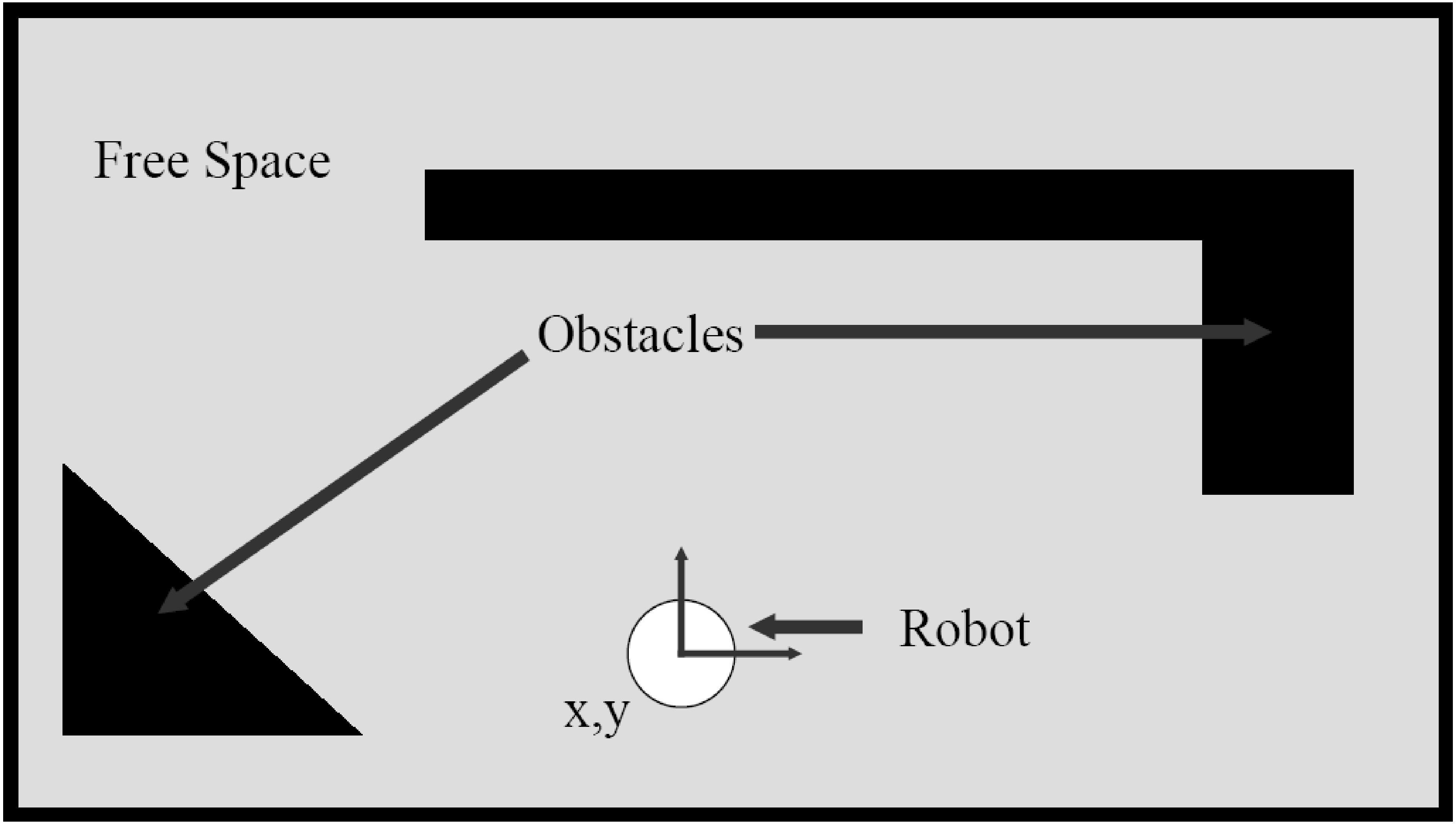
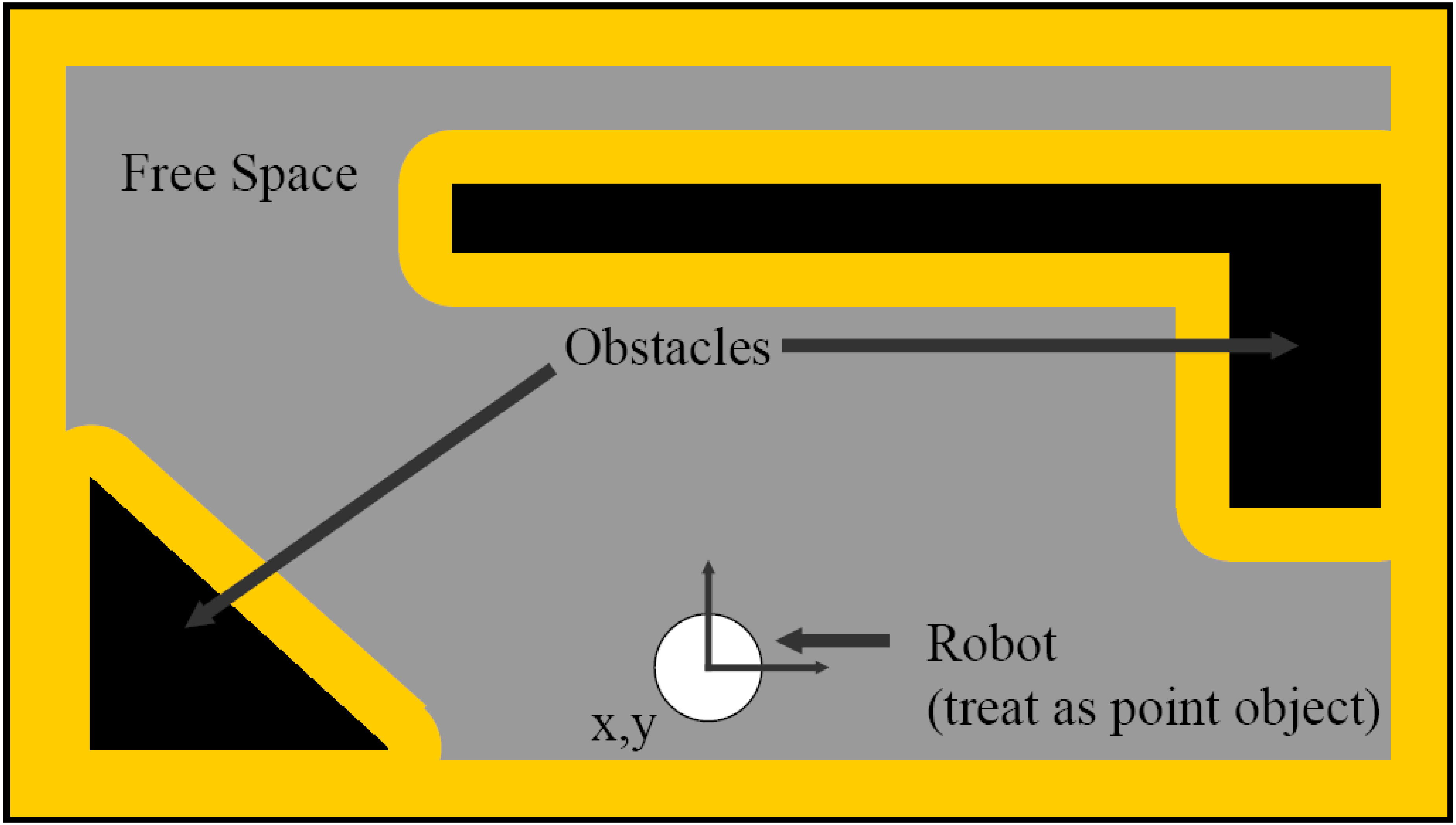
Configuration Space

Configuration Space



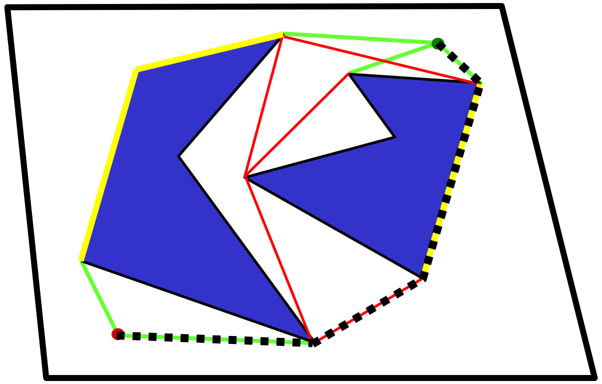
Configuration Space



Definition

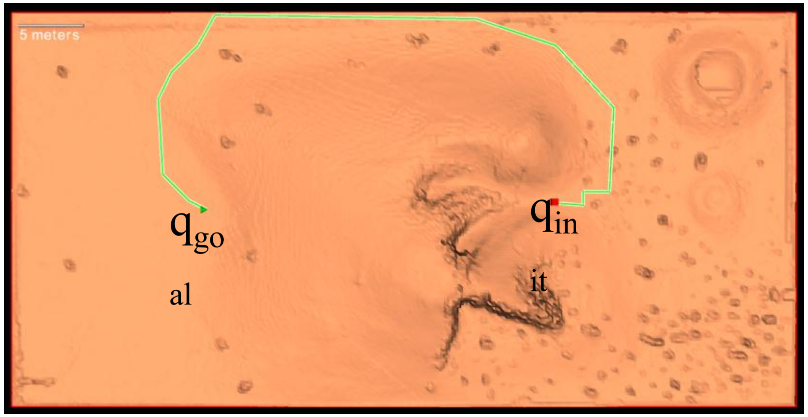
- A robot configuration is a specification of the positions of all robot points relative to a fixed coordinate system
- Usually a configuration is expressed as a "vector" of position/orientation parameters

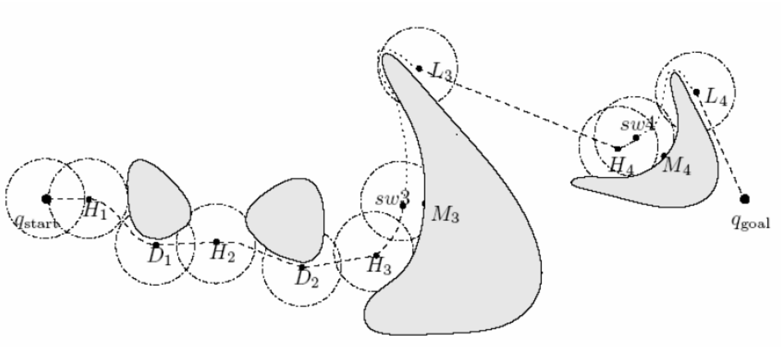
What is a Path?



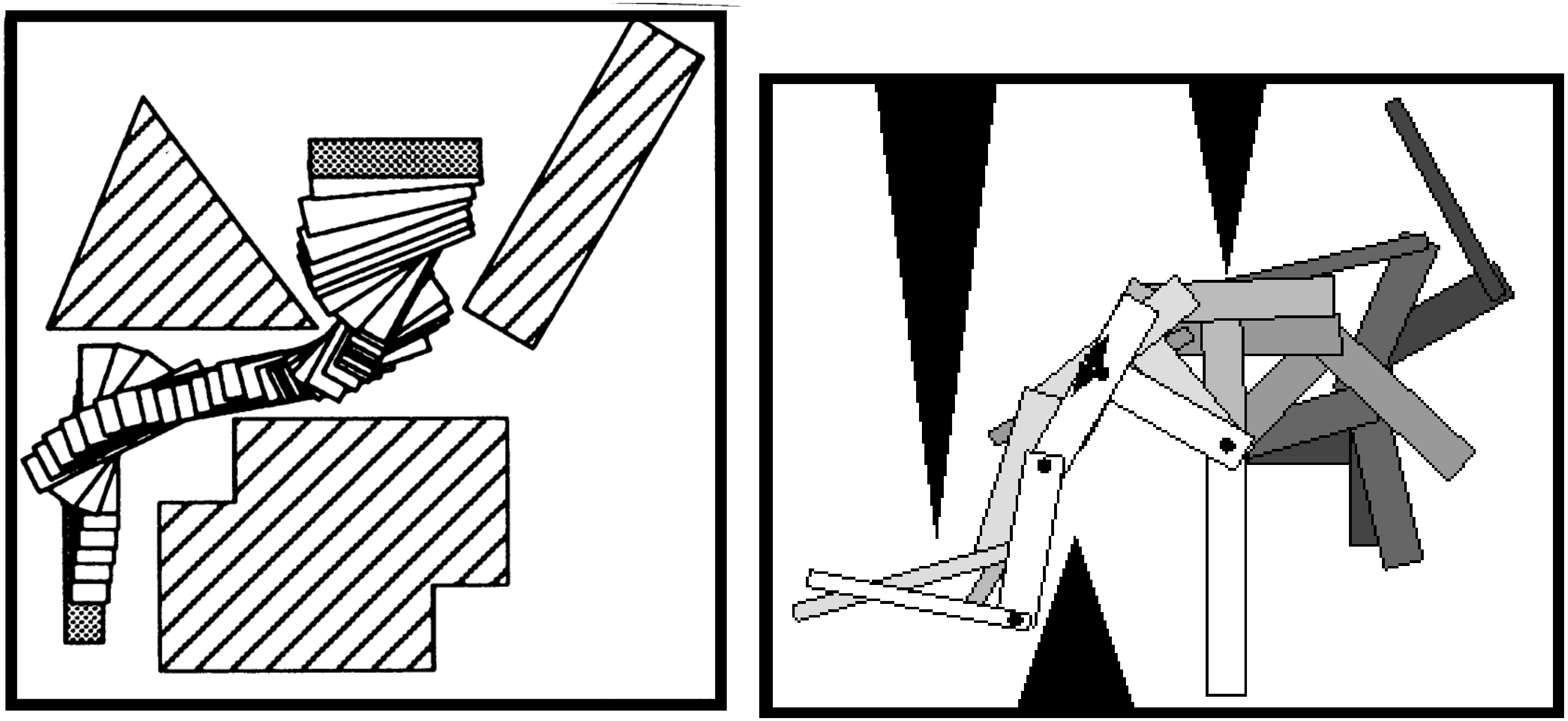
- ● q_{init}
- ● q_{goa}

l

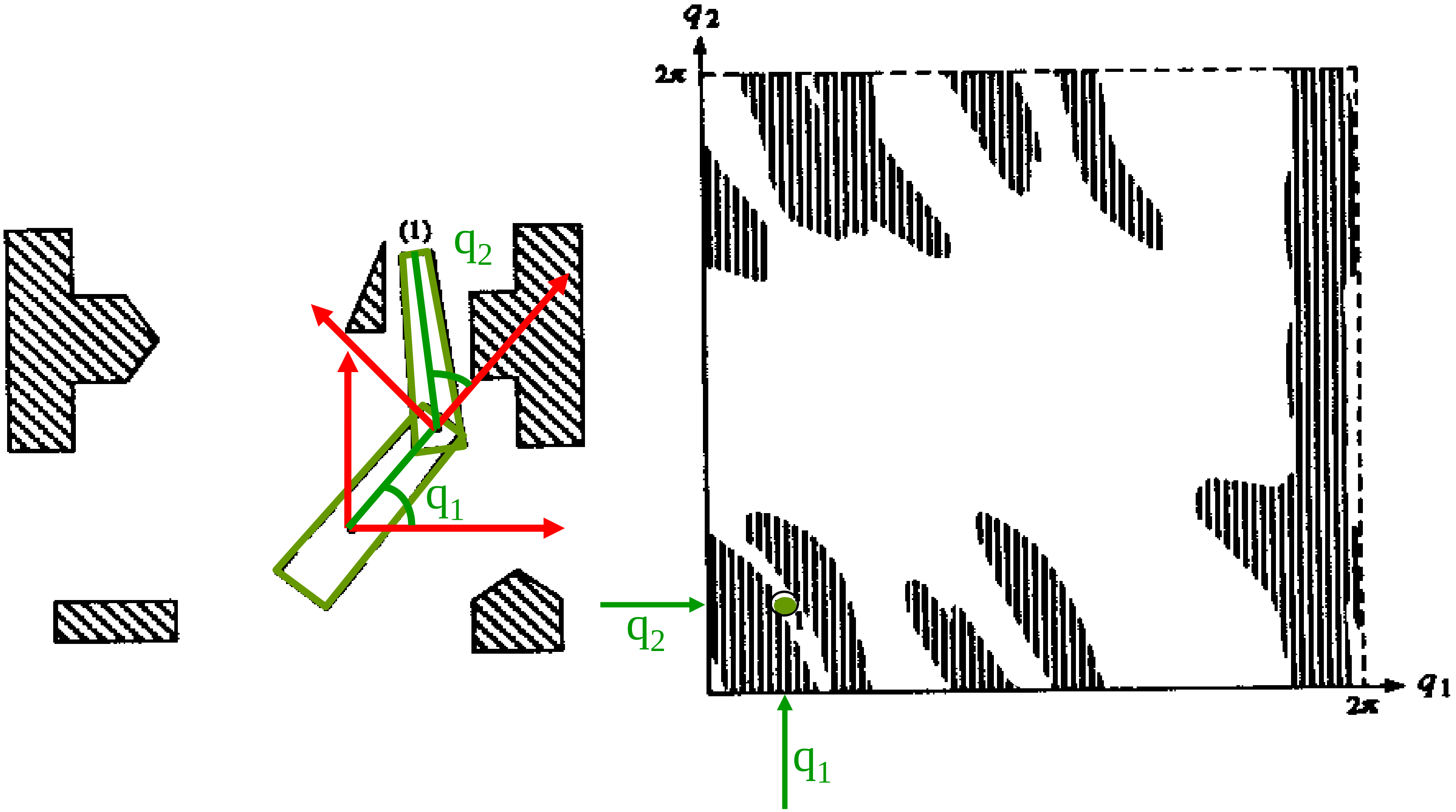




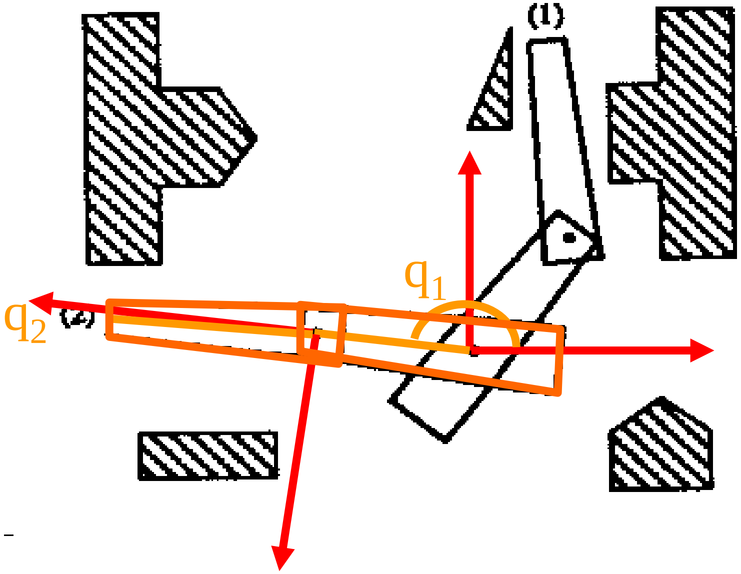
What is a Path?

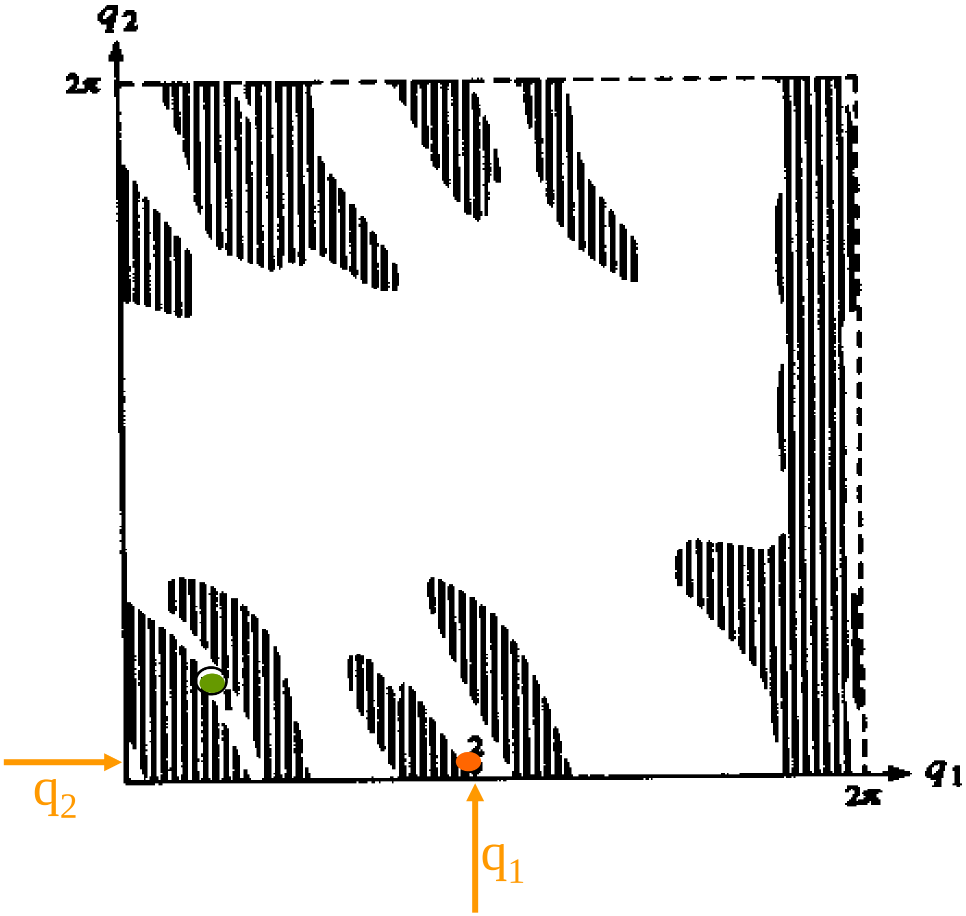


Tool: Configuration Space (C-Space C)

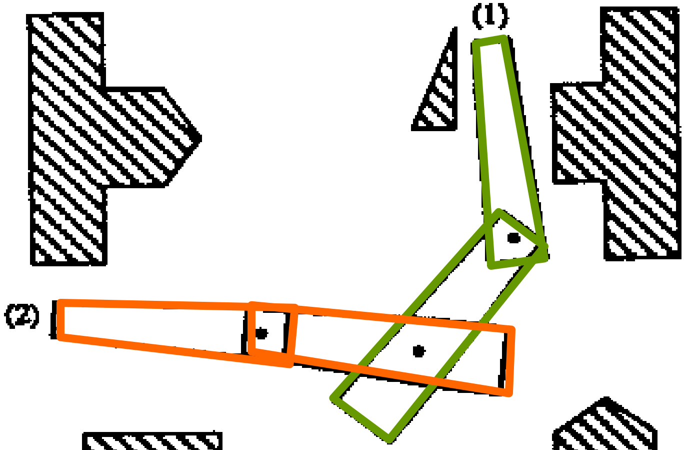


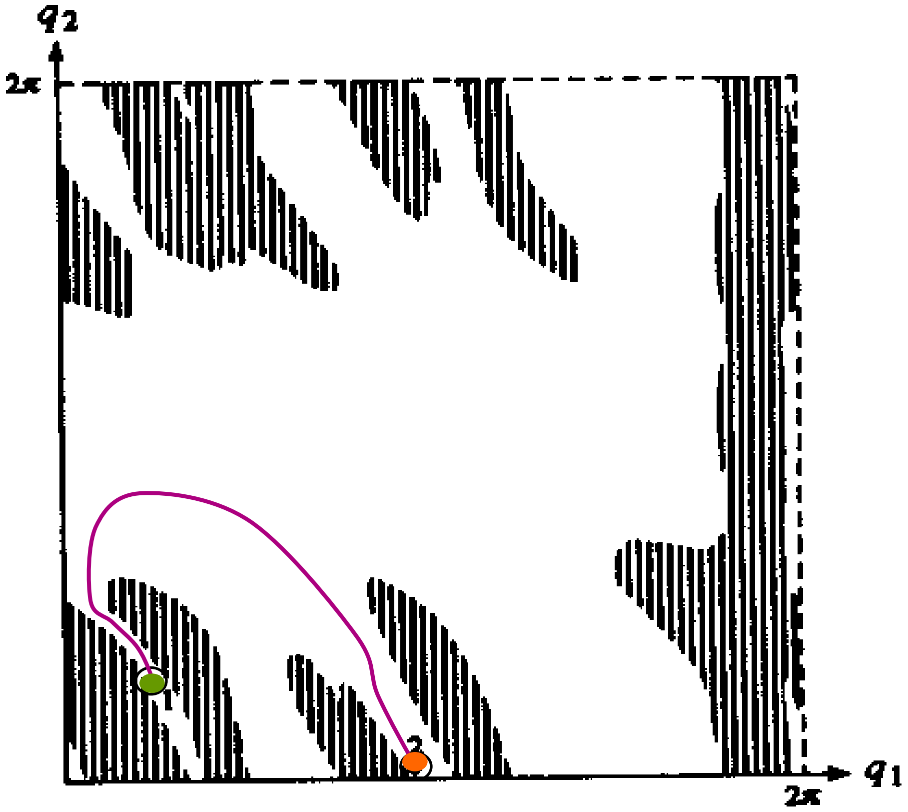
Tool: Configuration Space
(C-Space C)



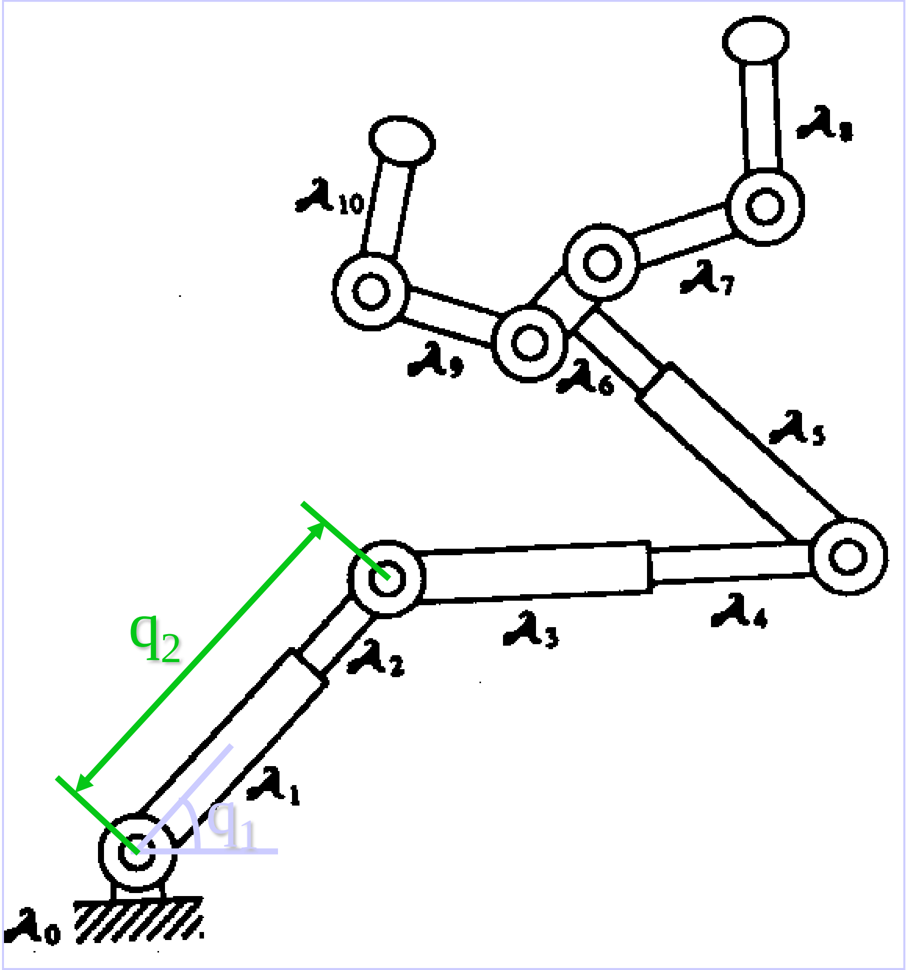


Tool: Configuration Space
(C-Space C)





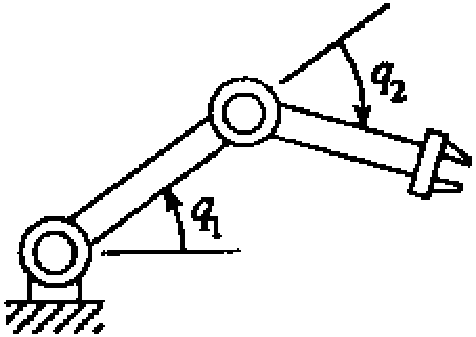
Articulated Robot Example



$$q = (q_1, q_2, \dots, q_{10})$$

Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space



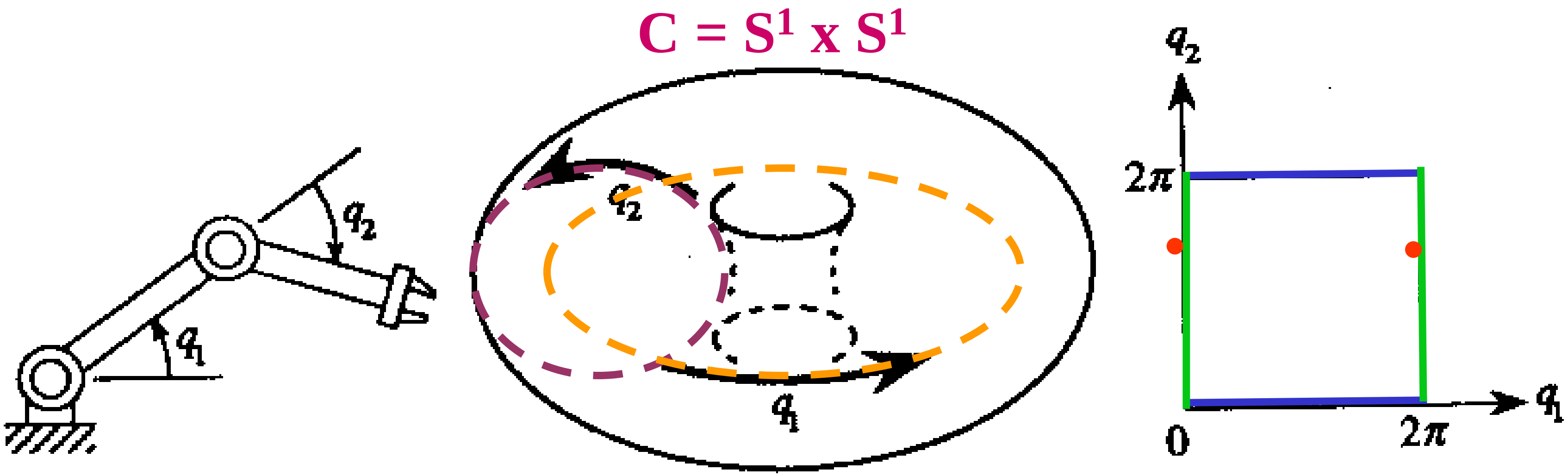
Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space

