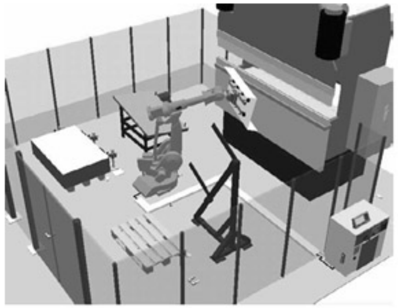
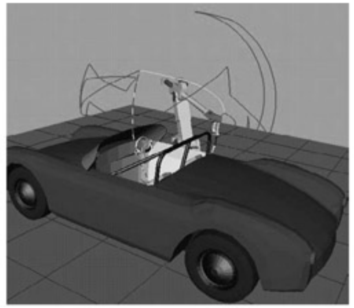
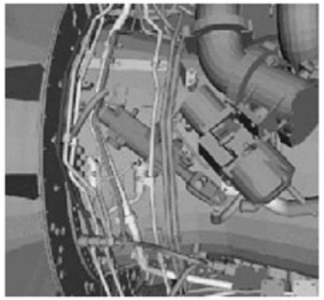
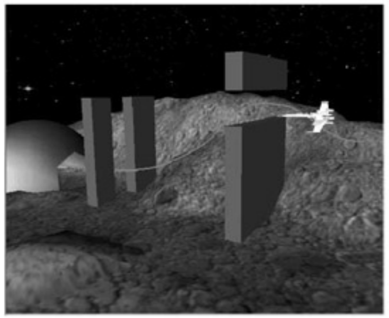
Sampling-based algorithms

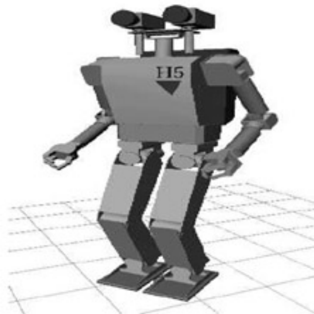
- For problems with a lot of Degrees of Freedom or constraints (kinematic and dynamic).
- Instead of finding an optimal solution considering the whole environment, only few samples are considered.
- Each sample is a robot configuration.
- Solution to path planning will be a sequence of connected samples which all bellong to Q_{free} and connect the start and goal positions.
- A procedure is used to determine if a configuration is in Q_{free} or not.
- Algorithms can also guarantee the finding of the solution (completeness), they are probabilistic completeness.







