

# Final Review

CSCI 545 Introduction to Robotics  
Instructor: Stefanos Nikolaidis

# Linear Programming

- Consider the Linear Program

$$\max z = x_1 + 2x_2$$

*s.t.*

$$x_1 \leq 3$$

$$x_1 + x_2 \leq 5$$

$$x_1, x_2 \geq 0$$

# Linear Programming

- Consider the Linear Program

$$\max z = x_1 + 2x_2$$

*s.t.*

$$x_1 \leq 3$$

$$x_1 + x_2 \leq 5$$

$$x_1, x_2 \geq 0$$

- Illustrate graphically:
  - The hyperplanes defined by the constraints
  - The gradient of the cost function
  - The feasible region
  - The optimal solution

# Linear Programming

$x_2$

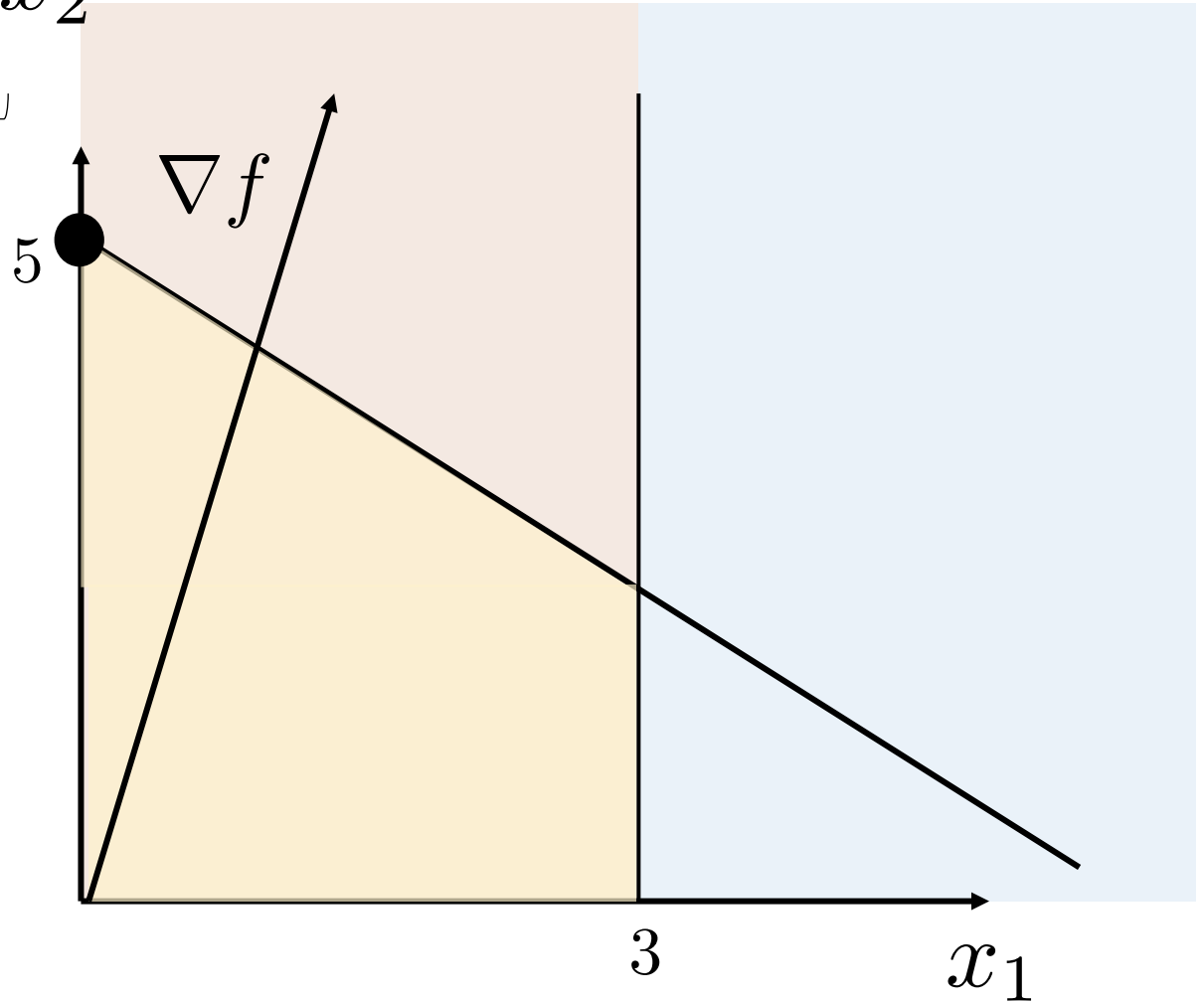
$$\max z = x_1 + 2x_2$$

$$s.t. \quad f(x_1, x_2)$$

$$x_1 \leq 3$$

$$x_1 + x_2 \leq 5$$

$$x_1, x_2 \geq 0$$



# Configuration Spaces

- Match the following robots and configuration spaces

1. Mobile Robot Translating on a plane  $SE(2) \times T^2$
2. Mobile robot translating and rotating on a plane  $SE(2)$  or  $\mathbb{R}^2 \times S^1$
3. A spacecraft  $\mathbb{R}^2$
4. An n-joint revolute arm  $T^n$
5. A mobile planar robot with a two-joint arm  $SE(3)$  or  $\mathbb{R}^3 \times SO(3)$

# Configuration Spaces

• Match the following robots with their configuration spaces

1. Mobile Robot Translating on a plane  $\rightarrow SE(2) \times T^2$
  2. Mobile robot translating and rotating on a plane  $\rightarrow SE(2)$  or  $\mathbb{R}^2 \times S^1$
  3. A spacecraft  $\rightarrow \mathbb{R}^2$
  4. An n-joint revolute arm  $\rightarrow T^n$
  5. A mobile planar robot with a two-joint arm  $\rightarrow SE(3)$  or  $\mathbb{R}^3 \times SO(3)$
-

# Forward / Inverse Kinematics

- Formally define:
  1. A forward kinematics map for a planar robot

# Forward / Inverse Kinematics

- Formally define:

1. A forward kinematics map for a planar robot

$$\phi : Q \rightarrow SE(2)$$

$$\text{Phi} : Q \rightarrow SE(2)$$

2. An inverse kinematics map for a planar robot:

# Forward / Inverse Kinematics

- Formally define:

1. A forward kinematics map for a planar robot

$$\phi : Q \rightarrow SE(2)$$

2. An inverse kinematics map for a planar robot:

$$\phi^{-1} : SE(2) \rightarrow Q$$

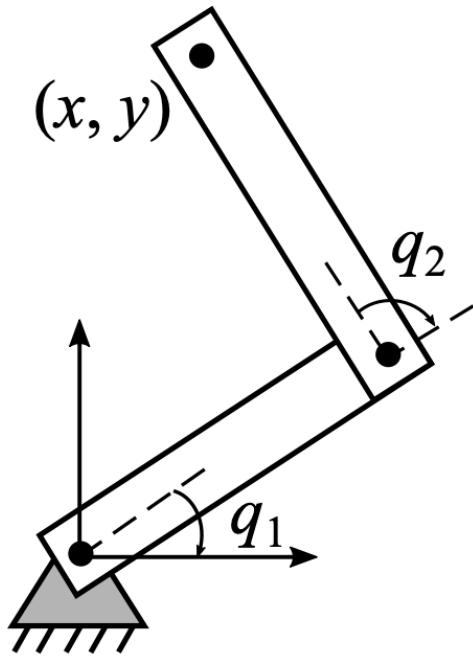
$$\text{Phi}^{-1} : SE(2) \rightarrow Q$$

(or  $\phi^{-1} : \mathbb{R}^2 \rightarrow Q$  )

$$\text{Phi}^{-1} : \mathbb{R}^2 \rightarrow Q$$

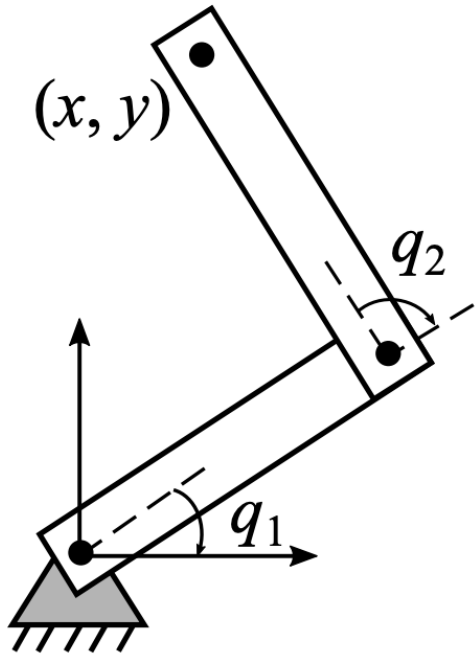
# Forward / Inverse Kinematics

- Compute the position  $(x, y)$  of the end-effector for the planar arm:



# Forward / Inverse Kinematics

- Compute the position  $(x, y)$  of the end-effector for the planar arm:

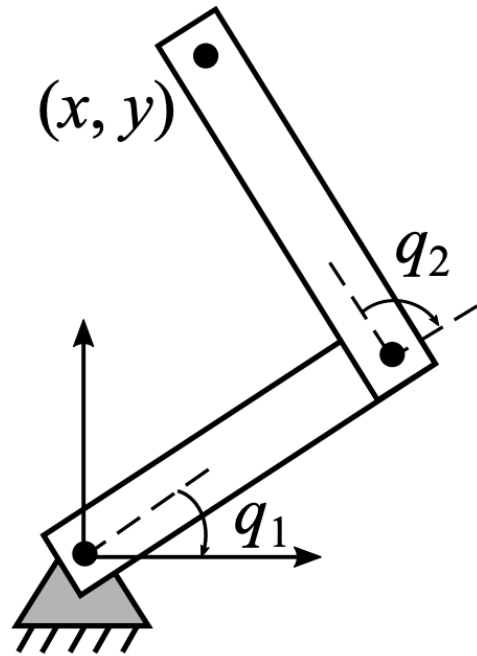


$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{bmatrix} \\ &= \phi(q_1, q_2) \\ &= \begin{bmatrix} \phi_1(q_1, q_2) \\ \phi_2(q_1, q_2) \end{bmatrix} \end{aligned}$$

$$\begin{aligned} x &= l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ y &= l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{aligned}$$

# Forward / Inverse Kinematics

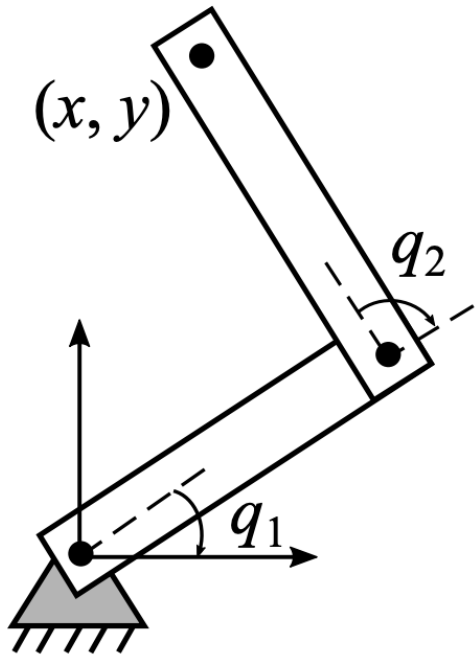
- Compute the velocity of the end-effector for the planar arm:



$$l_1 = l_2 = 1, q_1 = \frac{\pi}{4}, q_2 = \frac{\pi}{2}$$

$$\dot{q}_1 = 1, \dot{q}_2 = 0$$

# Example: 2-R Planar Manipulator



$$\begin{aligned}\dot{p} &= \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \\ &= J\dot{q} \\ &= \begin{bmatrix} -\sqrt{2} & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}\end{aligned}$$

$$\begin{bmatrix} -\sqrt{2} & -1/\sqrt{2} \\ 0 & -1/\sqrt{2} \end{bmatrix} = [1;0]$$

# Kinematic Singularities

- What are Kinematic Singularities? How can we identify them? What is their effect?