$C = S^1 \times S^1$

Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space

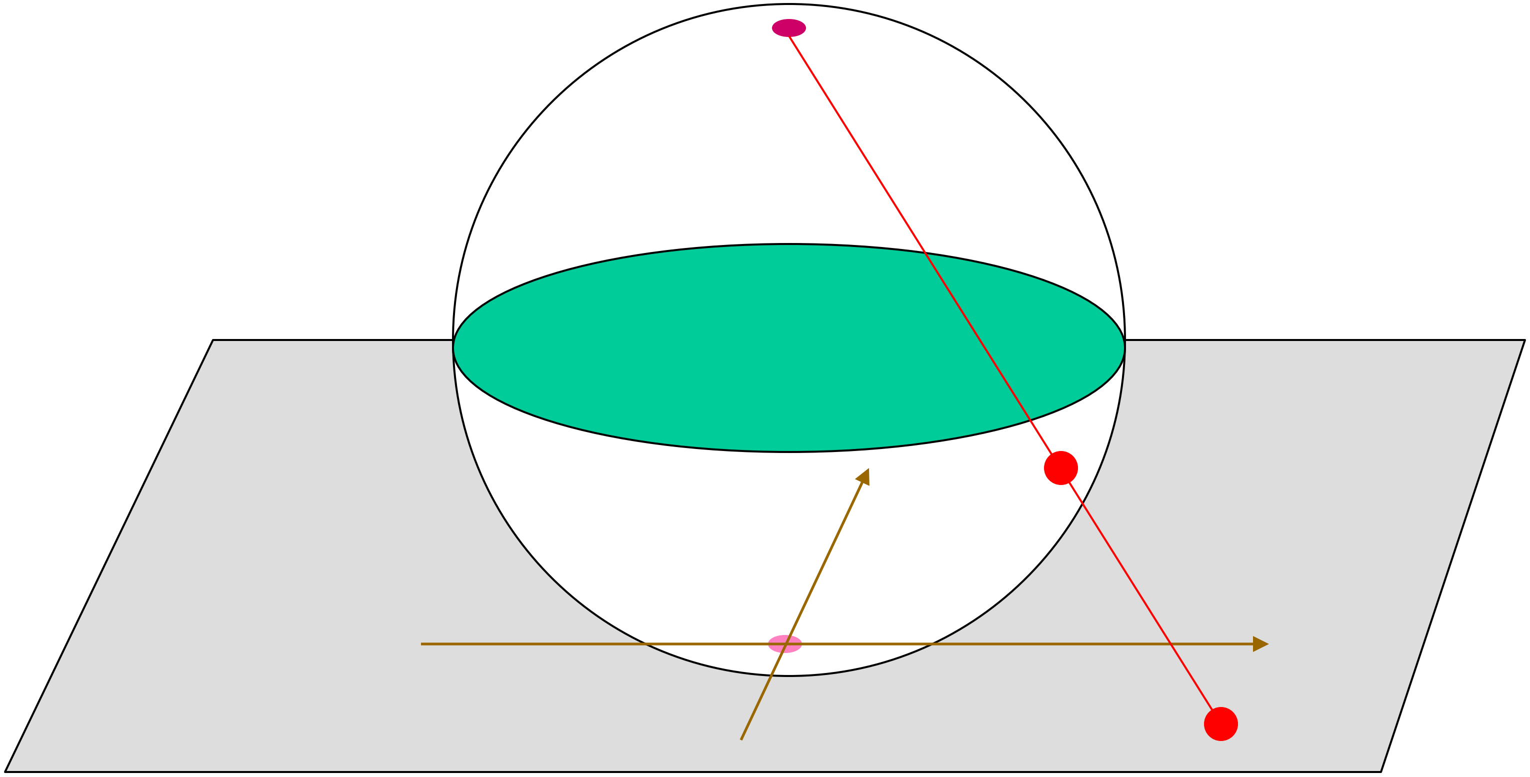


$C = S^1 \text{ x } S^1$

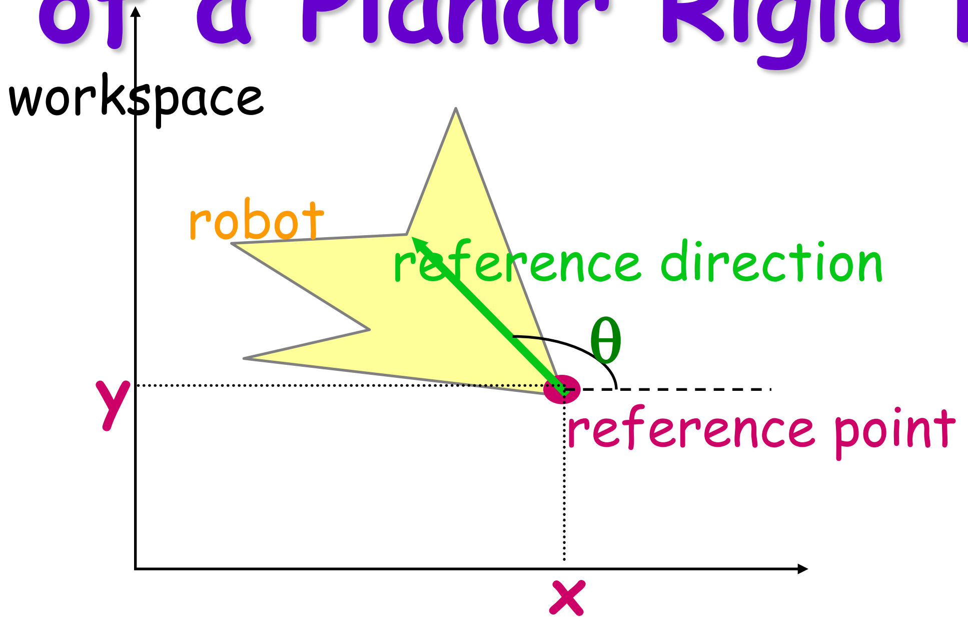
Structure of Configuration Space

- It is a manifold
  For each point q, there is a 1-to-1 map between a neighborhood of q and a Cartesian space $\mathbf{R}^n$, where n is the dimension of C
- This map is a local coordinate system called a chart.
  C can always be covered by a finite number of charts. Such a set is called an atlas

Example



Case of a Planar Rigid Robot



- 3-parameter representation: $q = (x,y,\theta)$ with $\theta \in [0,2\pi)$. Two charts are needed
- Other representation: $q = (x,y,\cos\theta,\sin\theta)$ →c-space is a 3-D cylinder $\mathbf{R}^2 \times S^1$ embedded in a 4-D space

Rigid Robot in 3-D Workspace

- $q = (x,y,z,\alpha,\beta,\gamma)$

The c-space is a 6-D space (manifold) embedded in a 12-D Cartesian space. It is denoted by $R^3xSO(3)$

- Other representation: $q = (x,y,z,r_{11},r_{12},\ldots,r_{33})$ where r_{11}, r_{12}, …, r_{33} are the elements of rotation matrix R:

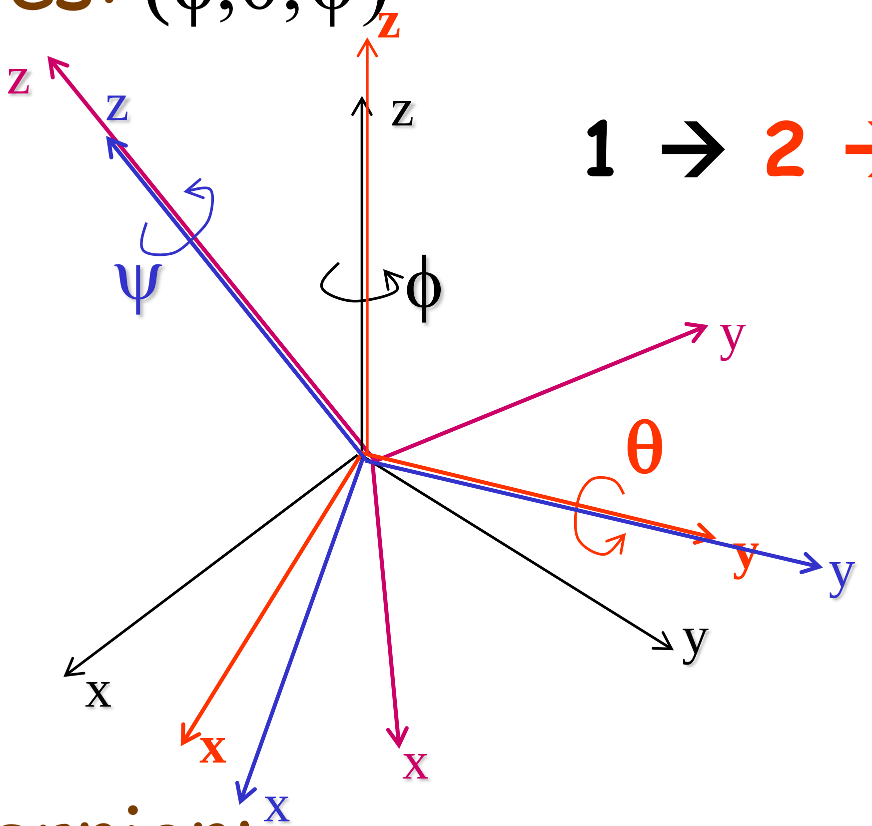
$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

with:

  - $r_{i1}^2+r_{i2}^2+r_{i3}^2 = 1$
  - $r_{i1}r_{j1} + r_{i2}r_{2j} + r_{i3}r_{j3} = 0$
  - $\det(R) = +1$

Parameterization of SO(3)

- Euler angles: (ϕ,θ,ψ)

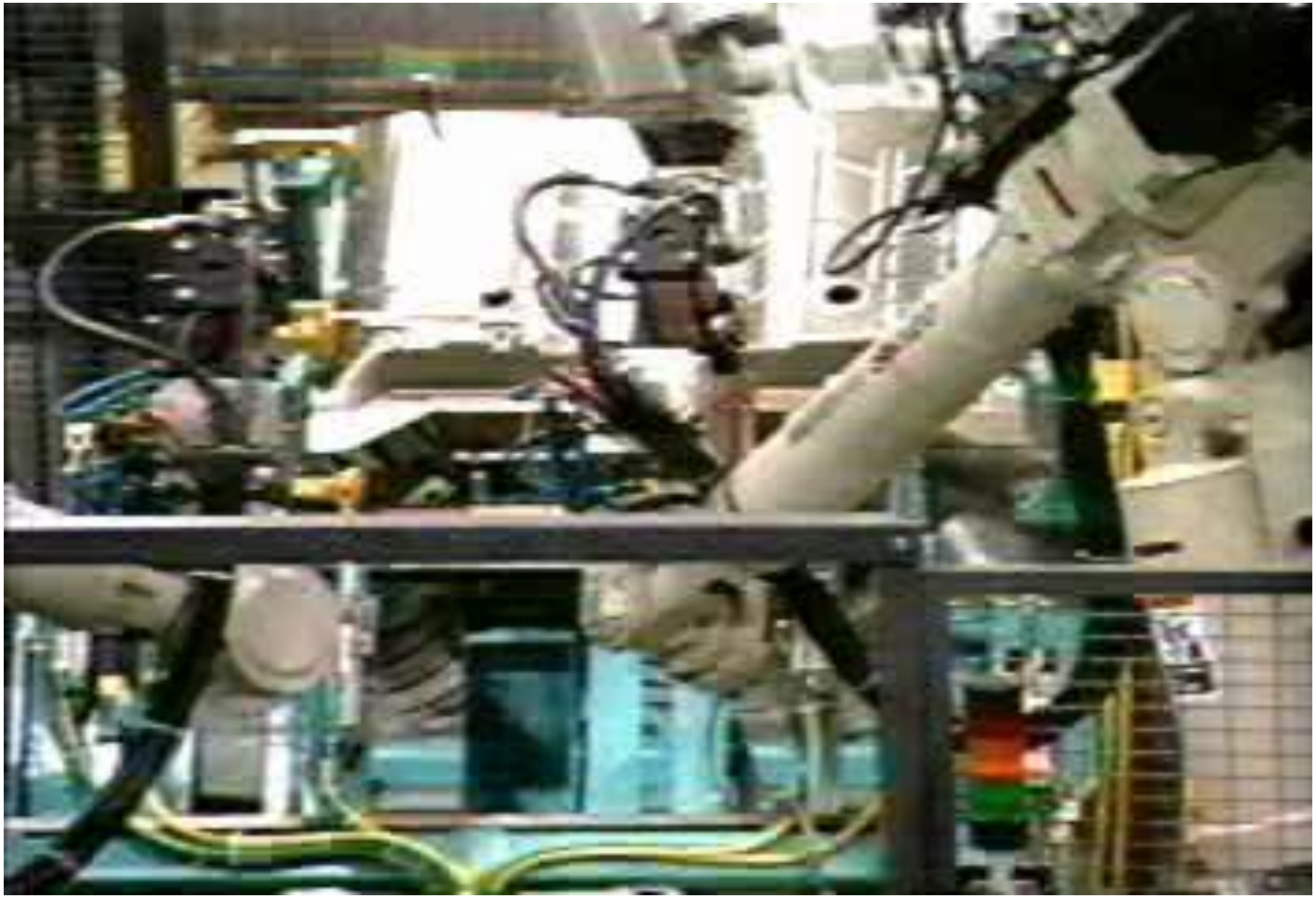


1 → 2 → 3 → 4

- Unit quaternion:

$(\cos\theta/2, n_1 \sin\theta/2, n_2 \sin\theta/2, n_3 \sin\theta/2)$

A welding robot



A Stuart Platform

This image cannot currently be displayed.

Barrett WAM arm

This image cannot currently be displayed.

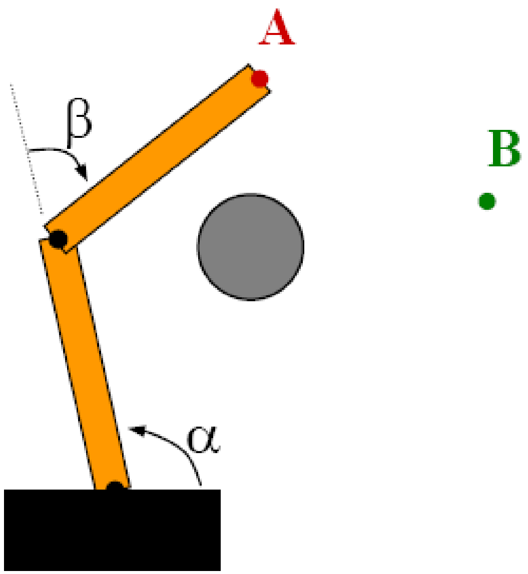
Barrett WAM arm on a mobile platform

This image cannot currently be displayed.

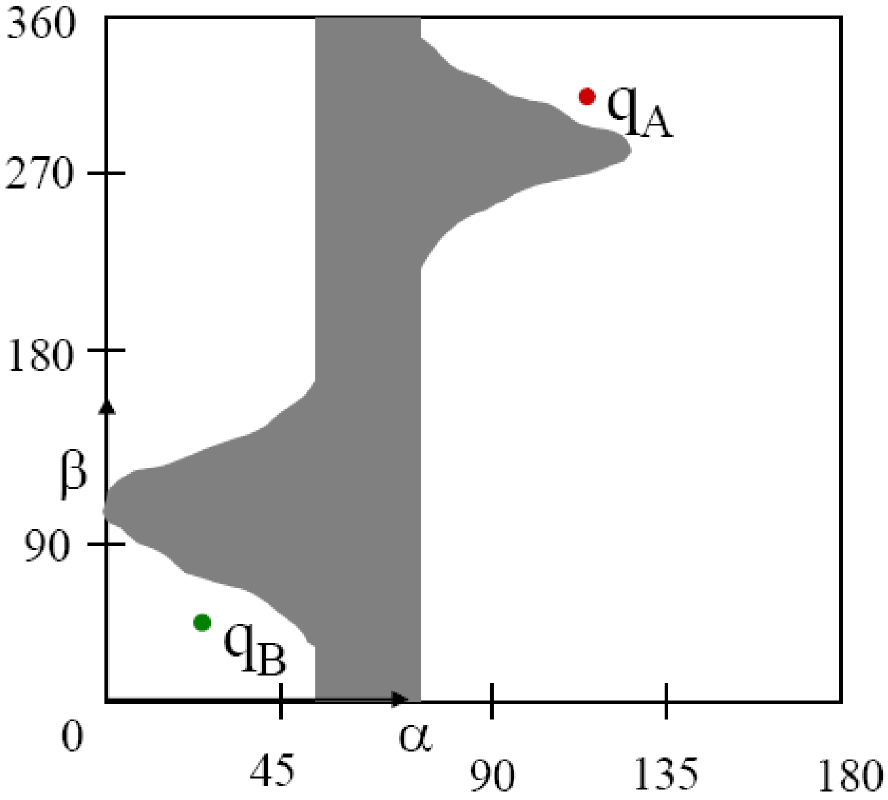
Configuration Space Obstacle

Reference *configuration*

How do we get from **A** to **B** ?

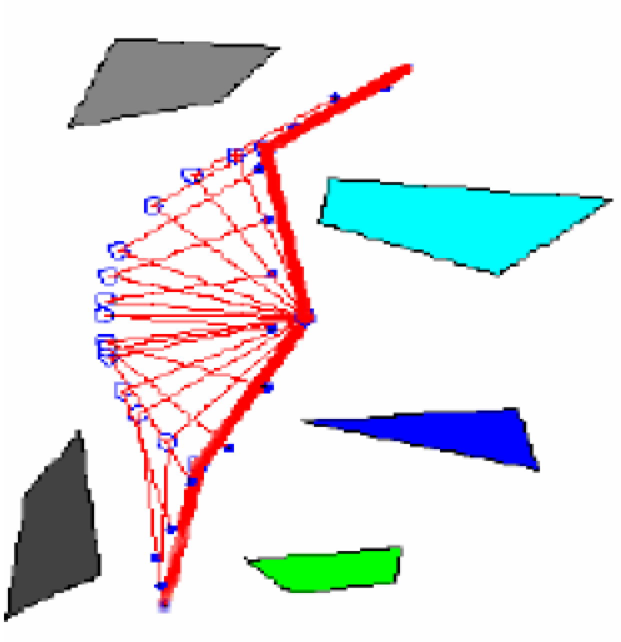


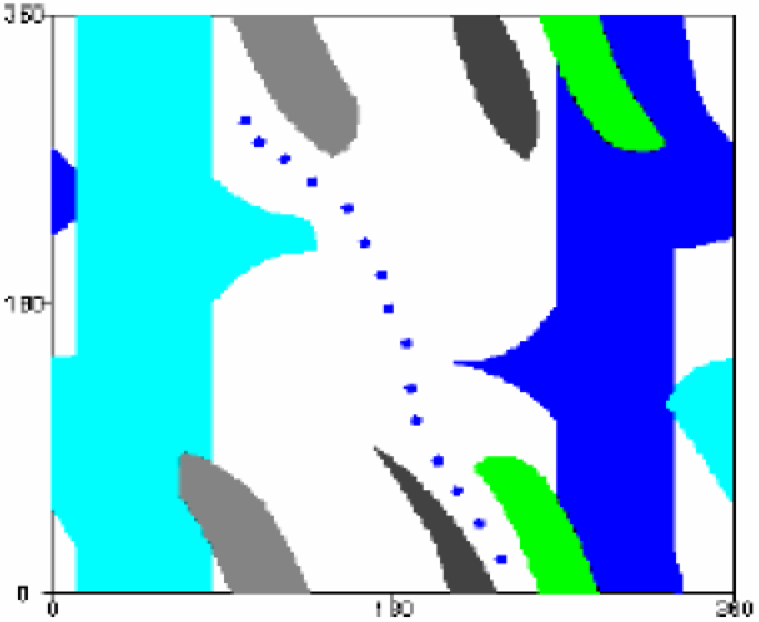
An obstacle in the robot’s workspace



The C-space representation of this obstacle…

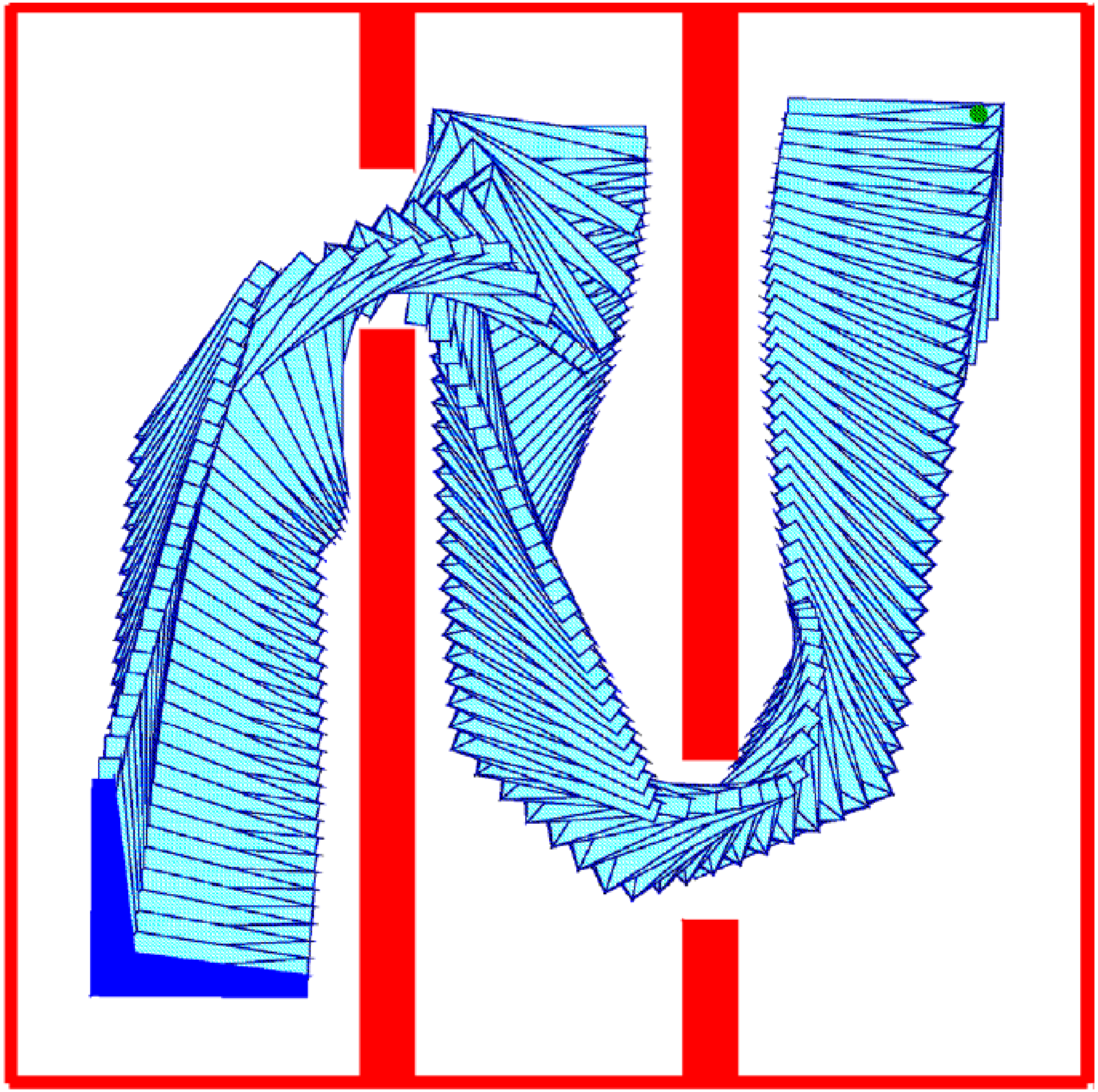
Two link path



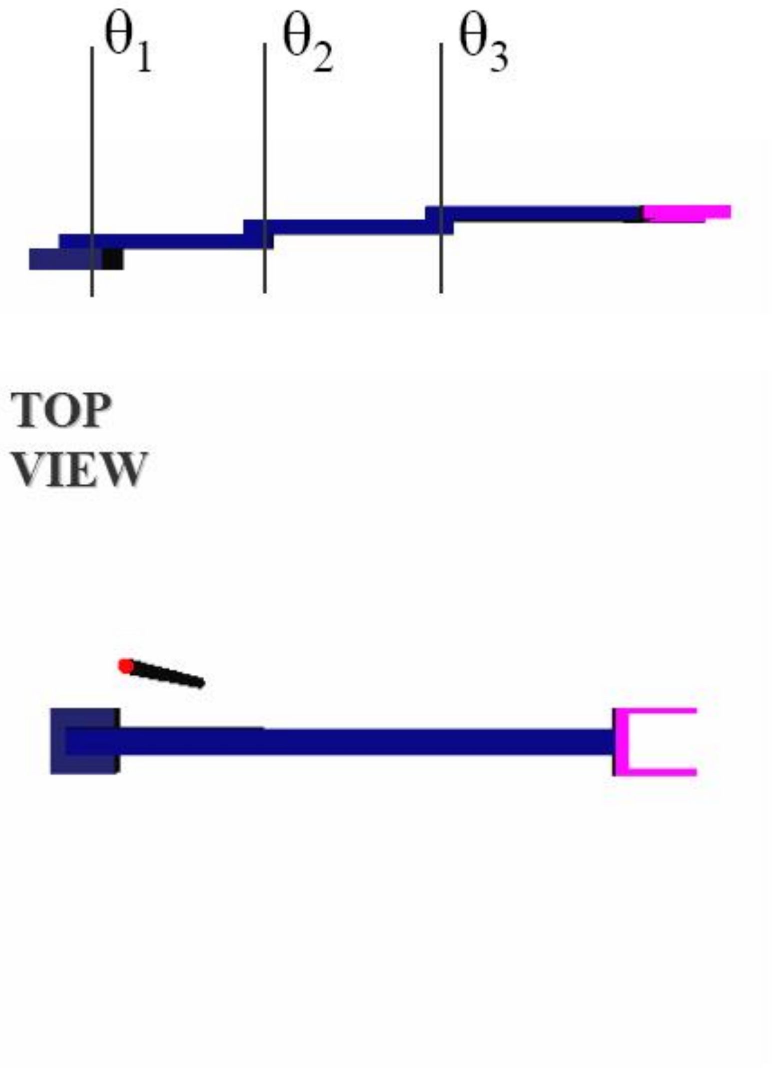


Thanks to Ken Goldberg

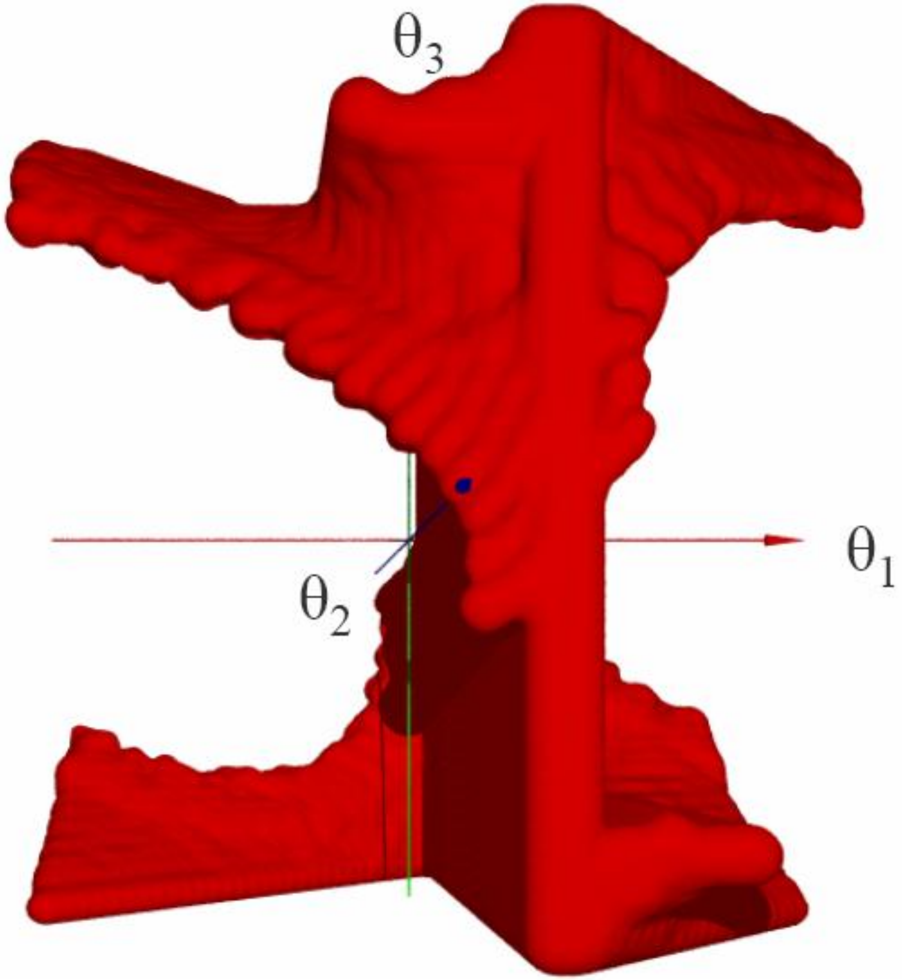
2D Rigid Object



The Configuration Space

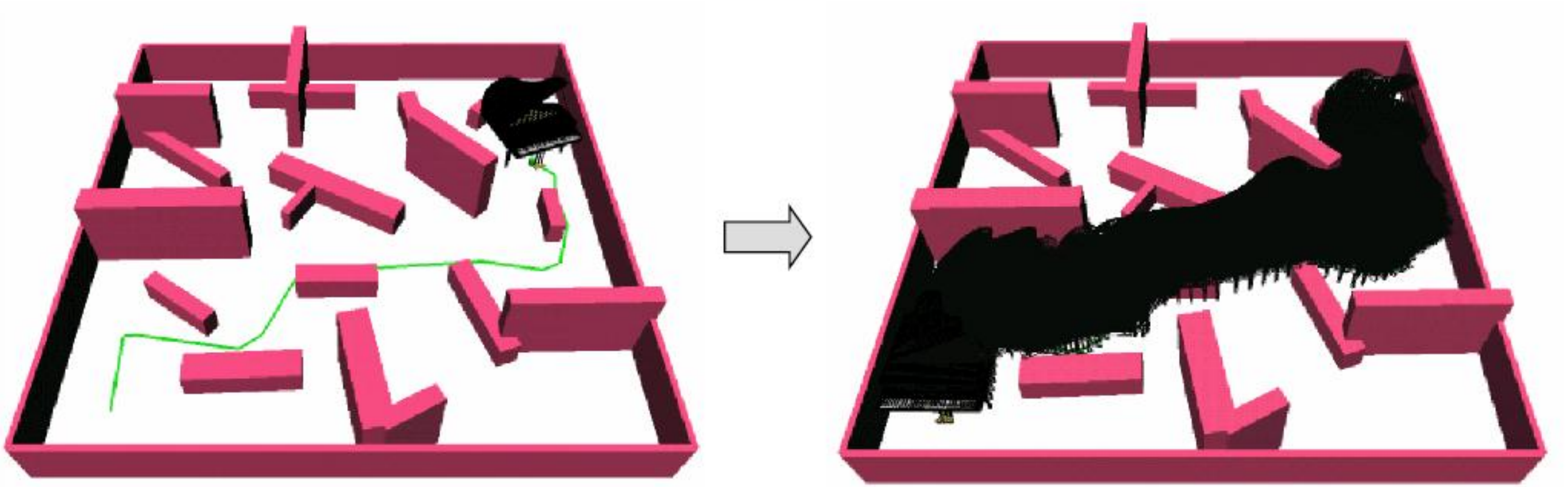


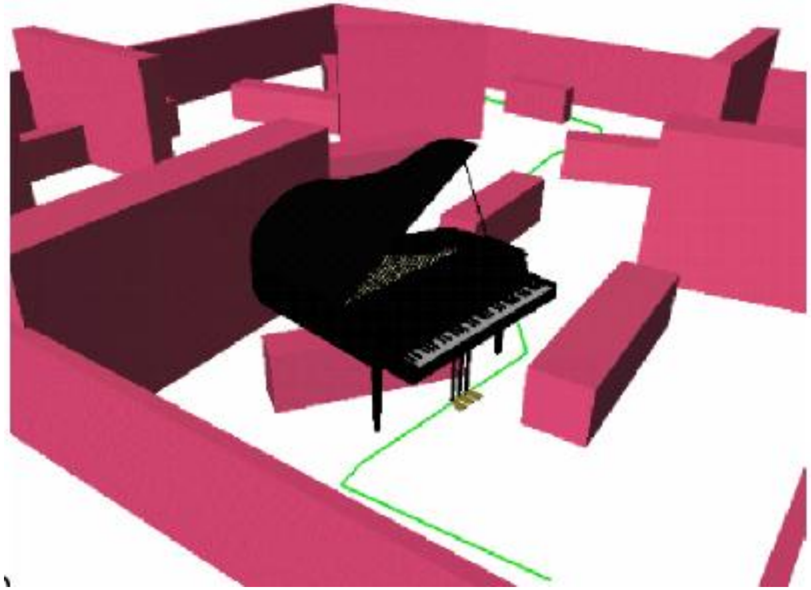
workspace



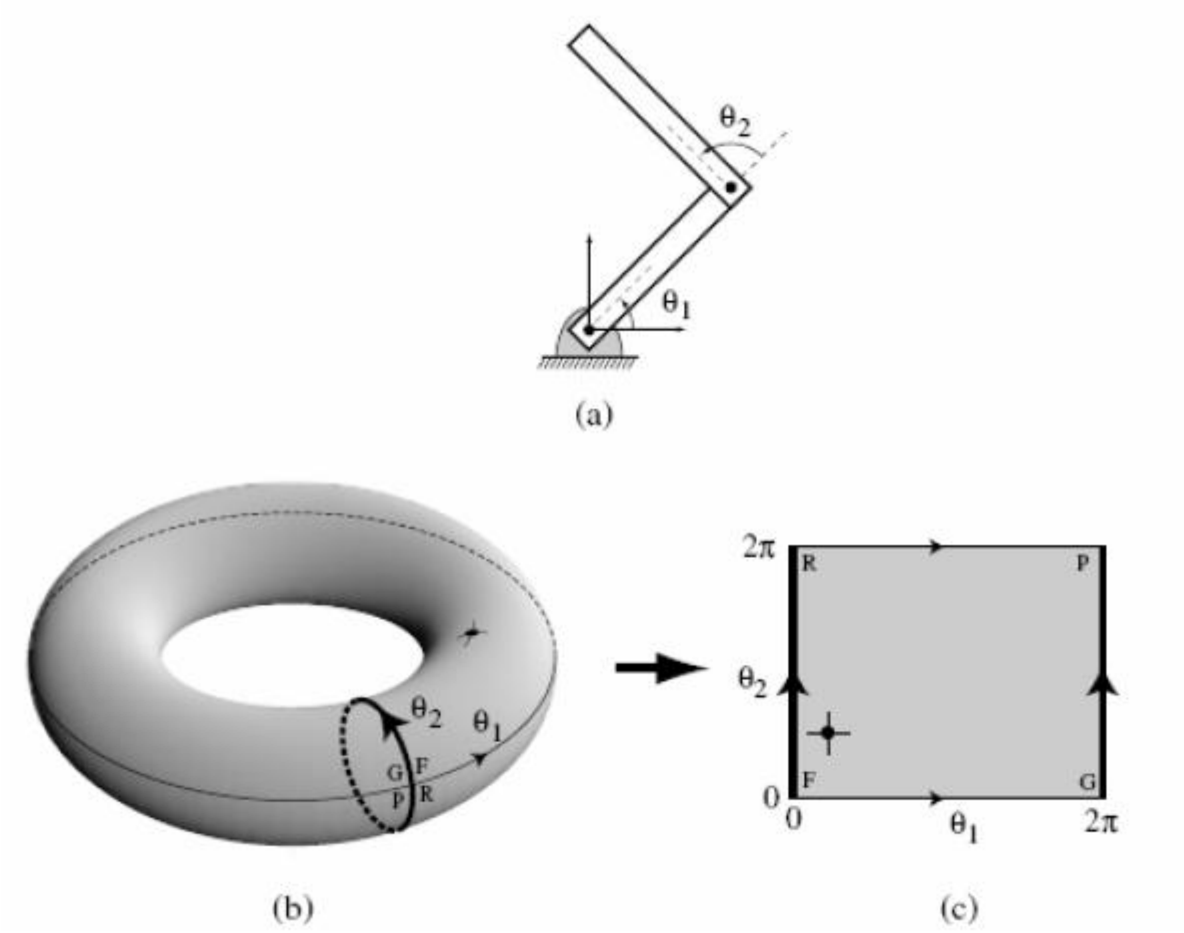
C-space

Moving a piano





Parameterization of Torus



(a)

(b)

(c)

$(\theta_1, \theta_2) \in \mathbb{R}^2$,

problems at $\theta_i = \{0, 2\pi\}$.

