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

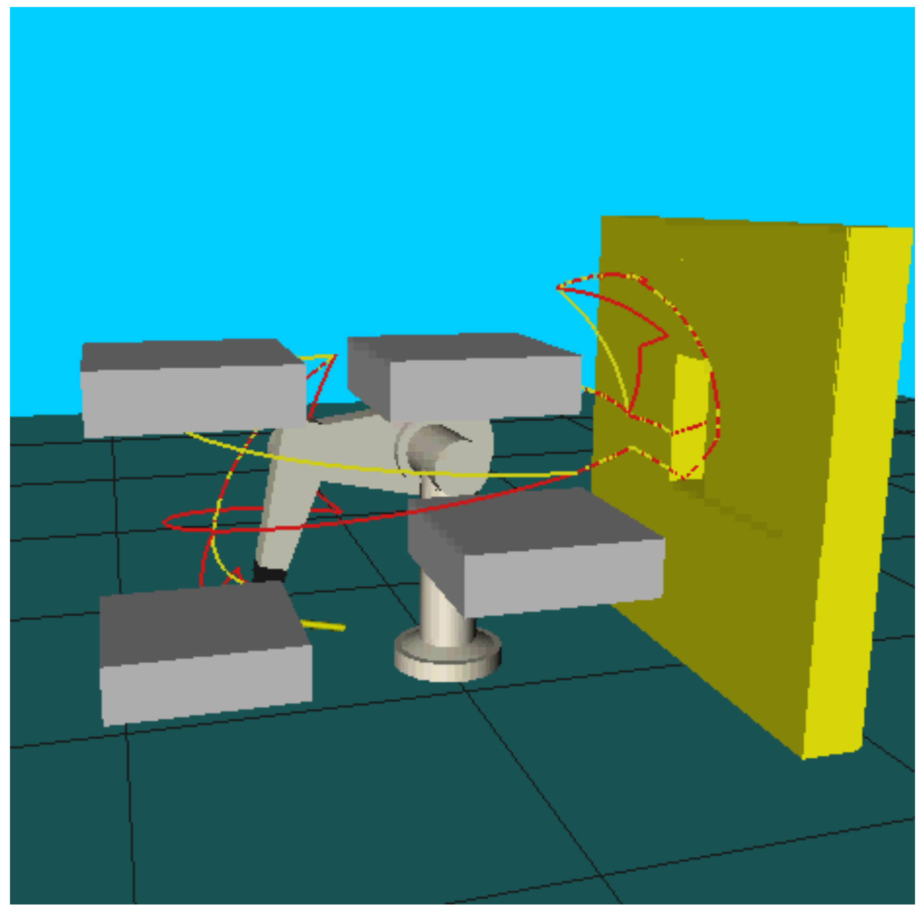
- Compute **motion strategies**, e.g.:
  - geometric paths
  - time-parameterized trajectories
  - sequence of sensor-based motion commands
- To achieve **high-level goals**, e.g.:
  - go to A without colliding with obstacles
  - assemble product P
  - build map of environment E
  - find object O

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

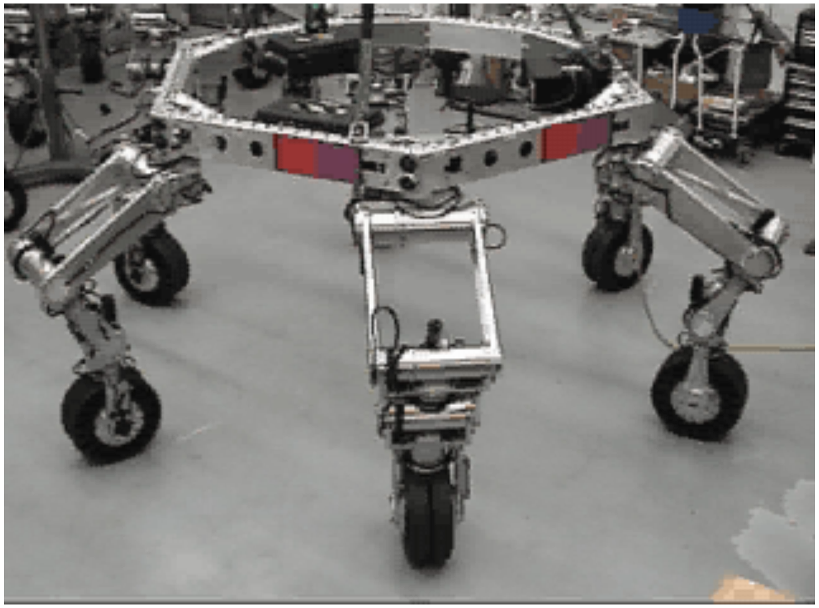


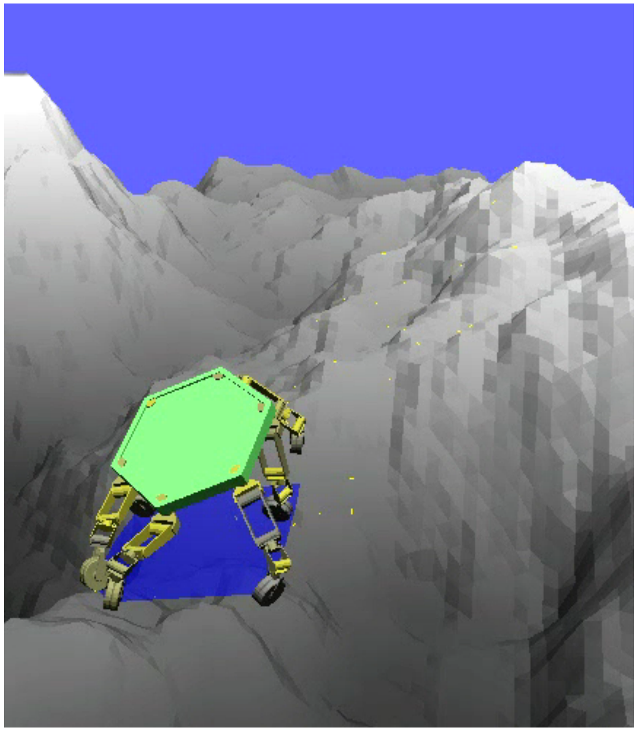
Extensions of Basic Problem

- 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

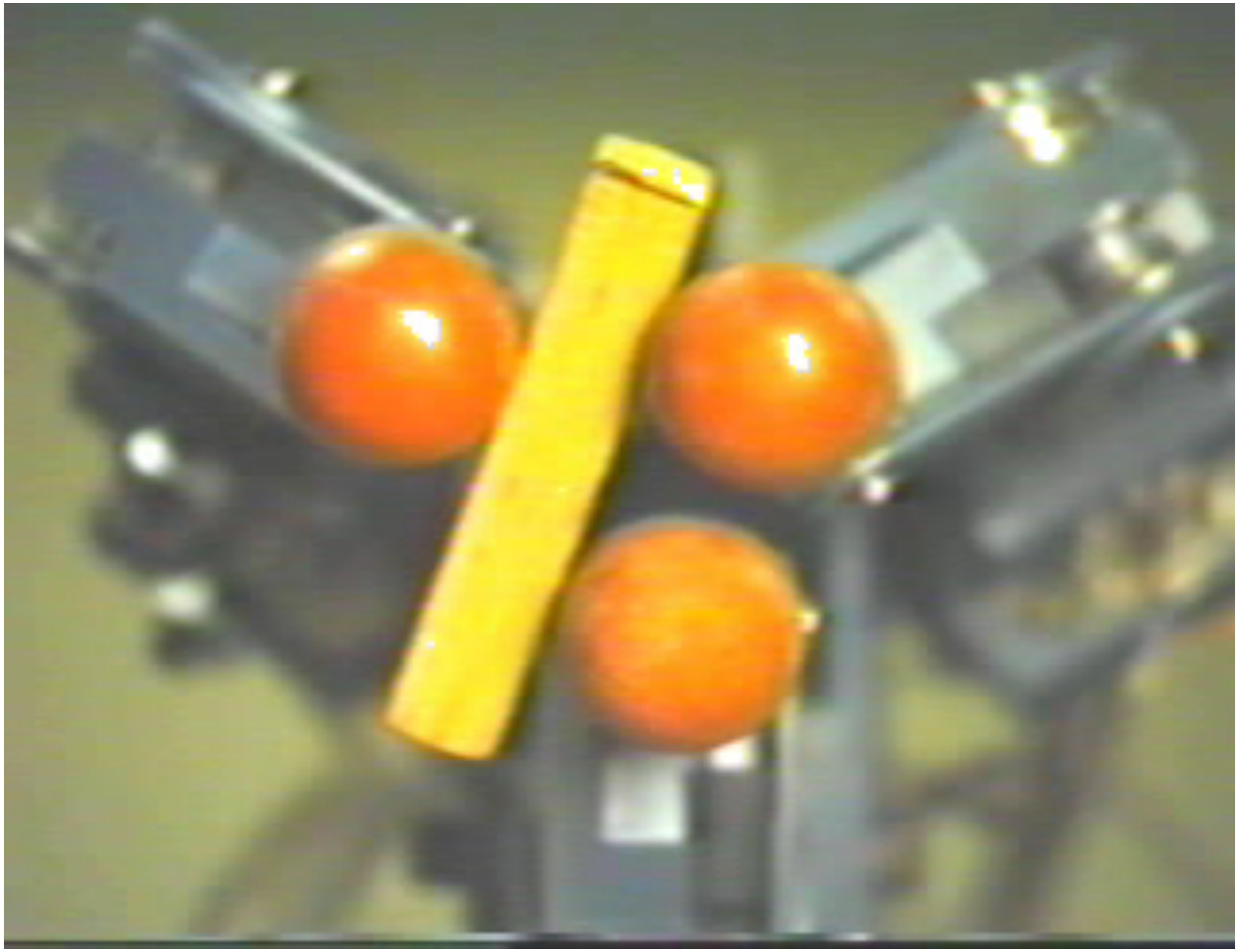
Some Applications

Lunar Vehicle (ATHLETE, NASA/JPL)

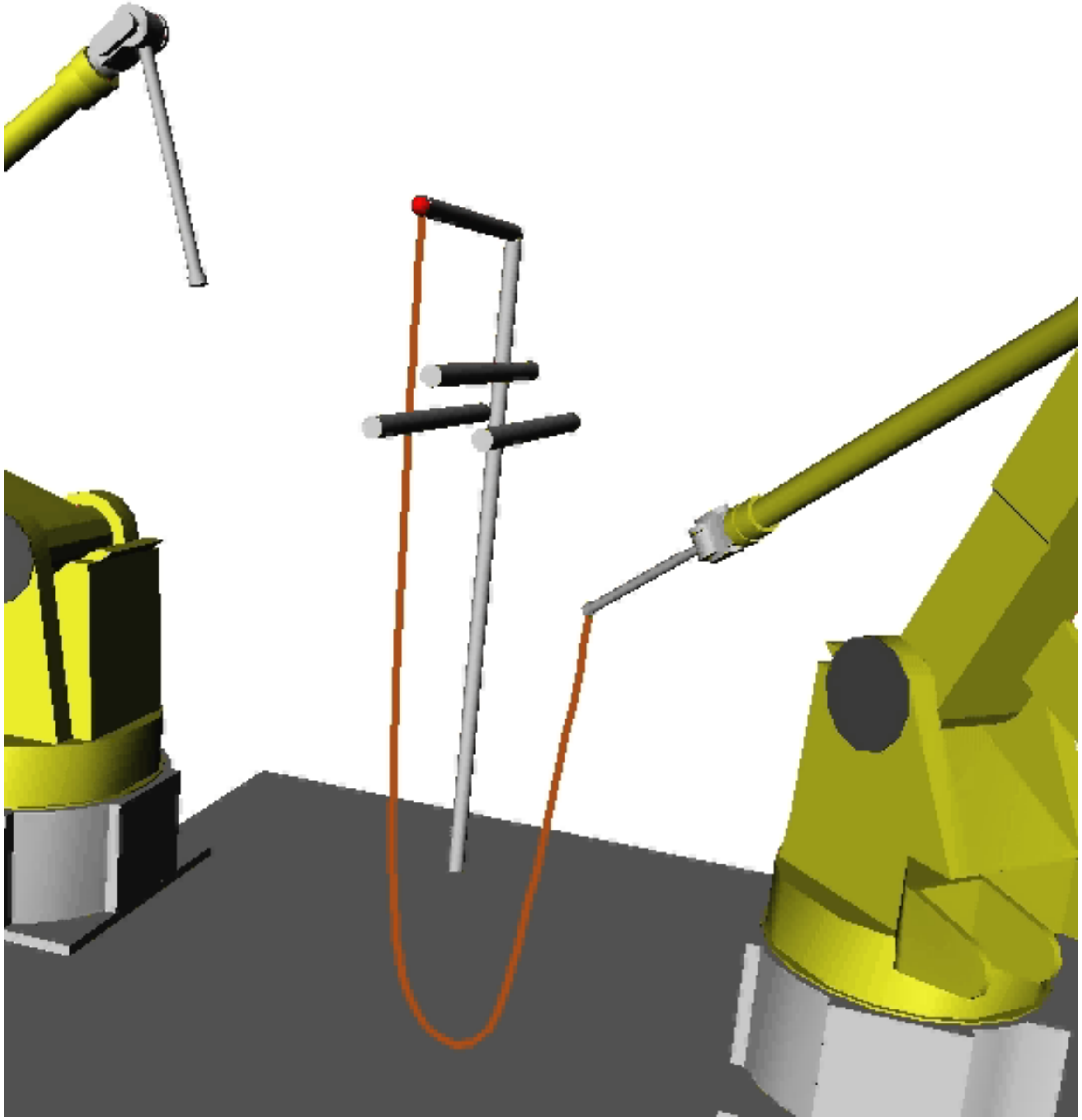




Dexterous Manipulation

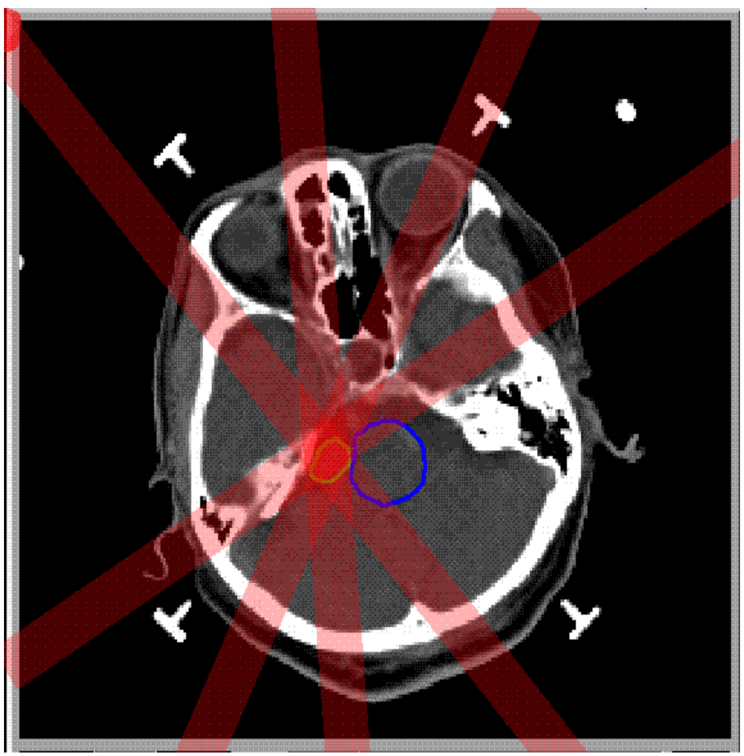


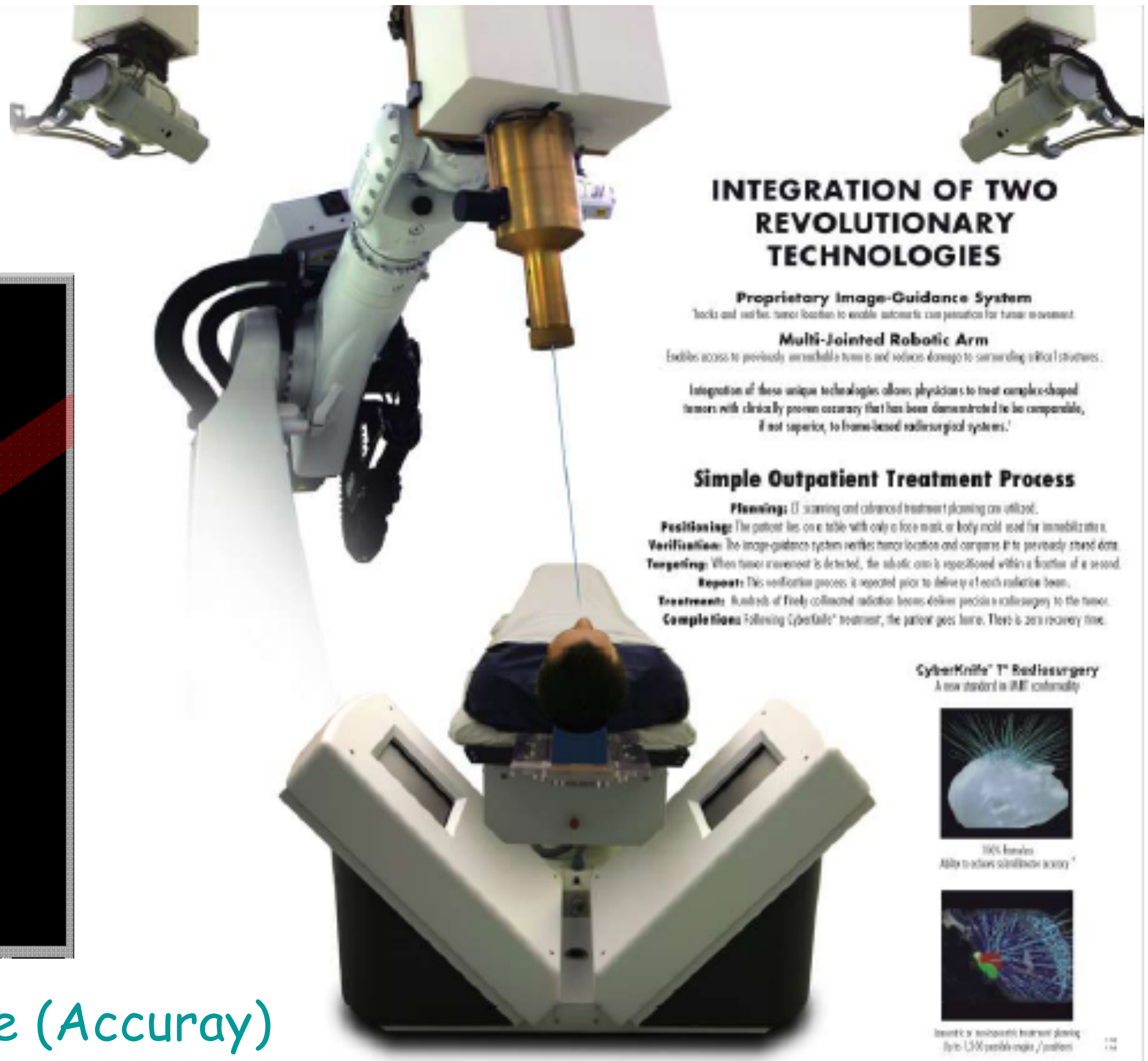
Manipulation of Deformable Objects



Topologically defined goal

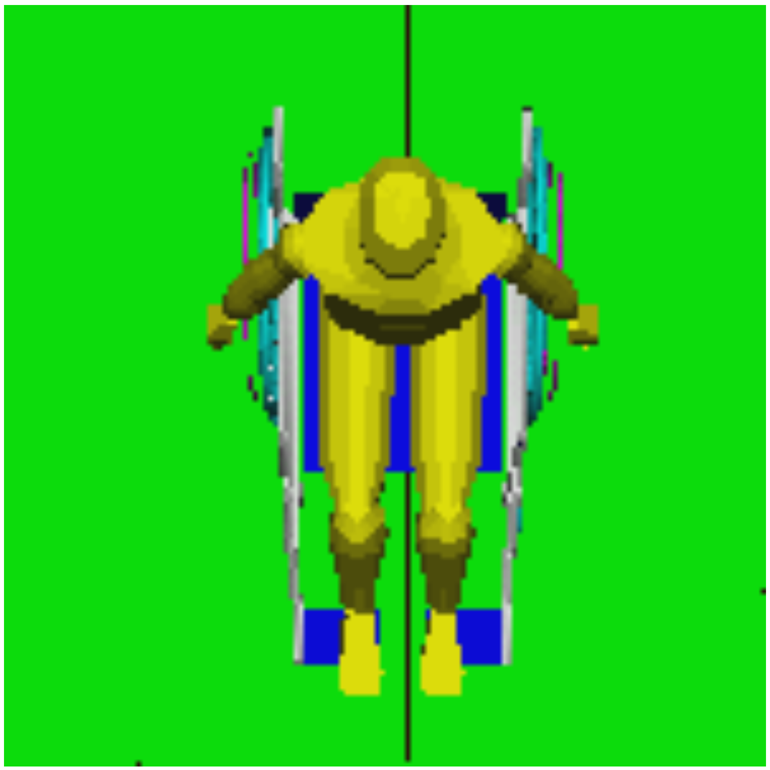
Radiosurgical Planning

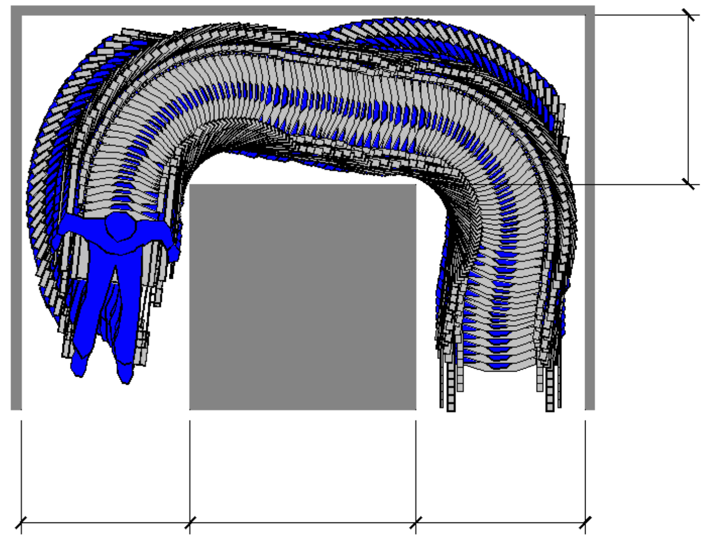


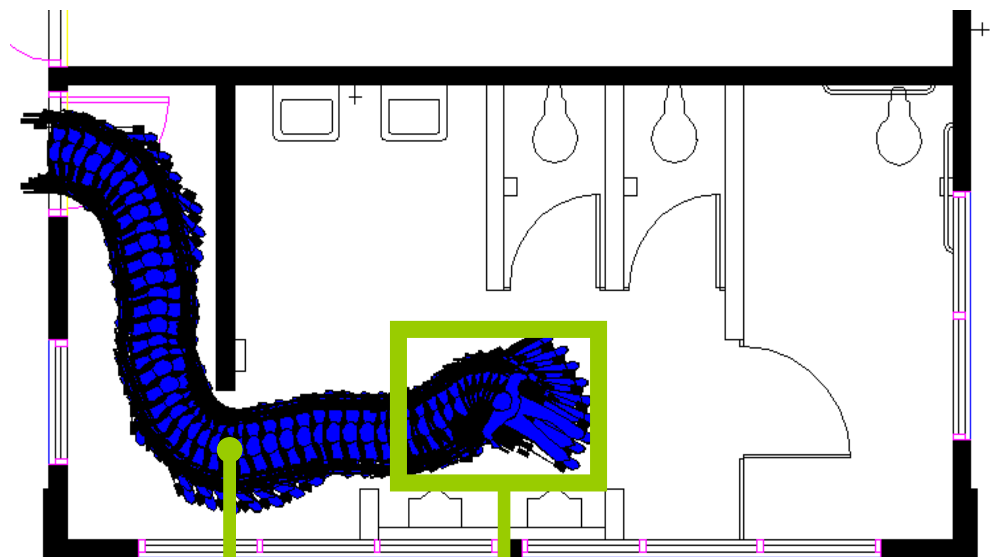


CyberKnife (Accuray)

