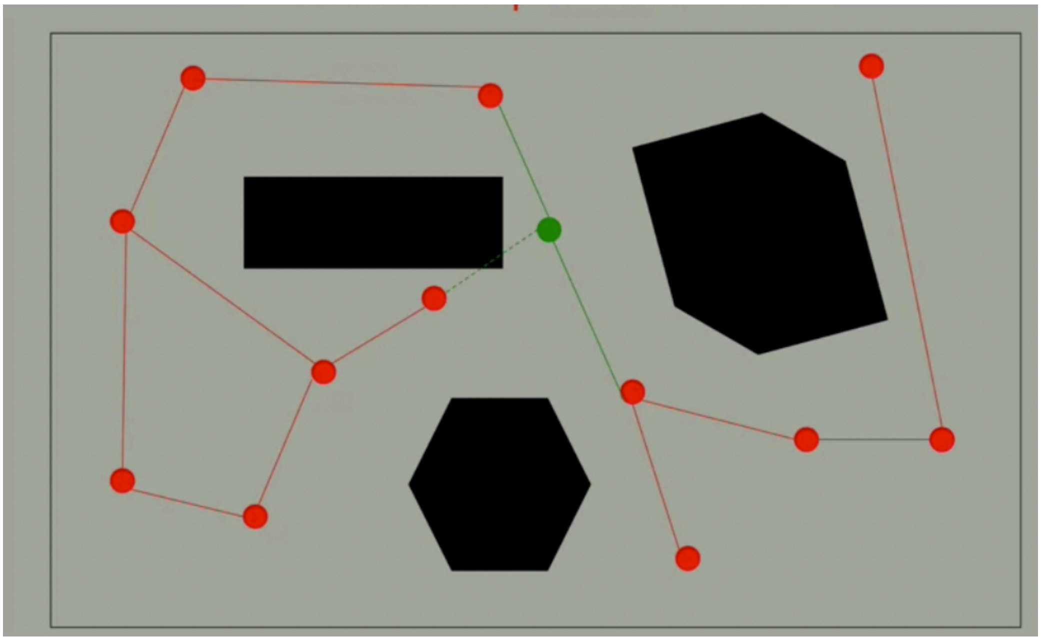
Robotics - Computational Motion Planning (Sampling-based)

Probabilistic Road Map (PRM)

- On every iteration, the system chooses a configuration in the configuration space **at random**
- And test whether it is in free space using the collision check function
    - 
    - Green Spot: the random node
    - Find route between this new node and the **closest** existing samples in the graph
    - Green solid line: the new links that are added
    - Dashed green lines: failure connection (due to collision check)

Pseudocode

- Repeat n times
    - Generate a random point in configuration space, x
    - If x is in free space
        - 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.

Goal: Construct a graph of configuration space points and edges that capture the underlying topology of the free space

Dist Function

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

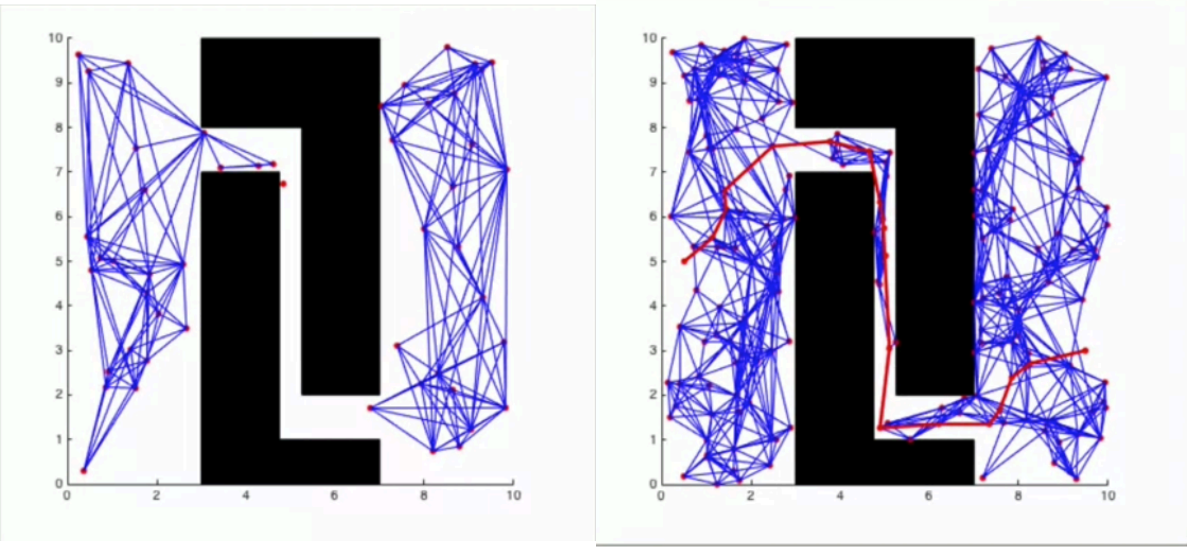
- Common choice for distance function include:
  - The L1 distance : $Dist_1 = \sum_i |x_i - y_i|$
  - The L2 distance : $Dist_2 = \sqrt{\sum_i (x_i - y_i)^2}$
  - Handling angular displacements: $Dist(\theta_1, \theta_2) = \min(|\theta_1 - \theta_2|, (360 - |\theta_1 - \theta_2|))$

