



**HAL**  
open science

## Motion planning

Florent Lamiraux

► **To cite this version:**

Florent Lamiraux. Motion planning. Doctoral. GdR Robotics Winter School: Robotica Principia, Centre de recherche Inria Sophia Antipolis – Méditerranée, France. 2019. cel-02130128

**HAL Id: cel-02130128**

**<https://inria.hal.science/cel-02130128>**

Submitted on 15 May 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

# Motion planning

Florent Lamiraux

CNRS-LAAS, Toulouse, France

# Motion planning

Introduction

Definitions

Random Sampling

Collision testing

Software

## Context

industrial robot



aerial vehicle



autonomous  
vehicle



### Autonomous mobile systems

- ▶ moving in an environment cluttered by obstacles
- ▶ possibly subject to kinematic or dynamic constraints

Motion planning : automatically compute a feasible collision-free path between two given configurations.

## Context

industrial robot



aerial vehicle



autonomous  
vehicle



### Autonomous mobile systems

- ▶ moving in an environment cluttered by obstacles
- ▶ possibly subject to kinematic or dynamic constraints

Motion planning : automatically compute a feasible collision-free path between two given configurations.

## Context

industrial robot



aerial vehicle



autonomous  
vehicle



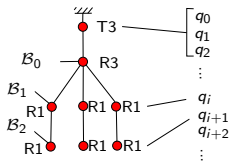
### Autonomous mobile systems

- ▶ moving in an environment cluttered by obstacles
- ▶ possibly subject to kinematic or dynamic constraints

Motion planning : automatically compute a feasible collision-free path between two given configurations.

## Robot

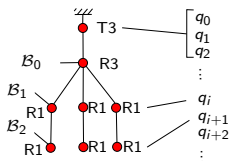
Set of rigid bodies  $\mathcal{B}_0, \dots, \mathcal{B}_m$ , linked to one another by *joints*.



Joint : mobility of a body in the reference frame of its parent, parameterized by one or several numbers.

## Robot configuration

The configuration  $\mathbf{q}$  of a robot is represented by the concatenation of the parameters of each joint.

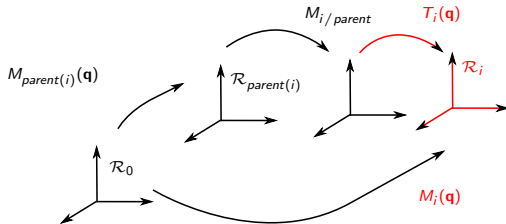




## Forward kinematics

Computation of the position of each joint in world frame.

$$M_i(\mathbf{q}) = M_{parent(i)}(\mathbf{q}) M_{i/parent} T_i(\mathbf{q})$$



## Definitions

- ▶ **Workspace** :  $\mathcal{W} = \mathbb{R}^2$  or  $\mathbb{R}^3$  : space in which the robot moves
- ▶ Workspace obstacle : compact subset of  $\mathcal{W}$ , denoted by  $\mathcal{O}$ .
- ▶ Configuration space :  $\mathcal{C}$ .
- ▶ Position in configuration  $\mathbf{q}$  of a point  $M \in \mathcal{B}_i$  :  $\mathbf{x}_i(M, \mathbf{q})$ .
- ▶ Configuration space obstacle :

$$\mathcal{C}_{obst} = \{ \mathbf{q} \in \mathcal{C}, \exists i \in \{1, \dots, m\}, \exists M \in \mathcal{B}_i, \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ ou} \\ \exists i, j \in \{1, \dots, m\}, \exists M_i \in \mathcal{B}_i, \exists M_j \in \mathcal{B}_j, \\ \mathbf{x}_i(M_i, \mathbf{q}) = \mathbf{x}_j(M_j, \mathbf{q}) \}$$

- ▶ Free configuration space :  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$ .

## Definitions

- ▶ Workspace :  $\mathcal{W} = \mathbb{R}^2$  or  $\mathbb{R}^3$  : space in which the robot moves
- ▶ Workspace obstacle : compact subset of  $\mathcal{W}$ , denoted by  $\mathcal{O}$ .
- ▶ Configuration space :  $\mathcal{C}$ .
- ▶ Position in configuration  $\mathbf{q}$  of a point  $M \in \mathcal{B}_i$  :  $\mathbf{x}_i(M, \mathbf{q})$ .
- ▶ Configuration space obstacle :

$$\mathcal{C}_{obst} = \{ \mathbf{q} \in \mathcal{C}, \exists i \in \{1, \dots, m\}, \exists M \in \mathcal{B}_i, \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ ou} \\ \exists i, j \in \{1, \dots, m\}, \exists M_i \in \mathcal{B}_i, \exists M_j \in \mathcal{B}_j, \\ \mathbf{x}_i(M_i, \mathbf{q}) = \mathbf{x}_j(M_j, \mathbf{q}) \}$$

- ▶ Free configuration space :  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$ .

## Definitions

- ▶ Workspace :  $\mathcal{W} = \mathbb{R}^2$  or  $\mathbb{R}^3$  : space in which the robot moves
- ▶ Workspace obstacle : compact subset of  $\mathcal{W}$ , denoted by  $\mathcal{O}$ .
- ▶ Configuration space :  $\mathcal{C}$ .
- ▶ Position in configuration  $\mathbf{q}$  of a point  $M \in \mathcal{B}_i$  :  $\mathbf{x}_i(M, \mathbf{q})$ .
- ▶ Configuration space obstacle :

$$\mathcal{C}_{obst} = \{ \mathbf{q} \in \mathcal{C}, \exists i \in \{1, \dots, m\}, \exists M \in \mathcal{B}_i, \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ ou} \\ \exists i, j \in \{1, \dots, m\}, \exists M_i \in \mathcal{B}_i, \exists M_j \in \mathcal{B}_j, \\ \mathbf{x}_i(M_i, \mathbf{q}) = \mathbf{x}_j(M_j, \mathbf{q}) \}$$

- ▶ Free configuration space :  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$ .

## Definitions

- ▶ Workspace :  $\mathcal{W} = \mathbb{R}^2$  or  $\mathbb{R}^3$  : space in which the robot moves
- ▶ Workspace obstacle : compact subset of  $\mathcal{W}$ , denoted by  $\mathcal{O}$ .
- ▶ Configuration space :  $\mathcal{C}$ .
- ▶ Position in configuration  $\mathbf{q}$  of a point  $M \in \mathcal{B}_i$  :  $\mathbf{x}_i(M, \mathbf{q})$ .
- ▶ Configuration space obstacle :

$$\mathcal{C}_{obst} = \{ \mathbf{q} \in \mathcal{C}, \exists i \in \{1, \dots, m\}, \exists M \in \mathcal{B}_i, \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ ou} \\ \exists i, j \in \{1, \dots, m\}, \exists M_i \in \mathcal{B}_i, \exists M_j \in \mathcal{B}_j, \\ \mathbf{x}_i(M_i, \mathbf{q}) = \mathbf{x}_j(M_j, \mathbf{q}) \}$$

- ▶ Free configuration space :  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$ .

## Definitions

- ▶ Workspace :  $\mathcal{W} = \mathbb{R}^2$  or  $\mathbb{R}^3$  : space in which the robot moves
- ▶ Workspace obstacle : compact subset of  $\mathcal{W}$ , denoted by  $\mathcal{O}$ .
- ▶ Configuration space :  $\mathcal{C}$ .
- ▶ Position in configuration  $\mathbf{q}$  of a point  $M \in \mathcal{B}_i$  :  $\mathbf{x}_i(M, \mathbf{q})$ .
- ▶ Configuration space obstacle :

$$\mathcal{C}_{obst} = \{ \mathbf{q} \in \mathcal{C}, \exists i \in \{1, \dots, m\}, \exists M \in \mathcal{B}_i, \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ ou} \\ \exists i, j \in \{1, \dots, m\}, \exists M_i \in \mathcal{B}_i, \exists M_j \in \mathcal{B}_j, \\ \mathbf{x}_i(M_i, \mathbf{q}) = \mathbf{x}_j(M_j, \mathbf{q}) \}$$

- ▶ Free configuration space :  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$ .

