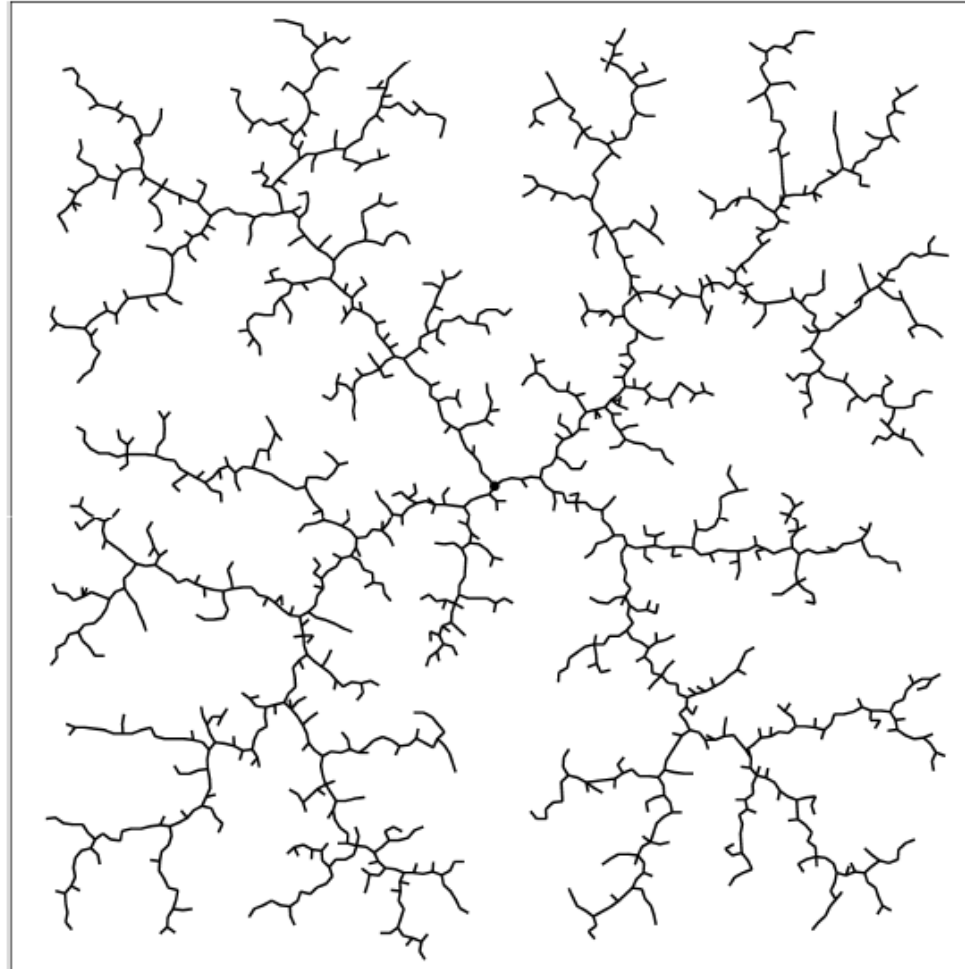


# Rapidly-Exploring Random Trees (RRTs)



These slides contain material aggregated/developed by Howie Choset (and others)

# Basic RRT Algorithm (no goal)

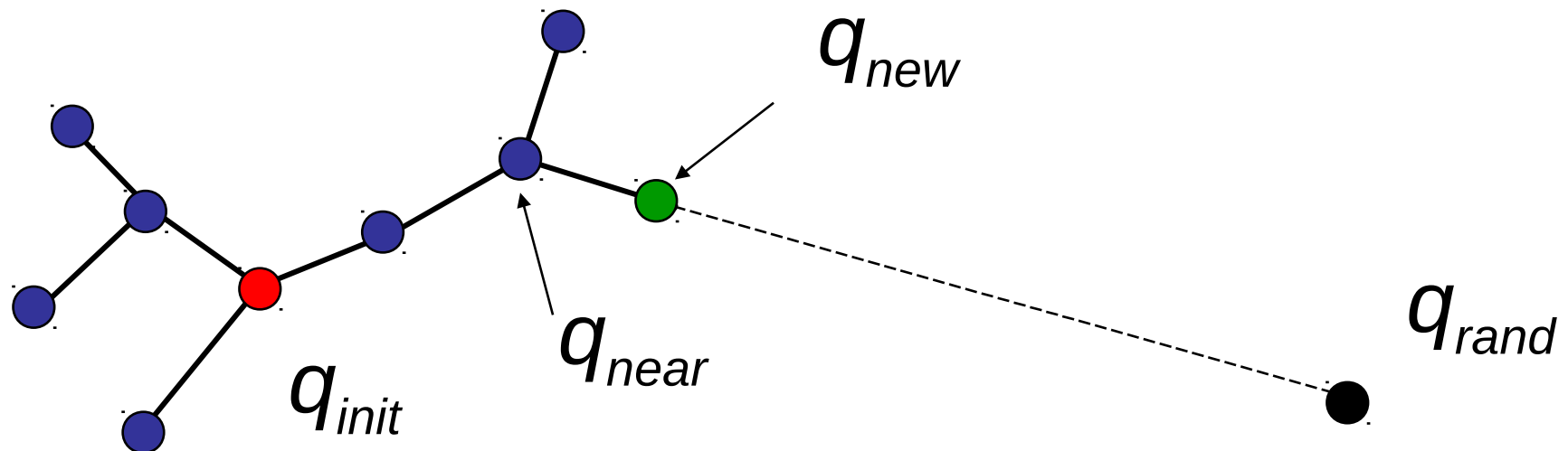
function RRT ( $q_{init}$ ):

$T = q_{init}$

for  $i = 1$  to  $K$  do:

$q_{rand} = \text{RANDOM\_CONFIG}();$

$T.\text{EXTEND}(q_{rand})$



# Basic RRT Algorithm (no goal)

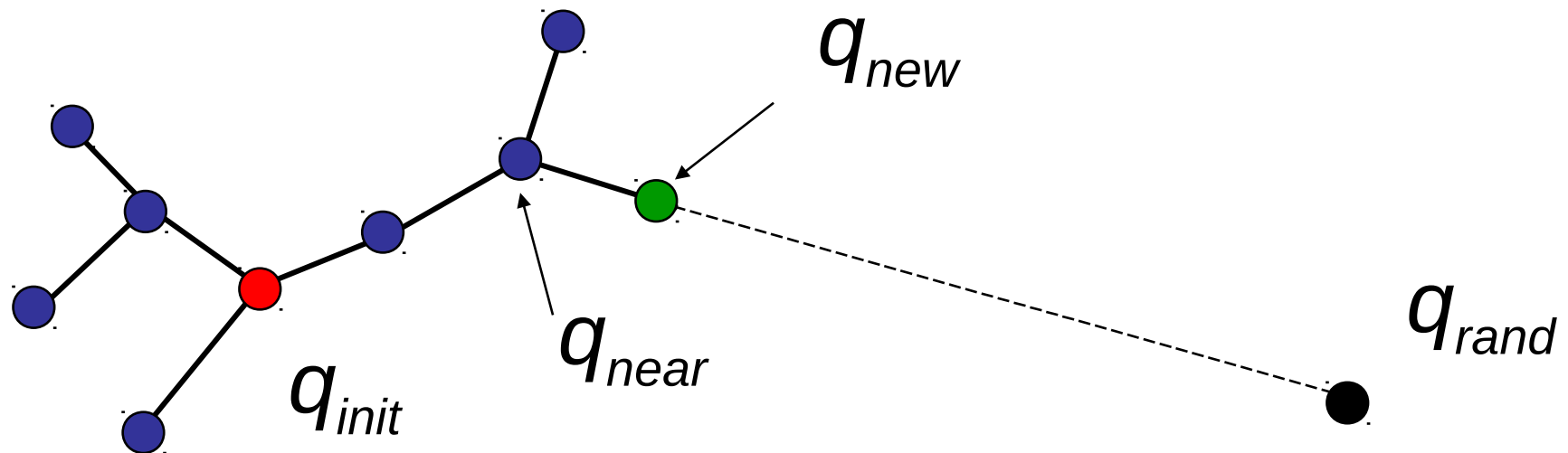
function RRT ( $q_{init}$ ):

$T = q_{init}$

for  $i = 1$  to  $K$  do:

$q_{rand} = \text{RANDOM\_CONFIG}();$

$T.\text{EXTEND}(q_{rand})$



# Basic RRT Algorithm (no goal)

function RRT ( $q_{init}$ ):

$T = q_{init}$

for  $i = 1$  to  $K$  do:

$q_{rand} = \text{RANDOM\_CONFIG}();$

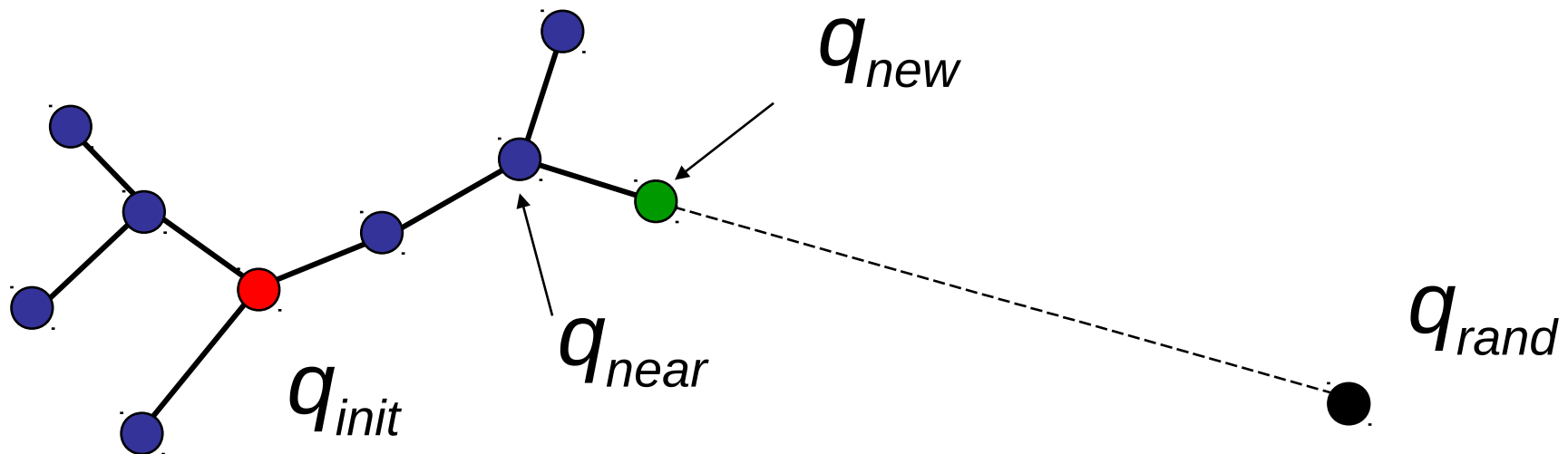
$T.\text{EXTEND}(q_{rand})$

STEP\_LENGTH: How far to sample

1. Sample just at end point
2. Sample all along
3. Small Step

Extend returns

1. Trapped, cant make it
2. Extended, steps toward node
3. Reached, connects to node



# Basic RRT Algorithm

```
function RRT ( $q_{init}, q_{goal}$ ):
```

```
   $T = q_{init}$ 
```

```
  for  $i = 1$  to  $K$  do:
```

```
    if  $\text{rand01}() < 0.1$ :
```

```
       $q_{rand} = q_{goal}$ 
```

```
    else
```

```
       $q_{rand} = \text{RANDOM\_CONFIG}();$ 
```

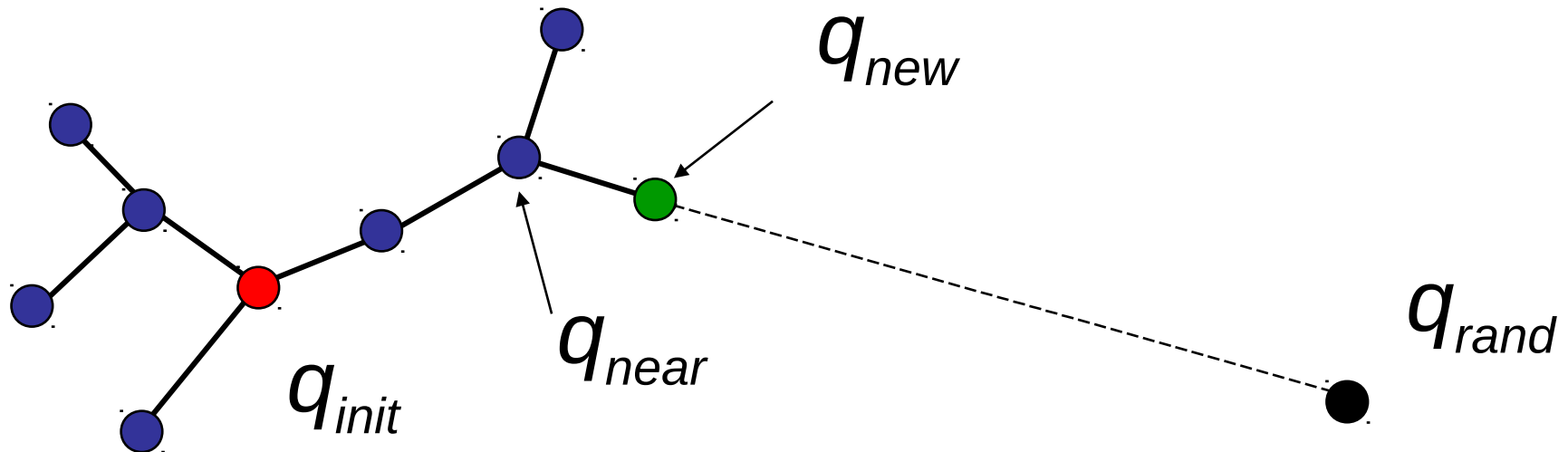
```
       $T.\text{EXTEND}(q_{rand})$ 
```

