CSCI/ECEN 3302
Introduction to Robotics

Alessandro Roncone

aroncone@colorado.edu https://hiro-group.ronc.one

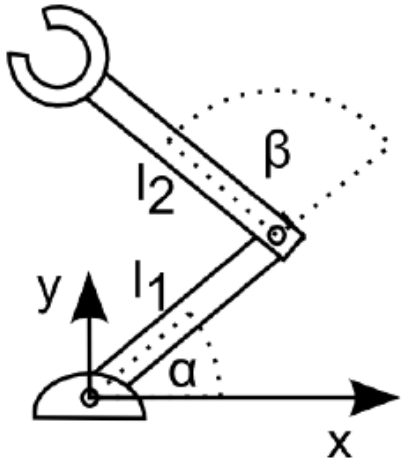
Administrivia

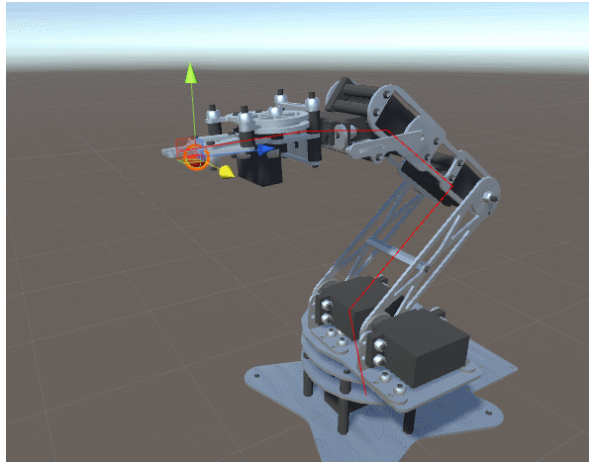
- Lab 2 and HW1 – deadline has passed
- Tomorrow (Wednesday):
  - Full hands-on Lab 3
- Lab 4 will be 1-week long!

Questions?

