

CS 369: Introduction to Robotics

Prof. Thao Nguyen
Spring 2026



Outline for today

- Sampling-based motion planning
- Robot perception

Outline for today

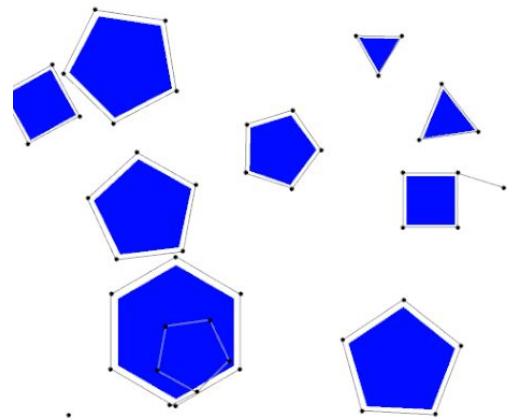
- Sampling-based motion planning
- Robot perception

Motion planning algorithms

	Graph search	Sampling-based
Completeness	Complete	Probabilistically complete
Path optimality	Optimal	Depends
Memory usage	High (store entire map)	Low (store sampled nodes)
Best for	Low-dimensional, 2D grids	High-dimensional, complex spaces

Probabilistic roadmap (PRM)

- Consists of two phases: construction and query
- Construction:
 - Take random samples from the robot's configuration space
 - Connect configurations to neighbors while avoiding collisions with obstacles
 - Add configurations and connections until the roadmap is dense enough
- Query:
 - Connect start and goal configurations to the graph
 - Obtain path with a graph search algorithm



Rapidly exploring random tree (RRT)

Grows a tree rooted at the start configuration by using random samples

Algorithm BuildRRT

Input: Initial configuration q_{init} , number of vertices in RRT K , incremental distance Δq

Output: RRT tree T

$T.init(q_{init})$

for $k = 1$ **to** K **do**

$q_{rand} \leftarrow \text{RAND_CONF}()$

$q_{near} \leftarrow \text{NEAREST_VERTEX}(q_{rand}, T)$

$q_{new} \leftarrow \text{NEW_CONF}(q_{near}, q_{rand}, \Delta q)$

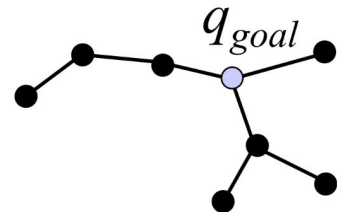
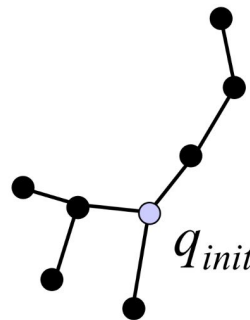
$T.add_vertex(q_{new})$ // Only add if connection is feasible

$T.add_edge(q_{near}, q_{new})$

return T

RRT-Connect

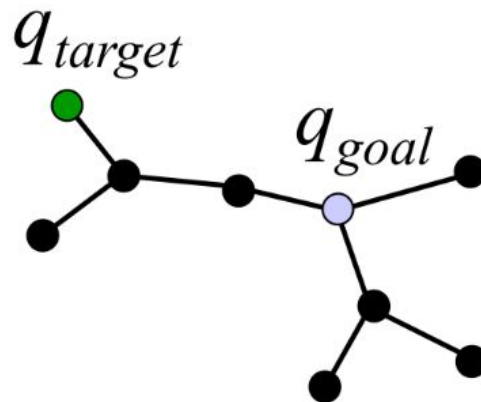
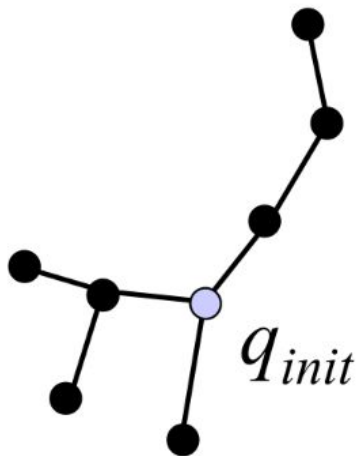
Grows two RRTs towards each other



```
RRT_CONNECT( $q_{init}, q_{goal}$ )  
   $T_a$ .init( $q_{init}$ );  $T_b$ .init( $q_{goal}$ )  
  for  $k = 1$  to  $K$  do  
     $q_{rand} = \text{RANDOM\_CONFIG}()$   
    if not (EXTEND( $T_a, q_{rand}$ ) = Trapped) then  
      if (EXTEND( $T_b, q_{new}$ ) = Reached) then  
        return PATH( $T_a, T_b$ )  
      SWAP( $T_a, T_b$ ) // Instead of switching, use  $T_a$  as smaller tree  
  return Failure
```

