Robot Motion Planning

Rafael Murrieta Cid
CIMAT
Email: murrieta@cimat.mx
Web Page:
http://www.cimat.mx/~murrieta/

Slides were borrowed from
Profs. J.C. Latombe and J.P. Laumond

Goal of Motion Planning

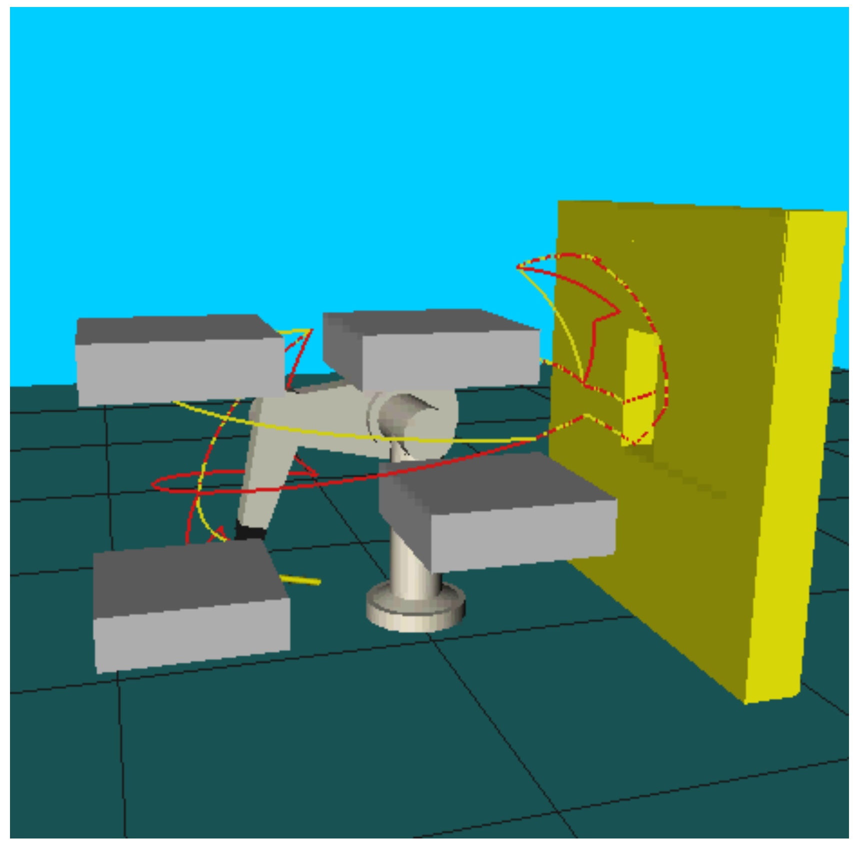
- Compute **motion strategies**, e.g.:
  - geometric paths
  - time-parameterized trajectories
  - sequence of sensor-based motion commands
- To achieve **high-level goals**, e.g.:
  - go to A without colliding with obstacles
  - assemble product P
  - build map of environment E
  - find object O

Basic Motion Planning Problem

Compute a **collision-free path** for a rigid or articulated object among static obstacles

- Inputs:
  - Geometry of moving object and obstacles
  - Kinematics of moving object (degrees of freedom)
  - Initial and goal configurations (placements)
- Output:

  Continuous sequence of collision-free robot configurations connecting the initial and goal configurations

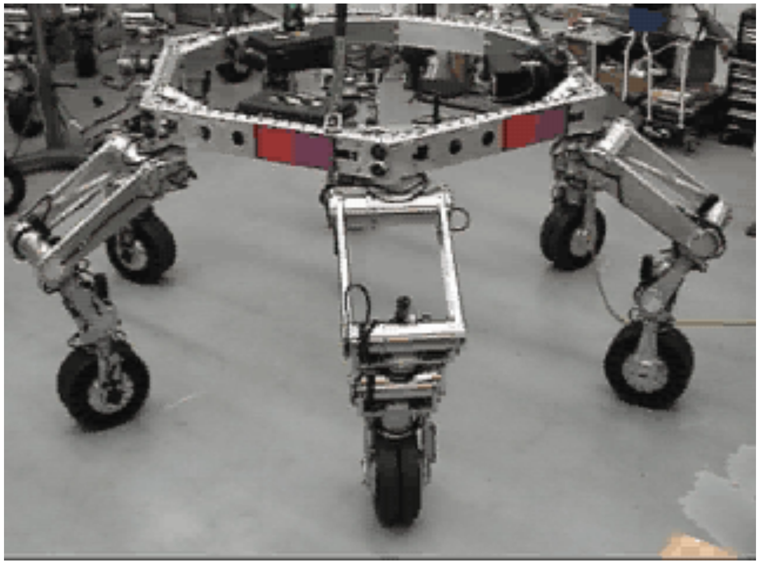


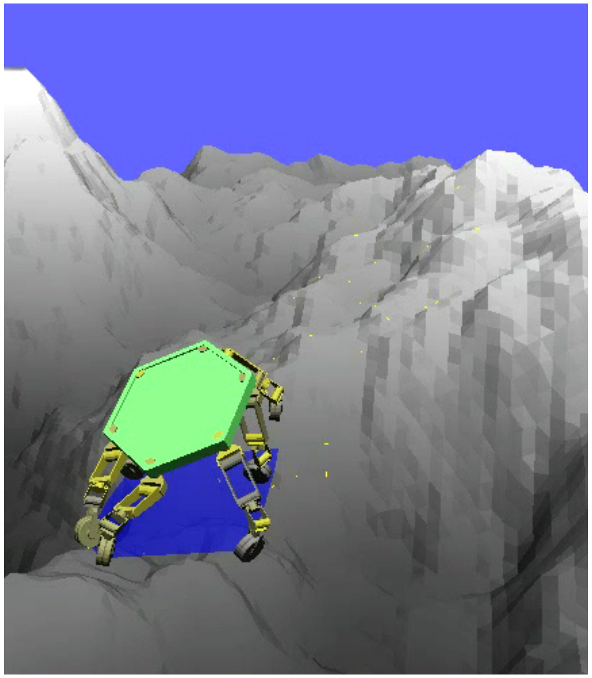
Extensions of Basic Problem

- Moving obstacles
- Multiple robots
- Movable objects
- Assembly planning
- Goal is to acquire information by sensing
  - Model building
  - Object finding/tracking
  - Inspection
- Nonholonomic constraints
- Dynamic constraints
- Stability constraints
- Optimal planning
- Uncertainty in model, control and sensing
- Exploiting task mechanics (sensorless motions, under-actualted systems)
- Physical models and deformable objects
- Integration of planning and control
- Integration with higher-level planning

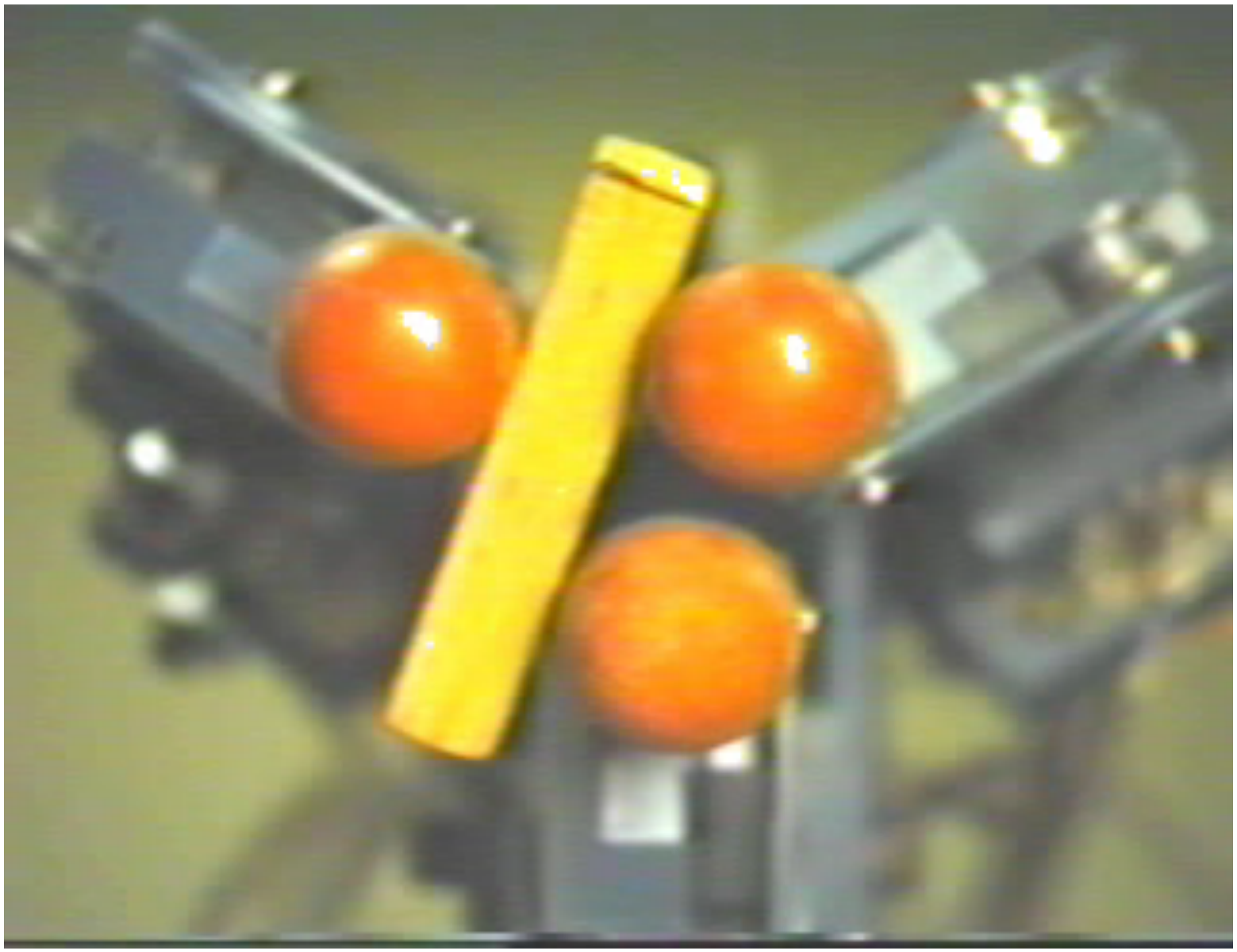
Some Applications

Lunar Vehicle (ATHLETE, NASA/JPL)