LocalPlanner

- Decide whether two points have a path between them

NOTE

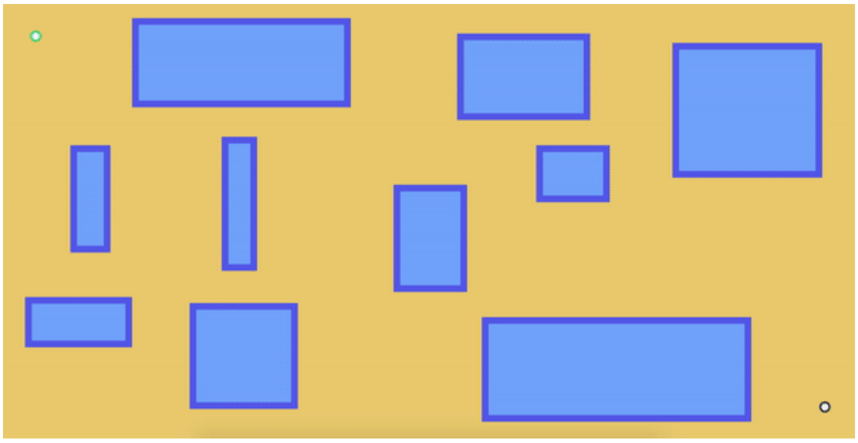
- May not find a path even it exists (Or take a long time)



- Can be applied to systems with a relatively high number of degrees of freedom

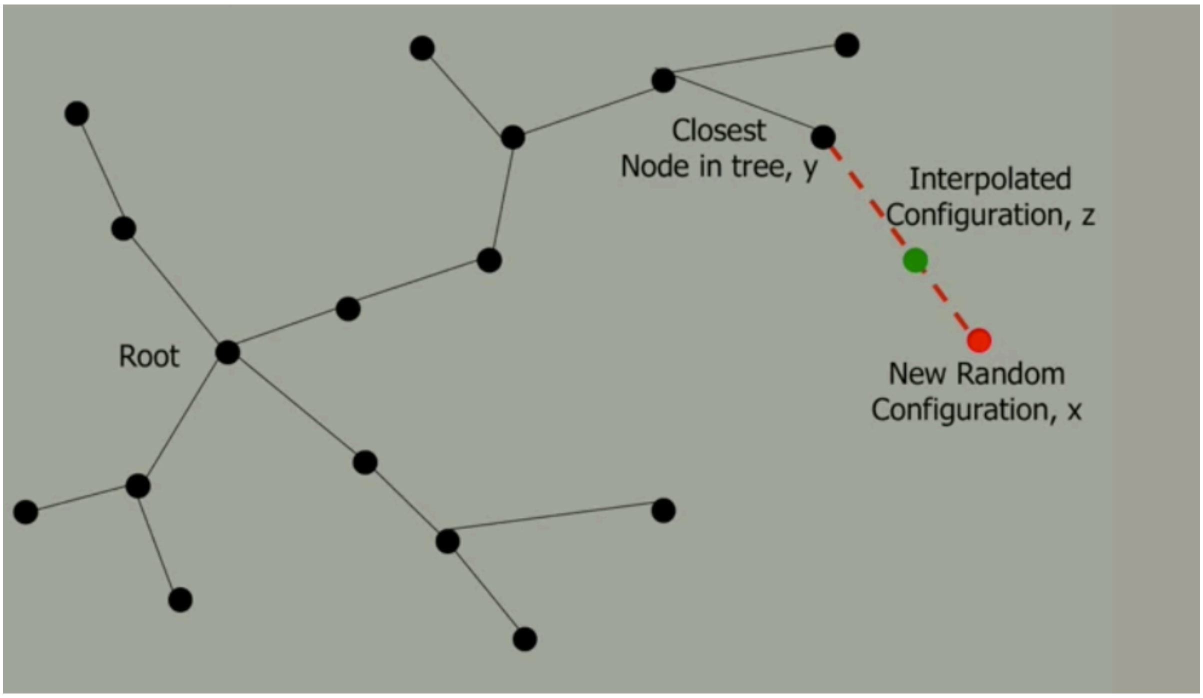
Rapidly Exploring Random Trees (RRT)