## Definitions

- ▶ Workspace :  $\mathcal{W} = \mathbb{R}^2$  or  $\mathbb{R}^3$  : space in which the robot moves
- ▶ Workspace obstacle : compact subset of  $\mathcal{W}$ , denoted by  $\mathcal{O}$ .
- ▶ Configuration space :  $\mathcal{C}$ .
- ▶ Position in configuration  $\mathbf{q}$  of a point  $M \in \mathcal{B}_i$  :  $\mathbf{x}_i(M, \mathbf{q})$ .
- ▶ Configuration space obstacle :

$$\mathcal{C}_{obst} = \{ \mathbf{q} \in \mathcal{C}, \exists i \in \{1, \dots, m\}, \exists M \in \mathcal{B}_i, \mathbf{x}_i(M, \mathbf{q}) \in \mathcal{O} \text{ ou} \\ \exists i, j \in \{1, \dots, m\}, \exists M_i \in \mathcal{B}_i, \exists M_j \in \mathcal{B}_j, \\ \mathbf{x}_i(M_i, \mathbf{q}) = \mathbf{x}_j(M_j, \mathbf{q}) \}$$

- ▶ Free configuration space :  $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$ .

# Motion

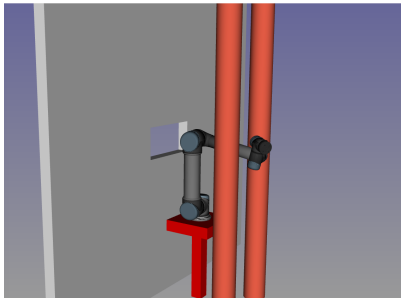
- ▶ Motion :
  - ▶ continuous mapping from  $[0, 1]$  into  $\mathcal{C}$ .
- ▶ Collision free motion :
  - ▶ continuous mapping from  $[0, 1]$  into  $\mathcal{C}_{free}$ .



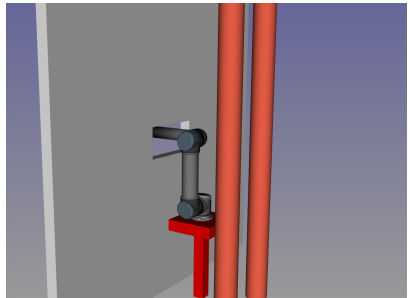
# Motion

- ▶ Motion :
  - ▶ continuous mapping from  $[0, 1]$  into  $\mathcal{C}$ .
- ▶ Collision free motion :
  - ▶ continuous mapping from  $[0, 1]$  into  $\mathcal{C}_{free}$ .

## Motion planning problem



initial configuration



goal configuration

$$\mathcal{C} = [-2\pi, 2\pi]^6$$

# History

- ▶ before the 1990's : mainly a mathematical problem
  - ▶ Real algebraic geometry
  - ▶ Decidability : Schwartz and Sharir 1982
    - ▶ Tarski theorem, Collins decomposition
  - ▶ Approximate cell decomposition
- ▶ from the 1990's : an algorithmic problem
  - ▶ random sampling (1993)
  - ▶ asymptotically optimal random sampling (2011)

# History

- ▶ before the 1990's : mainly a mathematical problem
  - ▶ Real algebraic geometry
  - ▶ Decidability : Schwartz and Sharir 1982
    - ▶ Tarski theorem, Collins decomposition
  - ▶ Approximate cell decomposition
- ▶ from the 1990's : an algorithmic problem
  - ▶ random sampling (1993)
  - ▶ asymptotically optimal random sampling (2011)

## Random sampling

- ▶ Random sampling motion planning methods appeared in the early 1990's
- ▶ Principle
  - ▶ sample random configurations
  - ▶ test whether they are collision-free
  - ▶ build a roadmap the nodes of which are the free configurations, and the edges of which are collision-free linear interpolations.

## Random sampling

- ▶ Random sampling motion planning methods appeared in the early 1990's
- ▶ Principle
  - ▶ sample random configurations
  - ▶ test whether they are collision-free
  - ▶ build a roadmap the nodes of which are the free configurations, and the edges of which are collision-free linear interpolations.

## Random sampling

- ▶ Random sampling motion planning methods appeared in the early 1990's
- ▶ Principle
  - ▶ sample random configurations
  - ▶ test whether they are collision-free
  - ▶ build a roadmap the nodes of which are the free configurations, and the edges of which are collision-free linear interpolations.

## Random sampling

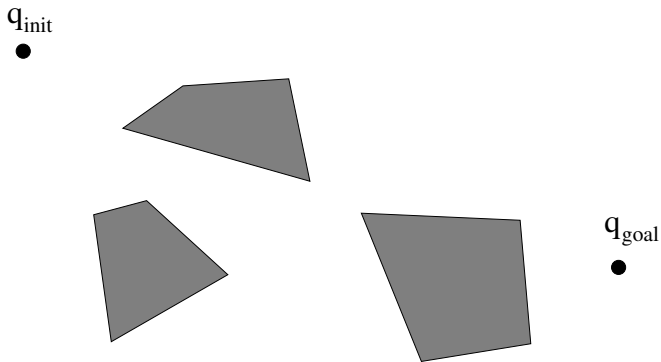
- ▶ Random sampling motion planning methods appeared in the early 1990's
- ▶ Principle
  - ▶ sample random configurations
  - ▶ test whether they are collision-free
  - ▶ build a roadmap the nodes of which are the free configurations, and the edges of which are collision-free linear interpolations.



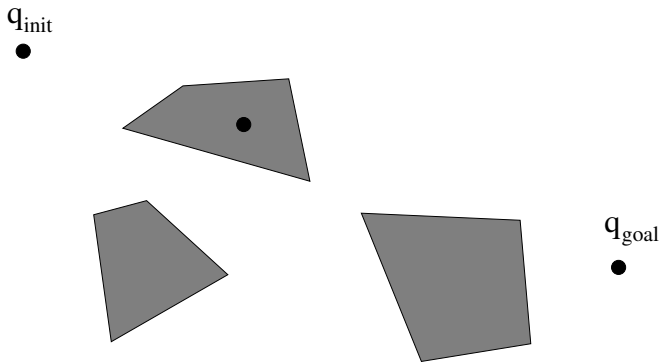
## Random sampling

- ▶ Random sampling motion planning methods appeared in the early 1990's
- ▶ Principle
  - ▶ sample random configurations
  - ▶ test whether they are collision-free
  - ▶ build a roadmap the nodes of which are the free configurations,
  - ▶ and the edges of which are collision-free linear interpolations.