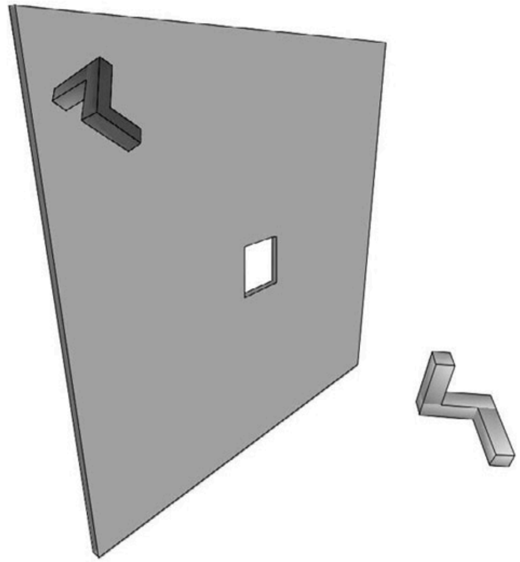


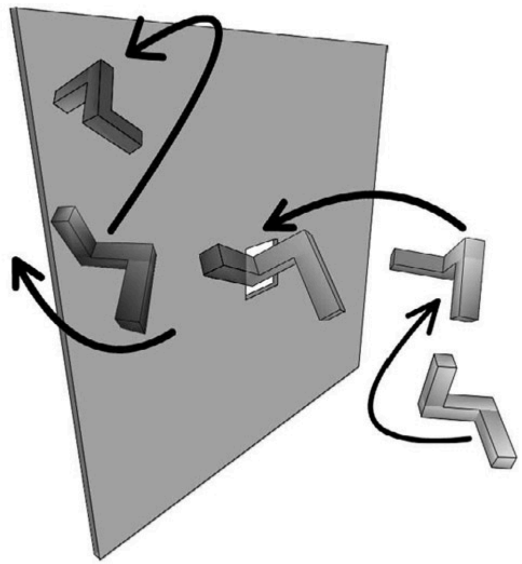


Sampling-based algorithms

Probabilistic RoadMap planner

- It is a multiple-query planner that creates a roadmap in Q_{free}.
- Coarse sampling using a uniform random distribution is used to obtain the nodes of the roadmap.
- The edges between nodes are planned, by a local planner, with fine sampling to ensure that all configurations belong to Q_{free}.
- Phases:
  - Learning phase, to create the roadmap.
  - Query phase, to plan particular paths between a start and a goal configurations.
- Roadmap is represented by a graph G=(V,E); V: vertices or nodes; E: edges generated by the local planner that correspond to a collision-free path from q_1 to q_2. Simplest form of the local planner: the straight line.
- In the query phase, q_{init} and q_{goal} are connected to two nodes q' and q'' respectively. The planner searches G for connecting q' and q'', and generates the path.





Sampling-based algorithms

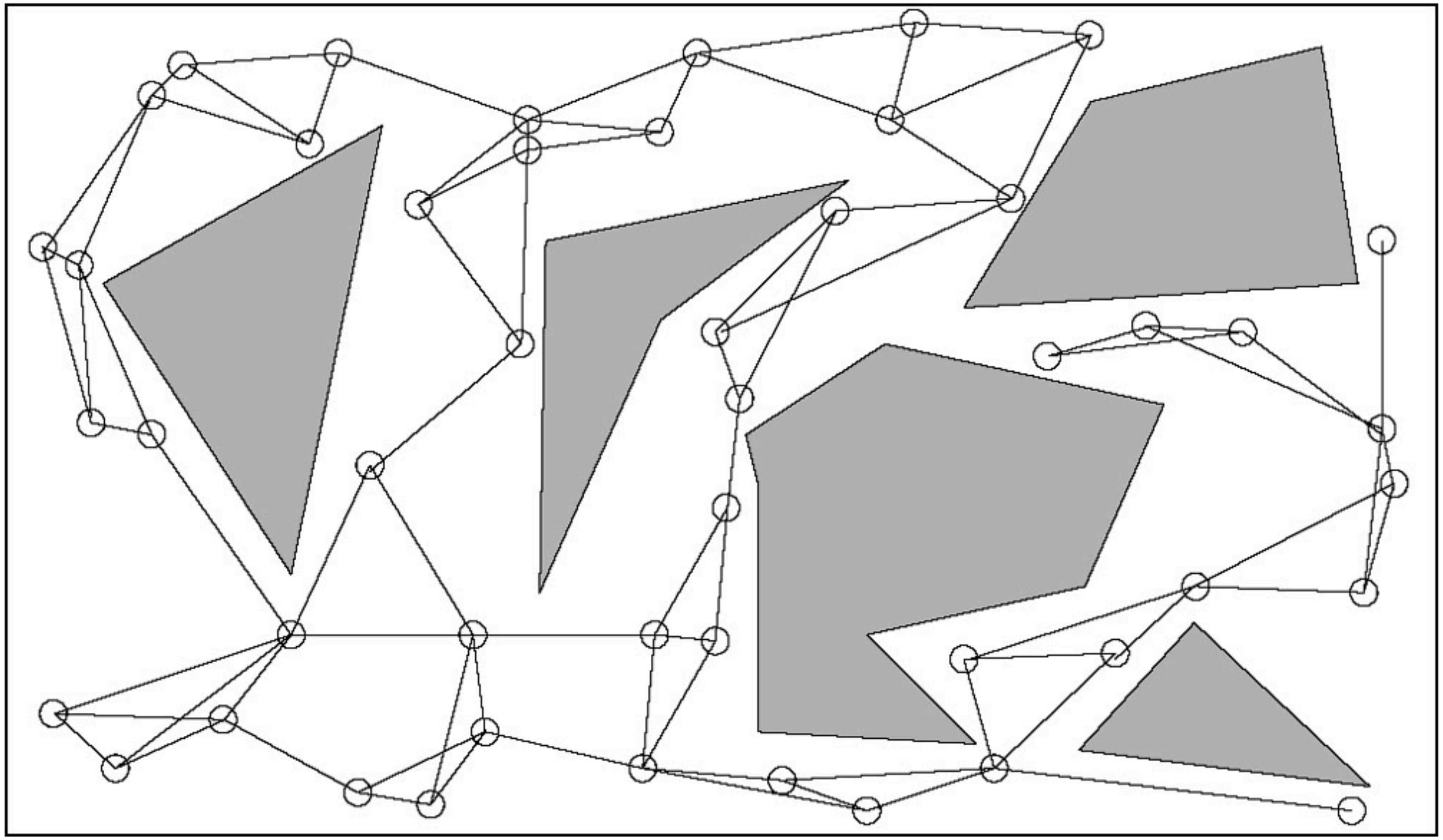
Algorithm 6: Roadmap Construction

```
Input:
n : number of nodes to put in the roadmap
k : number of closest neighbors to examine for each configuration
Output:
A roadmap G = (V, E)
V ← Ø
E ← Ø
while |V| < n do
  repeat
    q ← a random configuration in Q
  until q is collision-free
  V ← V ∪{q}
end while
for all q ∈ V do
    N_q ← the k closest neighbors of q chosen from V according to dist
    for all q′ ∈ N_q do
      if (q, q′) ∉ E and Δ(q, q′) ≠ NIL then
        E ← E ∪{(q, q′)}
      end if
    end for
end for
```