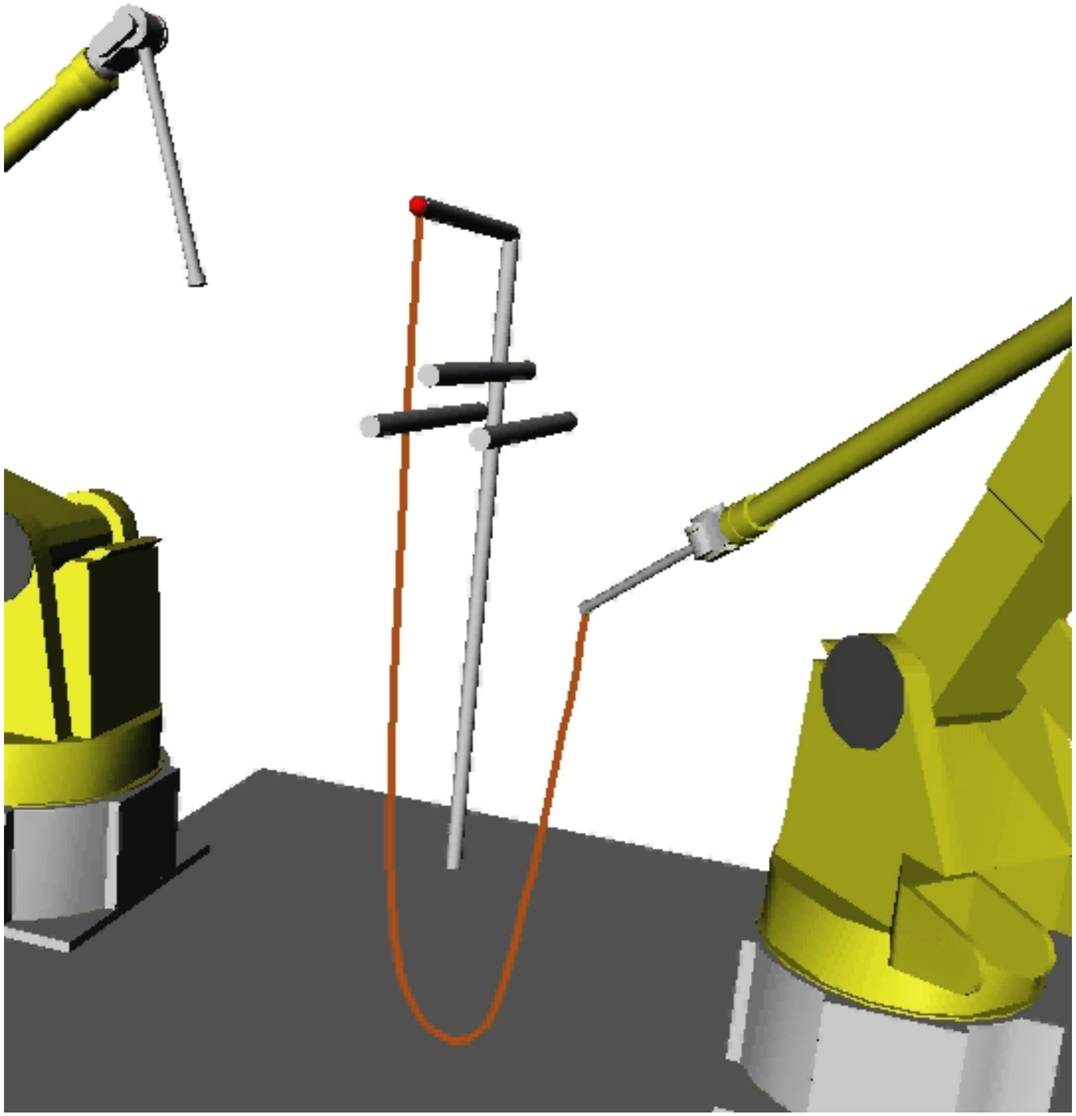




Dexterous Manipulation

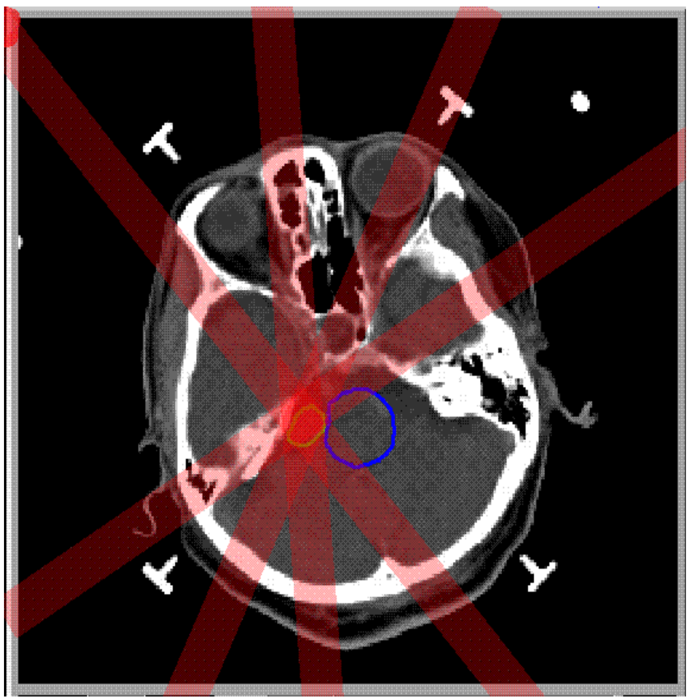


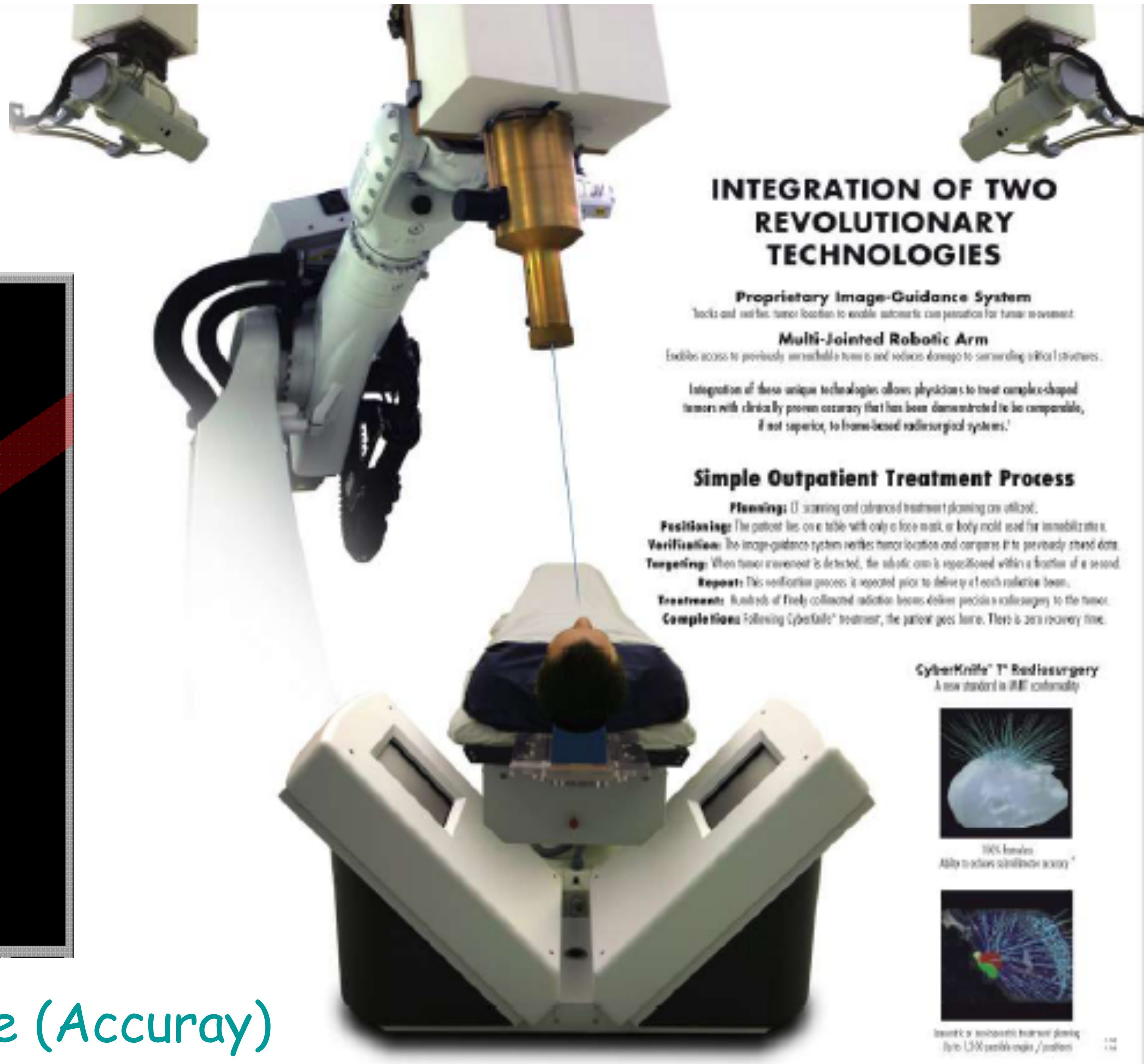
Manipulation of Deformable Objects



Topologically defined goal

Radiosurgical Planning



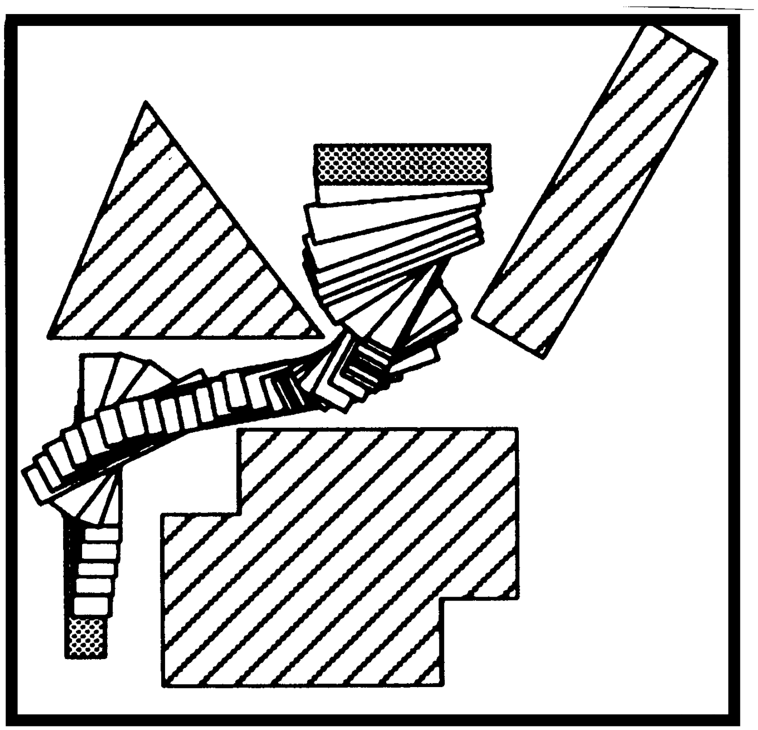


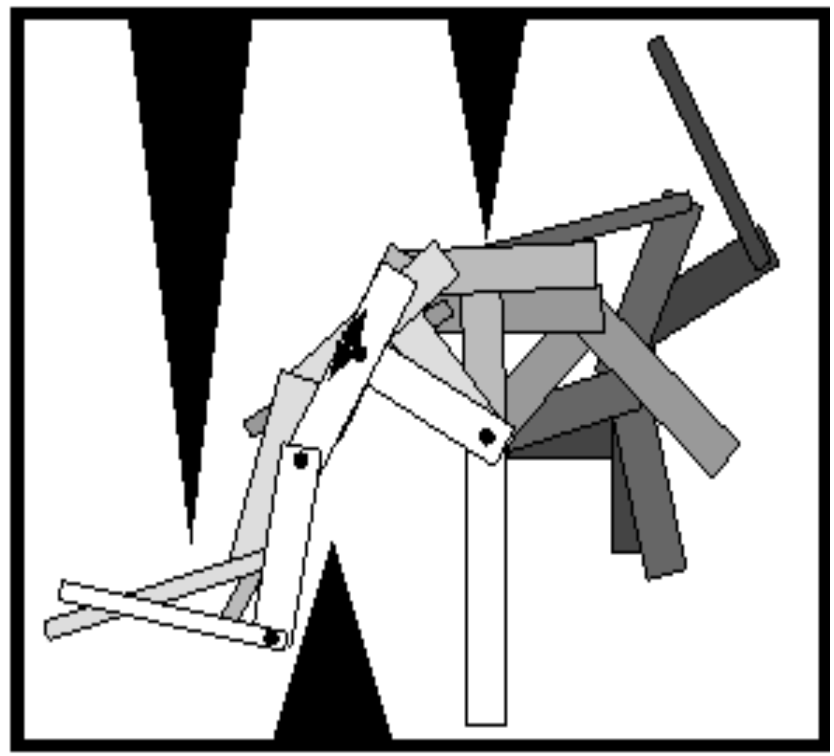
CyberKnife (Accuray)