- Constructing **tree**, where every node is connected to a single parent and the tree is rooted at a given starting location



Pseudocode

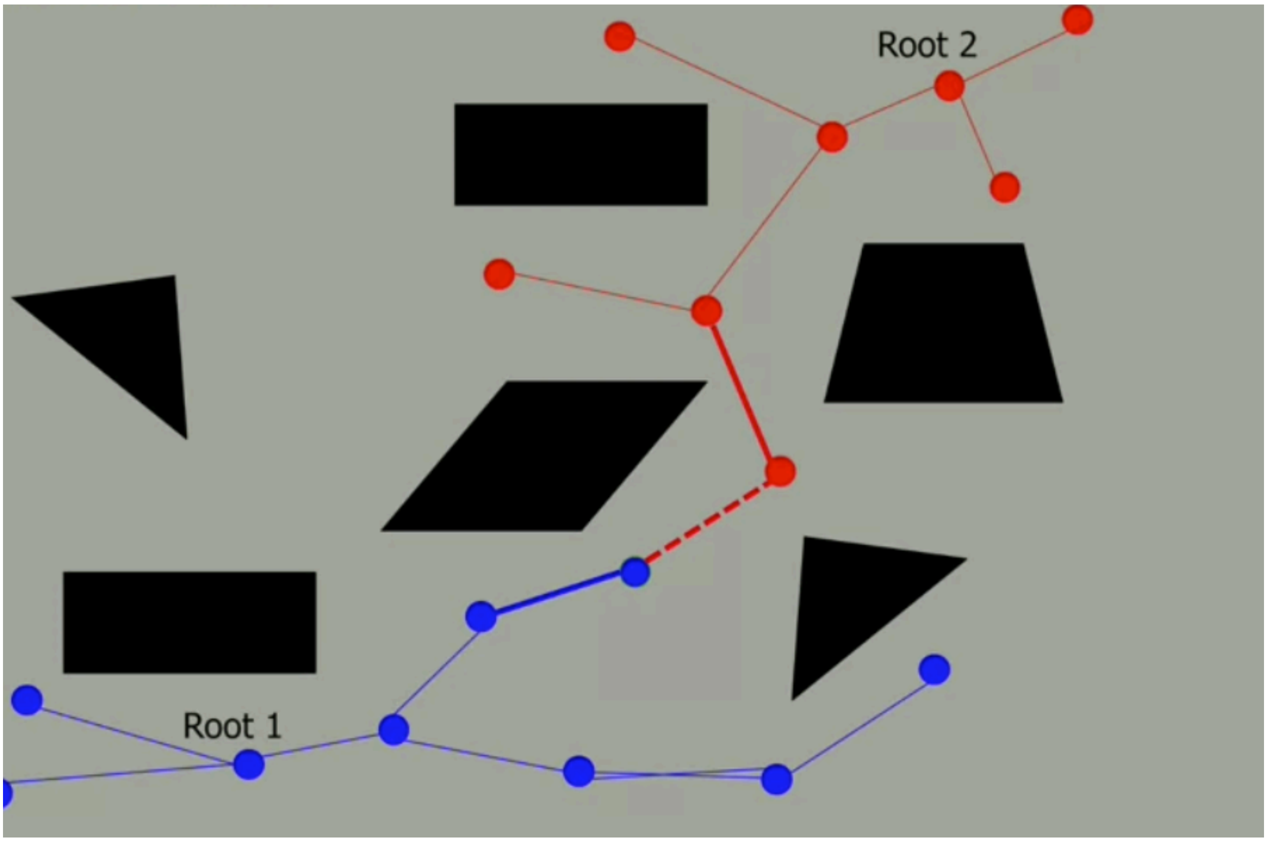
```
- Add start node to tree
- Repeat n times
    - Generate a random configuration, x
    - If x is in freespace using the CollisionCheck function
        - Find y, the closest node in the tree to the random configuration x
        - If (Dist (x,y) > delta) - check if x is too far from y
            - 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.
```



You can use two trees grow simultaneously (one roots from start point, another roots from end point)

Use two trees pseudocode

- 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 2 trees
        - Add edges 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 (grow together)



-