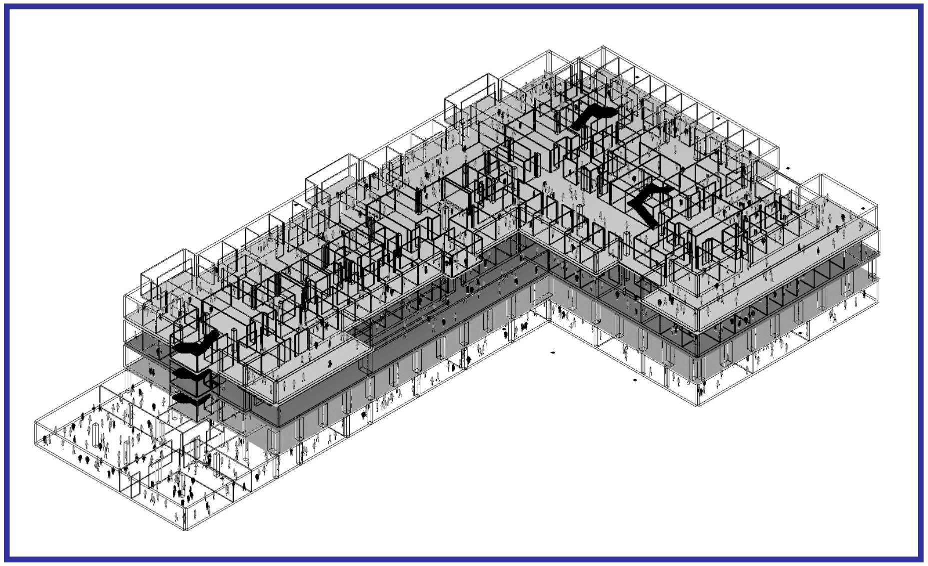
Building Code Verification

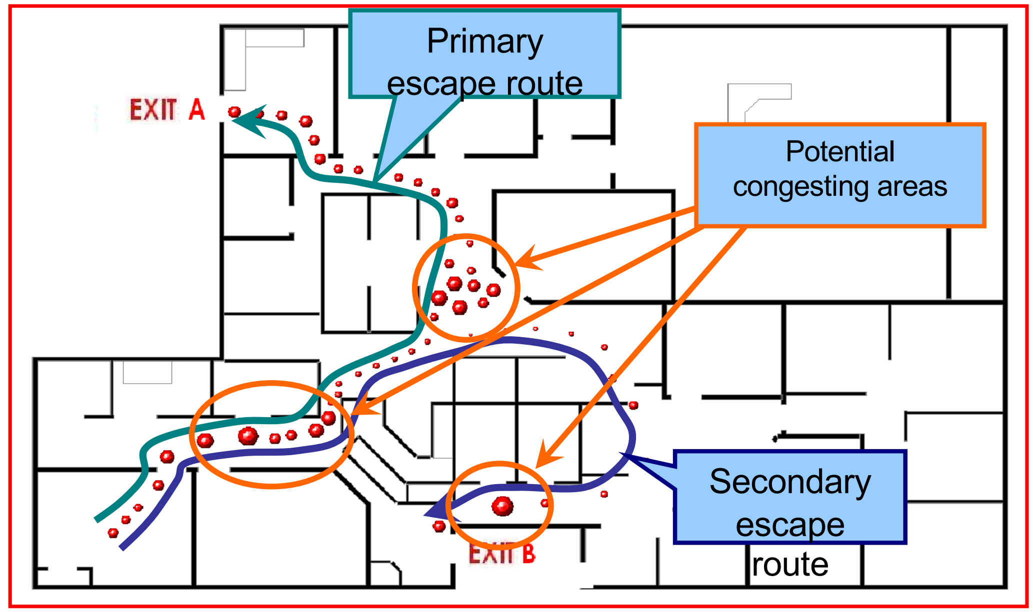


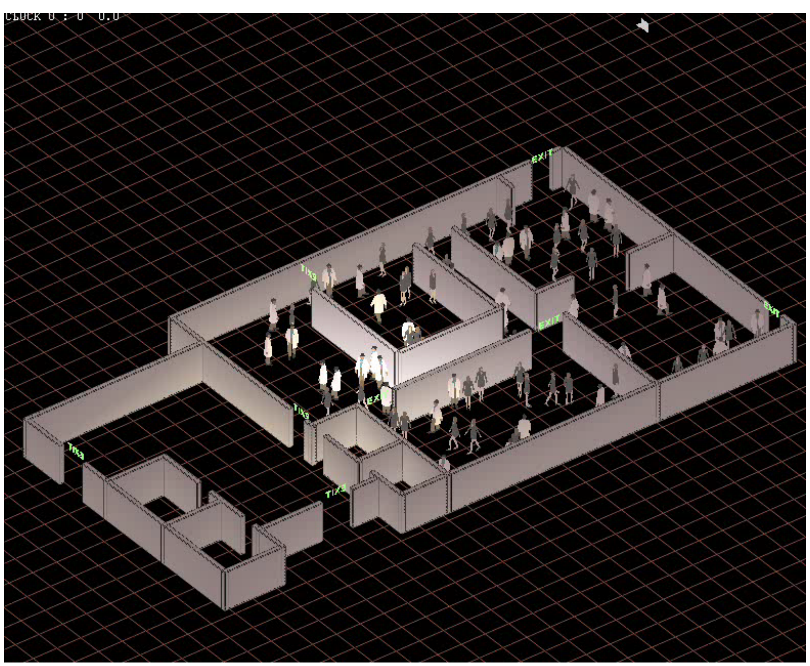




Egress Simulation



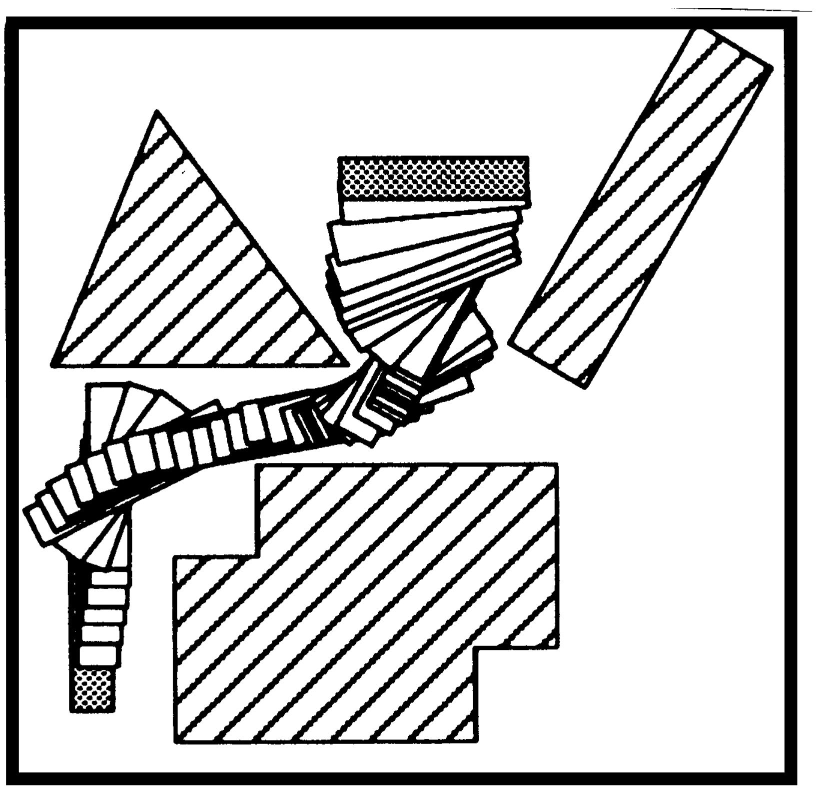


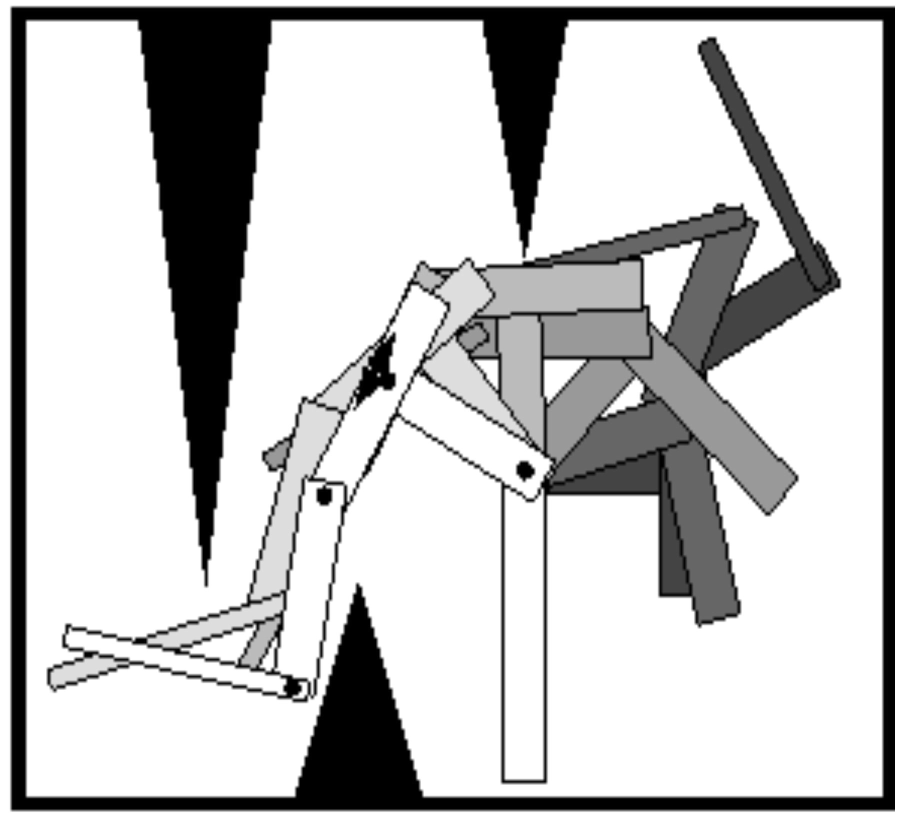


Transportation of A380 Fuselage through Small Villages

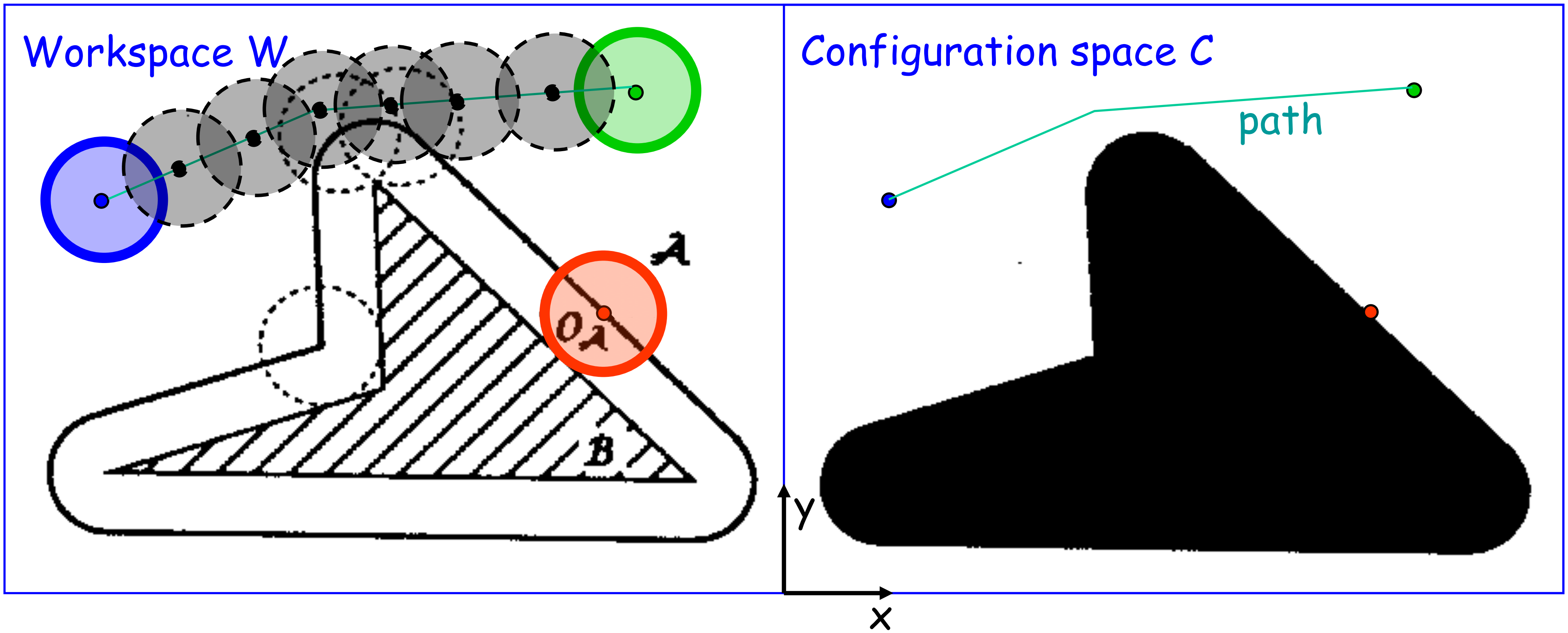
Kineo

Paths



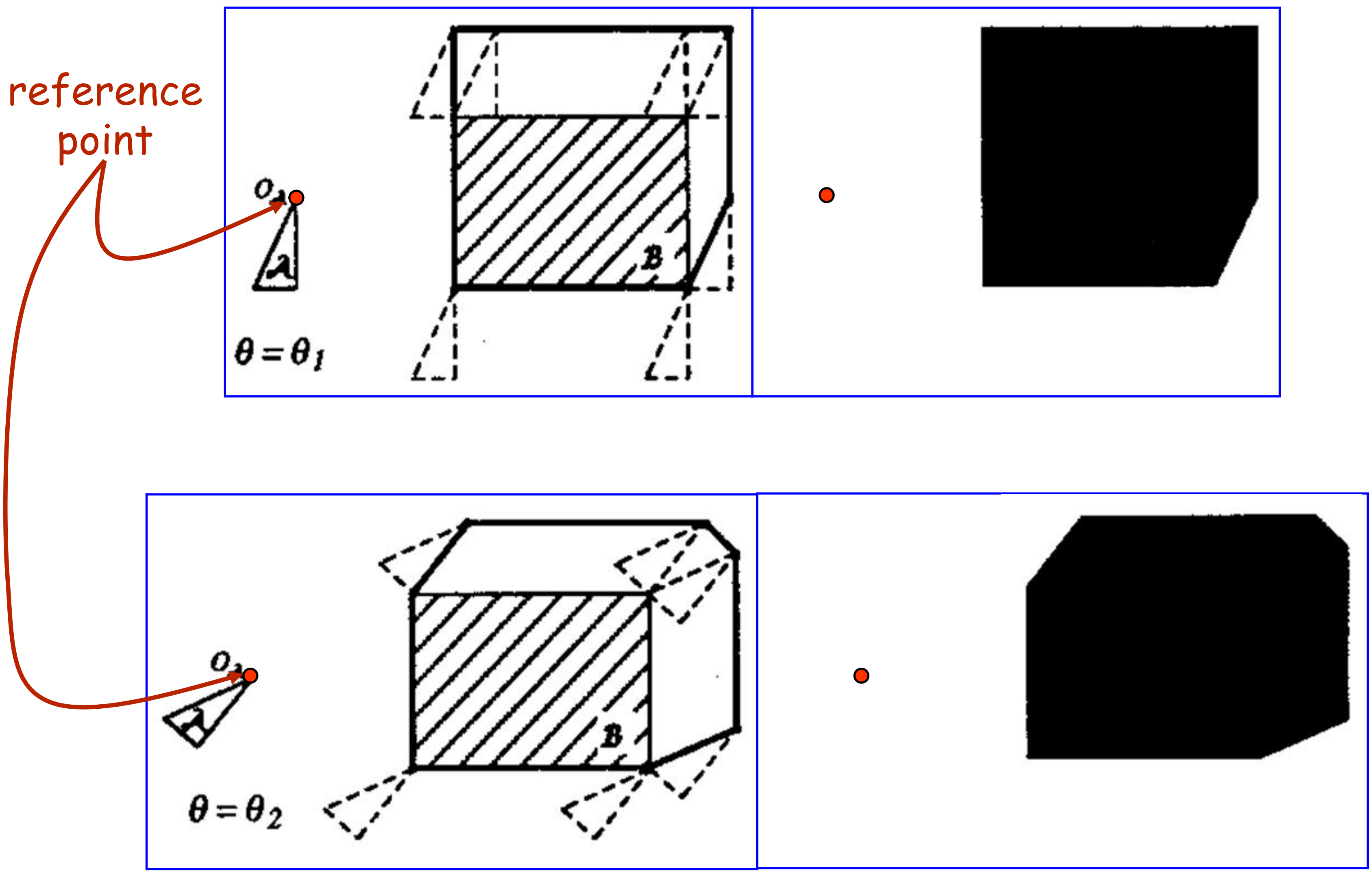


Disc Robot in 2-D Workspace

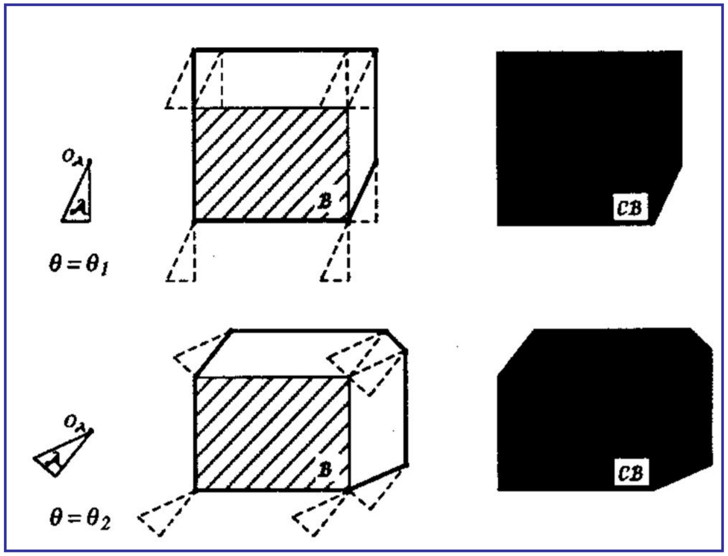


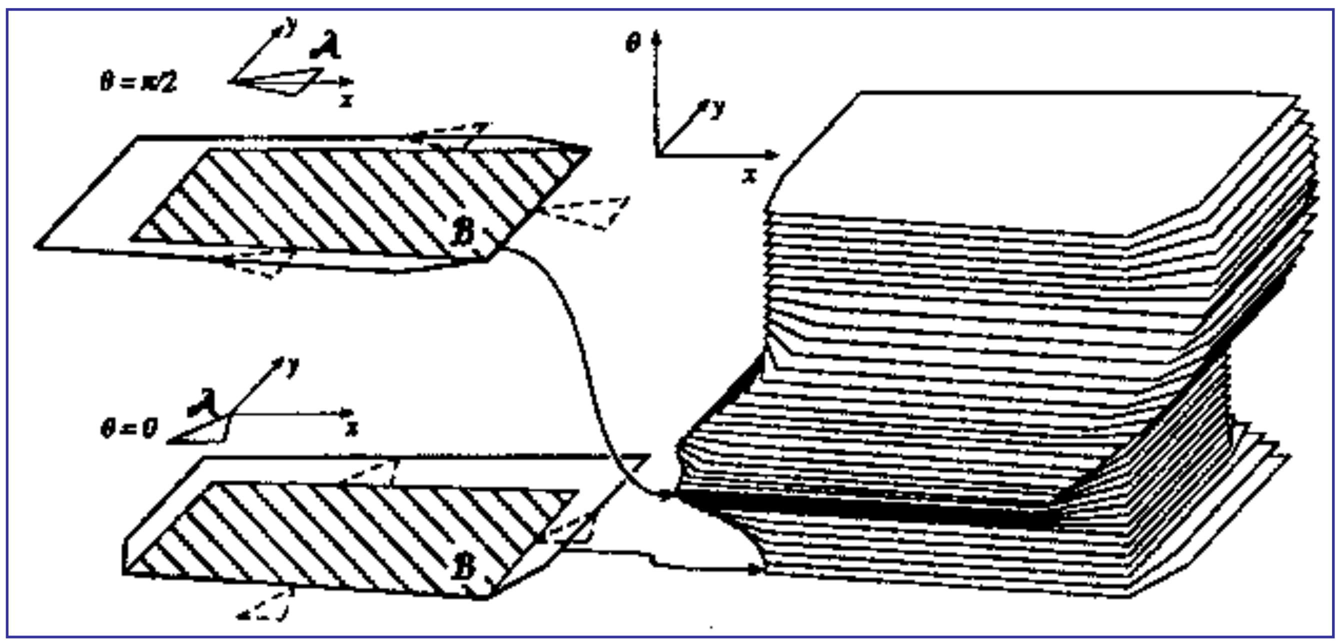
configuration = coordinates (x,y) of robot's center
configuration space C = {(x,y)}
free space F = subset of collision-free configurations