Transportation of A380 Fuselage through Small Villages

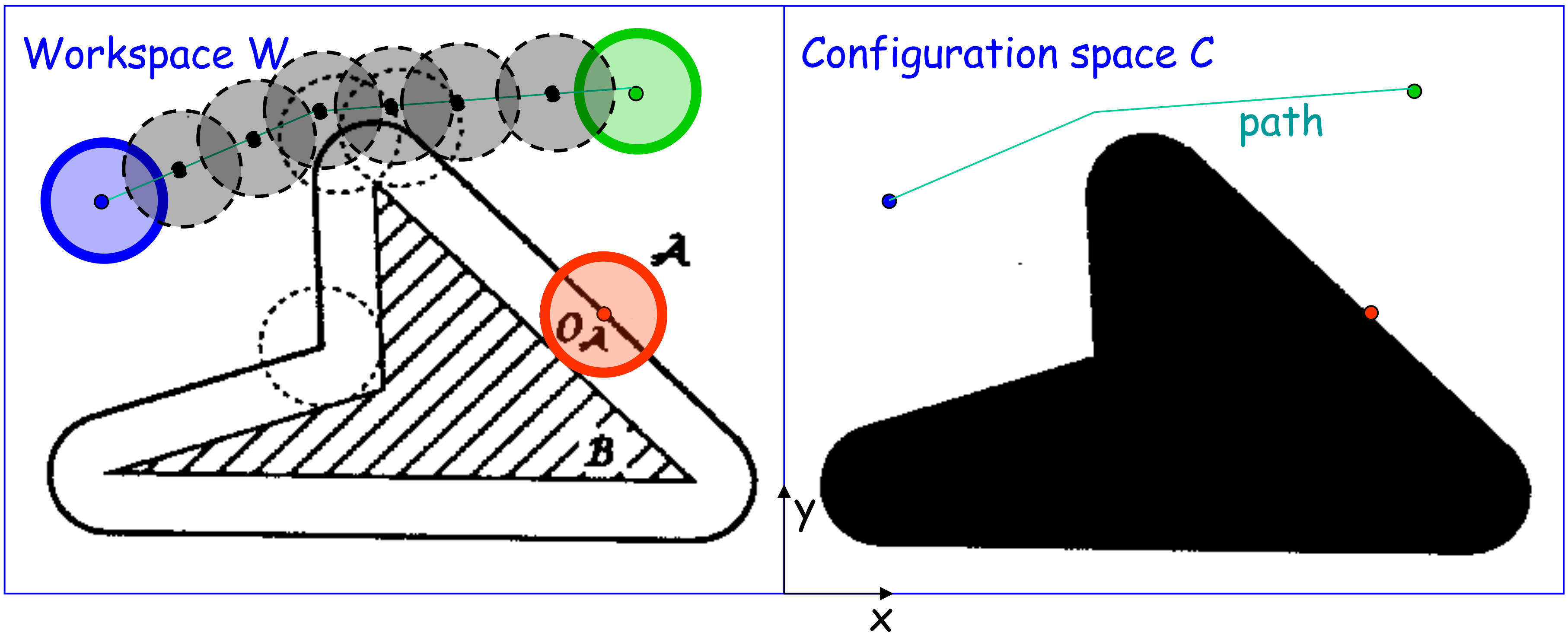
Kineo

Paths



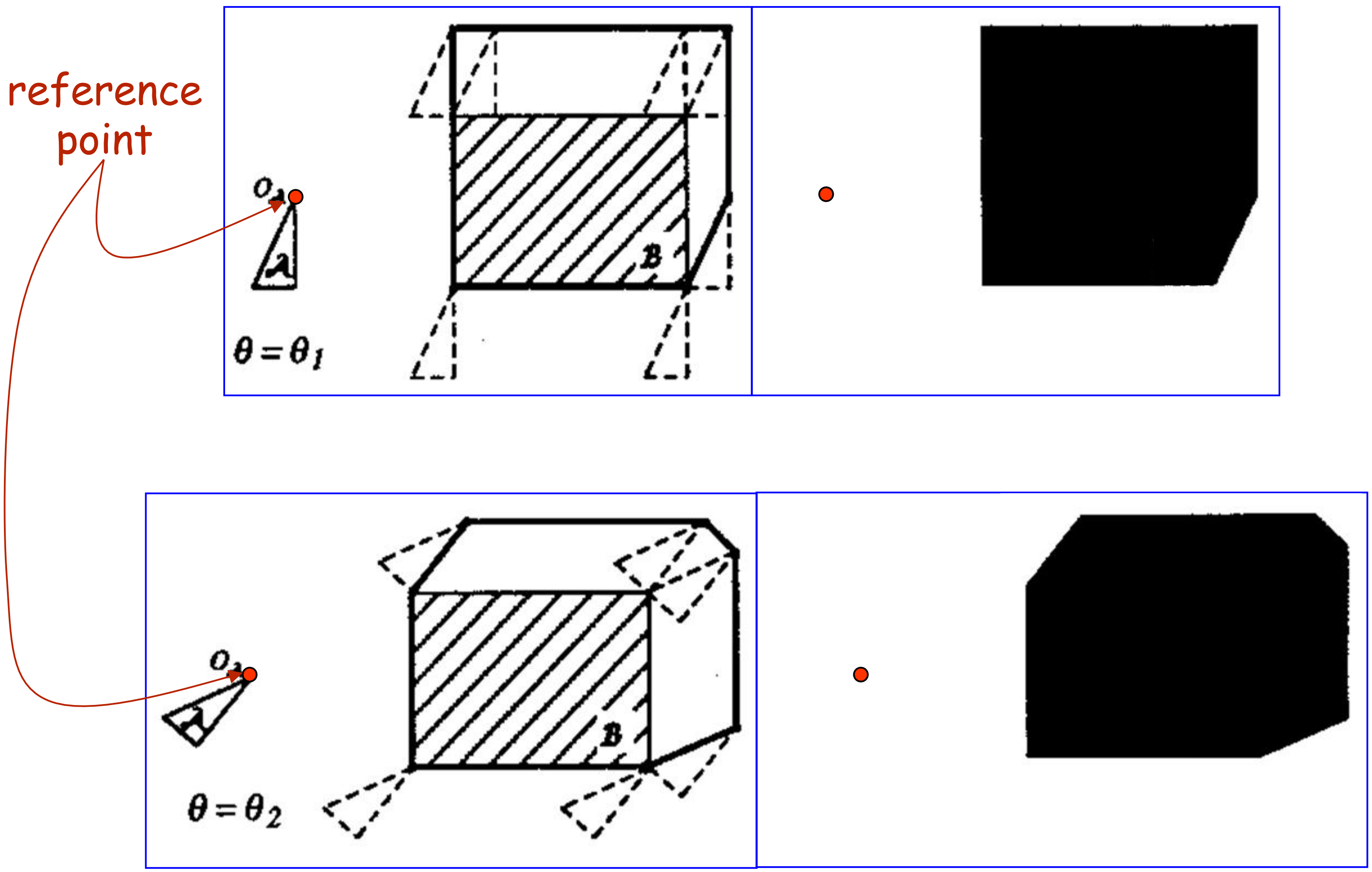


Disc Robot in 2-D Workspace

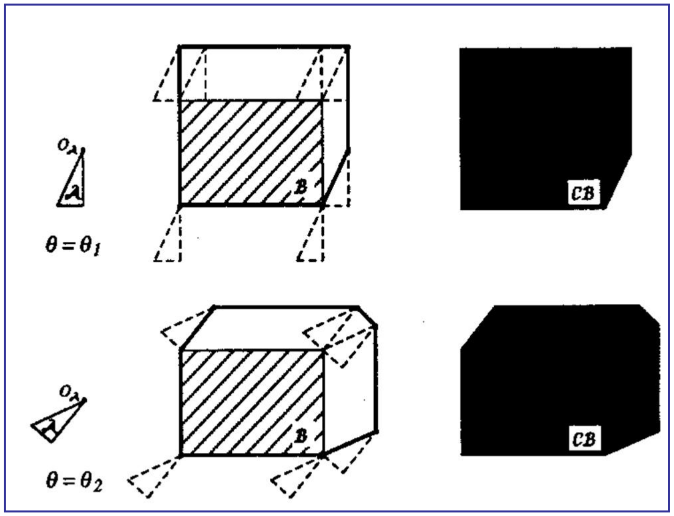


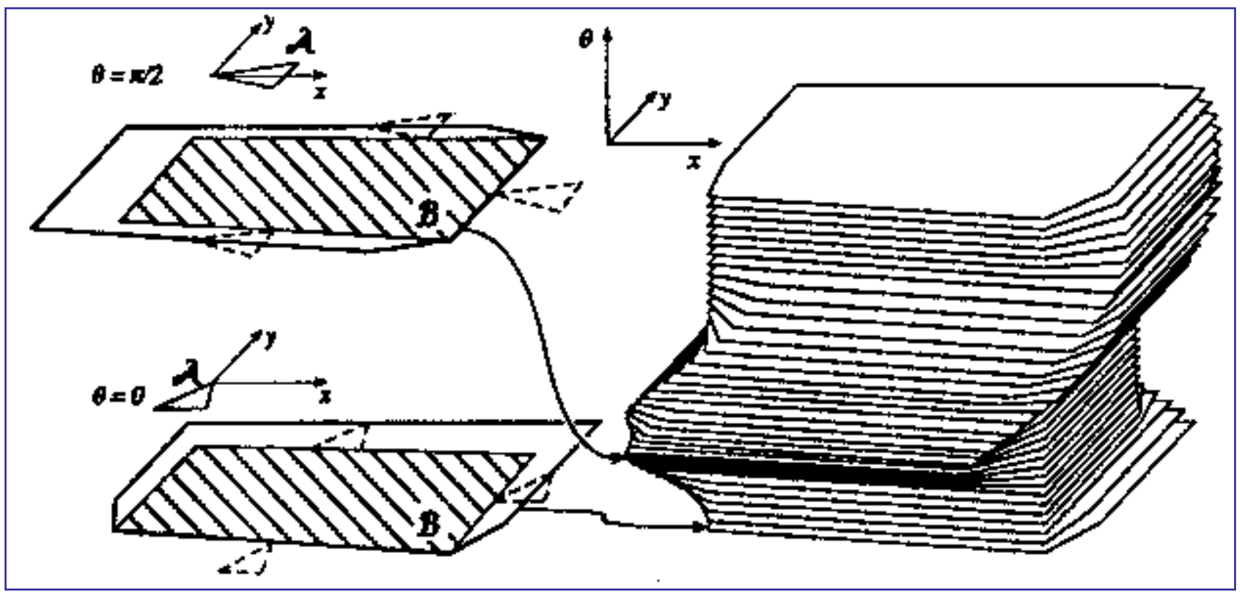
configuration = coordinates (x,y) of robot's center
configuration space C = {(x,y)}
free space F = subset of collision-free configurations

Translating Polygon in 2-D Workspace

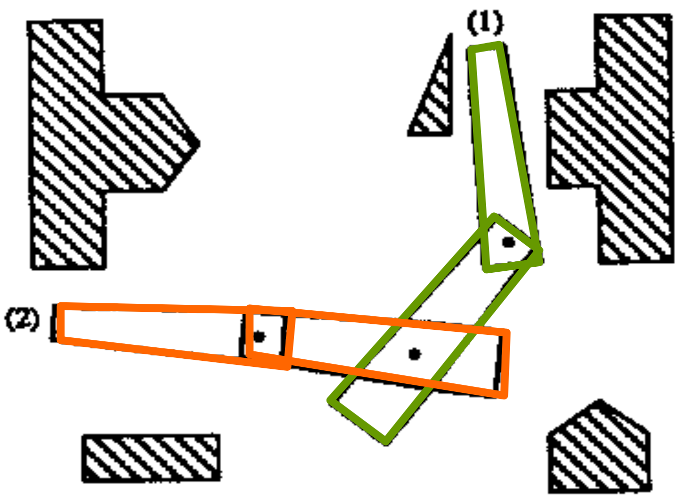


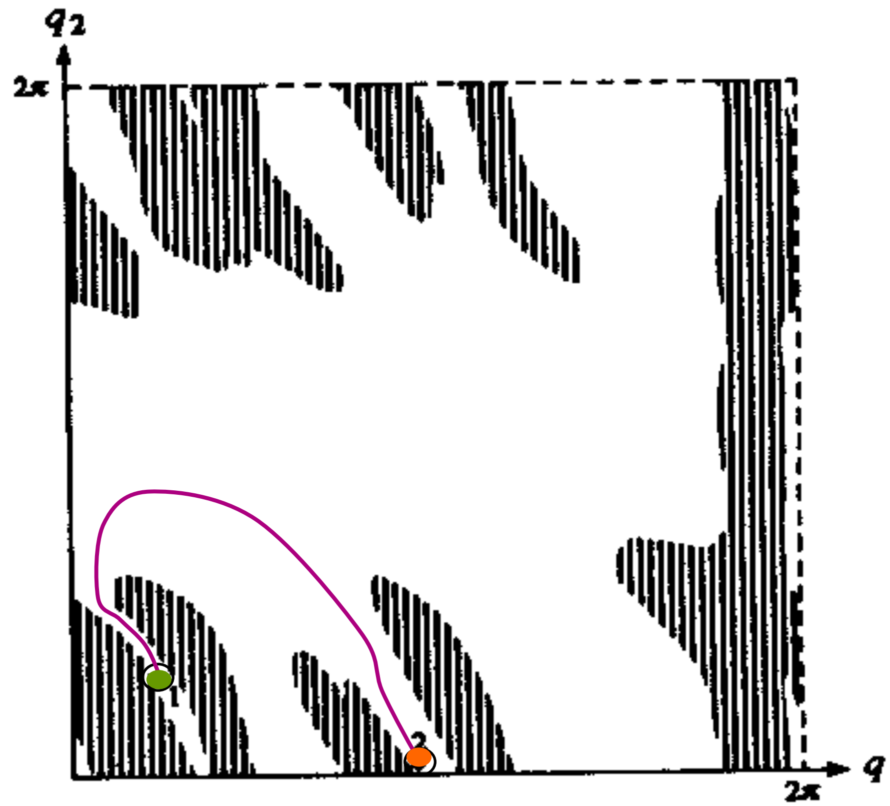
Translating & Rotating Polygon in 2-D Workspace





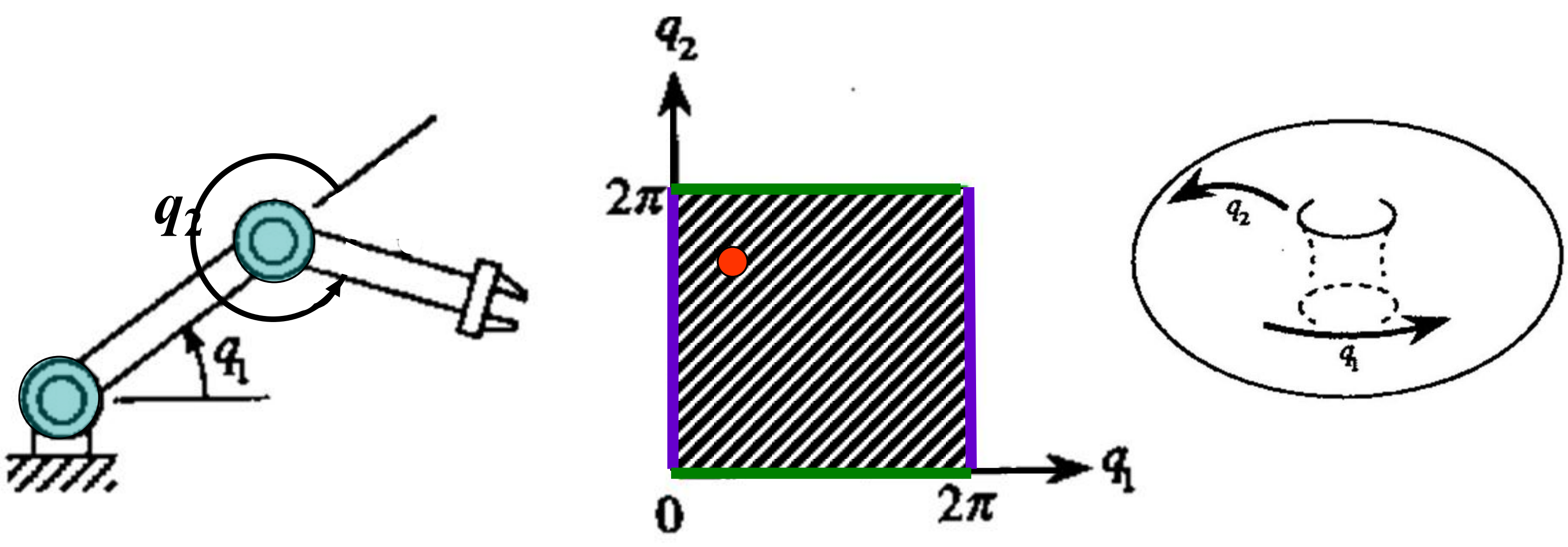
Tool: Configuration Space (C-Space C)