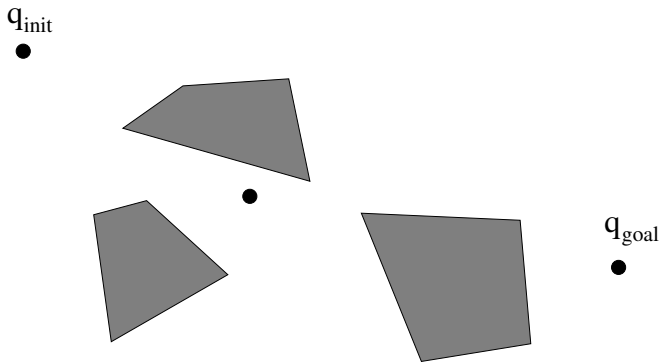
## Probabilistic roadmap (PRM) 1994



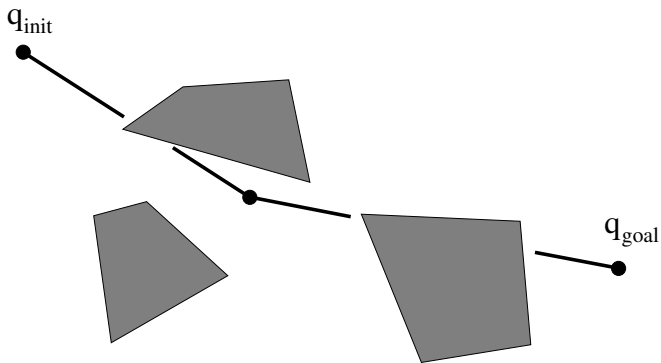
## Probabilistic roadmap (PRM) 1994



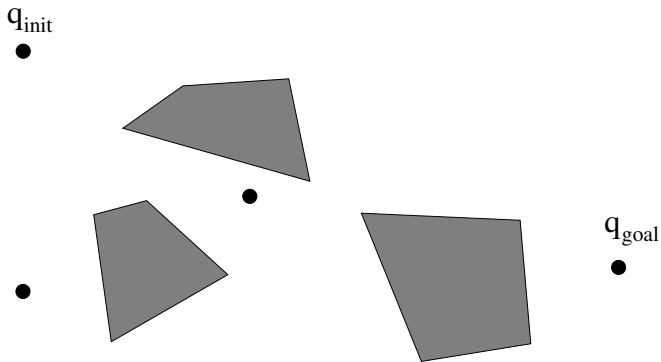
## Probabilistic roadmap (PRM) 1994



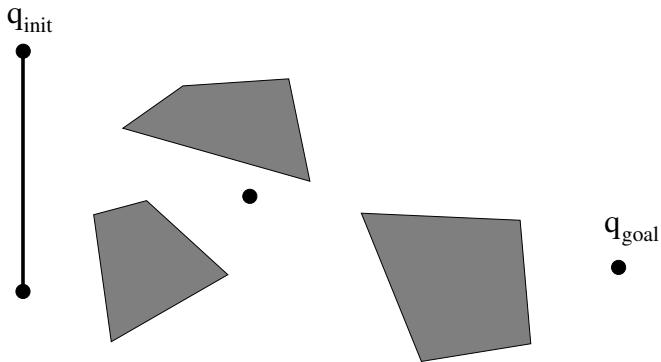
## Probabilistic roadmap (PRM) 1994



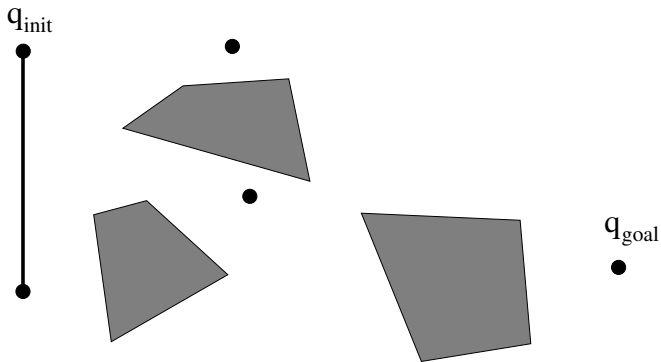
## Probabilistic roadmap (PRM) 1994



## Probabilistic roadmap (PRM) 1994

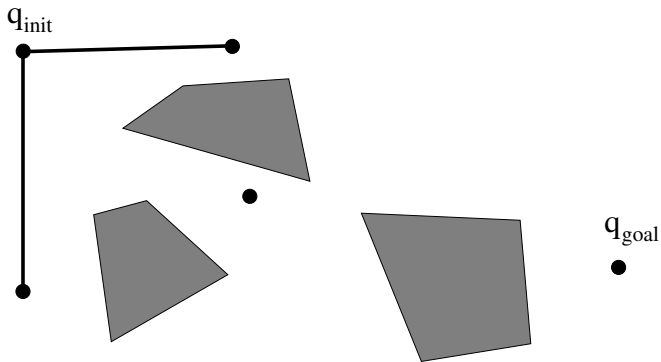


## Probabilistic roadmap (PRM) 1994

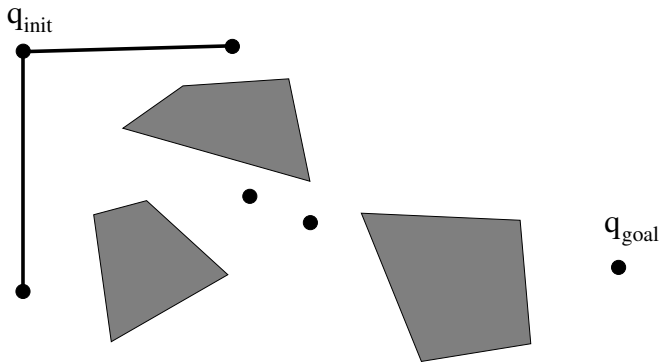




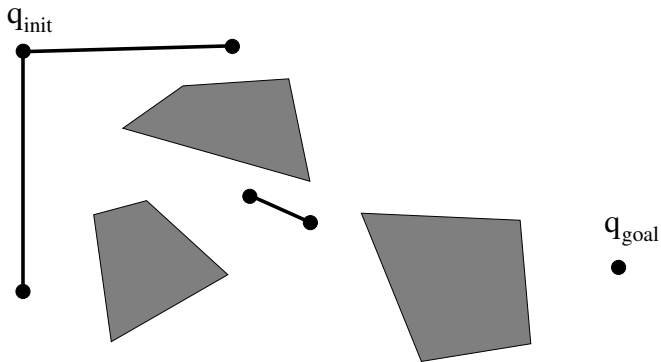
## Probabilistic roadmap (PRM) 1994



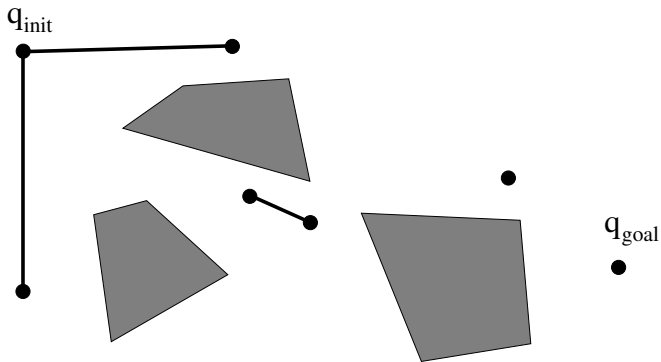
## Probabilistic roadmap (PRM) 1994



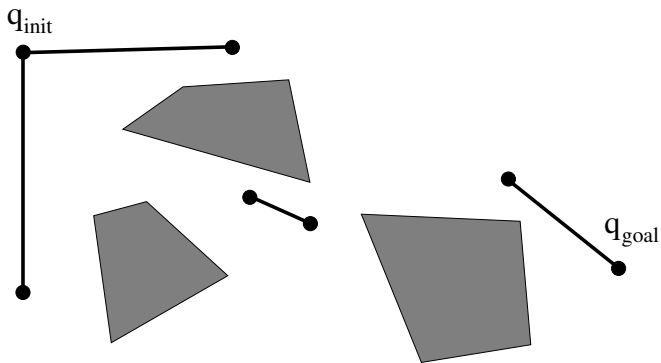
## Probabilistic roadmap (PRM) 1994



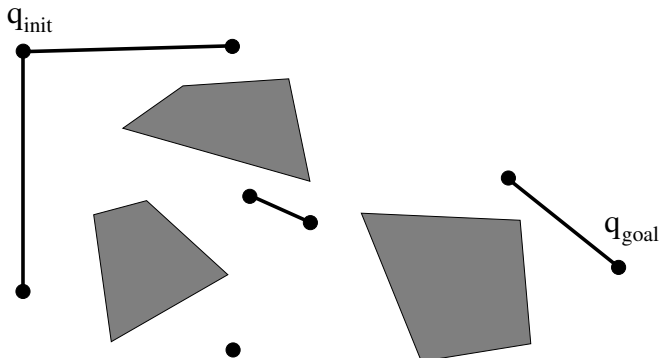
## Probabilistic roadmap (PRM) 1994



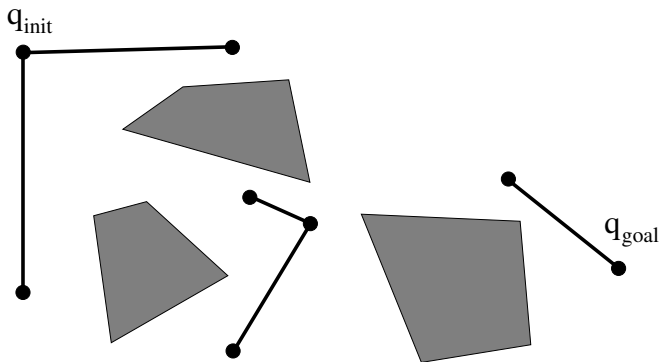
## Probabilistic roadmap (PRM) 1994



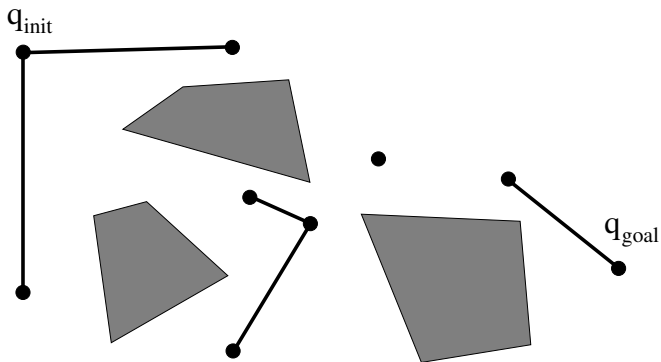
## Probabilistic roadmap (PRM) 1994



## Probabilistic roadmap (PRM) 1994

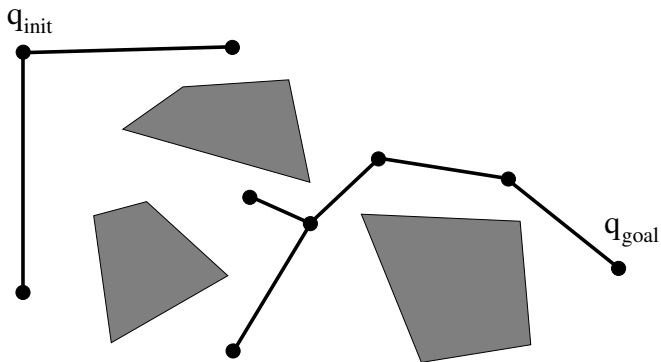