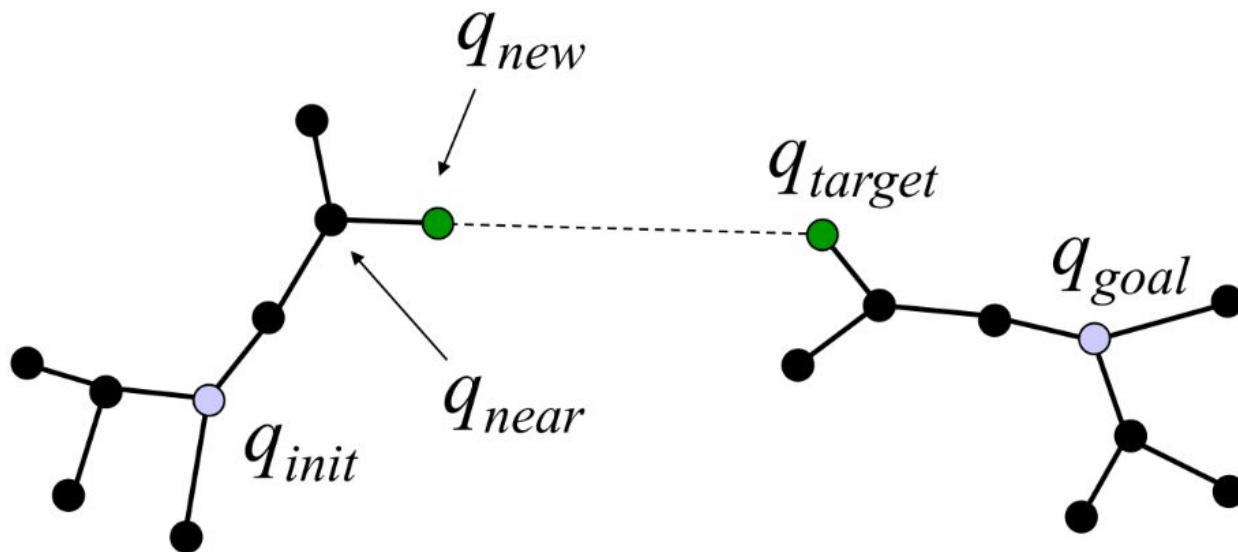
One iteration of RRT-Connect

- One tree grown using random sample
- New node becomes target for other tree



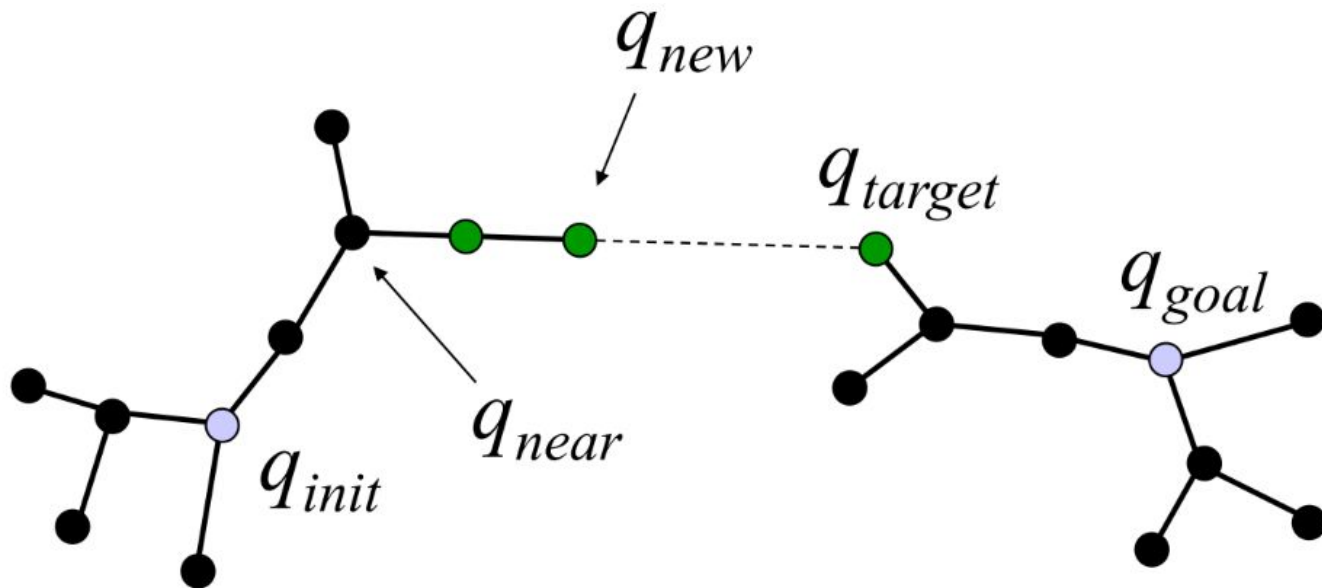
One iteration of RRT-Connect (continued)

