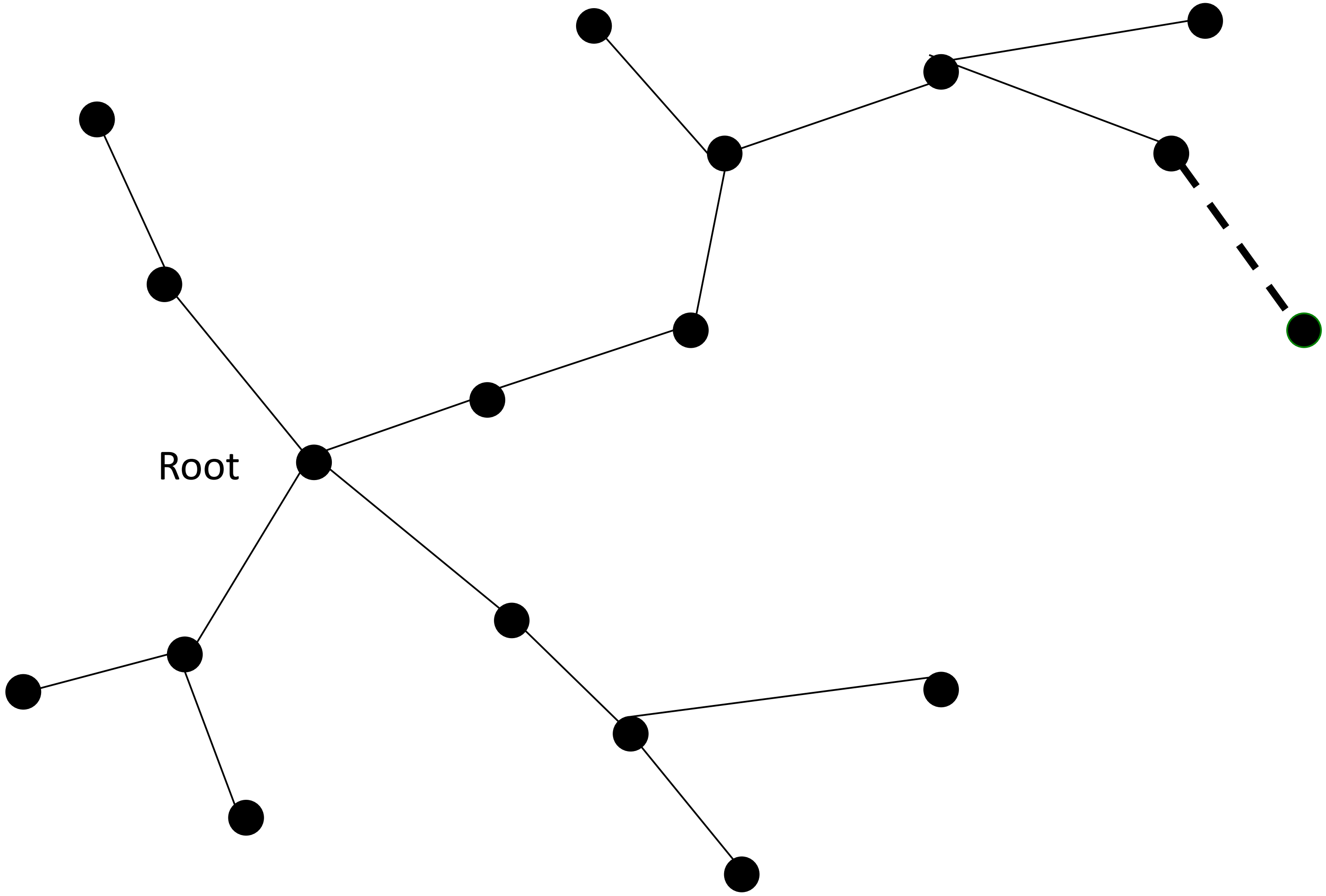
RRT Extension Procedure



RRT Extension Procedure



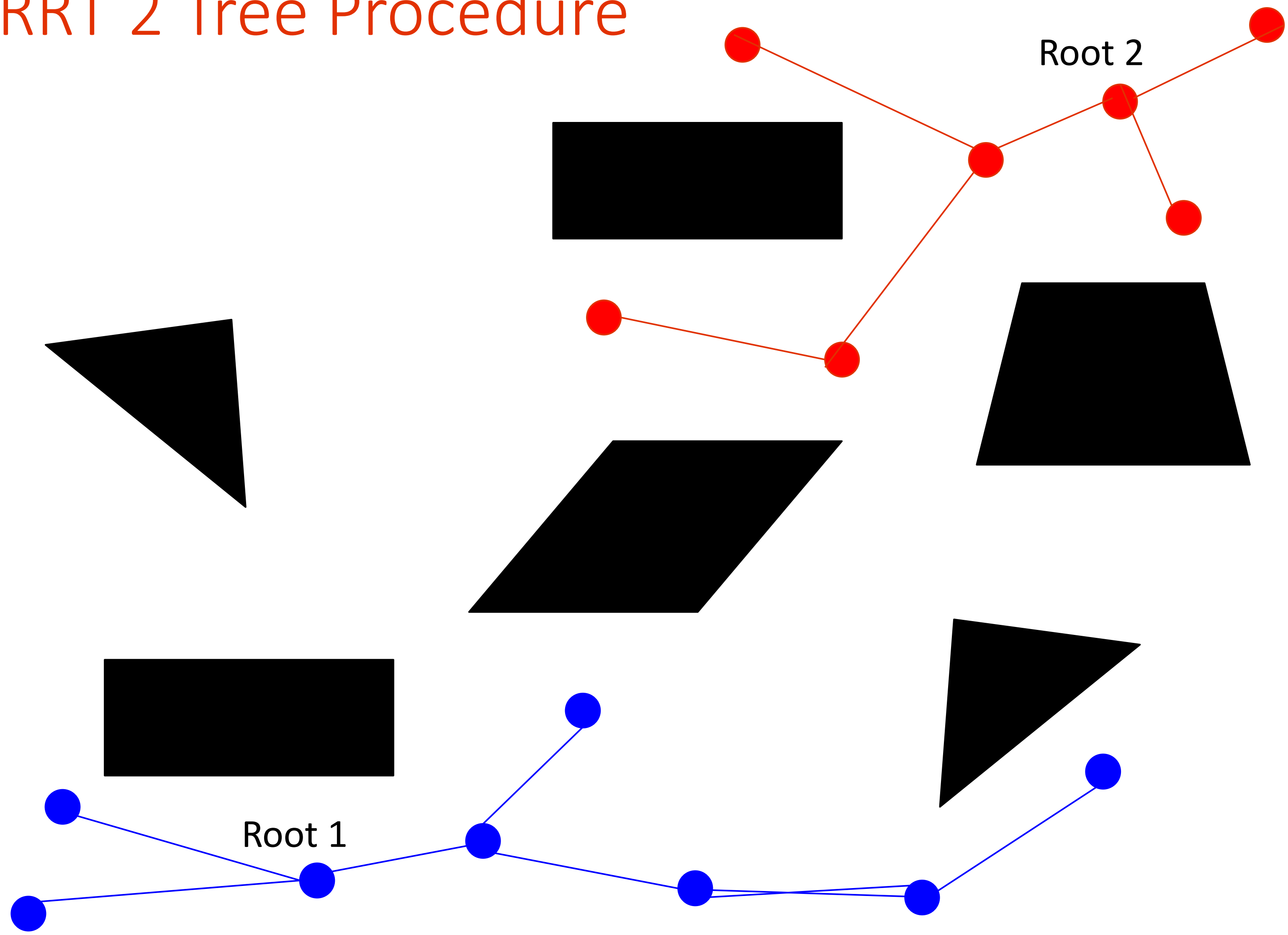
RRT Extension Procedure – Graph after extension