## Probabilistic roadmap (PRM) 1994

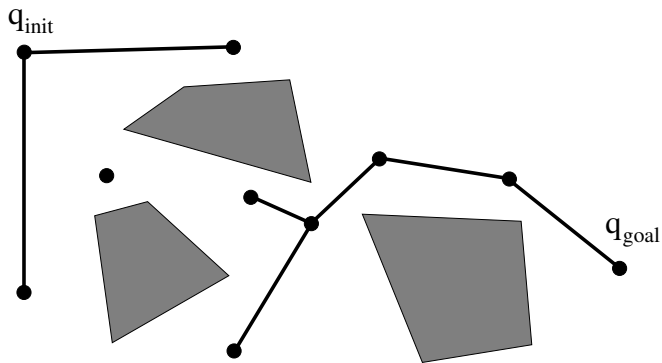




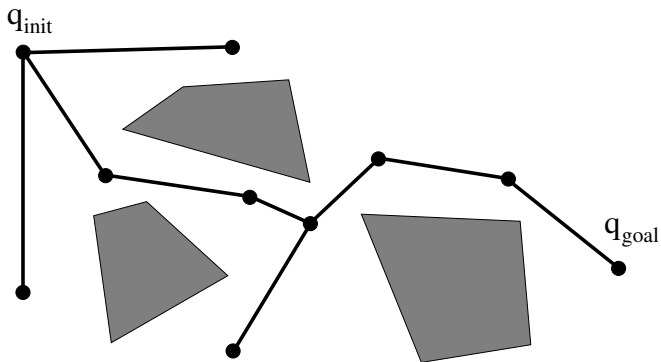
## Probabilistic roadmap (PRM) 1994



## Probabilistic roadmap (PRM) 1994



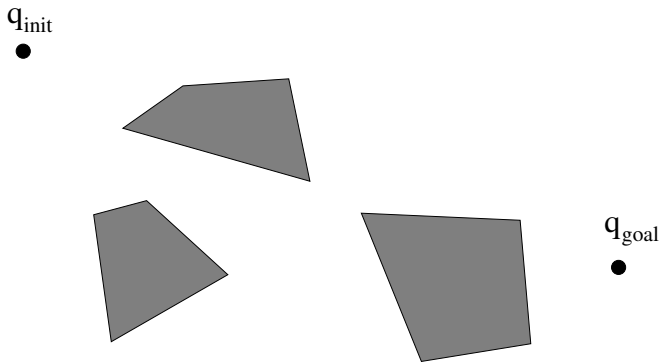
## Probabilistic roadmap (PRM) 1994



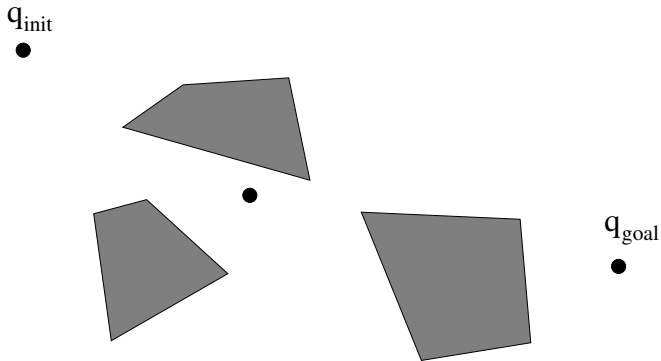
## Probabilistic roadmap (PRM)

- ▶ Numerous useless nodes are created
  - ▶ this makes the connection of new nodes more time consuming
- ▶ Variant : visibility-based PRM
  - ▶ only *interesting* nodes are kept.

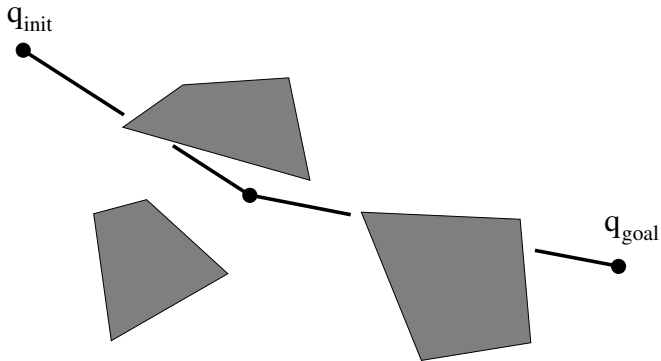
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



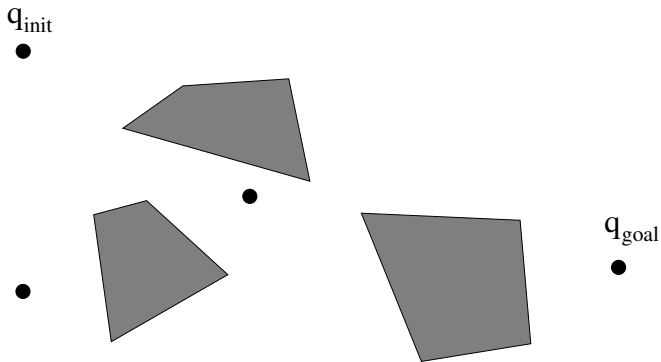
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



## Visibility-based probabilistic roadmap (Visi-PRM) 1999

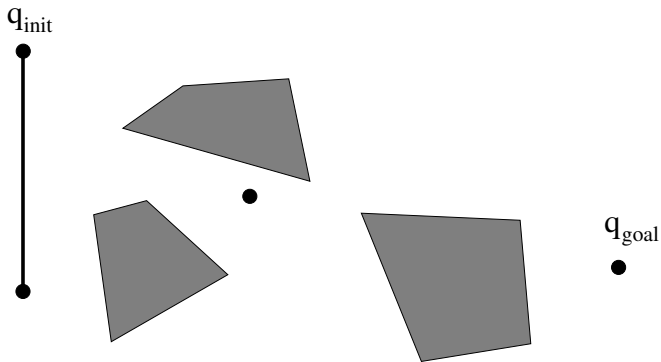


## Visibility-based probabilistic roadmap (Visi-PRM) 1999

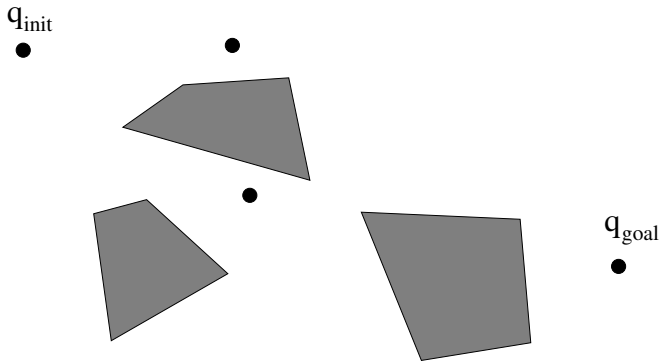




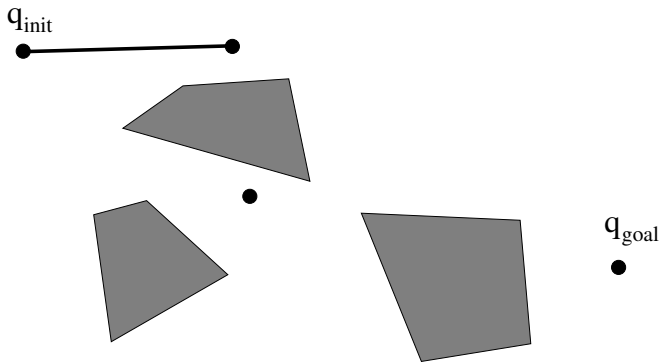
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