- Calculate node “nearest” to target
- Try to add new collision-free branch



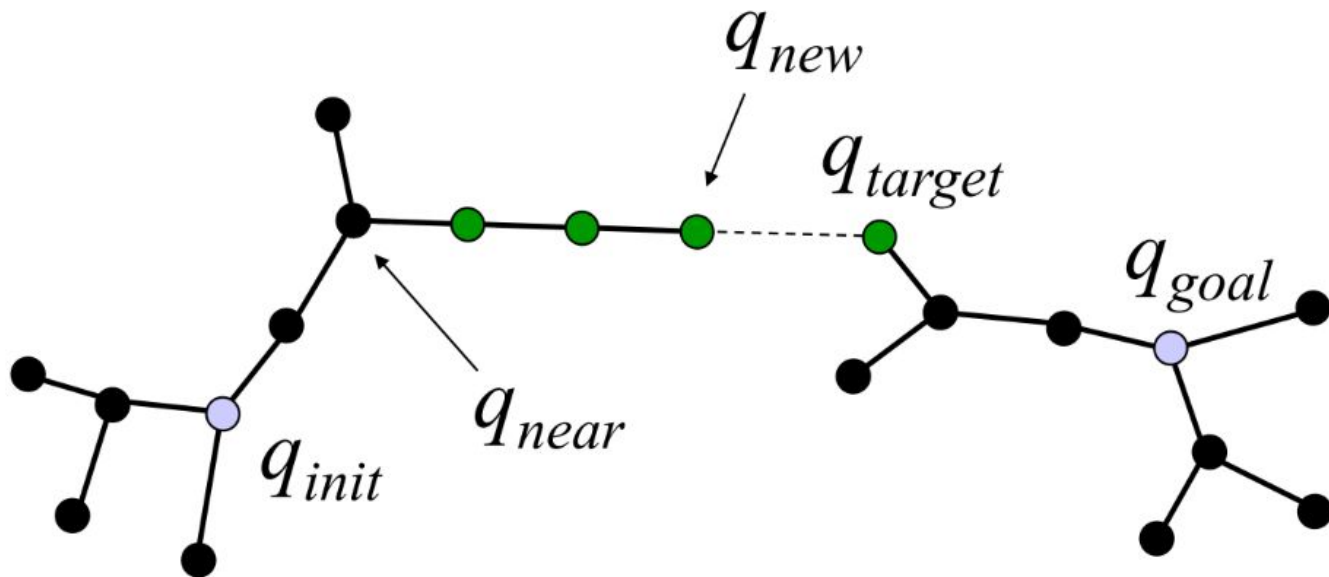
One iteration of RRT-Connect (continued)