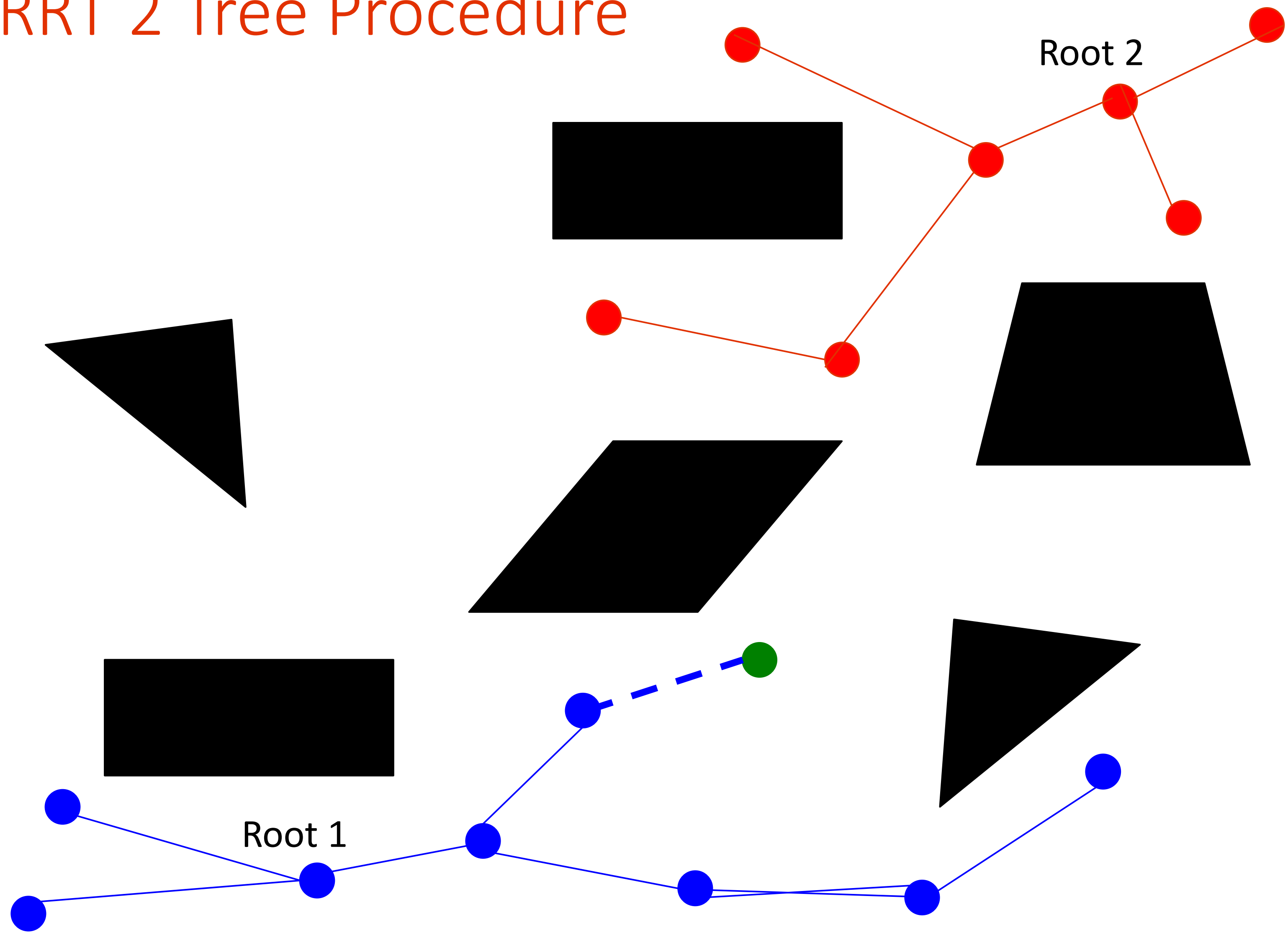
RRT 2 tree procedure

- While not done
  - Extend Tree A by adding a new node, x
  - Find the closest node in Tree B to x, y
  - If (LocalPlanner(x,y)) – Check if you can bridge the 2 trees
    - Add edge between x and y.
    - This completes a route between the root of Tree A and the root of Tree B. Return this route
    - Else
    - Swap Tree A and Tree B