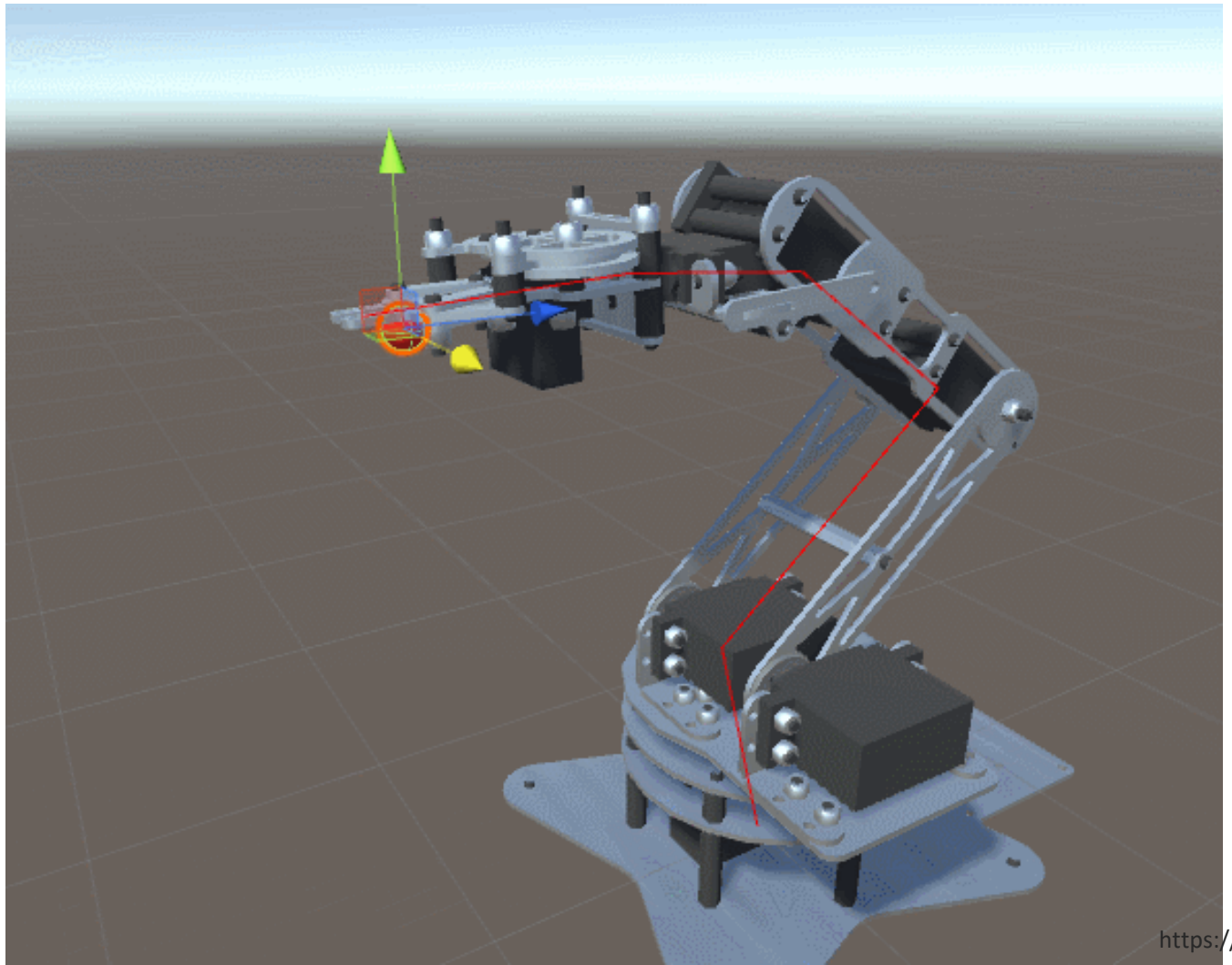
Section 3.3

RECAP
Inverse Kinematics



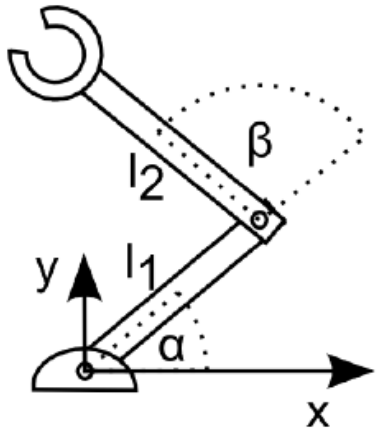


End-effector Position Control

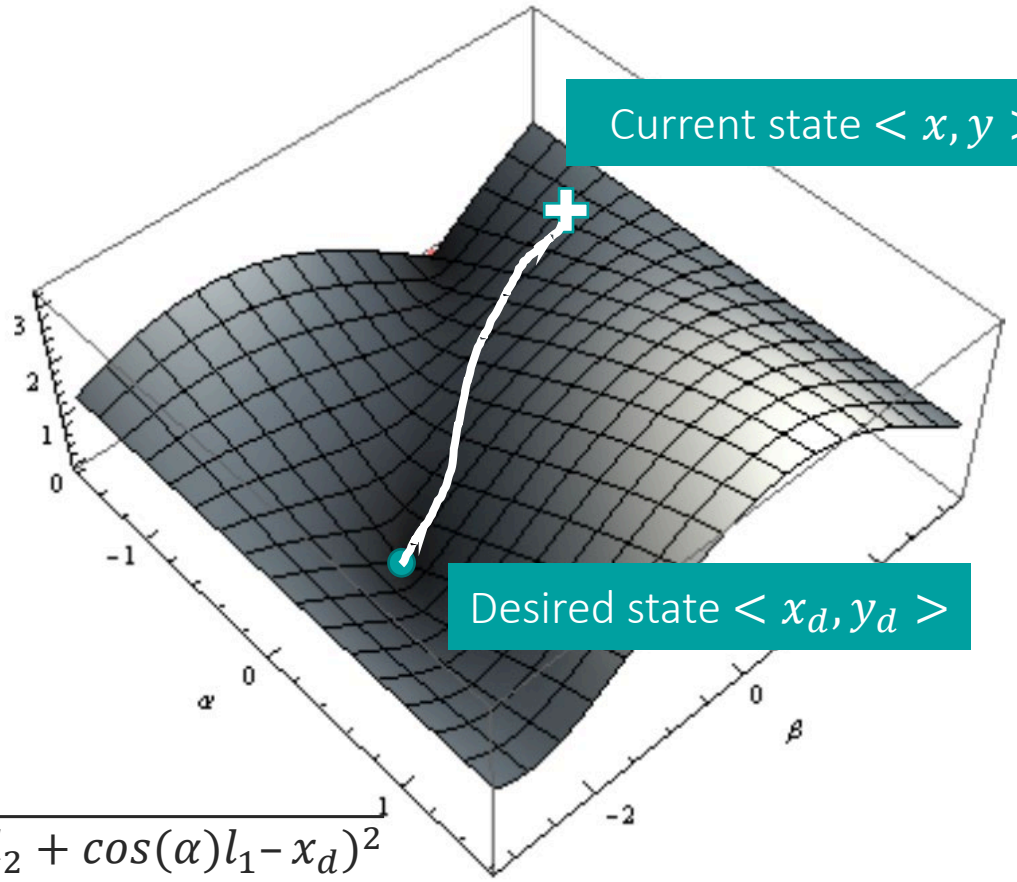


https://www.alanzucconi.com/2017/04/10/robotic-arms/

Motion Planning in EE Space



$$x = l_1 \cos(\alpha) + l_2 \cos(\alpha + \beta)$$
$$y = l_1 \sin(\alpha) + l_2 \sin(\alpha + \beta)$$



Just the Euclidean distance between two vectors!

$$f_{x,y}(\alpha, \beta) = \sqrt{(\sin(\alpha + \beta) l_2 + \sin(\alpha) l_1 - y_d)^2 + (\cos(\alpha + \beta) l_2 + cos(\alpha) l_1 - x_d)^2}$$

Optimization-based Solutions

When we don't have an analytical solution available,
optimization-based methods provide a "best-effort" solution

Optimization methods tend to require very little
knowledge about the parameter space or domain

Therefore, they are general algorithms, underpinning deep learning
and many other popular machine learning methods

Gradient Descent for Solving IK

Given a "distance-from-goal" function f and motors $\alpha_0, \alpha_1, \alpha_2$:

$$\nabla f(\alpha_0, \alpha_1, \alpha_2) = [\nabla f_{\alpha_0}(\alpha_0, \alpha_1, \alpha_2), \nabla f_{\alpha_1}(\alpha_0, \alpha_1, \alpha_2), \nabla f_{\alpha_2}(\alpha_0, \alpha_1, \alpha_2)]$$

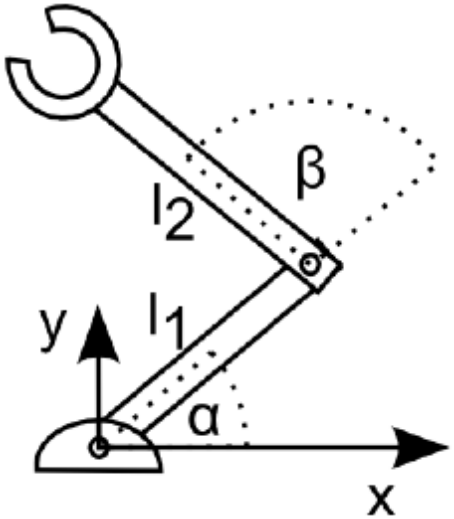
- $\nabla f_{\alpha_0} = (\alpha_0, \alpha_1, \alpha_2) = \frac{f(\alpha_0+\Delta_x, \alpha_1, \alpha_2) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta x}$
- $\nabla f_{\alpha_1} = (\alpha_0, \alpha_1, \alpha_2) = \frac{f(\alpha_0, \alpha_1+\Delta_y, \alpha_2) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta y}$
- $\nabla f_{\alpha_2} = (\alpha_0, \alpha_1, \alpha_2) = \frac{f(\alpha_0, \alpha_1, \alpha_2+\Delta_z) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta z}$

How do we Move the Robot?

- Linear equations dictate end-effector position:

$$x_e(\alpha,\beta) = l_1\cos(\alpha) + l_2\cos(\alpha+\beta)$$
$$y_e(\alpha,\beta) = l_1 sin(\alpha) + l_2\sin(\alpha+\beta)$$

Forward Kinematics Equations



- Relationship between position change and angle change:

$$\delta x_e = \frac{\delta x_e(\alpha,\beta)}{\delta\alpha}\Delta\alpha + \frac{\delta x_e(\alpha,\beta)}{\delta\beta}\Delta\beta$$

$$\delta y_e = \frac{\delta y_e(\alpha,\beta)}{\delta\alpha}\Delta\alpha + \frac{\delta y_e(\alpha,\beta)}{\delta\beta}\Delta\beta$$

- $J = \begin{bmatrix} \frac{\delta x_e}{\delta\alpha} & \frac{\delta x_e}{\delta\beta} \\ \frac{\delta y_e}{\delta\alpha} & \frac{\delta y_e}{\delta\beta} \end{bmatrix}$ Change in Position $= J \cdot \dot{\boldsymbol{q}}$

x_e: x position of end effector

y_e: y position of end effector

q: Robot pose in C-space

$\dot{q}$: Change in C-space

Using the Jacobian to Move the Robot

- $\frac{dp_e}{dt} = \boldsymbol{J}\frac{d\boldsymbol{q}}{dt}$, or in other words , $v_e = \boldsymbol{J} \cdot \dot{\boldsymbol{q}}$

- $\dot{q} = \boldsymbol{J^{-1}} \cdot \left[v_{e,d} + K\left(p_{e,d} - p\right)\right]$

$\dot{q}$: Change in C-space

K: gain

$v_{e,d}$: Desired velocity

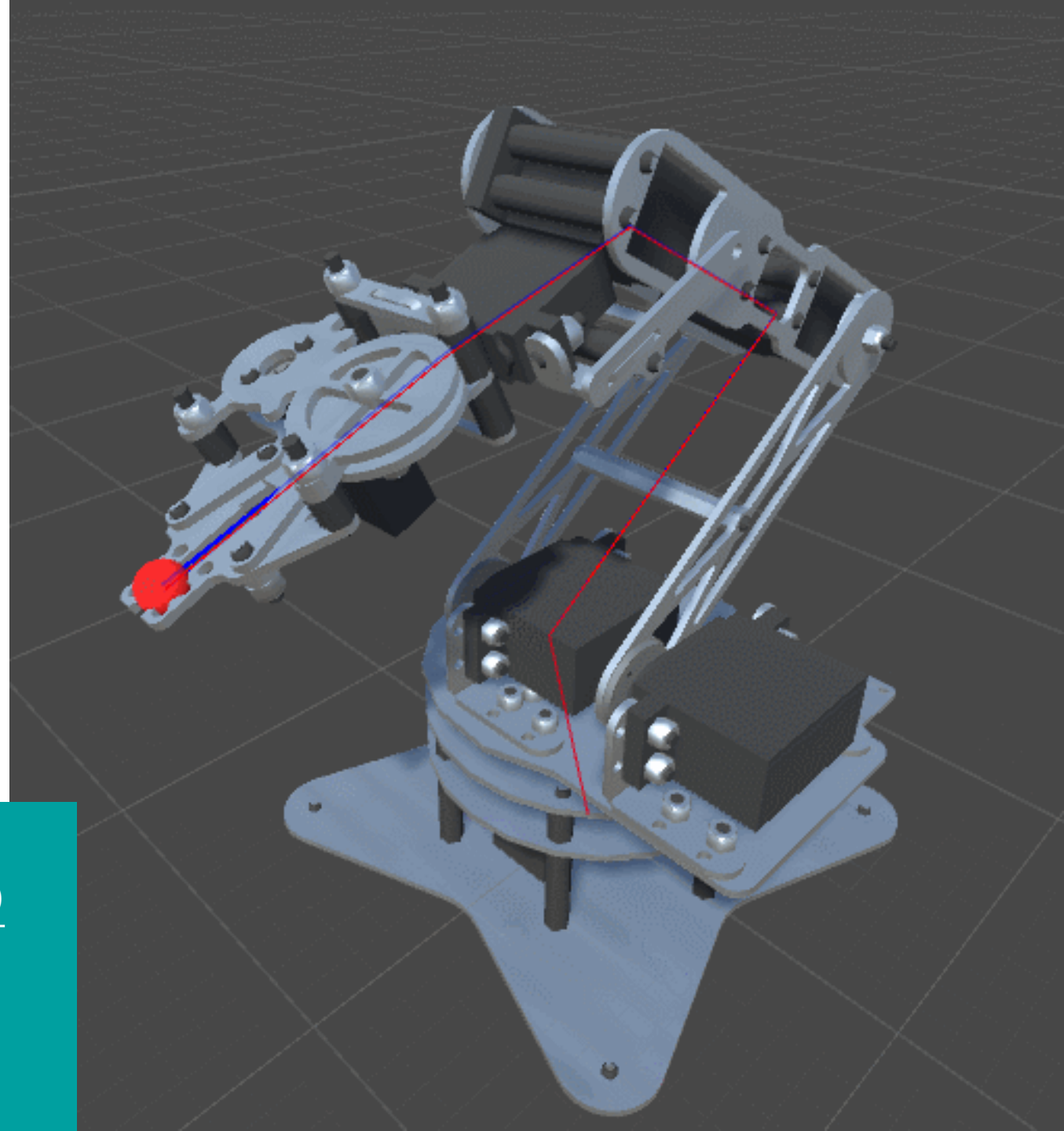
$p_{e,d}$: Desired position

Convergence

Problems!