- If successful, keep extending branch



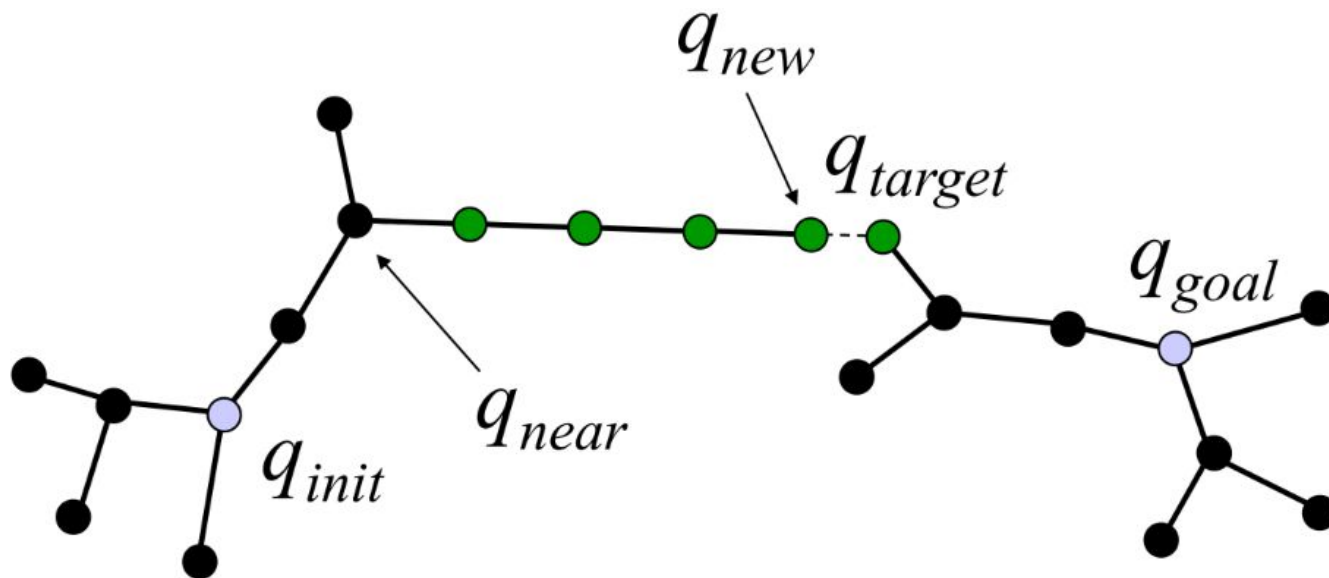
One iteration of RRT-Connect (continued)

- If successful, keep extending branch



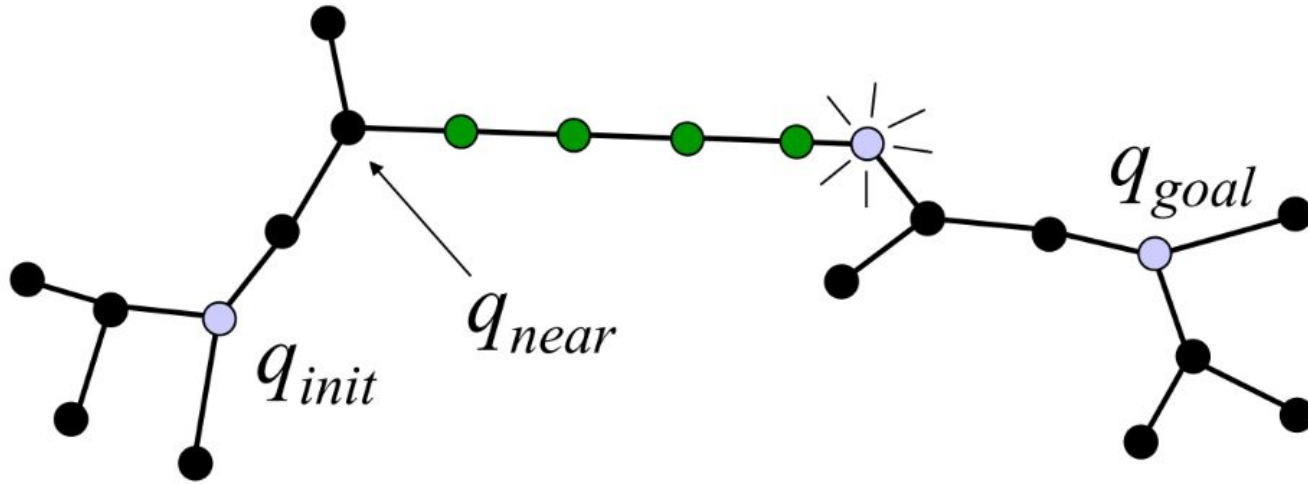
One iteration of RRT-Connect (continued)