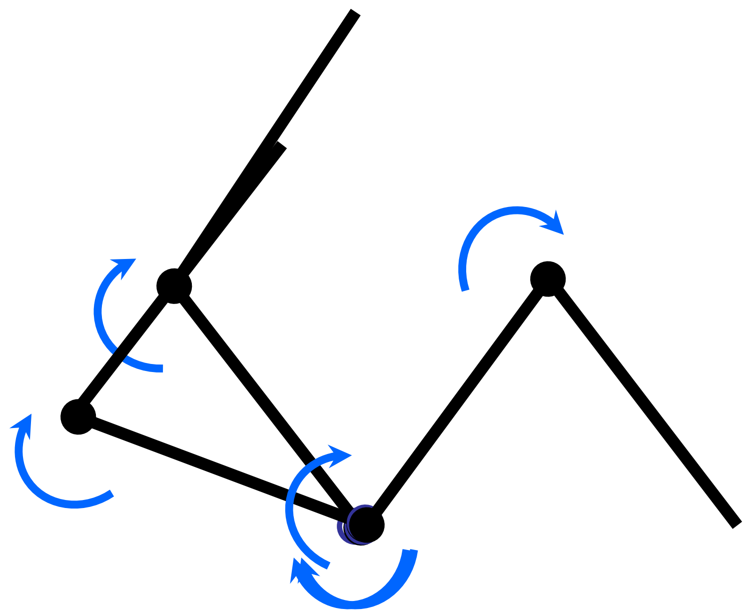


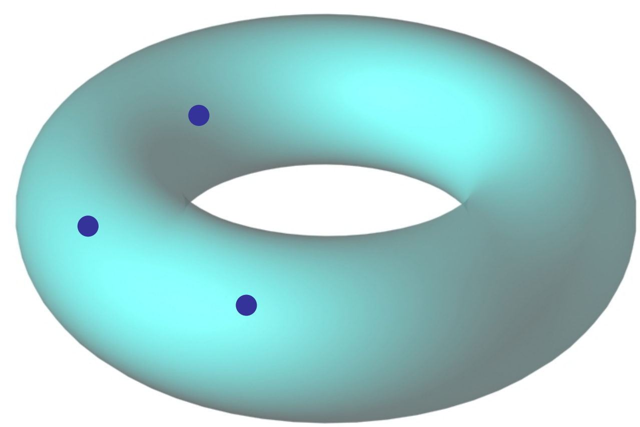
Configuration Space

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

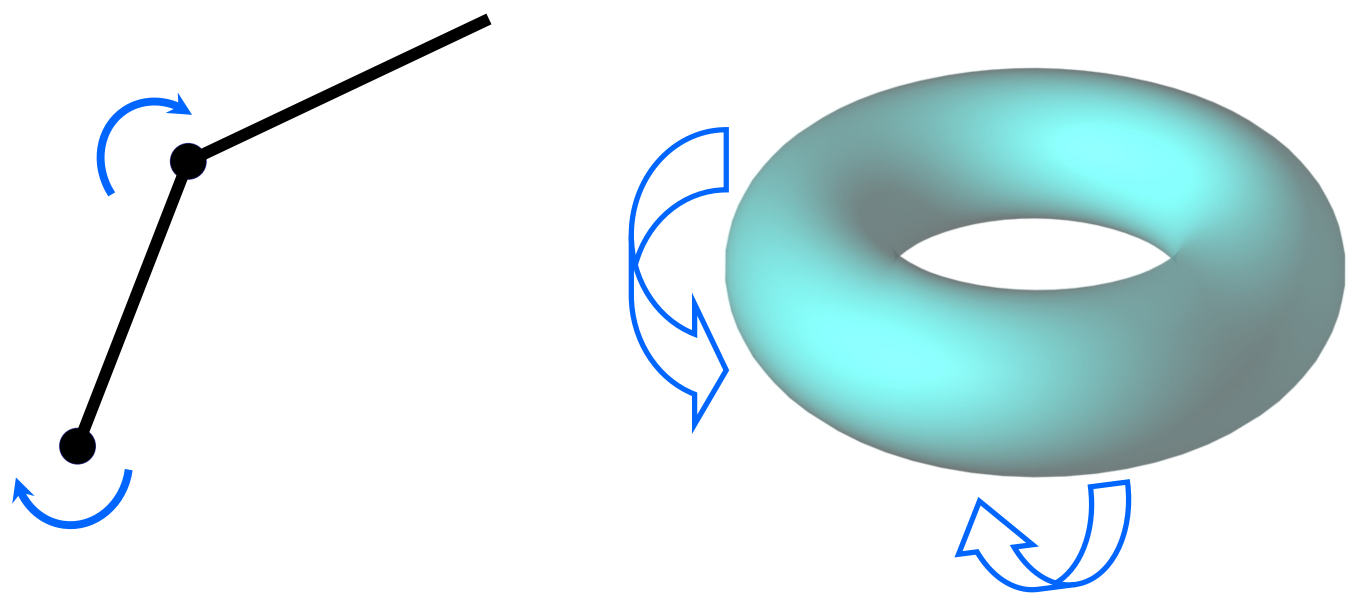


Move

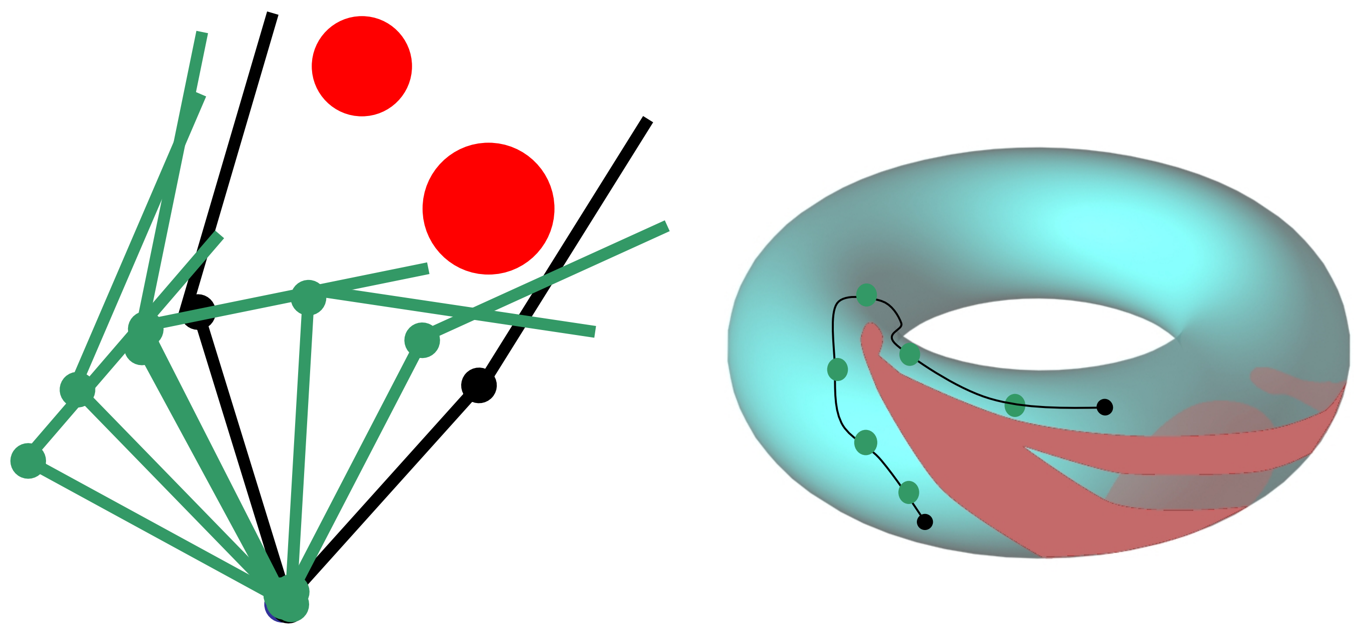




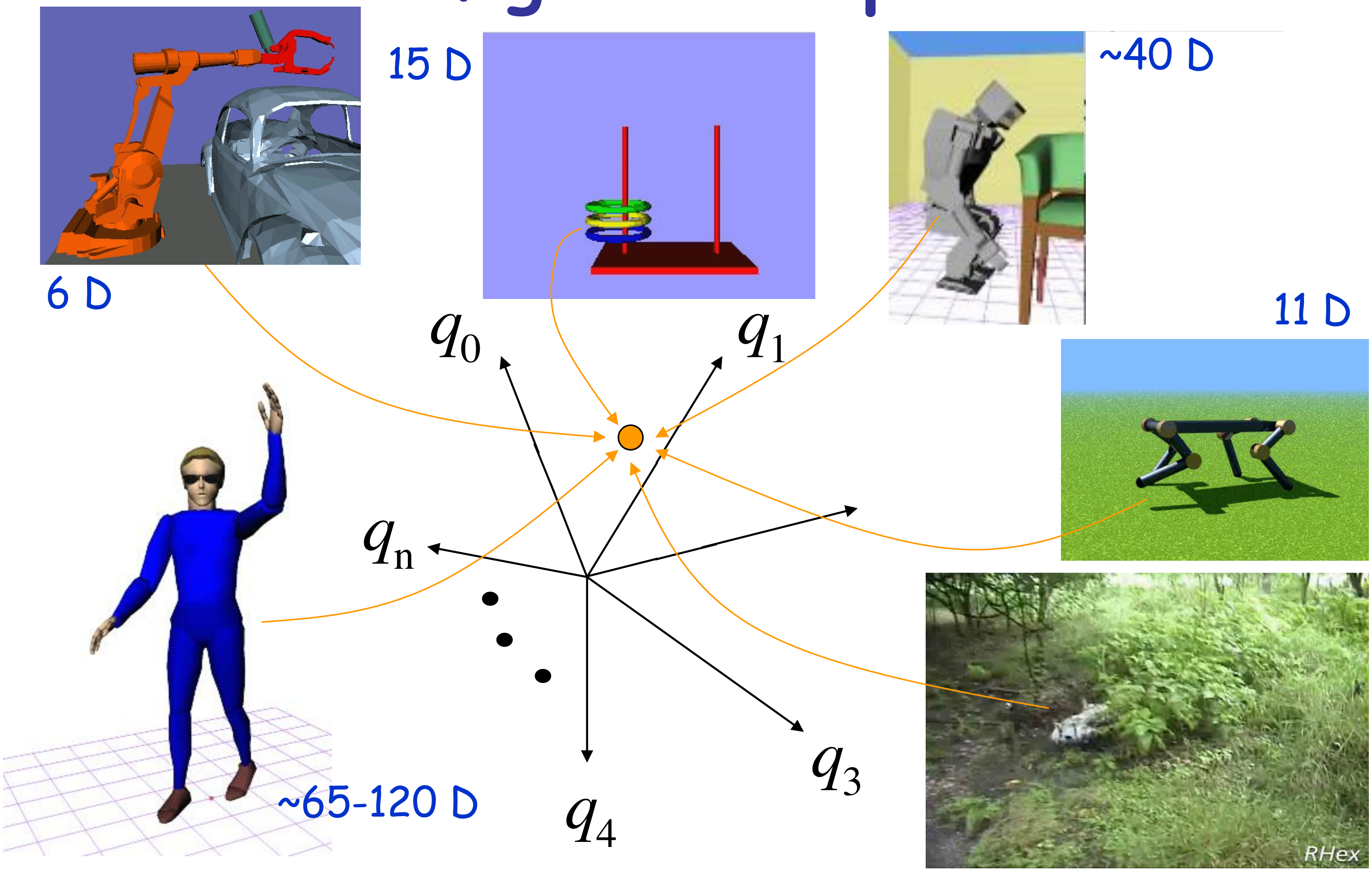
Configuration Space



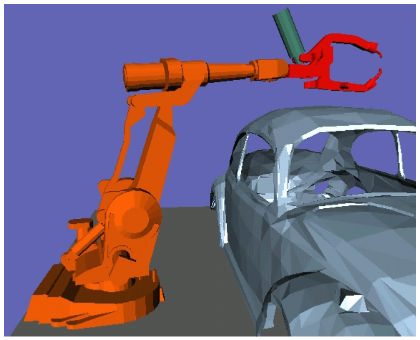
Configuration Space

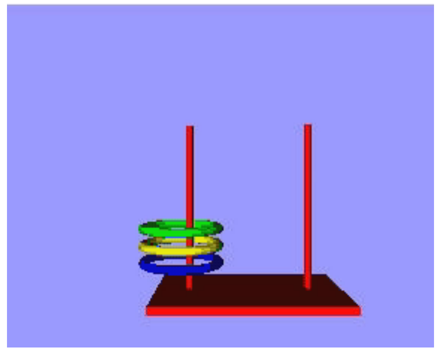


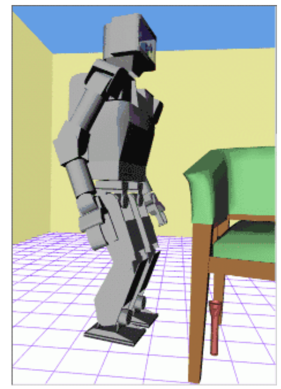
Every robot maps to a point in its configuration space ...