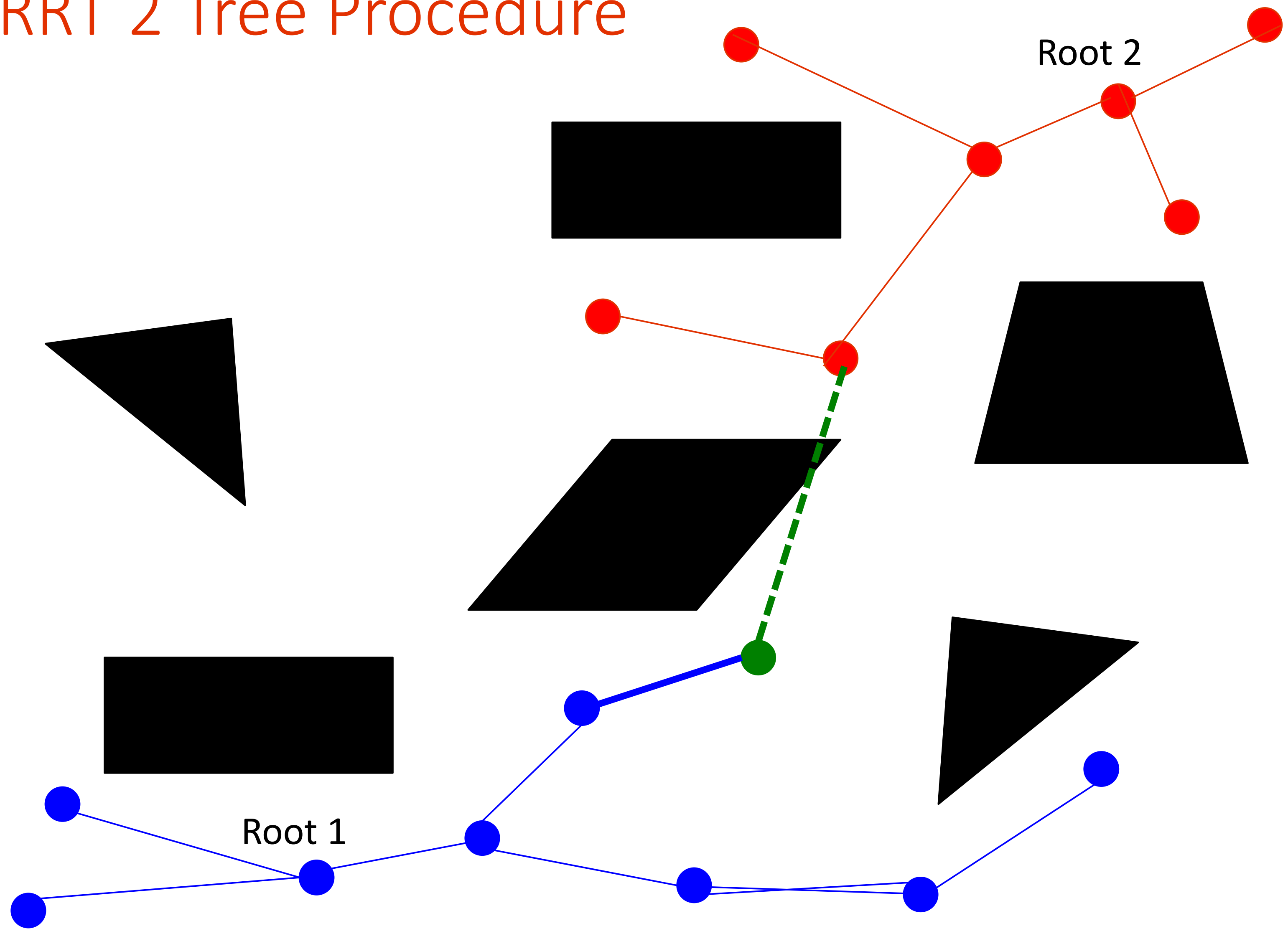
RRT 2 Tree Procedure



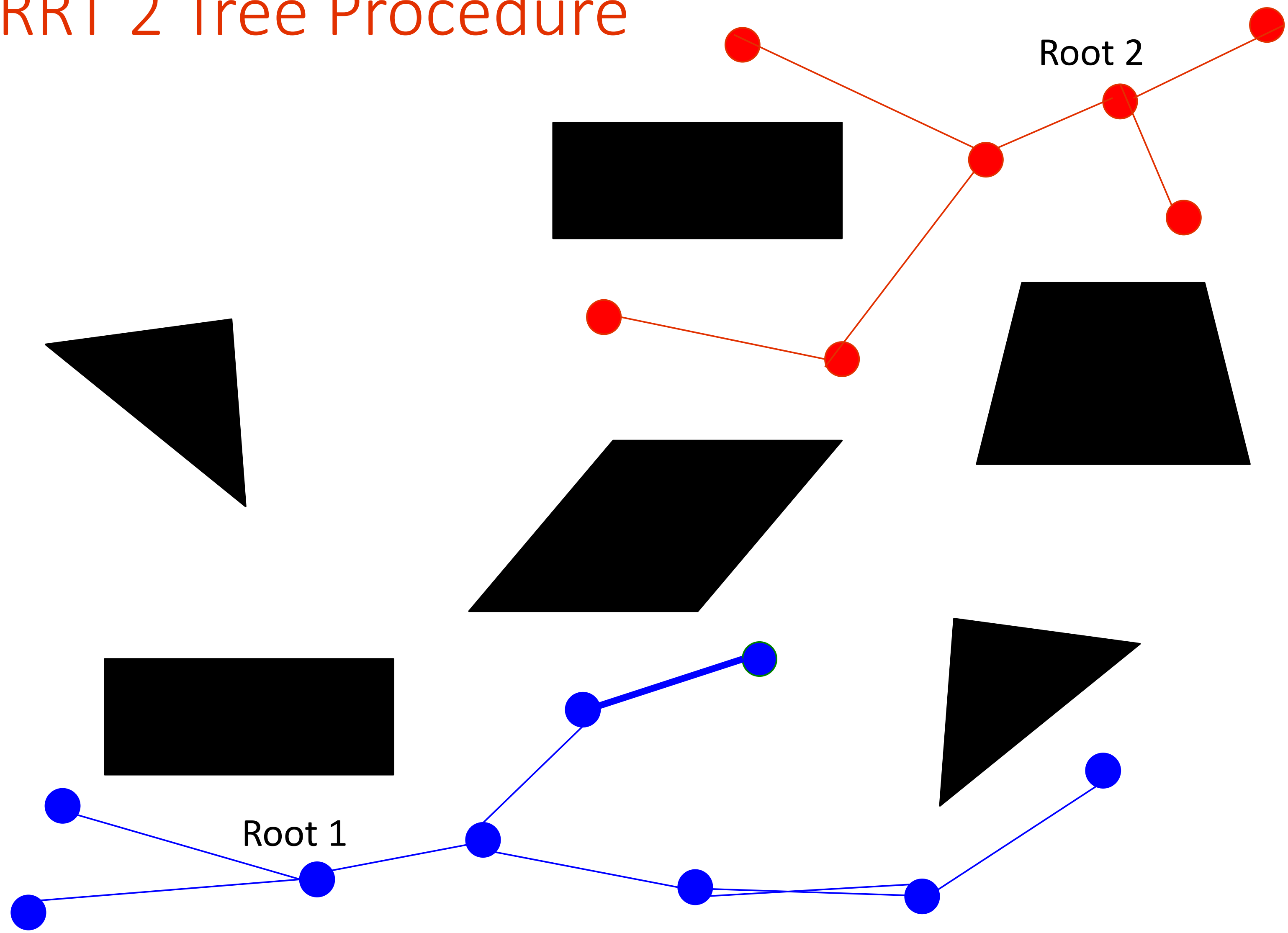
RRT 2 Tree Procedure



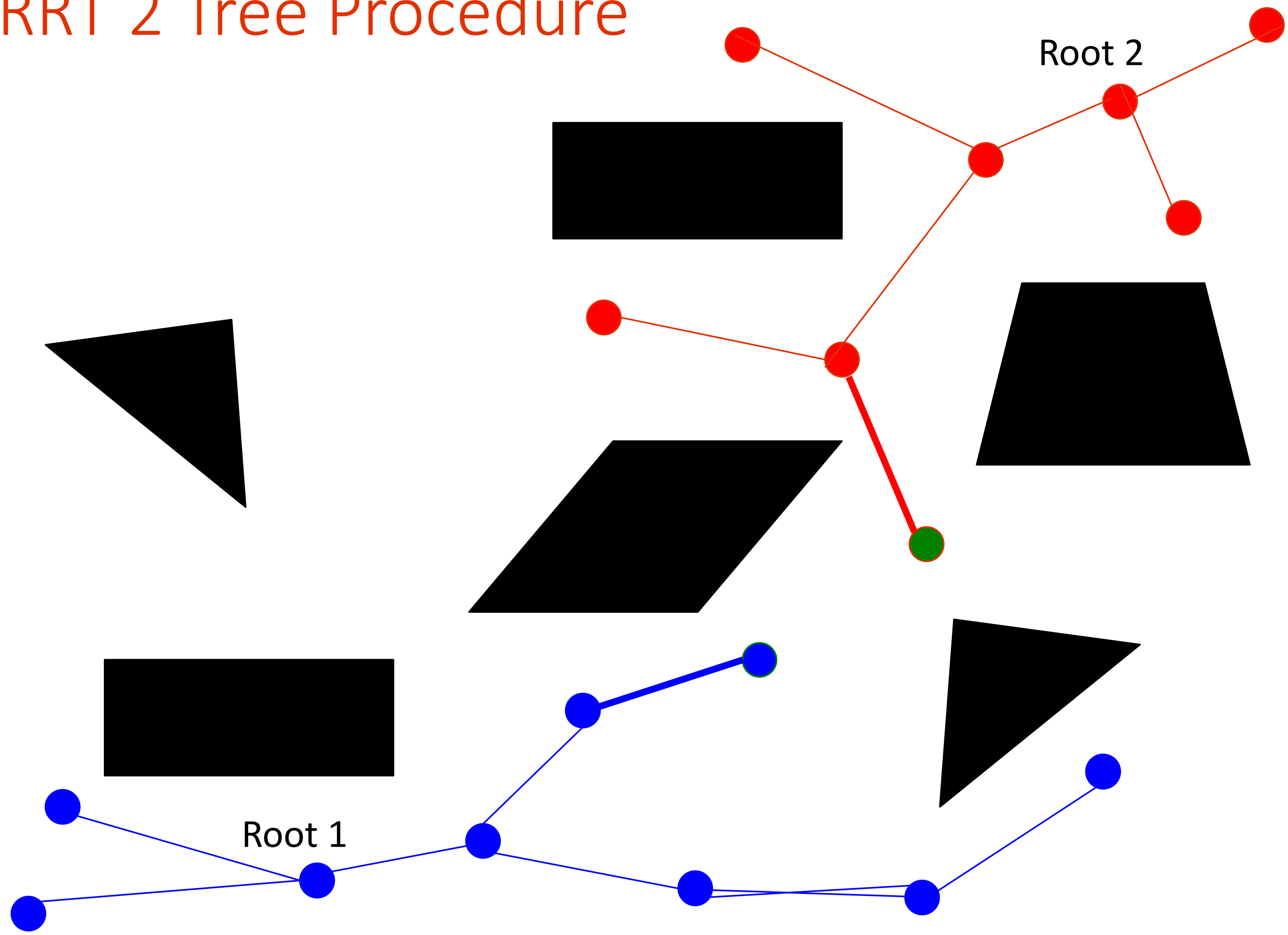
RRT 2 Tree Procedure



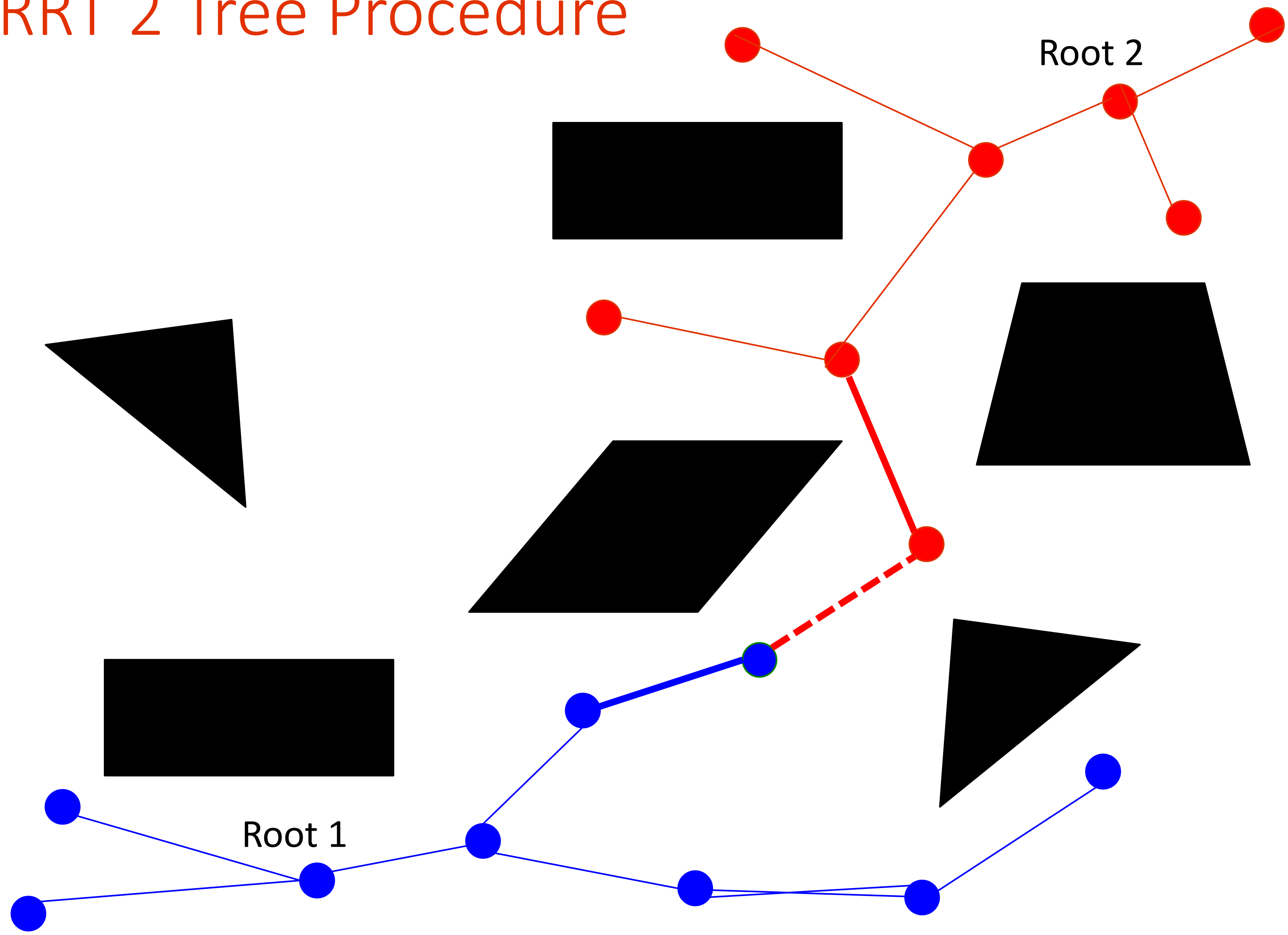