- *Being Δ the local planner and dist a metric function to measure distance between two configurations*

Probabilistic RoadMap planner

- Roadmap in a 2D space, local planner: straight line planner, n=50, k=3.



Algorithm 7: Solve Query Algorithm

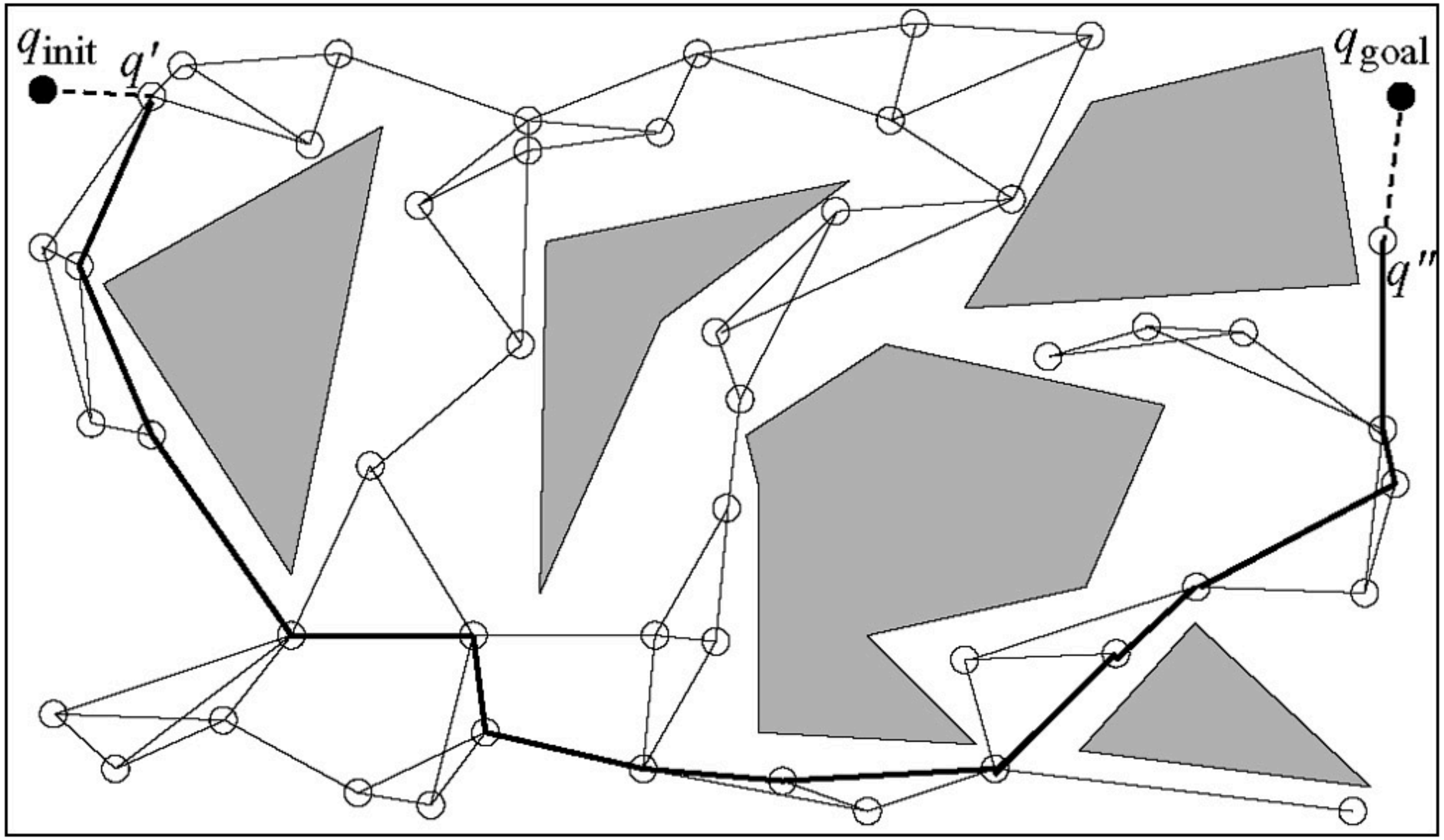
Input:
q_{init}: the initial configuration
q_{goal}: the goal configuration
k: the number of closest neighbors to examine for each configuration
$G = (V, E)$: the roadmap computed by algorithm 6
Output:
A path from q_{init} to q_{goal} or failure