STEP\_LENGTH: How far to sample

1. Sample just at end point
2. Sample all along
3. Small Step

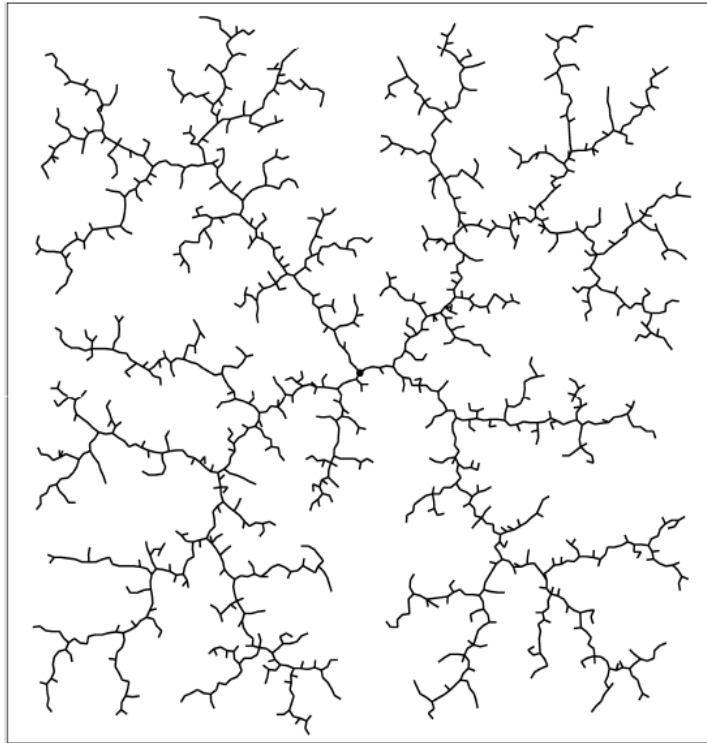
Extend returns

1. Trapped, cant make it
2. Extended, steps toward node
3. Reached, connects to node

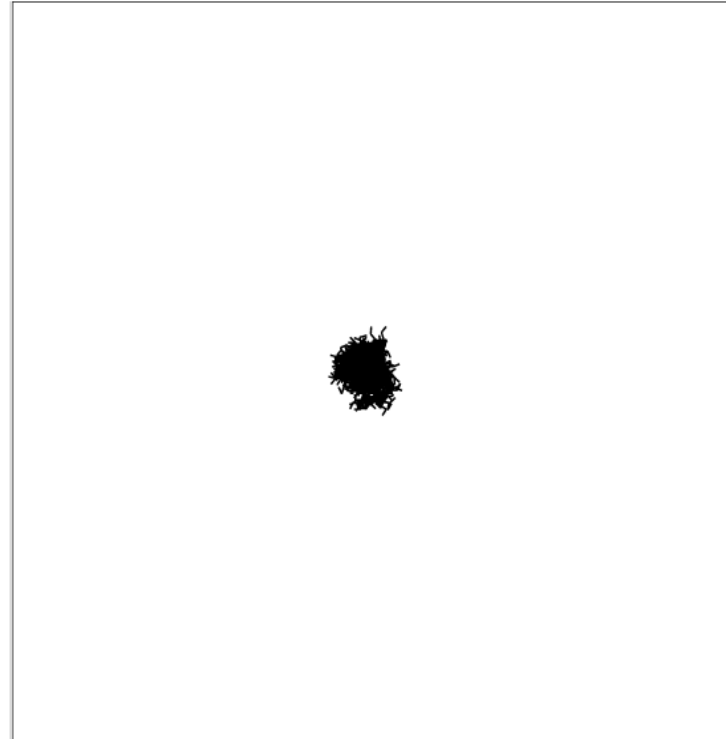


# RRT versus a naïve random tree

RRT



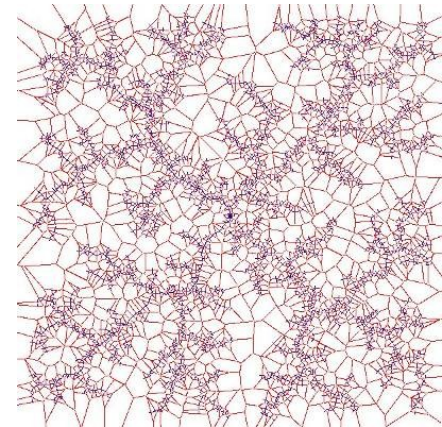
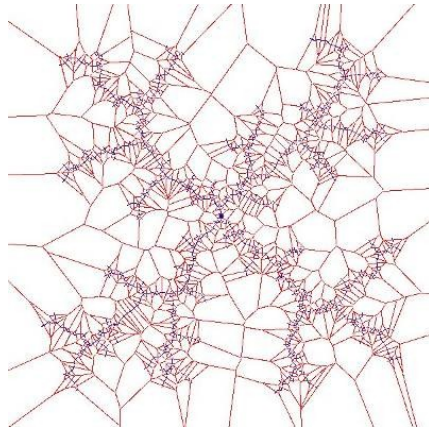
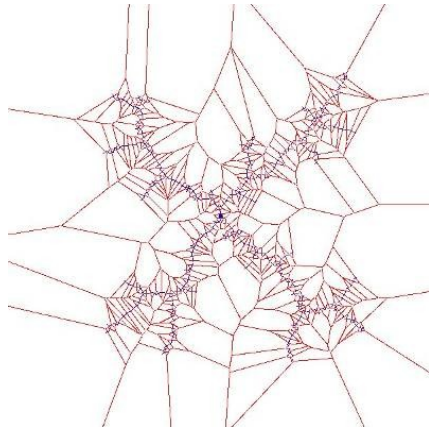
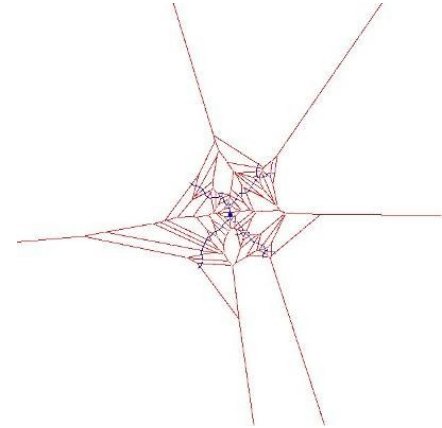
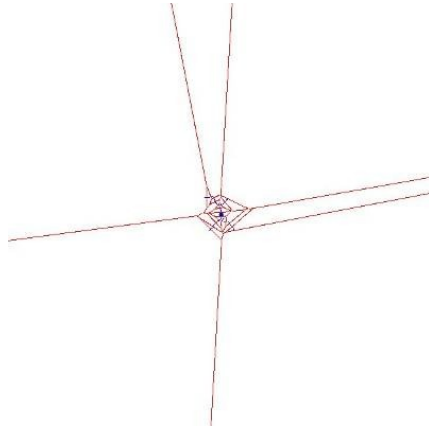
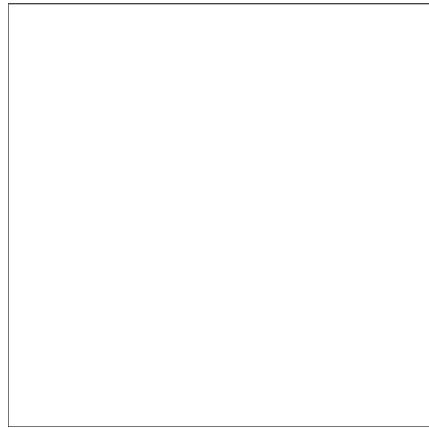
Naïve random tree



Growing the naïve random tree:

1. pick a node at random
2. sample a new node near it
3. grow tree from random node to new node

# RRTs and Bias toward large Voronoi regions



<http://msl.cs.uiuc.edu/rrt/gallery.html>

# Biases

- Bias toward larger spaces
- Bias toward goal
  - When generating a random sample, with some probability pick the goal instead of a random node when expanding
  - This introduces another parameter
  - 5-10% is probably the right choice



# RRT probabilistic completeness

Theorem (LaValle and Kuffner, 2001):

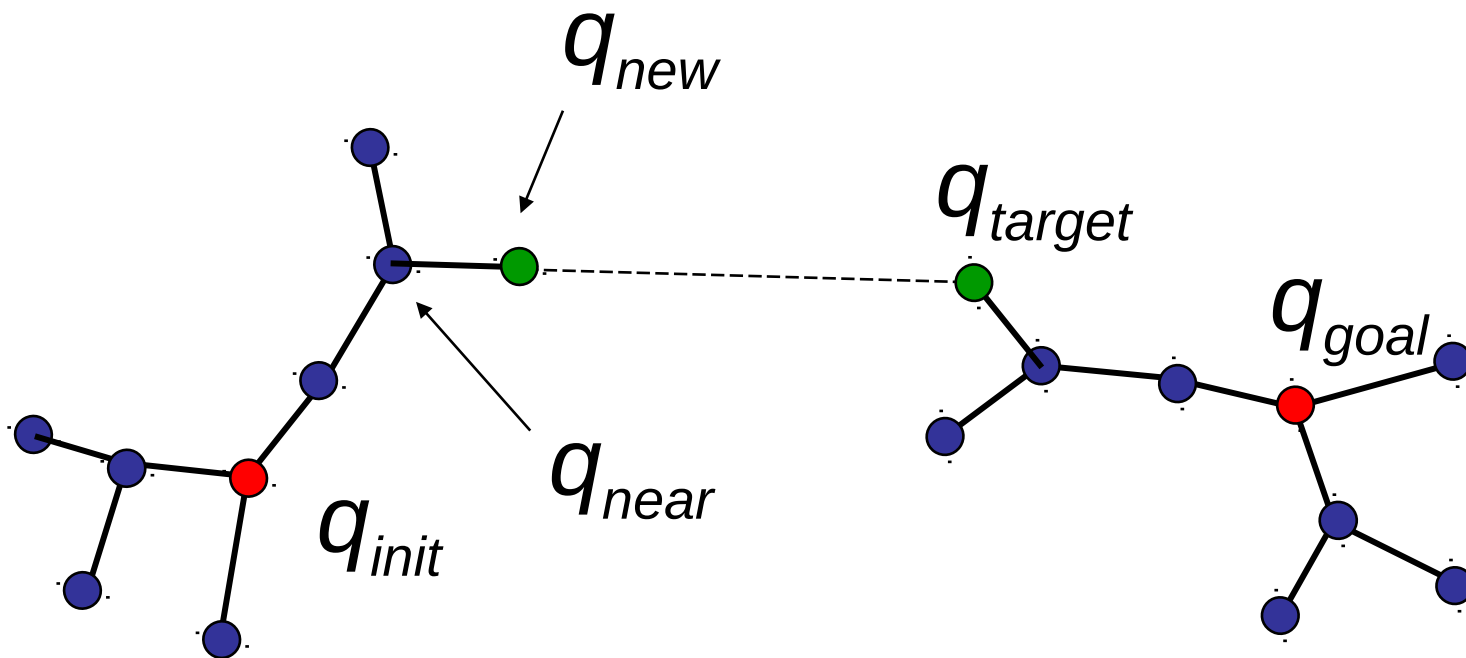
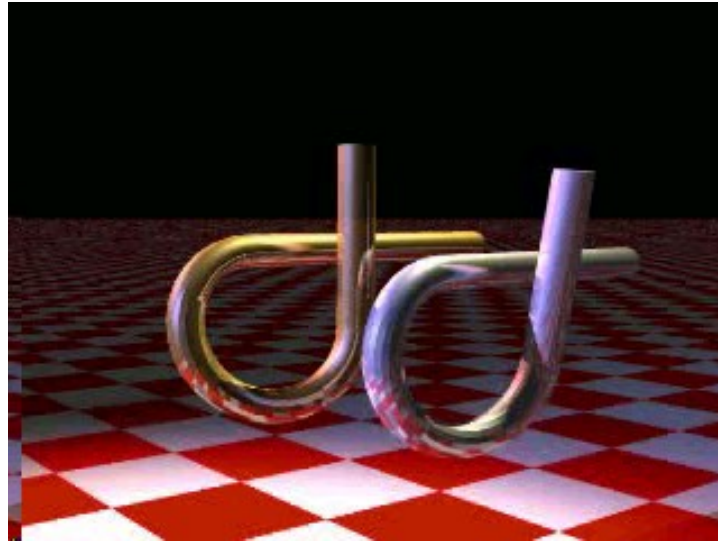
If a path planning problem is feasible, then there exist constants  $n_0$  and  $a > 0$ , such that:

$$P(\text{a path is found}) \geq 1 - e^{-an}$$

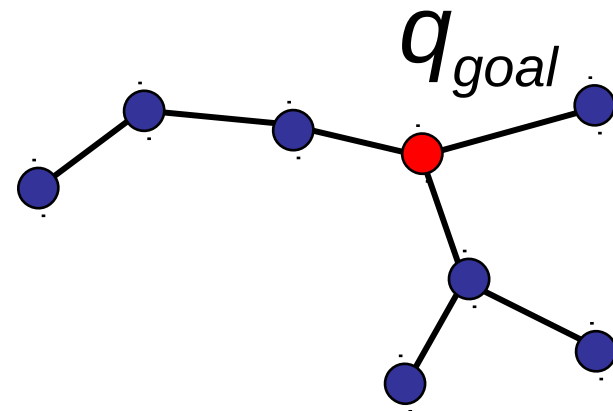
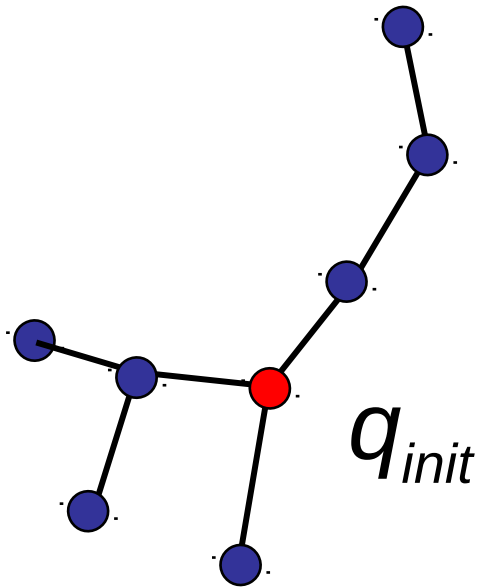
where  $n > n_0$  is the number of samples

Notice that this is exactly the same theorem as given for PRMs...