Metric in Configuration Space

A metric or distance function d in C is a map

$$d: (q_1,q_2) \in C^2 \rightarrow d(q_1,q_2) \geq 0$$

such that:

- $d(q_1,q_2) = 0$ if and only if $q_1 = q_2$
- $d(q_1,q_2) = d(q_2,q_1)$
- $d(q_1,q_2) \leq d(q_1,q_3) + d(q_3,q_2)$

Metric in Configuration Space

Example:

- Robot ***A*** and point x of ***A***
- x(q): location of x in the workspace when *A* is at configuration q
- *A* distance d in *C* is defined by:

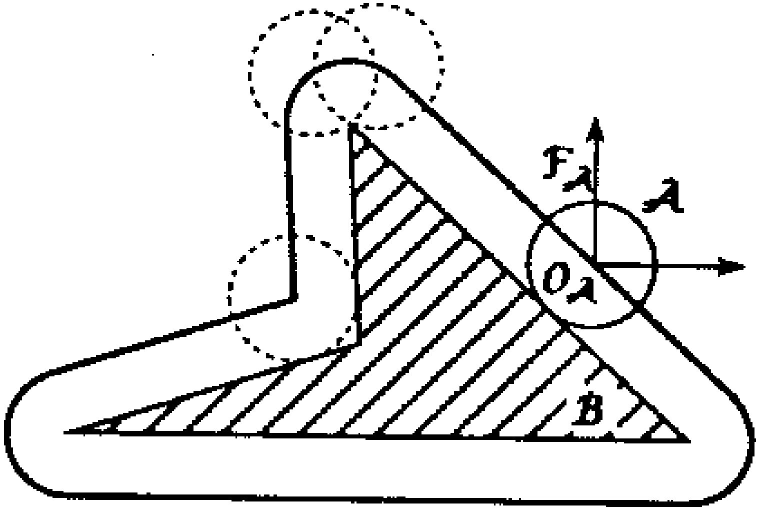
$$d(q,q') = \max_{x \in A} ||x(q)-x(q')||$$

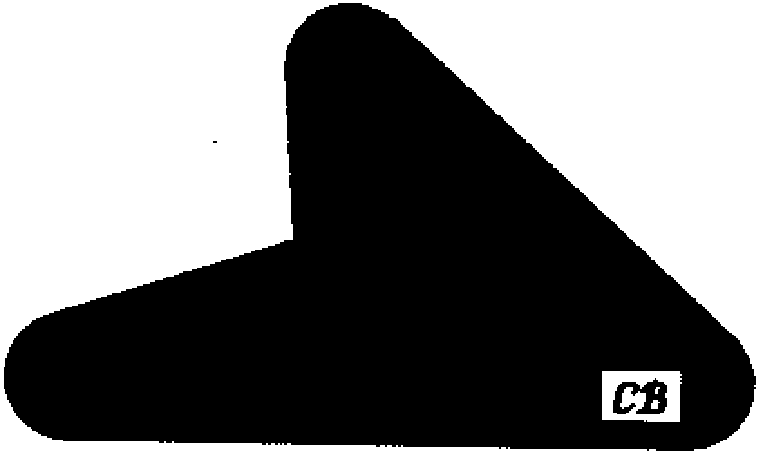
where ||a - b|| denotes the Euclidean distance between points a and b in the workspace

Obstacles in C-Space

- A configuration q is collision-free, or free, if the robot placed at q has null intersection with the obstacles in the workspace
- The free space F is the set of free configurations
- A C-obstacle is the set of configurations where the robot collides with a given workspace obstacle
- A configuration is semi-free if the robot at this configuration touches obstacles without overlap

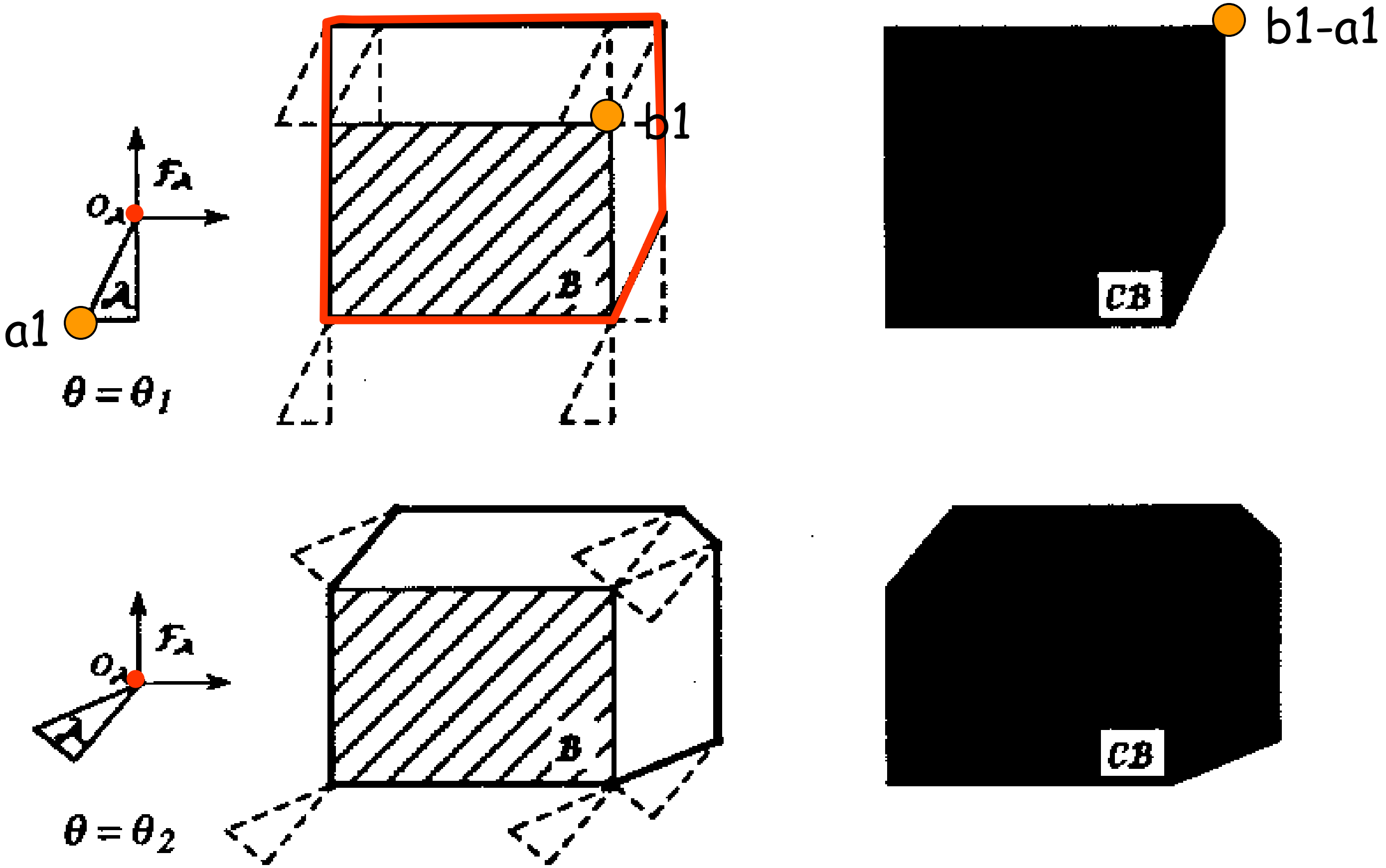
Disc Robot in 2-D Workspace





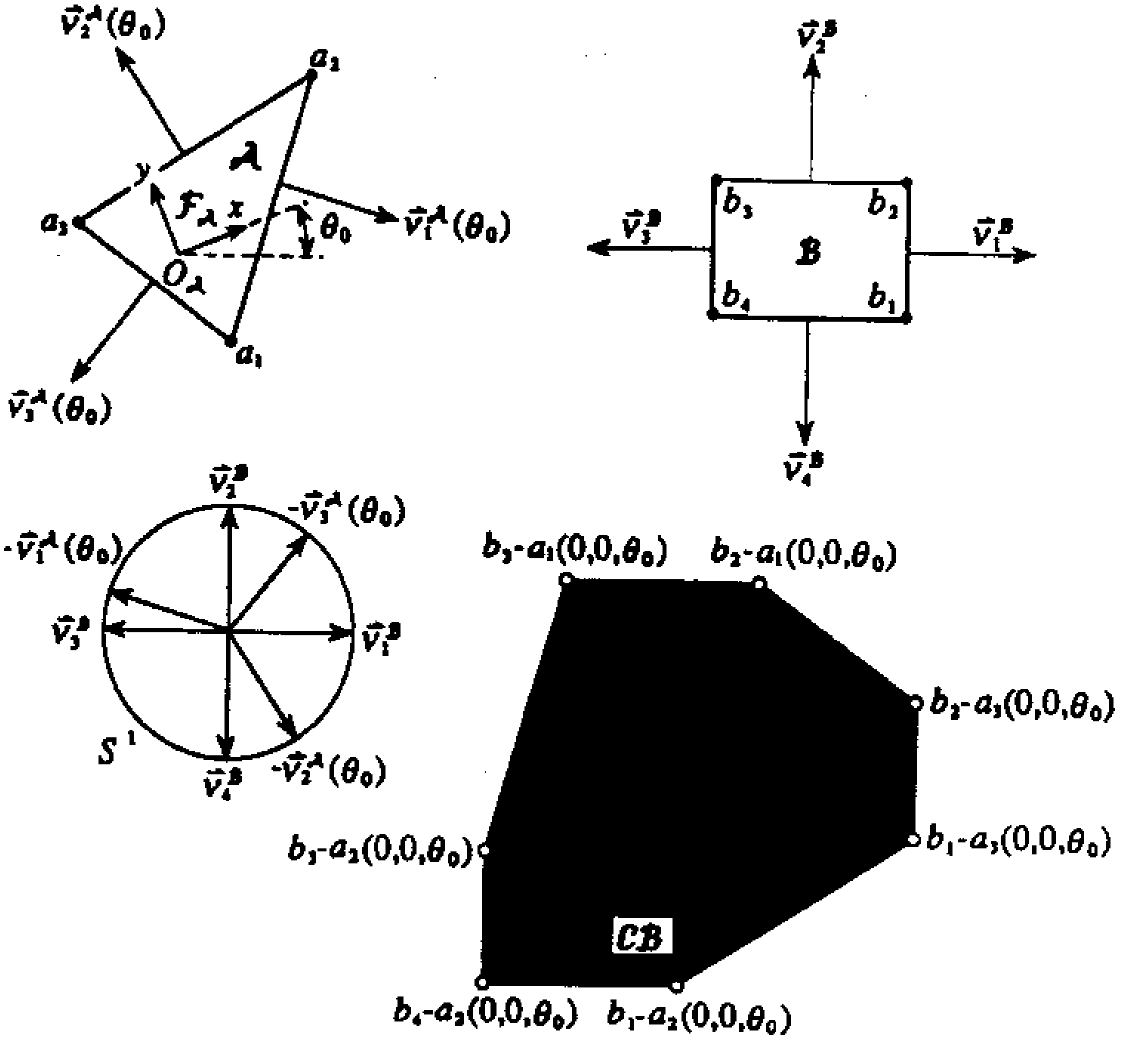
Rigid Robot Translating in 2-D

$CB = B \ominus A = \{b-a \mid a \in A,\ b \in B\}$

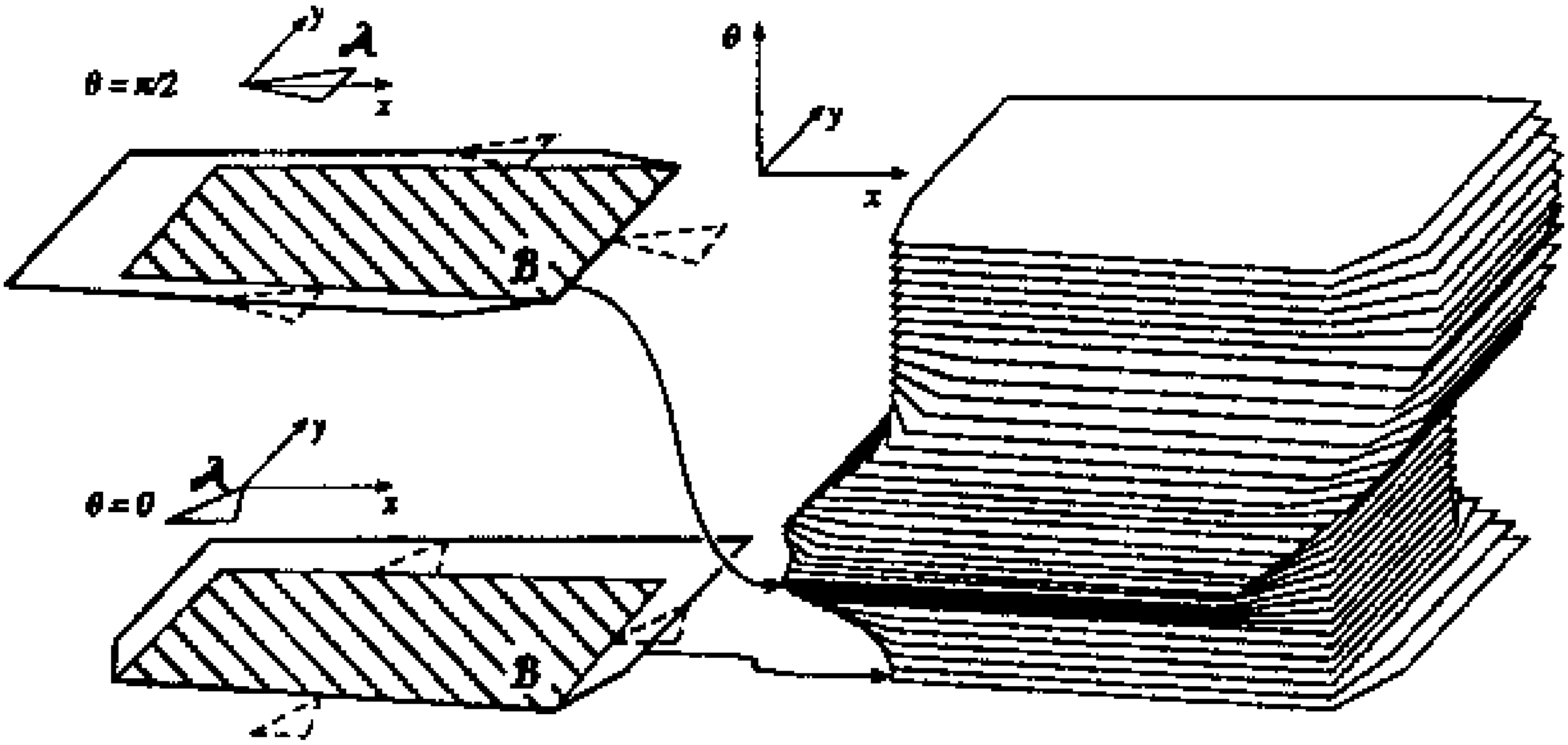


Linear-Time Computation of C-Obstacle in 2-D

polygons)



Rigid Robot Translating and Rotating in 2-D

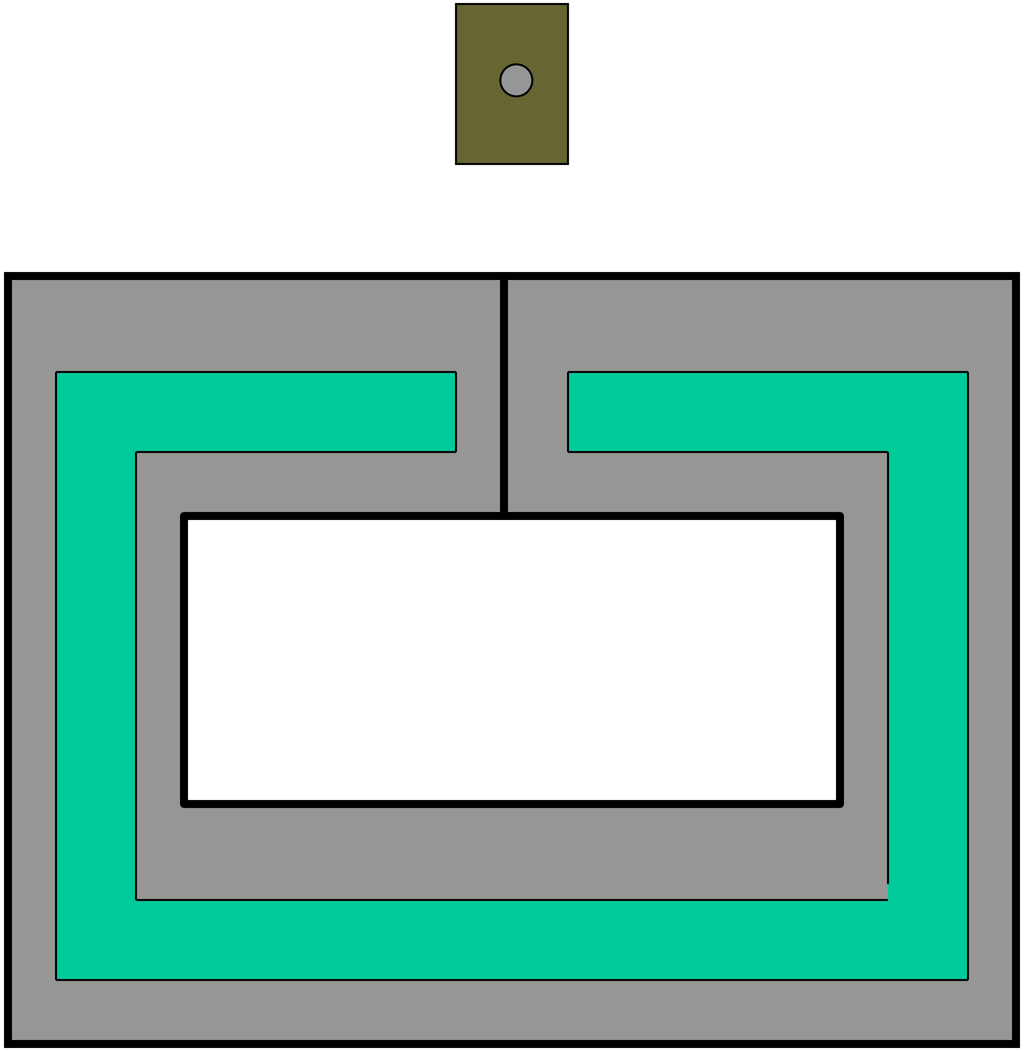


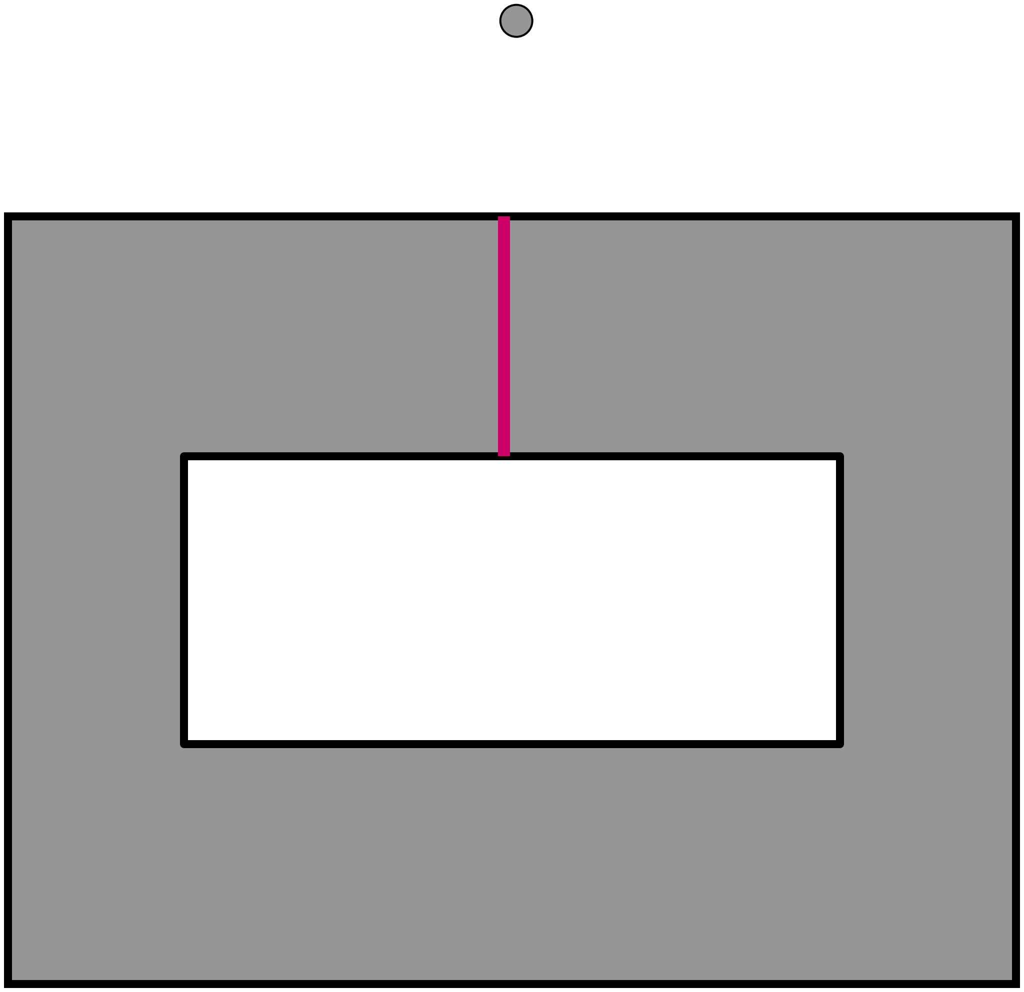
Free and Semi-Free Paths

- A **free path** lies entirely in the free space F
- A **semi-free path** lies entirely in the semi-free space

Remarks on Free-Space Topology

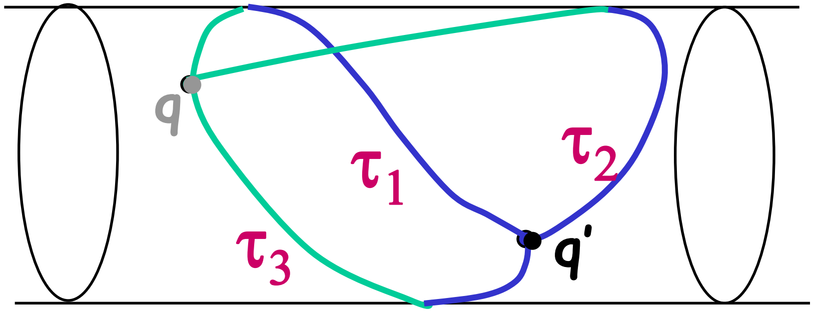
- The robot and the obstacles are modeled as closed subsets, meaning that they contain their boundaries
- One can show that the C-obstacles are closed subsets of the configuration space C as well
- Consequently, the free space F is an open subset of C. Hence, each free configuration is the center of a ball of non-zero radius entirely contained in F
- The semi-free space is a closed subset of C. Its boundary is a superset of the boundary of F





Notion of Homotopic Paths

- Two paths with the same endpoints are homotopic if one can be continuously deformed into the other
- R x S^1 example:

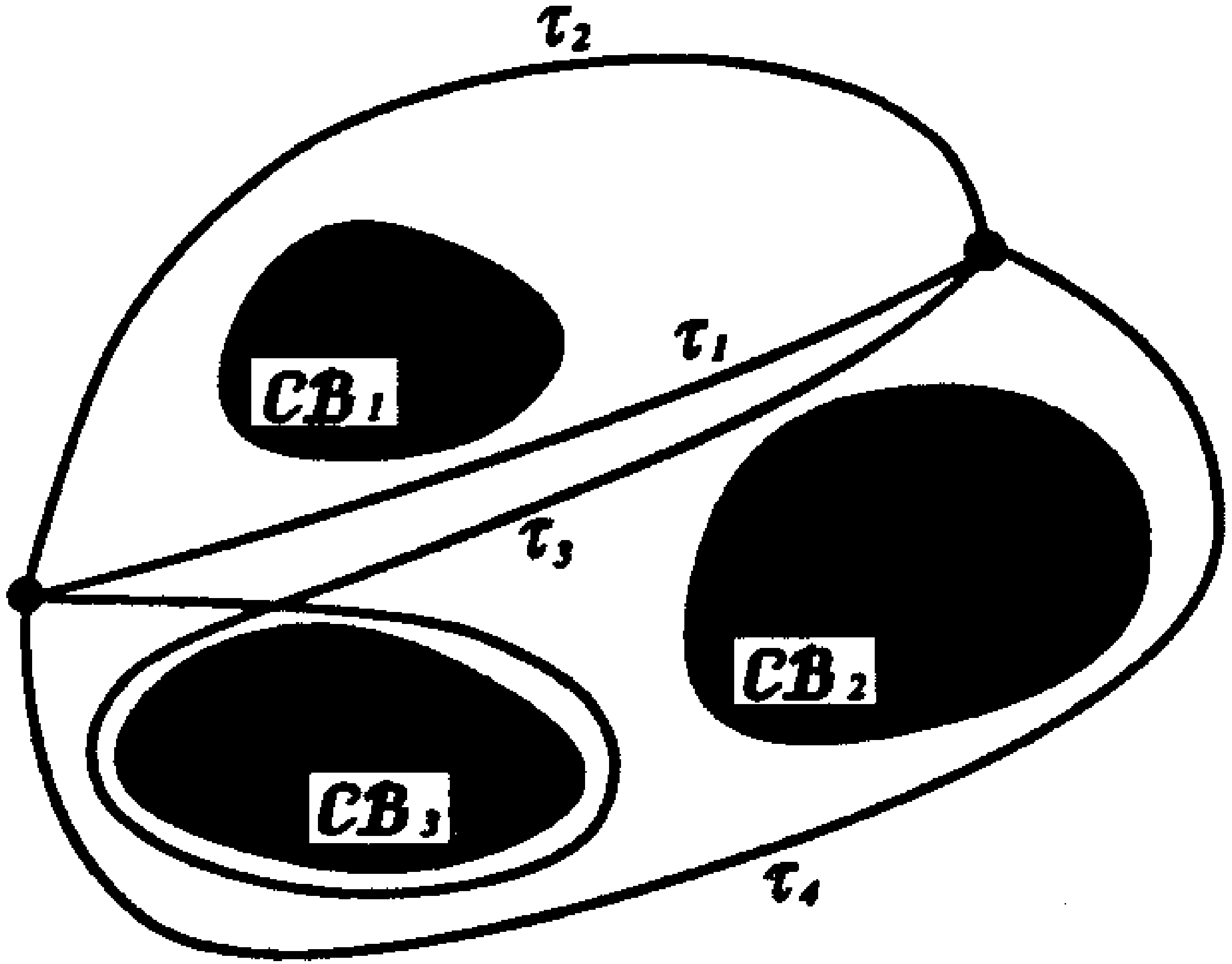


- τ_1 and τ_2 are homotopic
- τ_1 and τ_3 are not homotopic
- In this example, infinity of homotopy classes

Connectedness of C-Space

- C is connected if every two configurations can be connected by a path
- C is simply-connected if any two paths connecting the same endpoints are homotopic
  Examples: **R**^{2} or **R**^{3}
- Otherwise C is multiply-connected
  Examples: S^1 and SO(3) are multiply- connected:
  - In S^1, infinity of homotopy classes
  - In SO(3), only two homotopy classes

Classes of Homotopic Free Paths

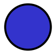


Probabilistic Roadmaps PRMs

Rapidly-exploring Random Trees

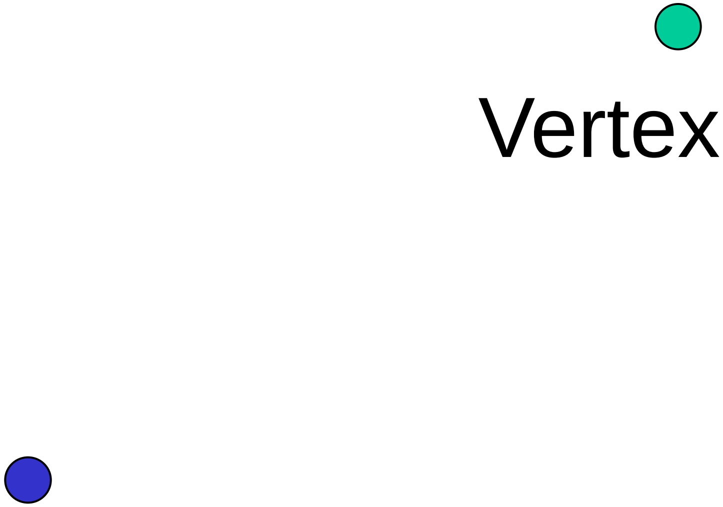
- A point P in C is randomly chosen.
- The nearest vertex in the RRT is selected.
- A new edge is added from this vertex in the direction of P, at distance ε.
- The further the algorithm goes, the more space is covered.