# Kinematic Singularities

- What are Kinematic Singularities? How can we identify them? What is their effect?
  - The Jacobian drops rank, mobility of the structure is reduced, small velocities of the end-effector may result in large joint velocities

# IK Computation as Optimization Problem

- General Formulation of IK problem of a planar robot with  $n$  joints, each with lower and upper limits  $\bar{l}_i, u_i$

# IK Computation as Optimization Problem

- General Formulation of IK problem of a planar robot with  $n$  joints, each with lower and upper limits  $l_i, u_i$

$$\begin{aligned} & \underset{\mathbf{q}}{\text{minimize}} && \|FK(\mathbf{q}) - \mathbf{p}_d\|^2 \\ & \text{subject to} && l_i \leq q_i \leq u_i, i = \{1, \dots, n\}. \end{aligned}$$

# IK Computation as Optimization Problem

- General Formulation of IK problem of a planar robot with  $n$  joints, each with lower and upper limits  $l_i, u_i$

$$\begin{aligned} & \underset{\mathbf{q}}{\text{minimize}} && \|FK(\mathbf{q}) - \mathbf{p}_d\|^2 \\ & \text{subject to} && l_i \leq q_i \leq u_i, i = \{1, \dots, n\}. \end{aligned}$$

- What if there are obstacles?

# IK Computation as Optimization Problem

- General formulation of IK problem of a planar robot with  $n$  joints, each with lower and upper limits  $l_i, u_i$

$$\begin{aligned} & \underset{\mathbf{q}}{\text{minimize}} && \|FK(\mathbf{q}) - \mathbf{p}_d\|^2 \\ & \text{subject to} && l_i \leq q_i \leq u_i, i = \{1, \dots, n\}. \end{aligned}$$

- What if there are obstacles?

Add obstacle collision constraints!

# IK Computation as Optimization Problem

- General formulation of IK problem of a planar robot with  $n$  joints, each with lower and upper limits  $l_i, u_i$

$$\begin{aligned} & \underset{\mathbf{q}}{\text{minimize}} && \|FK(\mathbf{q}) - \mathbf{p}_d\|^2 \\ & \text{subject to} && l_i \leq q_i \leq u_i, i = \{1, \dots, n\}. \end{aligned}$$

- What if there are obstacles?

Add obstacle collision constraints!

- Does the starting solution matter?

# IK Computation as Optimization Problem

- General formulation of IK problem of a planar robot with  $n$  joints, each with lower and upper limits  $l_i, u_i$

$$\begin{aligned} & \underset{\mathbf{q}}{\text{minimize}} && \|FK(\mathbf{q}) - \mathbf{p}_d\|^2 \\ & \text{subject to} && l_i \leq q_i \leq u_i, i = \{1, \dots, n\}. \end{aligned}$$

- What if there are obstacles?

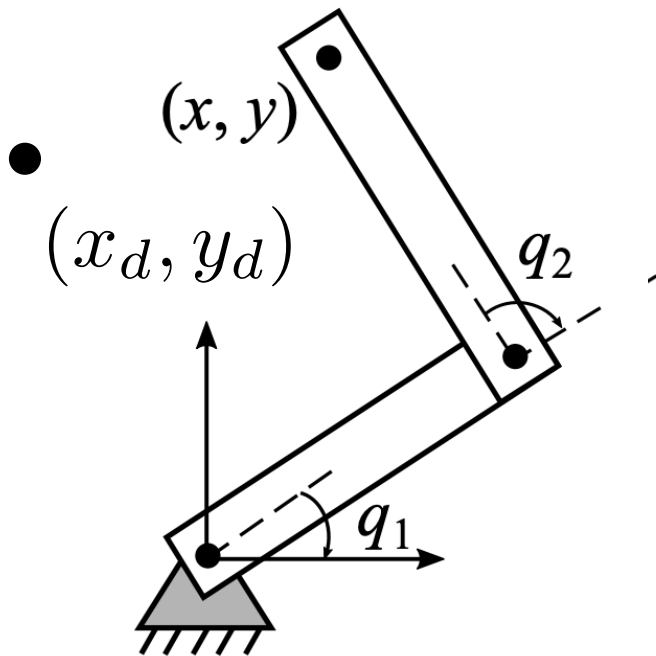
Add obstacle collision constraints!

- Does the starting solution matter?

# Inverse Kinematics as Constrained Optimization Problem

$$F = \frac{1}{2} \Delta q^T \Delta q$$

$$\text{s.t: } \Delta p = J \Delta q$$



How to **solve** this optimization problem?

# Inverse Kinematics as Constrained Optimization Problem

$$\Delta q = \underbrace{J^T (J J^T)^{-1}}_{J^\#} \Delta p$$

Jacobian Pseudoinverse

Gradient descent:

$$q_{t+1} = q_t + \alpha J^\# \Delta p$$

$$q_{\{t+1\}} = q_{\_t} + \alpha J^\# \text{Deltap}$$

# Inverse Kinematics as Constrained Optimization Problem

$$\Delta q = \underbrace{J^T (J J^T)^{-1}}_{J^\#} \Delta p$$

Jacobian Pseudoinverse

Gradient descent:

$$q_{t+1} = q_t + \alpha J^\# \Delta p$$

What is the effect of the step size  $\alpha$ ?