$N_{q\text{init}} \leftarrow$ the k closest neighbors of q_{init} from V according to *dist*
$N_{q\text{goal}} \leftarrow$ the k closest neighbors of q_{goal} from V according to *dist*
$V \leftarrow \{q_{\text{init}}\} \cup \{q_{\text{goal}}\} \cup V$
set q' to be the closest neighbor of q_{init} in $N_{q_{\text{init}}}$
repeat
  if $\Delta(q_{\text{init}}, q') \neq$ NIL **then**
    $E \leftarrow (q_{\text{init}}, q') \cup E$
  else
    set q' to be the next closest neighbor of q_{init} in $N_{q_{\text{init}}}$
  end if
until a connection was succesful or the set $N_{q\text{init}}$ is empty
set q' to be the closest neighbor of q_{goal} in $N_{q_{\text{goal}}}$
repeat
  if $\Delta(q_{\text{goal}}, q') \neq$ NIL **then**
    $E \leftarrow (q_{\text{goal}}, q') \cup E$
  else
    set q' to be the next closest neighbor of q_{goal} in $N_{q_{\text{goal}}}$
  end if
until a connection was succesful or the set $N_{q_{\text{goal}}}$ is empty
$P \leftarrow$ shortest path$(q_{\text{init}}, q_{\text{goal}}, G)$
if P is not empty **then**
  return P
else
return failure
end if

Sampling-based algorithms

Probabilistic RoadMap planner

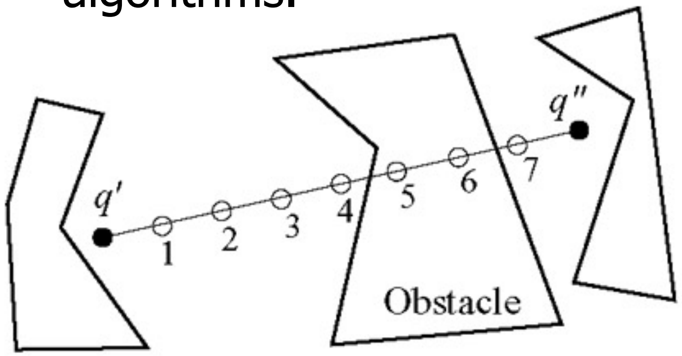
- Query solved with a graph-search algorithm (i.e. A*)