Rapidly-expanding Random Trees



Starting vertex

Rapidly-expanding Random Trees

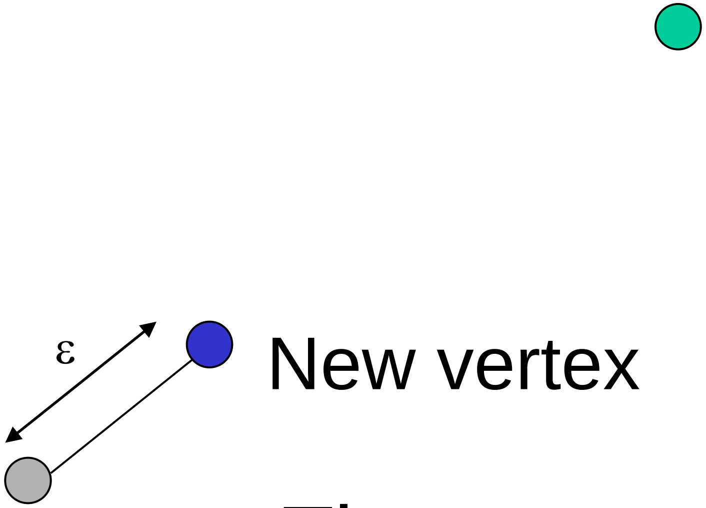


Vertex randomly drawn

Rapidly-expanding Random Trees

Nearest vertex

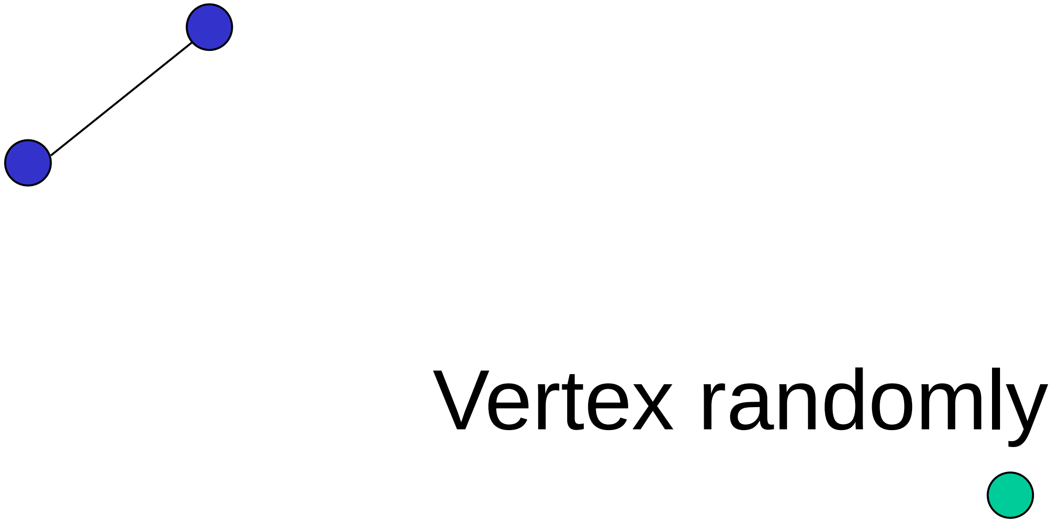
Rapidly-expanding Random Trees



New vertex

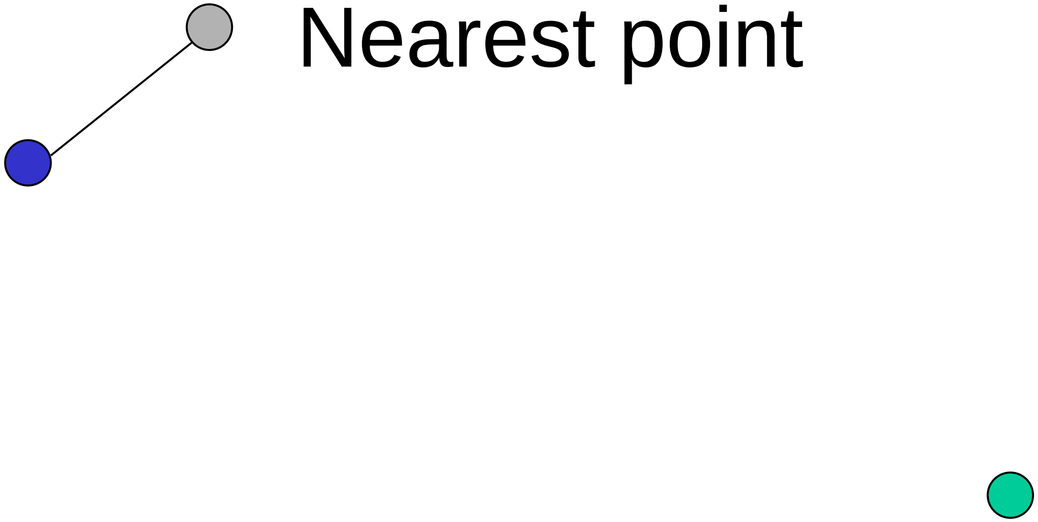
The vertex is in Cfree

Rapidly-expanding Random Trees



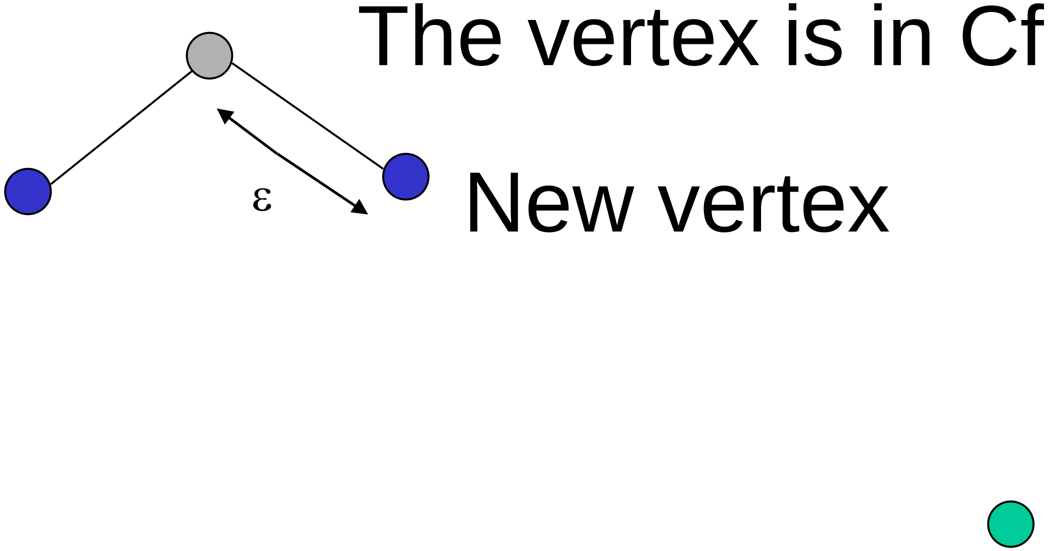
Vertex randomly drawn

Rapidly-expanding Random Trees



Nearest point

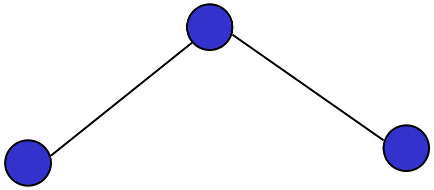
Rapidly-expanding Random Trees



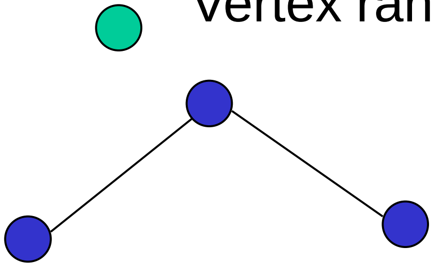
The vertex is in Cfree

New vertex

Rapidly-expanding Random Trees

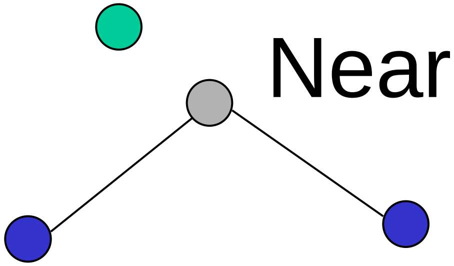


Rapidly-expanding Random Trees



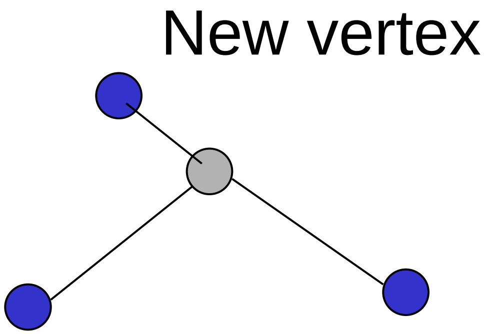
Vertex randomly drawn

Rapidly-expanding Random Trees

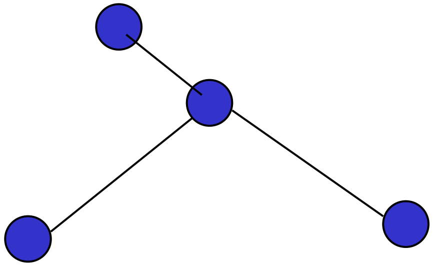


Nearest vertex

Rapidly-expanding Random Trees


New vertex

Rapidly-expanding Random Trees

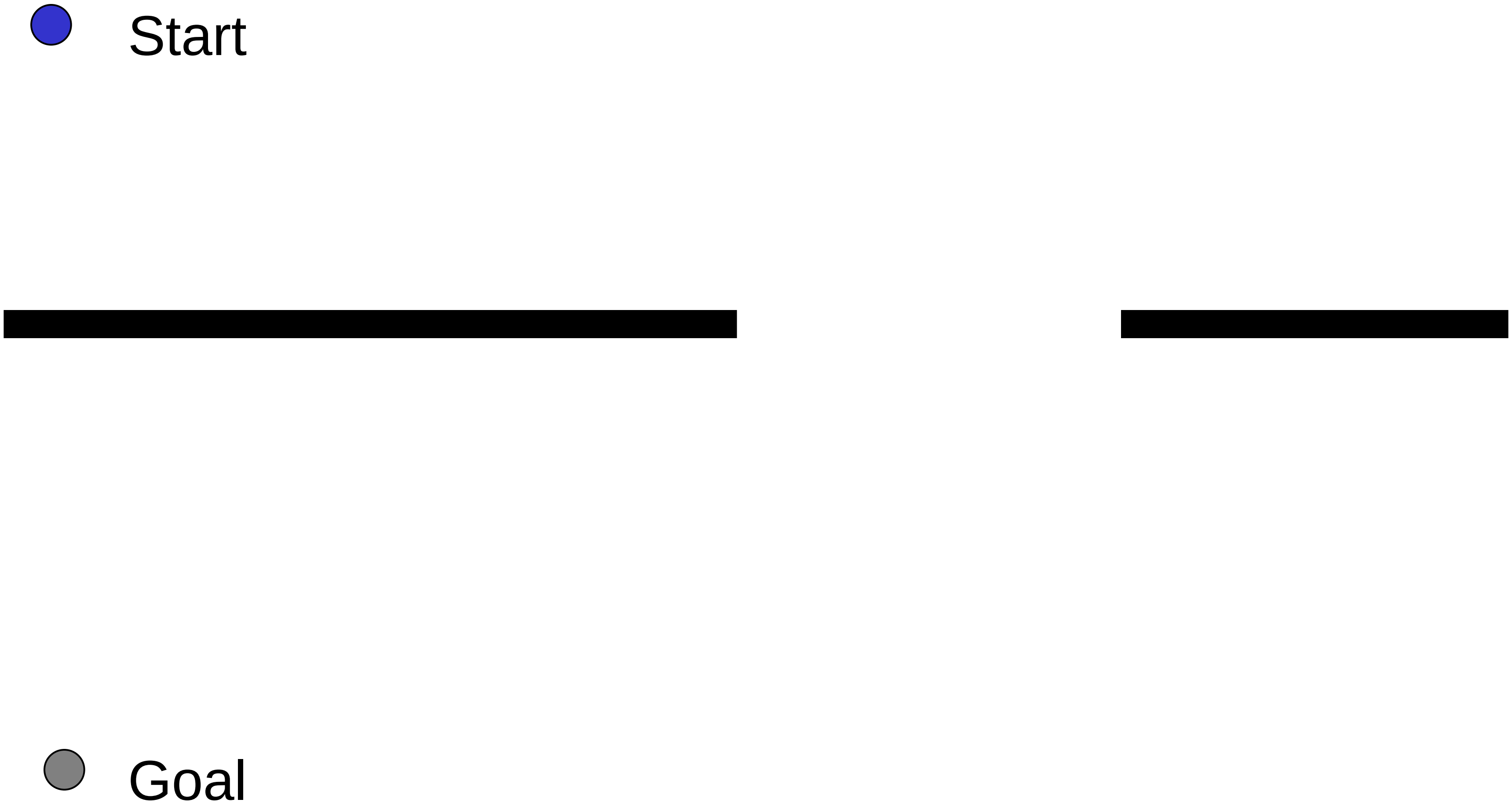


And it continues…

RRT-Connect

- We grow two trees, one from the beginning vertex and another from the end vertex
- Each time we create a new vertex, we try to greedily connect the two trees