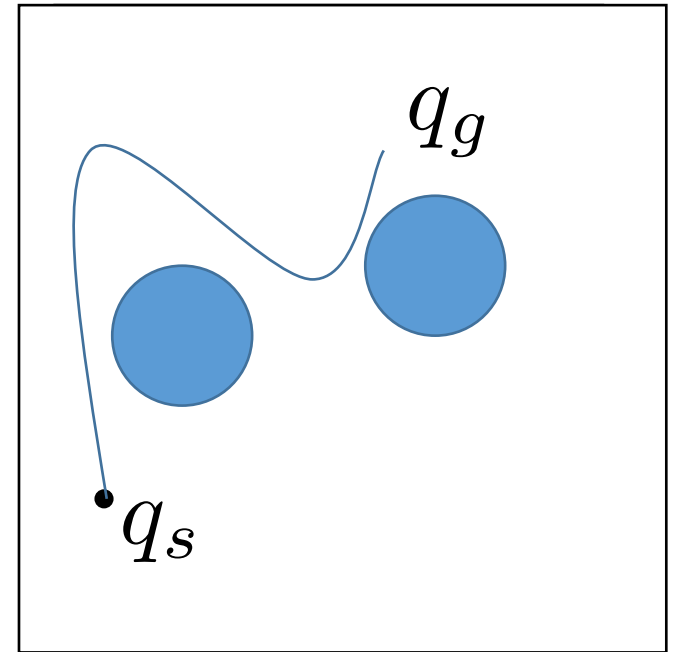
# Motion Planning

- Formulate the motion planning problem:

# Motion Planning

- Formulate the motion planning problem:

1. A world  $W$  in  $\mathbb{R}^2$  or  $\mathbb{R}^3$
2. An obstacle region  $O$  in  $W$
3. A robot  $A$  and its configuration space  $Q$
4.  $Q_{free} = Q \setminus Q_{obs}$
5.  $q_s \in Q_{free}$  : Initial Configuration
6.  $q_g \in Q_{free}$  : Goal Configuration
7. Compute a path  $\tau : [0, 1] \rightarrow Q_{free}$  so that  $q_s = \tau(0)$ ,  $q_g = \tau(1)$



# Single / Multi Query Algorithms

- What are single / multi-query algorithms?
- What are their advantages / disadvantages?

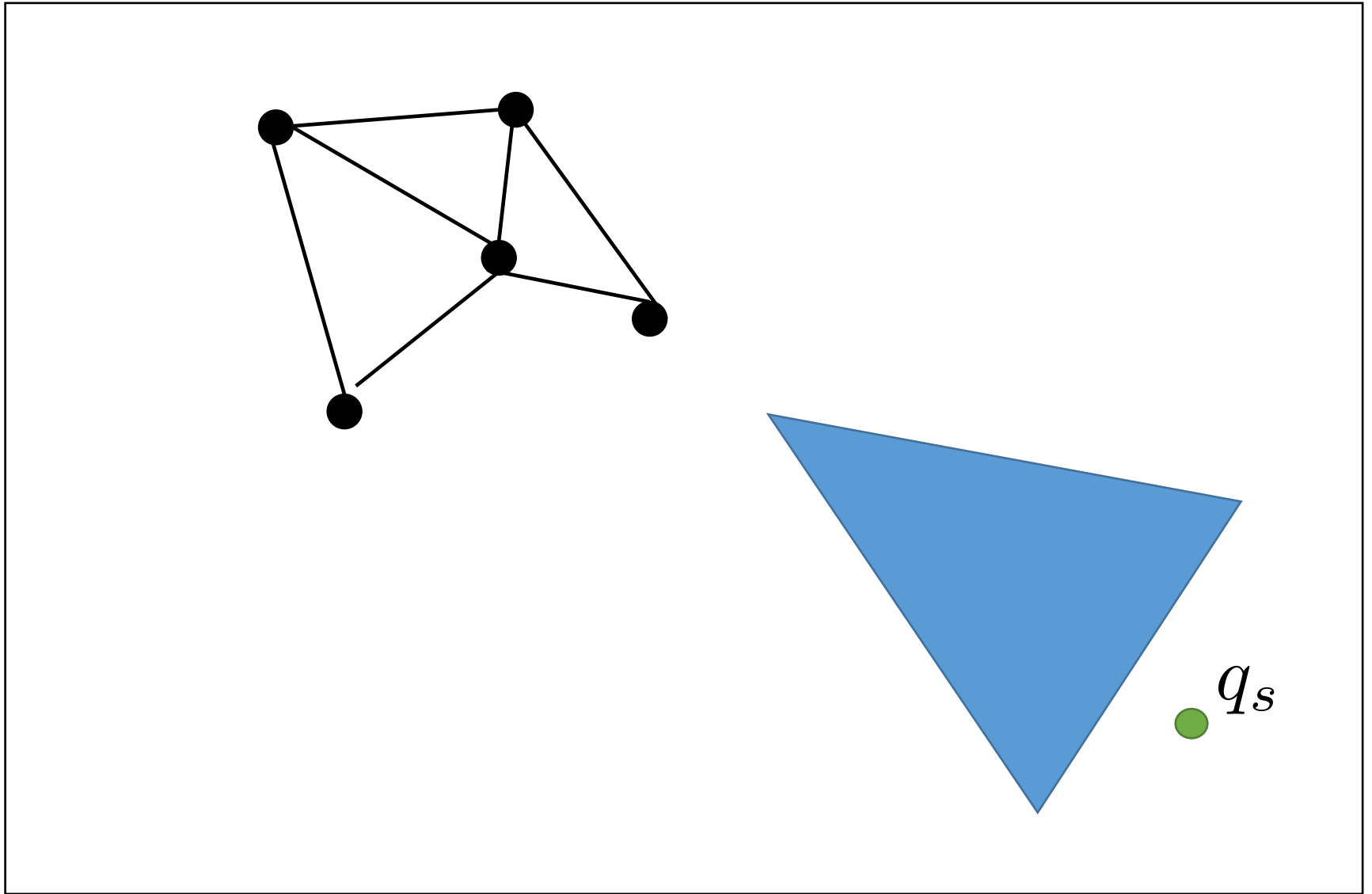
# Single/Multi Query Algorithms

- Multi-Query Planning Algorithms
  - Cache roadmap for multiple queries
  - Environment is static
- Single-Query Planning Algorithms
  - Recompute roadmap for each query
  - Environment can change across queries (not within)