Translating Polygon in 2-D Workspace

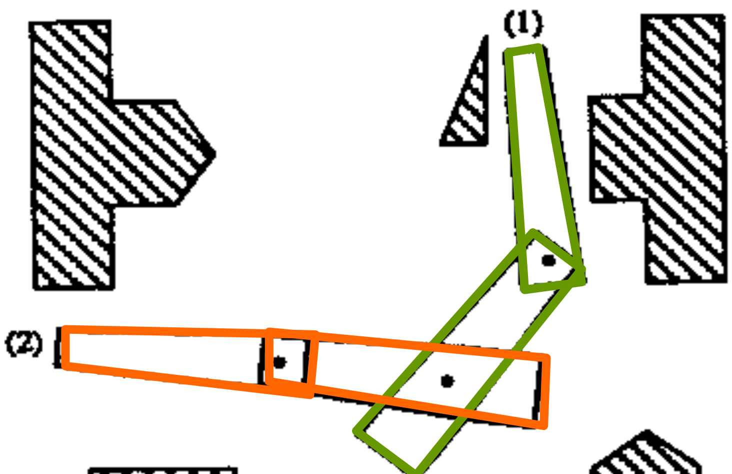


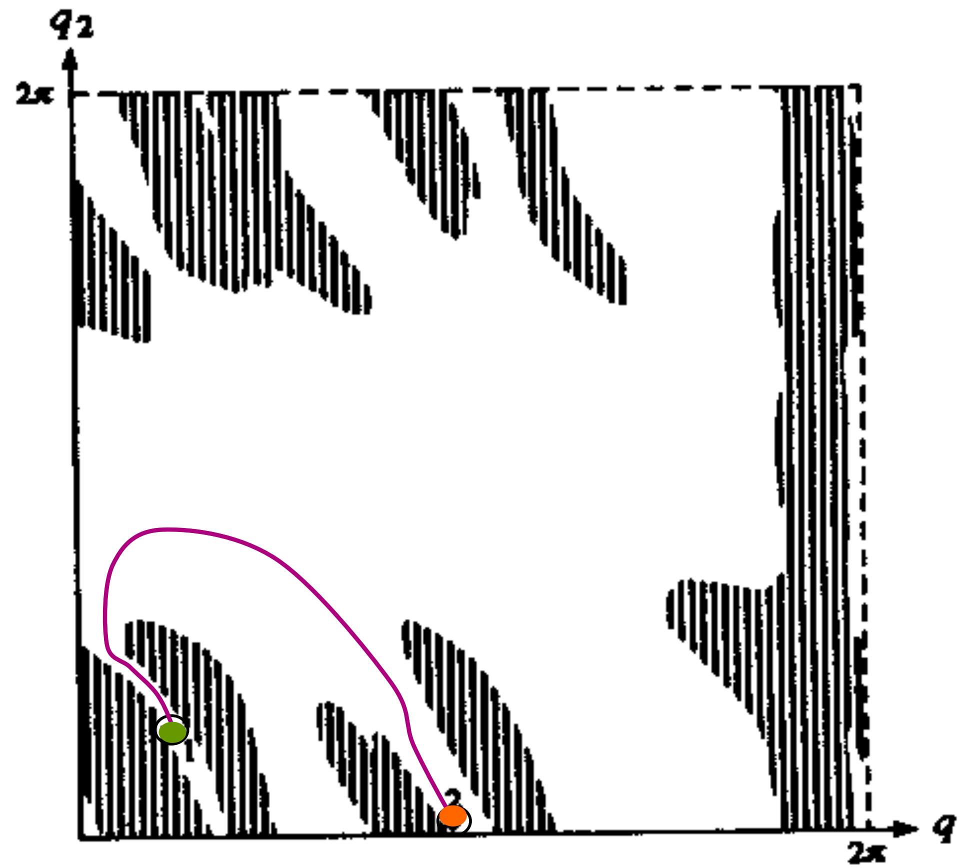
Translating & Rotating Polygon in 2-D Workspace





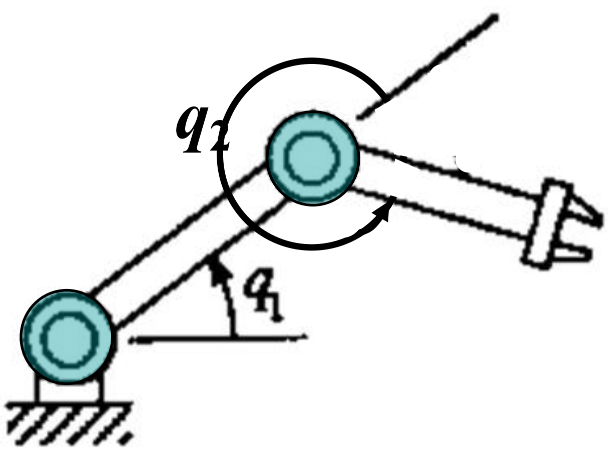
Tool: Configuration Space (C-Space C)

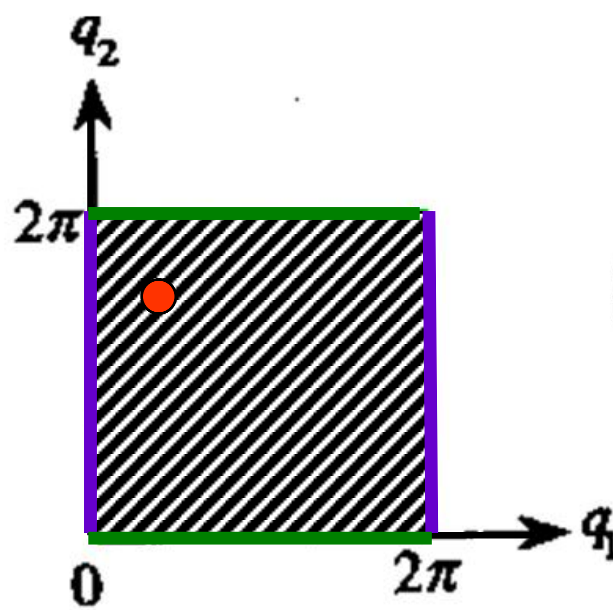


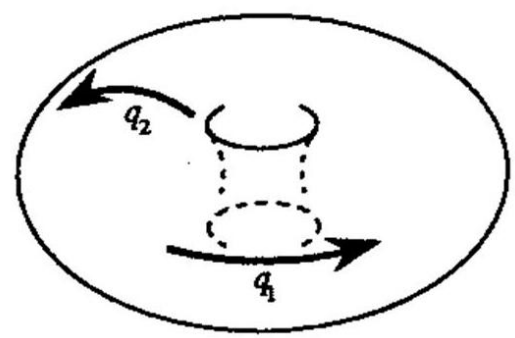


Configuration Space

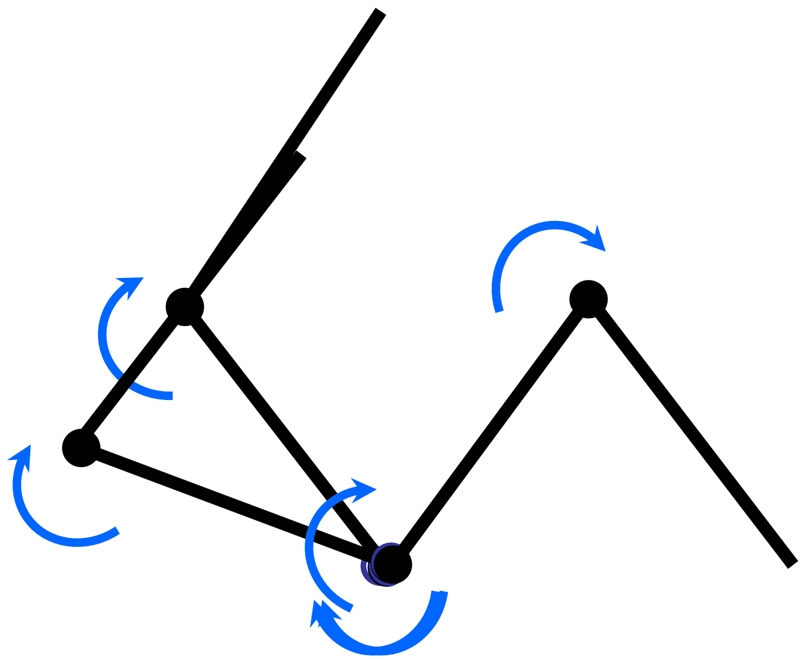
- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space

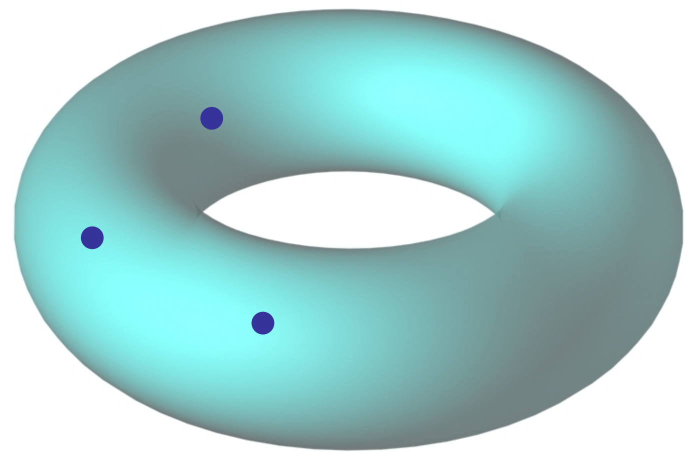




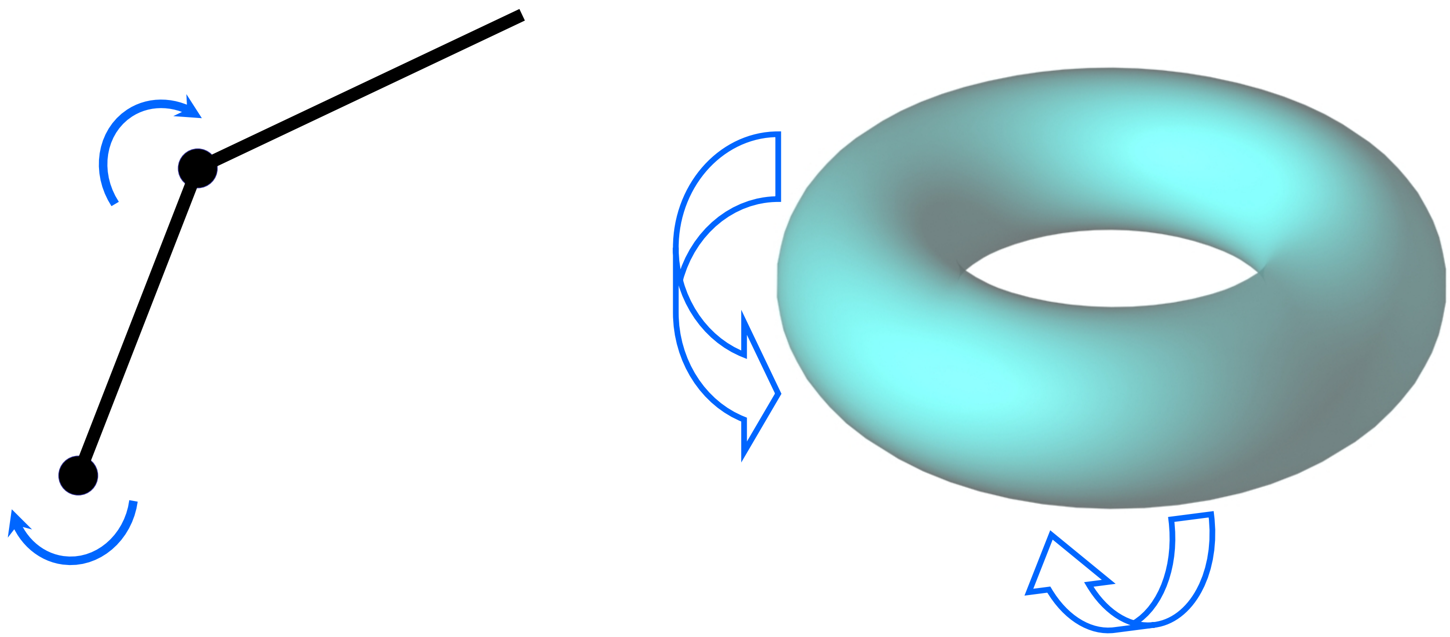


Move

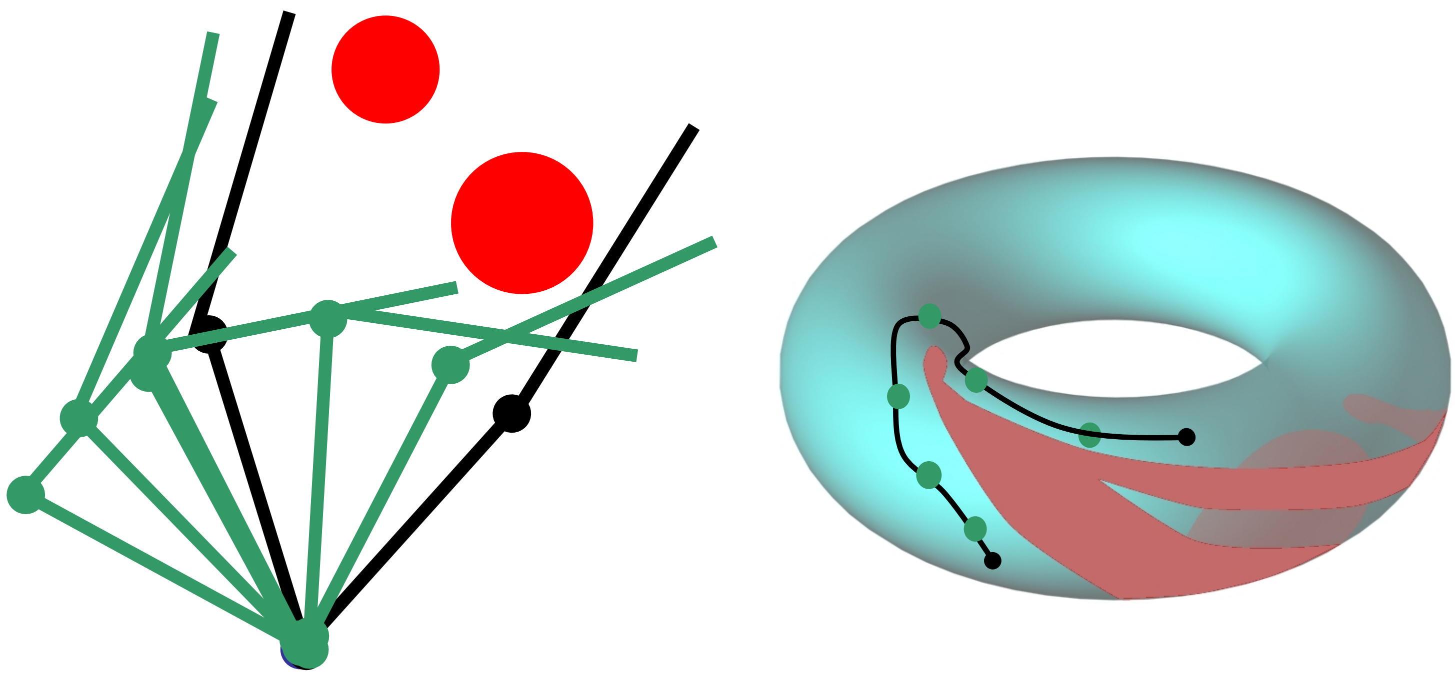




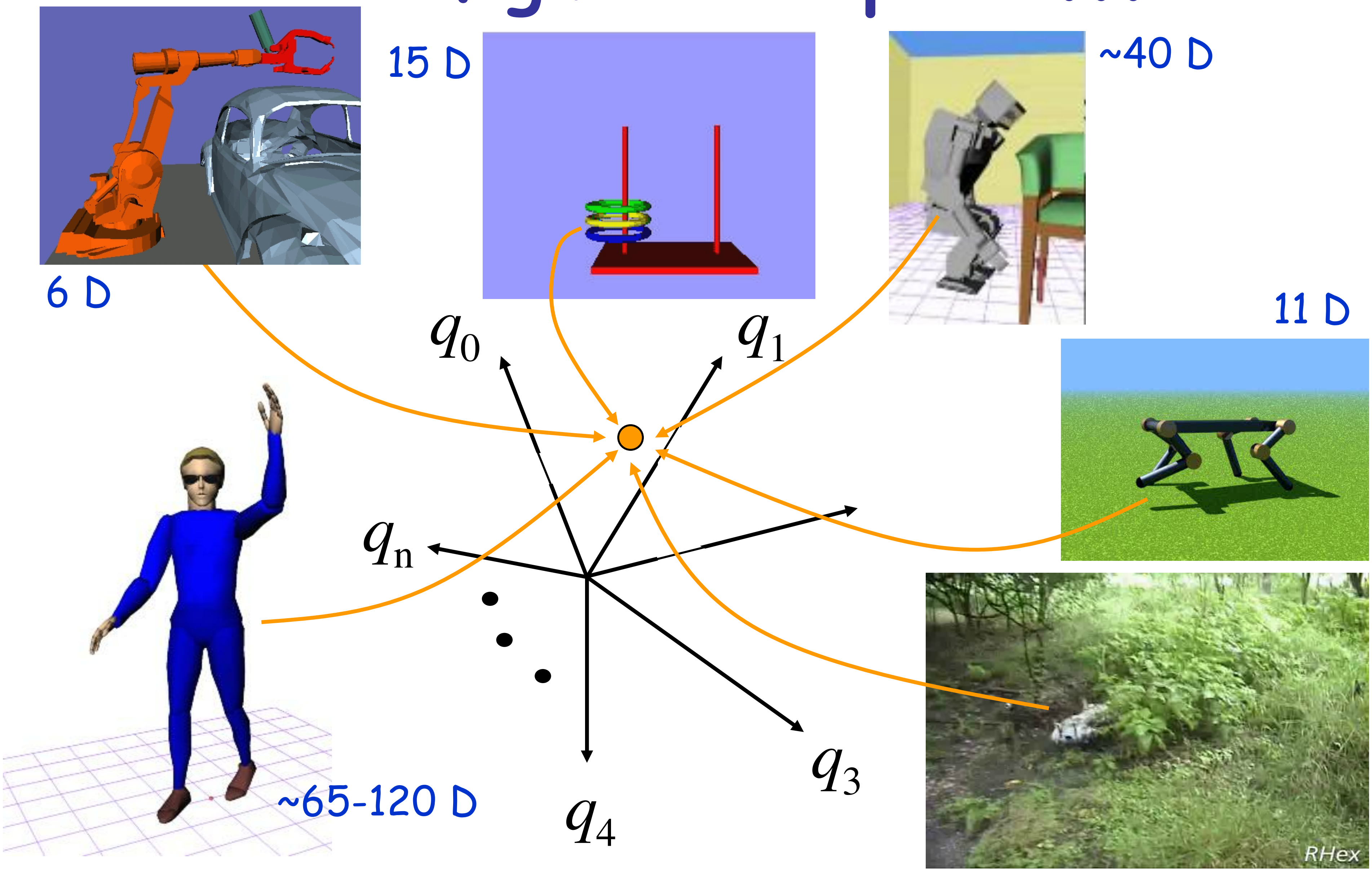
Configuration Space



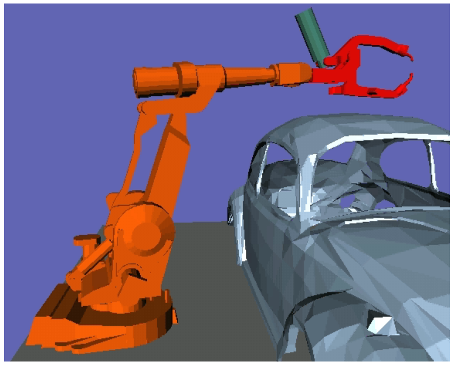
Configuration Space

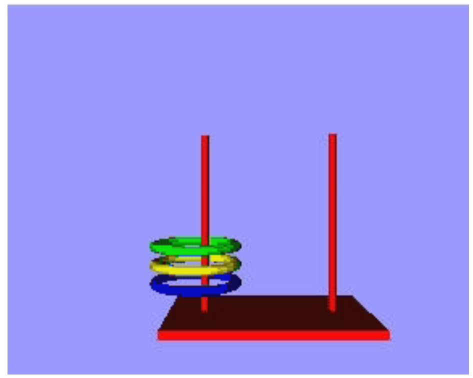


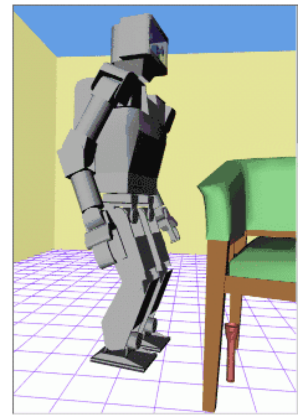
Every robot maps to a point in its configuration space ...