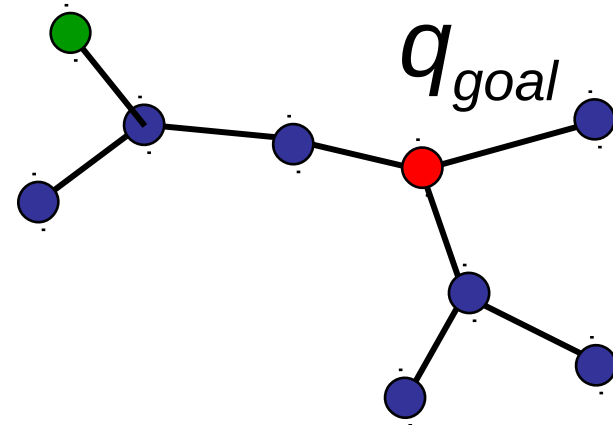
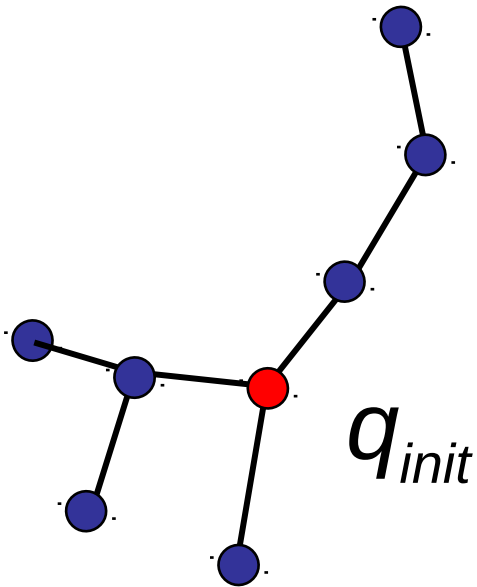
# RRT-Connect



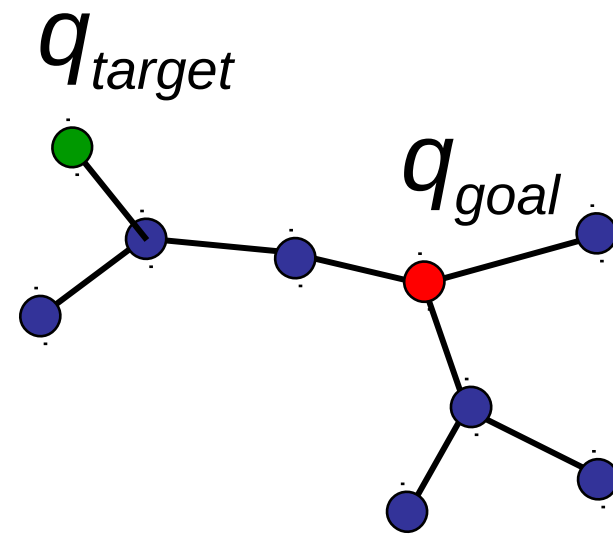
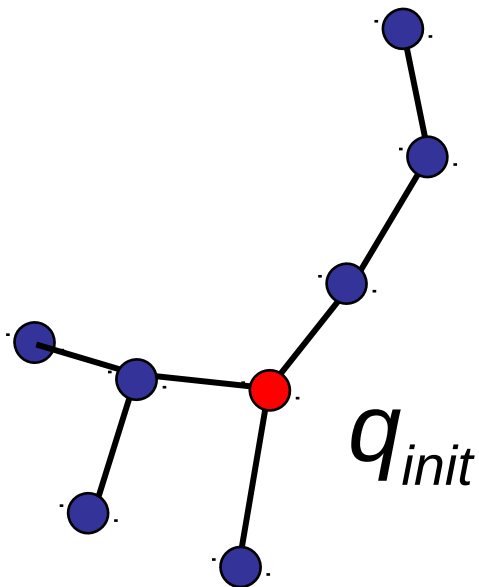
# A single RRT-Connect iteration...



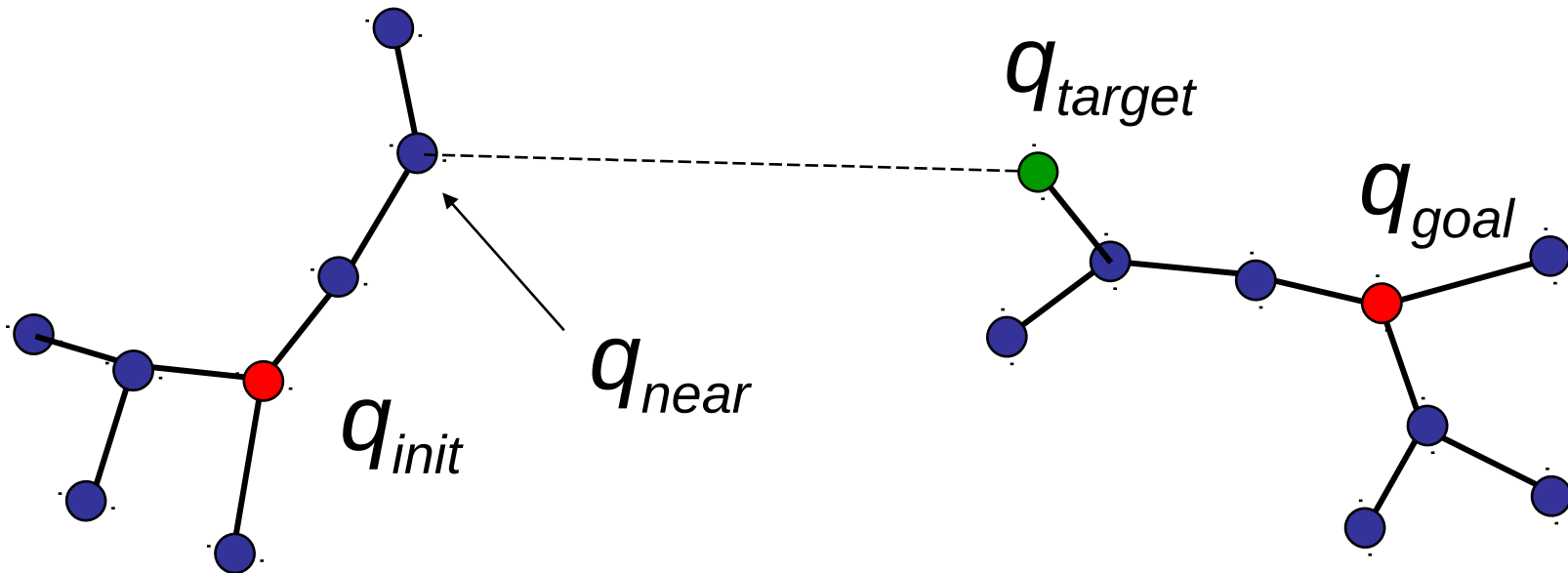
# 1) One tree grown using random target



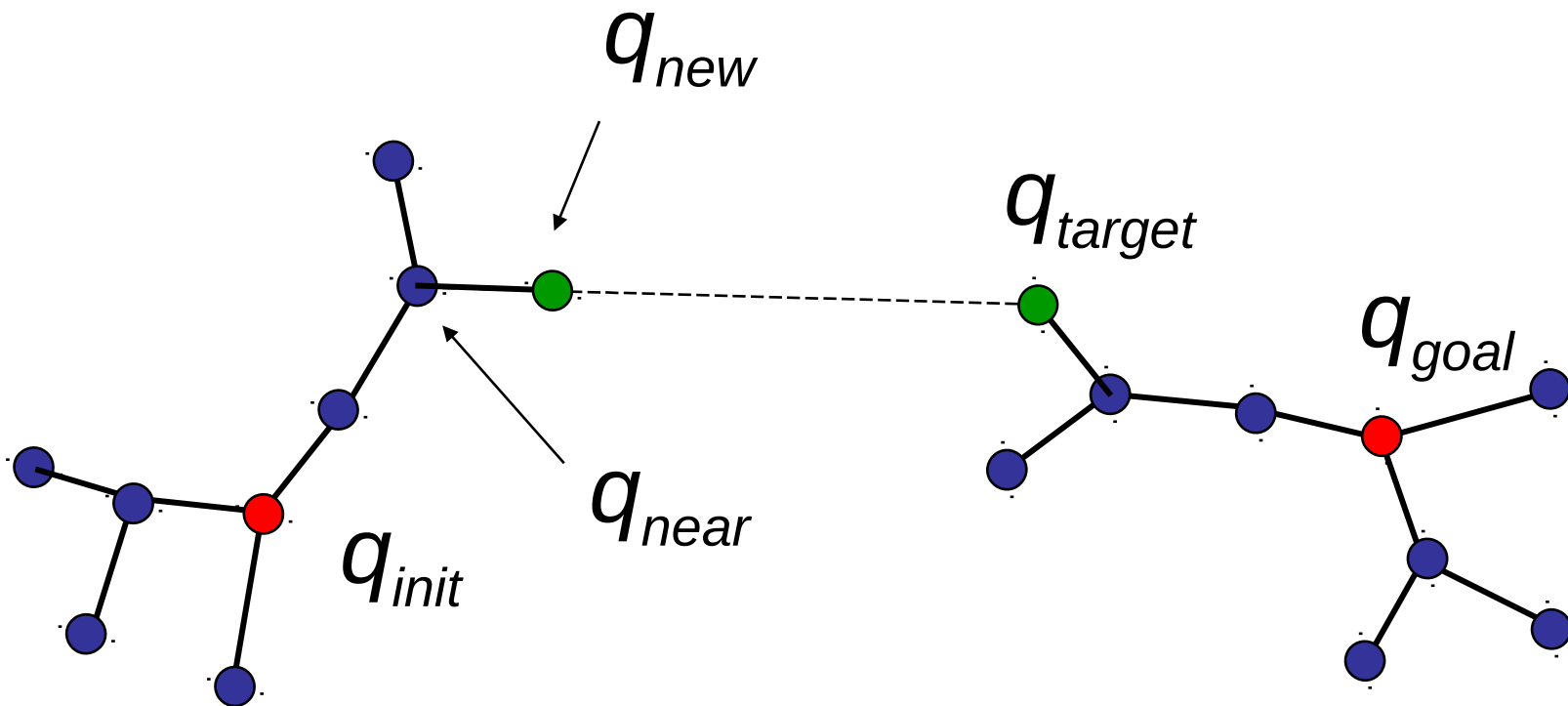
## 2) New node becomes target for other tree