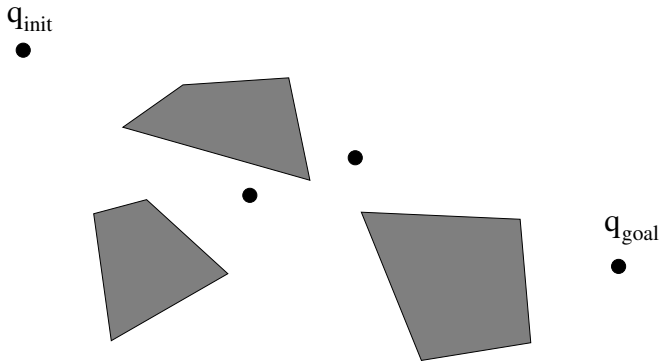
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



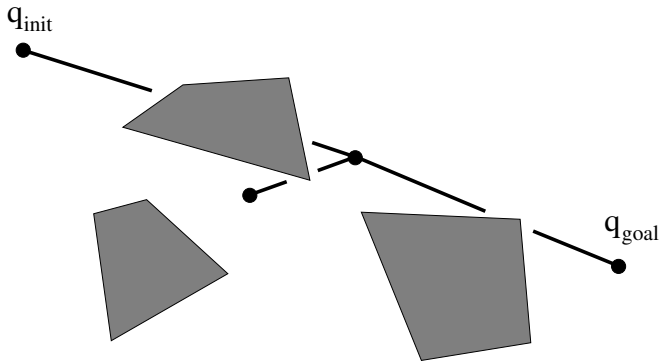
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



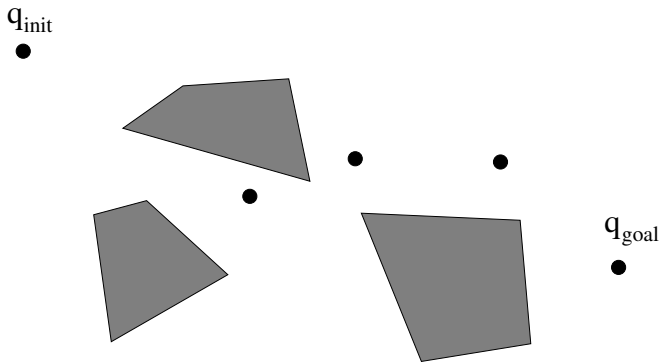
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



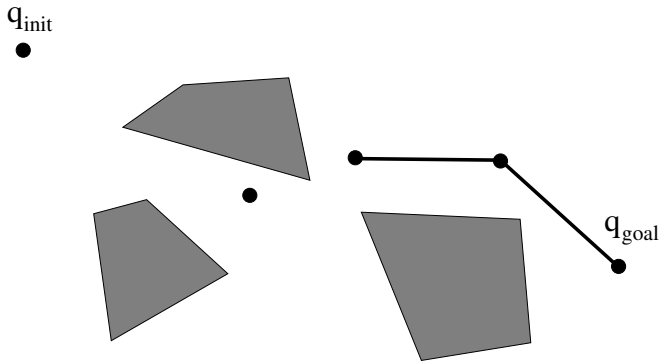
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



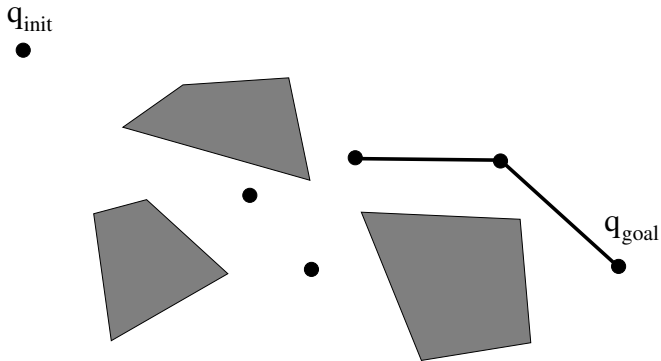
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



## Visibility-based probabilistic roadmap (Visi-PRM) 1999

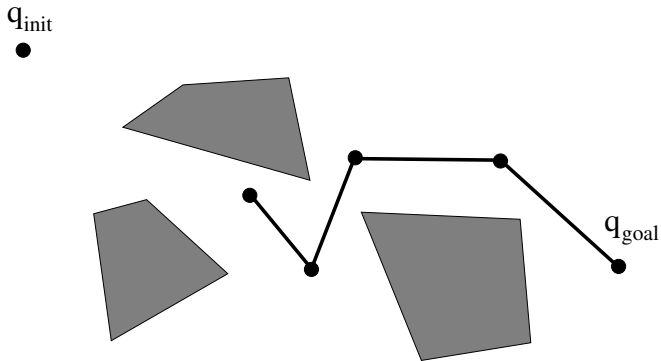


## Visibility-based probabilistic roadmap (Visi-PRM) 1999

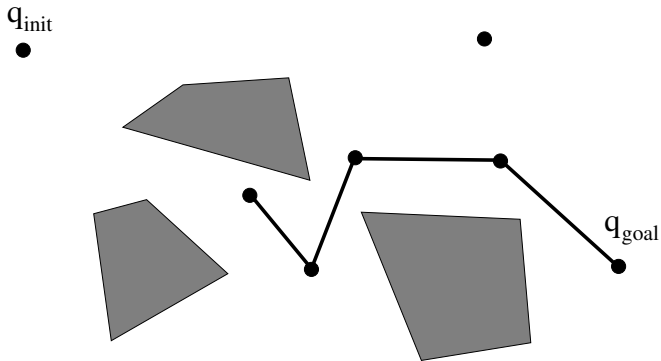




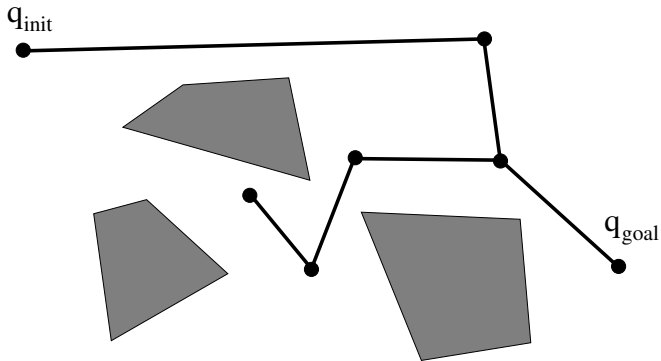
## Visibility-based probabilistic roadmap (Visi-PRM) 1999



## Visibility-based probabilistic roadmap (Visi-PRM) 1999

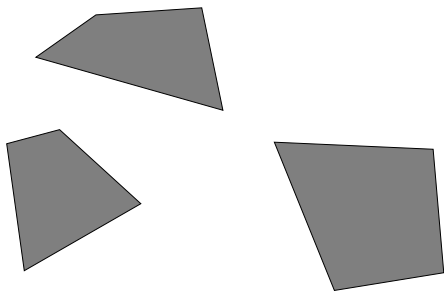


## Visibility-based probabilistic roadmap (Visi-PRM) 1999



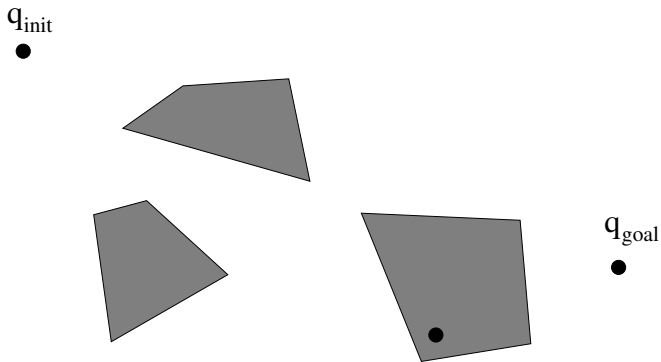
## Rapidly exploring Random Tree (RRT) 2000