$$\nabla f_{\alpha_0} = (\alpha_0, \alpha_1, \alpha_2) = \frac{f(\alpha_0 + \Delta_x, \alpha_1, \alpha_2) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta x}$$

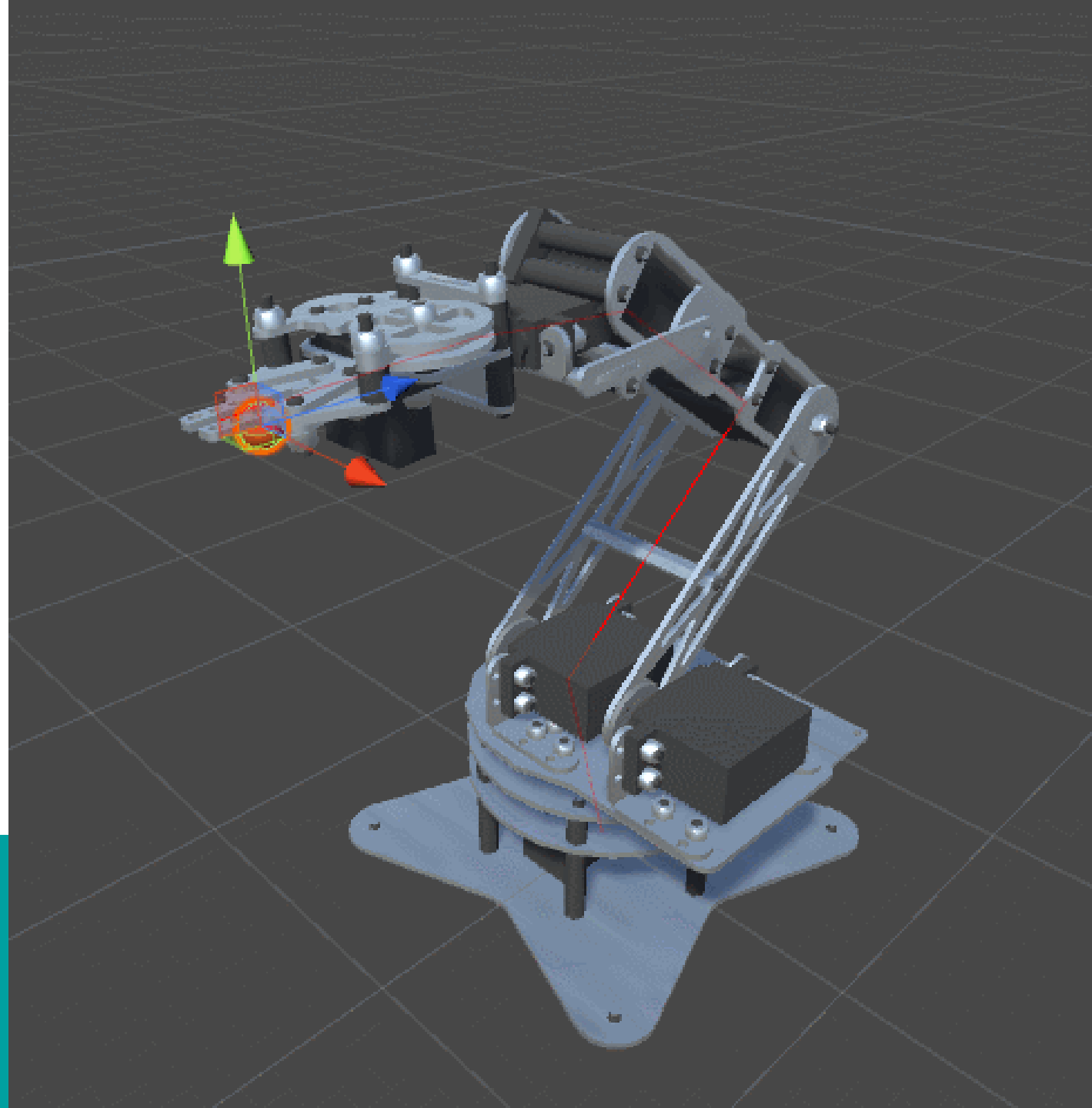
$$\alpha_0 \leftarrow \alpha_0 - L\nabla f_{\alpha_0}(\alpha_0, \alpha_1, \alpha_2)$$



Joint Angle

Limitation Problems!

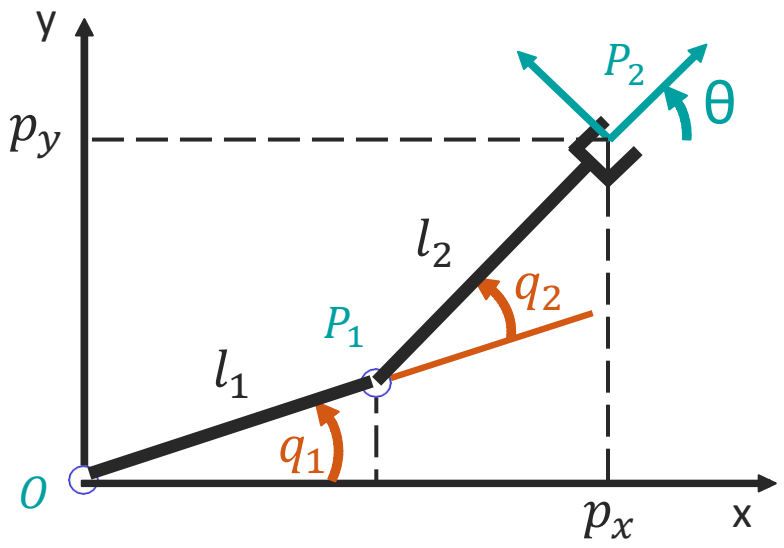
How do we fix this?