RRT 2 Tree Procedure



RRT 2 Tree Procedure



RRT 2 Tree Procedure



Video 11.4

CJ Taylor

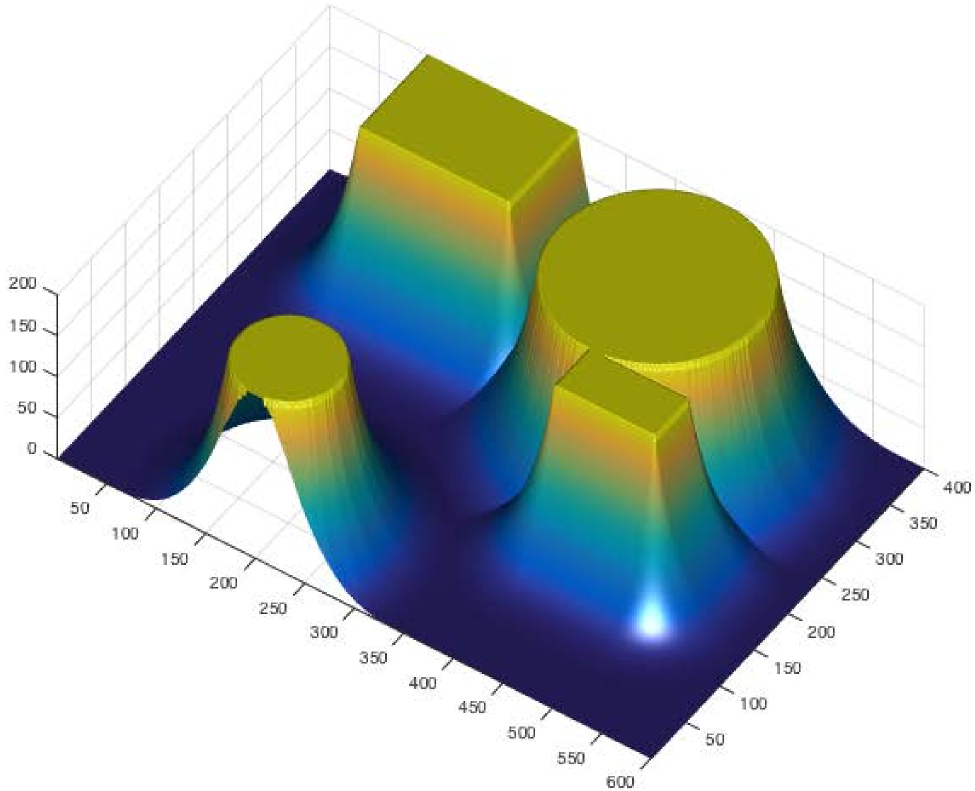
Constructing a Repulsive Potential Field

- A repulsive potential function in the plane, $f_r(\mathbf{x})$, can be constructed based on a function, $\rho(\mathbf{x})$, that returns the distance to the closest obstacle from a given point in configuration space, $\mathbf{x}$.