$q_{init}$   
●

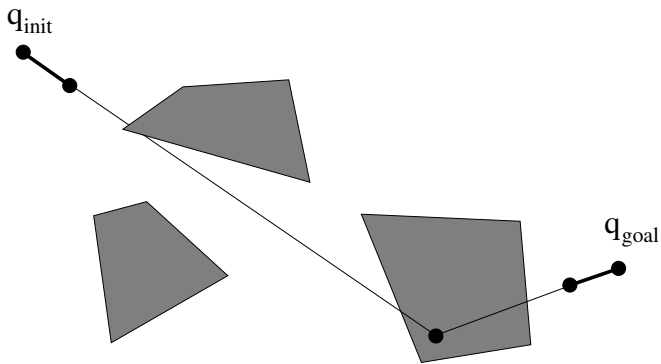


$q_{goal}$   
●

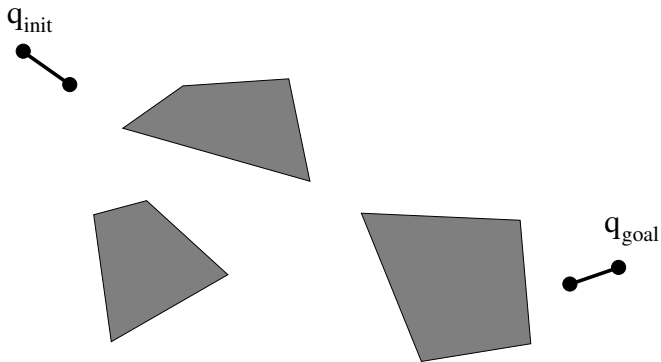
## Rapidly exploring Random Tree (RRT) 2000



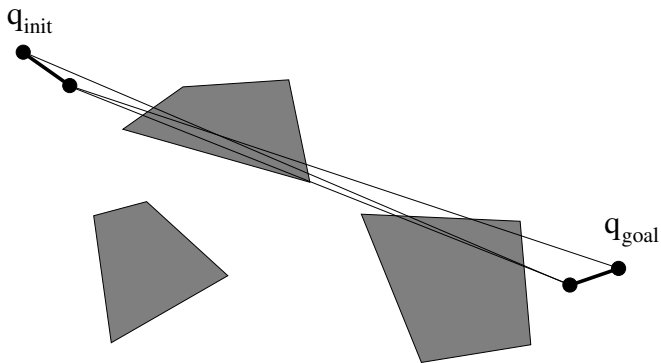
## Rapidly exploring Random Tree (RRT) 2000



## Rapidly exploring Random Tree (RRT) 2000

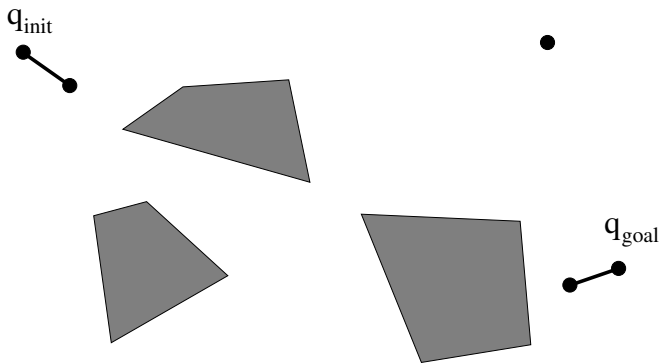


## Rapidly exploring Random Tree (RRT) 2000

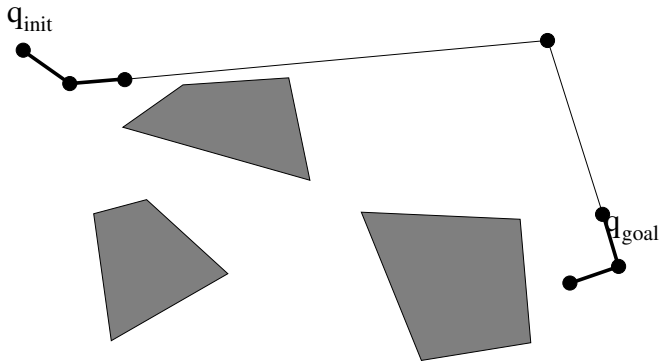




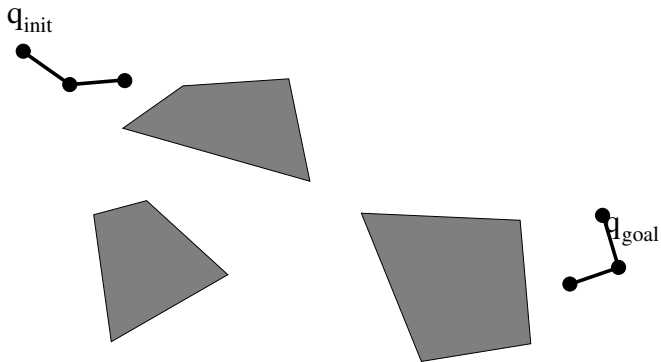
## Rapidly exploring Random Tree (RRT) 2000



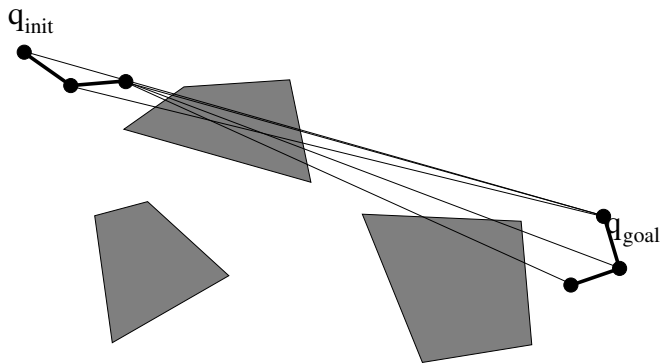
## Rapidly exploring Random Tree (RRT) 2000



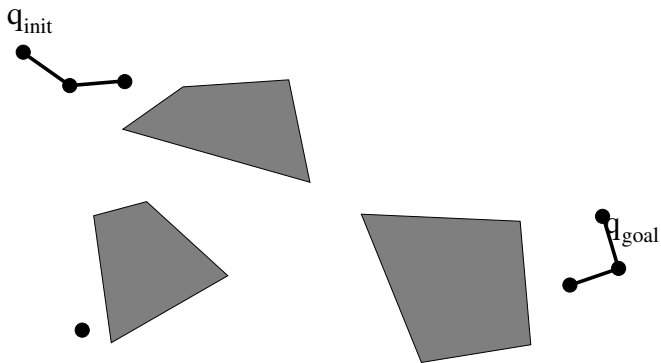
## Rapidly exploring Random Tree (RRT) 2000



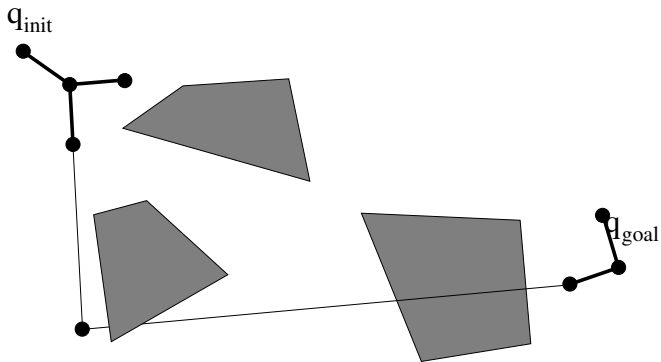
## Rapidly exploring Random Tree (RRT) 2000