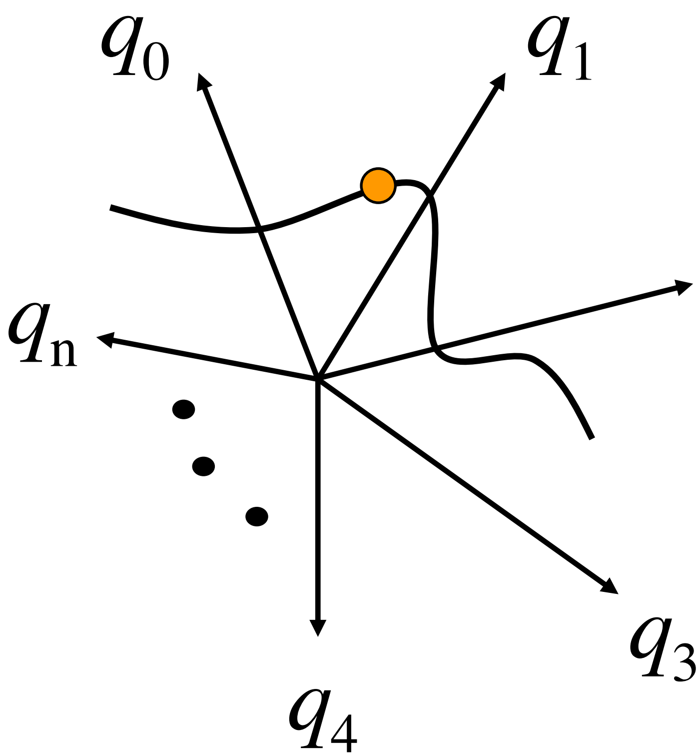


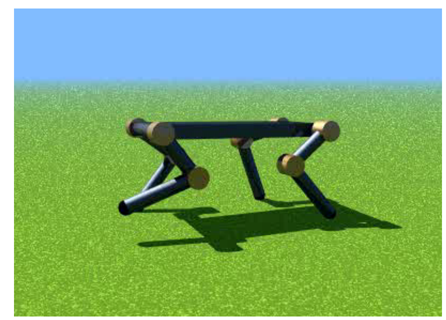
... and every robot path is a curve in configuration space



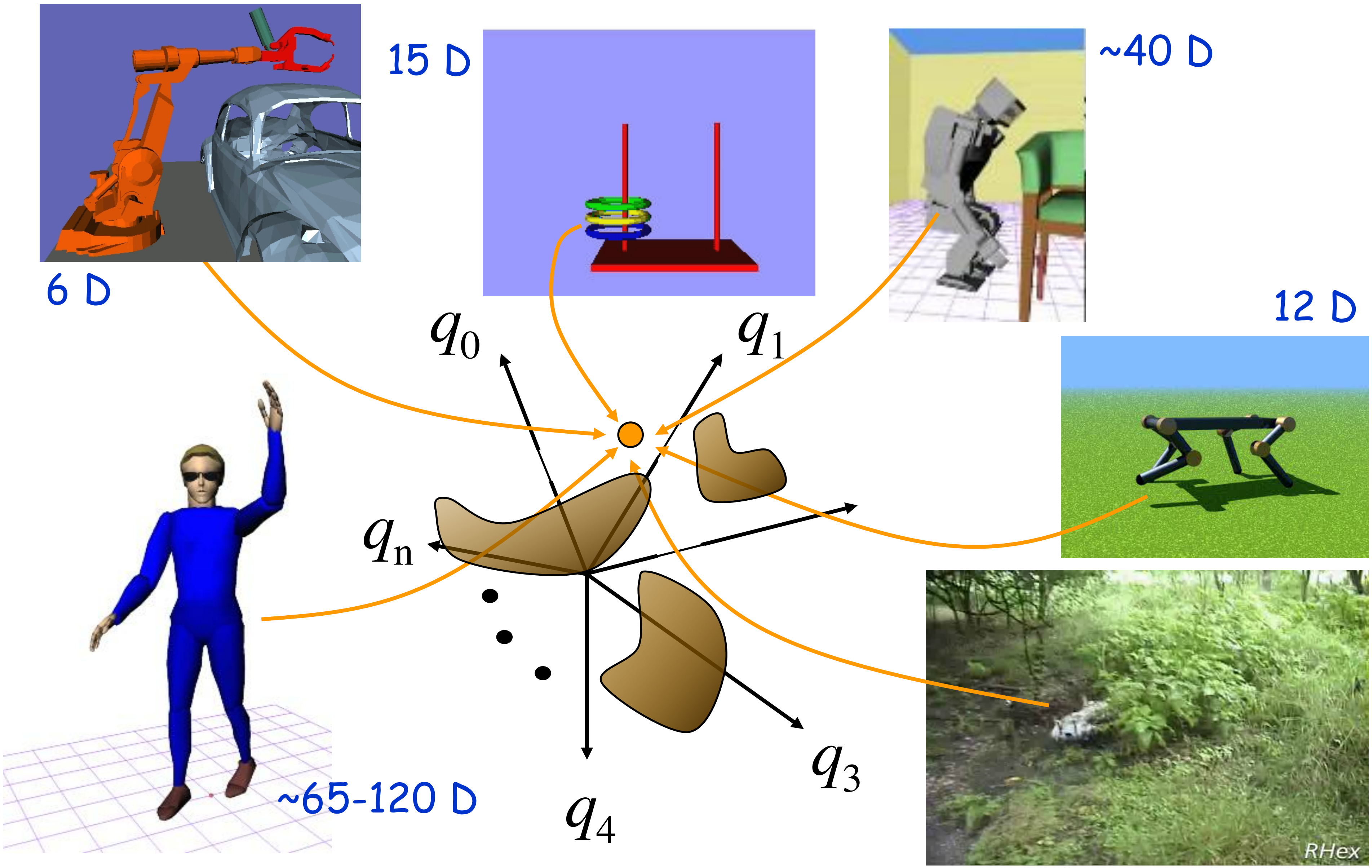




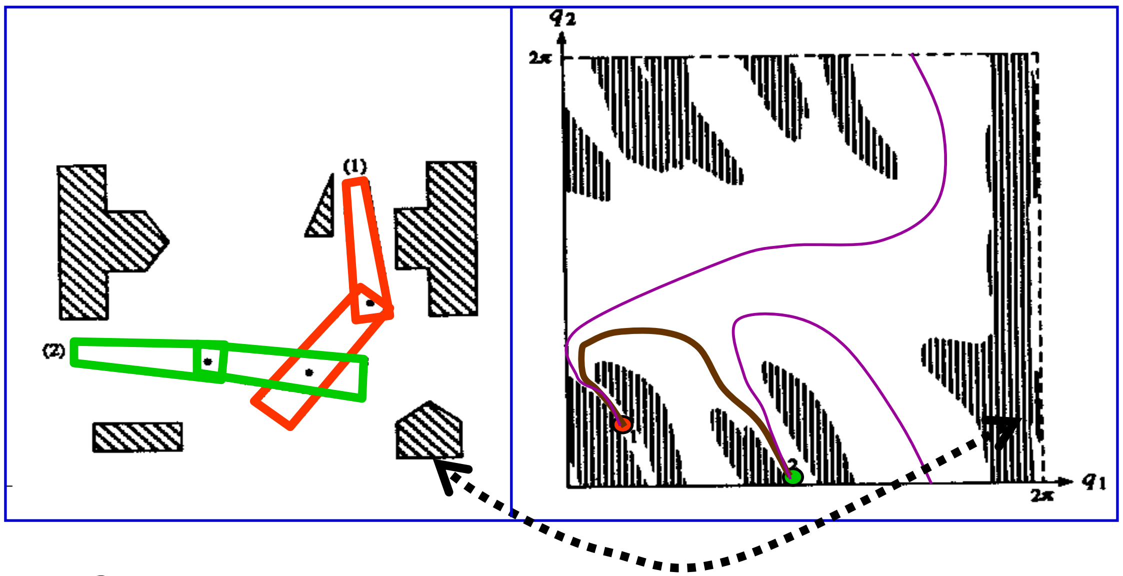




But how do obstacles (and other constraints) map in configuration space?

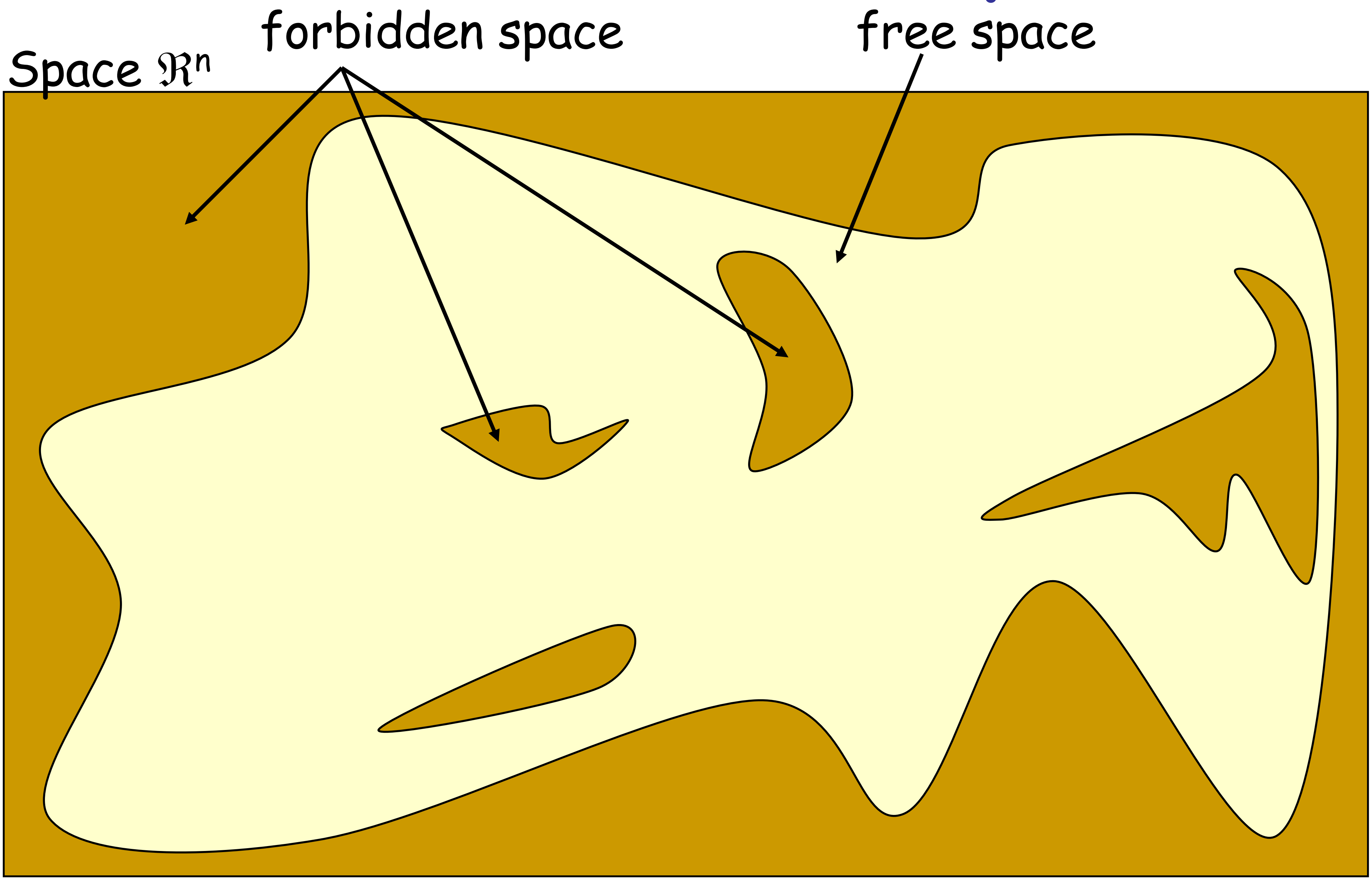


Probabilistic Roadmaps (Sampling-Based Planning)



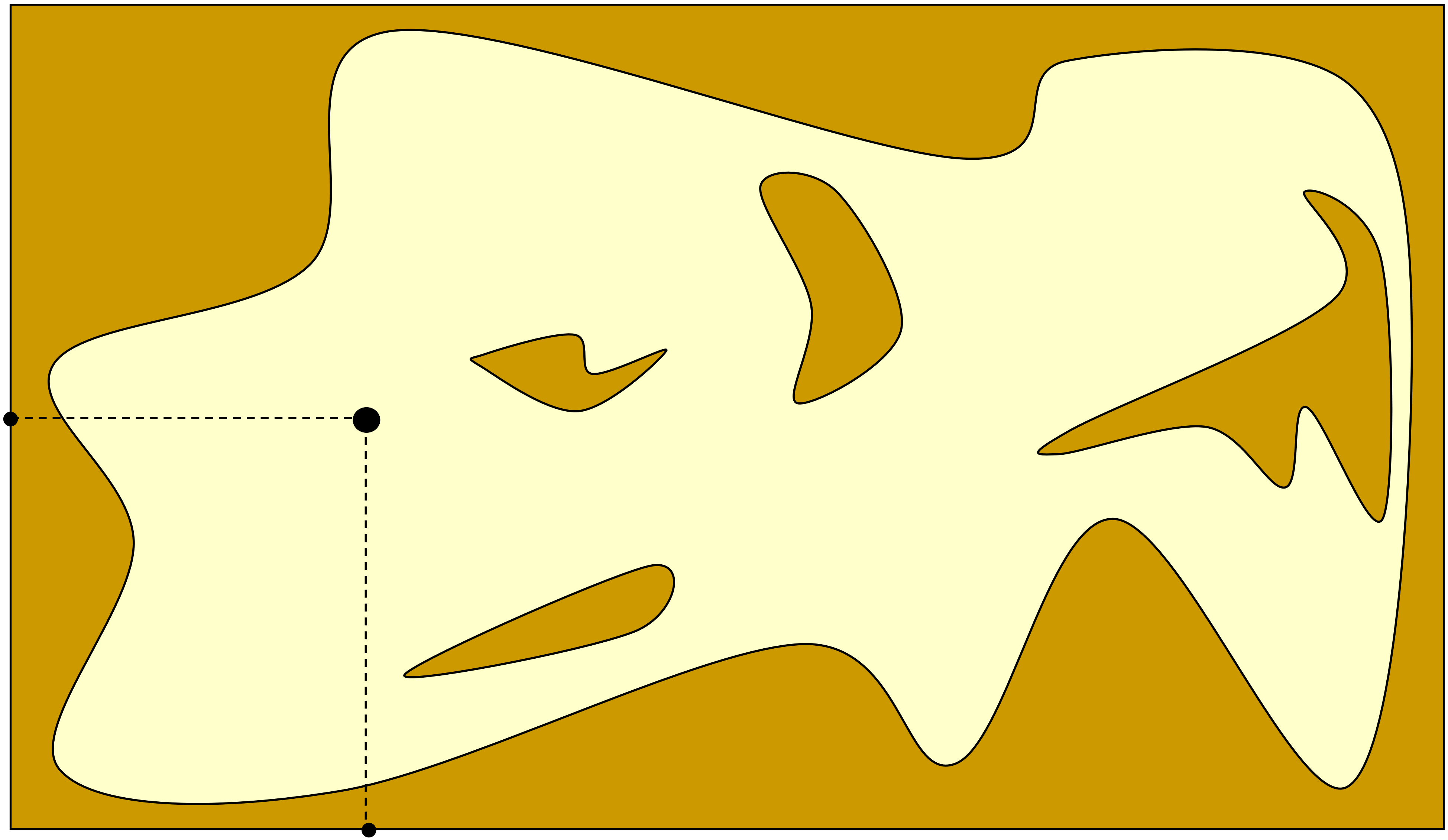
- The cost of computing an exact representation of the configuration space is often prohibitive.
- But very fast algorithms exist to check if a robot at a given configuration collides with obstacles.
- → Basic idea of Probabilistic Roadmaps (PRMs): Compute a very simplified representation of the free space by sampling configurations at random.

Probabilistic Roadmap (PRM)



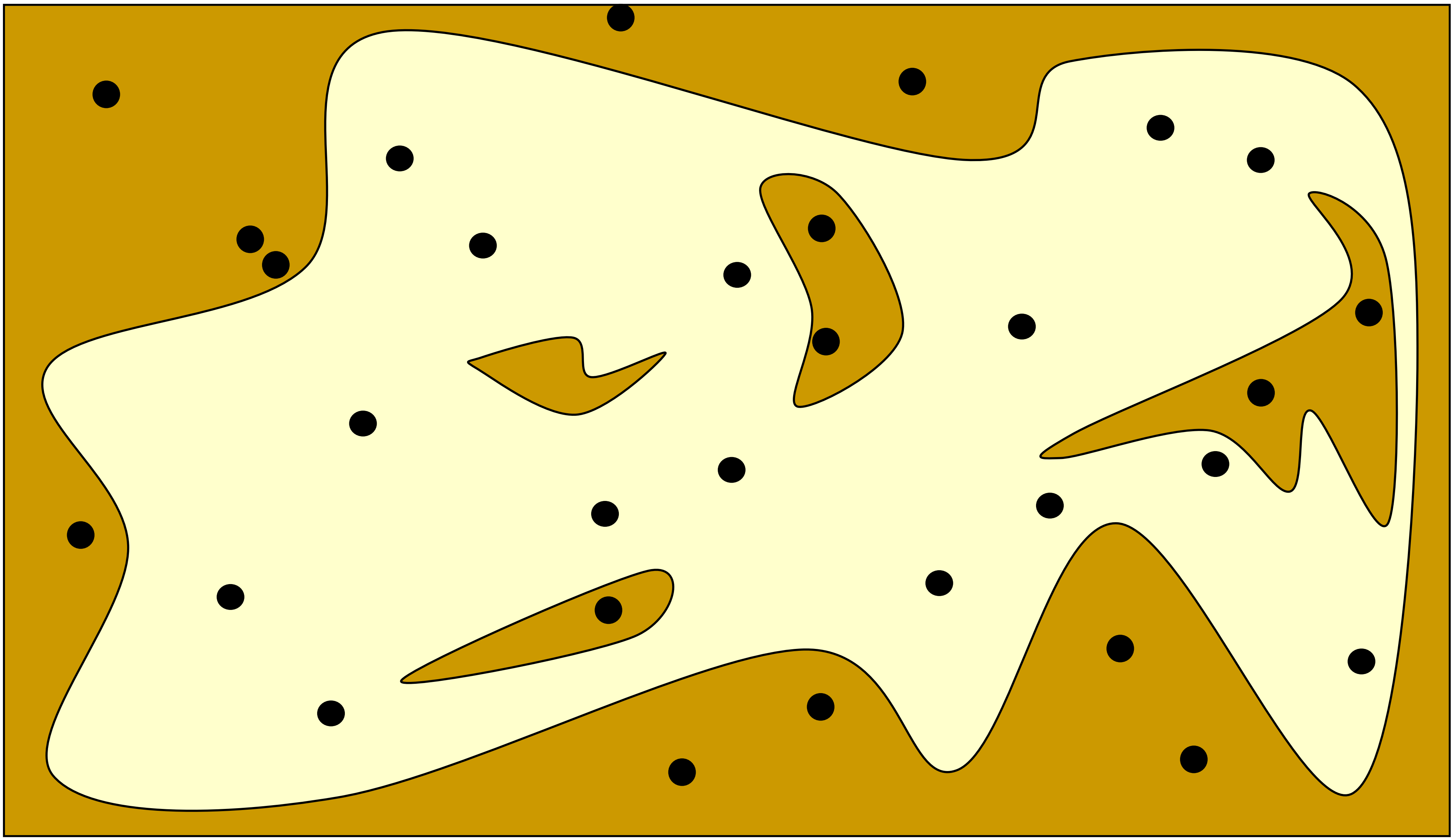
Probabilistic Roadmap (PRM)

Configurations are sampled by picking coordinates at random



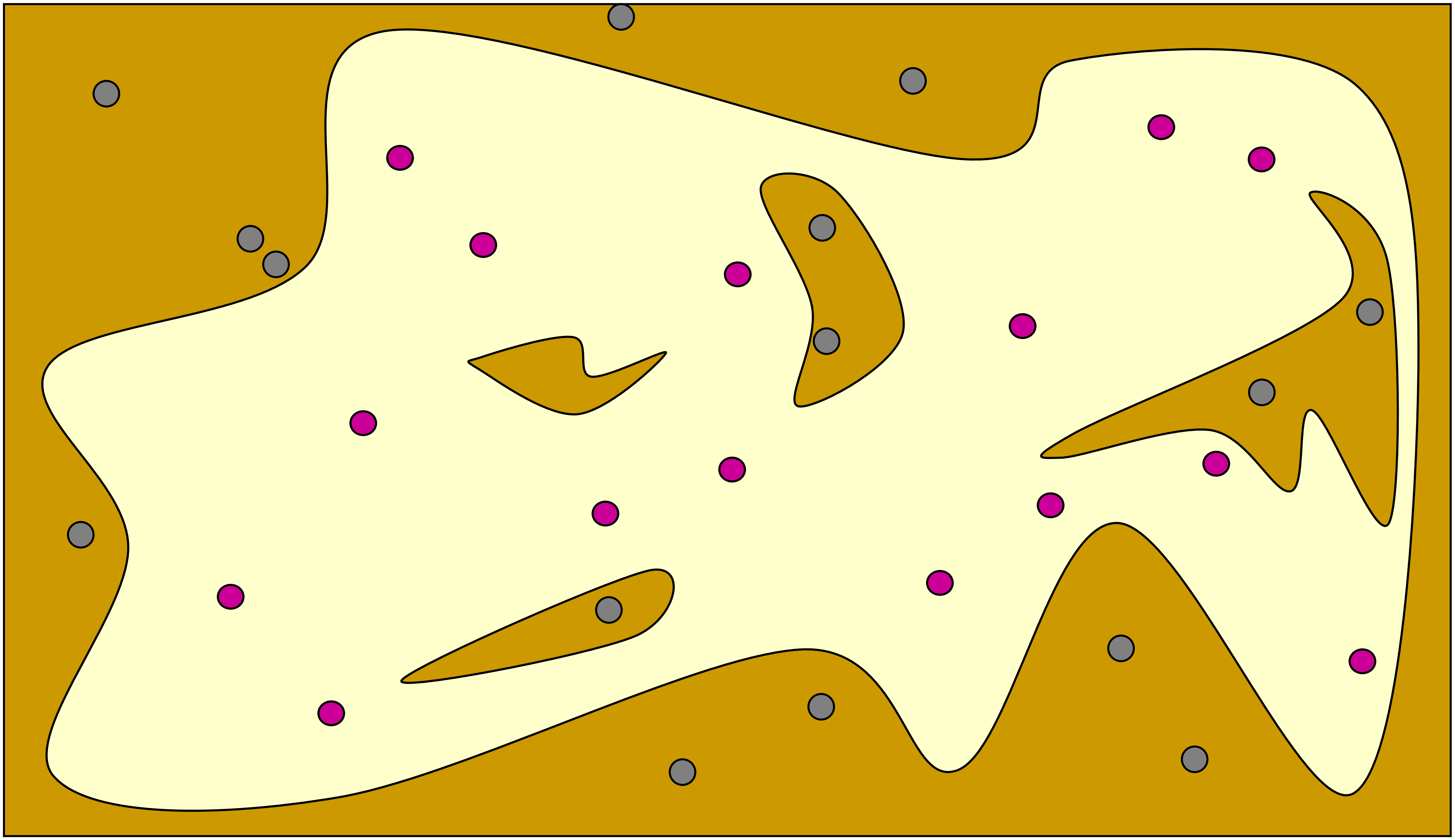
Probabilistic Roadmap (PRM)

Configurations are sampled by picking coordinates at random



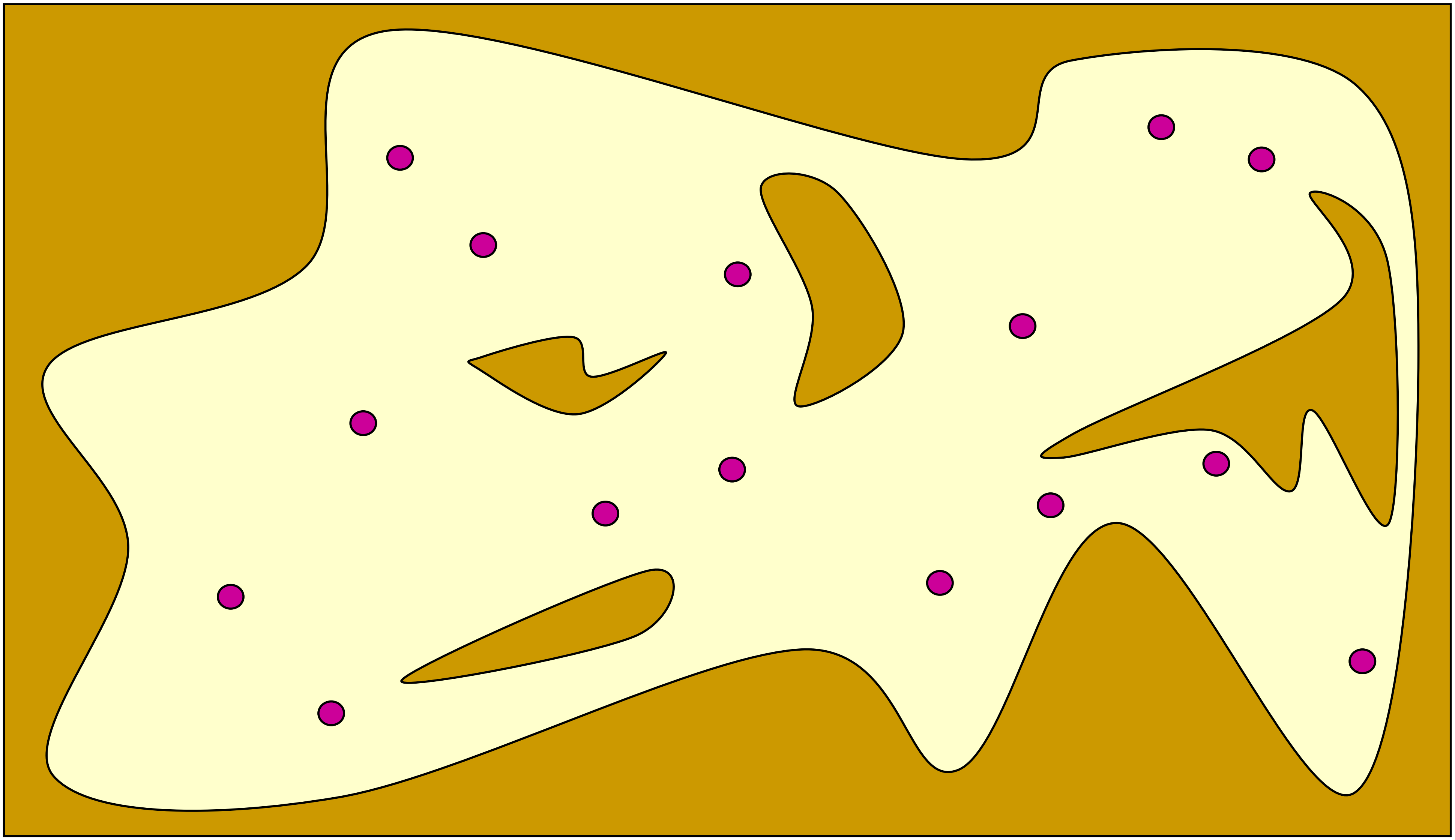
Probabilistic Roadmap (PRM)

Sampled configurations are tested for collision (in workspace!)