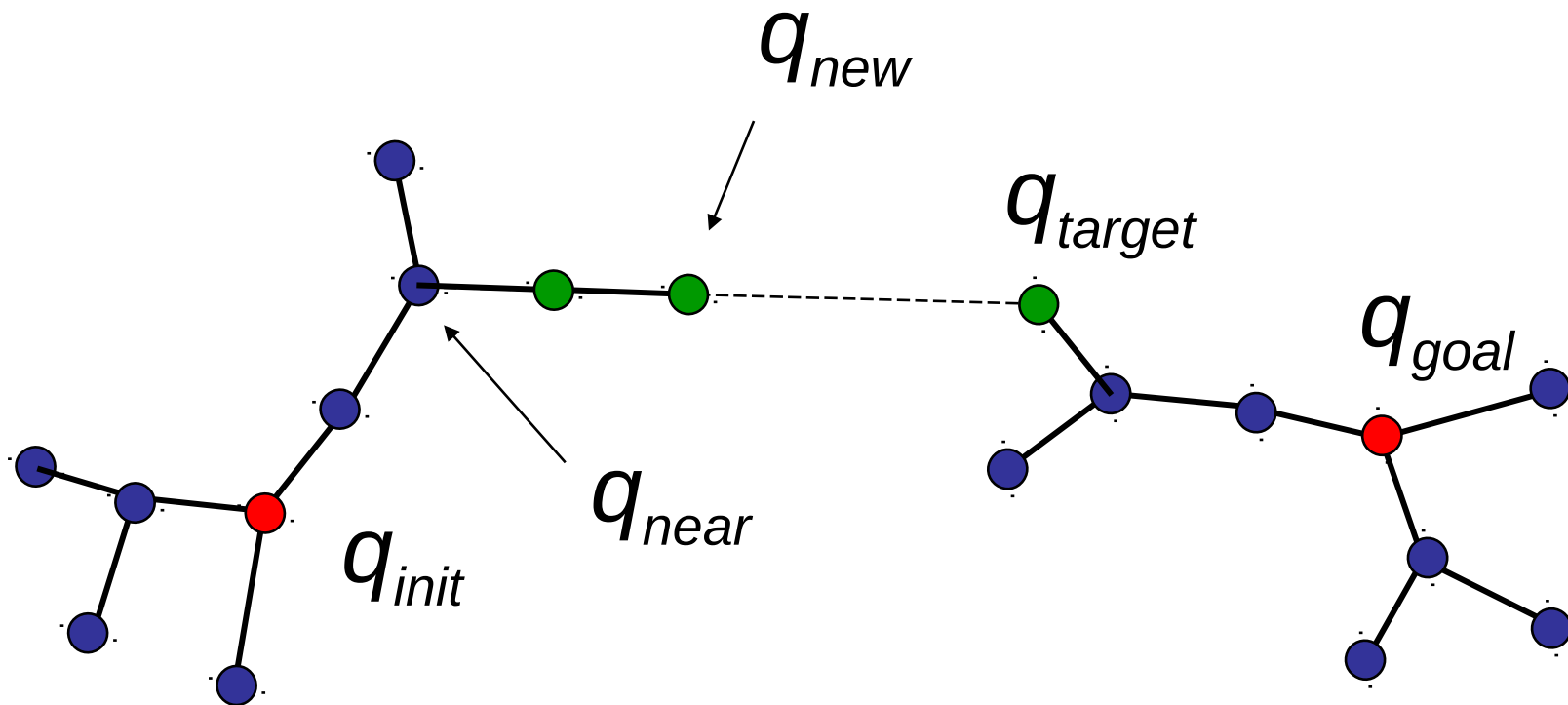
### 3) Calculate node "nearest" to target



## 4) Try to add new collision-free branch

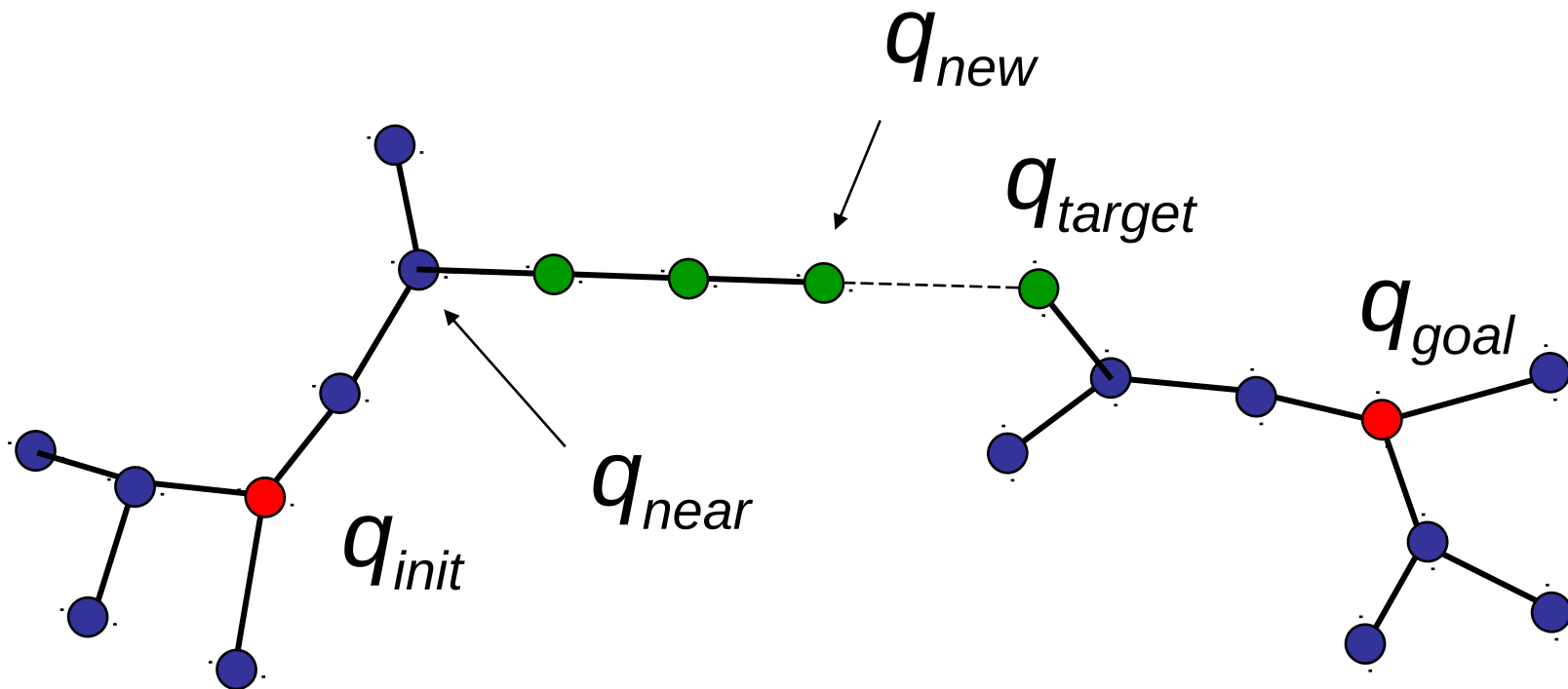


5) If successful, keep extending branch



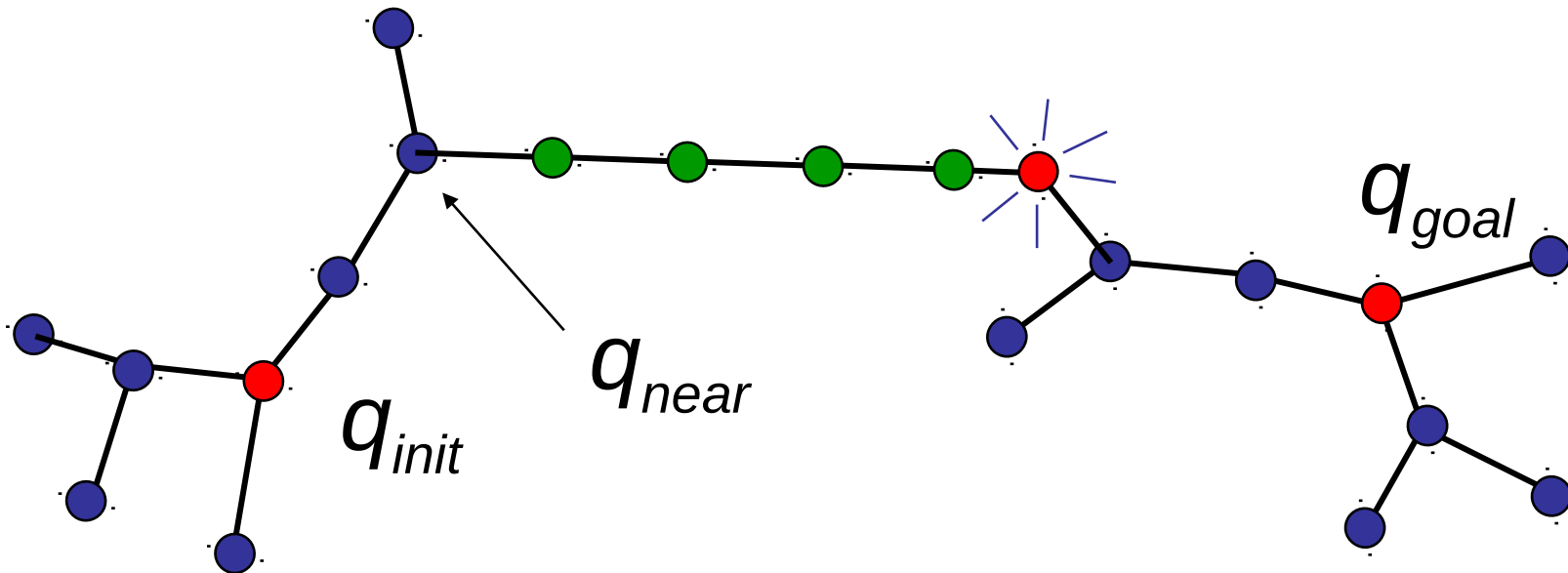


5) If successful, keep extending branch

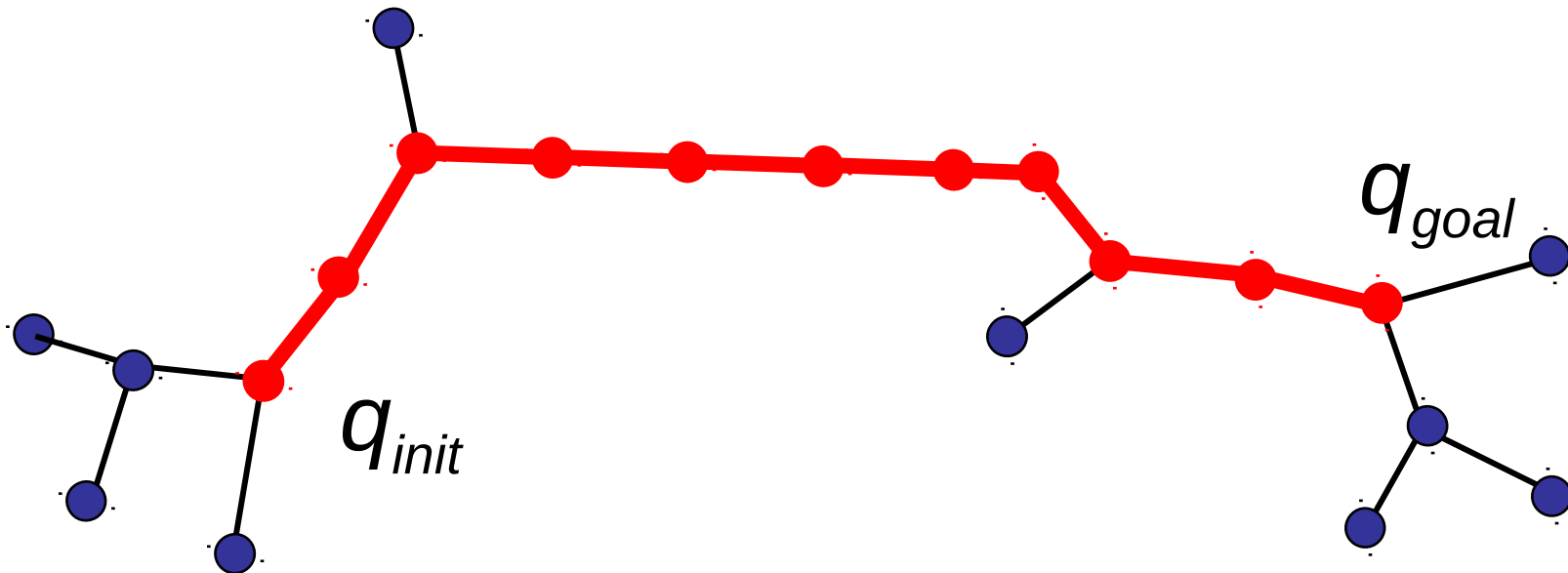




## 6) Path found if branch reaches target



## 7) Return path connecting start and goal



# Basic RRT-Connect

```
function RRT_CONNECT ( $q_{init}$ ,  $q_{goal}$ ):
```

```
   $T_a = q_{init}$ ;  $T_b = q_{goal}$ ;
```

```
  for  $i = 1$  to  $K$  do
```

```
     $q_{rand} = \text{RANDOM\_CONFIG}()$ ;
```

```
    if ( $T_a$ .EXTEND( $q_{rand}$ ) = extended) then
```

```
      if ( $T_b$ .EXTEND( $q_{new}$ ) = reached) then
```

```
        Return PATH( $T_a$ ,  $T_b$ );
```

```
  SWAP( $T_a$ ,  $T_b$ );
```

# Basic RRT-Connect

```
function RRT_CONNECT ( $q_{init}, q_{goal}$ ):
```

```
   $T_a = q_{init}; T_b = q_{goal};$ 
```

```
  for  $i = 1$  to  $K$  do
```

```
     $q_{rand} = \text{RANDOM\_CONFIG}();$ 
```

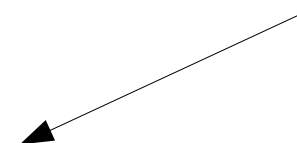
```
    if ( $T_a.\text{EXTEND}(q_{rand}) = \text{extended}$ ) then
```

```
      if ( $T_b.\text{EXTEND}(q_{new}) = \text{reached}$ ) then
```

```
        Return  $\text{PATH}(T_a, T_b);$ 
```

```
  SWAP( $T_a, T_b$ );
```