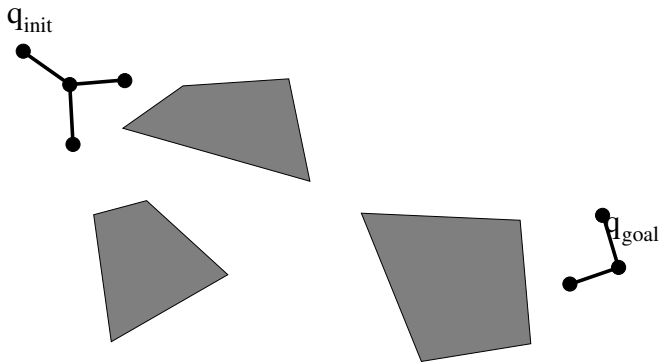
## Rapidly exploring Random Tree (RRT) 2000



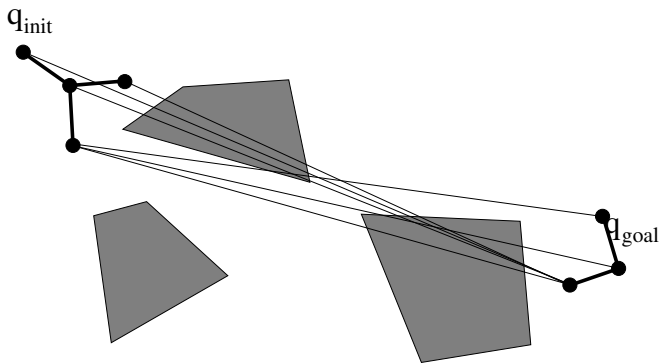
# Rapidly exploring Random Tree (RRT) 2000



## Rapidly exploring Random Tree (RRT) 2000

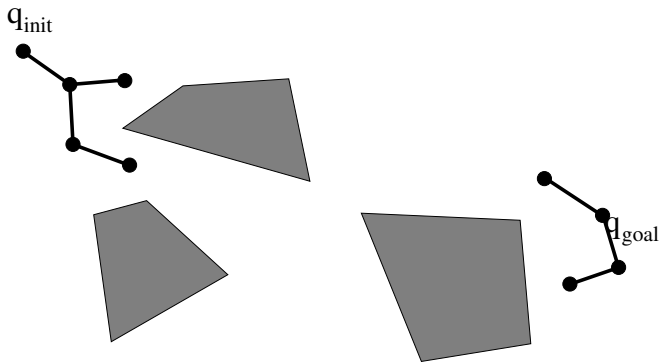


# Rapidly exploring Random Tree (RRT) 2000

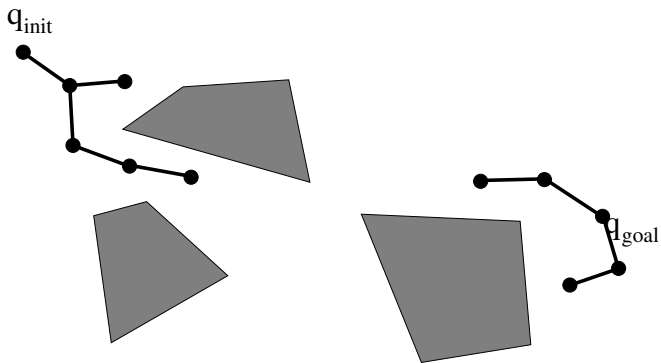




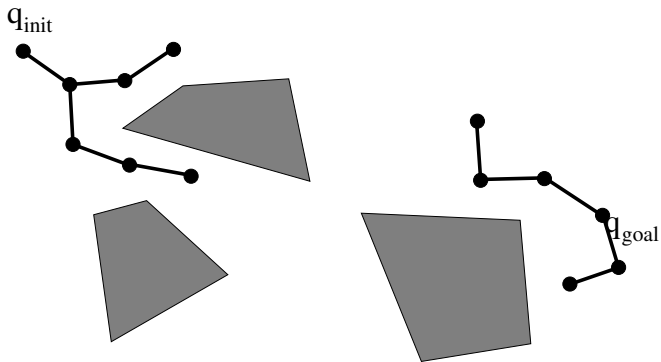
## Rapidly exploring Random Tree (RRT) 2000



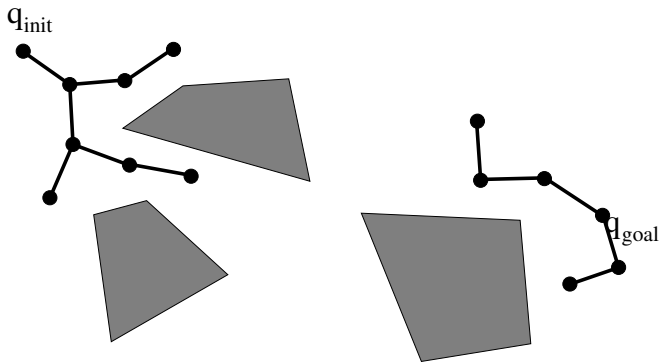
## Rapidly exploring Random Tree (RRT) 2000



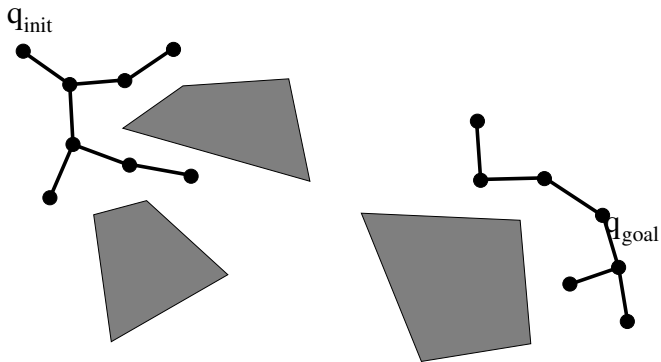
## Rapidly exploring Random Tree (RRT) 2000



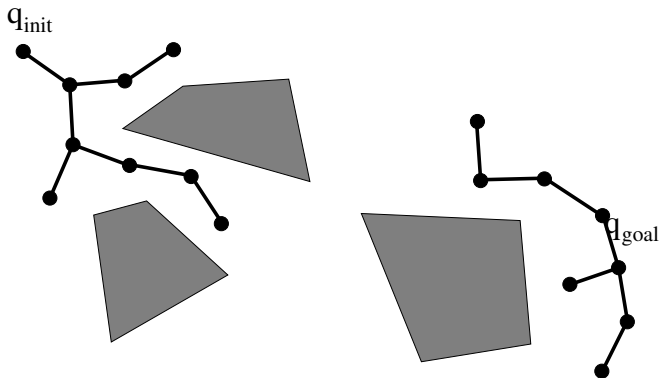
## Rapidly exploring Random Tree (RRT) 2000



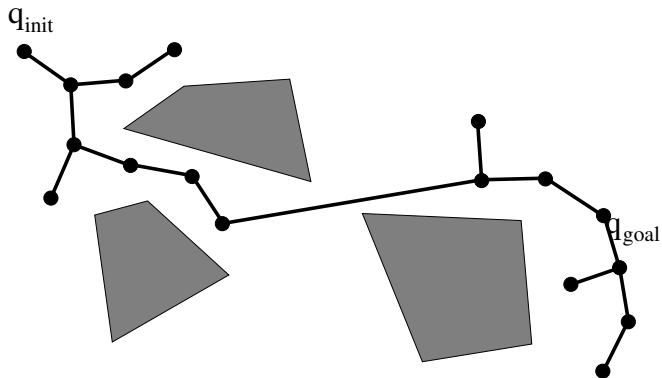
## Rapidly exploring Random Tree (RRT) 2000



## Rapidly exploring Random Tree (RRT) 2000



## Rapidly exploring Random Tree (RRT) 2000



## Random sampling

- ▶ Pros :
  - ▶ no explicit computation of the configuration space
  - ▶ easy to implement,
  - ▶ robust.
- ▶ Cons :
  - ▶ no completeness, only *probrabilistic* completeness
  - ▶ difficult to find narrow passages.
- ▶ required operators :
  - ▶ collision checking
    - ▶ for configurations (static)
    - ▶ for linear interpolation (dynamic)



## Random sampling

- ▶ Pros :
  - ▶ no explicit computation of the configuration space
  - ▶ easy to implement,
  - ▶ robust.
- ▶ Cons :
  - ▶ no completeness, only *probrabilistic* completeness
  - ▶ difficult to find narrow passages.
- ▶ required operators :
  - ▶ collision checking
    - ▶ for configurations (static)
    - ▶ for linear interpolation (dynamic)

## Random sampling

- ▶ Pros :
  - ▶ no explicit computation of the configuration space
  - ▶ easy to implement,
  - ▶ robust.
- ▶ Cons :
  - ▶ no completeness, only *probrabilistic* completeness
  - ▶ difficult to find narrow passages.
- ▶ required operators :
  - ▶ collision checking
    - ▶ for configurations (static)
    - ▶ for linear interpolation (dynamic)

## Asymptotically optimal random sampling

Variants of PRM and RRT exist, and are asymptotically optimal :

- ▶ when the number of nodes tends to infinity,
- ▶ the solution computed by the algorithm tends to the optimal collision-free path.