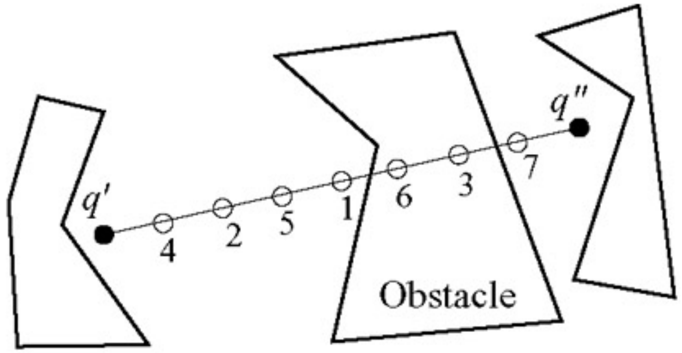


Sampling-based algorithms

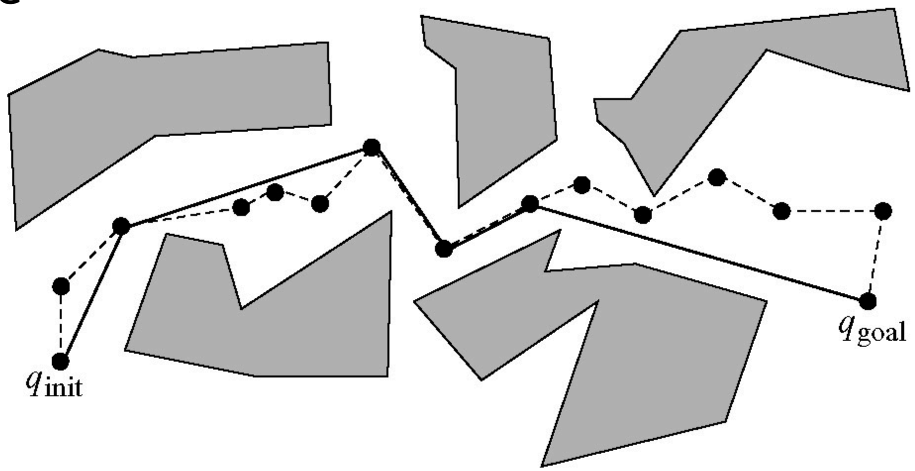
Sampling-based algorithms implementation details

- **Straight-line local planner** implementation:
  - Discretization of the line according to a small step size.
  - Collision checking strategies: incremental (left) and subdivision (right) algorithms.





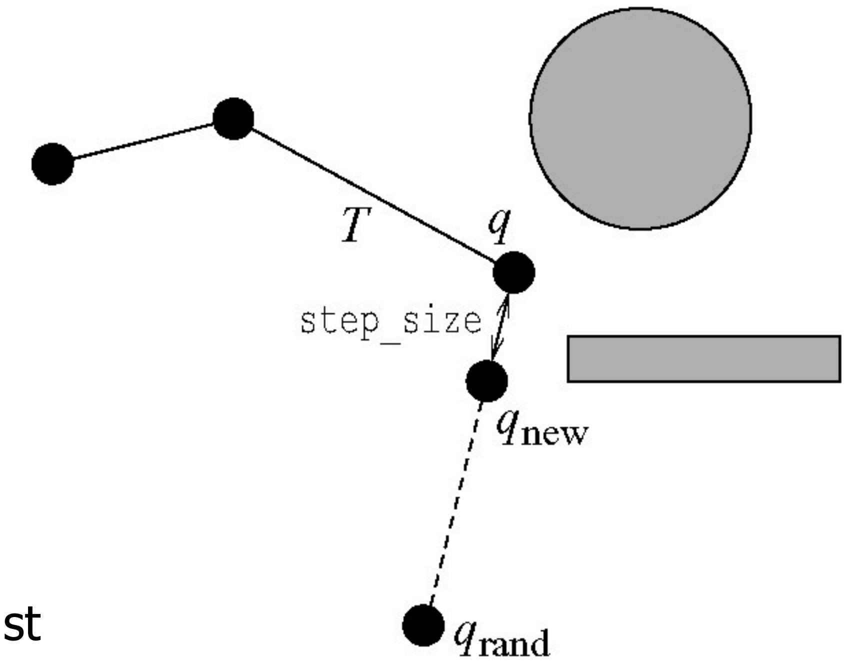
- **Postprocessing queries** to improve shortness and smoothness.
  - Greedy approach: connect q_{goal} from q_{init}, if it fails try from a closer position until it connects. Once q_{goal} connected start again with its directly connected position.