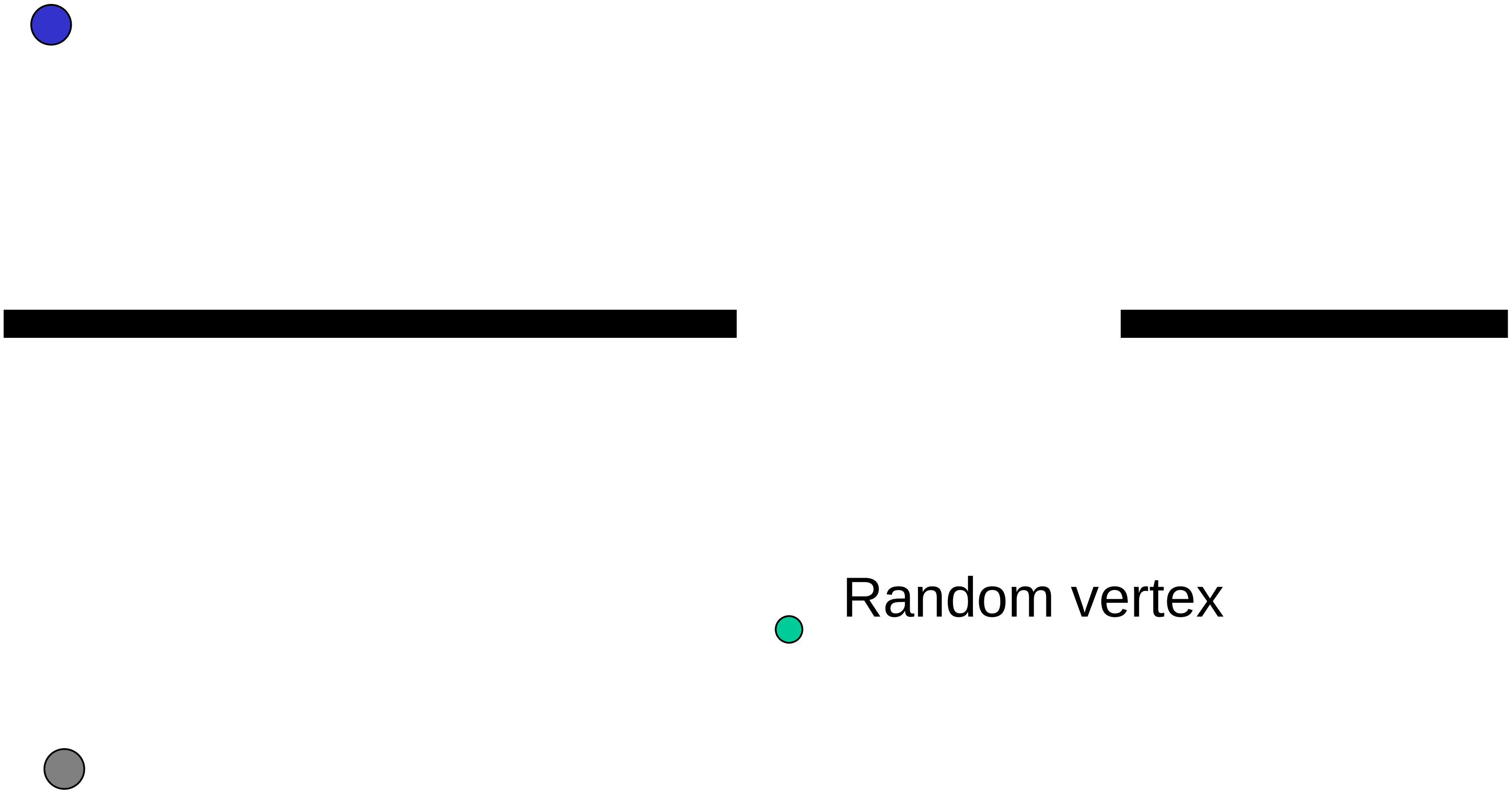
RRT-Connect: example



Start

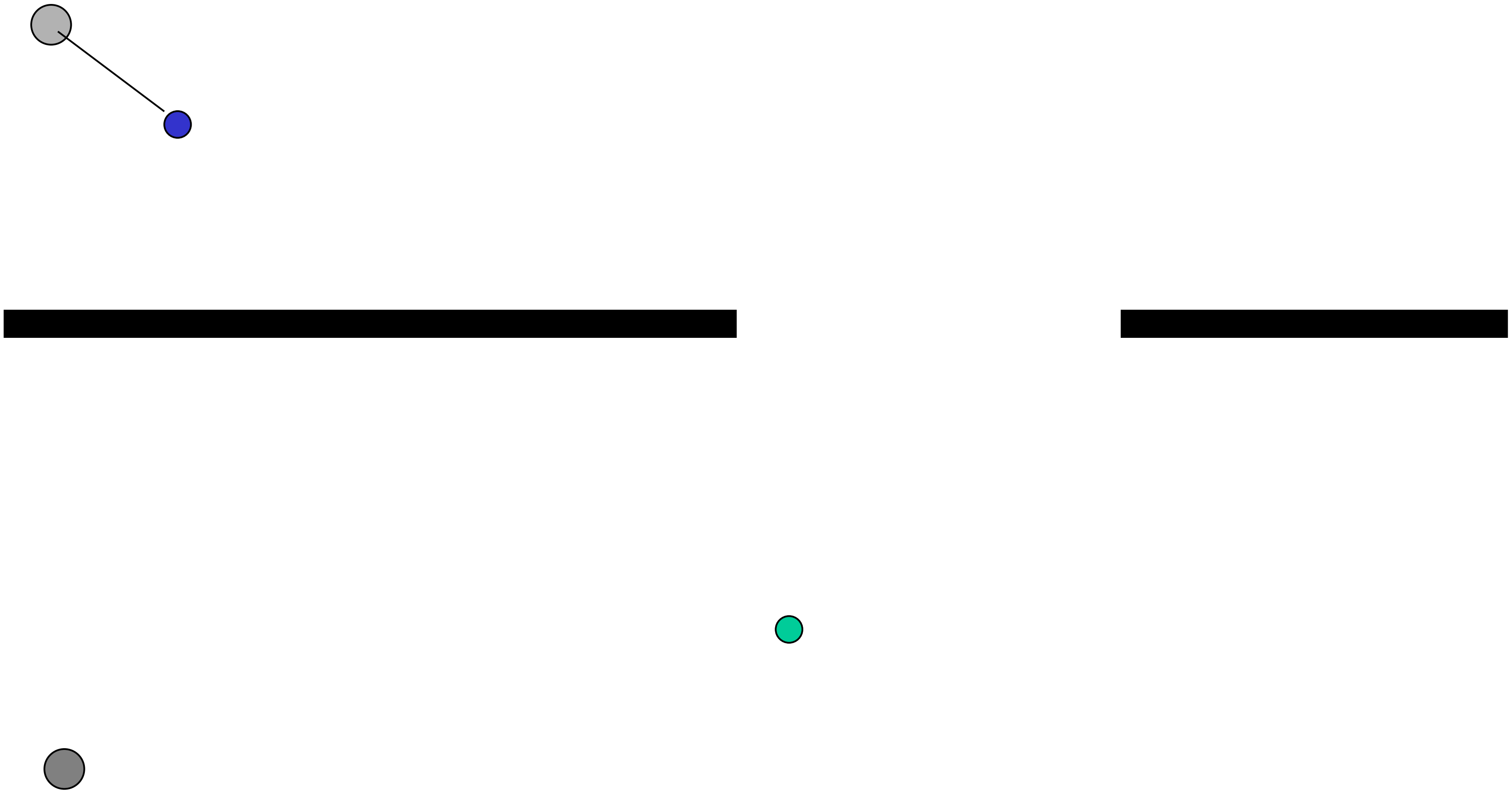
Goal

RRT-Connect: example

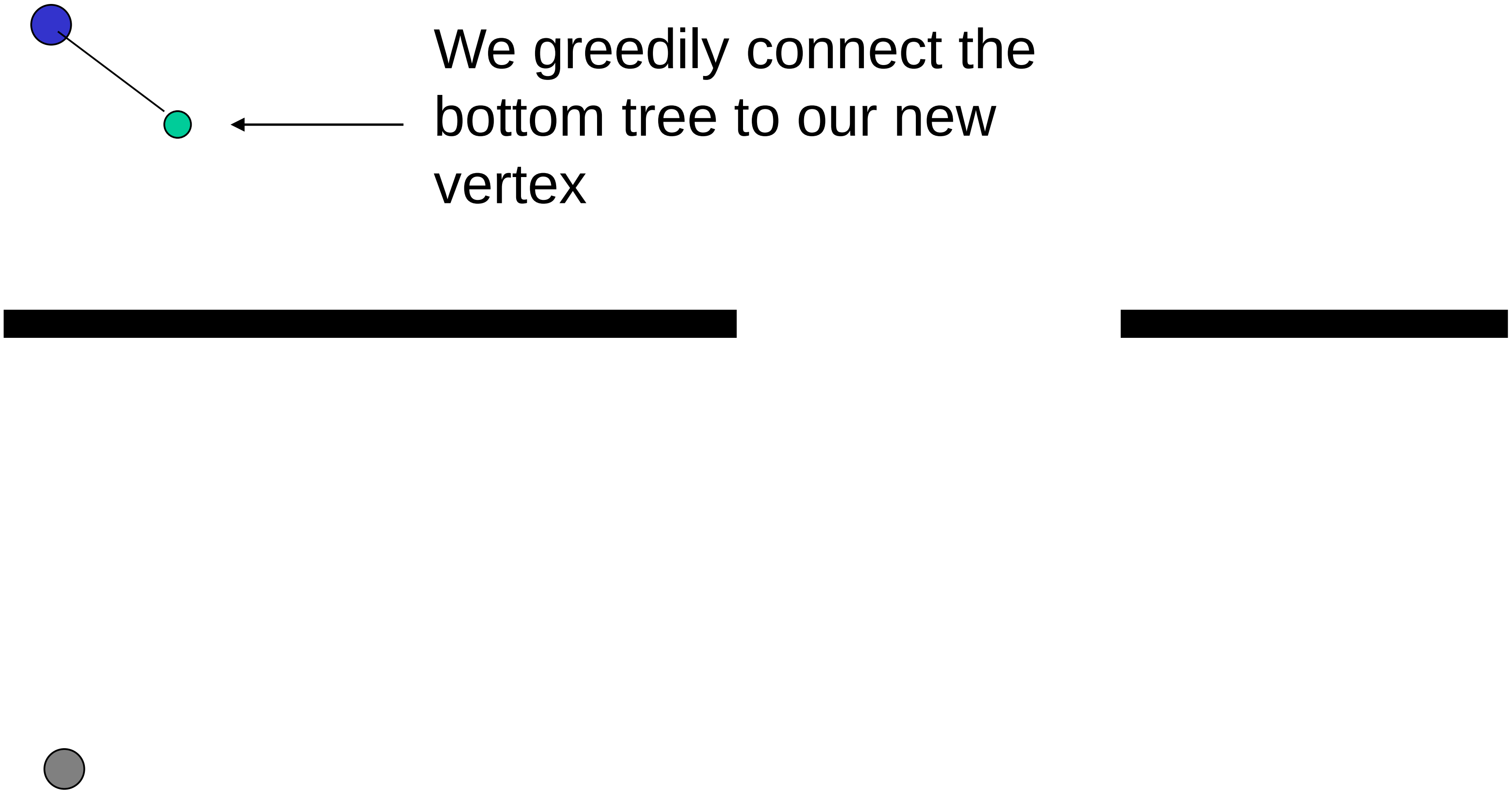


Random vertex

RRT-Connect: example

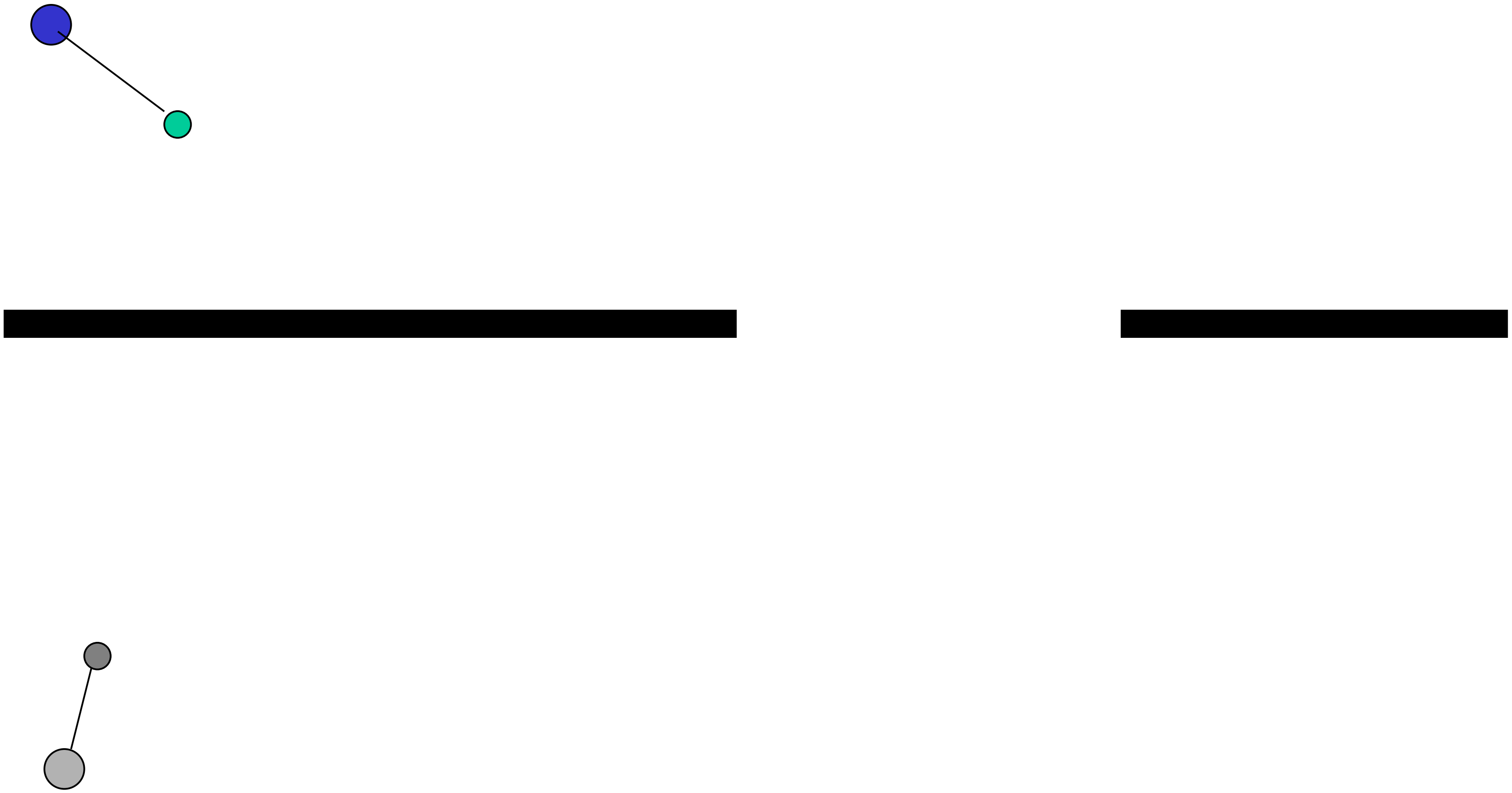


RRT-Connect: example

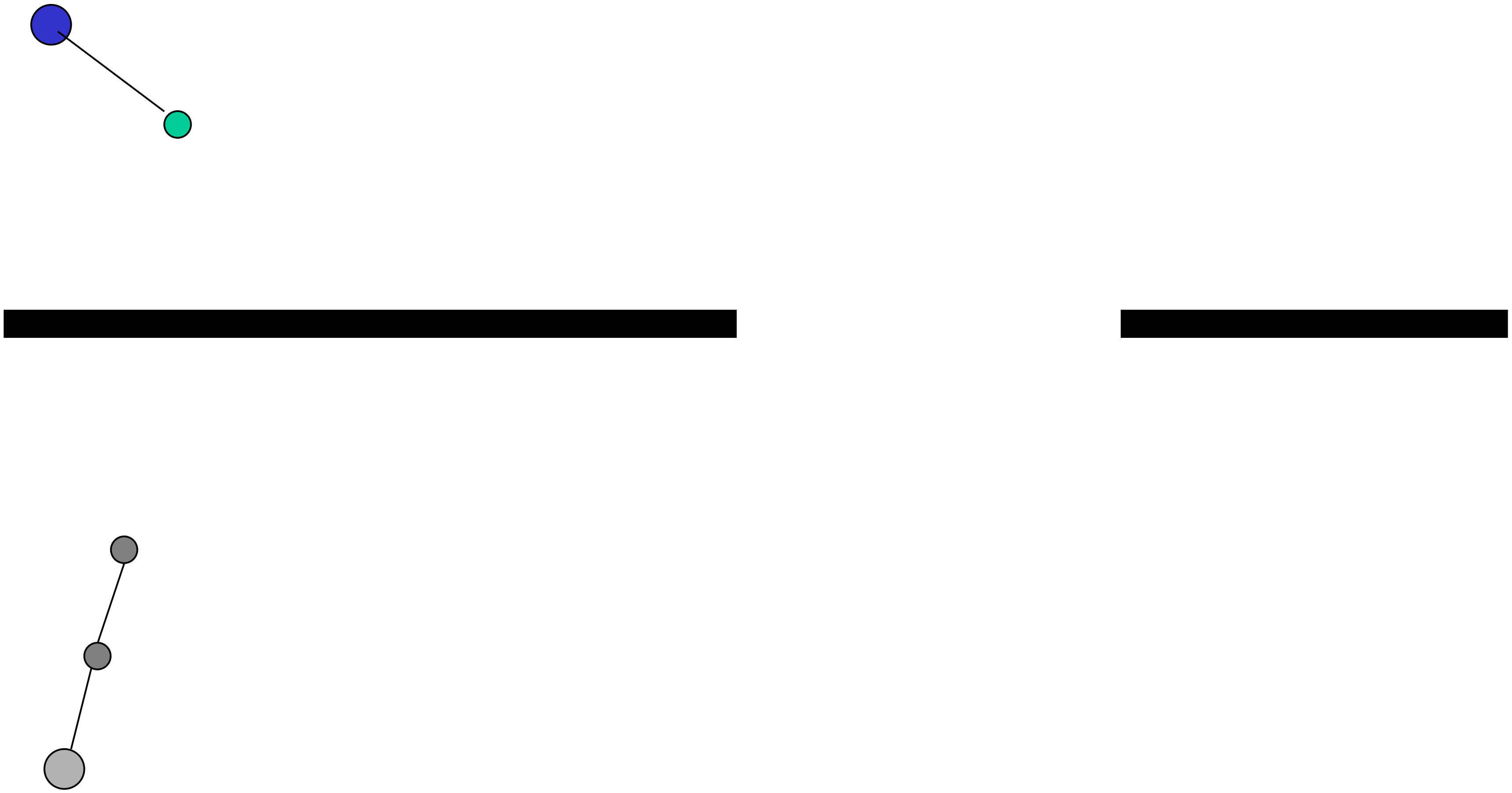


We greedily connect the bottom tree to our new vertex

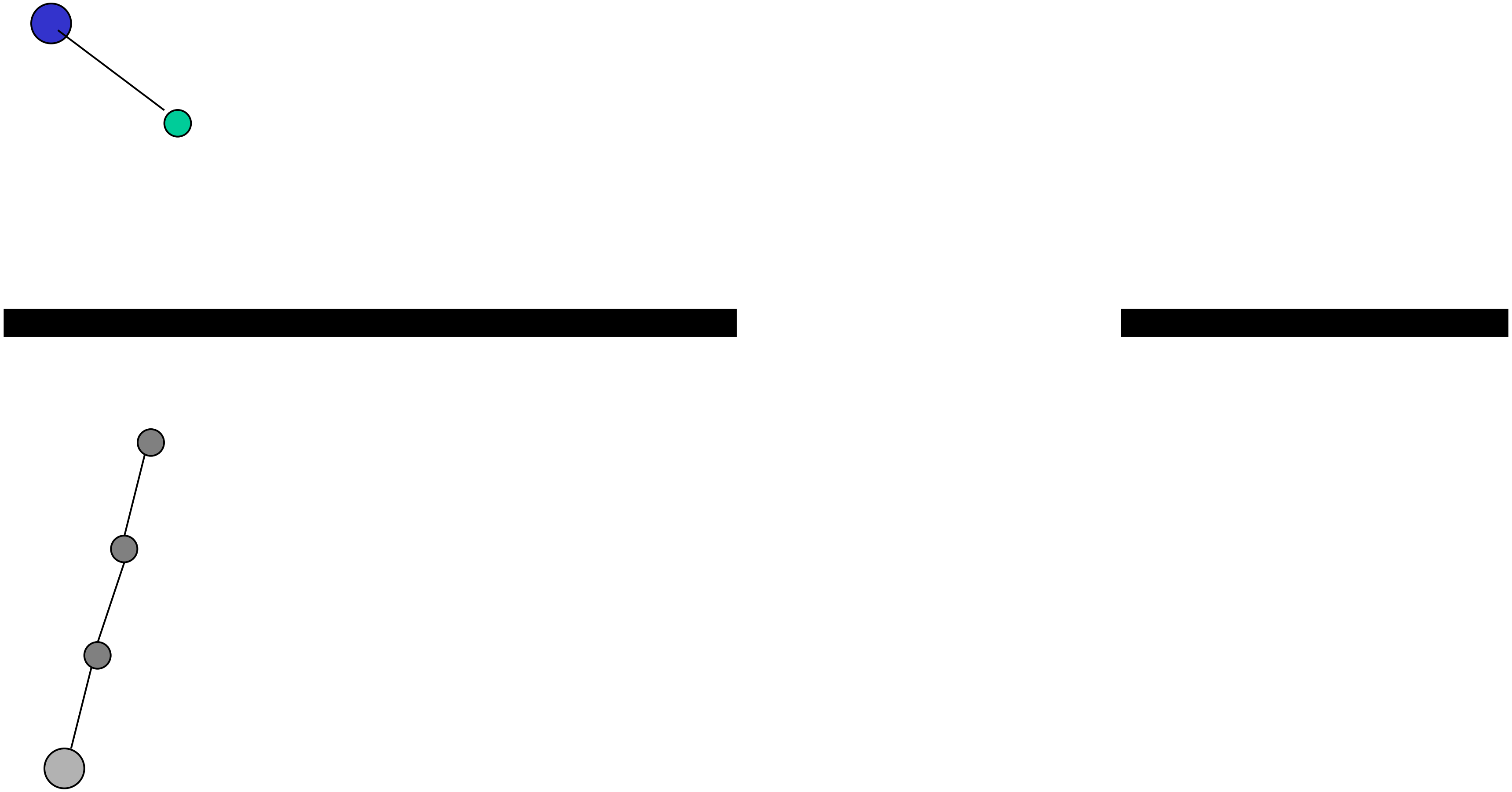
RRT-Connect: example



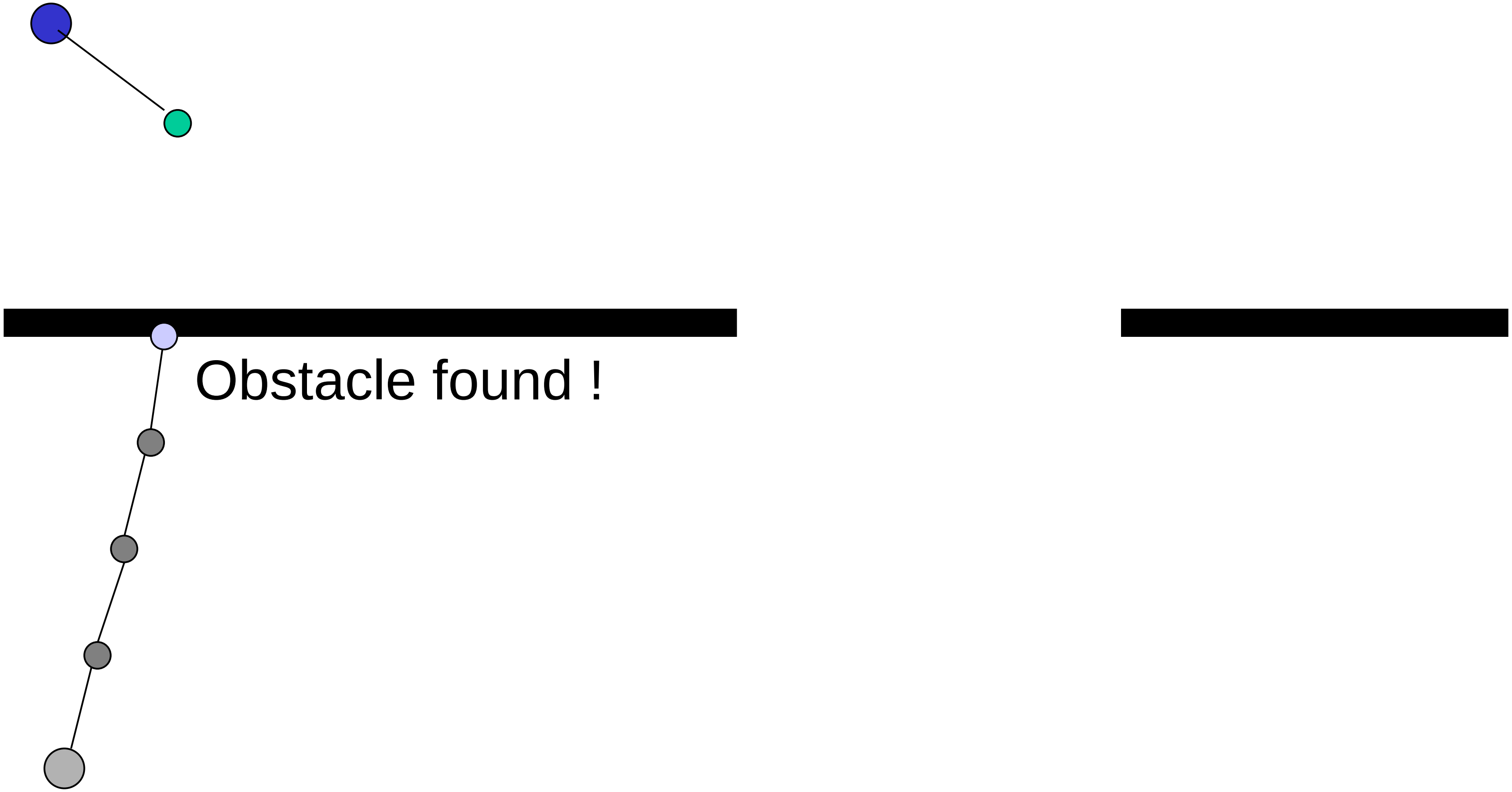
RRT-Connect: example



RRT-Connect: example

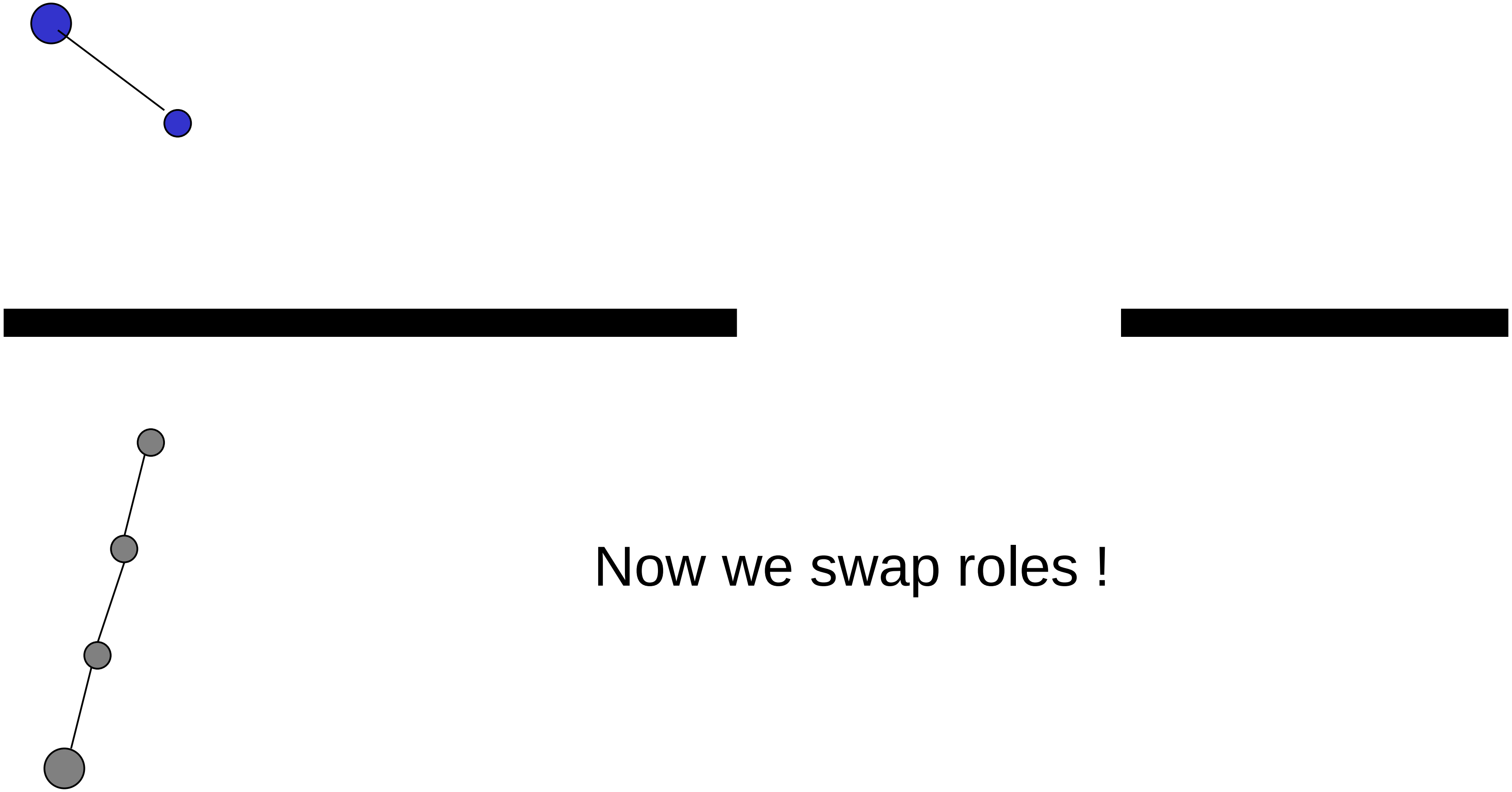


RRT-Connect: example



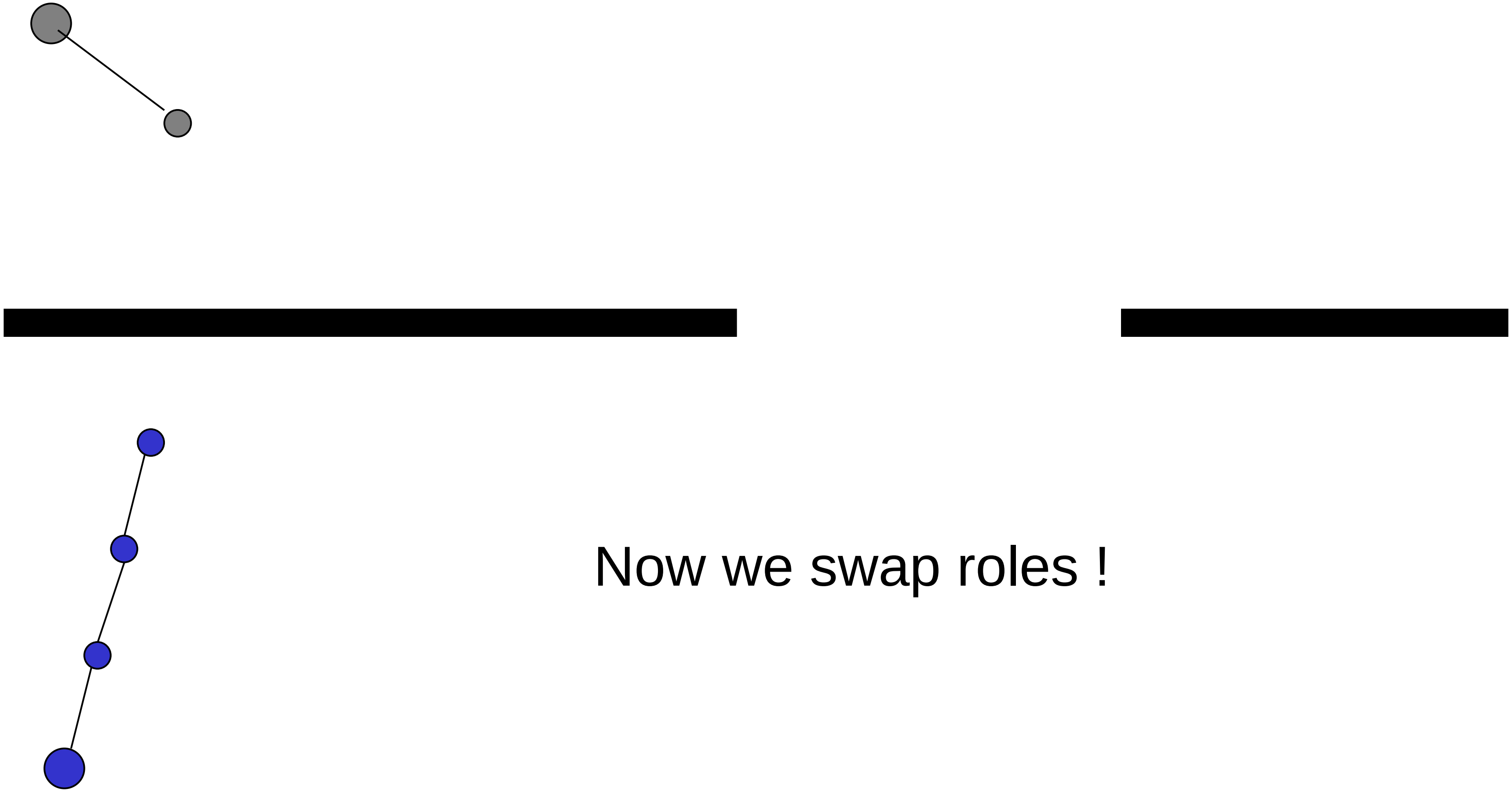
Obstacle found !

RRT-Connect: example



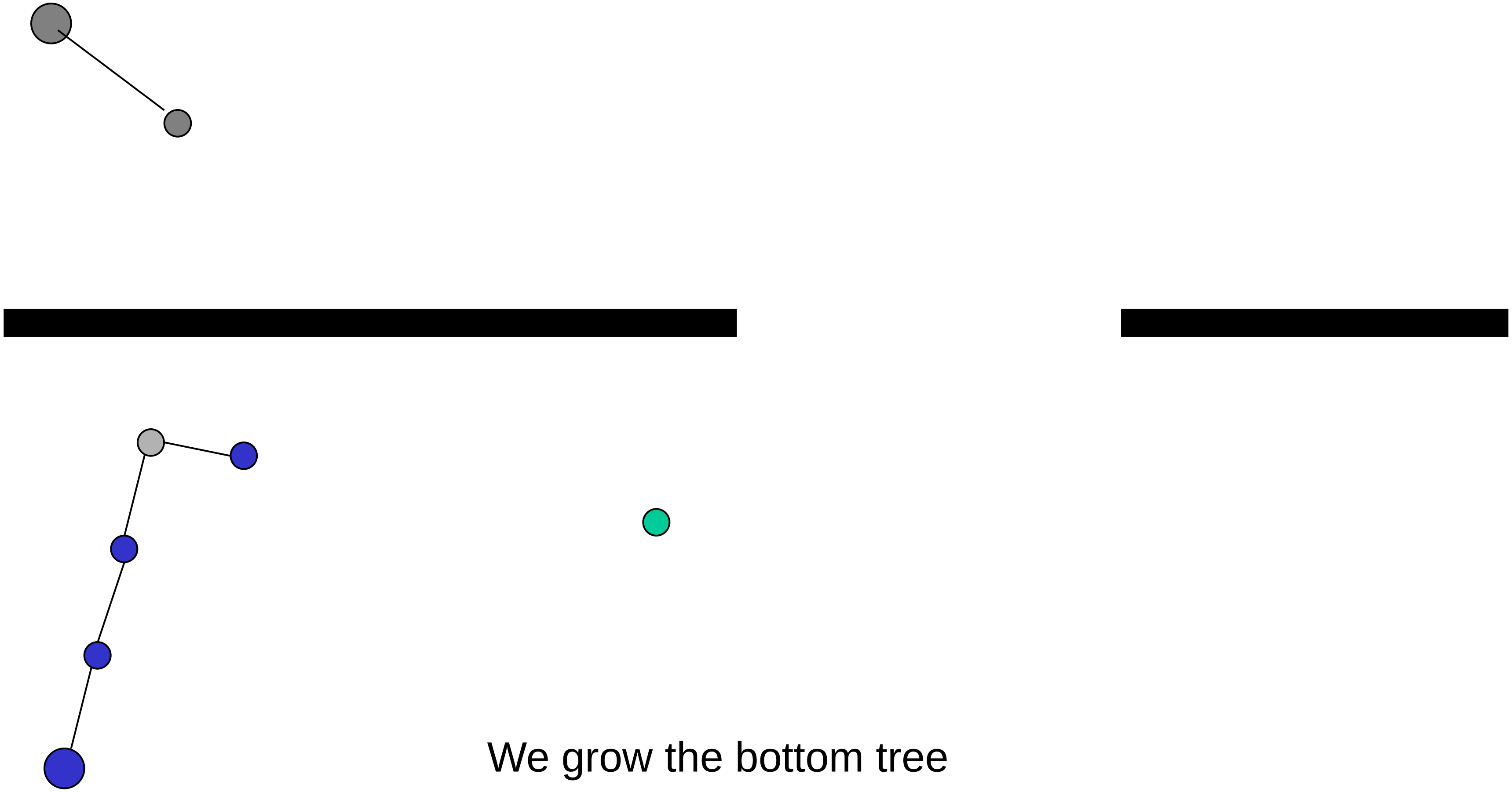
Now we swap roles !

RRT-Connect: example



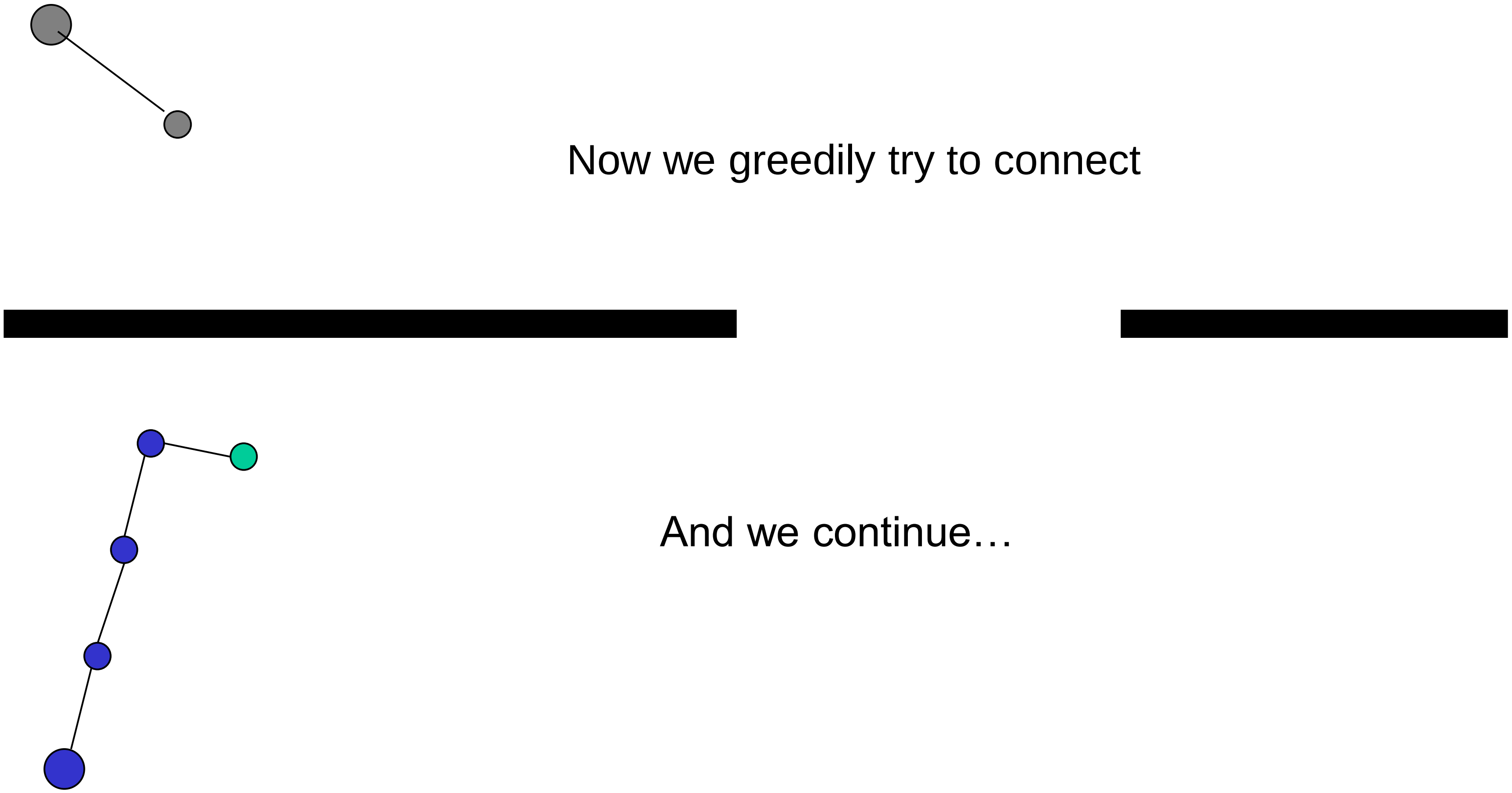
Now we swap roles !