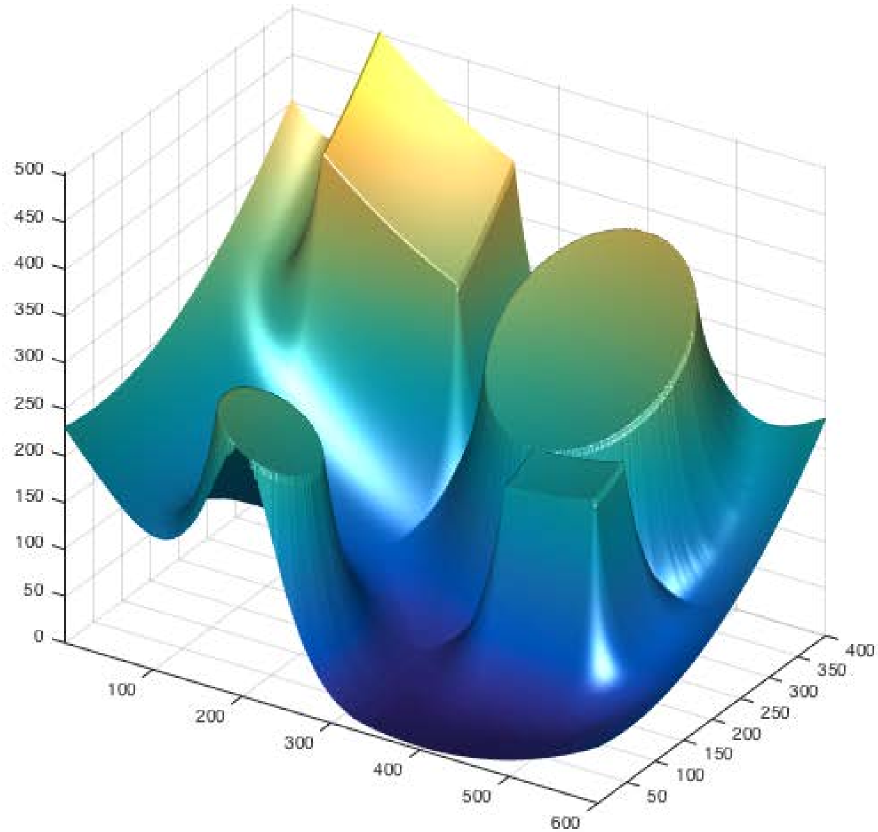
$$f_r(\mathbf{x}) = \begin{cases} \eta(\frac{1}{\rho(\mathbf{x})} - \frac{1}{d_0})^2 & \text{if } \rho(\mathbf{x}) \leq d_0 \\ 0 & \text{if } \rho(\mathbf{x}) > d_0 \end{cases}$$

- Here η is simply a constant scaling parameter and d_0 is a parameter that controls the influence of the repulsive potential

Visualizing the Repulsive Potential Field



Visualizing the Combined Potential field



Gradient Based Control Strategy

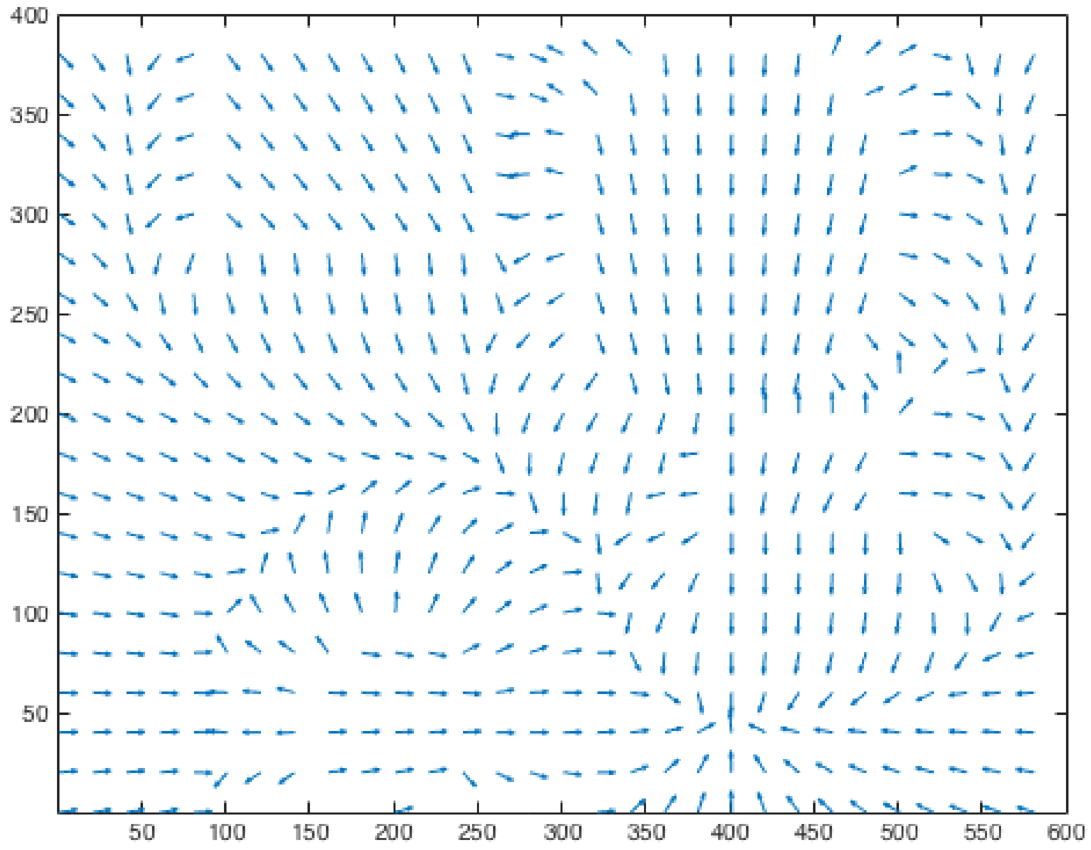
- While robot position is not close enough to goal
  - Choose direction of robot velocity based on the gradient of the artificial potential field:

$$\mathbf{v} \propto -\nabla f(\mathbf{x}) = -\begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \end{pmatrix} \tag{1}$$