Section 3.3.3 in the book

Inverse Kinematics

IK for manipulator

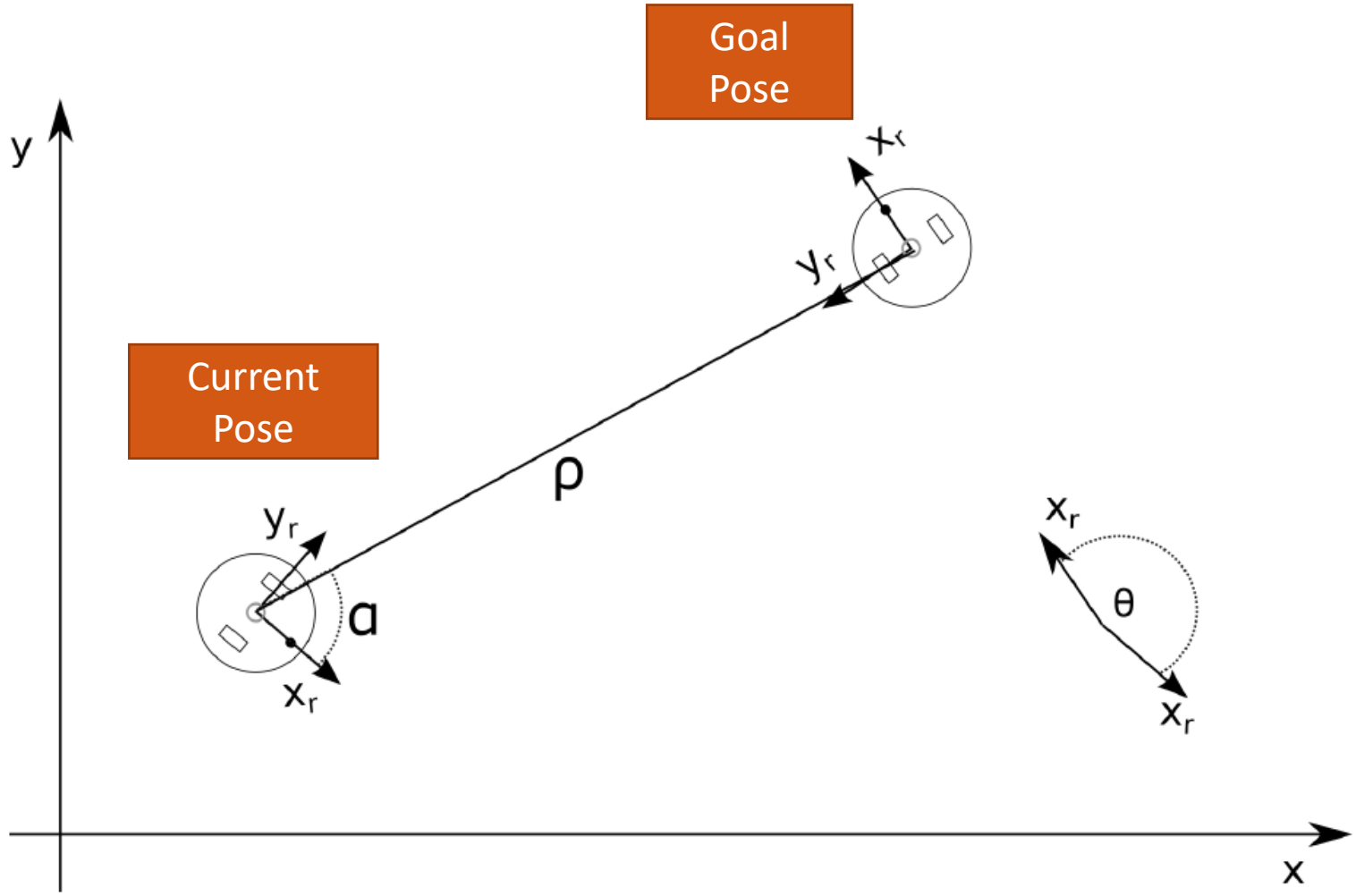


Given point (x, y, θ) Find angles (q_1, q_2)

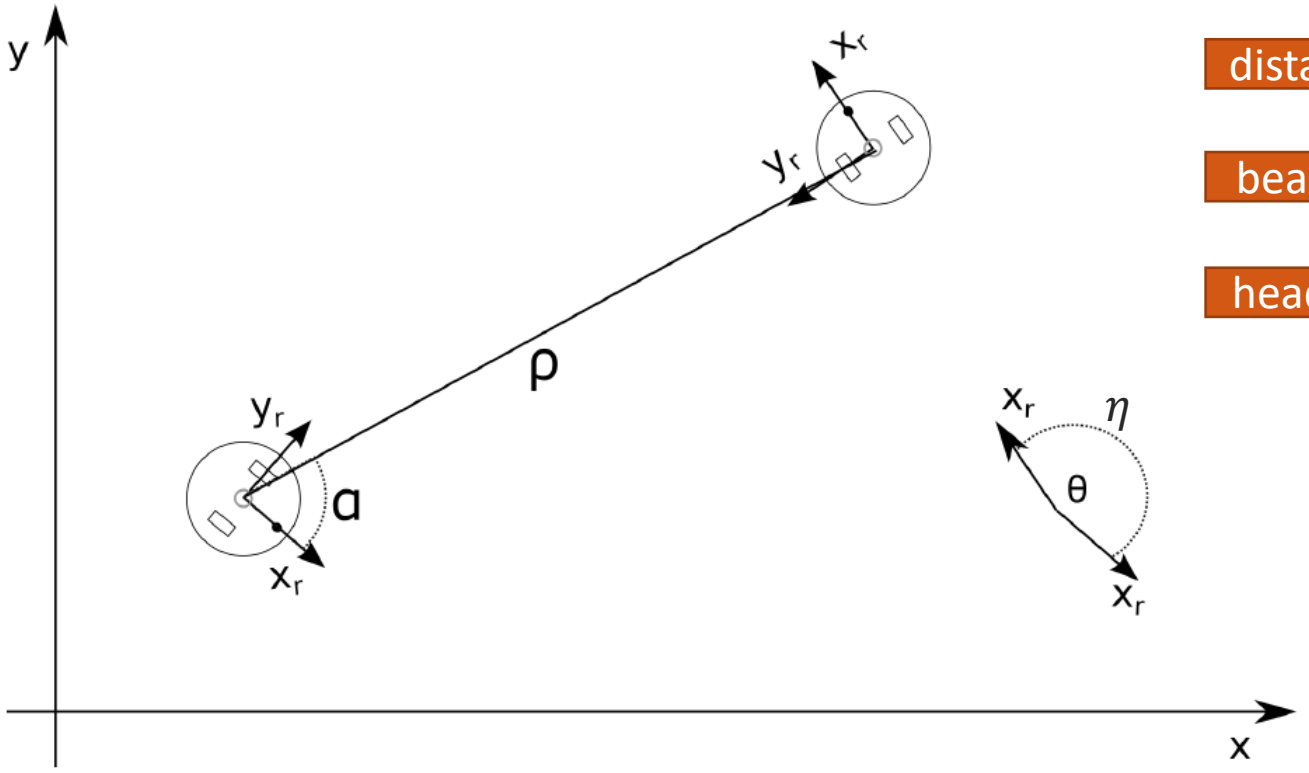
IK for e-puck

Given: (x, y, θ) Find: (ϕ_l, ϕ_r)