Successfully added  
a node to tree



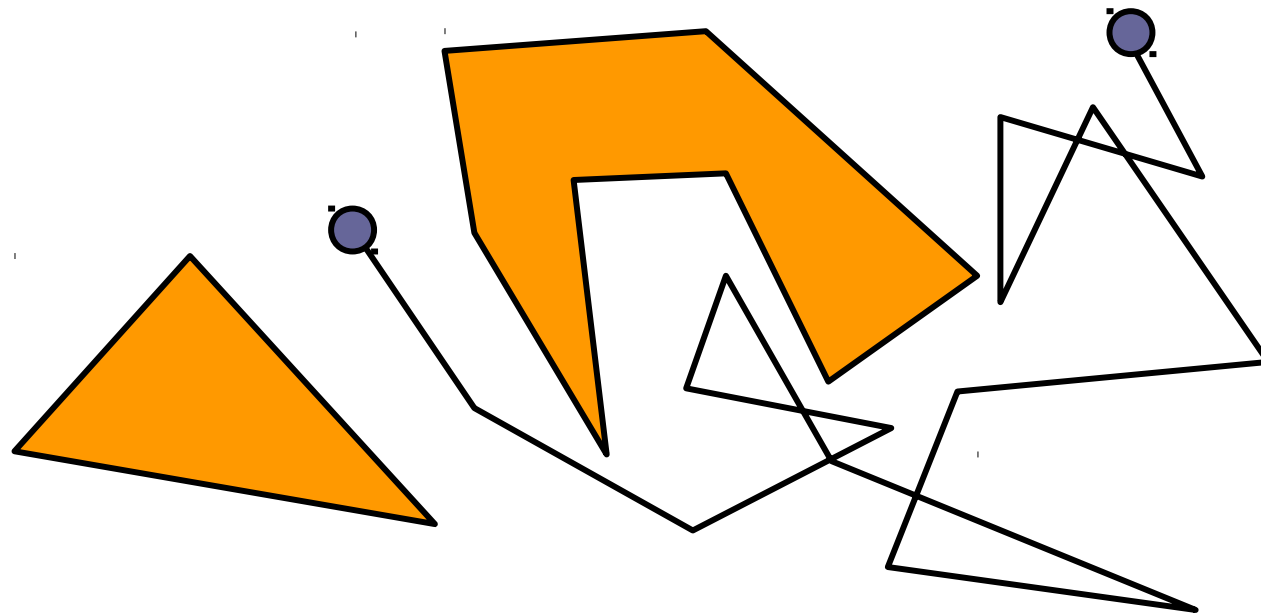
Successfully connected  
 $T_b$  to  $q_{new}$



Instead of switching, use  $T_a$  as smaller tree.