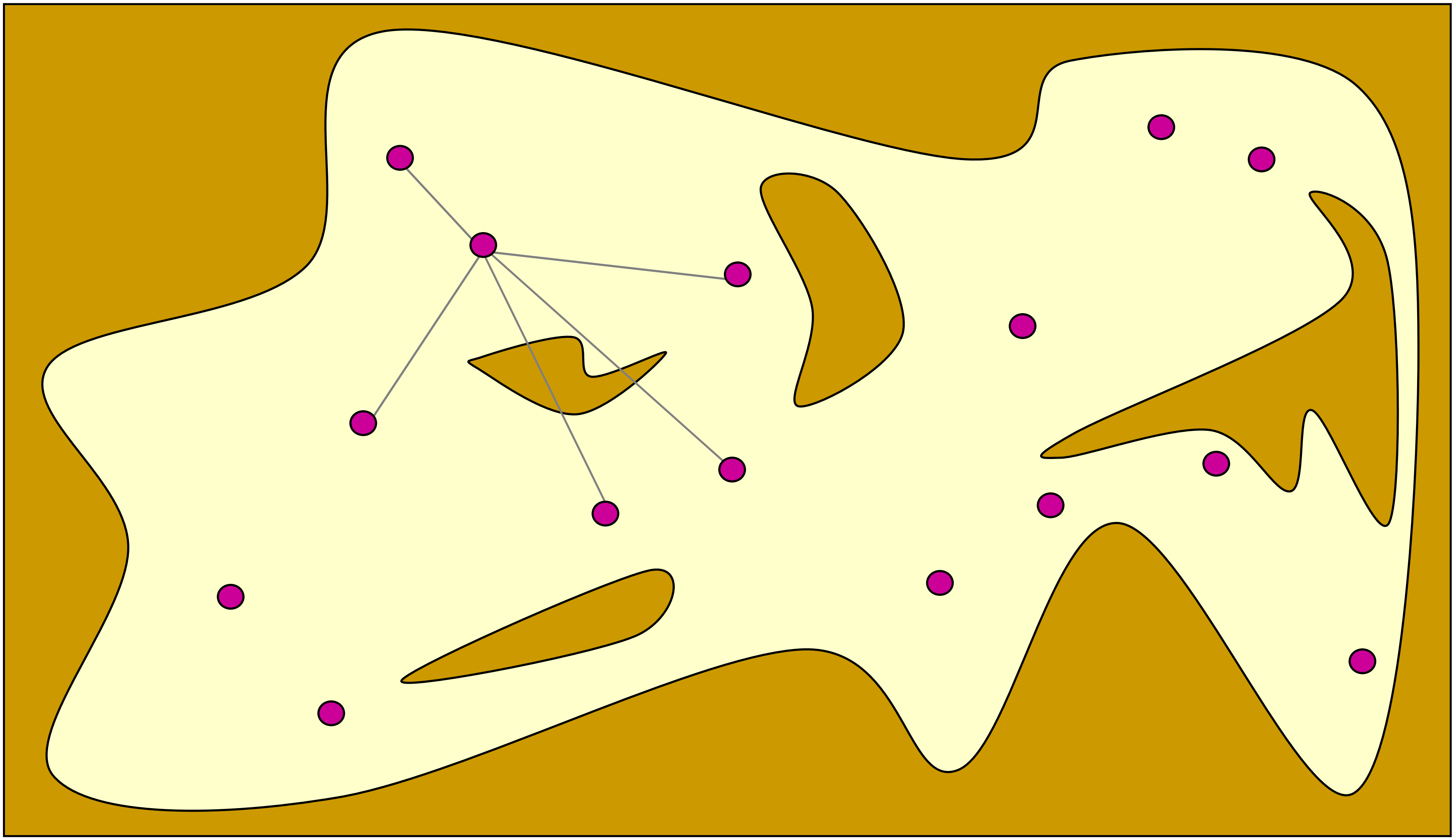
Probabilistic Roadmap (PRM)

The collision-free configurations are retained as “milestones”



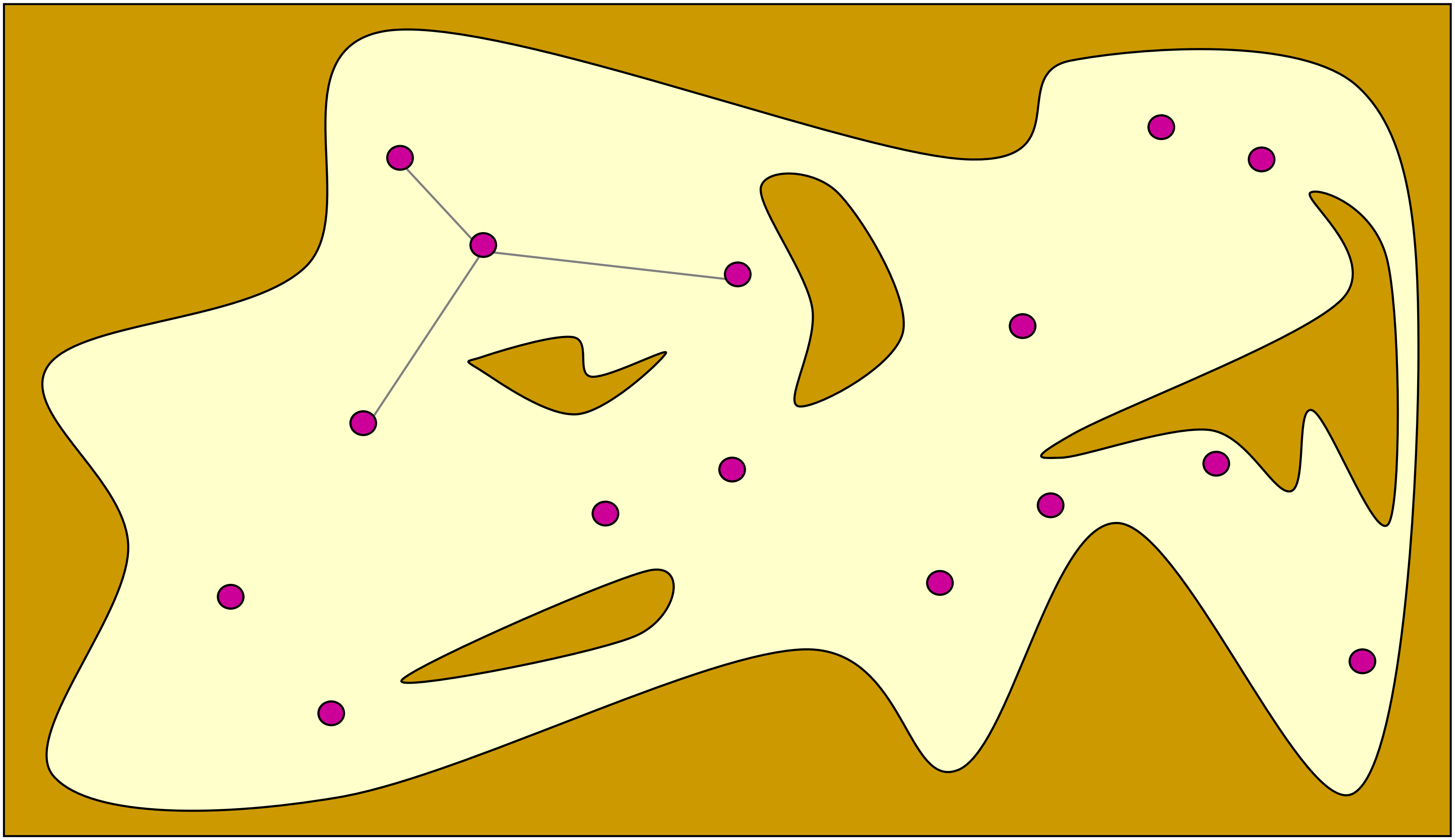
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



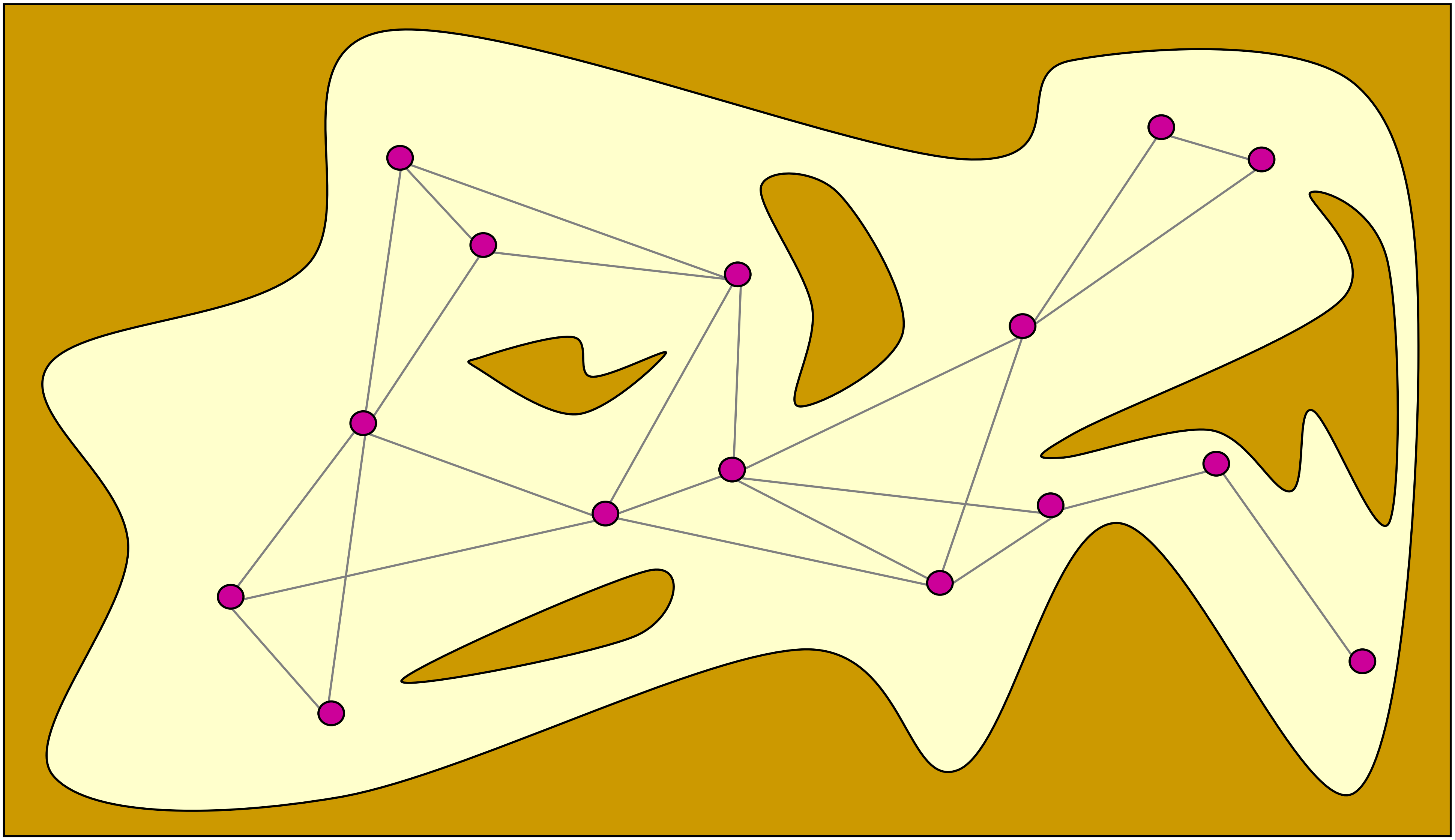
Probabilistic Roadmap (PRM)

Each milestone is linked by straight paths to its k-nearest neighbors



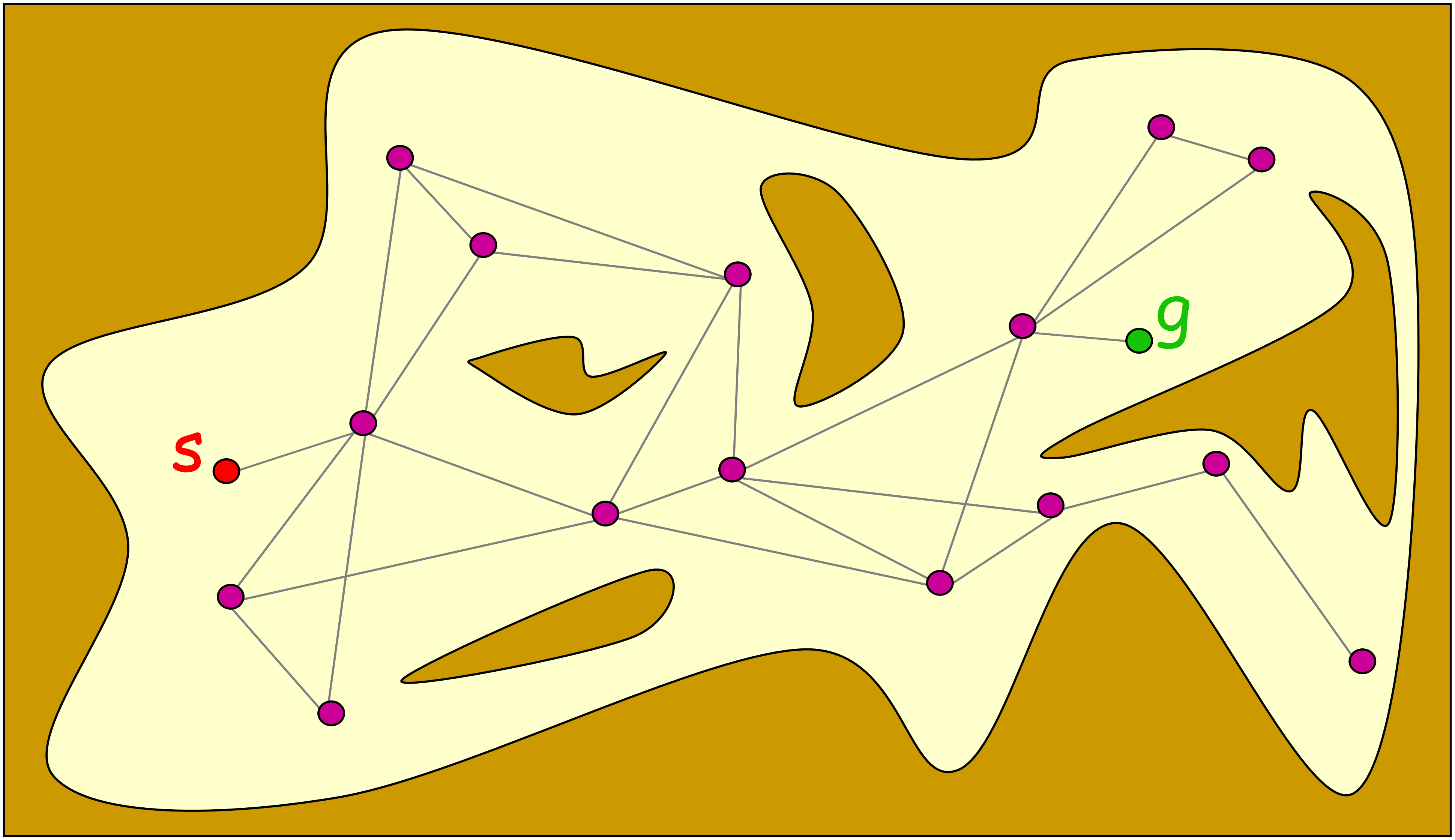
Probabilistic Roadmap (PRM)

The collision-free links are retained to form the PRM



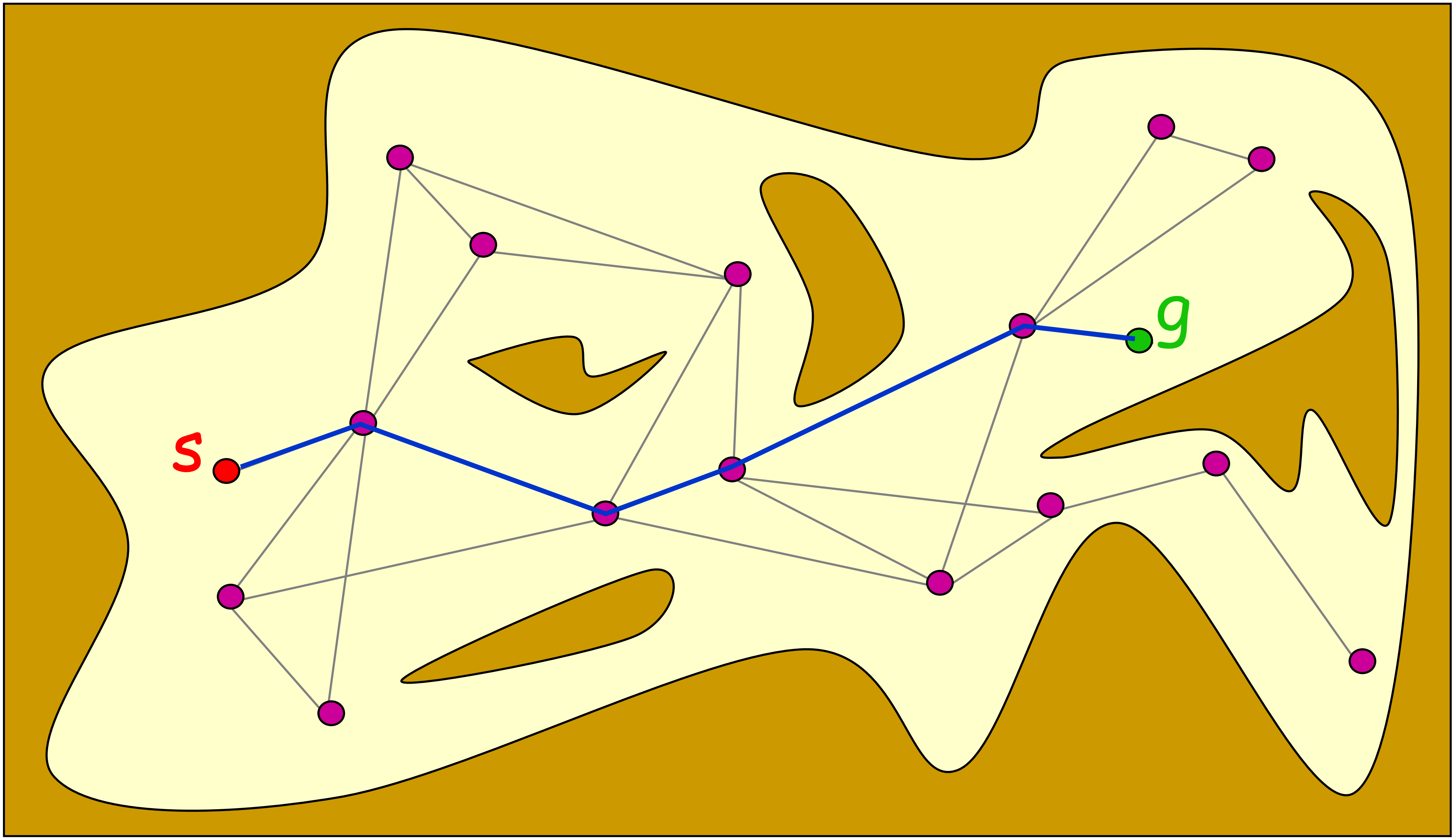
Probabilistic Roadmap (PRM)

The start and goal configurations are included as milestones



Probabilistic Roadmap (PRM)

The PRM is searched for a path from s to g



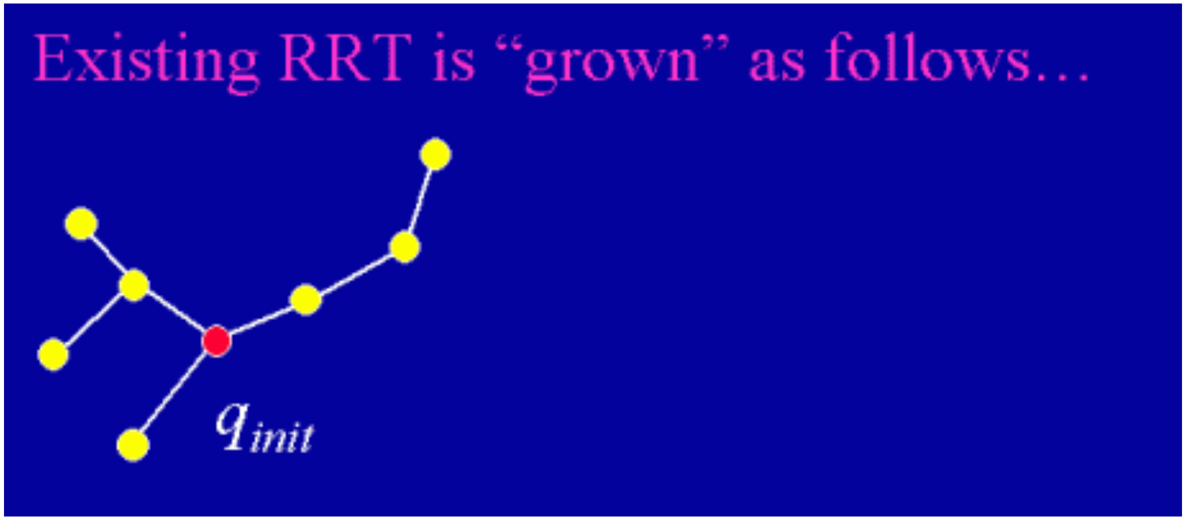
Many degrees of freedom

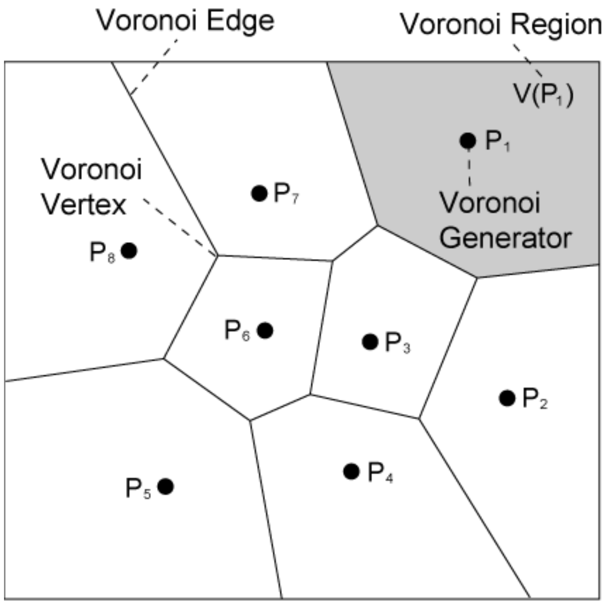
RRTs

[LaValle, 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)
T.init(x_init);
for k = 1 to K do
    x_rand ← RANDOM_STATE();
    EXTEND(T, x_rand);
Return T
```