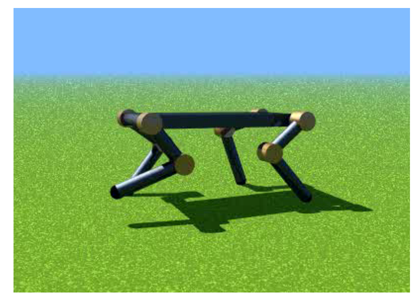


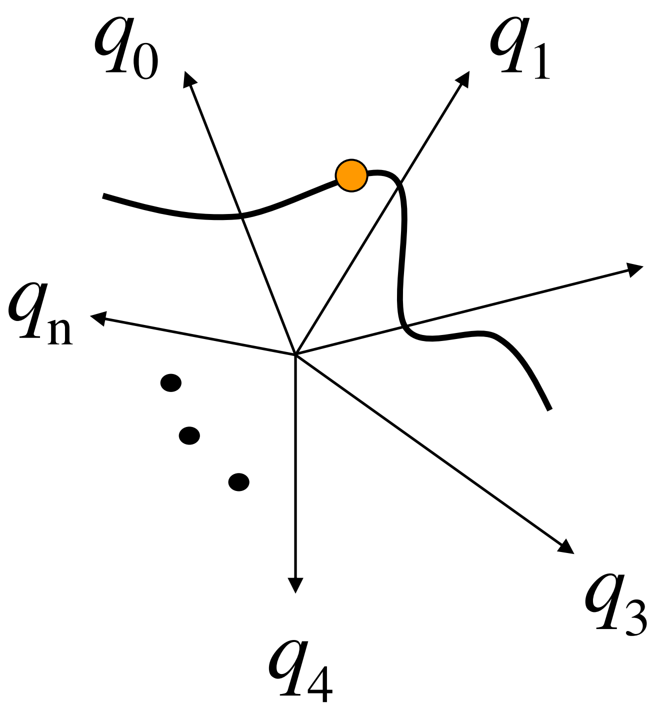
... and every robot path is a curve in configuration space



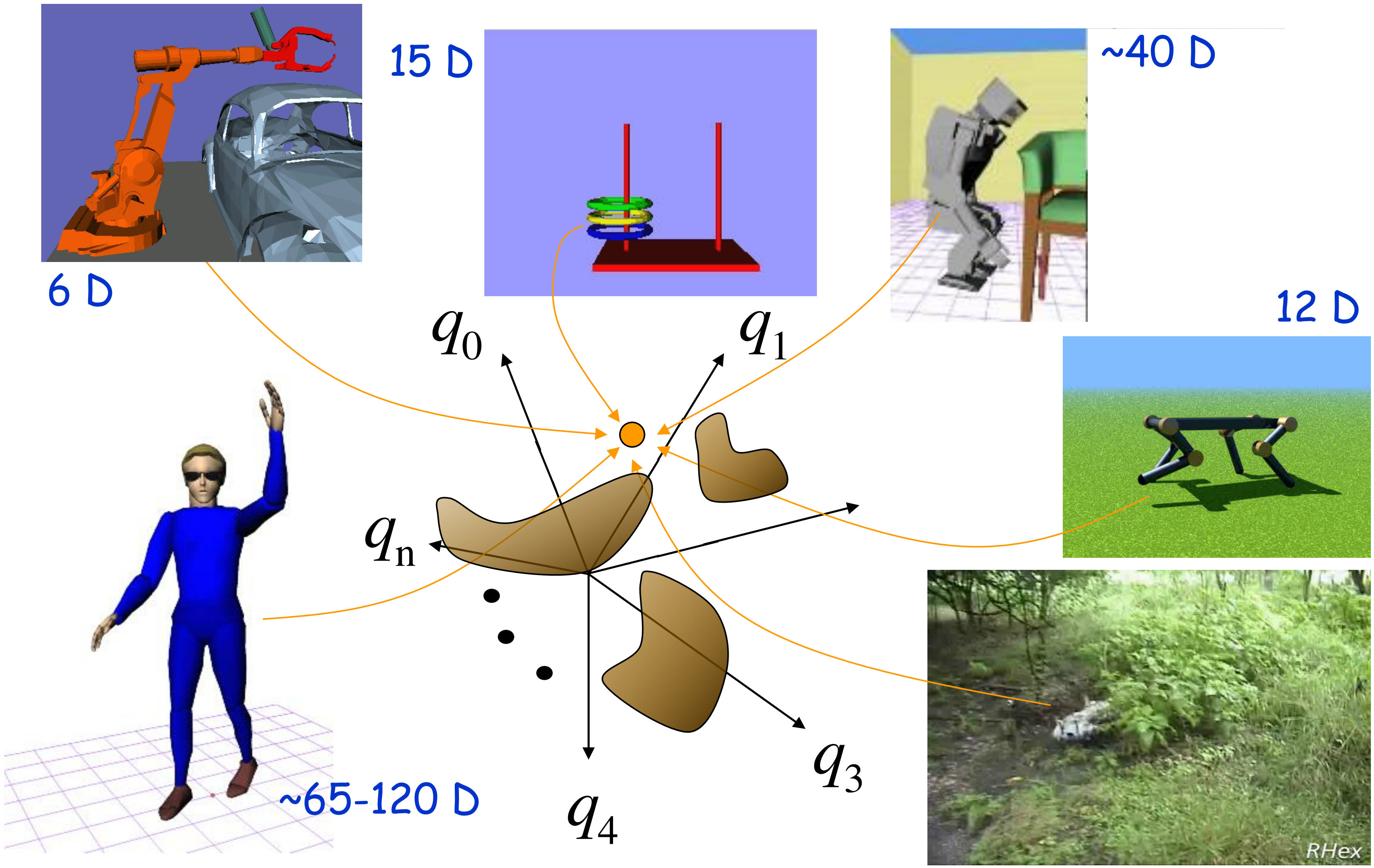








But how do obstacles (and other constraints) map in configuration space?

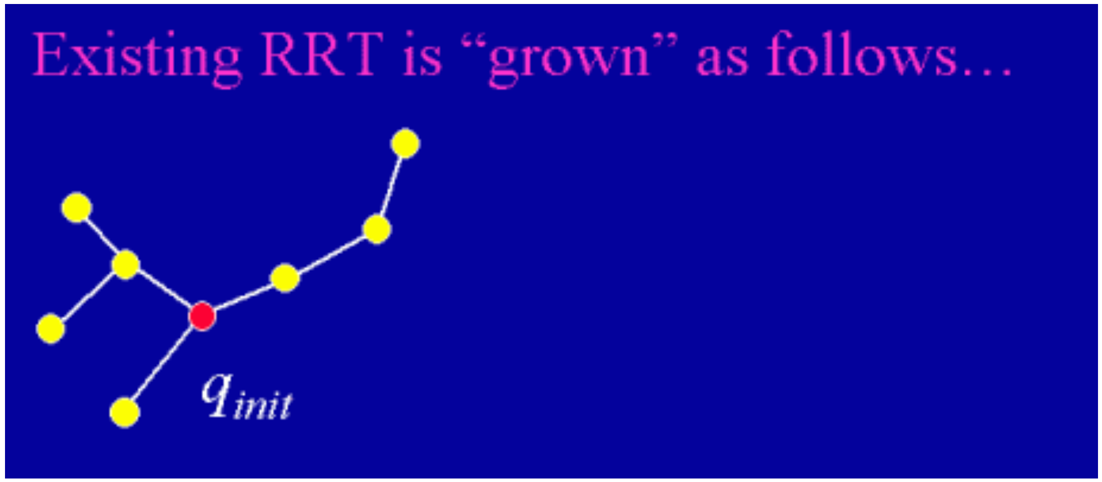


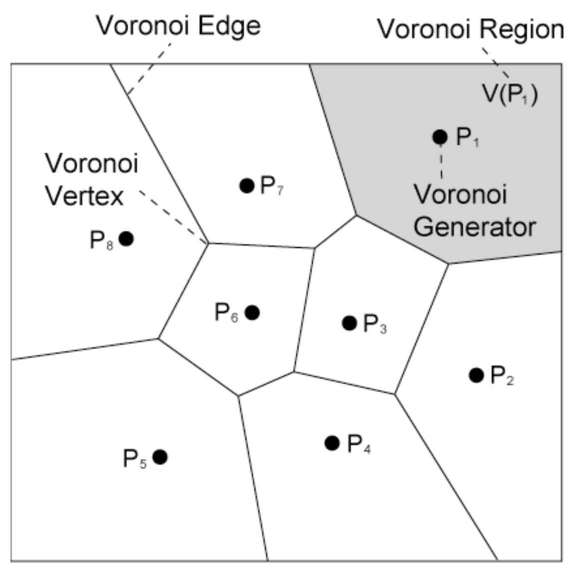
RRTs

[LaVelle, Kuffner, IJRR02]

- RRT is a data structure and algorithm that is designed for efficiently searching no convex high-dimensional spaces.
- RRT can be considered as a Monte-Carlo way of biasing search into largest Voronoi Regions.

Basic construction





RRT Algorithm

BUILD_RRT(x_{init})

$\mathcal{T}$.init(x_{init});
for $k = 1$ **to** K **do**
    $x_{rand} \leftarrow$ RANDOM_STATE();
    EXTEND($\mathcal{T}, x_{rand}$);
Return $\mathcal{T}$

EXTEND($\mathcal{T}, x$)

$x_{near} \leftarrow$ NEAREST_NEIGHBOR($x, \mathcal{T}$);
if NEW_STATE($x, x_{near}, x_{new}, u_{new}$) **then**
    $\mathcal{T}$.add_vertex(x_{new});
    $\mathcal{T}$.add_edge($x_{near}, x_{new}, u_{new}$);
    if $x_{new} = x$ **then**
        Return *Reached*;
    else
        Return *Advanced*;
Return *Trapped*;

Sampling-based Algorithms for Optimal Motion Planning

RRT*-Map Specs

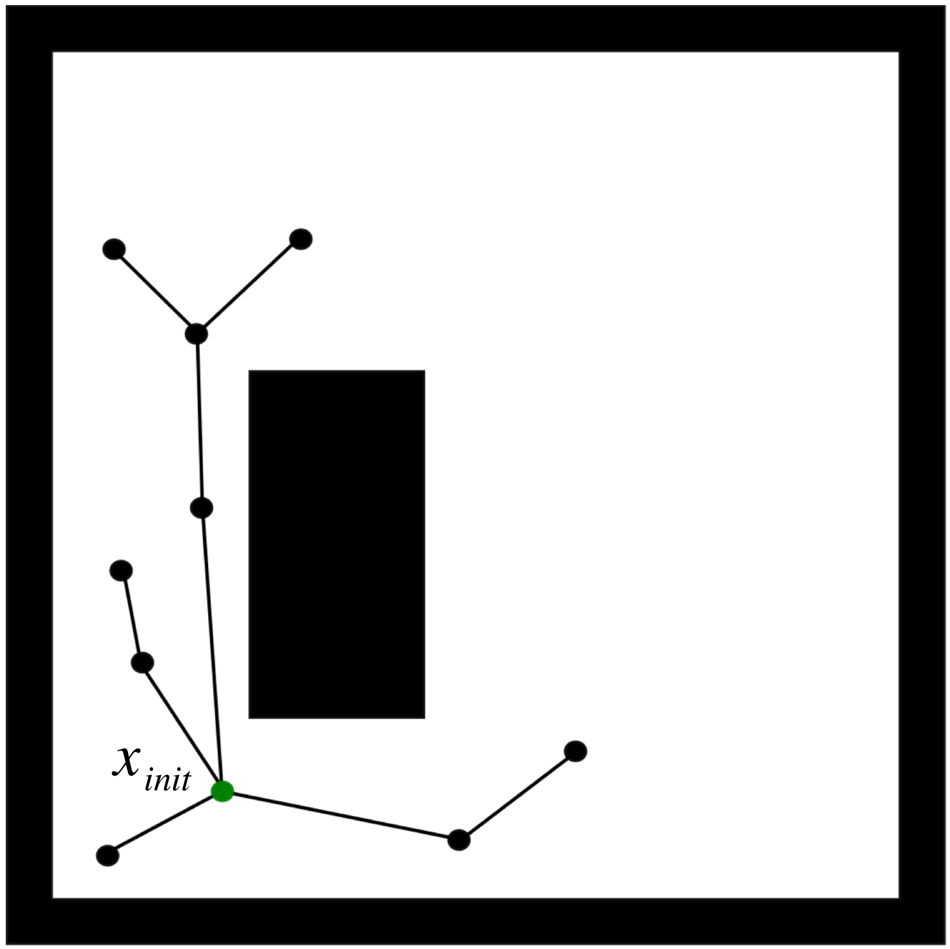
$$\mu\left(X_{free}\right) = 0.92$$
$$d = 2$$
$$\zeta_d = \pi$$

$$\gamma_{RRT} > \left(2(1+1/d)\right)^{1/d}\left(\frac{\mu\left(X_{free}\right)}{\zeta_d}\right)^{1/d} \approx 0.9373$$