Position Change Using Feedback Control



Position Change Using Feedback Control



Error Terms

distance $\rho = \sqrt{(x_r - x_g)^2 + (y_r - y_g)^2}$

bearing $\alpha = \tan^{-1}\left(\frac{y_g - y_r}{x_g - x_r}\right) - \theta_r$

heading $\eta = \theta_g - \theta_r$

Update Rules

translation $\dot{x} = p_1\rho$

rotation $\dot{\theta} = p_2\alpha + p_3\eta$

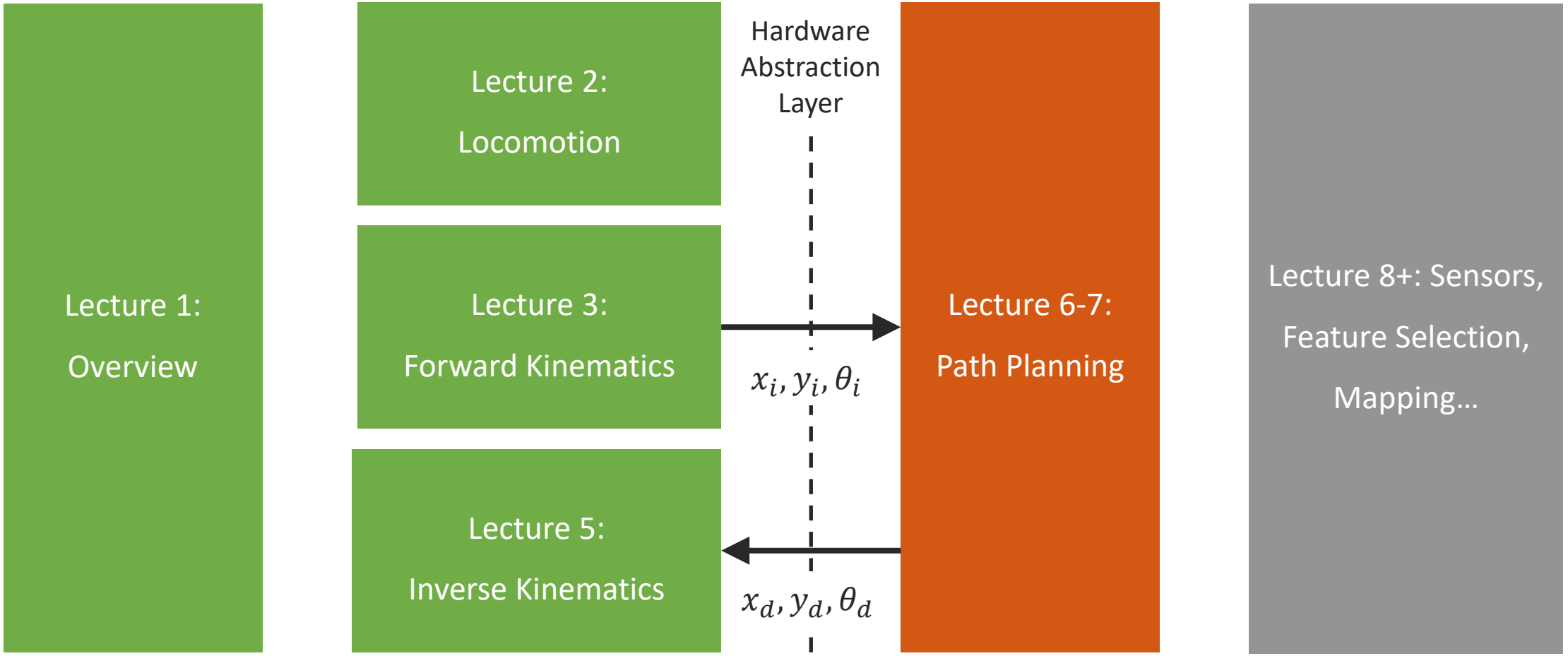
p_1, p_2, p_3 are controller gains

Chapter 4

Module 2 – COMPUTATION

Part I – Motion and Trajectory Planning

Roadmap


Lecture 1:
Overview
Lecture 2:
Locomotion
Lecture 3:
Forward Kinematics
Lecture 5:
Inverse Kinematics
Hardware
Abstraction
Layer
x_i, y_i, θ_i
x_d, y_d, θ_d
Lecture 6-7:
Path Planning
Lecture 8+: Sensors,
Feature Selection,
Mapping…

Some taxonomy

- **Kinematics**: geometrical relationships in terms of position/velocity between joint-space and task-space
- **Dynamics**: relationships between joint torques and dynamical properties of the plant with links/end-effector motions
- **Control**: computation of the control actions (i.e. joint torques) to achieve a desired motion.
- **Planning**: planning of the *desired* movements of the manipulator

Motion Planning

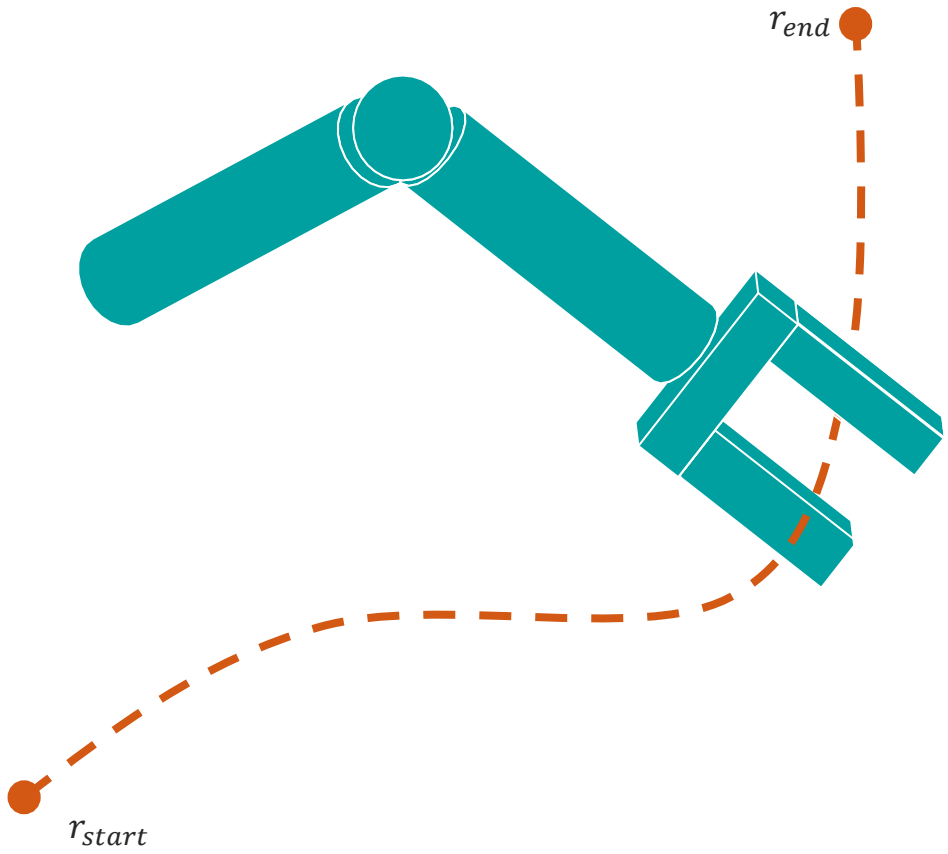
ξ → "xi"
Ξ → capital "xi"
Both pronounced "ksee"

- **Goal: Find a trajectory in configuration space from one point to another**
  - A trajectory is usually denoted as $\xi: t \in [0, T] \rightarrow C$
  - *(A function mapping time to robot configurations)*
  - The set of all trajectories is Ξ

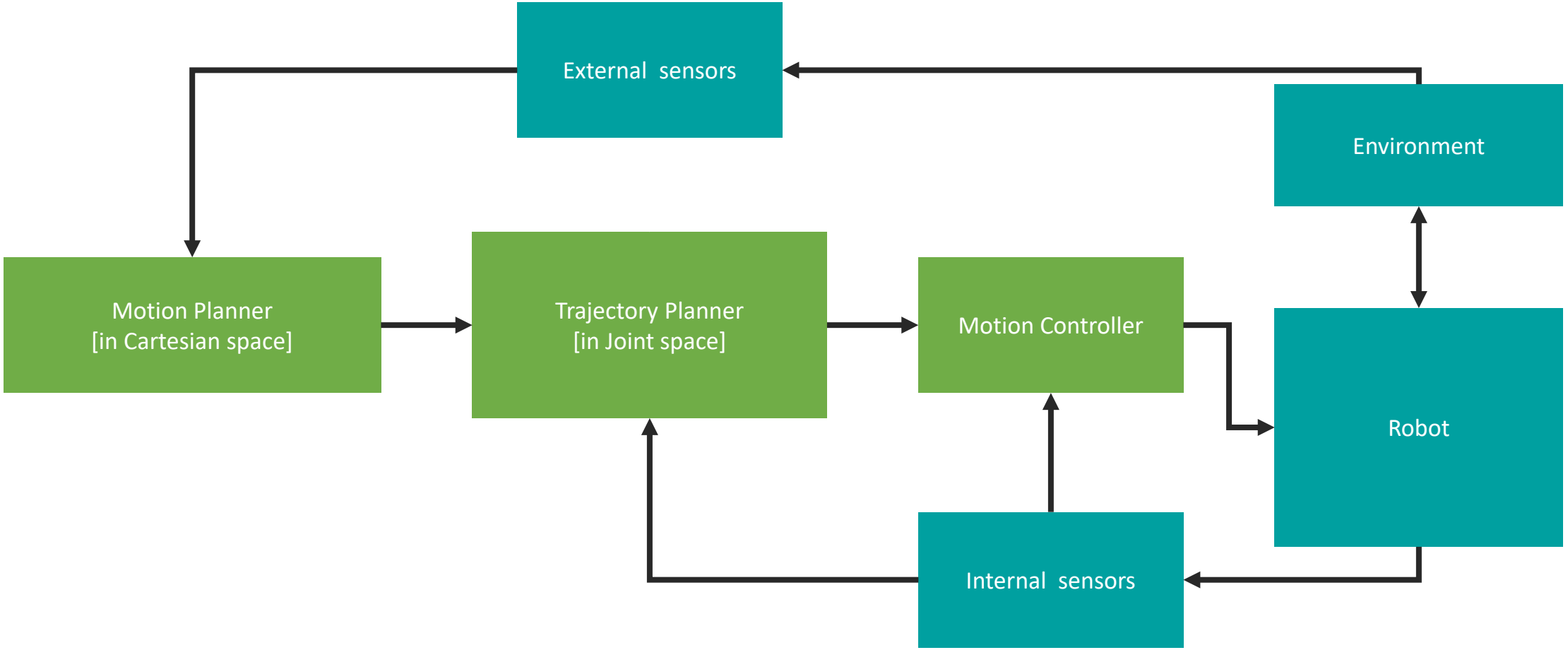
- The Bad News:
  - All complete search algorithms scale exponentially!!
  - Motion planning problems are high dimensional, e.g., big exponent.