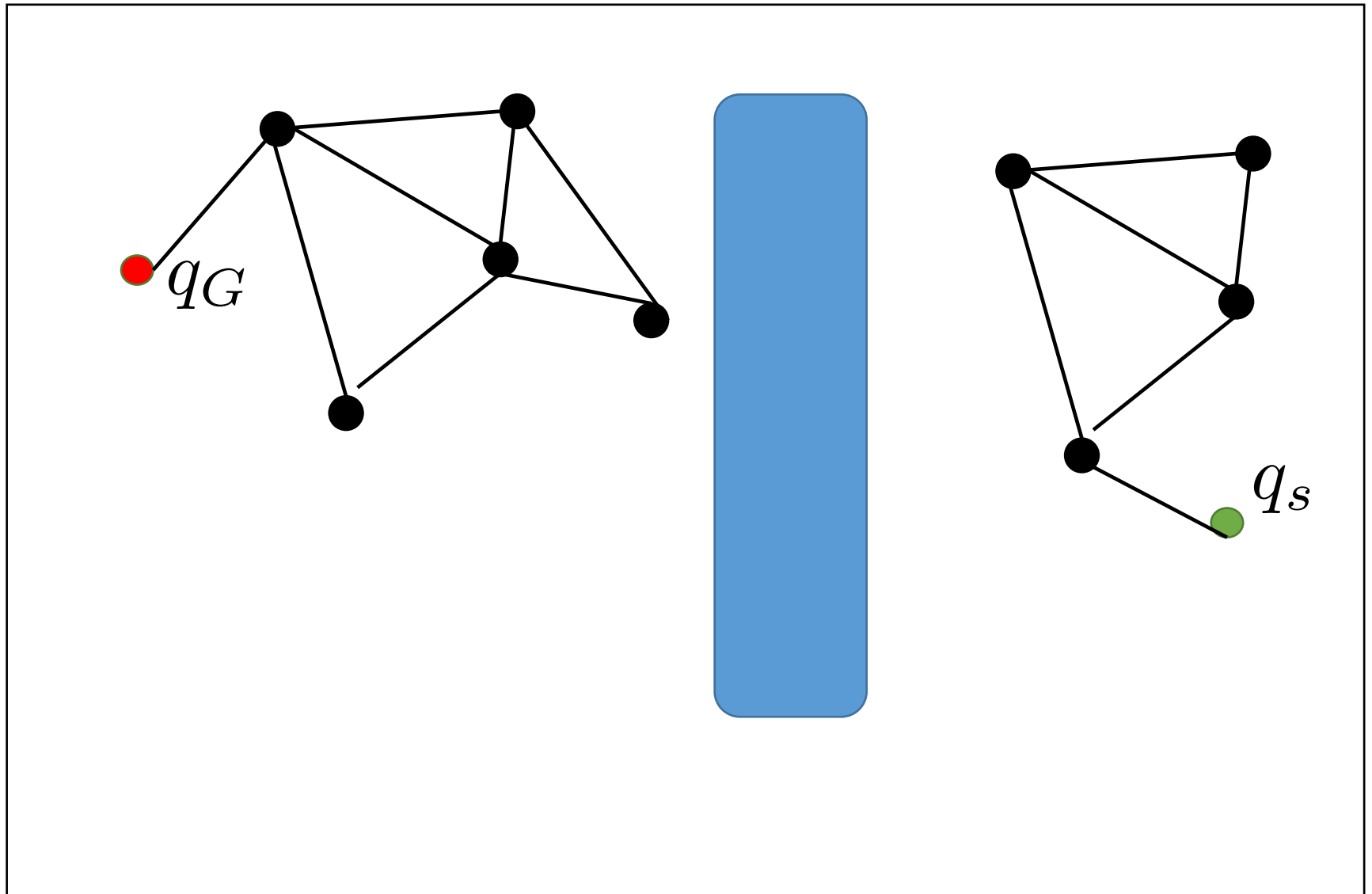
# Probabilistic Roadmaps

- What are common reasons of failure of the PRM algorithms?