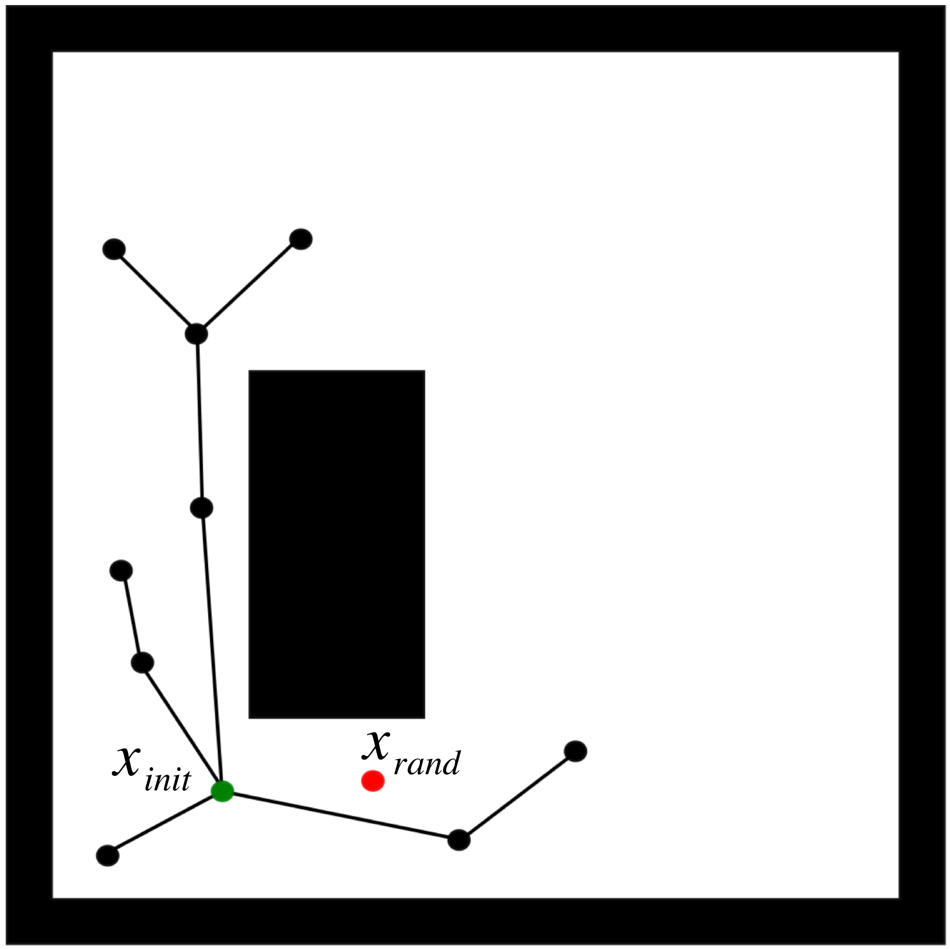
$$r_n = \gamma_{RRT}\left(\frac{\log(n)}{n}\right)^{1/d}$$