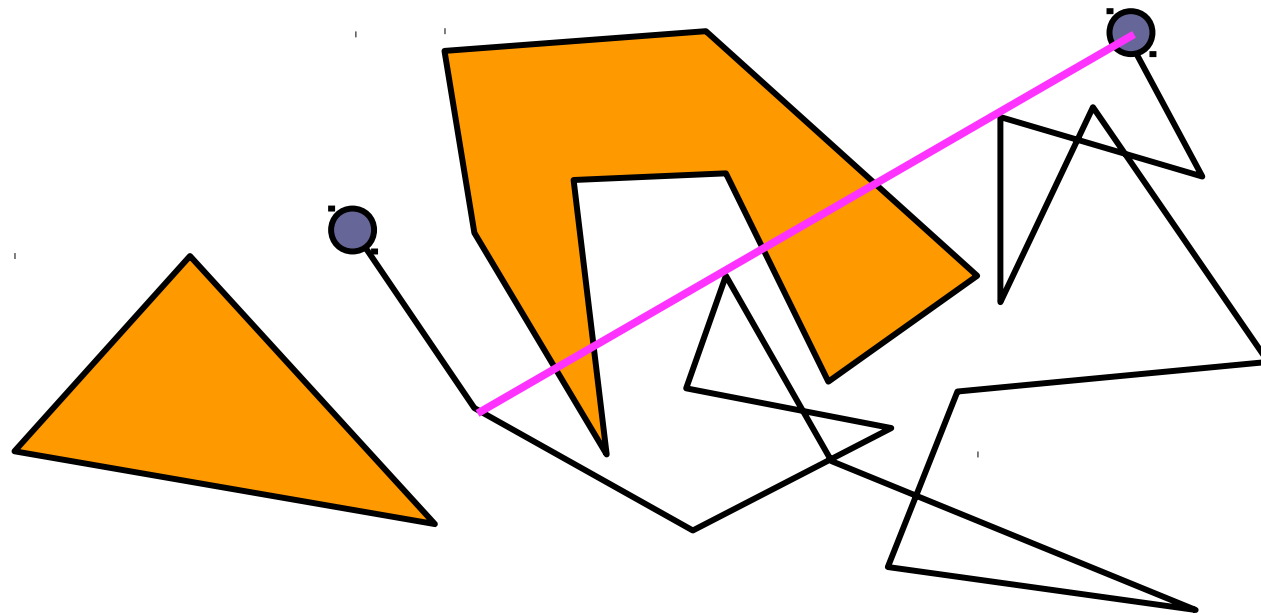
# Smoothing the path



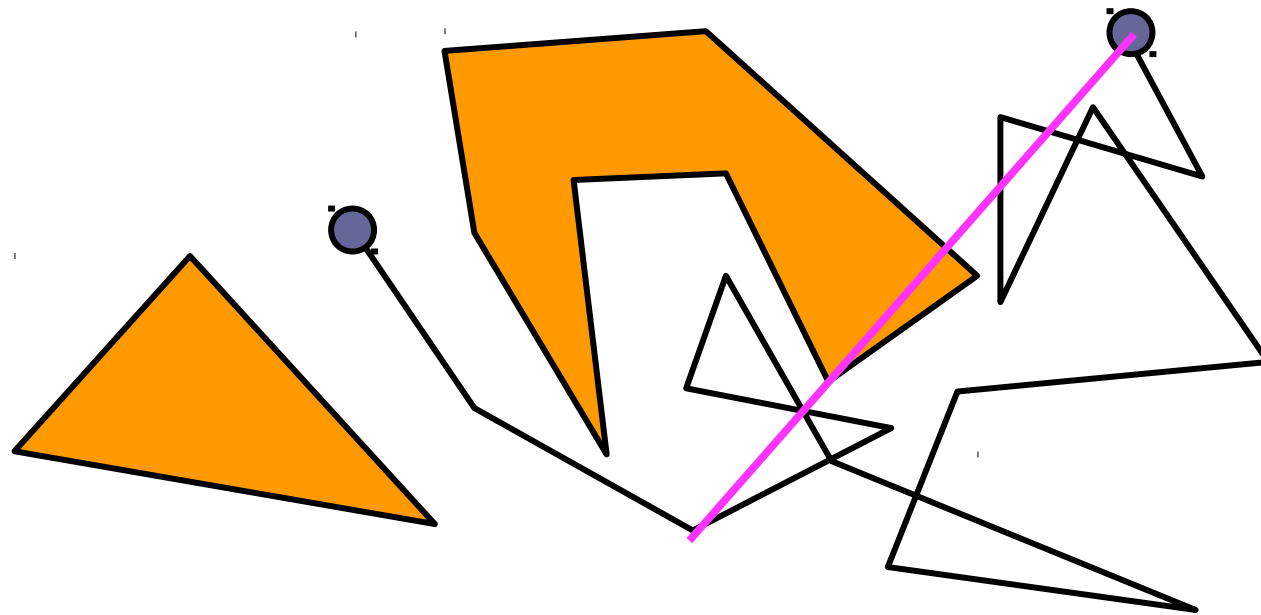




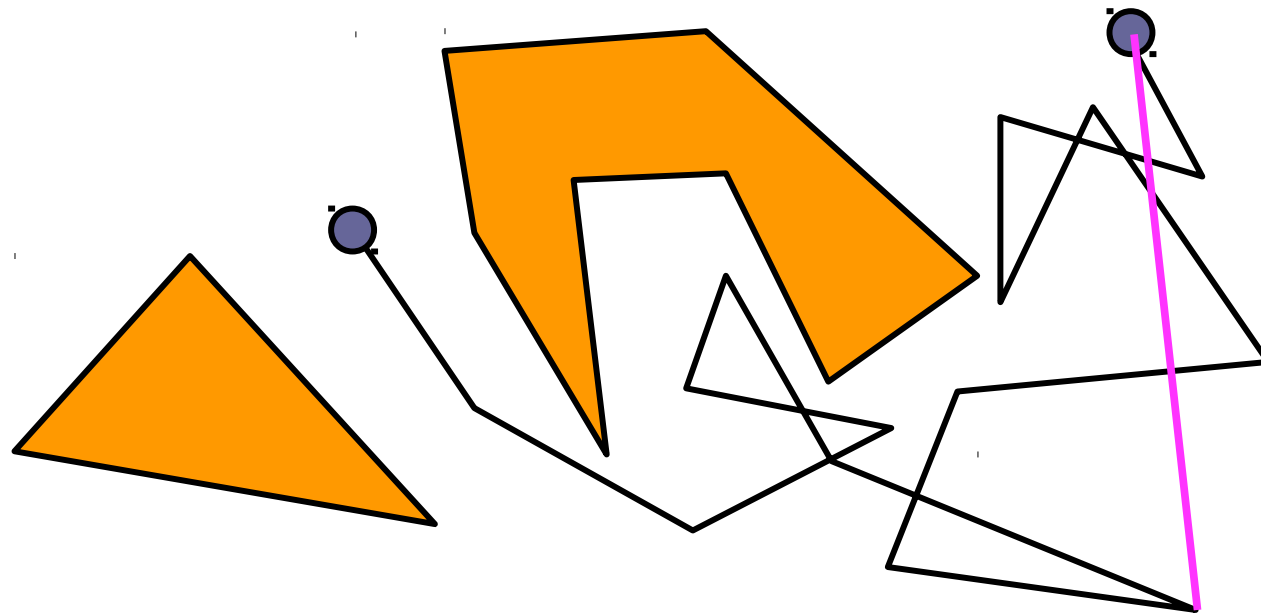
# Smoothing the path



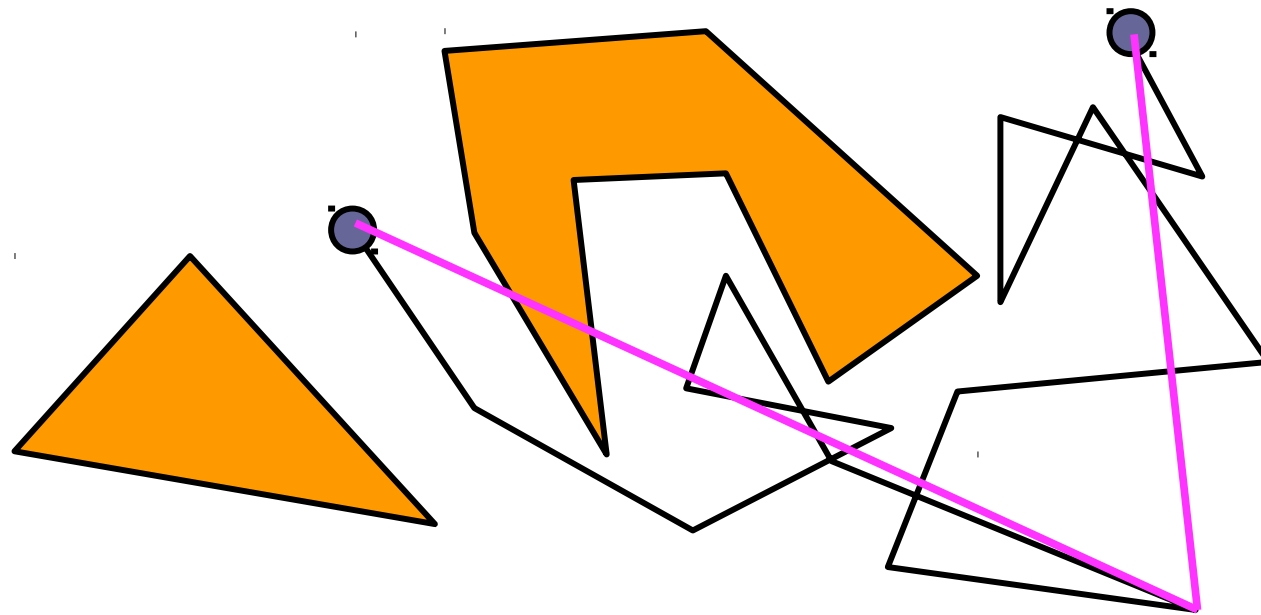
# Smoothing the path



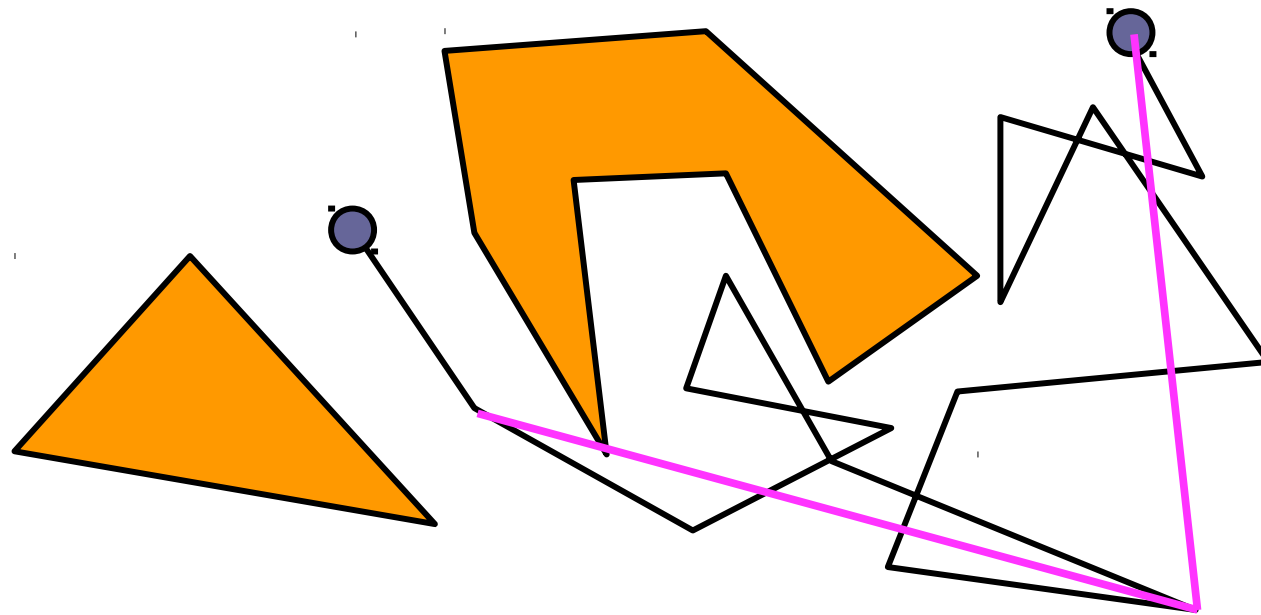
# Smoothing the path



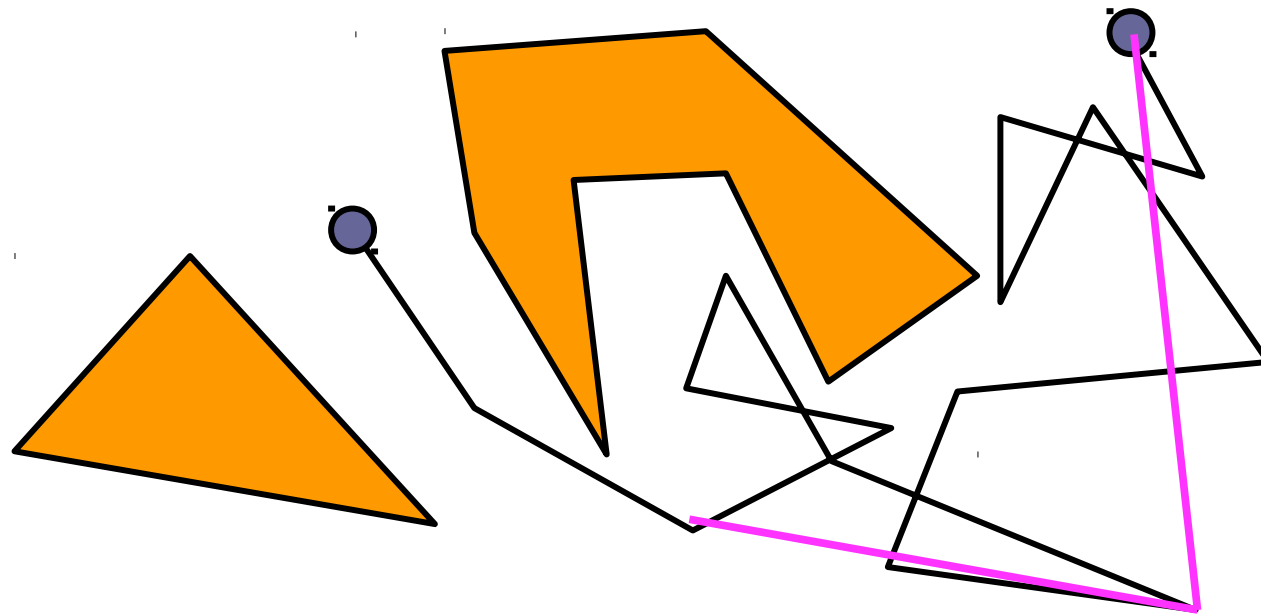
# Smoothing the path



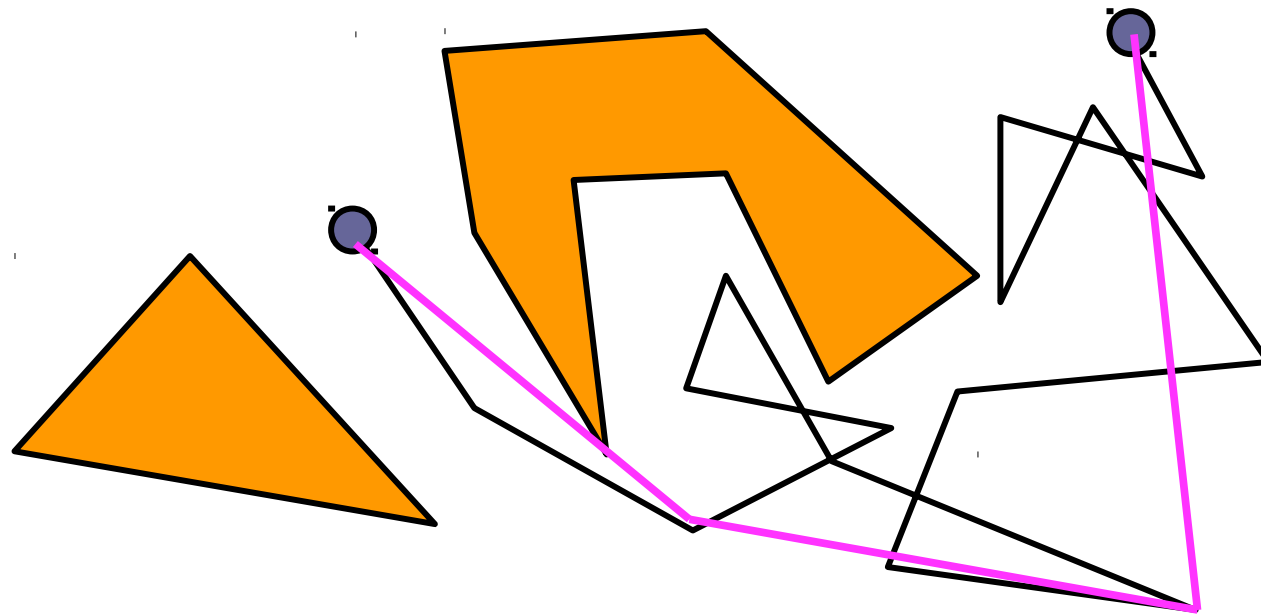
# Smoothing the path



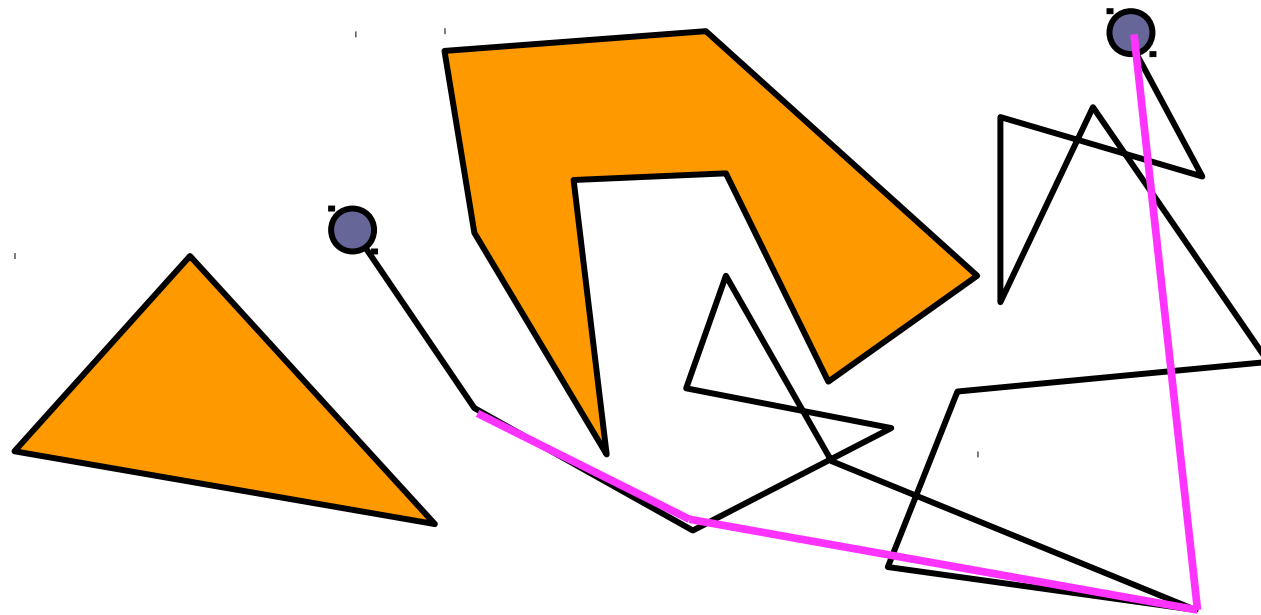
# Smoothing the path



# Smoothing the path

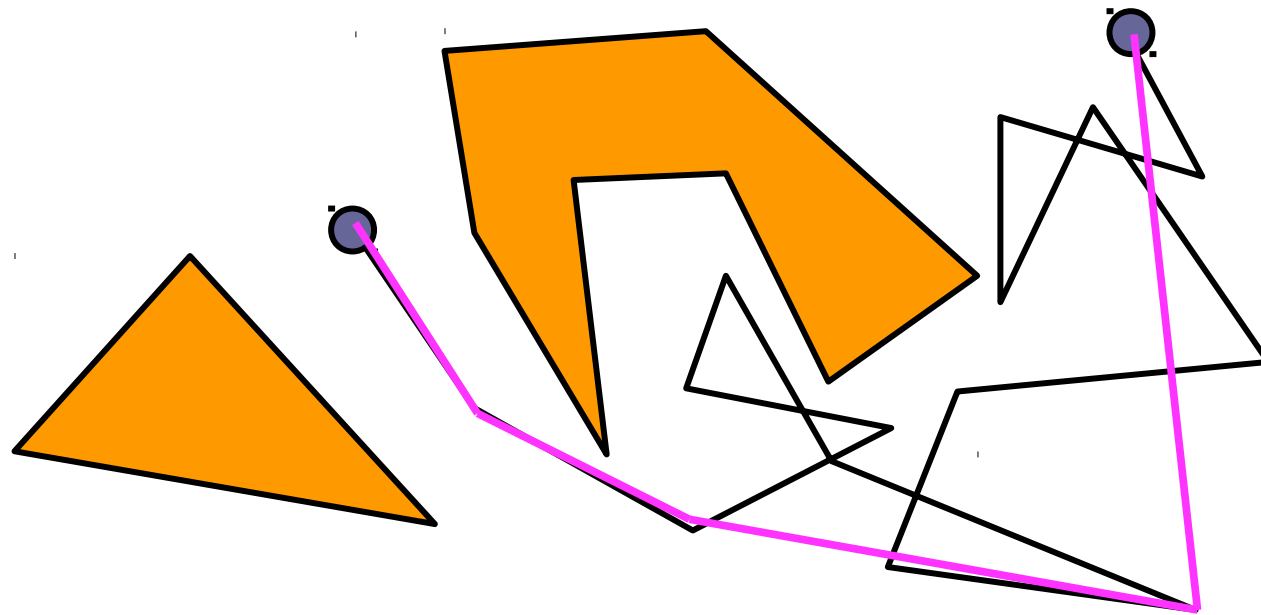


# Smoothing the path

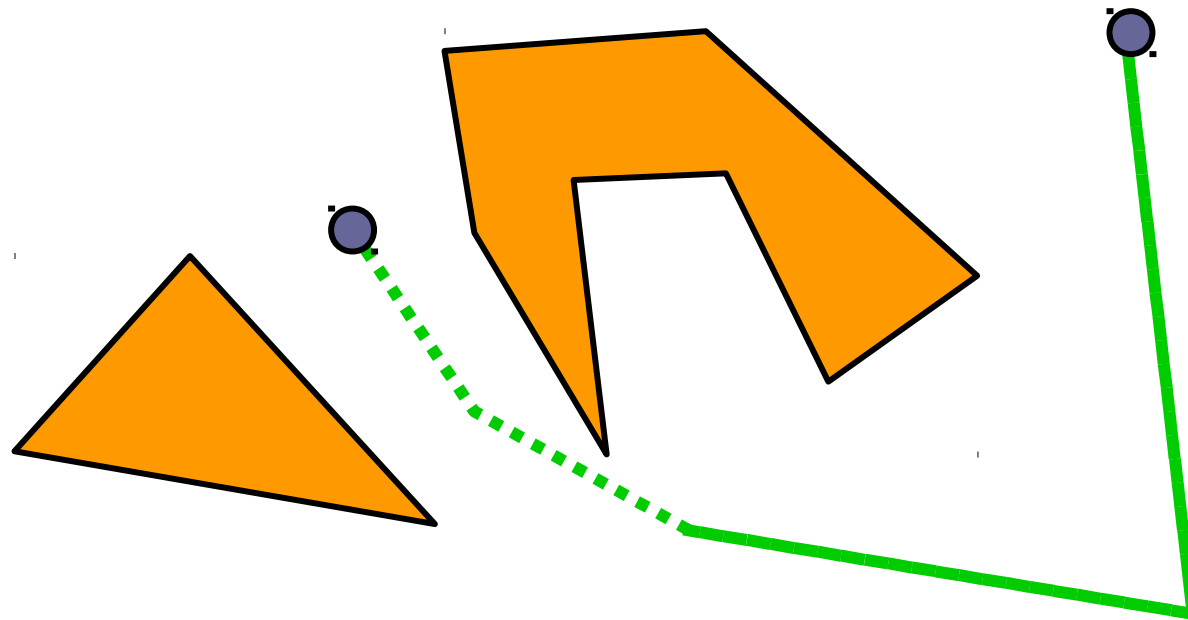




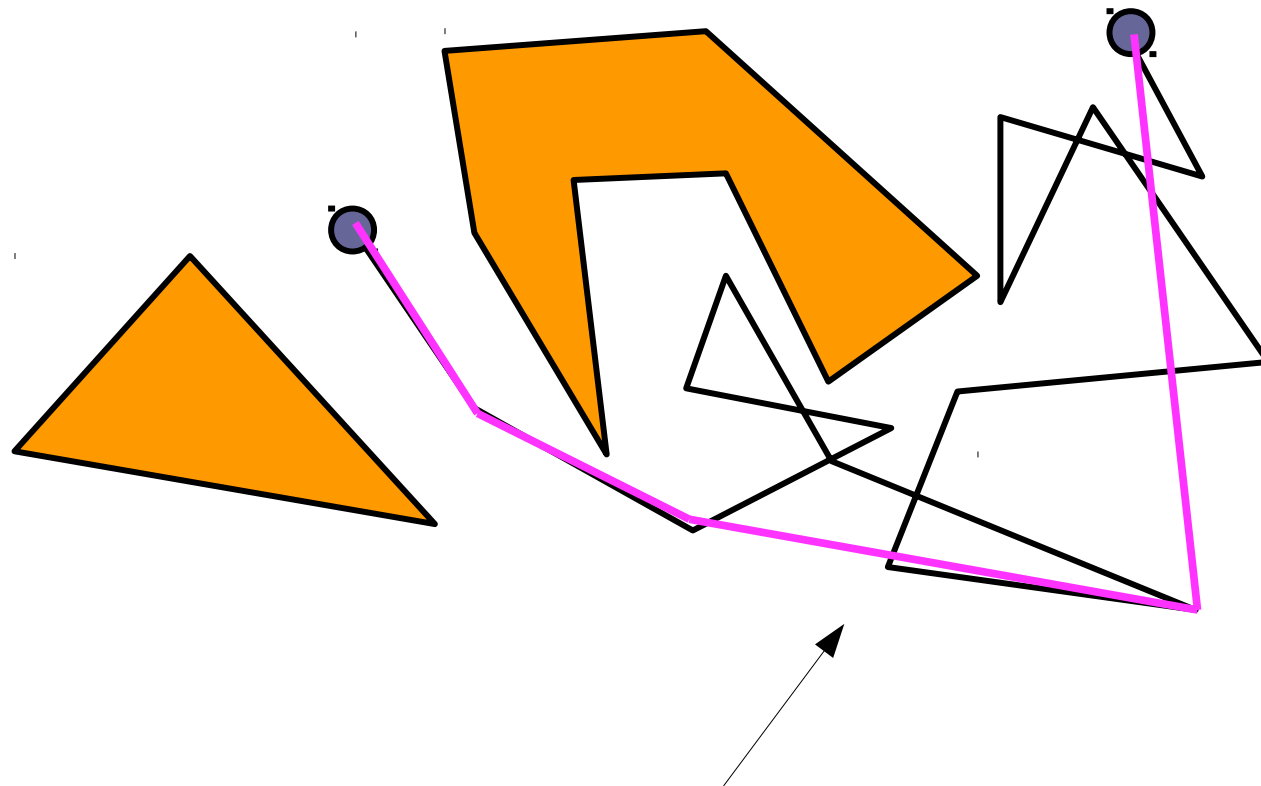
# Smoothing the path



# Smoothing the path



# Smoothing the path



Notice that it wasn't the shortest path...  
– we're just smoothing, not optimizing

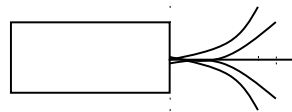
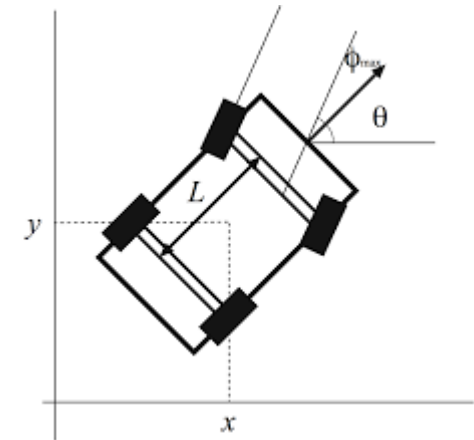