- If successful, keep extending branch



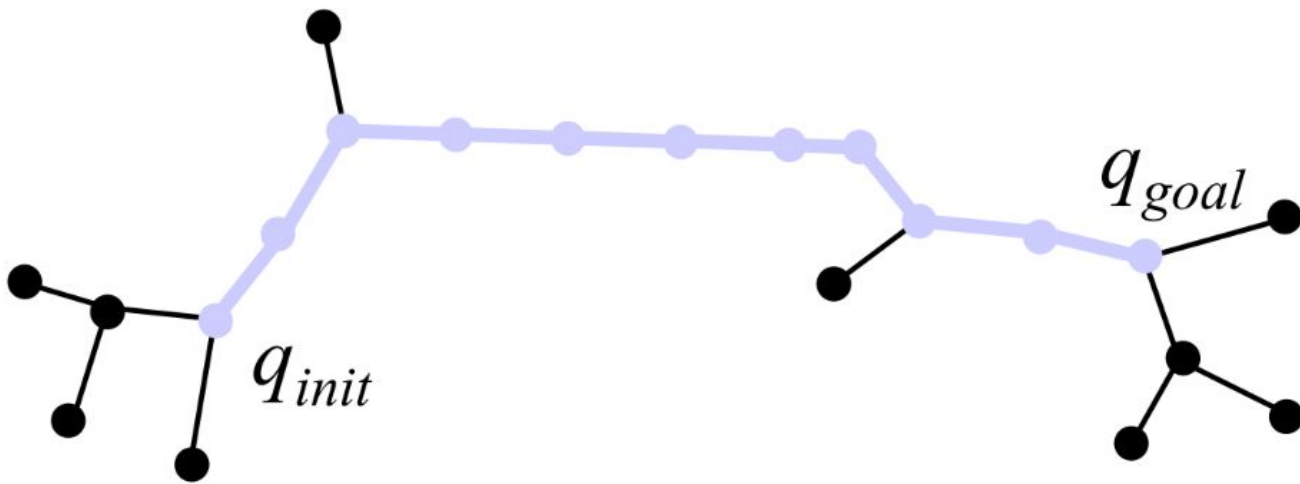
One iteration of RRT-Connect (continued)

- Path found if branch reaches target



One iteration of RRT-Connect (continued)

- Return path connecting start and goal



RRT*

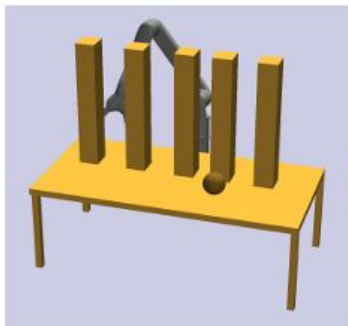
- RRT with two improvements:
 - **Best neighbor search:** when adding a new node, RRT* checks nearby nodes in the tree to connect to the one that results in the lowest cost from the root, rather than just the single nearest node
 - **Tree rewiring:** after adding a new node, the algorithm checks if neighboring nodes can have their path cost reduced by switching their parent to the newly added node
- Computationally more expensive, asymptotically optimal

RRT* pseudo code

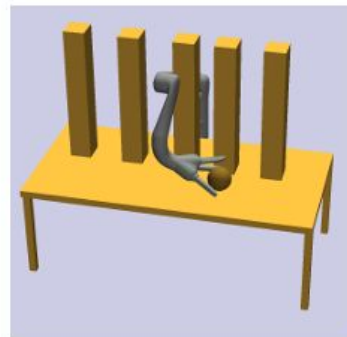
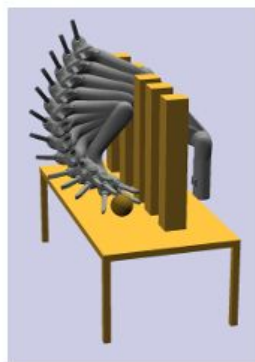
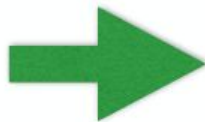
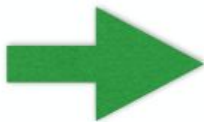
```
T.init( $q_{init}$ )
for  $i = 0$  to  $max\_iter$ :
     $q_{rand} = \text{RAND\_CONF}()$ 
    if  $\text{OBS}(q_{rand}) == \text{True}$ , try again
     $q_{near} = \text{NEAREST}(q_{rand}, T)$ 
    if  $\text{EXTEND}(q_{near}, q_{rand}) \neq \text{Trapped}$ :
         $\text{UPDATE}(q_{rand}, q_{near})$ 
     $q_{best}, neighbors = \text{FIND\_NEIGHBORS}(T, q_{rand}, rad)$ 
    if  $q_{best} \neq q_{near}$  and  $\text{EXTEND}(q_{best}, q_{rand}) \neq \text{Trapped}$ :
         $\text{UPDATE}(q_{rand}, q_{best})$ 
    for  $n$  in  $neighbors$ :
        if  $\text{COST}(q_{rand}) + \text{DIST}(q_{rand}, n) < \text{COST}(n)$  and  $\text{EXTEND}(q_{rand}, n) \neq \text{Trapped}$ :
             $\text{UPDATE}(n, q_{rand})$ 
return  $T$ 
```

} RRT

Putting it all together



start pose



goal