$$n = 10$$
$$r_{10} > 0.4497$$

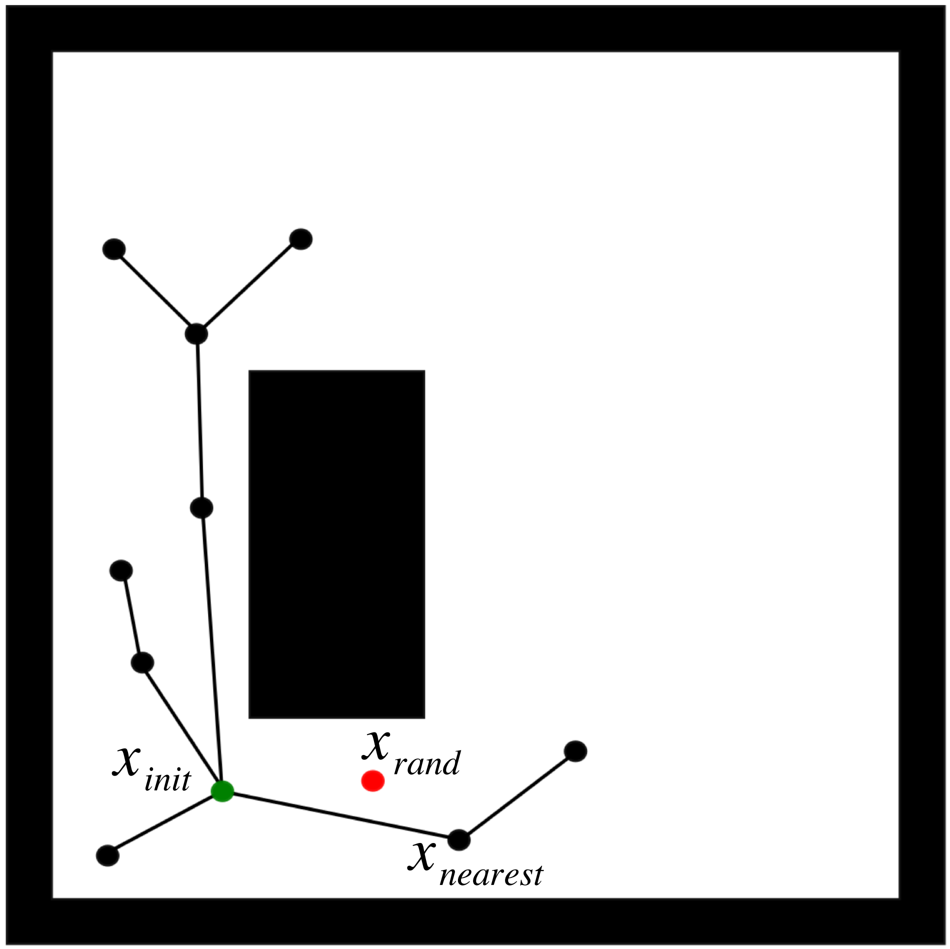
RRT*-Tree after iteration 9



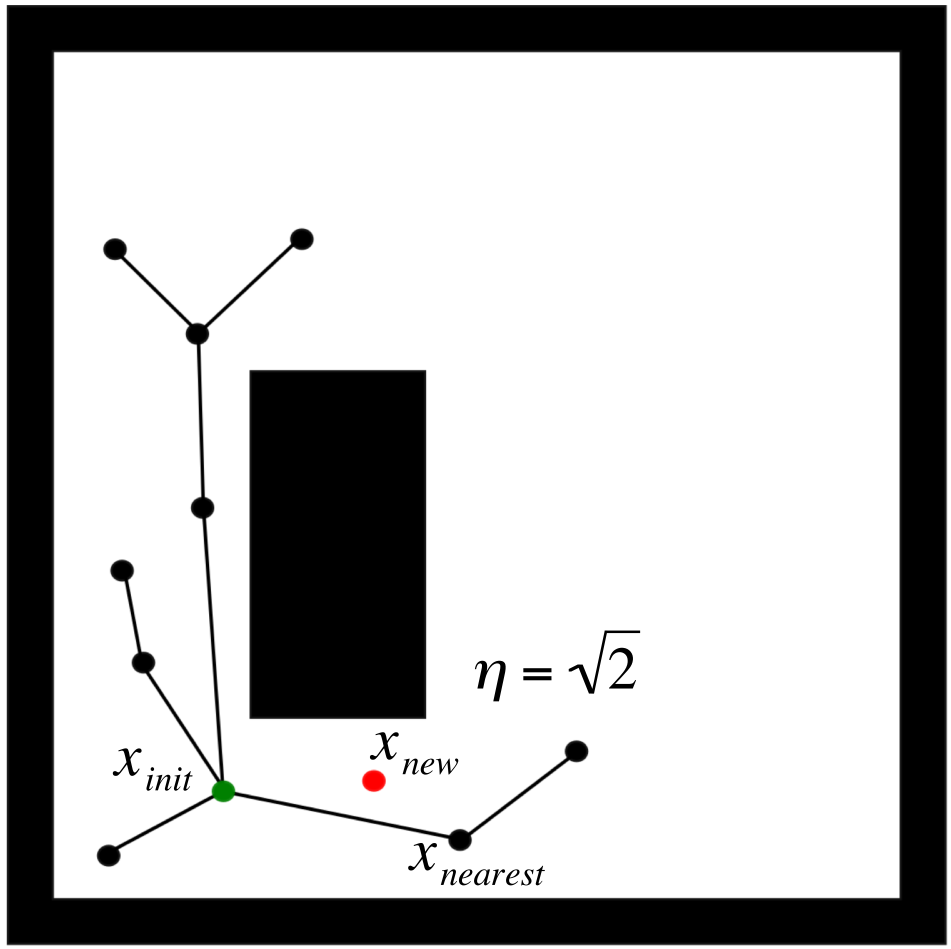
RRT*-New Sample



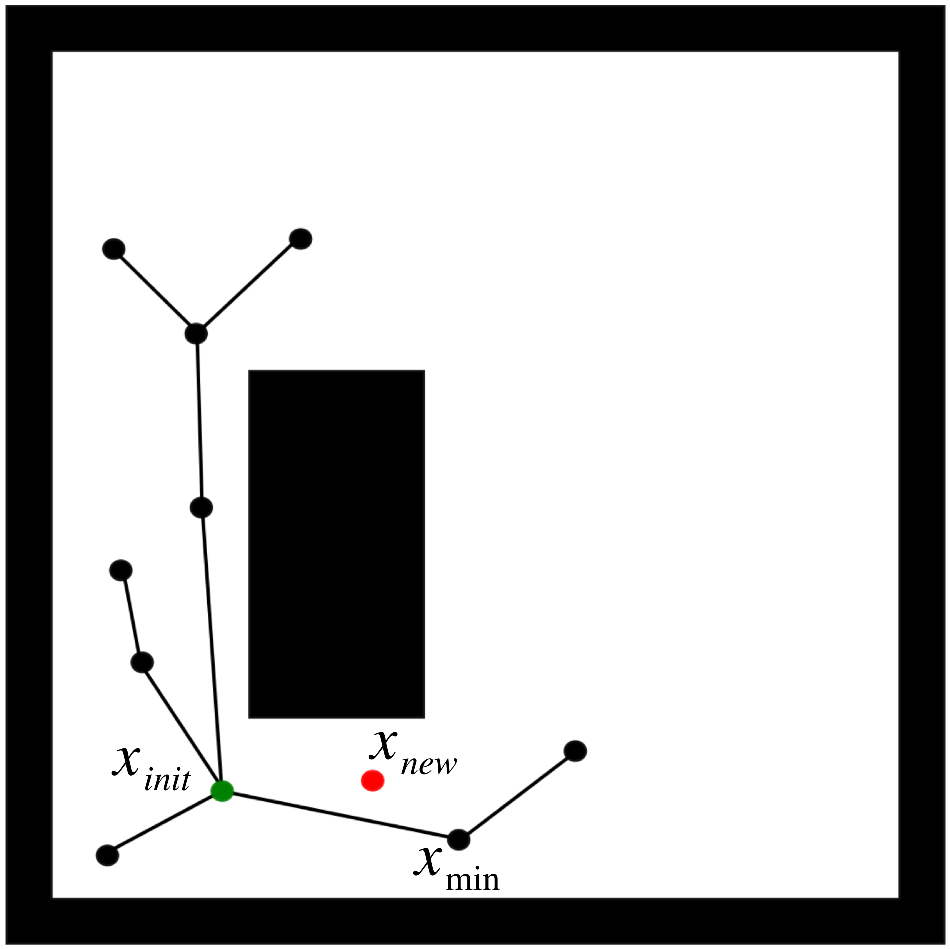
RRT*-New Sample



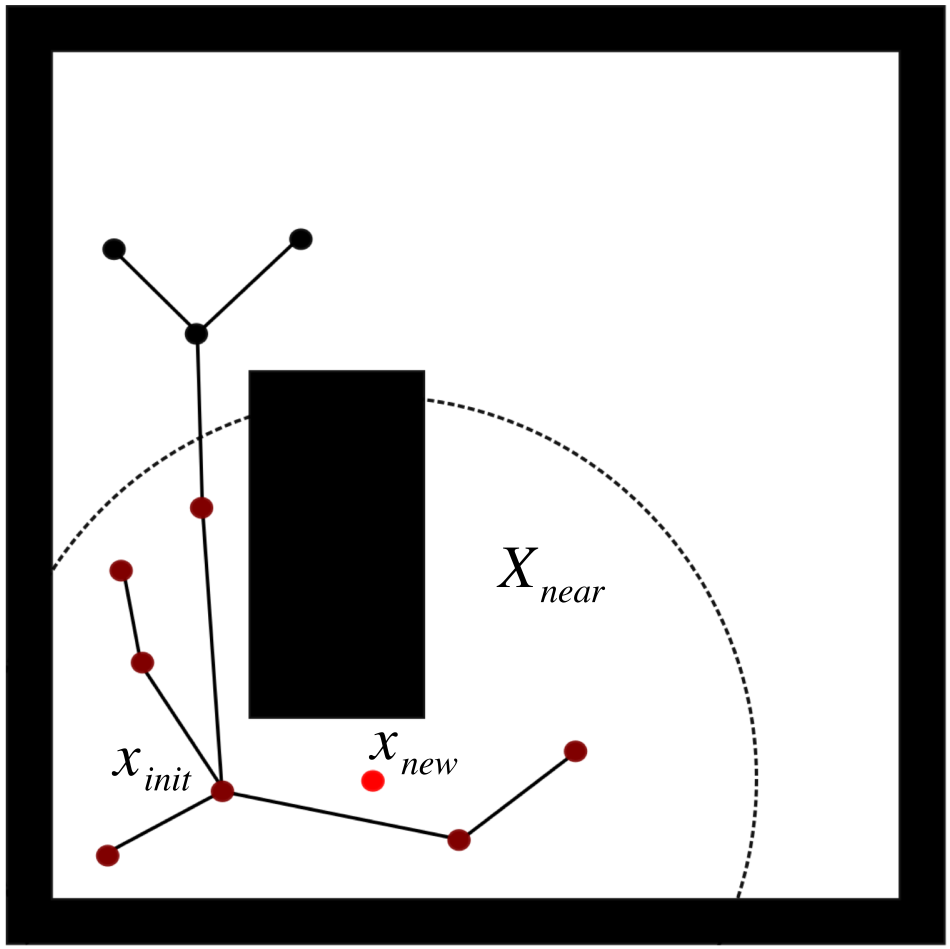
RRT*-New Sample



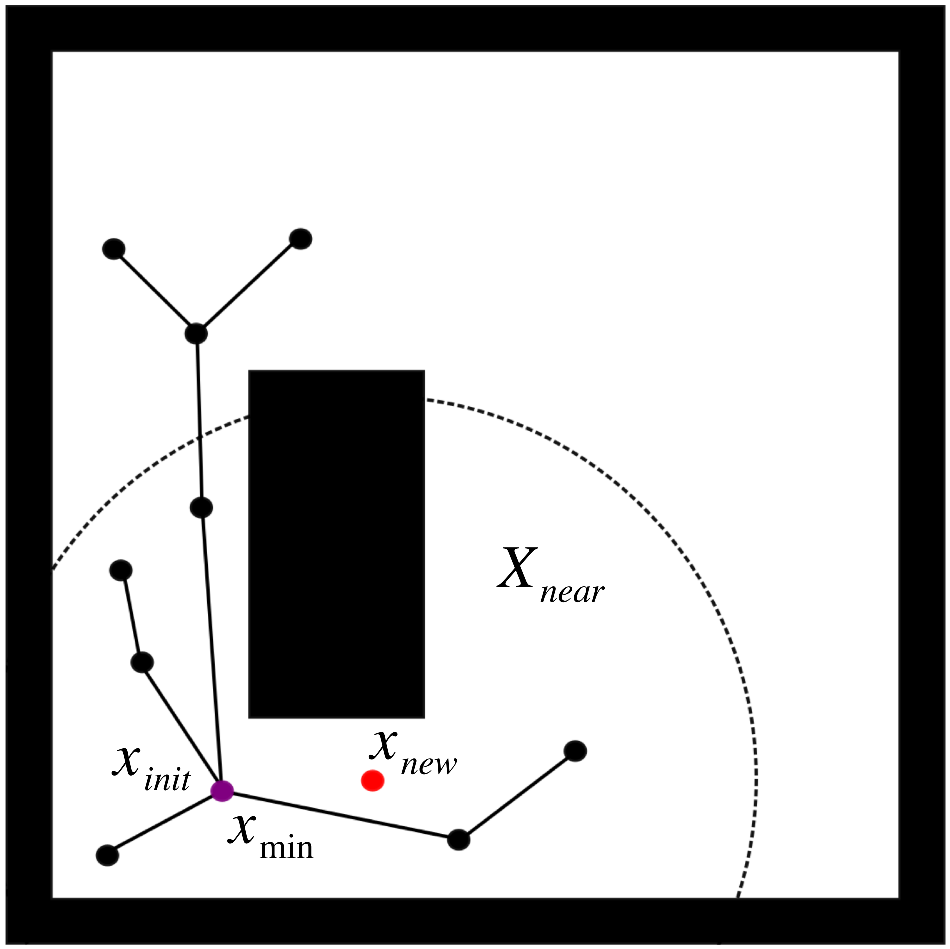
RRT*-New Sample



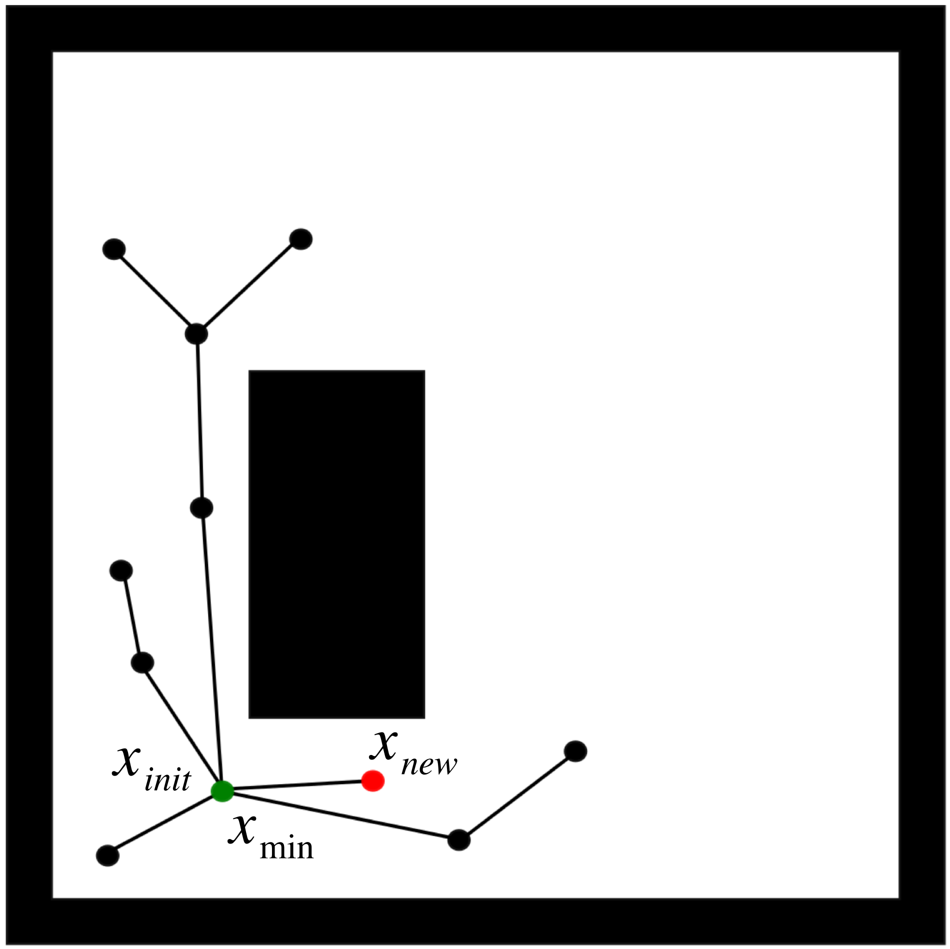
RRT*-Connect Min Cost Path



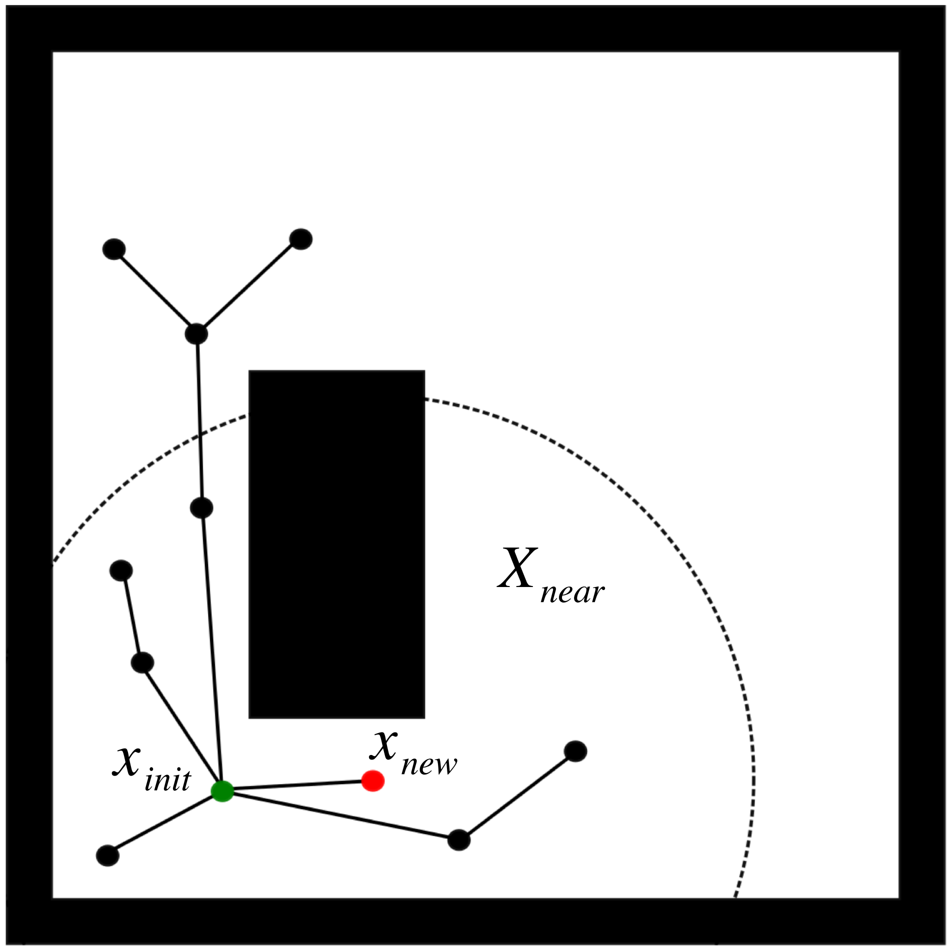
RRT*-Connect Min Cost Path



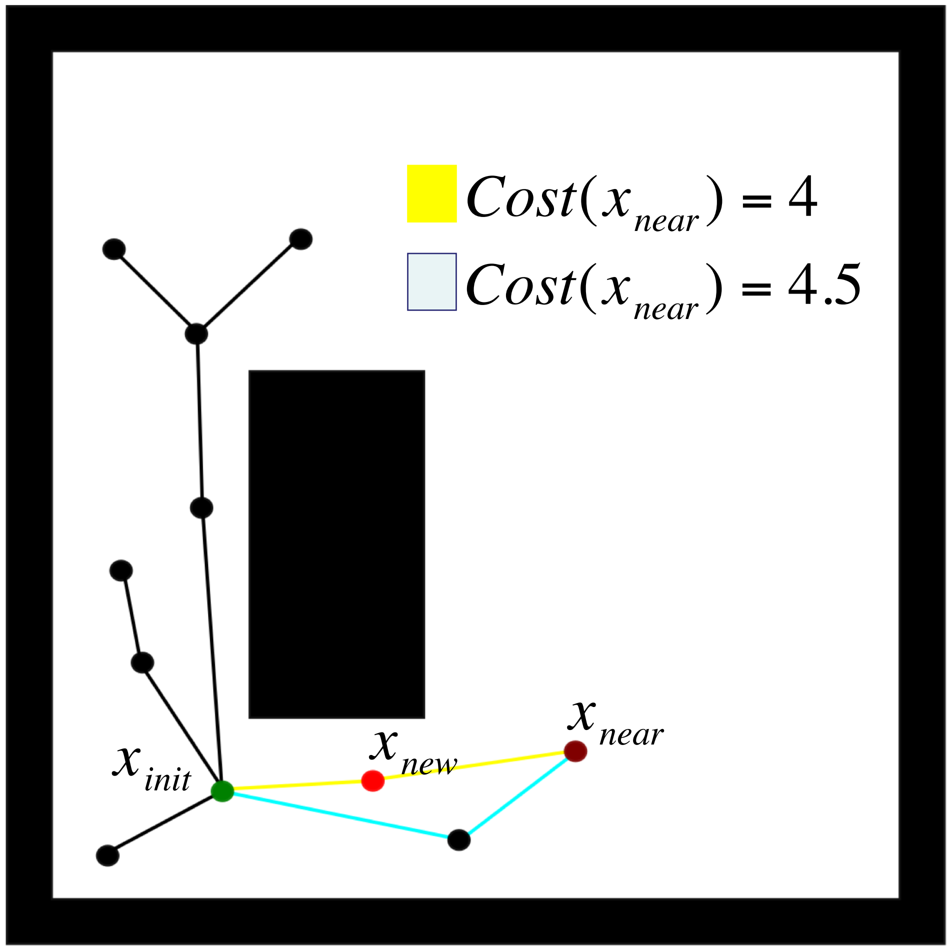
RRT*-Connect Min Cost Path



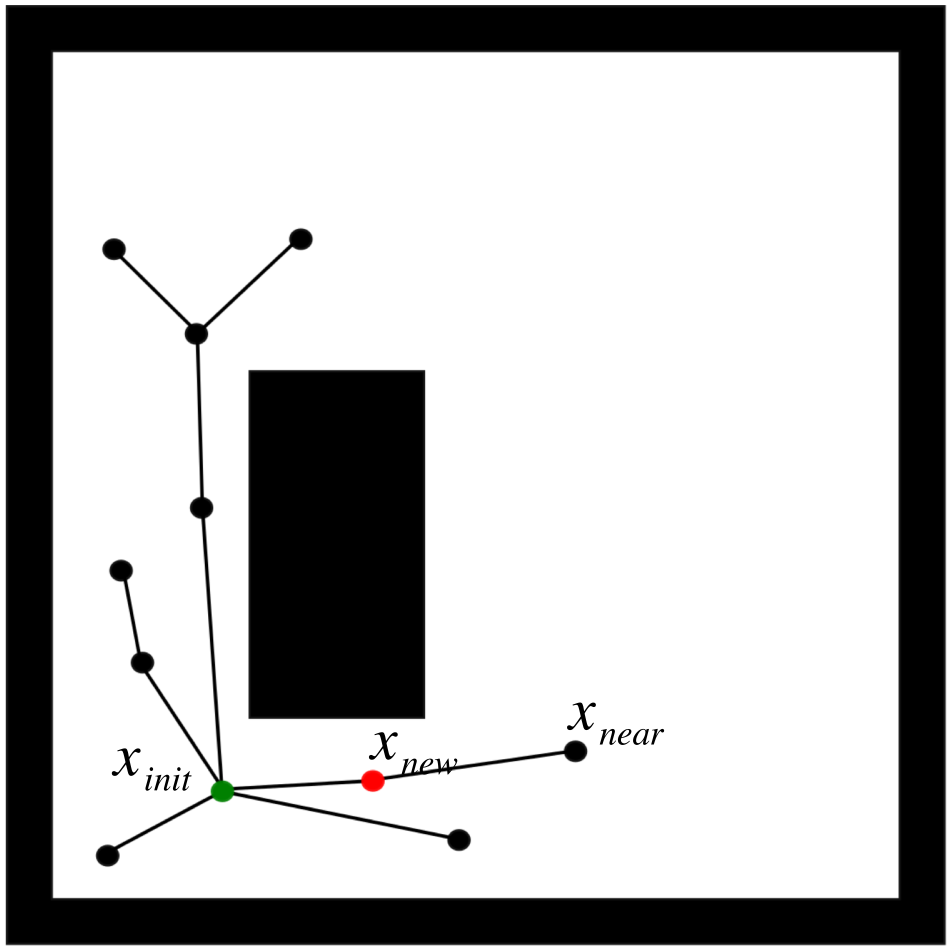
RRT*-Rewire



RRT*-Rewire



RRT*-Rewire



RRT algorithm

Algorithm 3: RRT

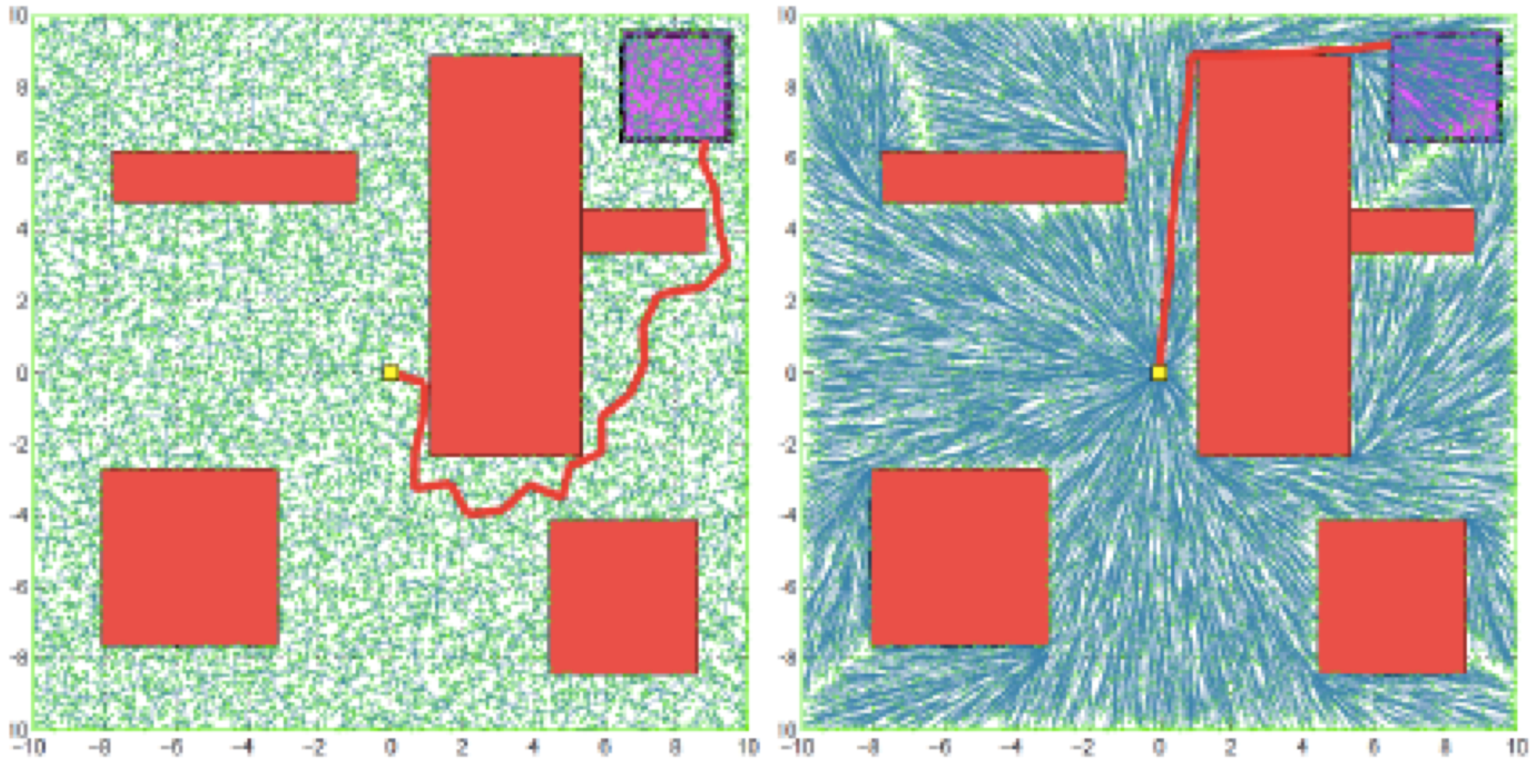
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
    $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
    $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
    $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
    if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
        $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$;