RRT-Connect: example



We grow the bottom tree

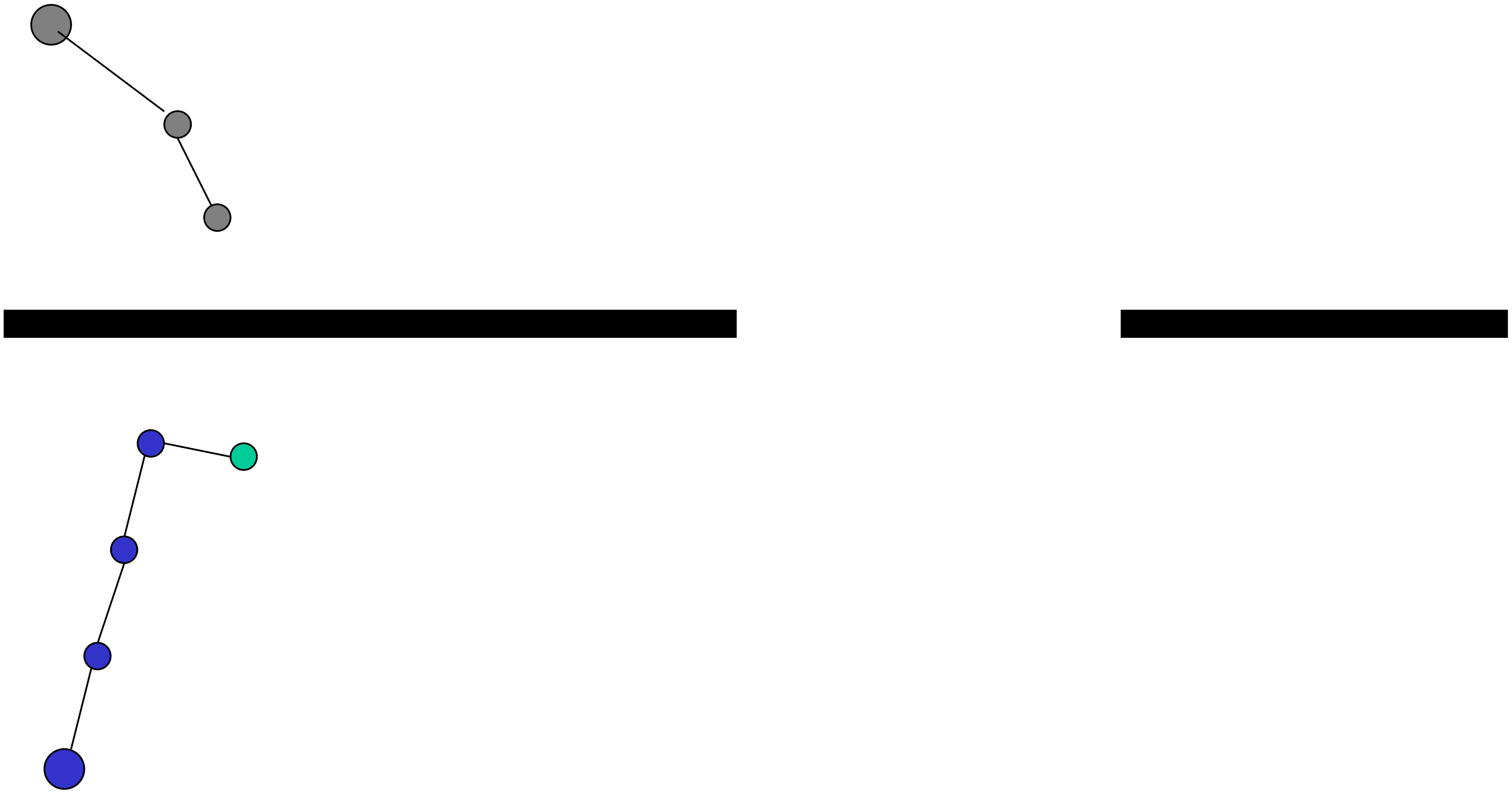
RRT-Connect: example



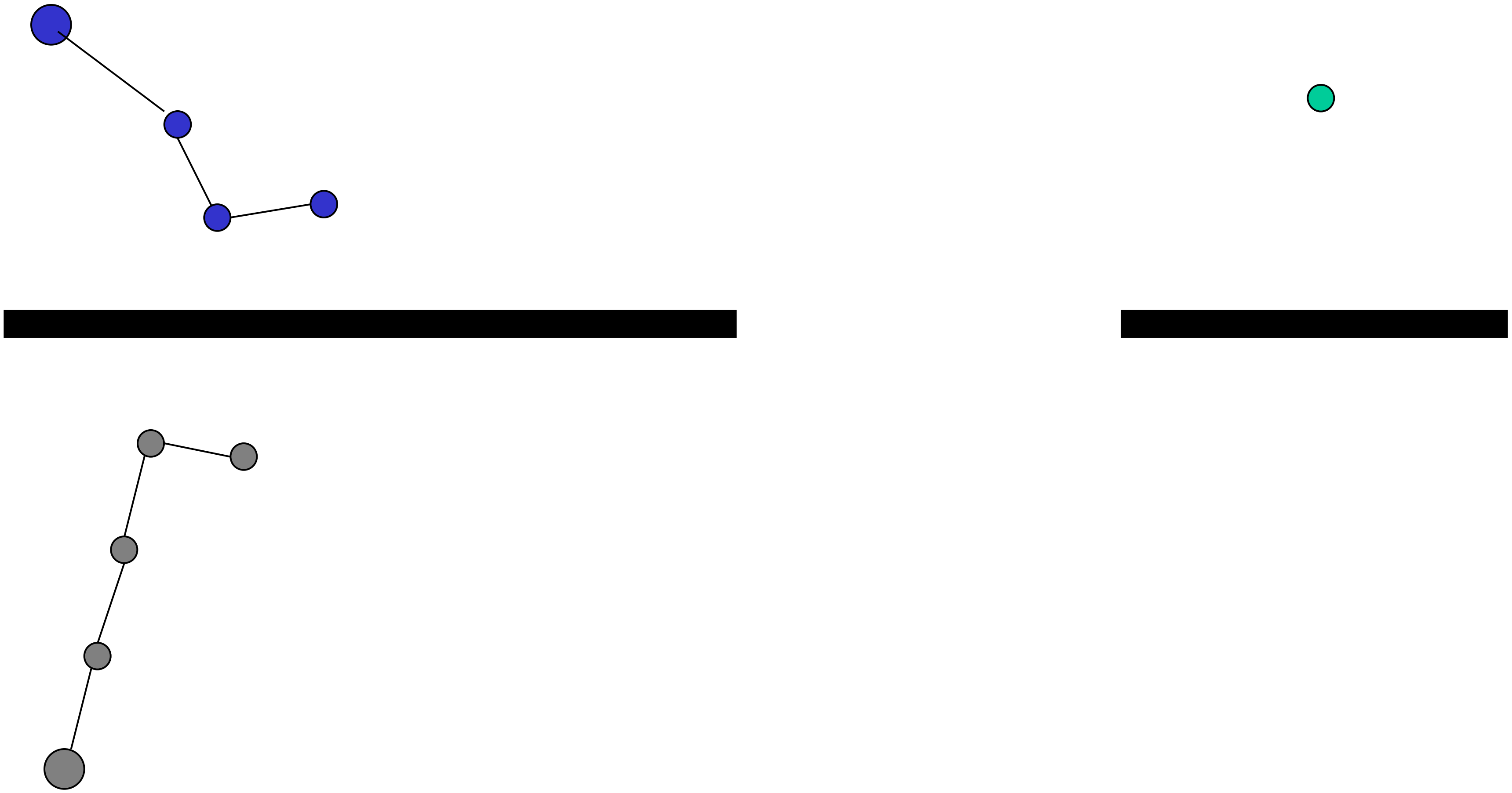
Now we greedily try to connect

And we continue…

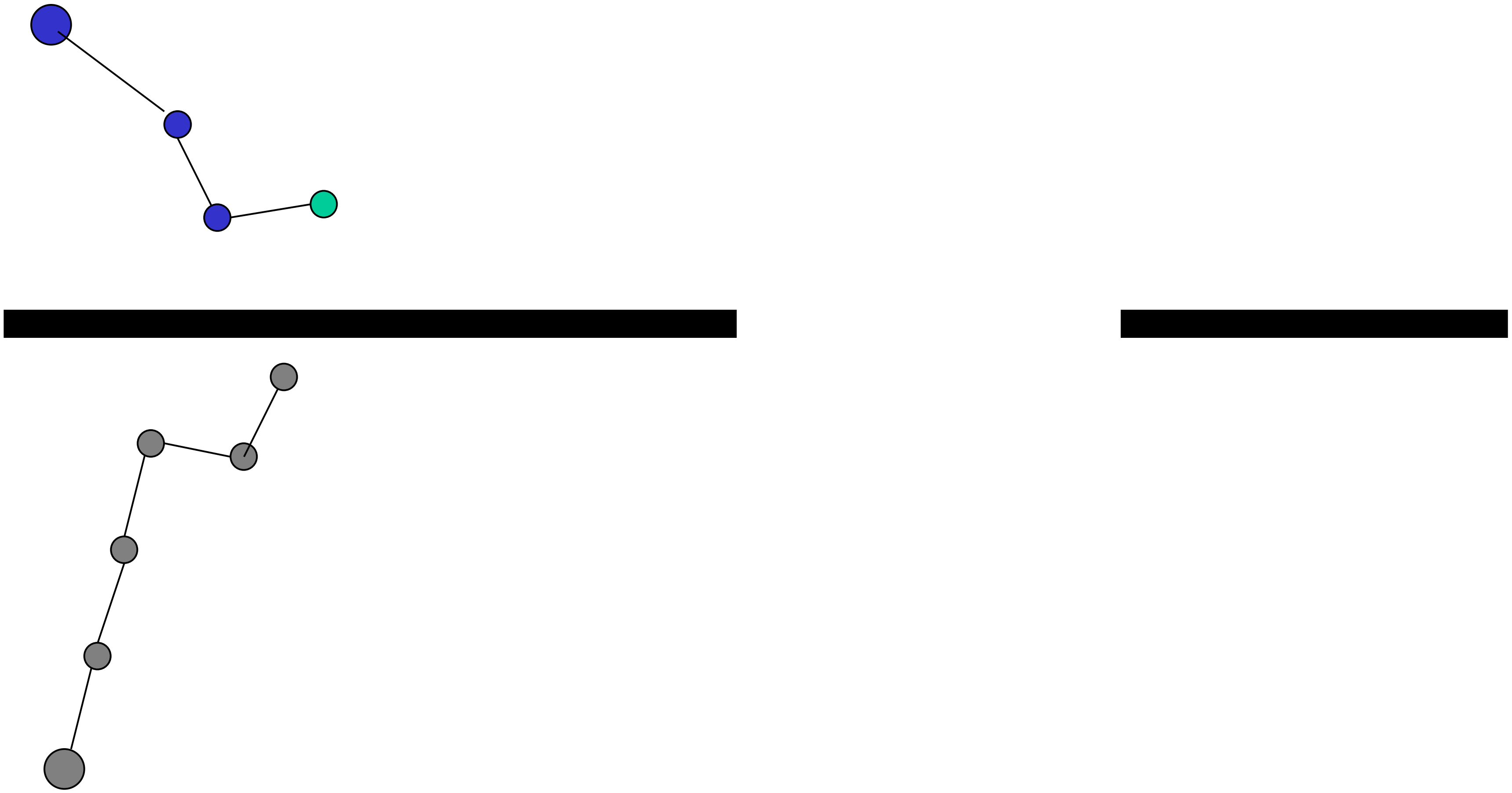
RRT-Connect: example



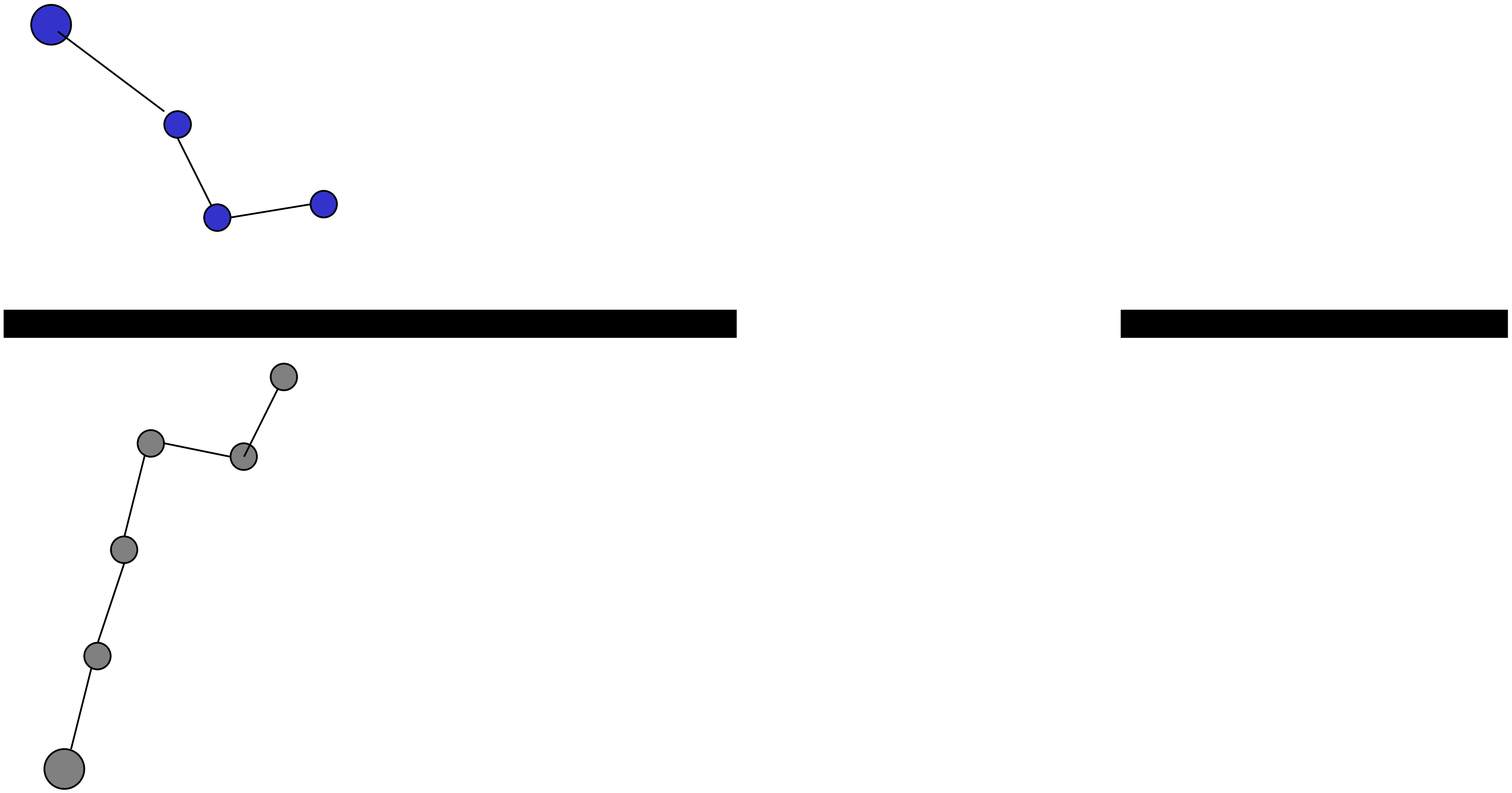
RRT-Connect: example



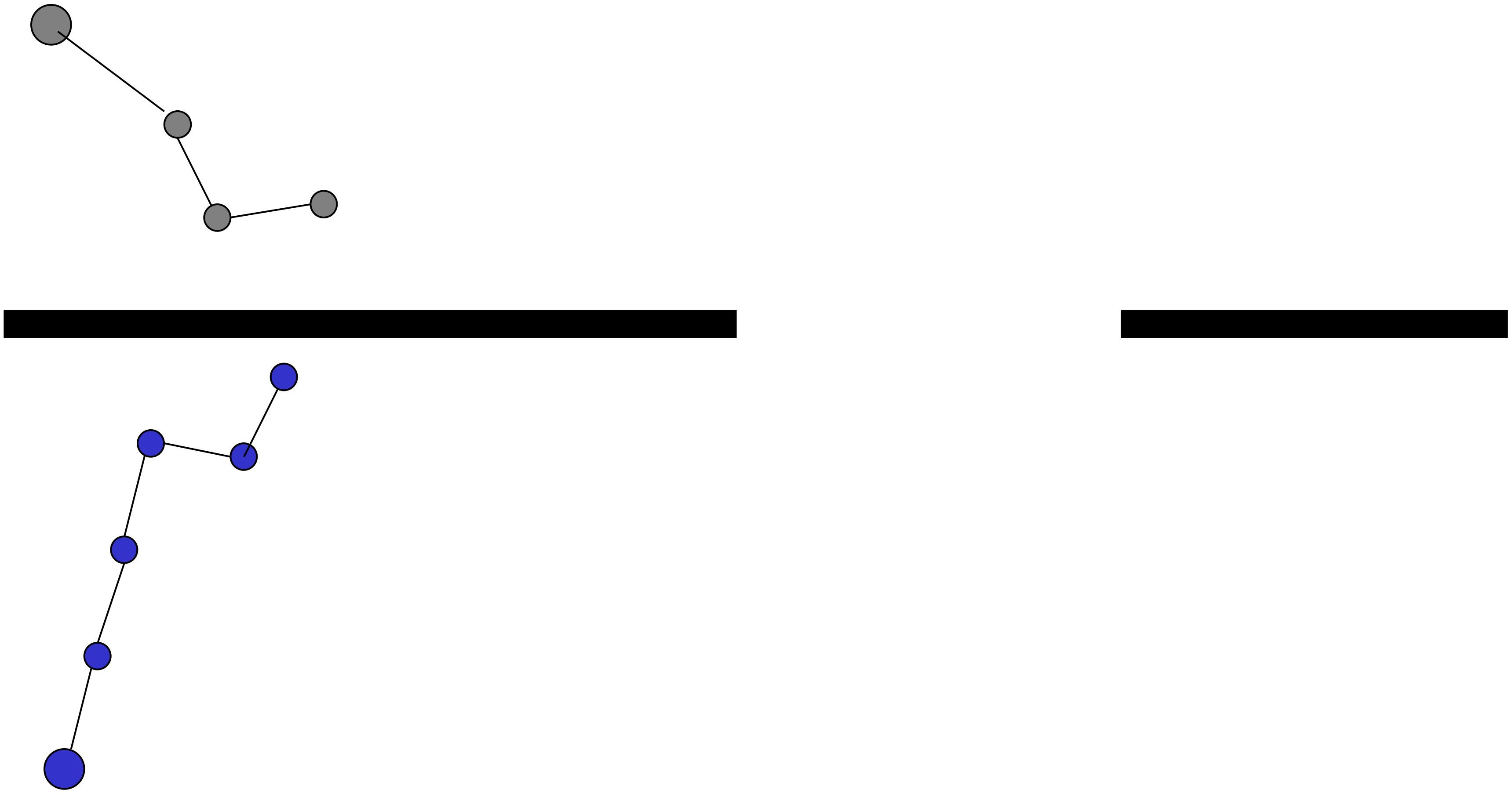
RRT-Connect: example



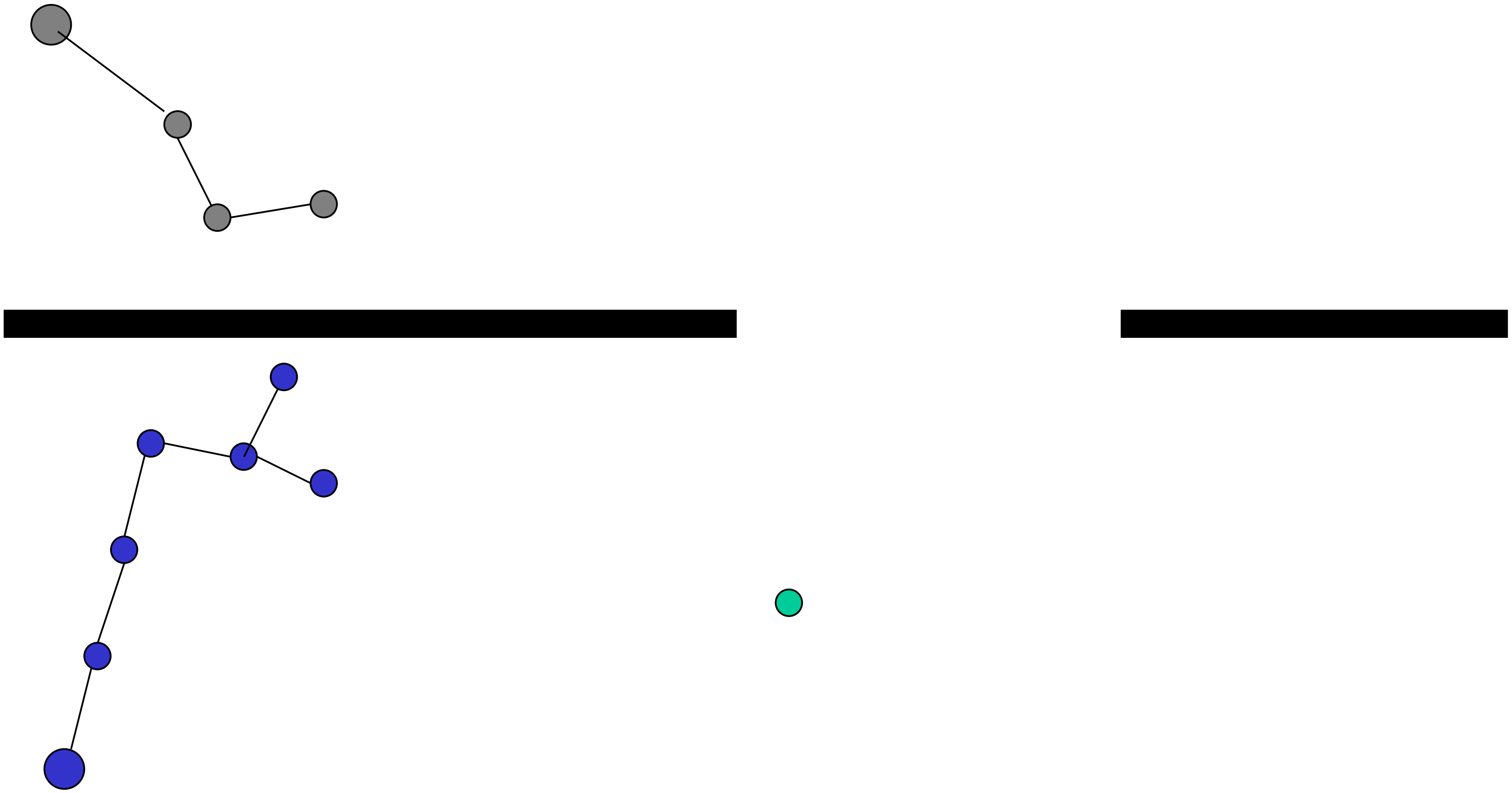
RRT-Connect: example



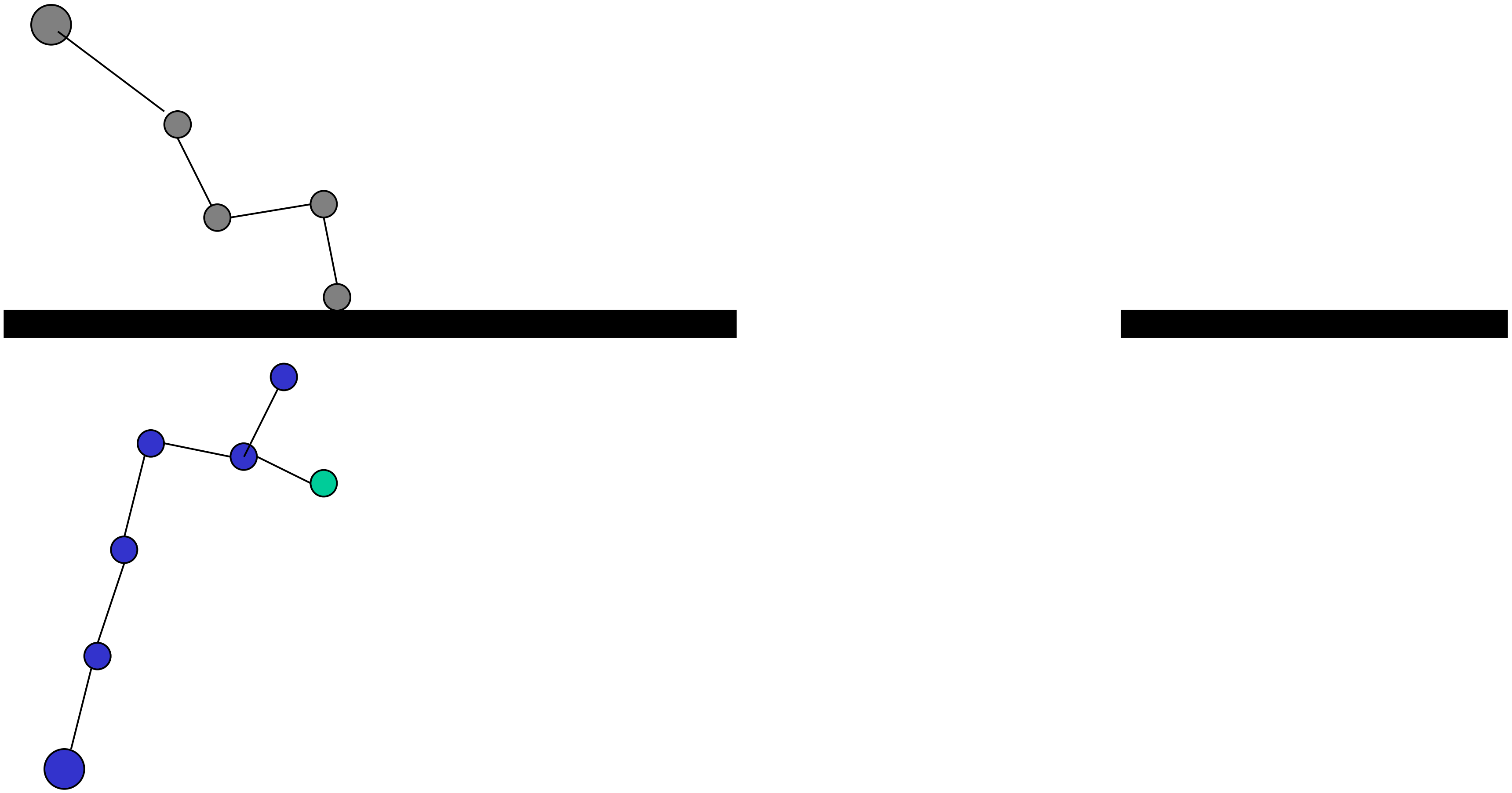
RRT-Connect: example



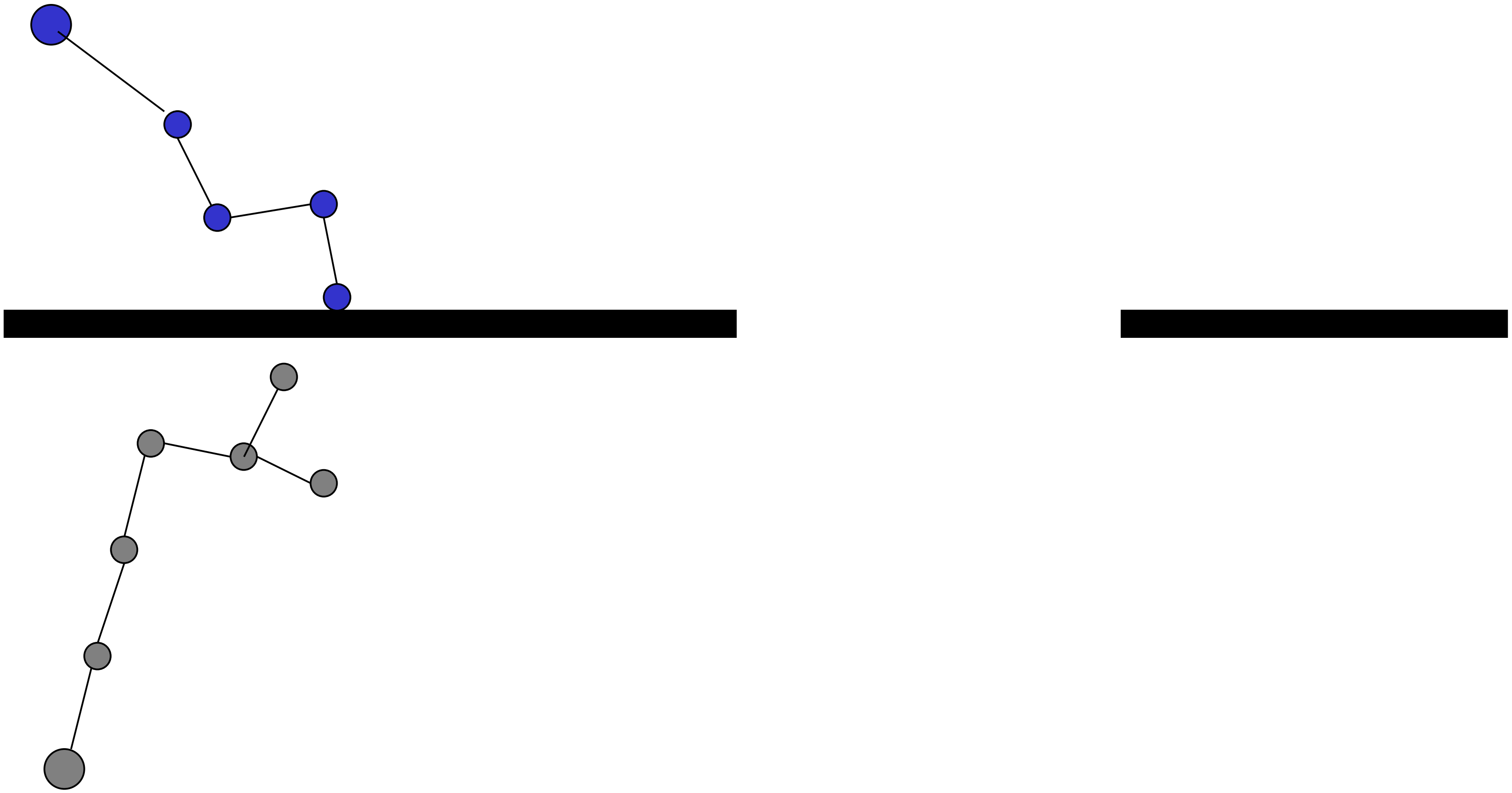
RRT-Connect: example



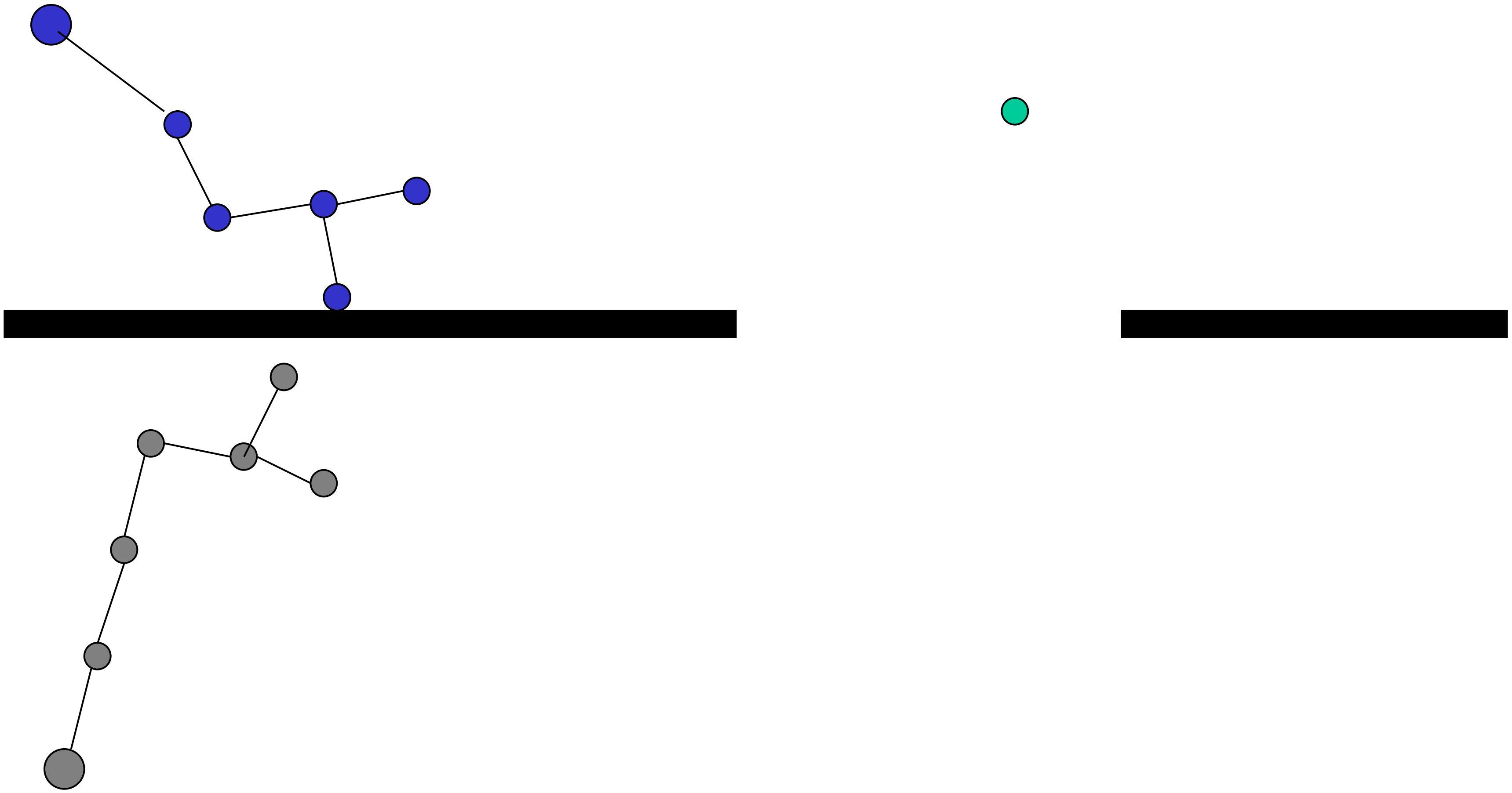
RRT-Connect: example



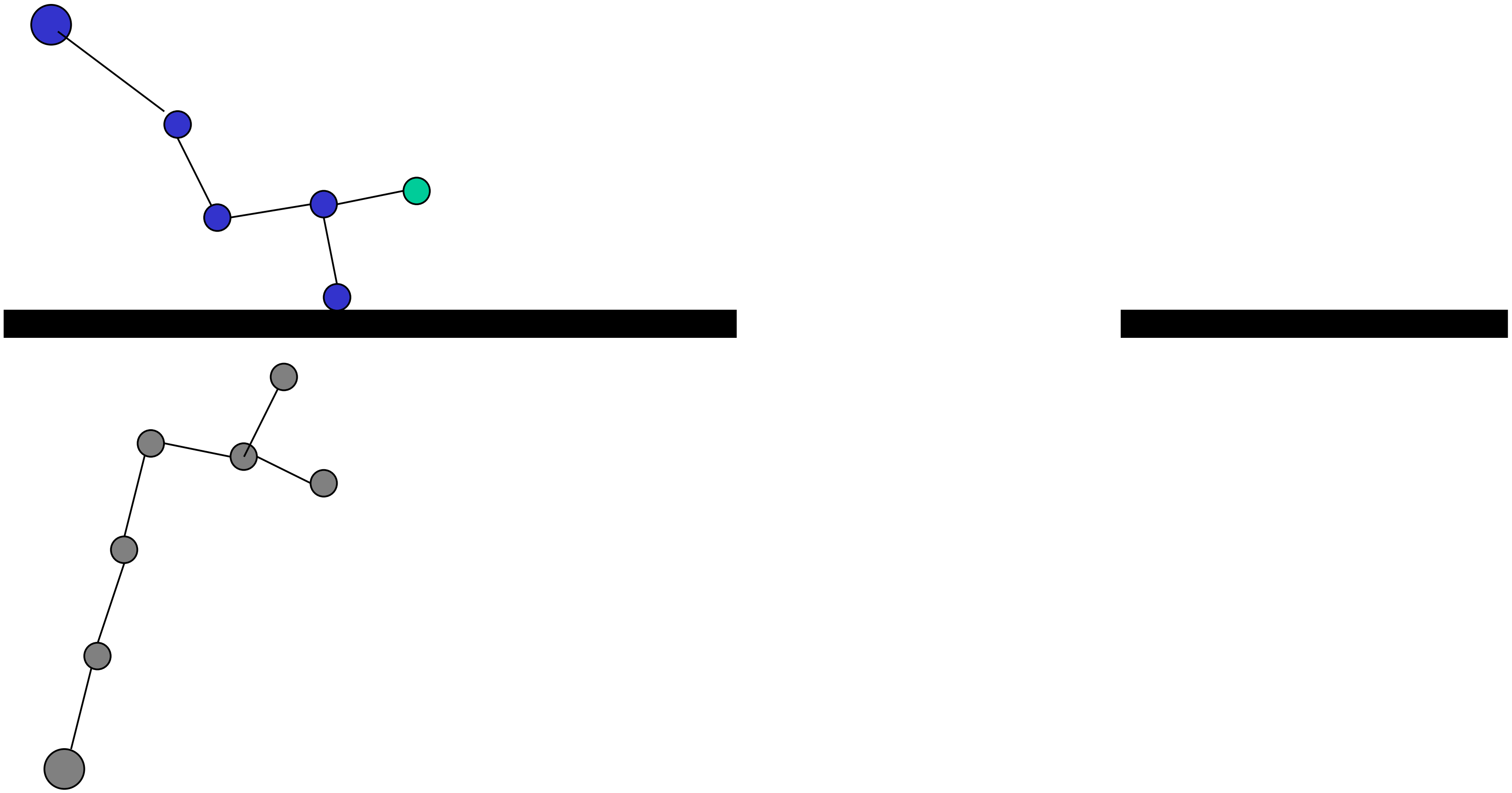
RRT-Connect: example



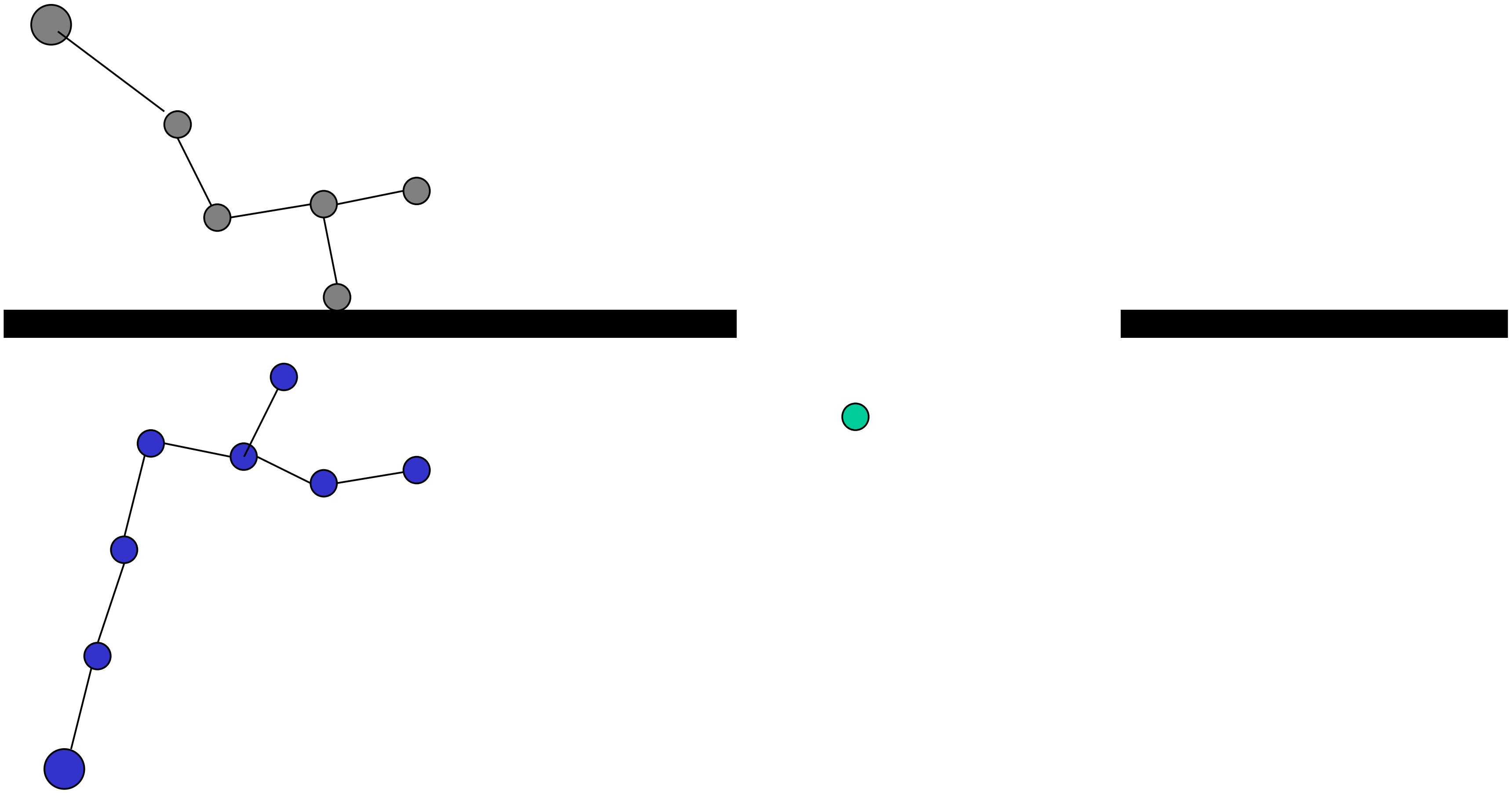
RRT-Connect: example



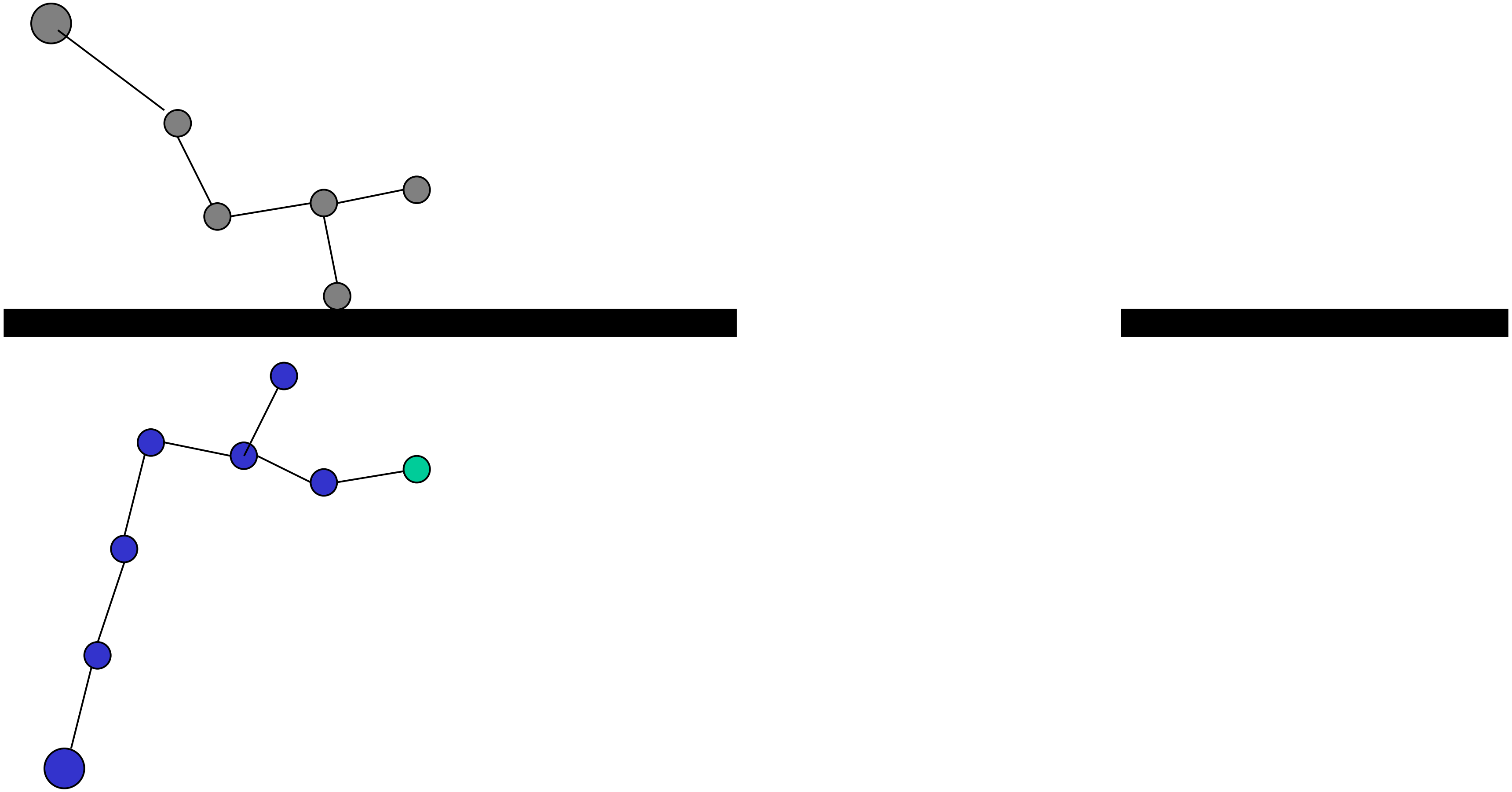
RRT-Connect: example



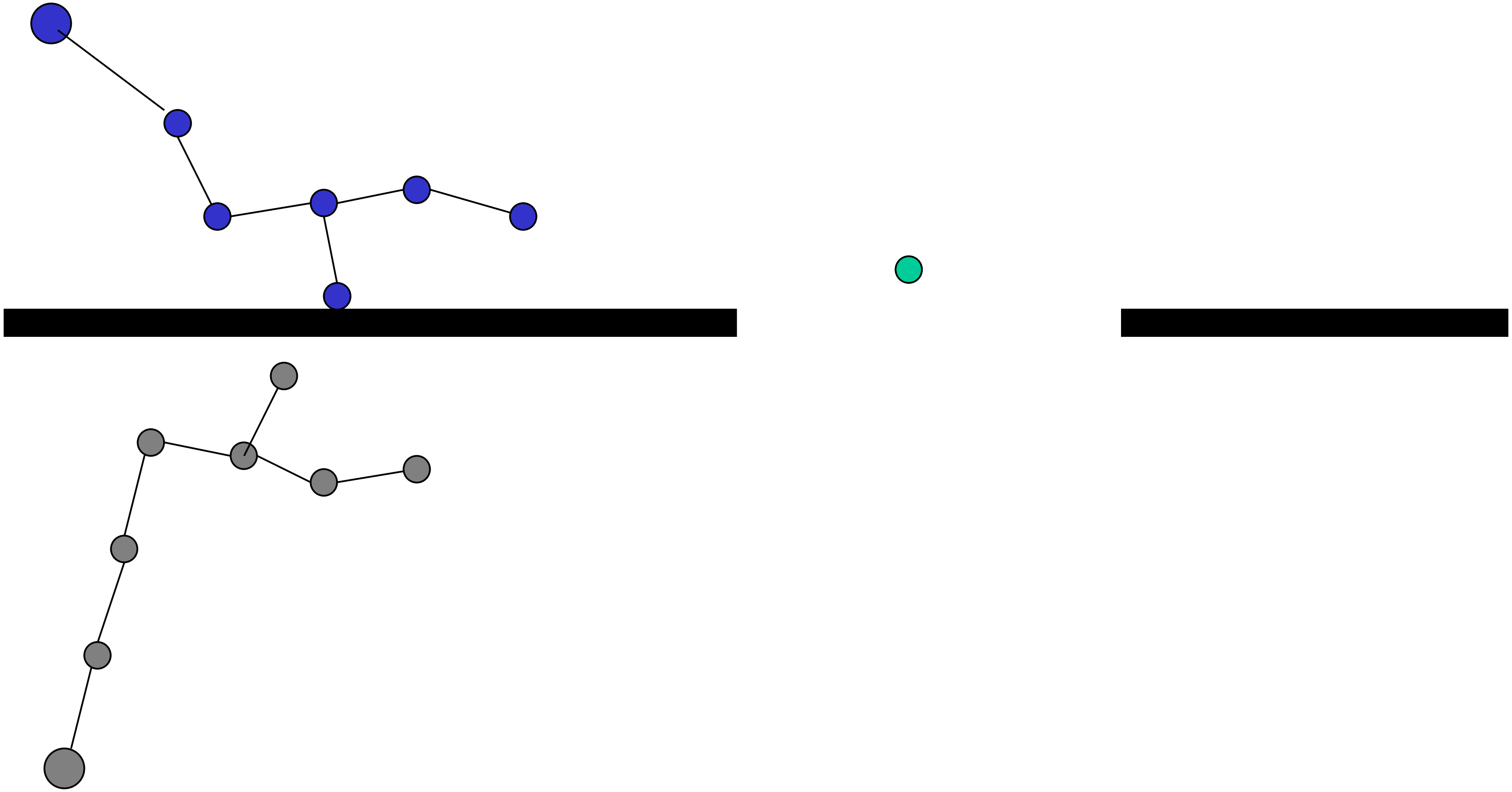
RRT-Connect: example



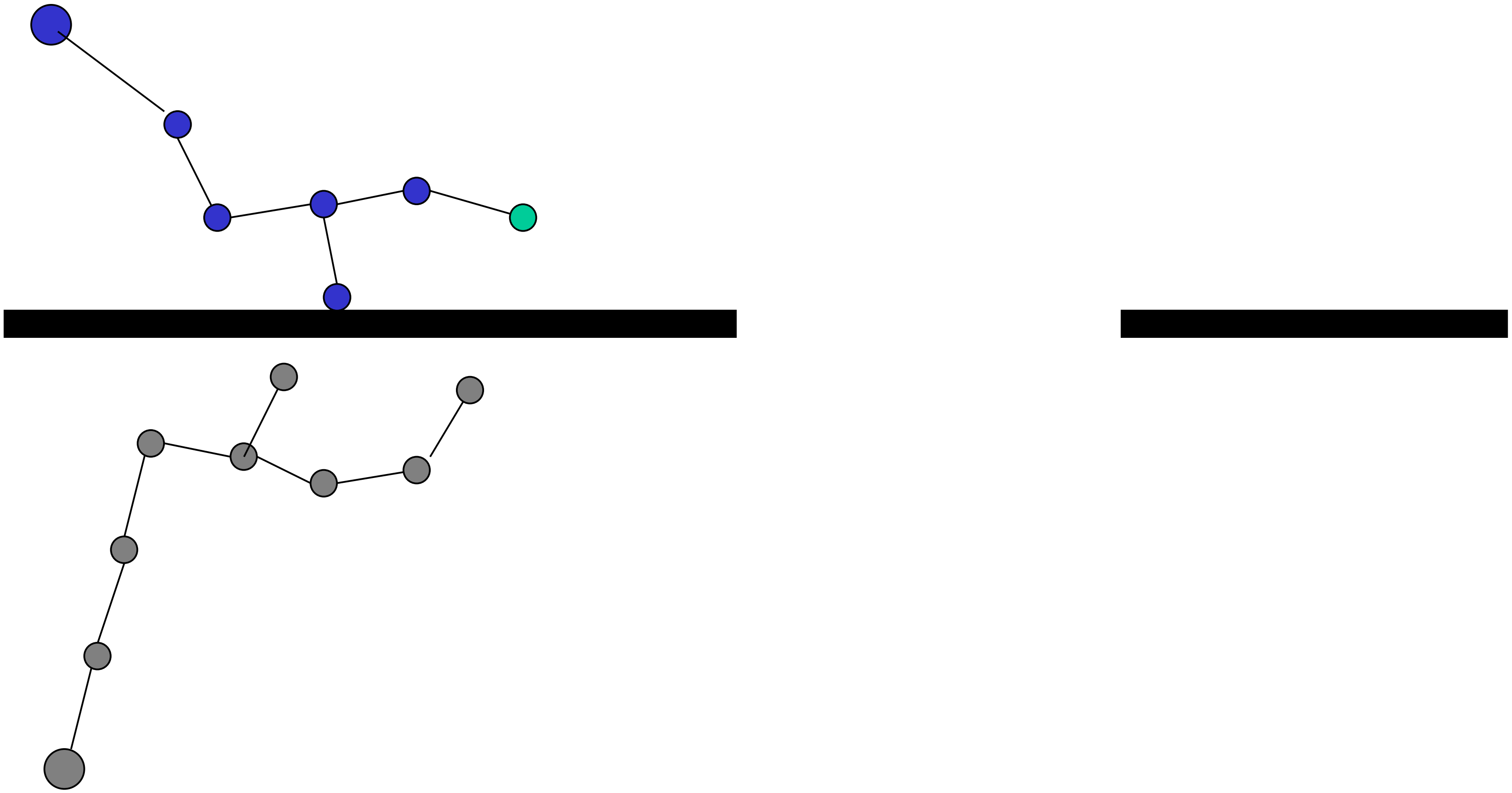
RRT-Connect: example



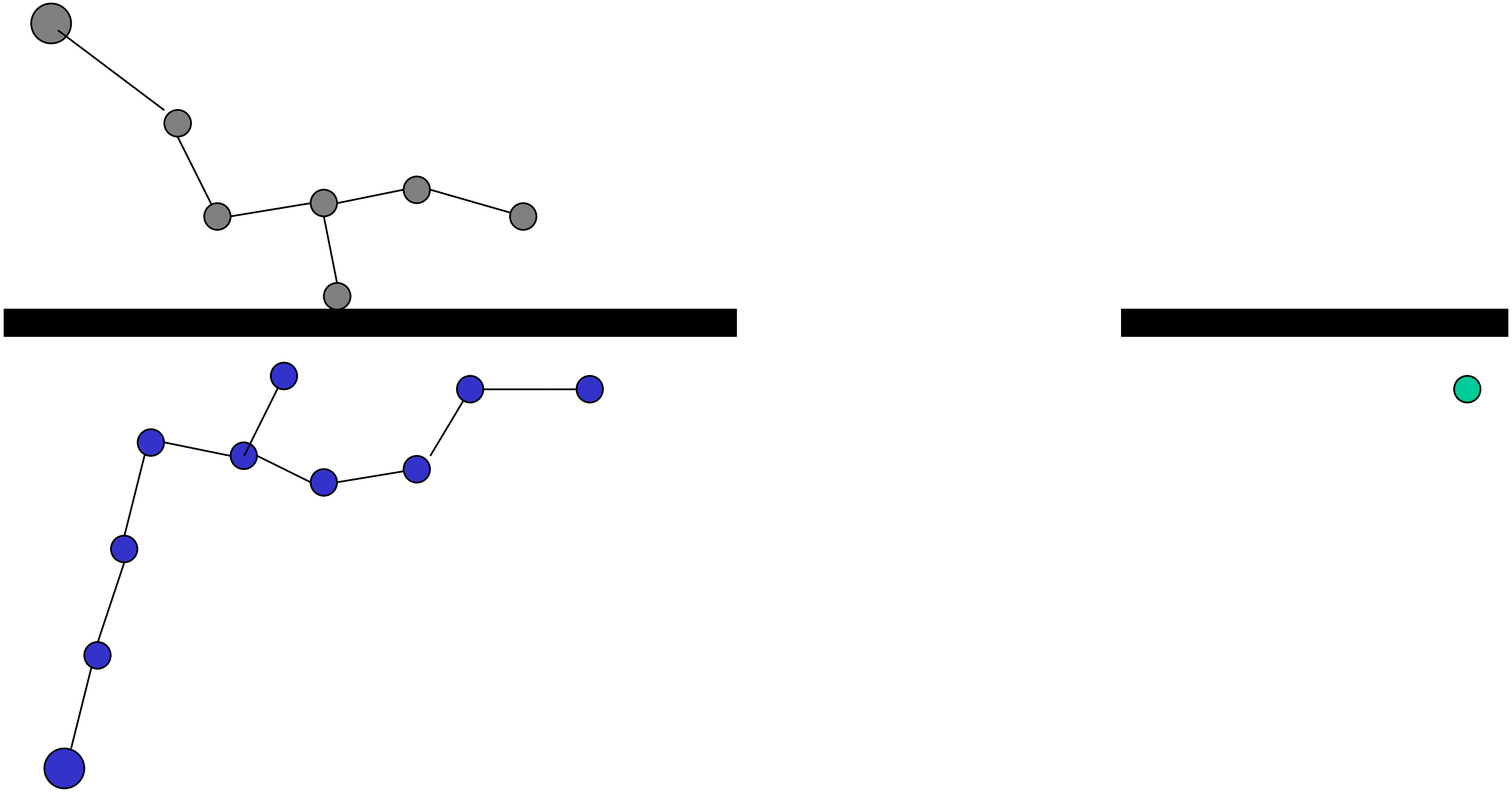