# Kinodynamic planning with RRTs

So far, we have assumed that the system has no dynamics

- the system can instantaneously move in any direction in c-space
- but what if that's not true?

Consider the Dubins car:

- c-space: x-y position and velocity, angle
- control forward velocity and steering angle
- plan a path through c-space with the corresponding control signals

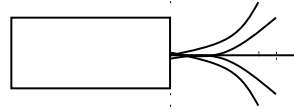


$$x_{t+1} = f(x_t, u_t)$$

where:

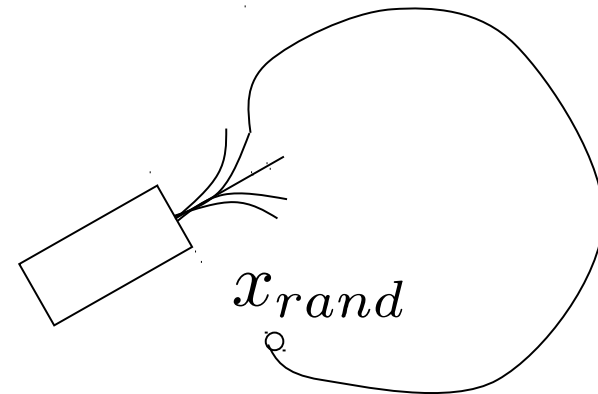
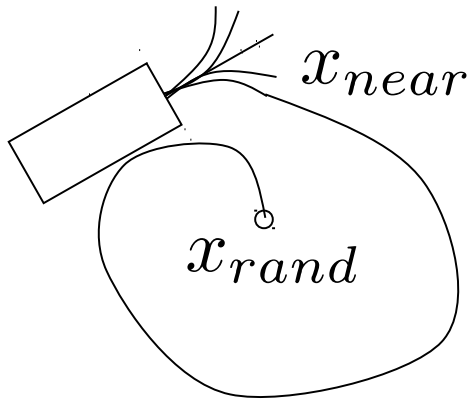
$x_t$  – state (x/y position and velocity, steering angle)  
 $u_t$  – control signal (forward velocity, steering angle)

# Kinodynamic planning with RRTs



$$x_{t+1} = f(x_t, u_t)$$

$$u^* = \arg \min_u (d(x_{rand}, f(x_{near}, u)))$$

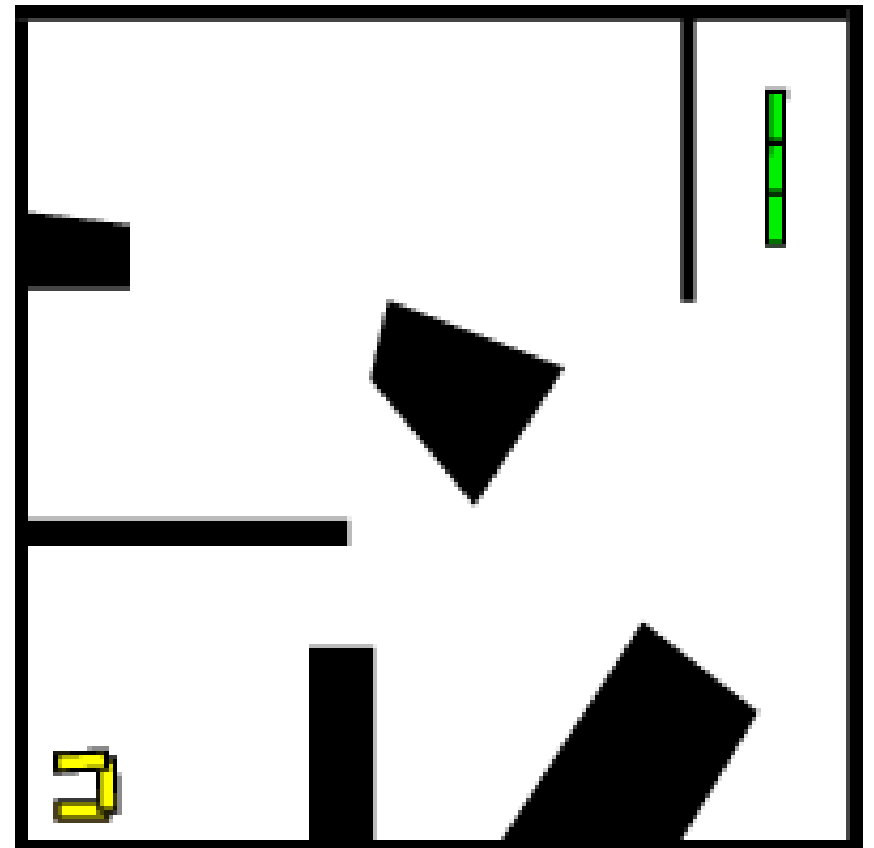
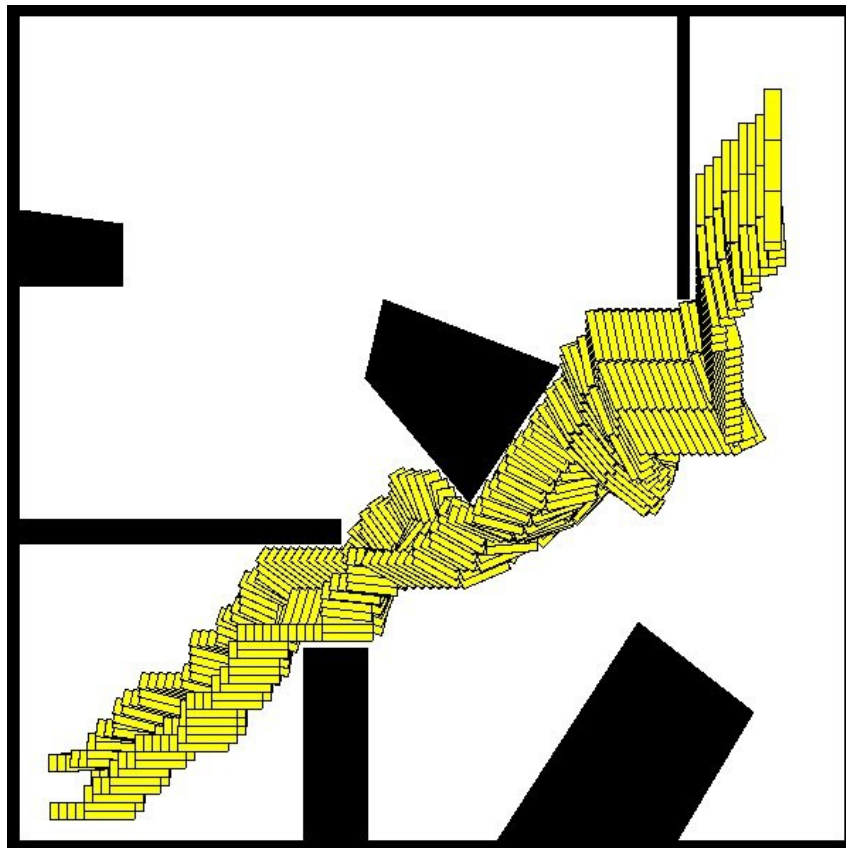


But, what if  $x_{near}$  isn't the right node to expand ???

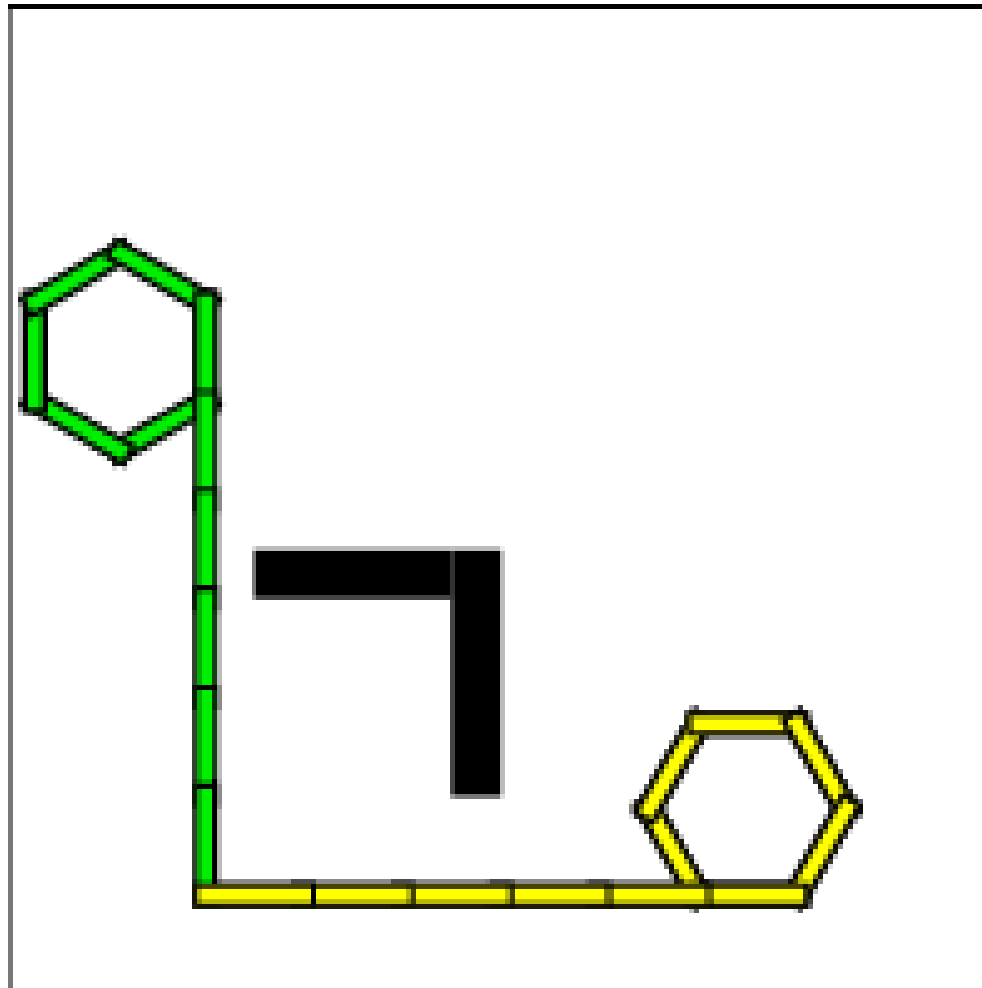
# So, what do they do?