# PRM\*

## PRM

```

V ← ∅, E ← ∅
for i ∈ {0, …, n} do
    xrand ← SampleFree;
    U ← G.Near(xrand, r)
    for all u ∈ U in order of increasing ||u - xrand|| do
        if xrand and u in different
            connected components
        then
            TryConnect (xrand, u)
        end if
    end for
end for
    
```

## PRM\*

```

V ← SampleFreei=1, …, n, E ← ∅
for v ∈ V do
    U ← G.Near(v, r*) \ v
    for all u ∈ U do
        TryConnect (v, u)
    end for
end for
r* = γPRM (log(n)/n)1/d
    
```

# PRM\*

## PRM

```

V ← ∅, E ← ∅
for i ∈ {0, …, n} do
    xrand ← SampleFree;
    U ← G.Near(xrand, r)
    for all u ∈ U in order of increasing ||u - xrand|| do
        if xrand and u in different
           connected components
        then
            TryConnect (xrand, u)
        end if
    end for
end for
    
```

## PRM\*

```

V ← SampleFreei=1, …, n, E ← ∅
for v ∈ V do
    U ← G.Near(v, r*) \ v
    for all u ∈ U do
        TryConnect (v, u)
    end for
end for
r* = γPRM (log(n)/n)1/d
    
```

# kPRM\*

## kPRM

```

V ← ∅, E ← ∅
for i ∈ {0, …, n} do
    xrand ← SampleFree;
    U ← G.Nearest(xrand, k)
    for all u ∈ U in order of increasing ||u - xrand|| do
        if xrand and u in different
            connected components
        then
            TryConnect (xrand, u)
        end if
    end for
end for
    
```

## kPRM\*

```

V ← SampleFreei=1, …, n, E ← ∅
for v ∈ V do
    U ← G.Nearest(v, k*) \ v
    for all u ∈ U do
        TryConnect (v, u)
    end for
end for
k* = kPRM log(n), kPRM > e(1 + 1/d)
    
```

# kPRM\*

## kPRM

```

V ← ∅, E ← ∅
for i ∈ {0, …, n} do
  xrand ← SampleFree;
  U ← G.Nearest(xrand, k)
  for all u ∈ U in order of increasing ||u - xrand|| do
    if xrand and u in different
       connected components
    then
      TryConnect (xrand, u)
    end if
  end for
end for
    
```

## kPRM\*

```

V ← SampleFreei=1, …, n, E ← ∅
for v ∈ V do
  U ← G.Nearest(v, k*) \ v
  for all u ∈ U do
    TryConnect (v, u)
  end for
end for
k* = kPRM log(n), kPRM > e(1 + 1/d)
    
```

# PRM\*, kPRM\*

Note that :

- ▶ PRM\*, kPRM\* are not iterative anymore,
- ▶ making them iterative is not trivial.



## Rapidly Exploring Random trees

There exists also asymptotically optimal variants of RRT

- ▶ RRG, RRT\*

but they are specific to a given problem ( $\mathbf{q}_{init}, \mathbf{q}_{goal}$ ).

## Collision tests

- ▶ static : for configurations
    - ▶ problem : given
      - ▶ two rigid objects made of triangles
      - ▶ the relative position of one with respect to the other one
- determine whether they are colliding.

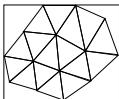
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



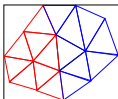
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



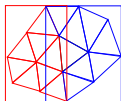
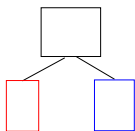
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



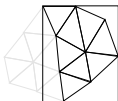
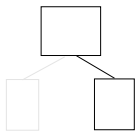
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



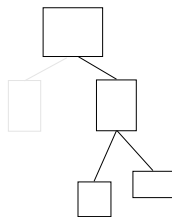
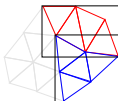
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



## Bounding volume hierarchies

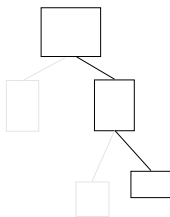
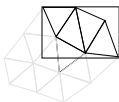
- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.





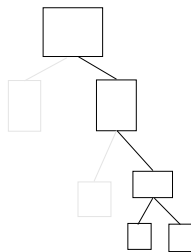
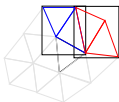
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



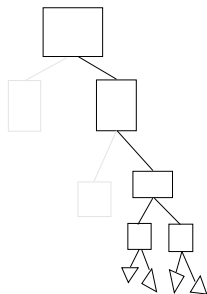
## Bounding volume hierarchies

- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



## Bounding volume hierarchies

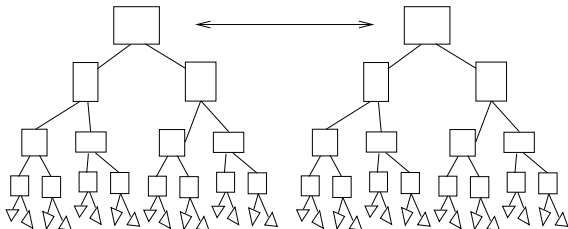
- ▶ binary tree of bounding volumes such that
  - ▶ each node has two children,
  - ▶ leaves are triangles.



## Collision testing for configurations

### ► Algorithm

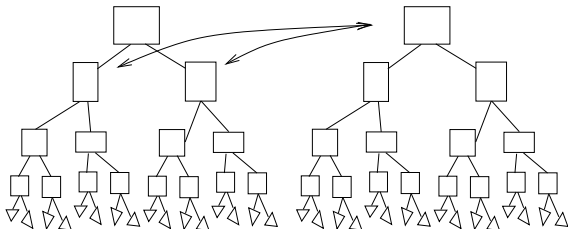
- test root nodes of each tree,
- if two bounding volumes collide, test one with the children of the other one.



## Collision testing for configurations

### ► Algorithm

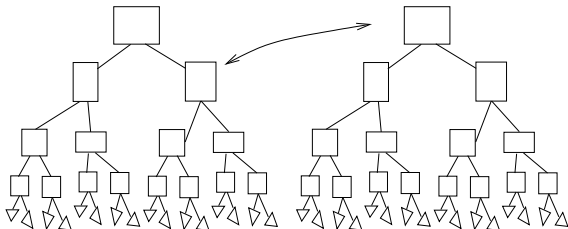
- test root nodes of each tree,
- if two bounding volumes collide, test one with the children of the other one.



## Collision testing for configurations

### ► Algorithm

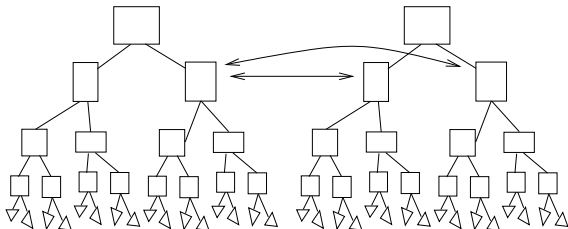
- test root nodes of each tree,
- if two bounding volumes collide, test one with the children of the other one.



## Collision testing for configurations

### ► Algorithm

- test root nodes of each tree,
- if two bounding volumes collide, test one with the children of the other one.



## Open source software platform

Several open-source platforms for motion planning are available

- ▶ OMPL (Rice University)
  - ▶ no kinematic chain,
  - ▶ no collision checking.
- ▶ Openrave (CMU)
- ▶ MoveIt (ROS)
  - ▶ Integration in ROS of
  - ▶ fcl (collision checking), KDL (kinematic chain)
- ▶ Humanoid Path Planner
  - ▶ numerical constraints (quasi-static equilibrium)
  - ▶ advanced manipulation planning



# Humanoid Path Planner

<https://humanoid-path-planner.github.io/hpp-doc>