RRT-Connect: example



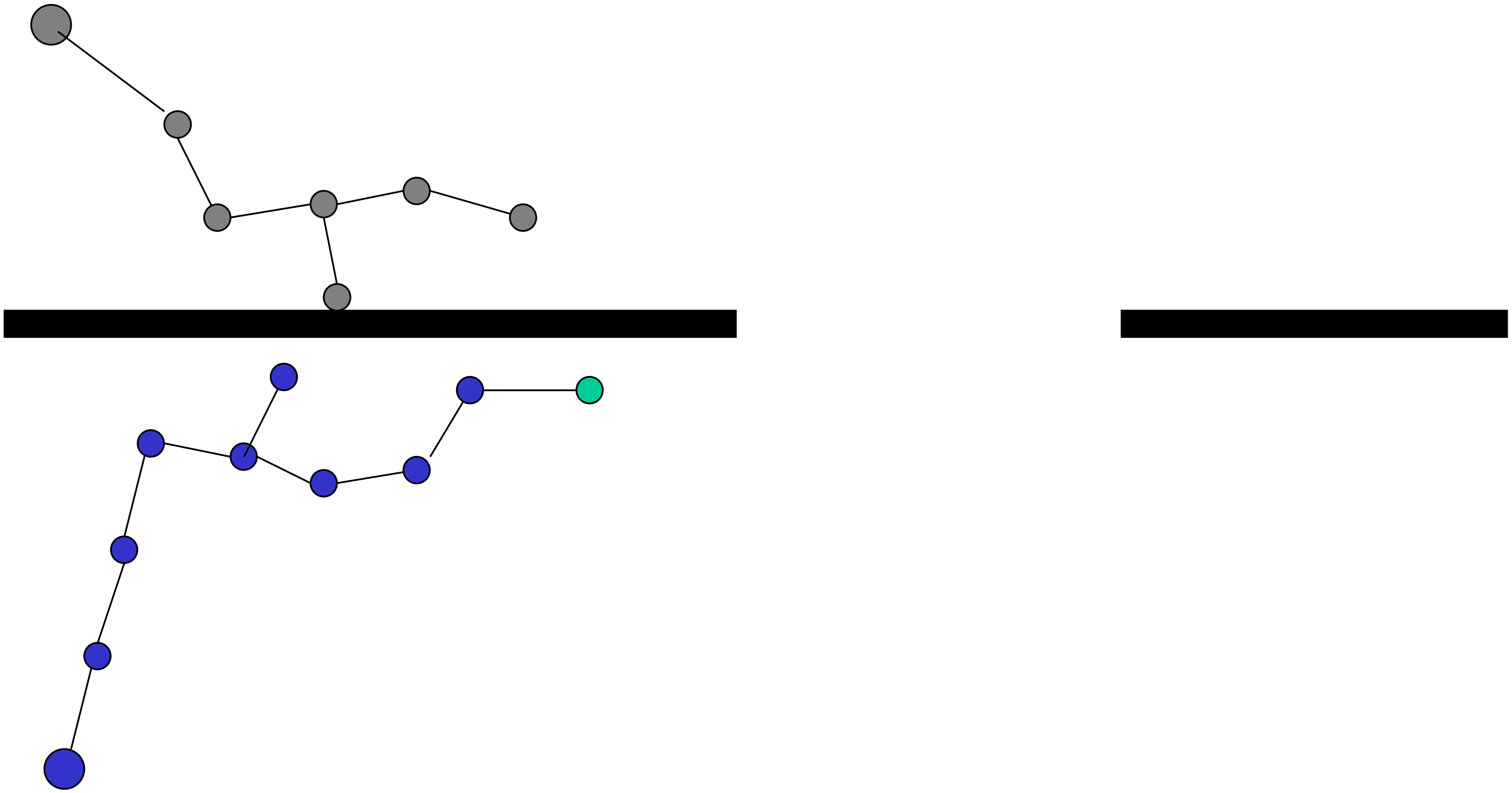
RRT-Connect: example



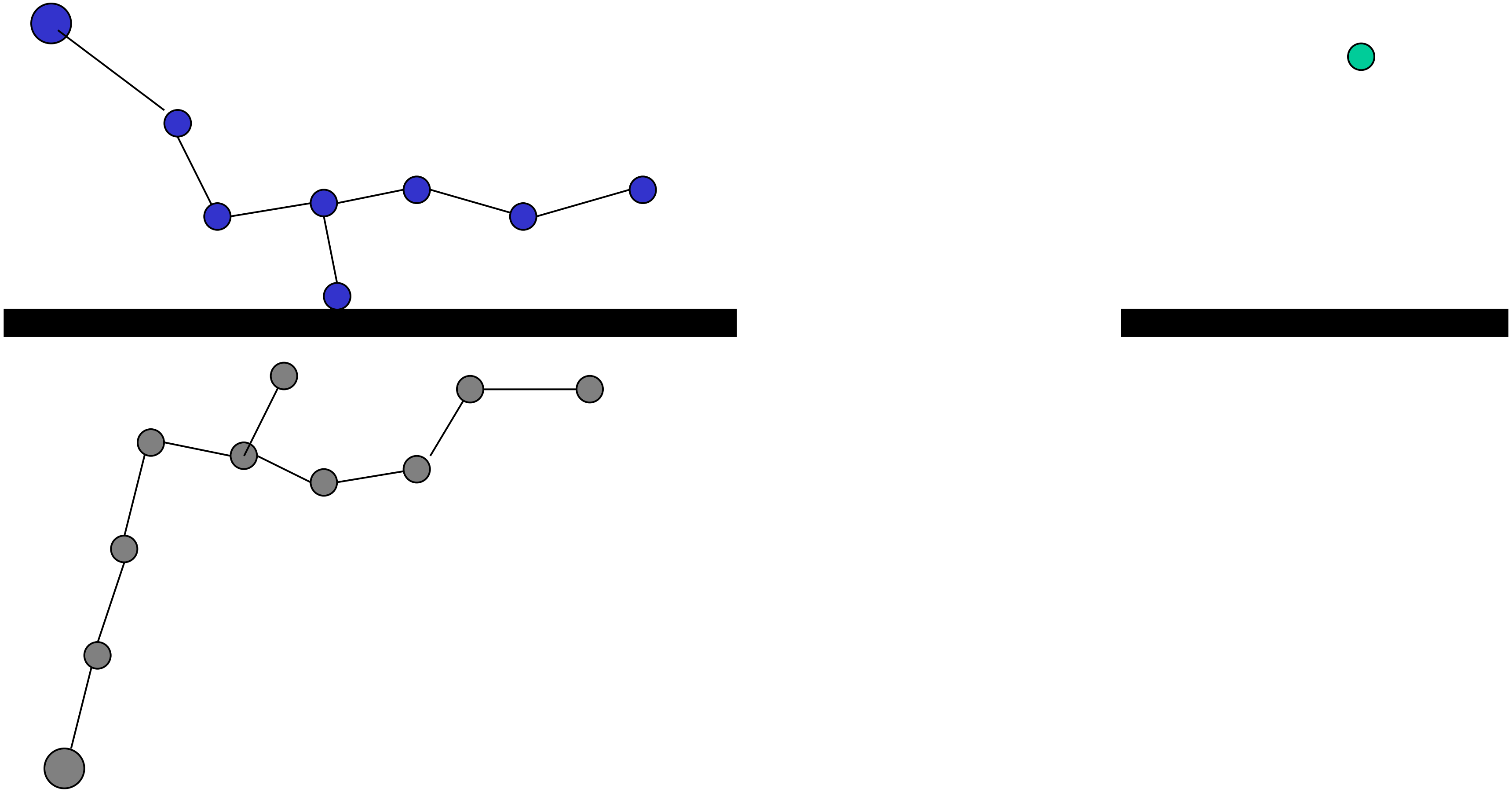
RRT-Connect: example



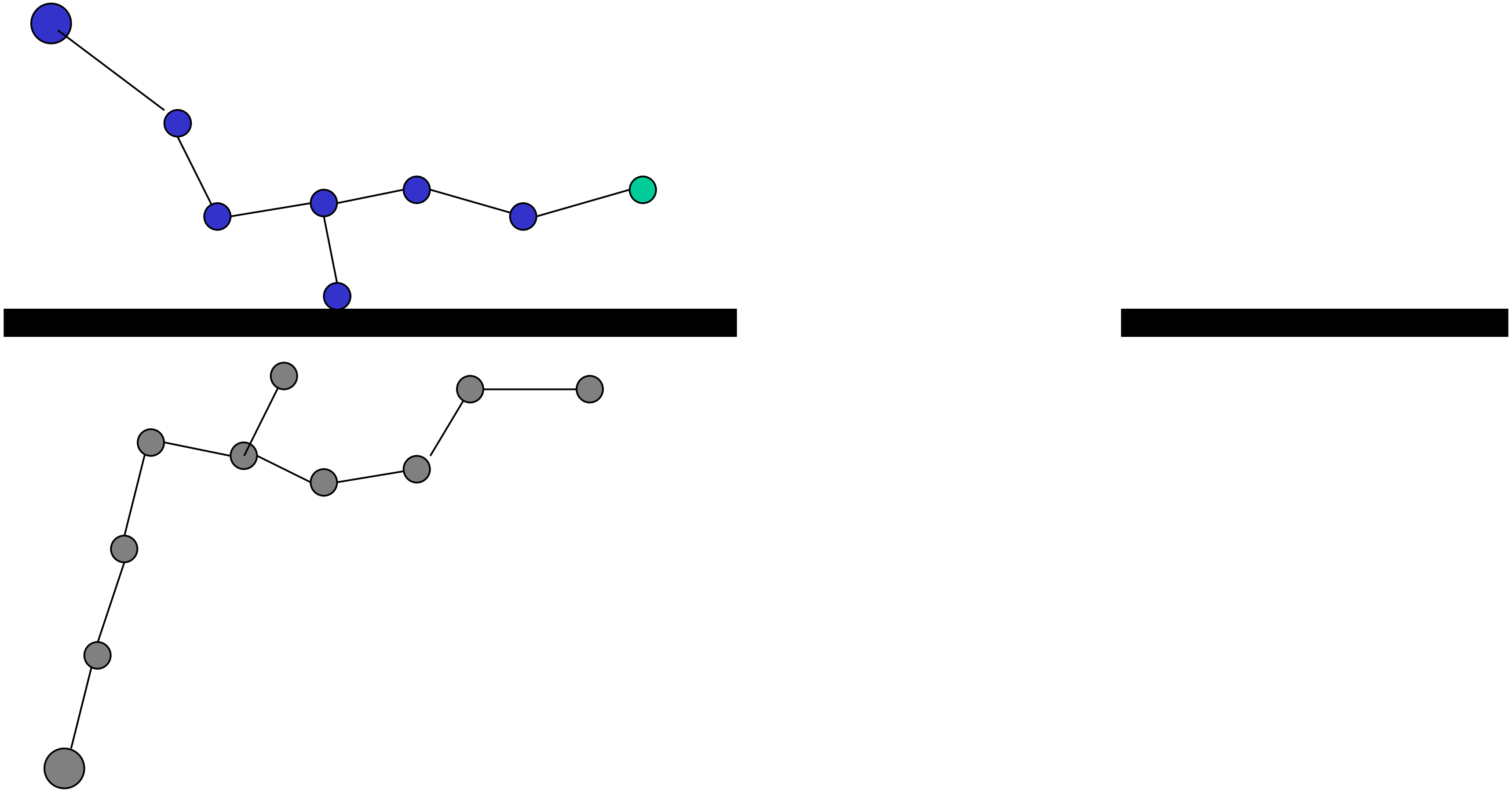
RRT-Connect: example



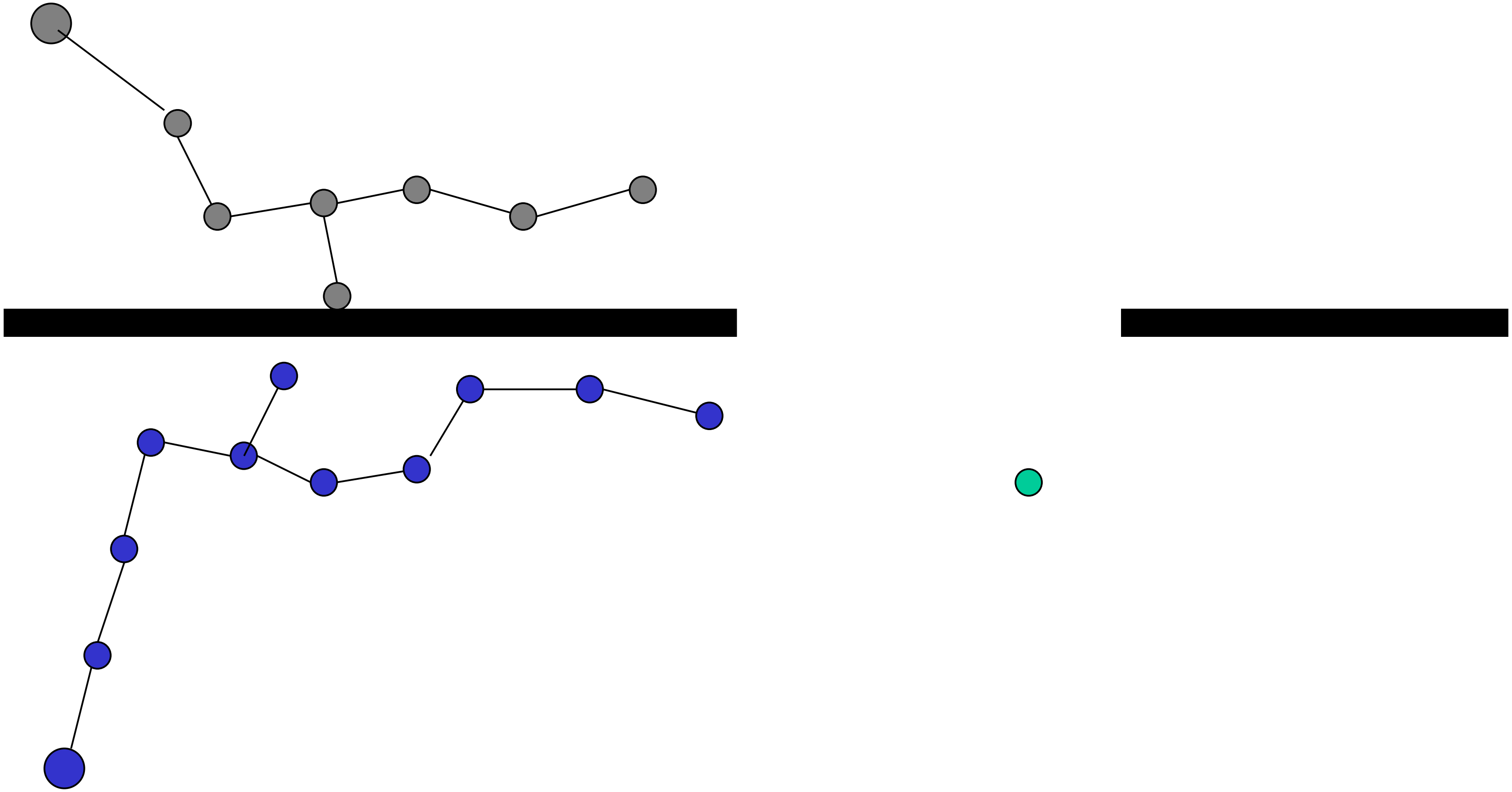
RRT-Connect: example



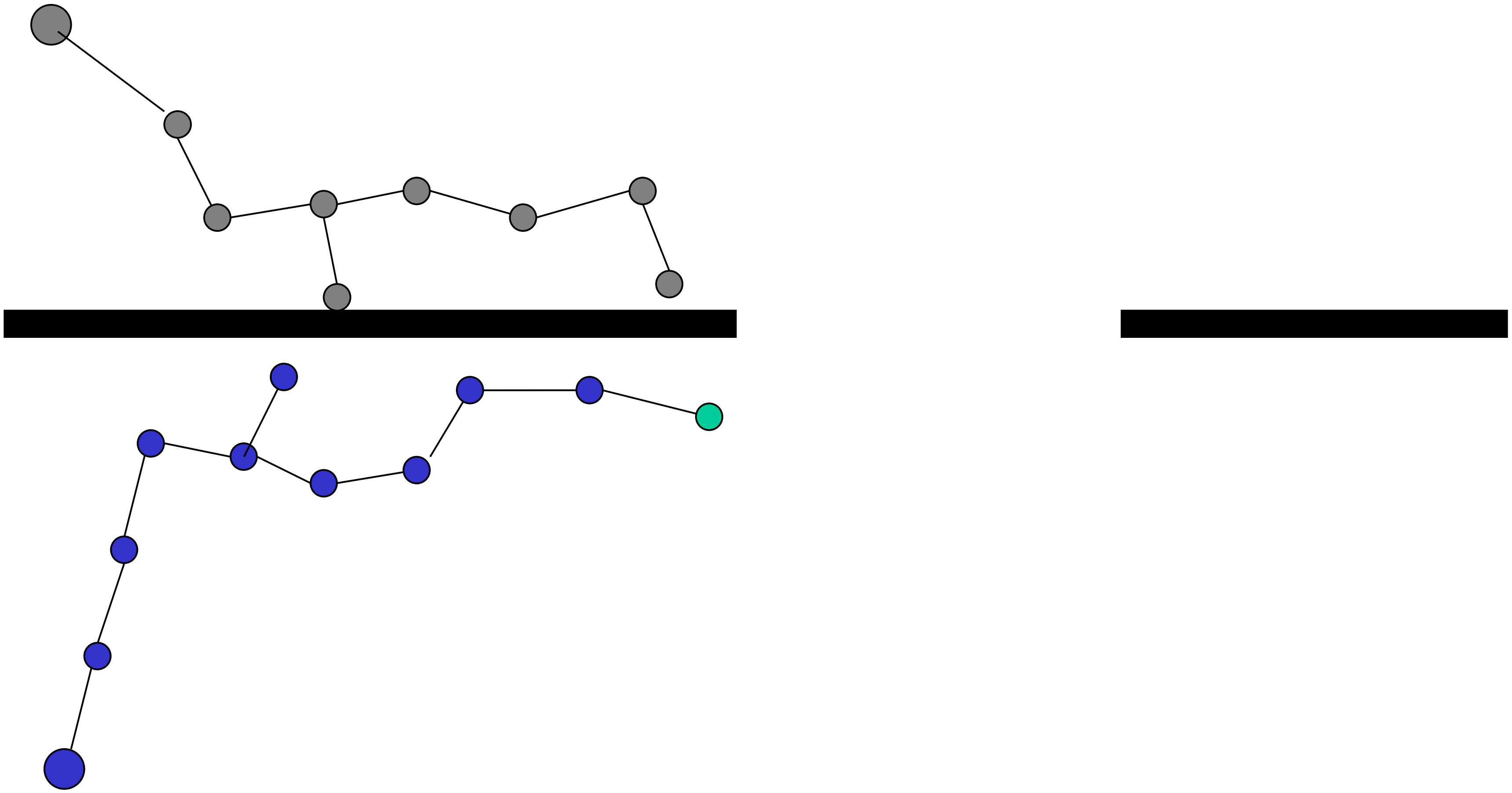
RRT-Connect: example



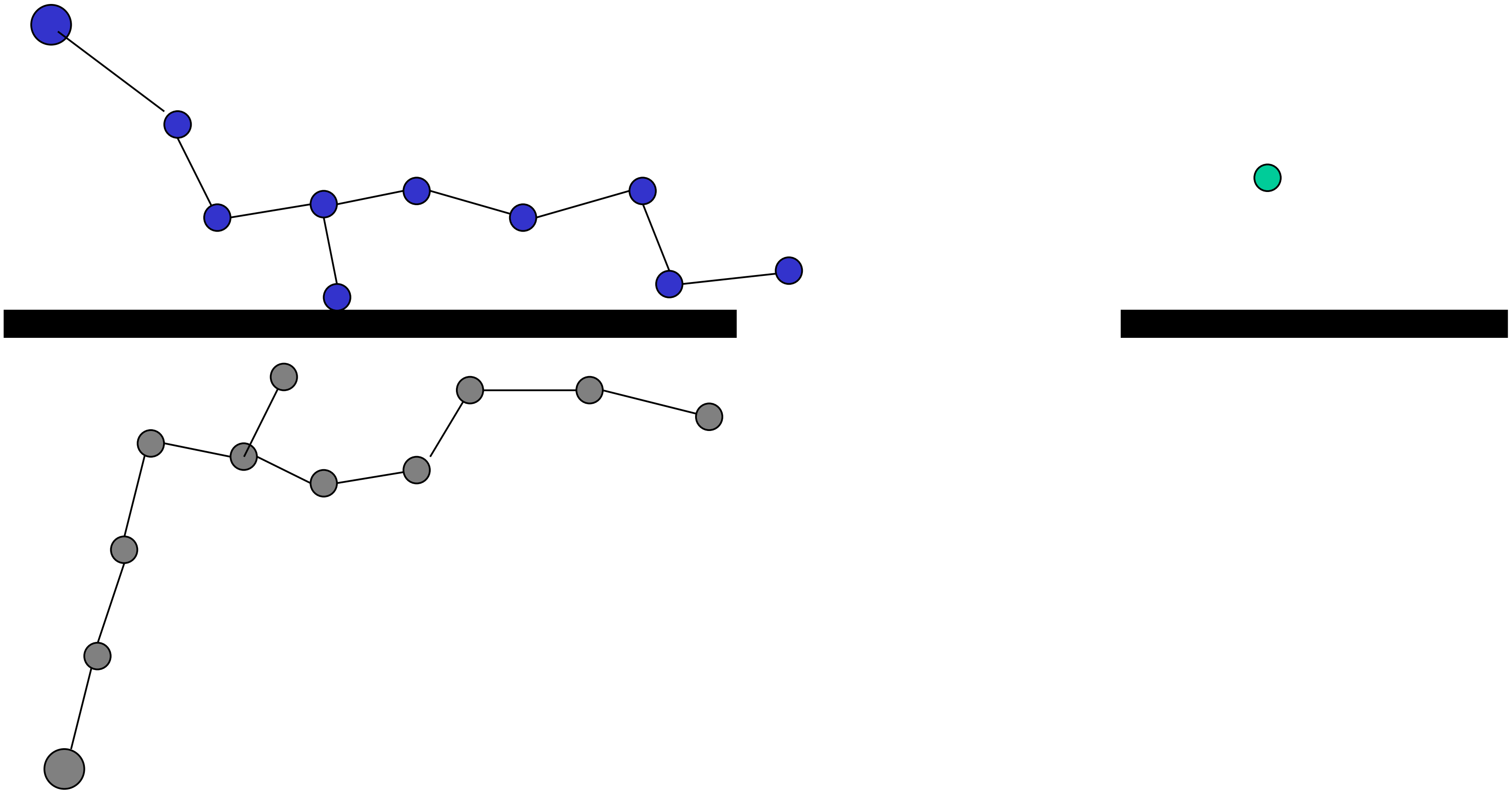
RRT-Connect: example



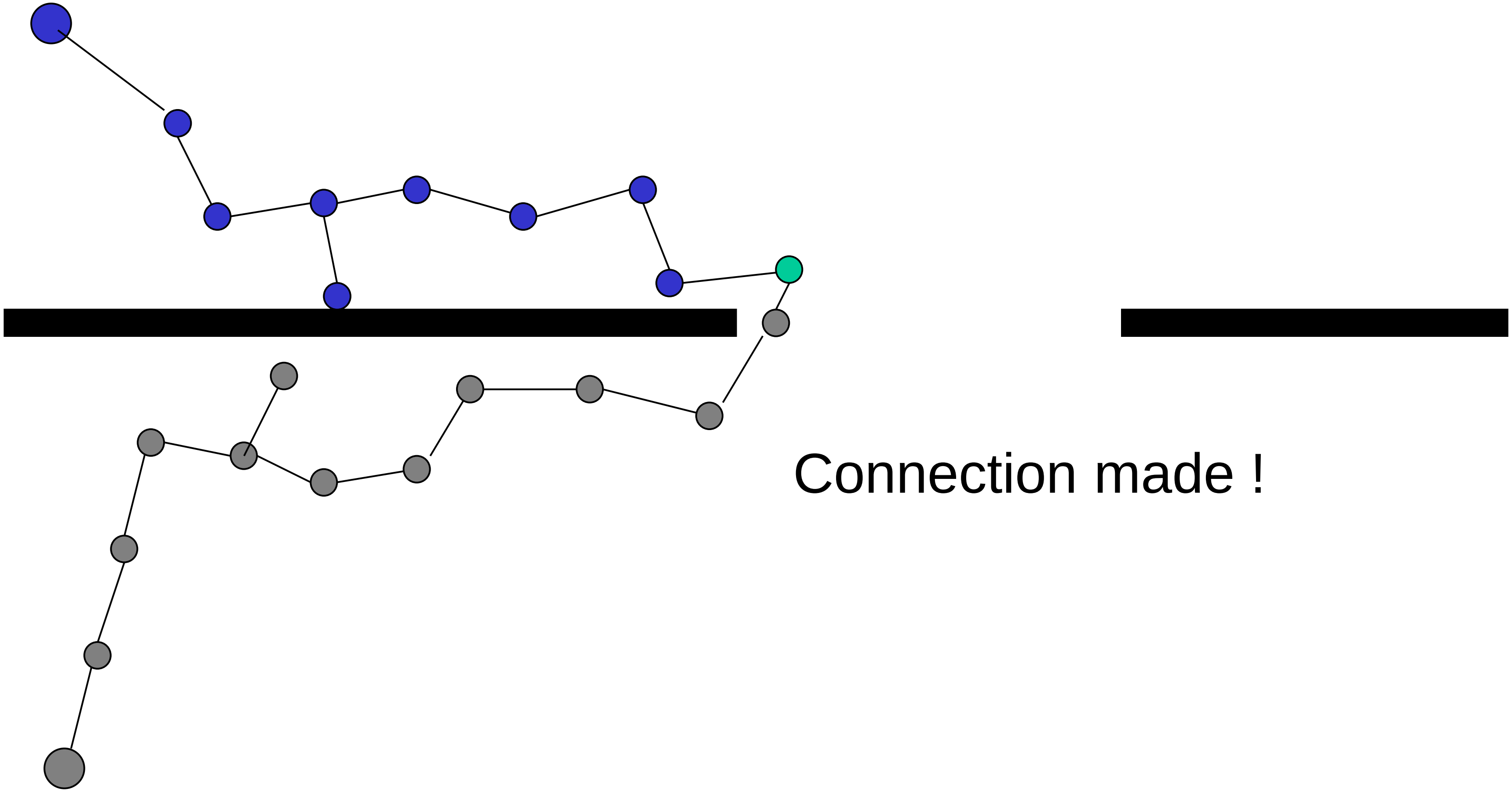
RRT-Connect: example



RRT-Connect: example

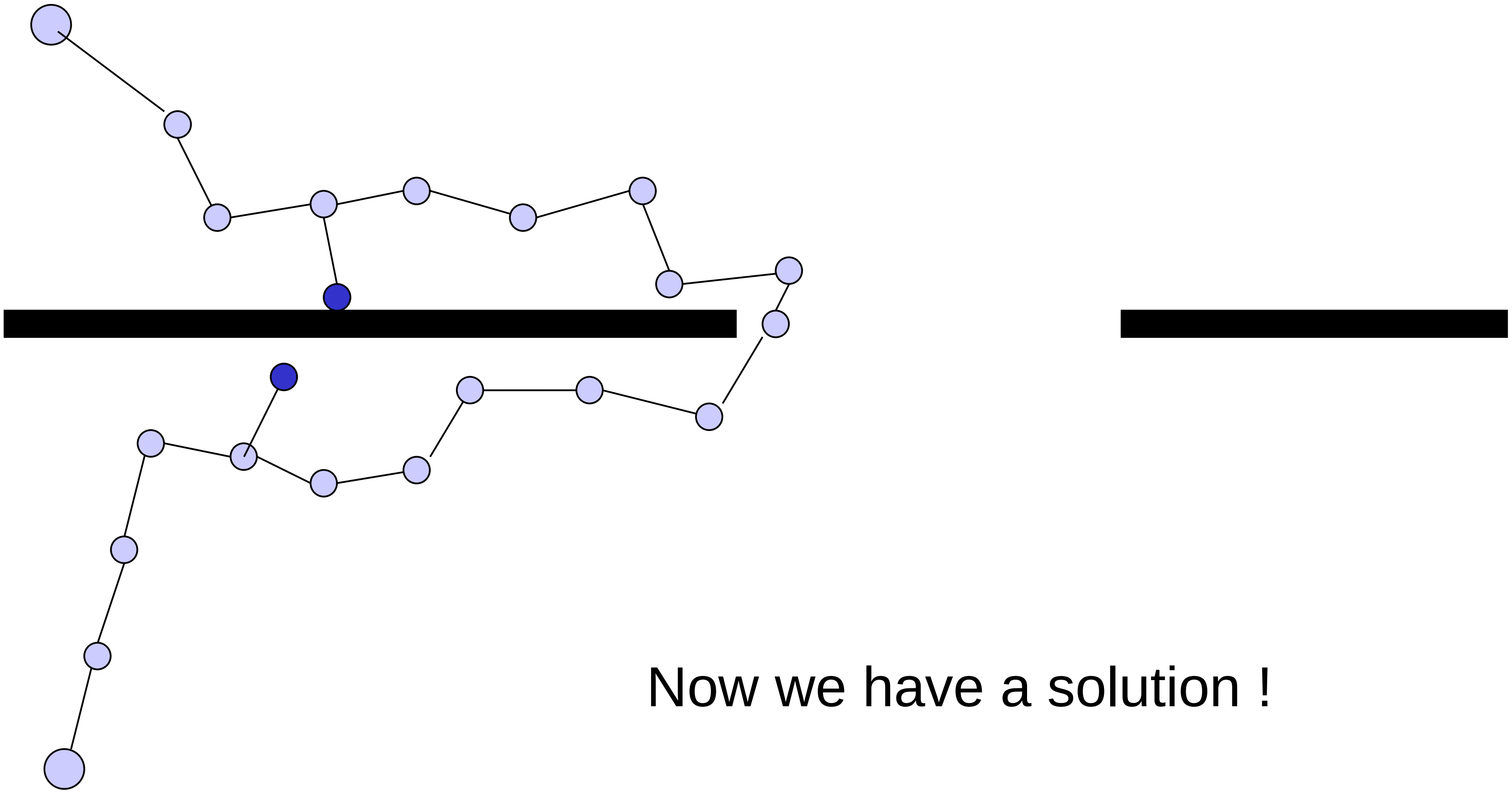


RRT-Connect: example



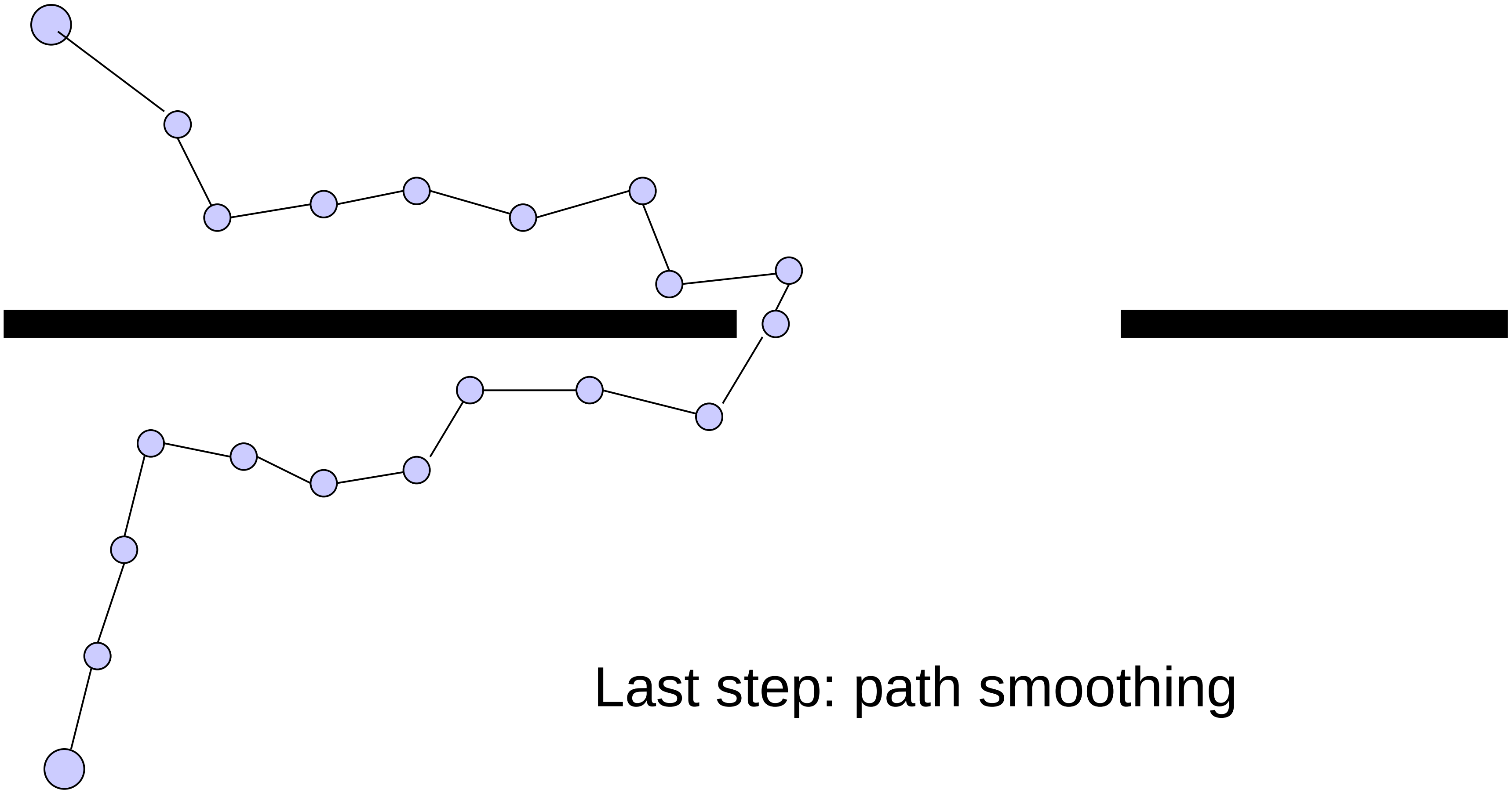
Connection made !

RRT-Connect: example



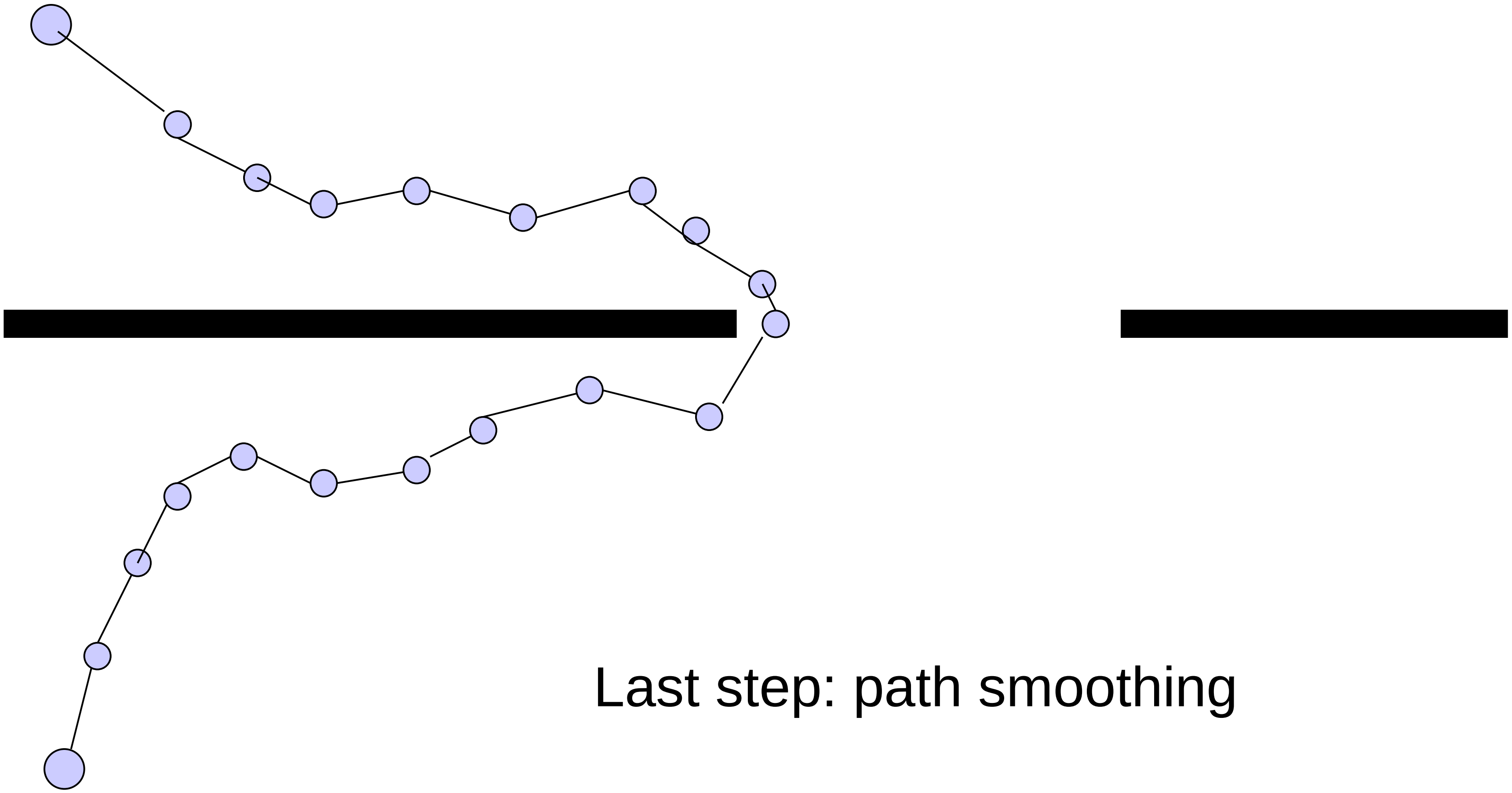
Now we have a solution !

RRT-Connect: example



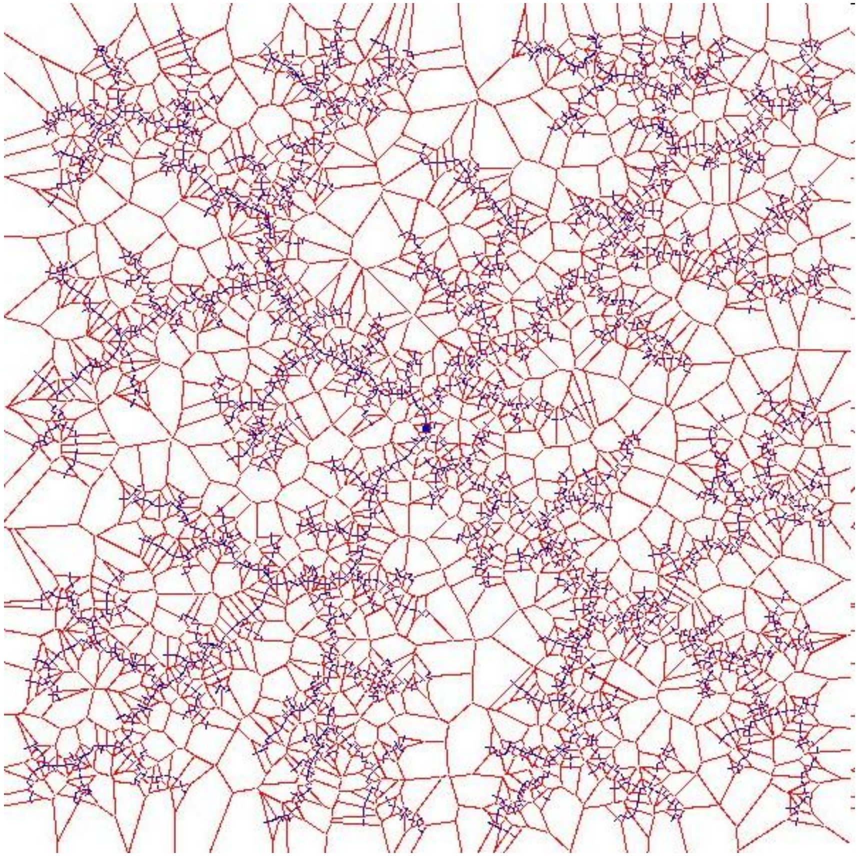
Last step: path smoothing

RRT-Connect: example



Last step: path smoothing

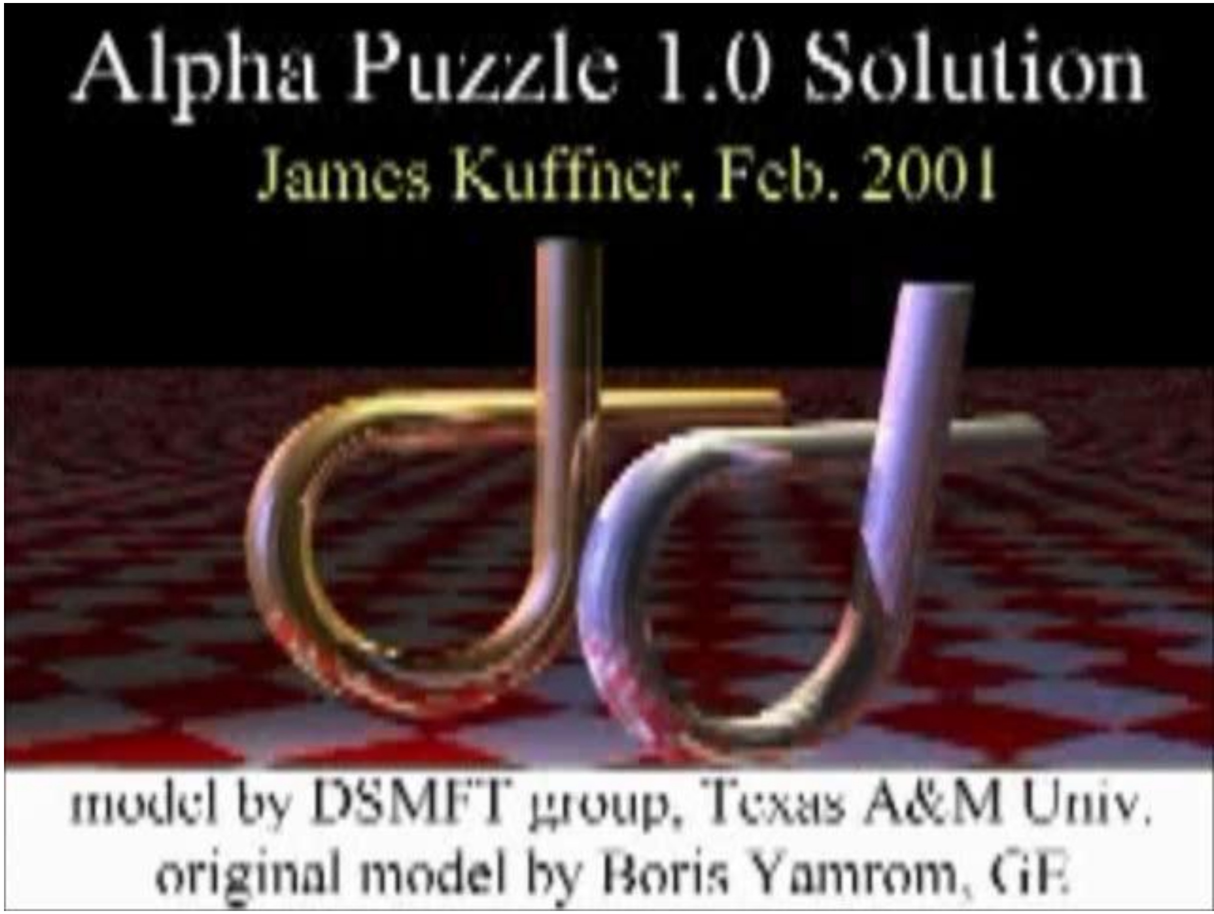
An RRT in 2D



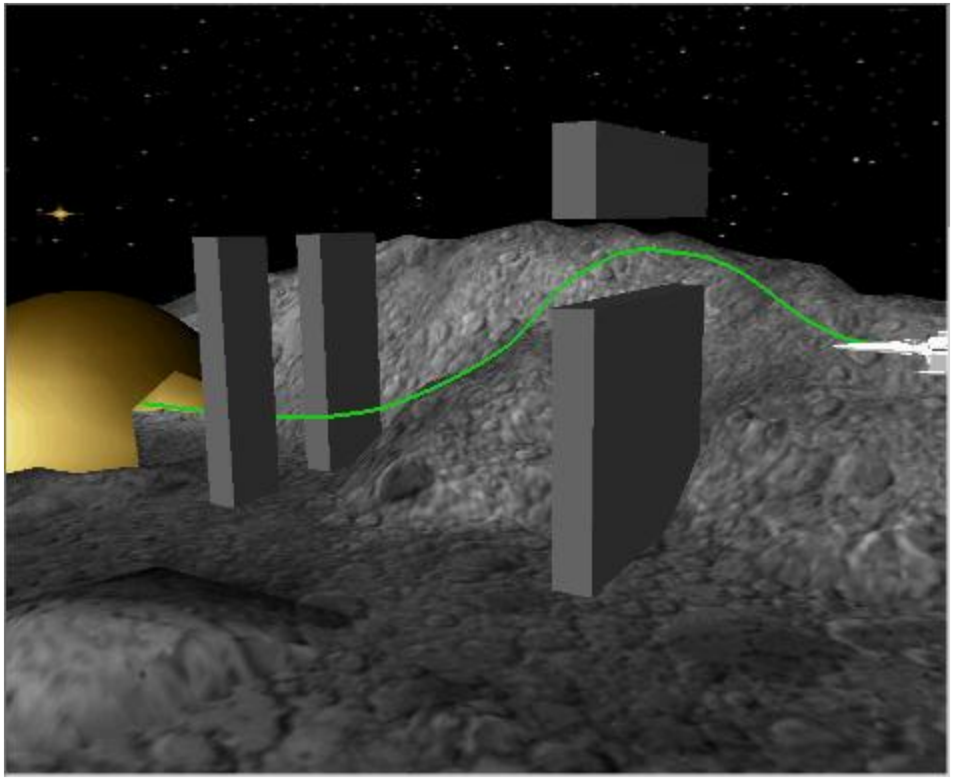
Example from: http://msl.cs.uiuc.edu/rrt/gallery_2drrt.html

A Puzzle solved using RRTs

The goal is the separate the two bars from each other. You might have seen a puzzle like this before. The example was constructed by Boris Yamrom, GE Corporate Research & Development Center, and posted as a research benchmark by Nancy Amato at Texas A&M University. It has been cited in many places as a one of the most challenging motion planning examples. In 2001, it was solved by using a balanced bidirectional RRT, developed by James Kuffner and Steve LaValle. There are no special heuristics or parameters that were tuned specifically for this problem. On a current PC (circa 2003), it consistently takes a few minutes to solve.



Lunar Landing



The following is an open loop trajectory that was planned in a 12-dimensional state space. The video shows an X-Wing fighter that must fly through structures on a lunar base before entering the hangar. This result was presented by Steve LaValle and James Kuffner at the Workshop on the Algorithmic Foundations of Robotics, 2000.