Sampling-based algorithms

Single-Query Sampling-Based Planners

- Different approaches to build directly, without the roadmap, the path between two configurations.
- For large number of degrees of freedom, or kinematic and dynamic constraints.
- **RRT algorithm** (Rapidly-Exploring Random Trees)
  - Most well known sampling algorithm
  - 2 trees, T_{init} and T_{goal}, grow rooted at q_{init} and q_{goal} respectively.
  - A random configuration q_{rand} is sampled uniformly in Q_{free}.
  - The nearest configuration q_{near} is found, and a new configuration q_{new} is generated at a step_size distance towards q_{rand}.
  - q_{new} and the edge (q_{near}, q_{new}) must belong to Q_{free}.



Sampling-based algorithms

Algorithm 10: Build RRT Algorithm

Input:
q_0: the configuration where the tree is rooted
n: the number of attempts to expand the tree
Output:
A tree $T = (V, E)$ that is rooted at q_0 and has $\leq n$ configurations

$V \leftarrow \{q_0\}$
$E \leftarrow \emptyset$
for $i = 1$ to n **do**
  $q_{\text{rand}} \leftarrow$ a randomly chosen free configuration
  extend RRT (T, q_{rand})
end for
return T

Algorithm 11: Extend RRT Algorithm

Input:
$T = (V, E)$: an RRT
q: a configuration toward which the tree T is grown
Output:
A new configuration q_{new} toward q, or NIL in case of failure

$q_{\text{near}} \leftarrow$ closest neighbor of q in T
$q_{\text{new}} \leftarrow$ progress q_{near} by step_size along the straight line in $\mathcal{Q}$ between q_{near} and q_{rand}
if q_{new} is collision-free **then**
  $V \leftarrow V \cup \{q^{\text{new}}\}$
  $E \leftarrow E \cup \{(q_{\text{near}}, q_{\text{new}})\}$
  return q_{new}
end if
return NIL

Sampling-based algorithms

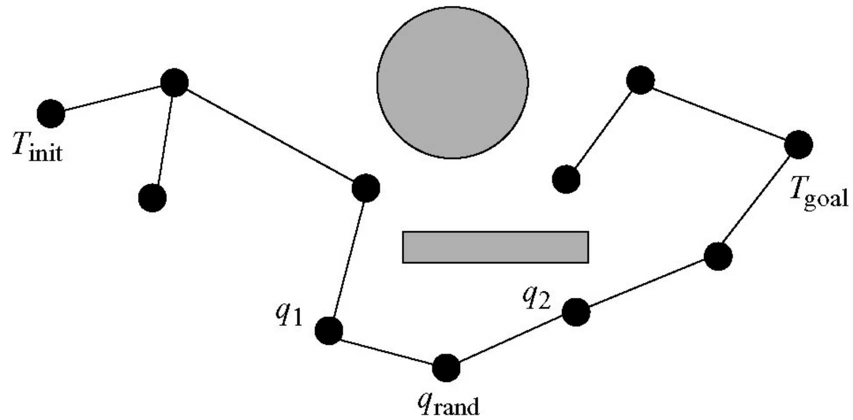
RRT algorithm

- The sampling is usually guided towards q_{goal} (or q_{init}) to improve the efficiency:
  - with p probability: $q_{rand} = q_{goal}$
  - with (1-p) probability: qrand = random uniform distribution
- Merging of trees, T_{init} and T_{goal},

Algorithm 13: Merge RRT Algorithm

Input:
T_1: first RRT
T_2: second RRT
ℓ: number of attempts allowed to merge T_1 and T_2
Output:
merged if the two RRTs are connected to each other; failure otherwise

for $i = 1$ to ℓ **do**
  $q_{rand} \leftarrow$ a randomly chosen free configuration
  $q_{new,1} \leftarrow$ extend RRT (T_1, q_{rand})
  if $q_{new,1} \neq$ NIL **then**
    $q_{new,2} \leftarrow$ extend RRT $(T_2, q_{new,1})$
    if $q_{new,1} = q_{new,2}$ **then**
      return merged
    end if
    SWAP(T_1, T_2)
  end if
end for
return failure



RRT algorithm, examples