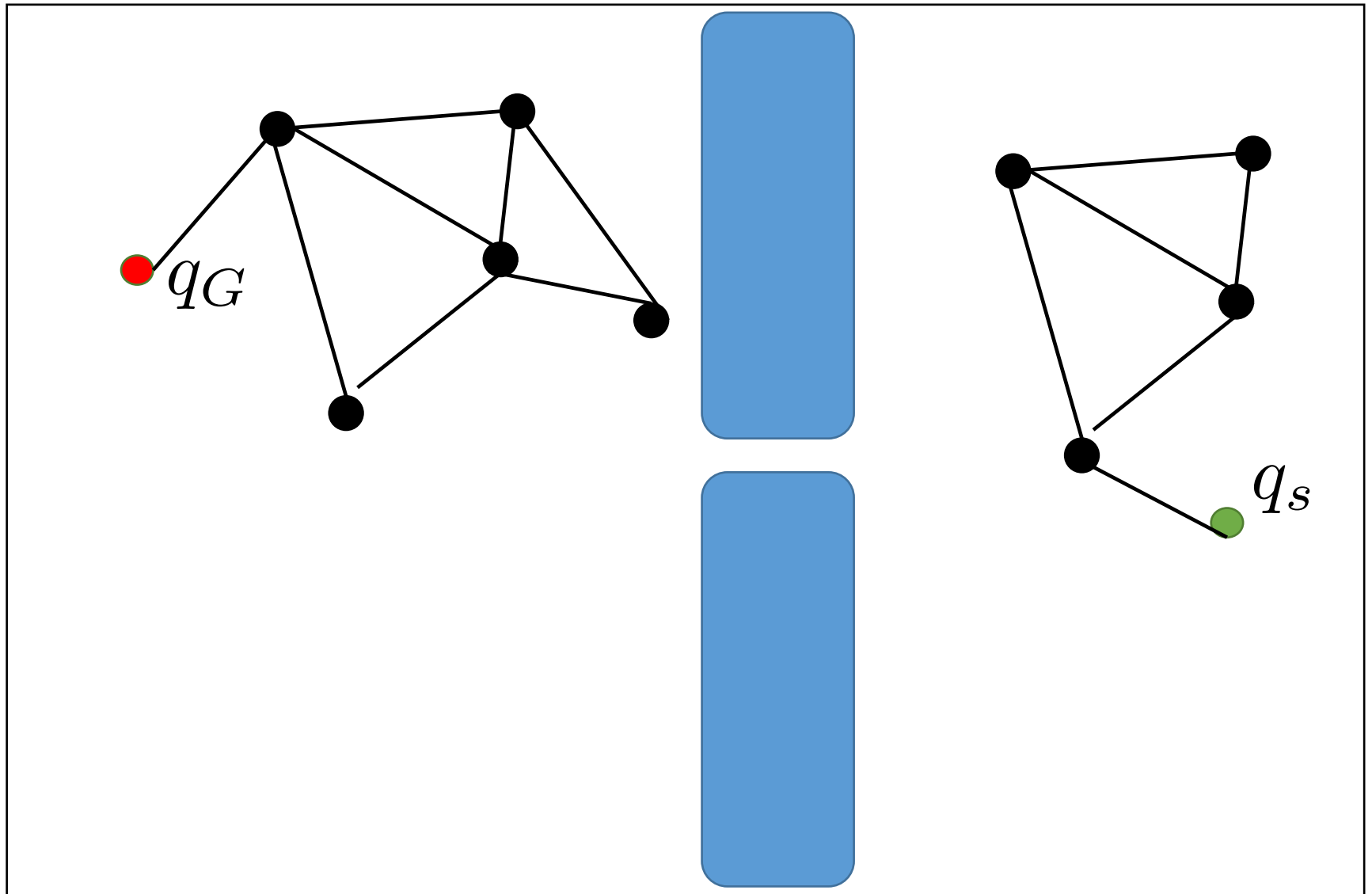
# Limitations: Coverage



# Limitations: Connectivity



# Example: Narrow Passage Problem



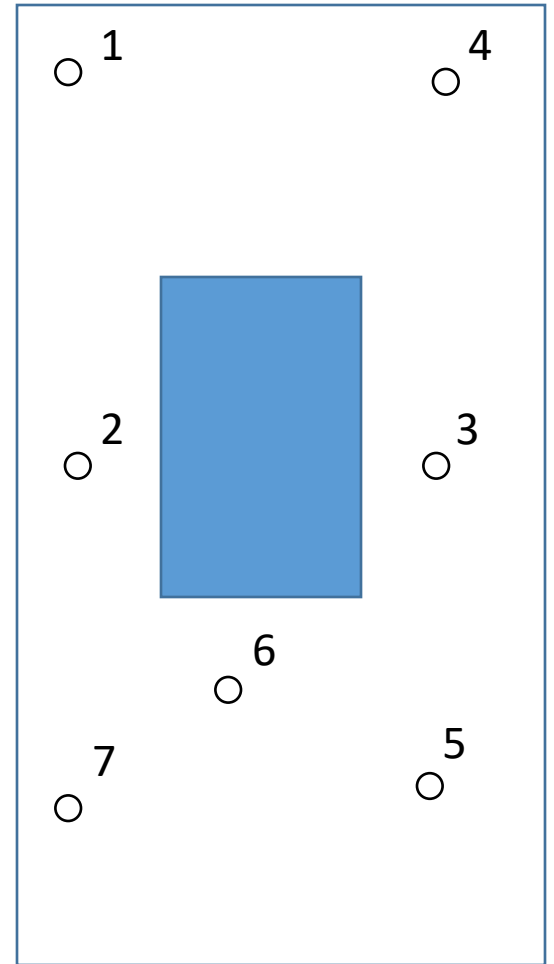
# Narrow Passage Problem

- What are solutions for the narrow passage problem?

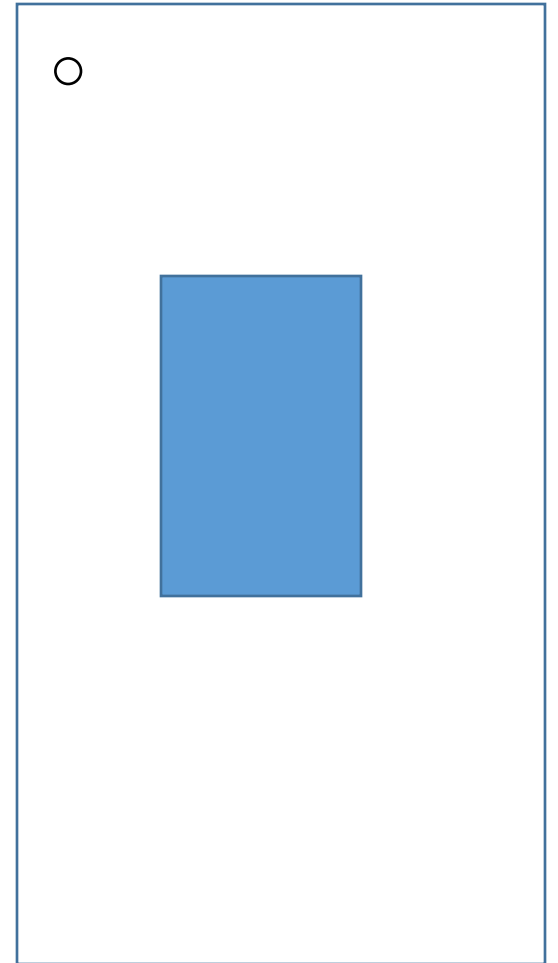
# Narrow Passage Problem

- What are solutions for the narrow passage problem?
  - Sample near obstacles
  - Visibility-based PRM