  - Choose an appropriate robot speed, $\|\mathbf{v}\|$

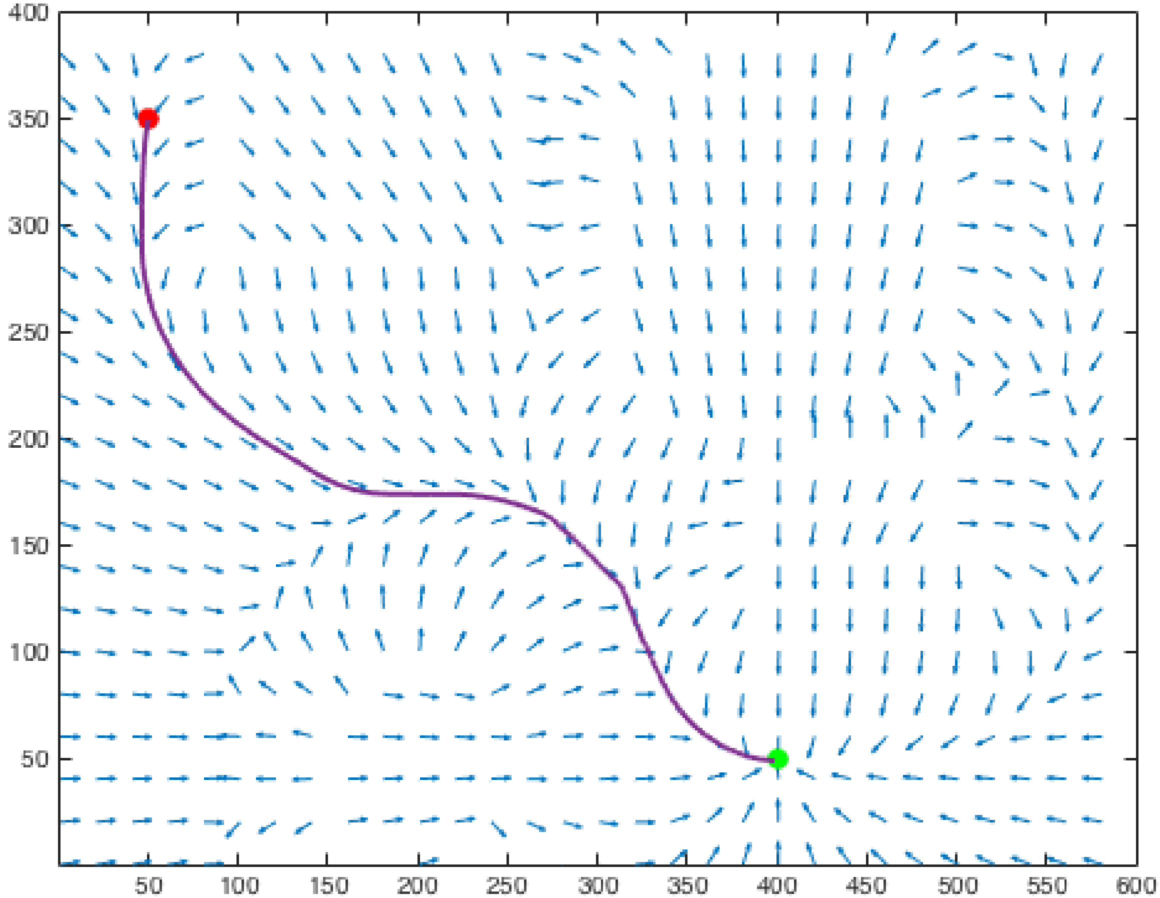
Quiver Plot

- The arrows in this figure denote the direction of the gradient vector at various points in the configuration space.