What is a trajectory?

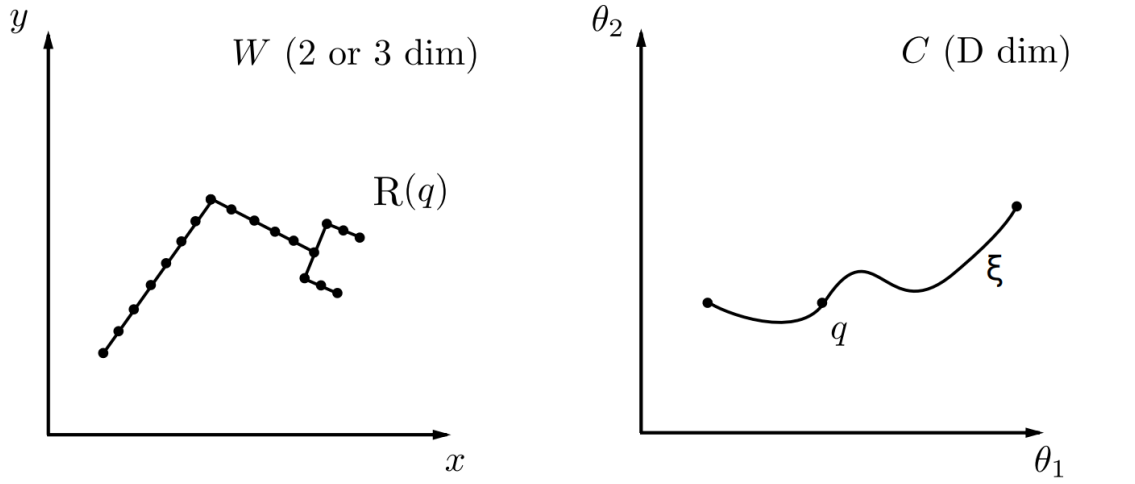
- **Path**: geometric description of a series of poses from r_{start} to r_{end} that respect specifications (e.g. avoiding obstacles or following a specific force profile)
- **Trajectory**: execution of path over time $r(t)$



Trajectory Planning Interfaces



Problem Specification: Spaces



Operational/World Space W($\mathbb{R}^3 or\ \mathbb{R}^6$)

Configuration/Joint Space C($\#DoF$)

Robot pose in World Space (set of points)

Single point in Configuration Space

Trajectory through Configuration Space (set of points)

Single point in Trajectory Space

Trajectory Optimization

Set of possible trajectories

Problem Statement

- Trajectory $\xi: t \in [0, T] \rightarrow C$ Maps time to configurations
- Objective Function $U: \Xi \rightarrow \mathbb{R}^+$ Maps trajectories to scalars

- The objective U encodes traits we want our path to have
  - Path length
  - Efficiency
  - Obstacle avoidance
  - Legibility
  - Uncertainty reduction
  - Human comfort
- Goal: Leverage the benefits of randomized sampling with asymptotic optimality

Our goal: Use a feedback controller that minimizes path length for our low dimensional problem

Trajectory Optimization

Problem Statement

- Trajectory $\xi: t \in [0, T] \rightarrow C$ Maps time to configurations
- Objective Function $U: \Xi \rightarrow \mathbb{R}^+$ Maps trajectories to scalars

- The objective U encodes traits we want our path to have
  - Path length
  - Efficiency
  - Obstacle avoidance
  - Legibility
  - Uncertainty reduction
  - Human comfort
- Goal: Leverage the benefits of randomized sampling with asymptotic optimality

Set of possible trajectories

Our goal: Use a feedback controller that minimizes path length for our low dimensional problem

Problem Specification: Optimization

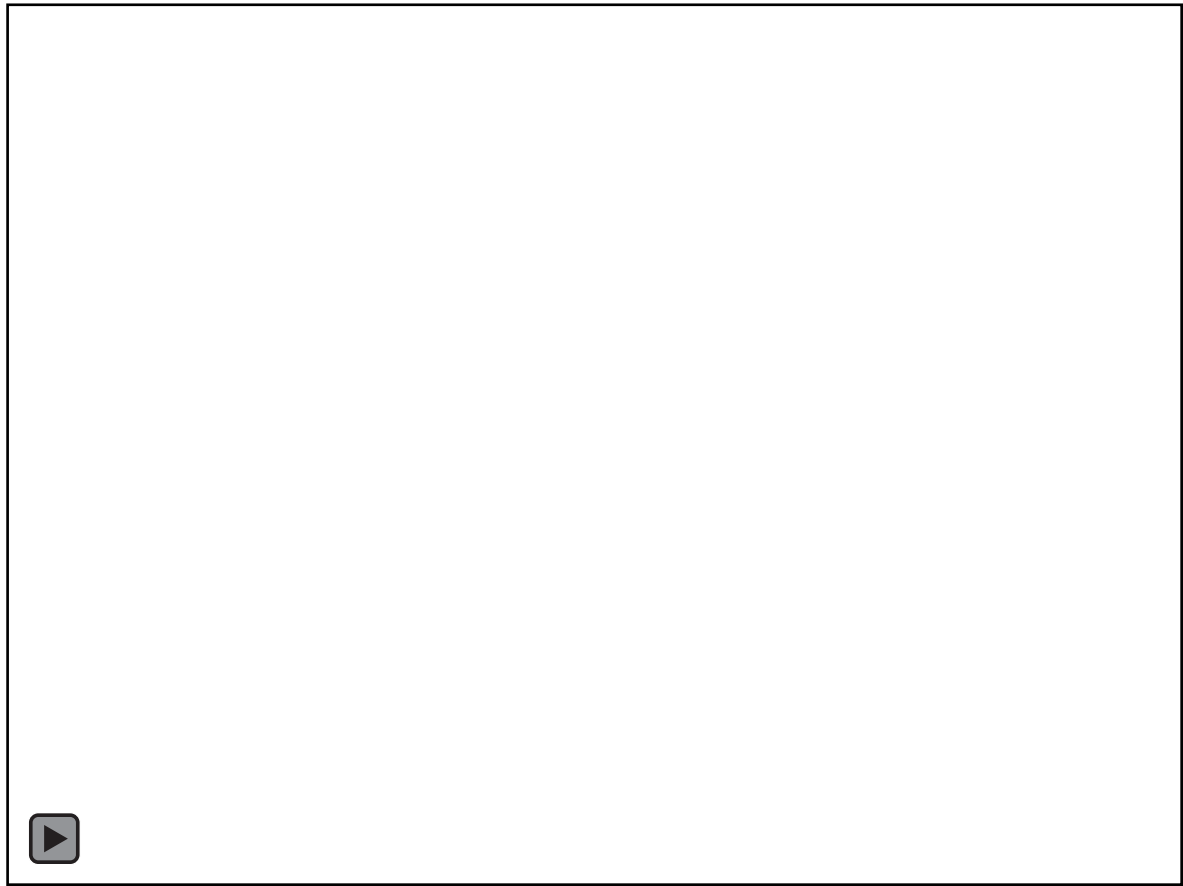
Trajectory Optimization seeks to find an optimal trajectory ξ^*:

$$\xi^* = argmin_{\{\xi \in \Xi\}} U[\xi]$$

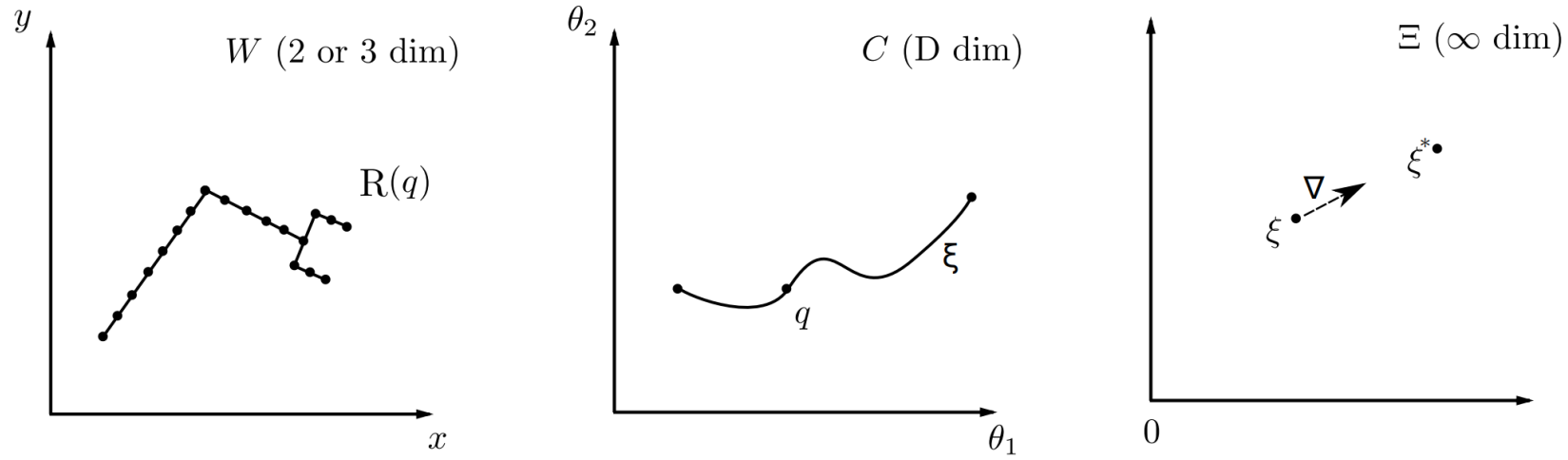
s.t. $\xi(0) = q_s$

$\xi(T) = q_g$

…(any other constraints we want)



Problem Specification: Optimization



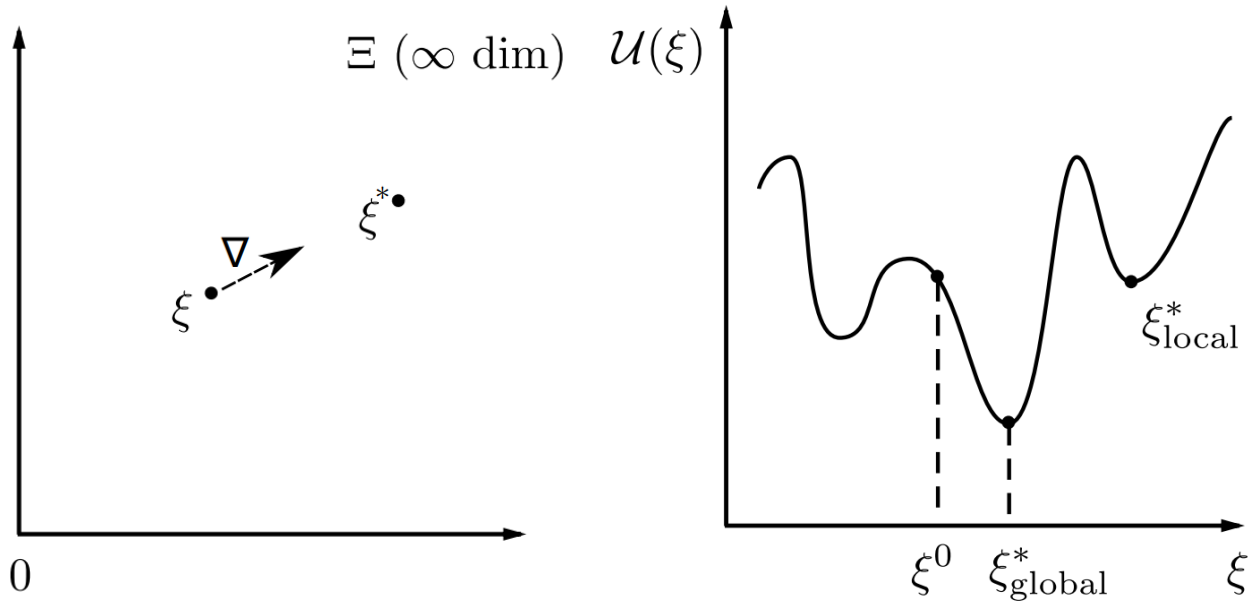
- Want to optimize ξ to a global minimum of our objective U

  ⇒ Usually too hard!

- Instead, optimize ξ to a local minimum of our objective U

  ⇒If the solution is bad, resample ξ and try again

Problem Specification: Optimization

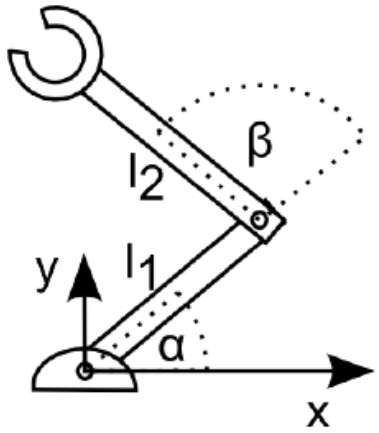


Points in the direction of greatest ascent for ξ

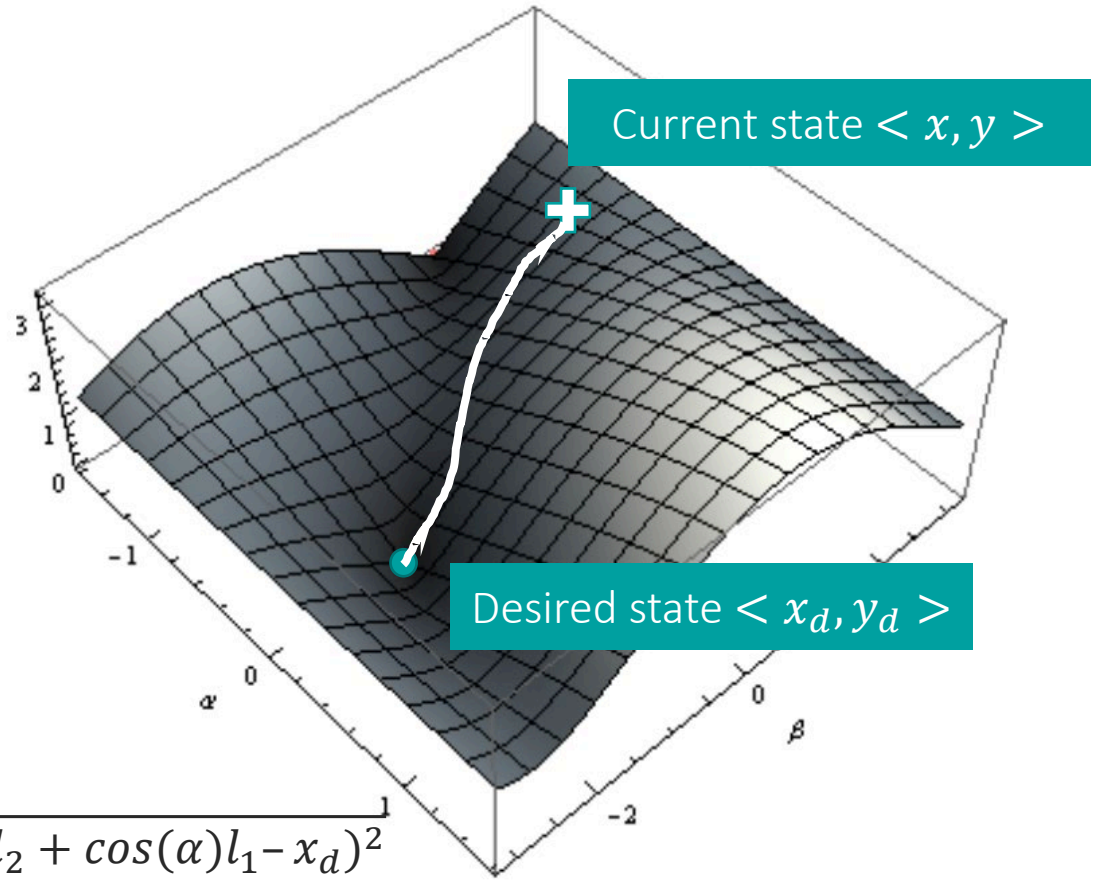
Gradient Descent is a popular technique for finding local minima