```
EXTEND(T, x)
x_near ← NEAREST_NEIGHBOR(x, T);
if NEW_STATE(x, x_near, x_new, u_new) then
    T.add_vertex(x_new);
    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

$$\mu\left(X_{free}\right) = 0.92$$

$$d = 2$$

$$\zeta_d = \pi$$

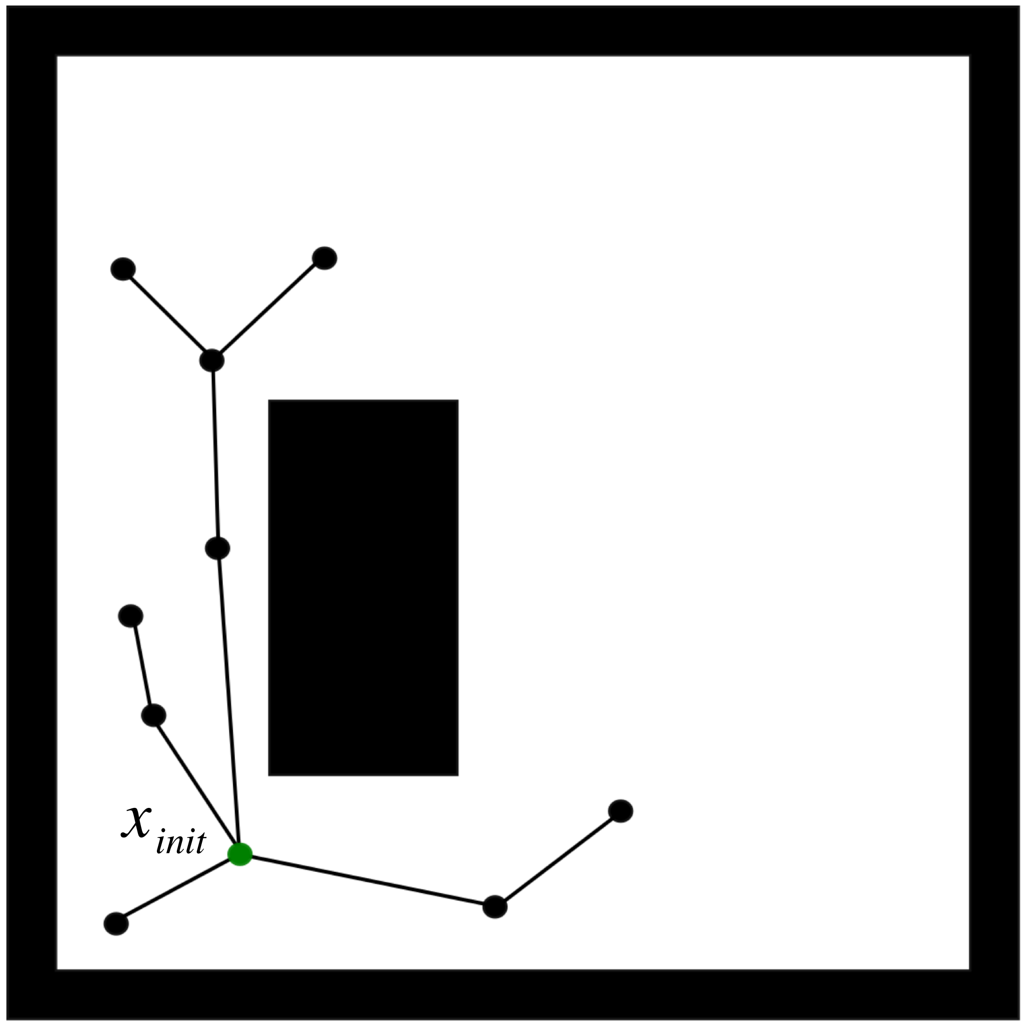
$$\gamma_{RRT} > \left(2(1+1/d)\right)^{1/d}\left(\frac{\mu\left(X_{free}\right)}{\zeta_d}\right)^{1/d} \approx 0.9373$$