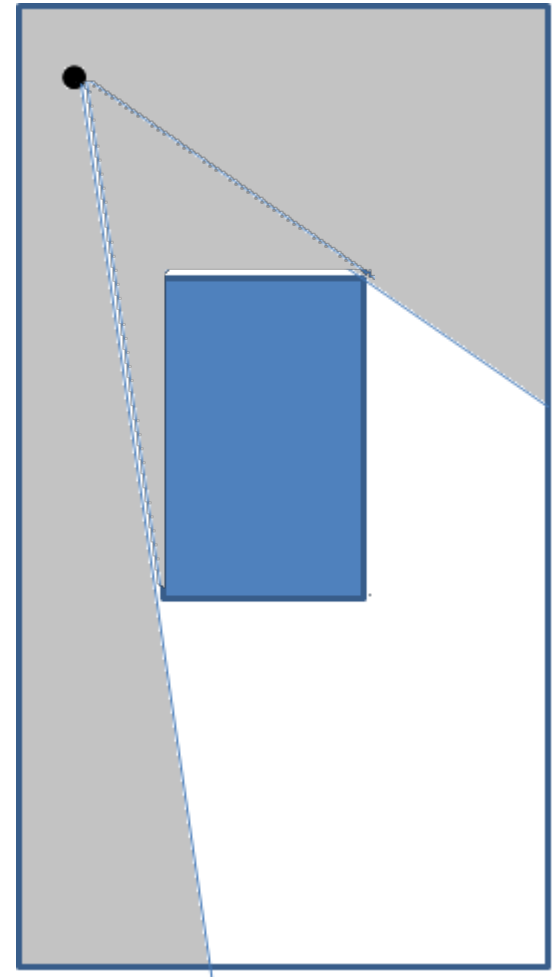
# Visibility-Based PRM



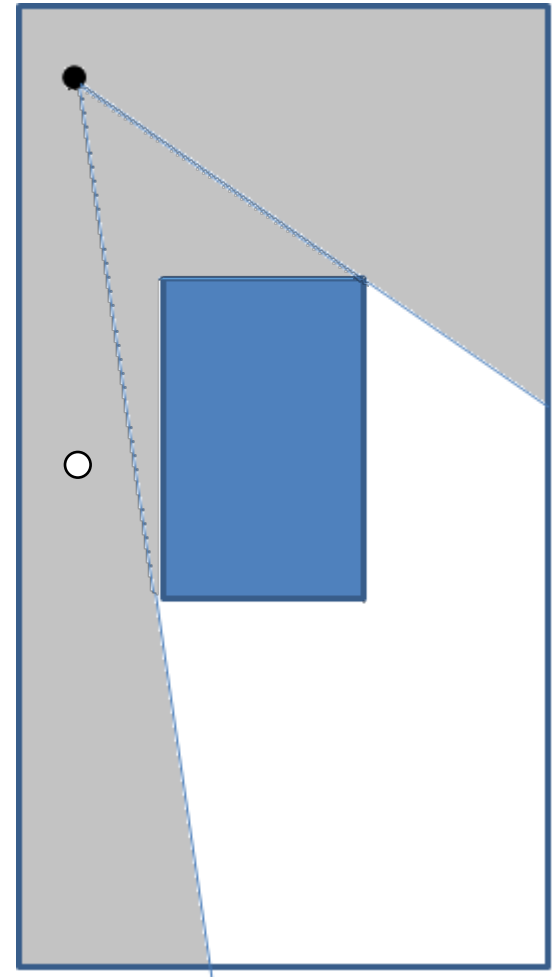
# Visibility-Based PRM



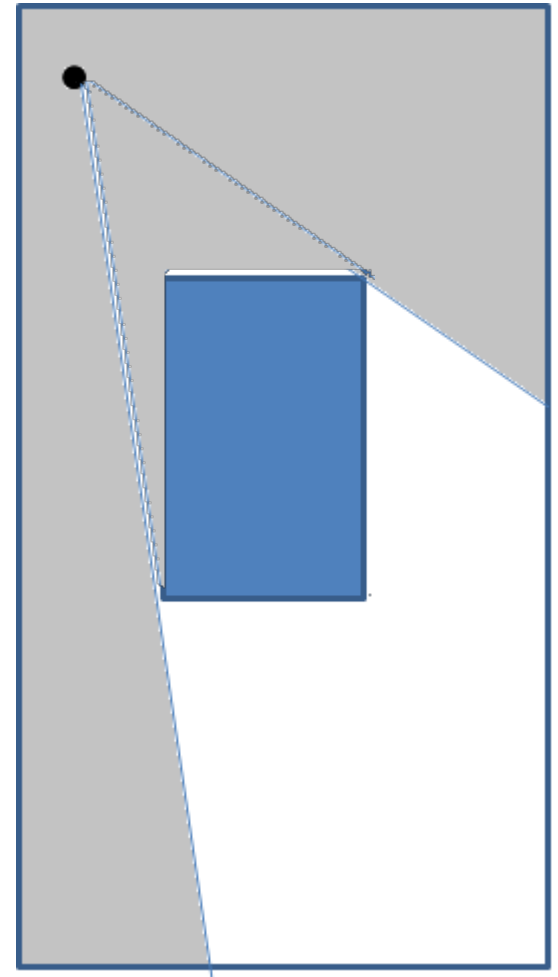
# Visibility-Based PRM



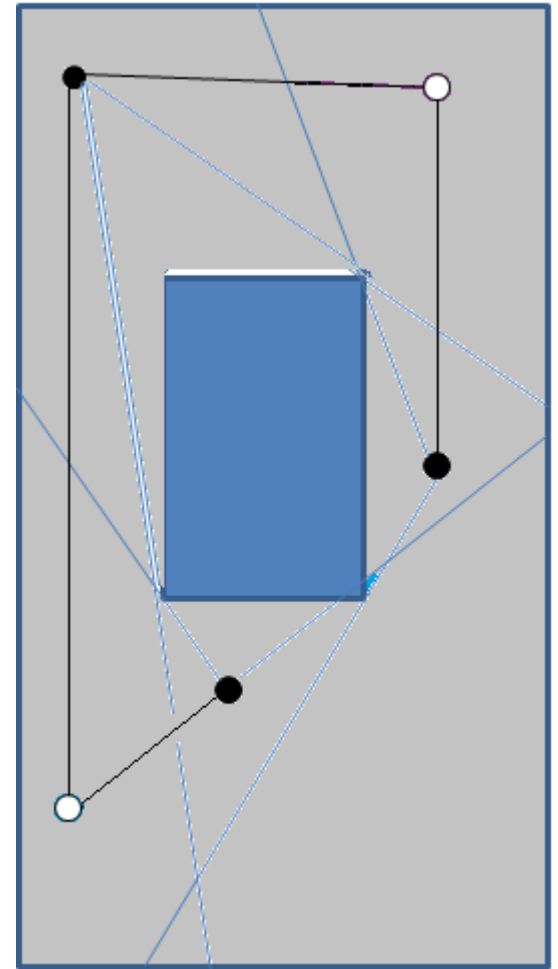
# Visibility-Based PRM



# Visibility-Based PRM



# Visibility-Based PRM



# RRTs

- What is the definition of probabilistic completeness?