- Use nearest neighbor anyway
- As long as heuristic is not bad, it helps  
(you have already given up completeness and optimality, so what the heck?)
- Nearest neighbor calculations begin to dominate the collision avoidance
- Remember K-D trees

# Articulated Robot

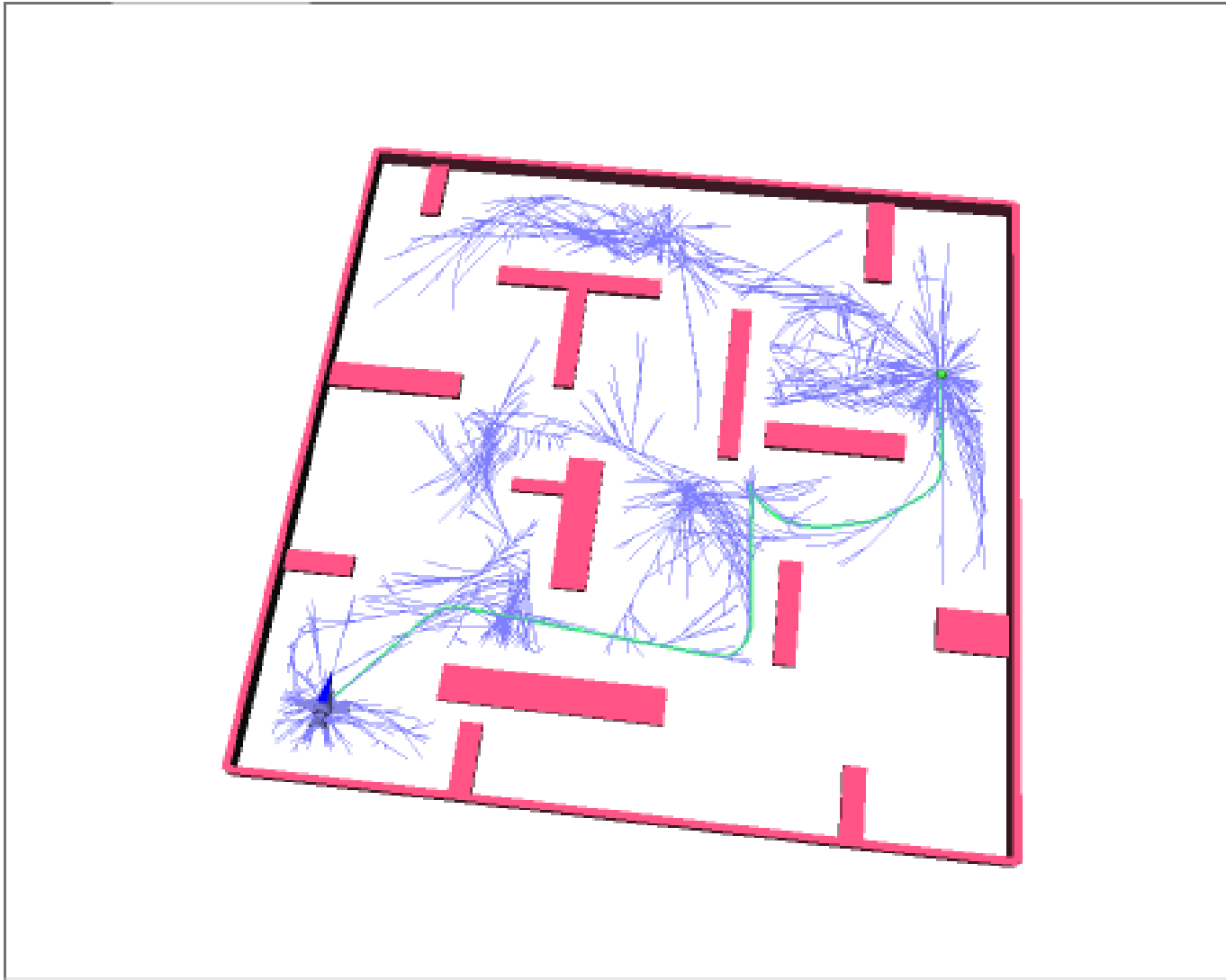


# Highly Articulated Robot

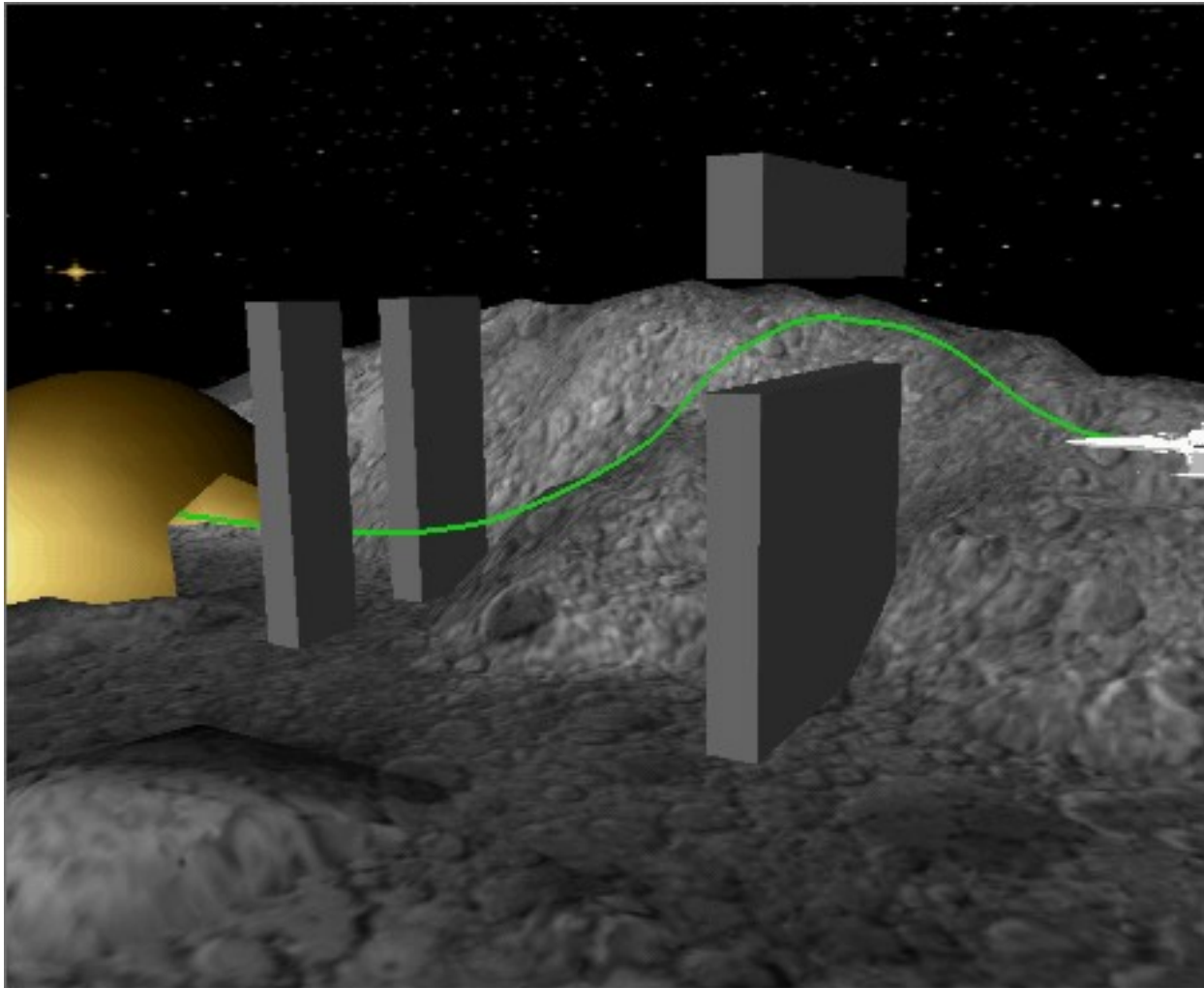




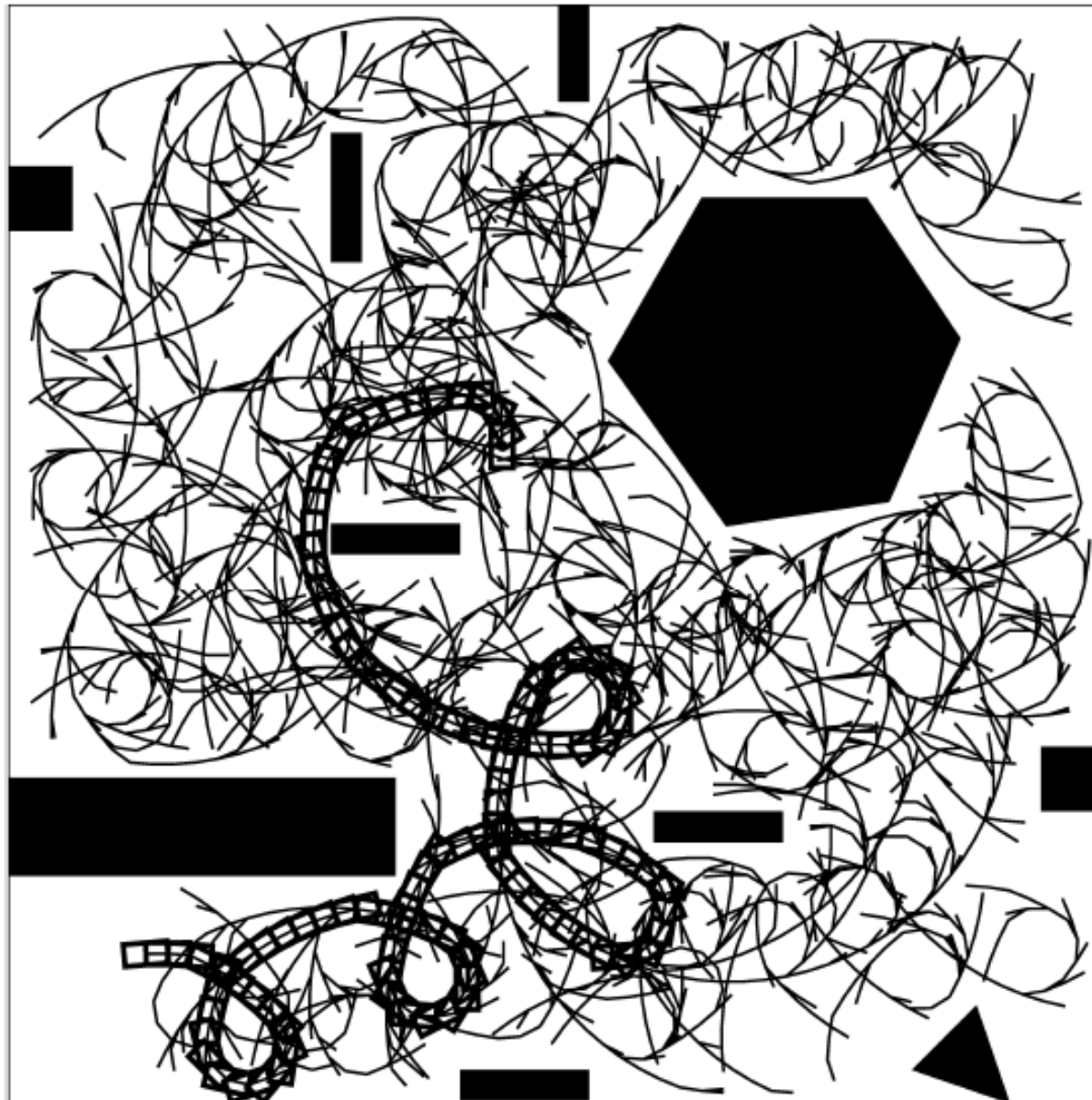
# Hovercraft with 2 Thrusters



# Out of This World Demo



# Left-turn only forward car



# Applications of RRTs

## Robotics Applications

- mobile robotics
- manipulation
- humanoids

## Other Applications

- biology (drug design)
- manufacturing and virtual prototyping (assembly analysis)
- verification and validation
- computer animation and real-time graphics
- aerospace

## RRT extensions

- discrete planning (STRIPS and Rubik's cube)
- real-time RRTs
- anytime RRTs
- dynamic domain RRTs
- deterministic RRTs
- parallel RRTs
- hybrid RRTs