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

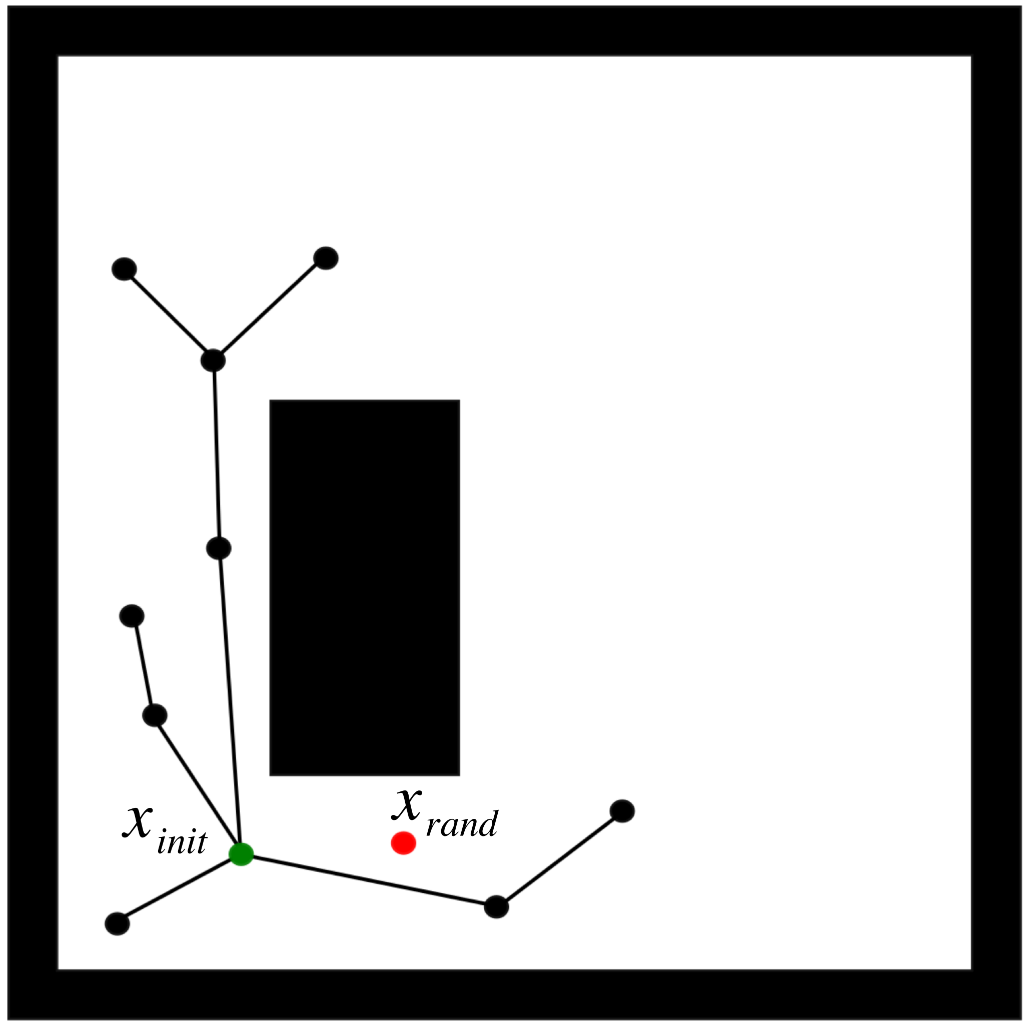
$$n = 10$$

$$r_{10} > 0.4497$$

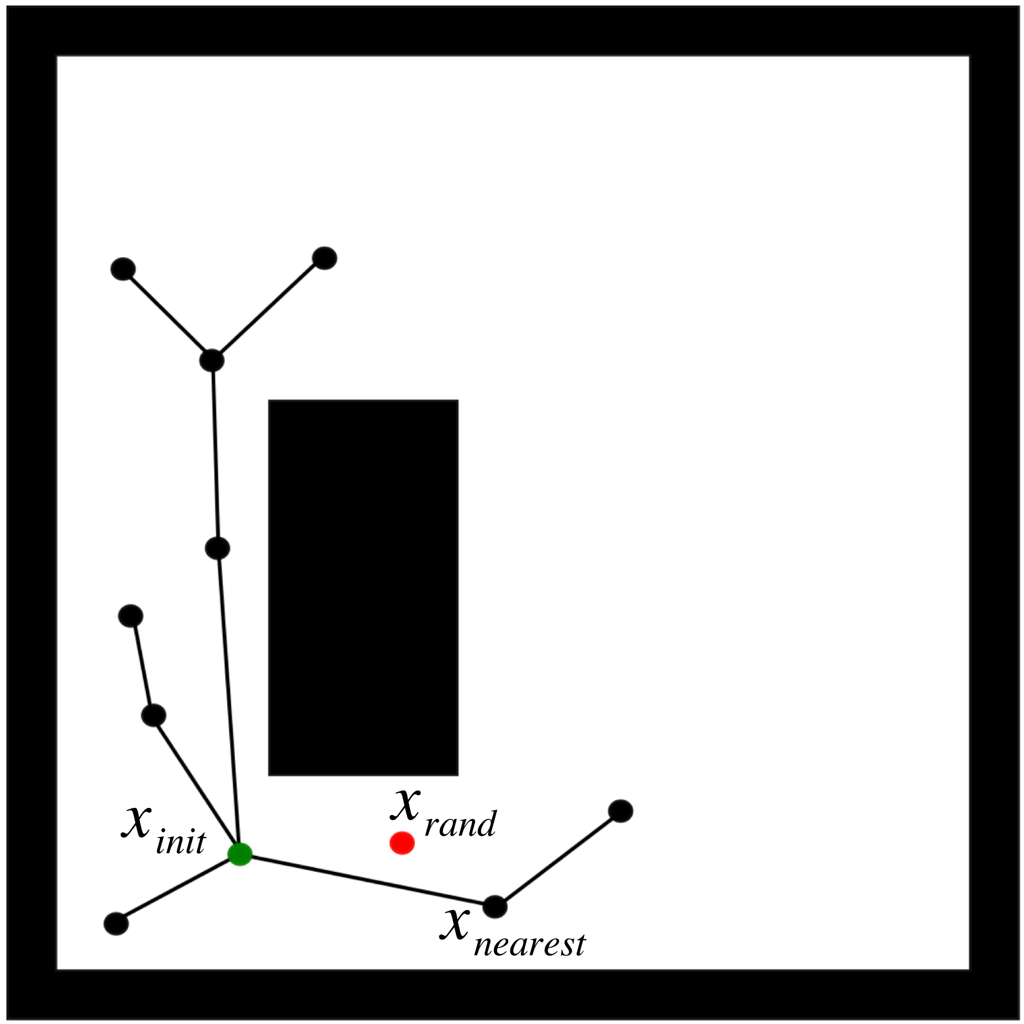
RRT*-Tree after iteration 9



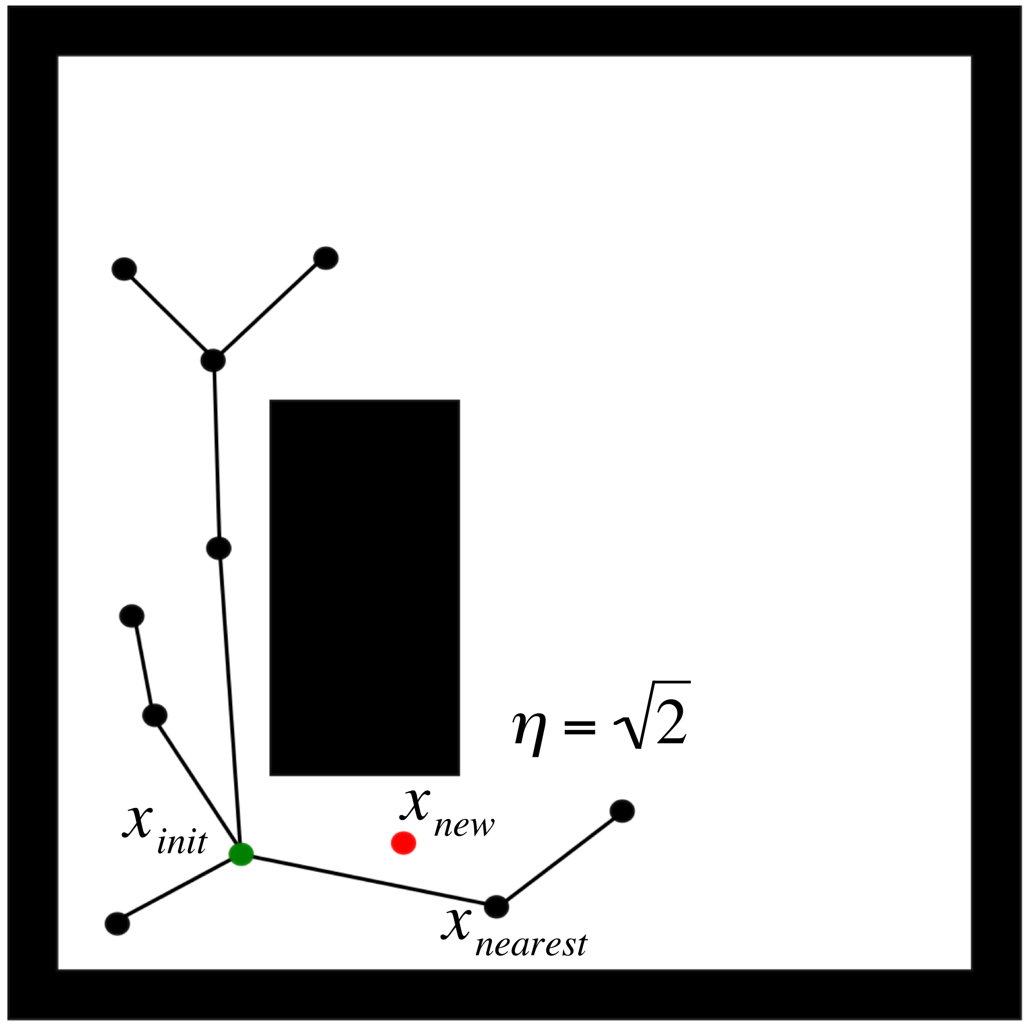
RRT*-New Sample



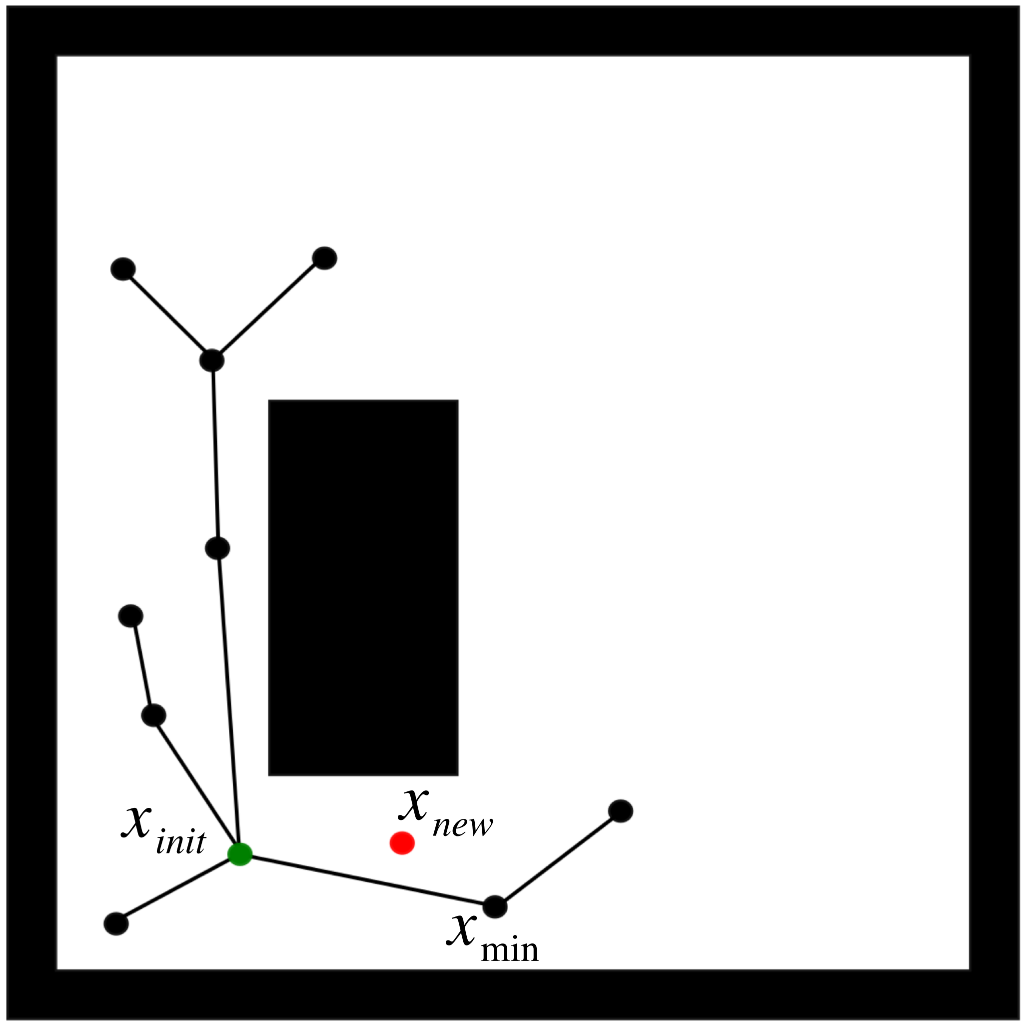
RRT*-New Sample



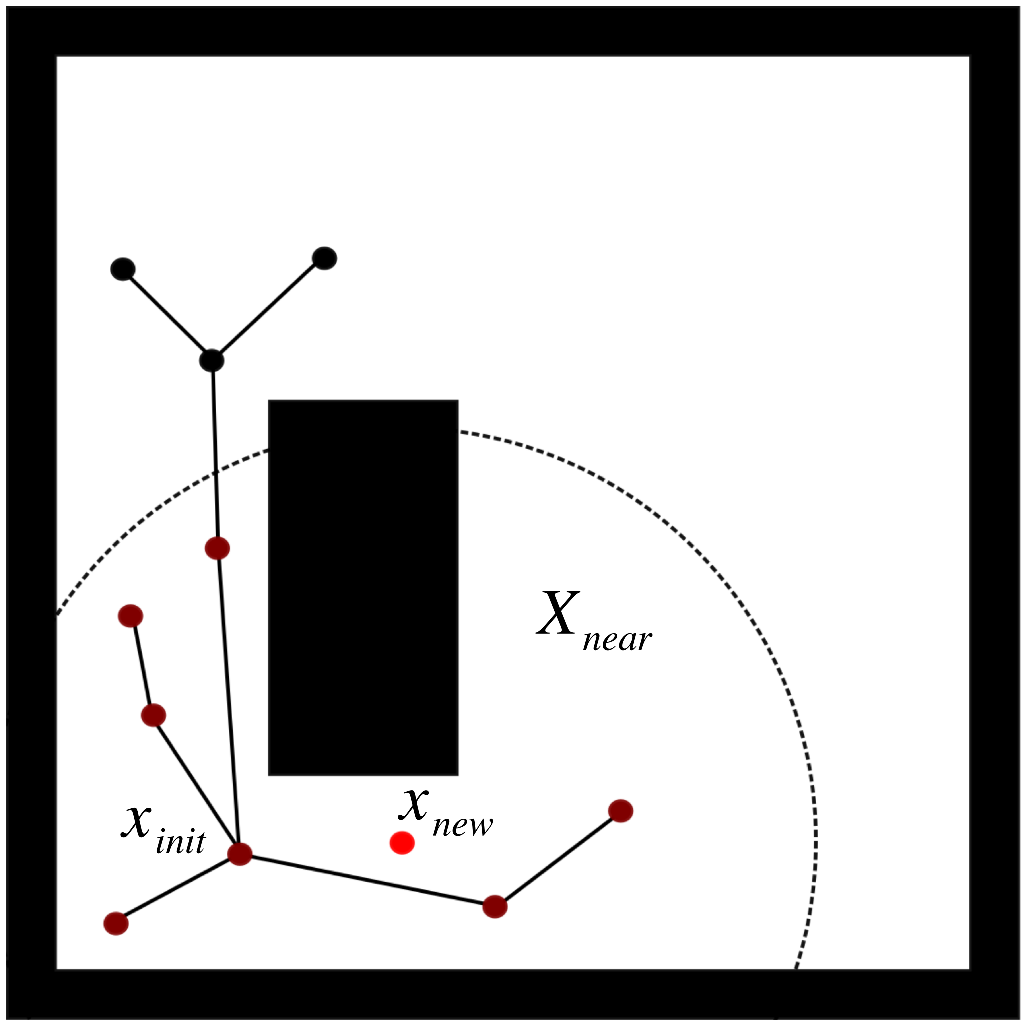
RRT*-New Sample



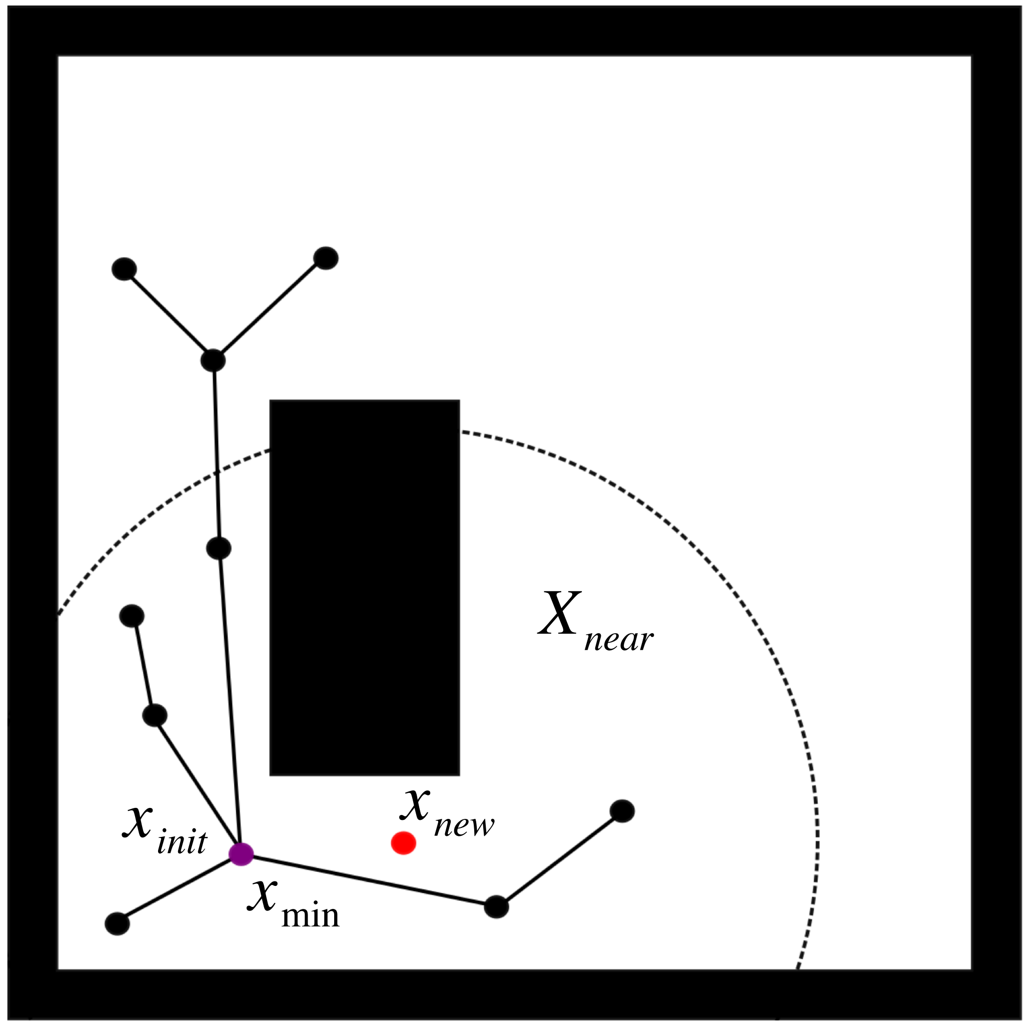
RRT*-New Sample



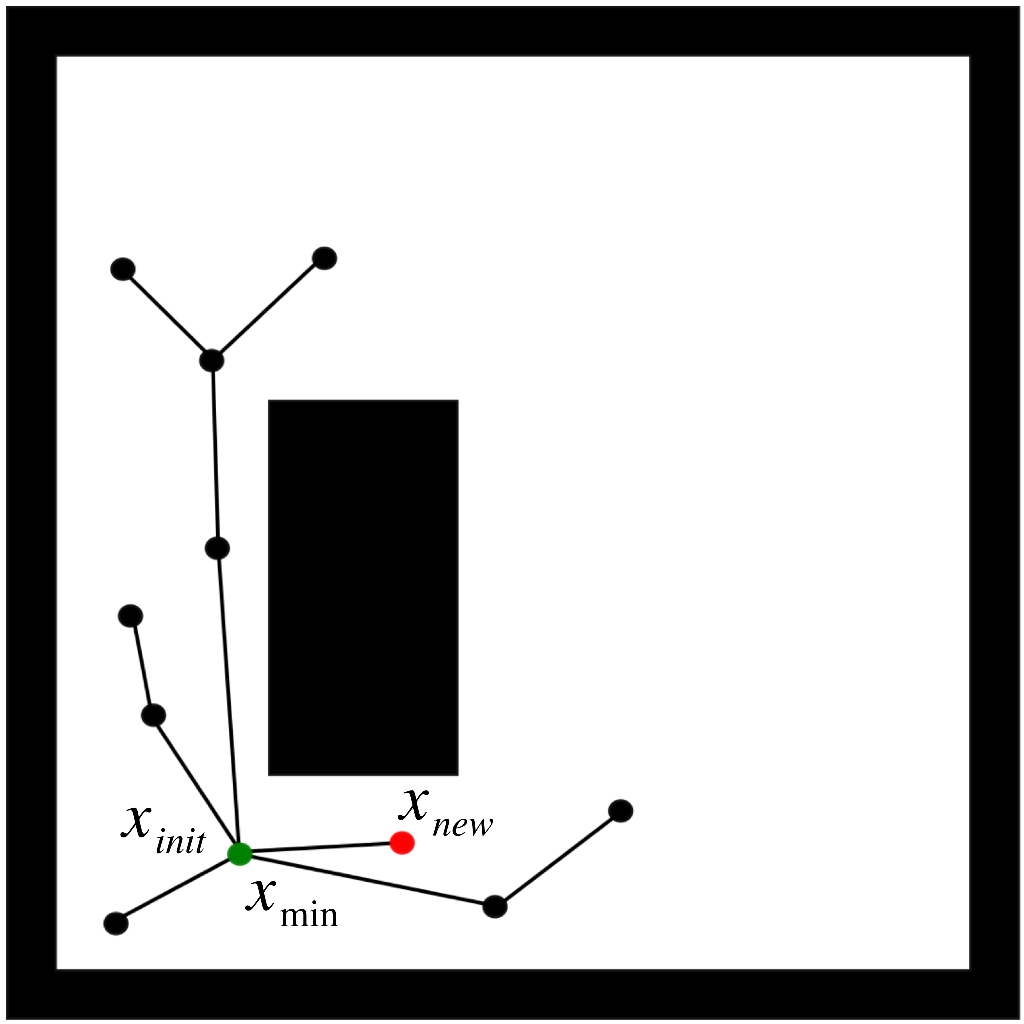
RRT*-Connect Min Cost Path



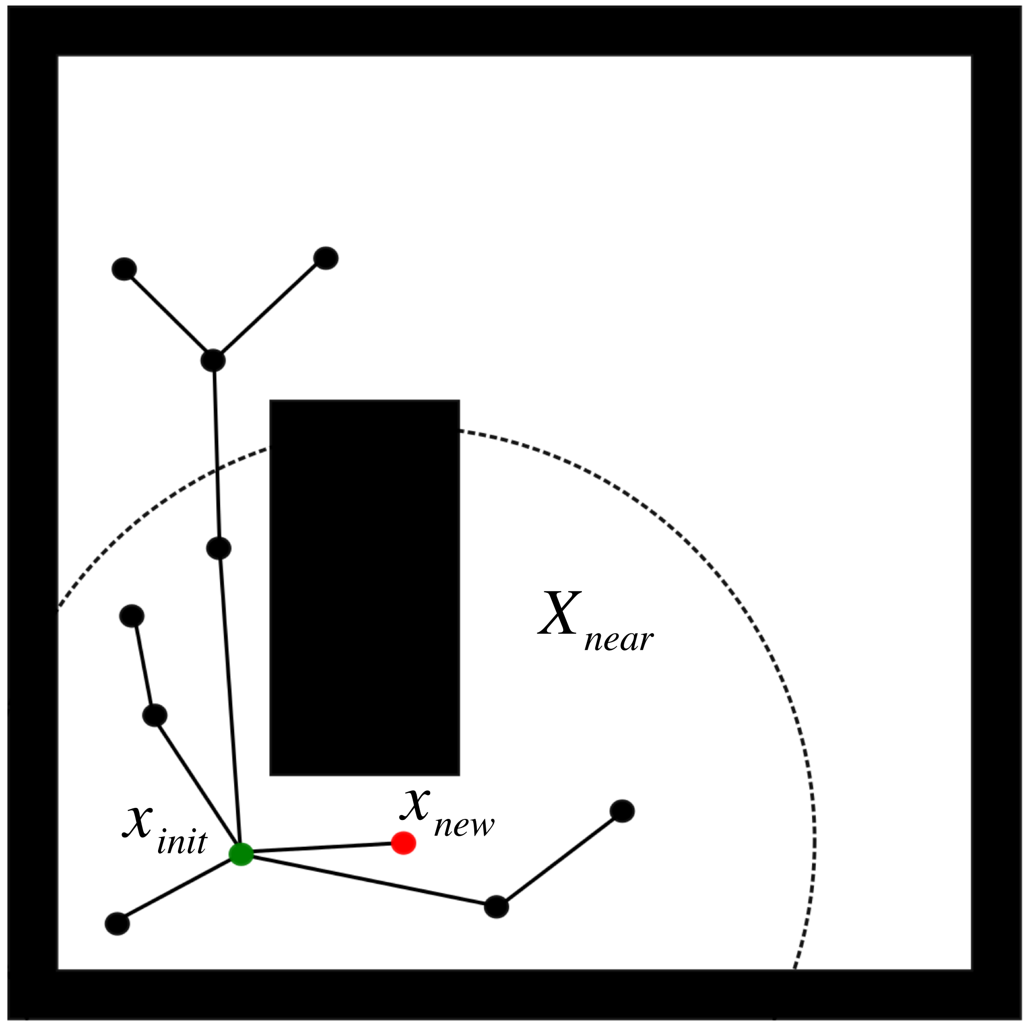
RRT*-Connect Min Cost Path



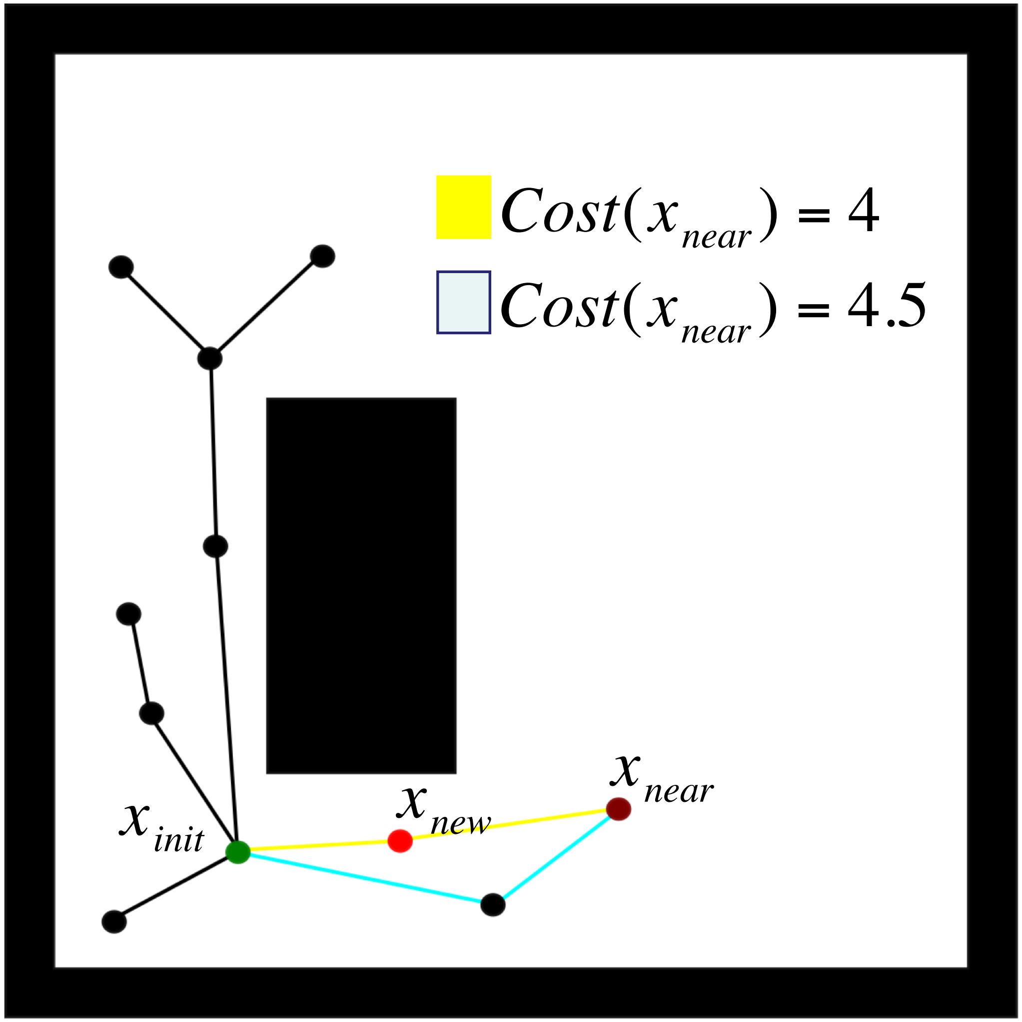
RRT*-Connect Min Cost Path



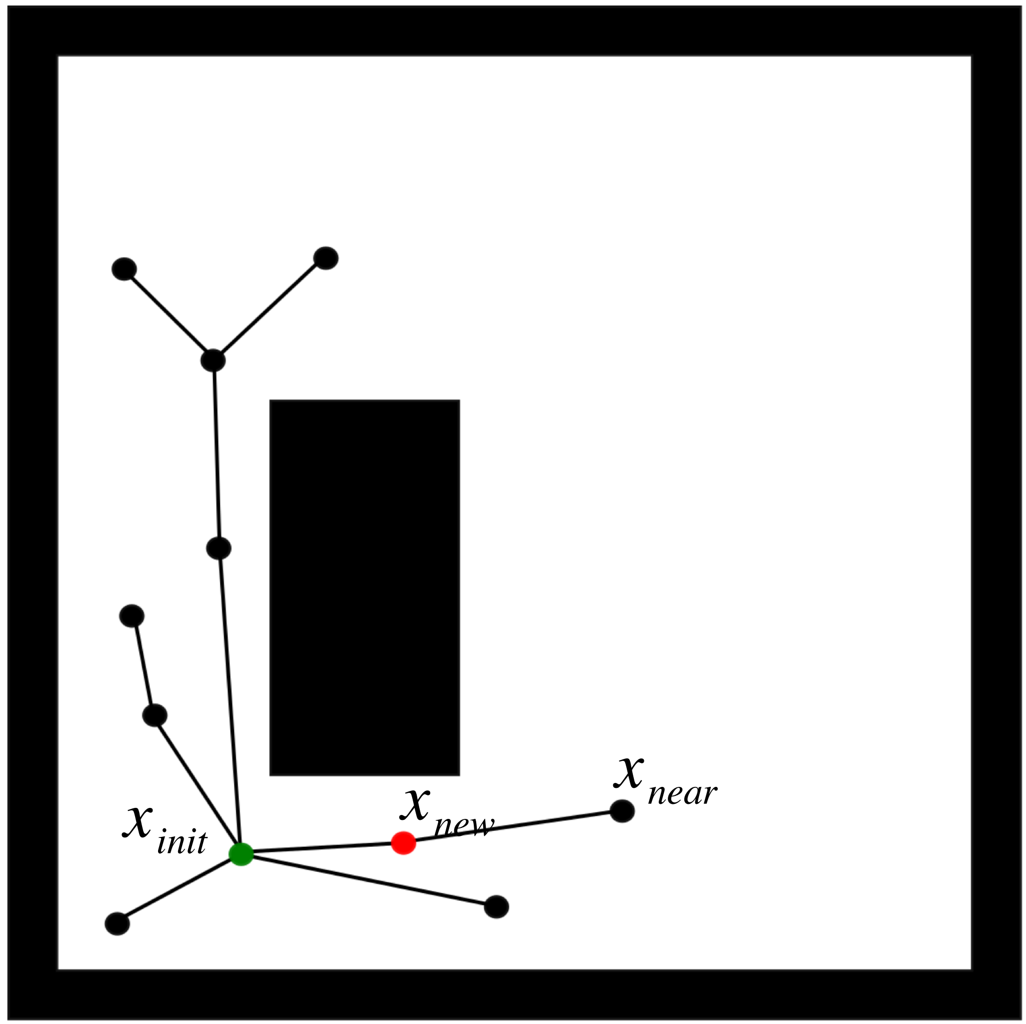
RRT*-Rewire



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 $G = (V, E)$;

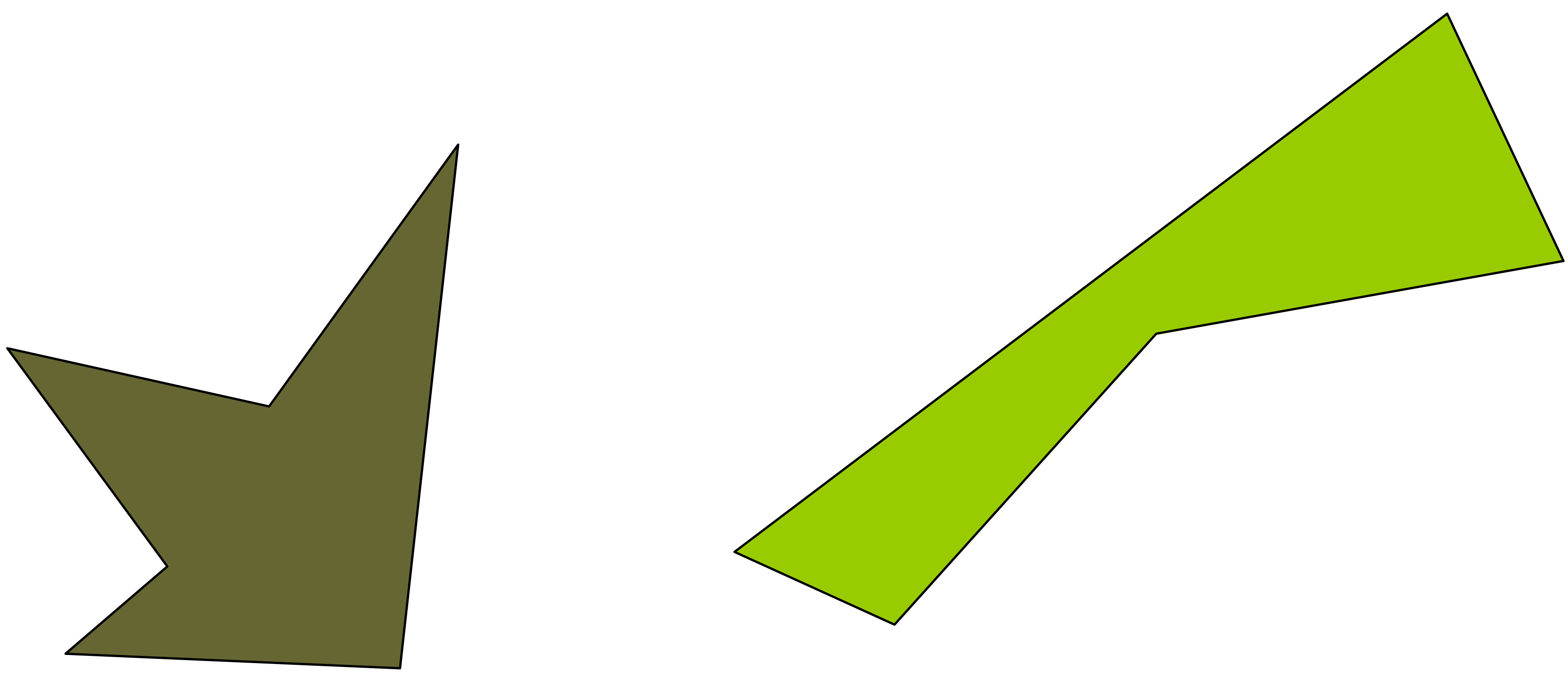
RRT* algorithm

Algorithm 6: RRT*

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

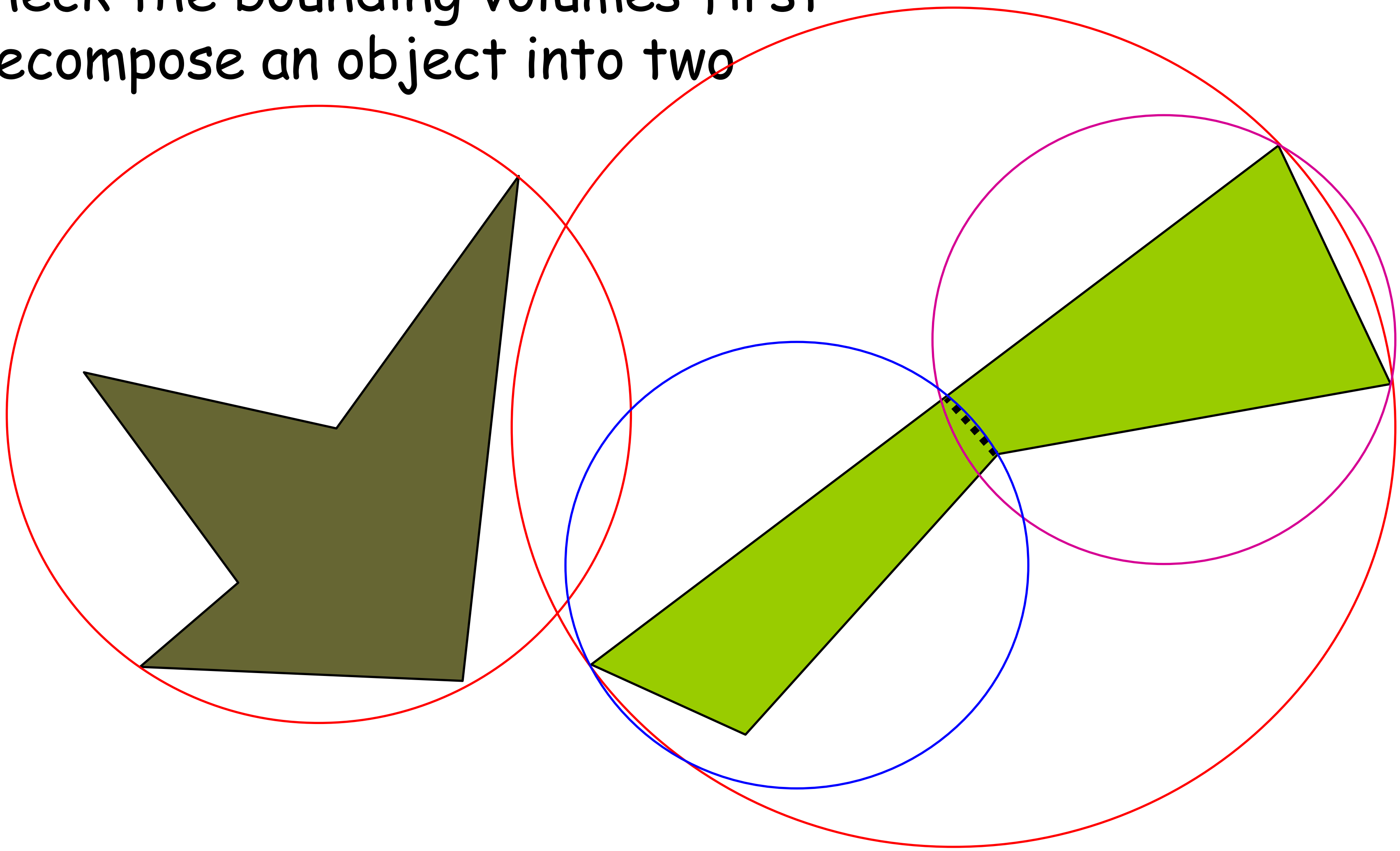
Collision Checking

- 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

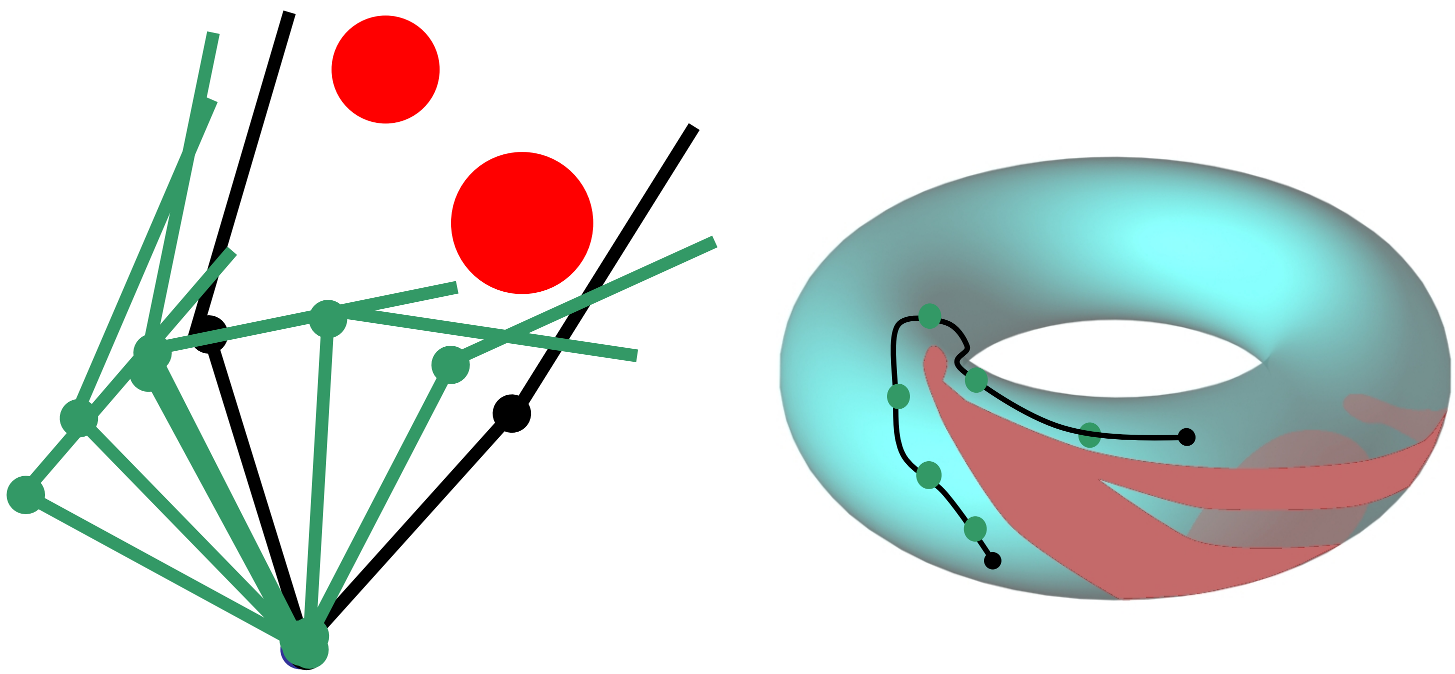


BVH of a 3D Triangulated Cat

Nonholonomic Motion Planning

- Any admissible motion for the **3D mechanical system** appears a collision-free path for a **point** in the CSpace