return $G = (V, E)$;

RRT* algorithm

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\text{min}} \leftarrow x_{\text{nearest}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ **then**
        $x_{\text{min}} \leftarrow x_{\text{near}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$
      then $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

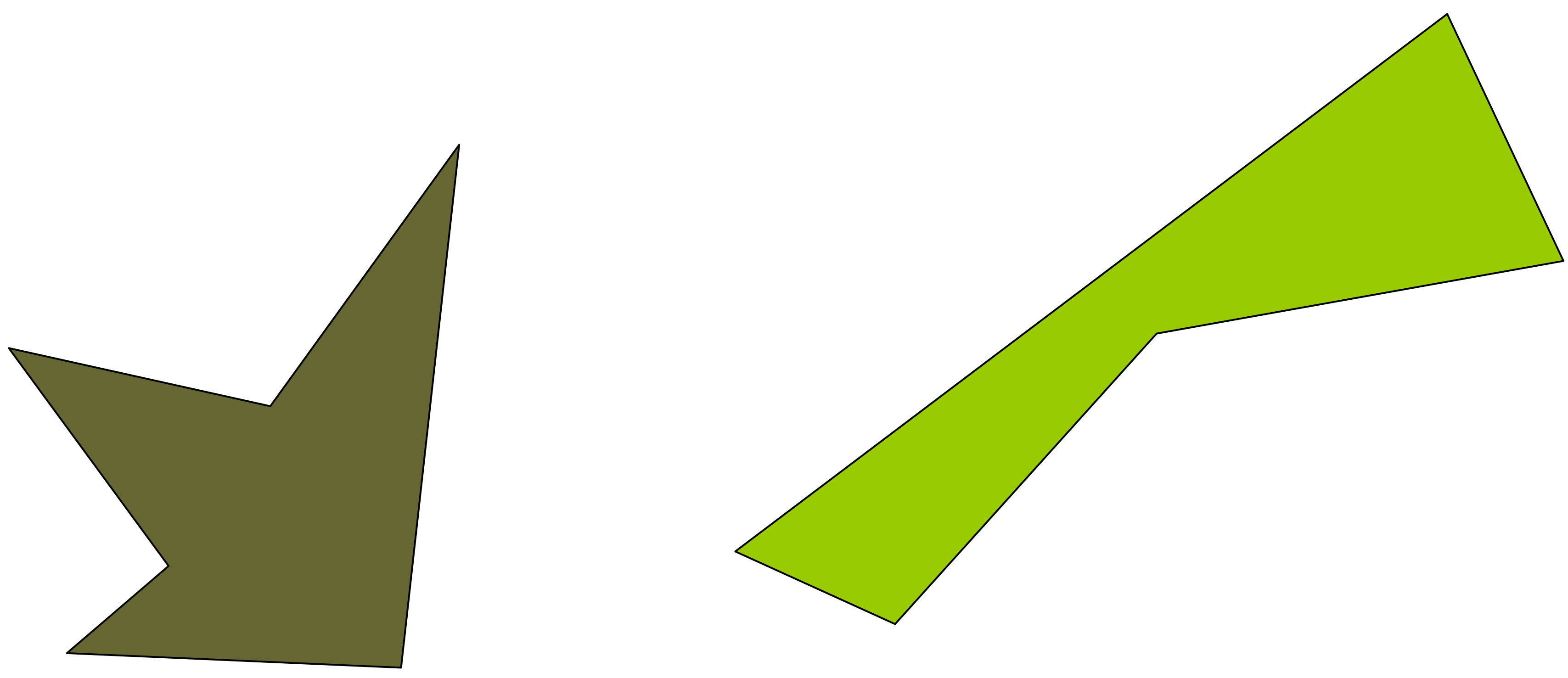
Results of RRT vs RRT*



(a) (b)

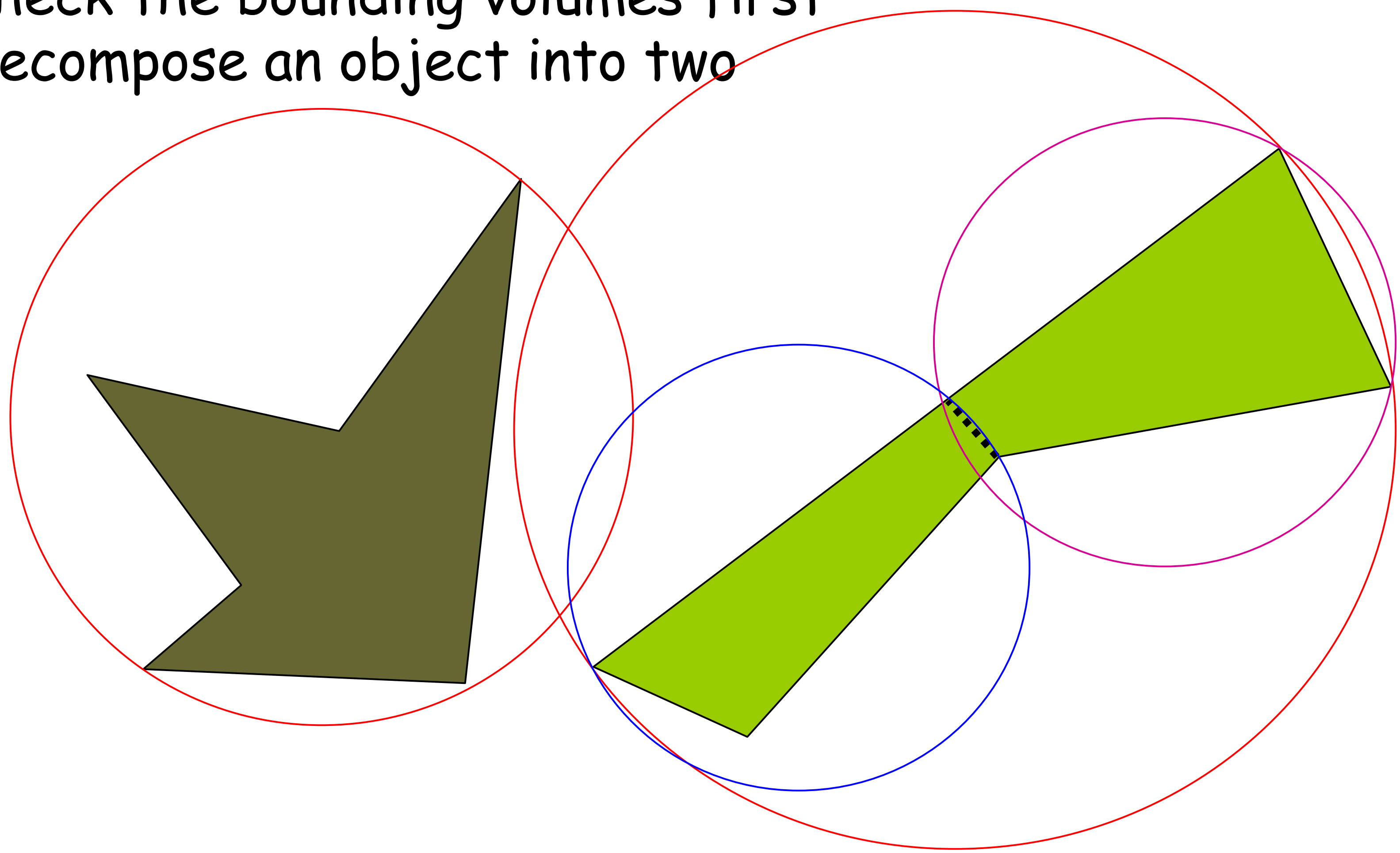
Collision Checking

- Check if objects overlap



Hierarchical Collision Checking

- Enclose objects into bounding volumes (spheres or boxes)
- Check the bounding volumes first
- Decompose an object into two



BVH of a 3D Triangulated Cat

Bibliography

- Robot motion planning, Kluwer,
  Jean-Claude Latombe, 1991.
- Robot motion planning and control, Springer, Jean-Paul Laumond, 1998.
- Planning algorithms, Cambridge, Steven M. LaValle, 2006.
- Sampling-based algorithms for optimal motion planning, Sertac Karaman and Emilio Frazzoli, Int. J. Robotics Research, 2011.