# Probabilistic Completeness

- An algorithm  $ALG$  is probabilistically complete if, for any robustly feasible motion planning problem defined by

$$P = (Q_{free}, q_s, q_g)$$

$$\lim_{N \rightarrow \infty} P(ALG \text{ returns a solution to } P) = 1$$

$$\lim (N \rightarrow \text{infty}) P(ALG \text{ returns a solution to } P) = 1$$

# RRTs

- What is the definition of probabilistic completeness?
- Is RRT a probabilistically complete algorithm?

# RRTs

- What is the definition of probabilistic completeness?
- Is RRT a probabilistically complete algorithm?
- Is the path returned by RRT asymptotically optimal?

# RRTs

- One approach that may improve the performance of the RRT algorithm is to sample randomly at the goal and attempt to connect directly to the goal
  - When can this succeed / fail?

# RRTs

- Name 3 factors that affect the performance of RRT algorithm

# RRTs

- Name 3 factors that affect the performance of RRT algorithm
  - Bounds of configuration space
  - Dimensionality of configuration space
  - Obstacles

# RRTs

- How can you make the trajectory of an RRT less jerky?

# RRTs

- How can you make the trajectory of an RRT less jerky?
  - Shortcutting

# Shortcutting

- Output of RRTs is almost unusable.
- Simple Shortcutting Algorithm:
  - Repeat:
    - Select two points randomly
    - Attempt to connect them
    - If path shortest than previous one, replace path

# Task Space Regions

- When planning in Task space regions, how would you project samples to the constraint manifold? Write the algorithm in pseudocode

# Task Space Regions

- When planning in Task space regions, how would you project samples to the constraint manifold? Write the algorithm in pseudocode

---

**Algorithm 1:  $\mathbf{J}^+$  Projection( $q$ )**

---

```
1 while true do  
2    $\Delta \mathbf{x} \leftarrow \text{DisplacementFromTSR}(q)$ ;  
3   if  $\|\Delta \mathbf{x}\| < \epsilon$  then  
4     return  $q$ ;  
5   end  
6    $\mathbf{J} \leftarrow \text{GetJacobian}(q)$ ;  
7    $\Delta q_{error} \leftarrow \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1} \Delta \mathbf{x}$ ;  
8    $q \leftarrow (q - \Delta q_{error})$ ;  
9 end
```

---

# Task Space Regions

- When planning in Task space regions, how would you project samples to the constraint manifold? Write the algorithm in pseudocode

1. while true do:
2.  $\Delta x \leftarrow \text{DisplacementfromTSR}(q)$
3. if  $\|\Delta x\| < \epsilon$  then:
4.   return  $q$ ;
5. end
6.  $J \leftarrow \text{GetJacobian}(q)$ ;
7.  $\Delta q_{\text{error}} \leftarrow J^T (JJ^T)^{-1} \Delta x$
8.  $q \leftarrow (q - \Delta q_{\text{error}}$
9. end

# Task Space Regions

- Specify the matrix of bounds  $B^w$  for a cup

# Task Space Regions

- Specify the matrix of bounds  $B^w$  for a cup



$$B^w = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\alpha & \alpha \\ 0 & 0 \\ 0 & 0 \\ -\pi & \pi \end{bmatrix}$$

$$B = [0,0;0,0;0,0;-a,a;0,0;0,0;-pi,pi]$$

# Task Space Regions

- Specify the matrix of bounds  $B^w$  for a cup, so that only poses on a semicircle around the cup (e.g., facing the robot) are sampled



# Task Space Regions

- Specify the matrix of bounds  $B^w$  for a cup, so that only poses on a semicircle around the cup (e.g., facing the robot) are sampled



$$B^w = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\alpha & \alpha \\ 0 & 0 \\ 0 & 0 \\ -\pi/2 & \pi/2 \end{bmatrix}$$

# Task Space Regions

- Specify the matrix of bounds  $B^w$  for a cup, so that only one **unique** pose is sampled



# Task Space Regions

- Specify the matrix of bounds  $B^w$  for a cup, so that only one **unique** pose is sampled



$$B^w = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \alpha & \alpha \\ 0 & 0 \\ 0 & 0 \\ \beta & \beta \end{bmatrix}$$

# Dynamics

- What is the physical meaning behind each term in the dynamics equation?

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

# Dynamics

- What is the forward dynamics problem? Why is it useful?
- What is the inverse dynamics problem?

# Forward Dynamics

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

We can determine  $\ddot{q}$  given  $\tau$

$$\ddot{q} = B(q)^{-1} (\tau - C(q, \dot{q})\dot{q} - G(q))$$

$$\begin{aligned} \ddot{q} &= B(q)^{-1} (\tau - C(q, \dot{q})\dot{q} \\ &\quad - G(q)) \end{aligned}$$

# Inverse Dynamics

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau$$

Given.  $\ddot{q}$ ,  $\dot{q}$ ,  $q$  compute the torques.

# Control

What is decentralized control? When can it fail?

- Independent PD control for each joint, fails when we have significant coupling terms

# Control

Will PD control work well for moving a robotic manipulator to a desired position with gravity? Propose two solutions

# Control

Will PD control work well for moving a robotic manipulator to a desired position with gravity? Propose two solutions

- PID
- PD with Gravity Compensation (feedback linearization)

# Control

Is the system stable when we apply PD with Gravity Compensation? Justify your answer

**Upcoming at Spring 2024**

**CSCI 641: Computational Human-Robot Interaction**