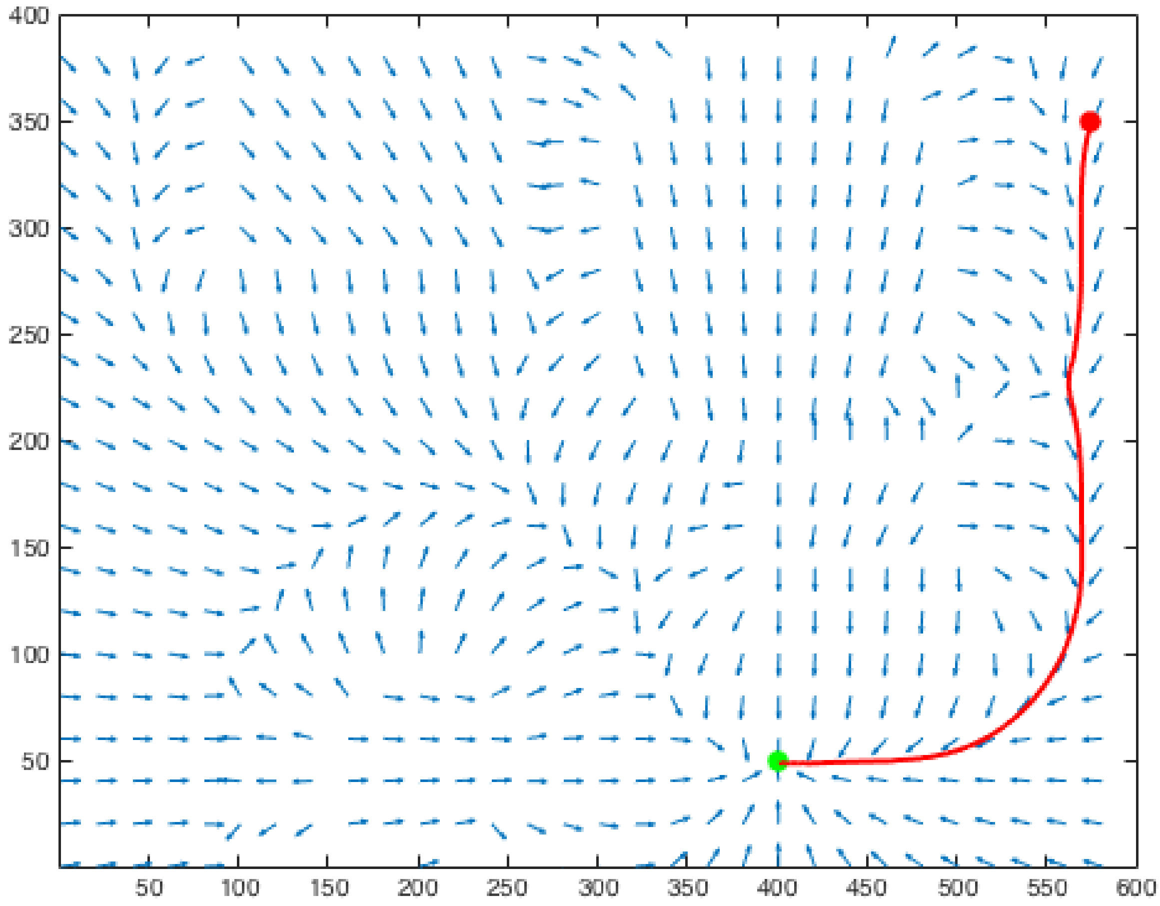
Trajectory Plot

- Example gradient based trajectory.



Trajectory Plot

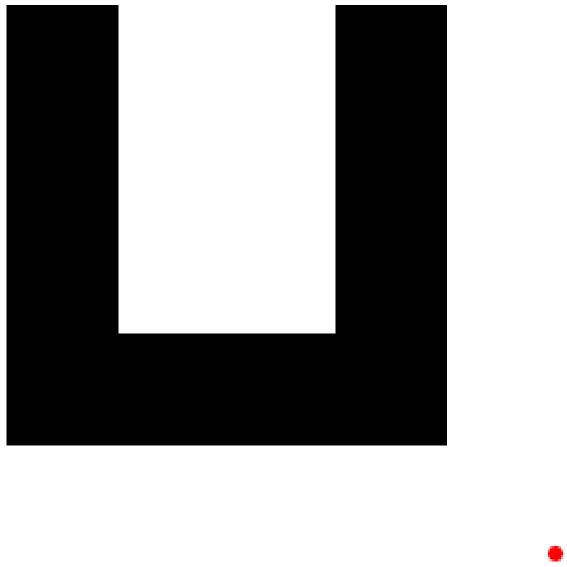
- Example gradient based trajectory.



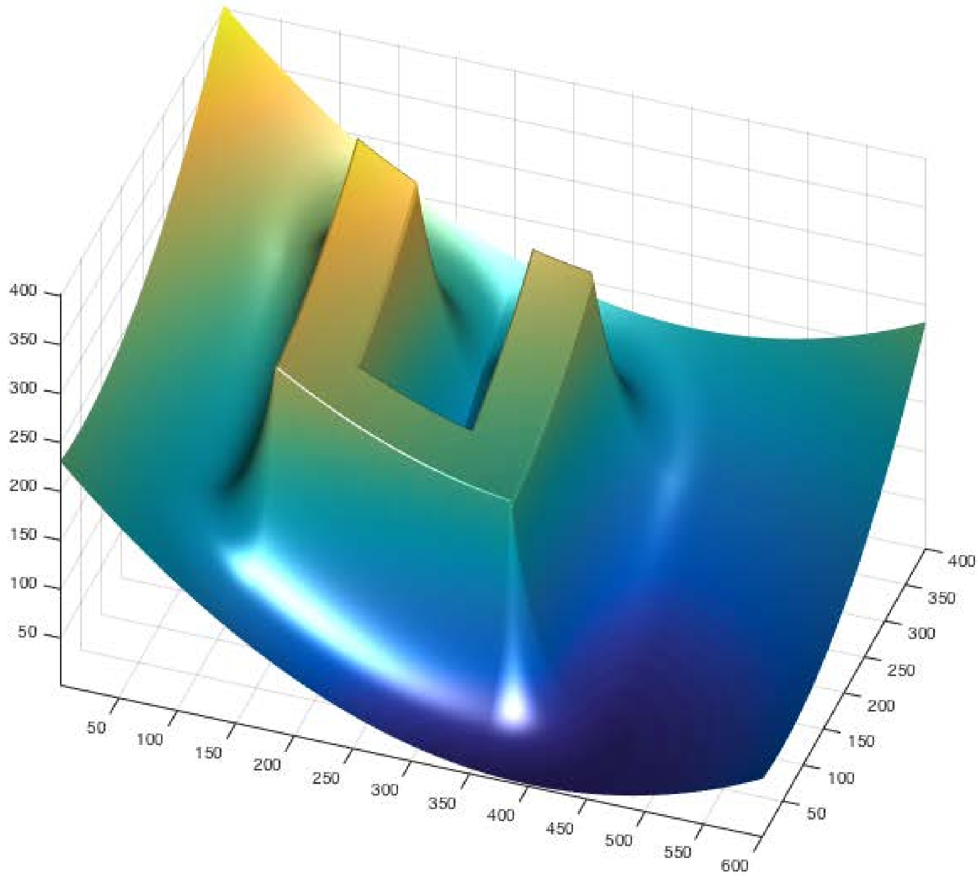
Video 11.5

CJ Taylor

Example Configuration Space



Artificial Potential Field



Generalizing Potential Fields

- One approach to generalizing artificial potential fields to more complicated robotic systems which can involve many degrees of freedom is by considering a set of control points distributed over the surface of the robot.
- The position of each of these control points can be computed as a function of the configuration space parameters, $P_i(\mathbf{x})$.
- For each of the control points we can construct an artificial potential field which repels it from obstacles and guides it to its desired location, $f_i(P_i(\mathbf{x}))$
- The final artificial potential function is computed by simply summing over all of the control points: $f(\mathbf{x}) = \sum_i f_i(P_i(\mathbf{x}))$.
- Once again we can construct a control law to move the robot by considering the gradient of the potential field with respect to the configuration space parameters.

$$\mathbf{v} \propto -\nabla f(\mathbf{x}) = - \begin{pmatrix} \frac{f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{f(\mathbf{x})}{\partial x_n} \end{pmatrix} \tag{1}$$