Configuration Space



- Translating the ***continuous*** problem into a ***combinatorial*** one

- Capturing the ***topology*** of *CSfree* with ***graphs***

- 80’s Configuration Space Approach, Decidability, Deterministic Approaches

  « Robot Motion Planning » Latombe,

  Kluwer Academics Pub., 1991

- 90’s Nonholonomy and Probabilistic Approaches

  « Robot Motion Planning and Control» Laumond,

  Springer Verlag, 1998

  http://www.laas.fr/~jpl (free of charge)

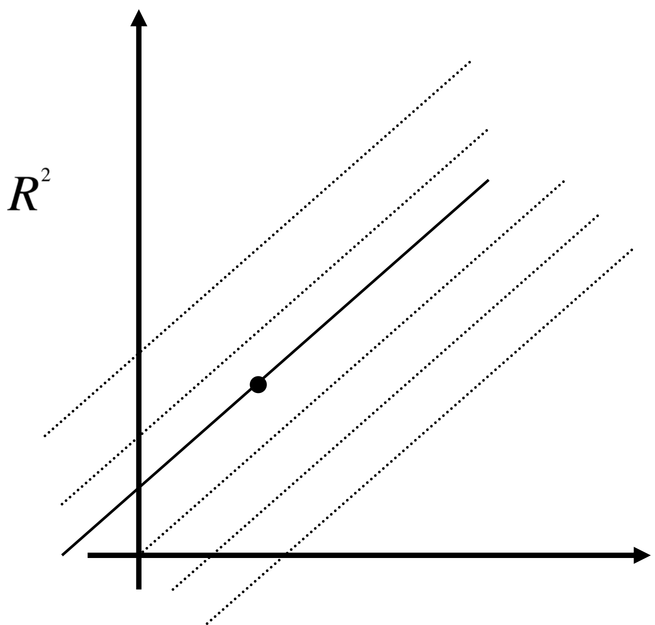
  Choset et al, 2005

  LaValle, 2006

- Any admissible motion for the 3D mechanical system appears a collision-free path for a point in the Cspace

- Holonomic systems: converse is **true**

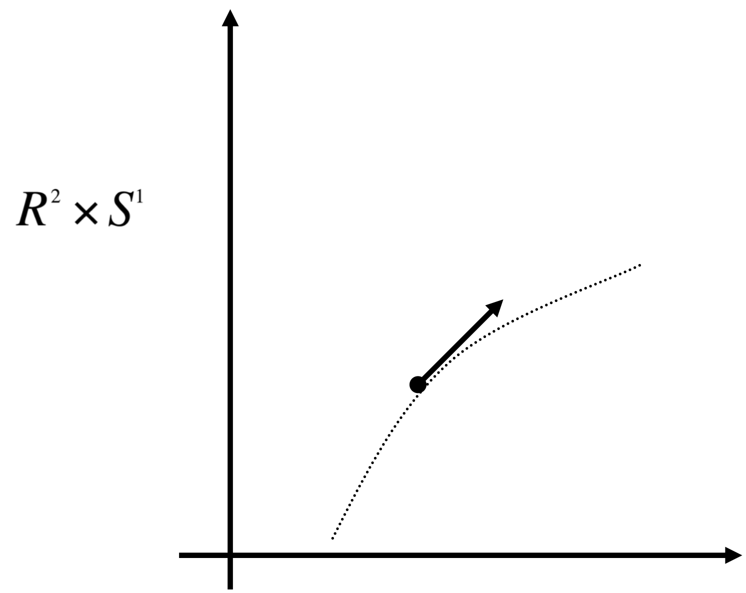
- Nonholonomic ones: converse is **not true**



$\dot{x} = \dot{y}$

Integrable

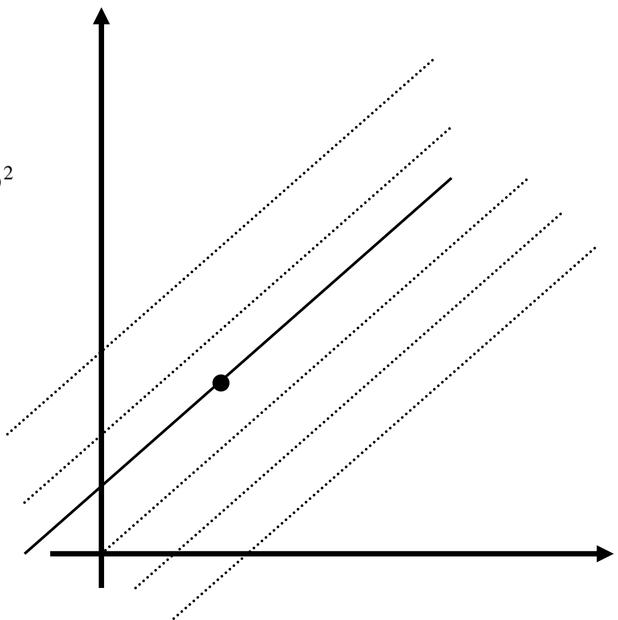
Dim(Reachable(q))=1



$\dot{x}\cos\theta - \dot{y}\sin\theta = 0$

Not integrable

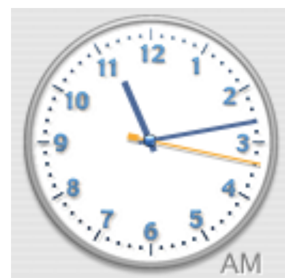
Dim(Reachable(q))=3

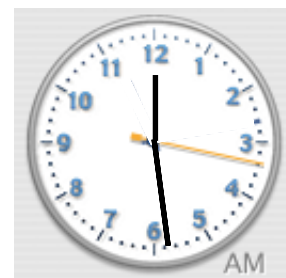


$\dot{x} = \dot{y}$

Integrable

Dim(Reachable(q))=1





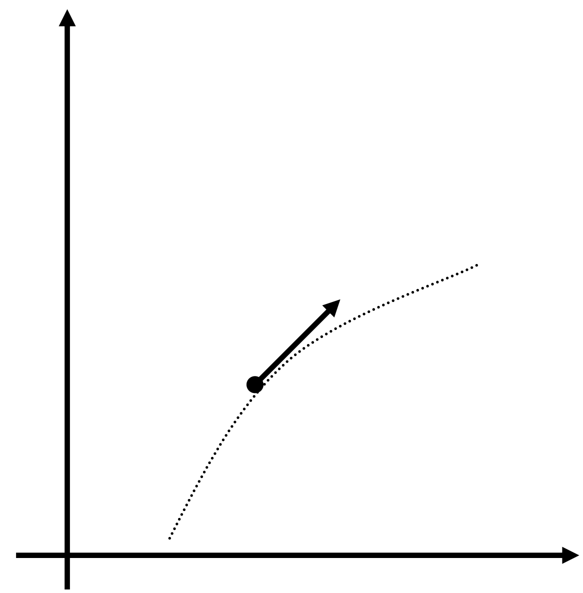
The time that will never happen!!!

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u_2$$

$$\left[\begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{pmatrix}$$

Lie Bracket

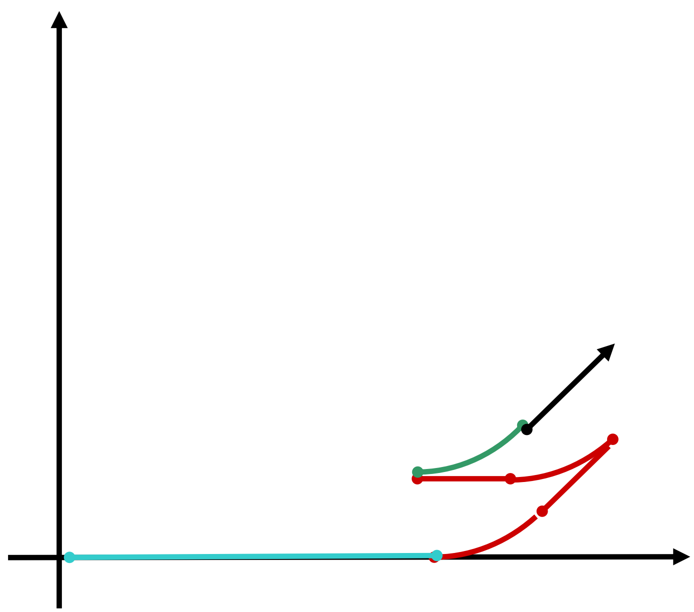
$R^2 \times S^1$



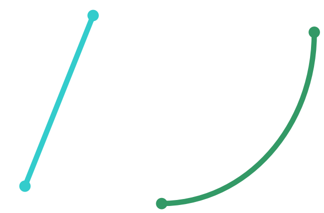
$\dot{x}\cos\theta - \dot{y}\sin\theta = 0$

Not integrable

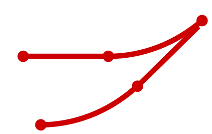
Dim(Reachable(q))=3

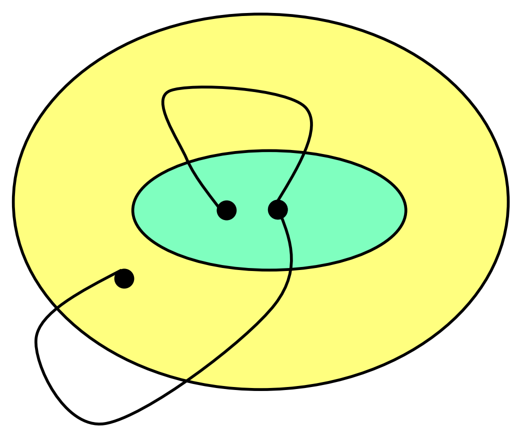


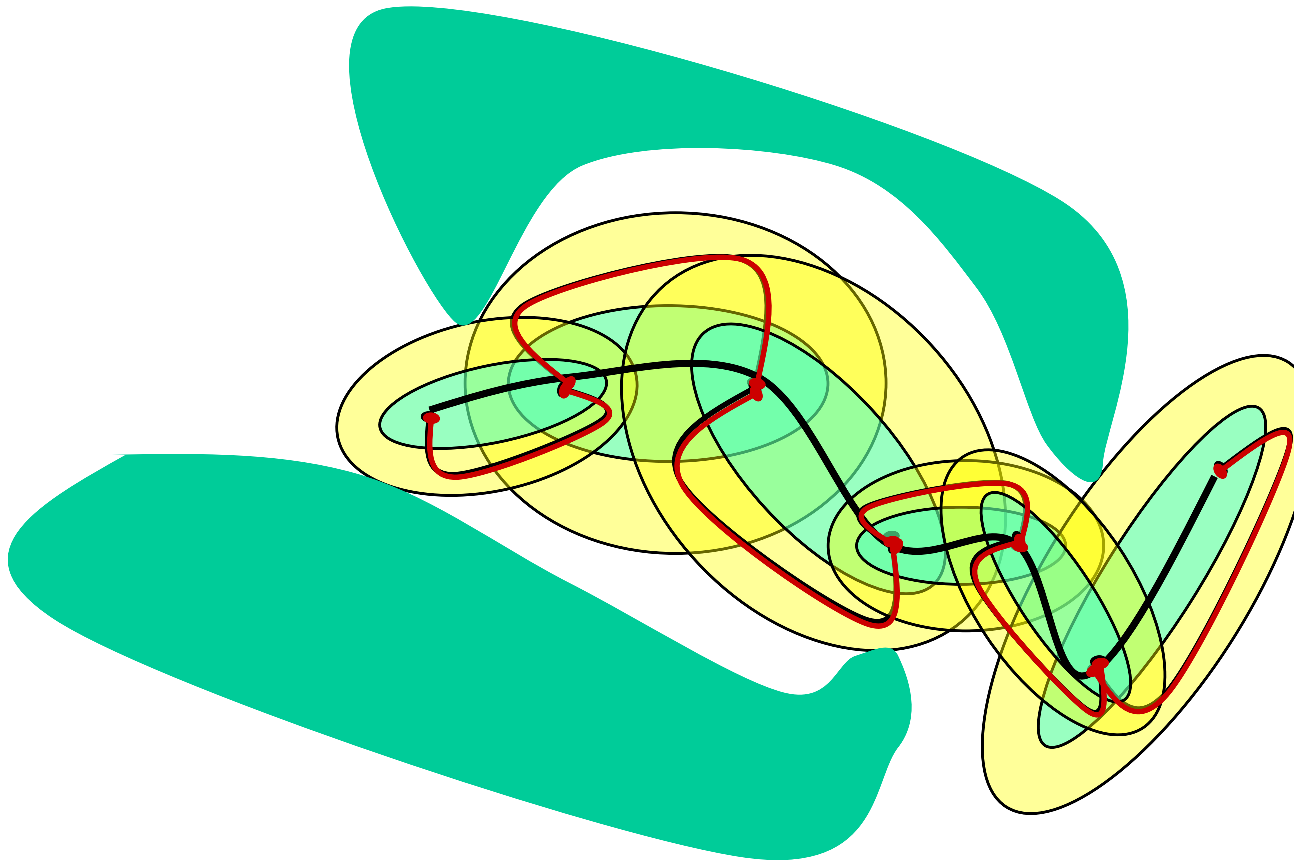
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} u + \begin{pmatrix} \cos\theta \\ \sin\theta \\ 1 \end{pmatrix} v$$

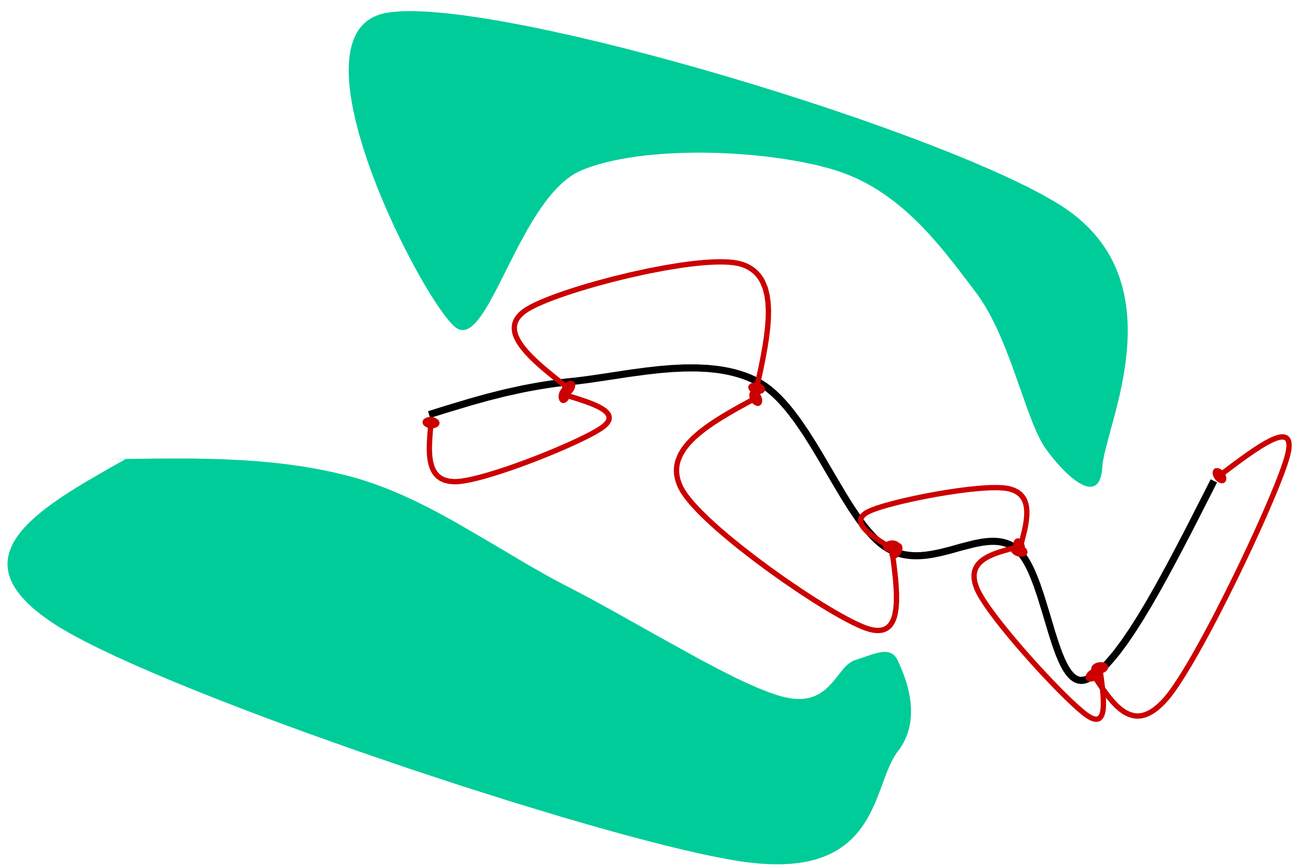


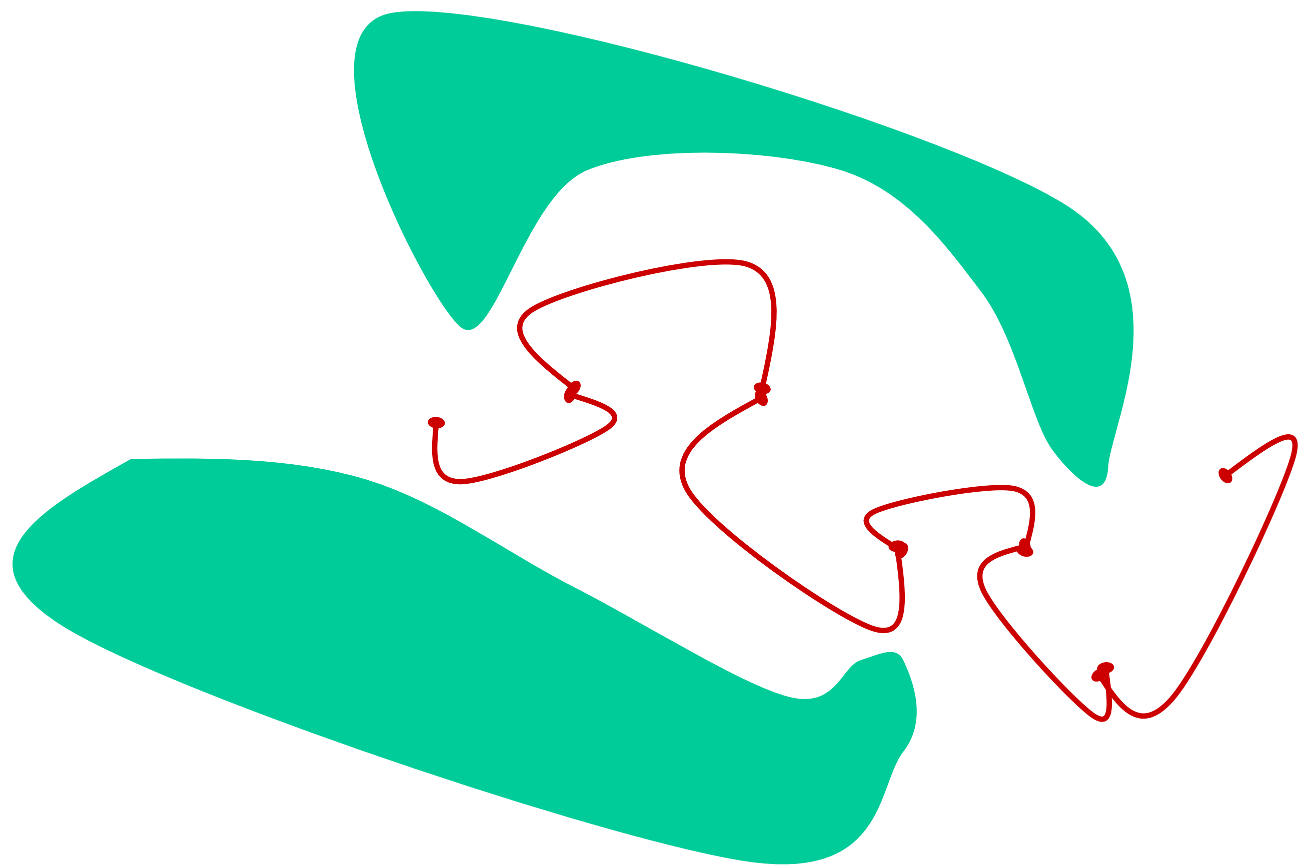
$$\left[\begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix}, \begin{pmatrix} \cos\theta \\ \sin\theta \\ 1 \end{pmatrix} \right]$$











- For small-time controllable systems the existence of an admissible motion is characterized by the existence of a path lying in an *open* connected component of free-CSpace

- Is my system nonholonomic ?
- Is my nonholonomic system small-time controllable ?

- Is my system nonholonomic ?

  `Frobenius Theorem`

- Is my nonholonomic system small-time controllable ?

  `Lie Algebra Rank Condition`

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.