$$\xi_{t+1} = \xi_t - \frac{1}{\alpha} \nabla_\xi U(\xi_t)$$

Recall: Motion Planning in EE Space



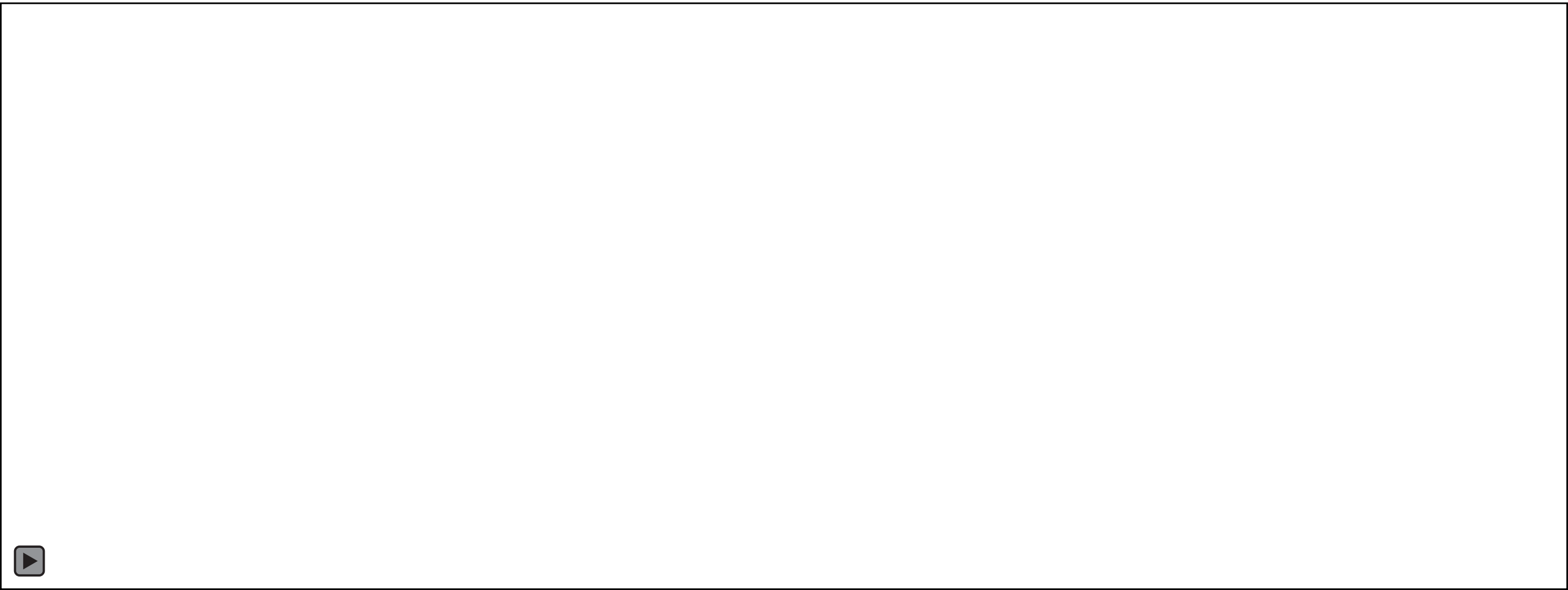
$$x = l_1 \cos(\alpha) + l_2 \cos(\alpha + \beta)$$
$$y = l_1 \sin(\alpha) + l_2 \sin(\alpha + \beta)$$



Just the Euclidean distance between two vectors!

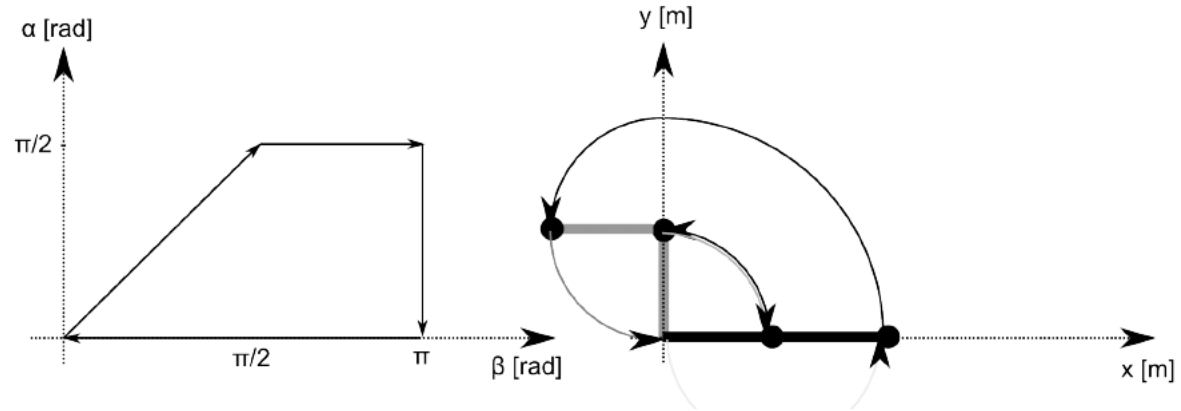
$$f_{x,y}(\alpha, \beta) = \sqrt{(\sin(\alpha + \beta) l_2 + \sin(\alpha) l_1 - y_d)^2 + (\cos(\alpha + \beta) l_2 + cos(\alpha) l_1 - x_d)^2}$$

Motivating problem: Multi-robot coordination



Motivating Problem 2: Moving e-puck from start to goal state

Configuration Space



- Allows us to reduce robot positions to a single point
  - Very convenient for planning!
- One axis of configuration space for each degree of freedom of the robot
  - Convenient for planning as the space is the same as what the robot controls.
  - Removes the need to figure out how to get to a point in (X, Y)-space
- We still need to figure out how to put obstacles into configuration space!

Configuration Space

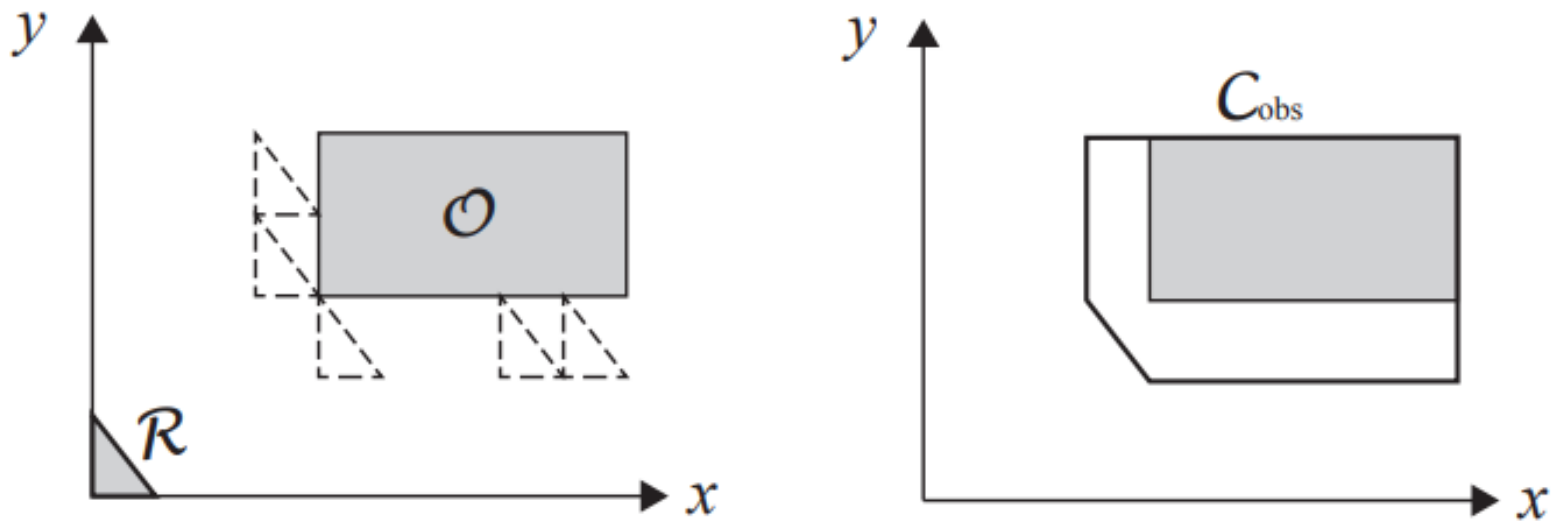
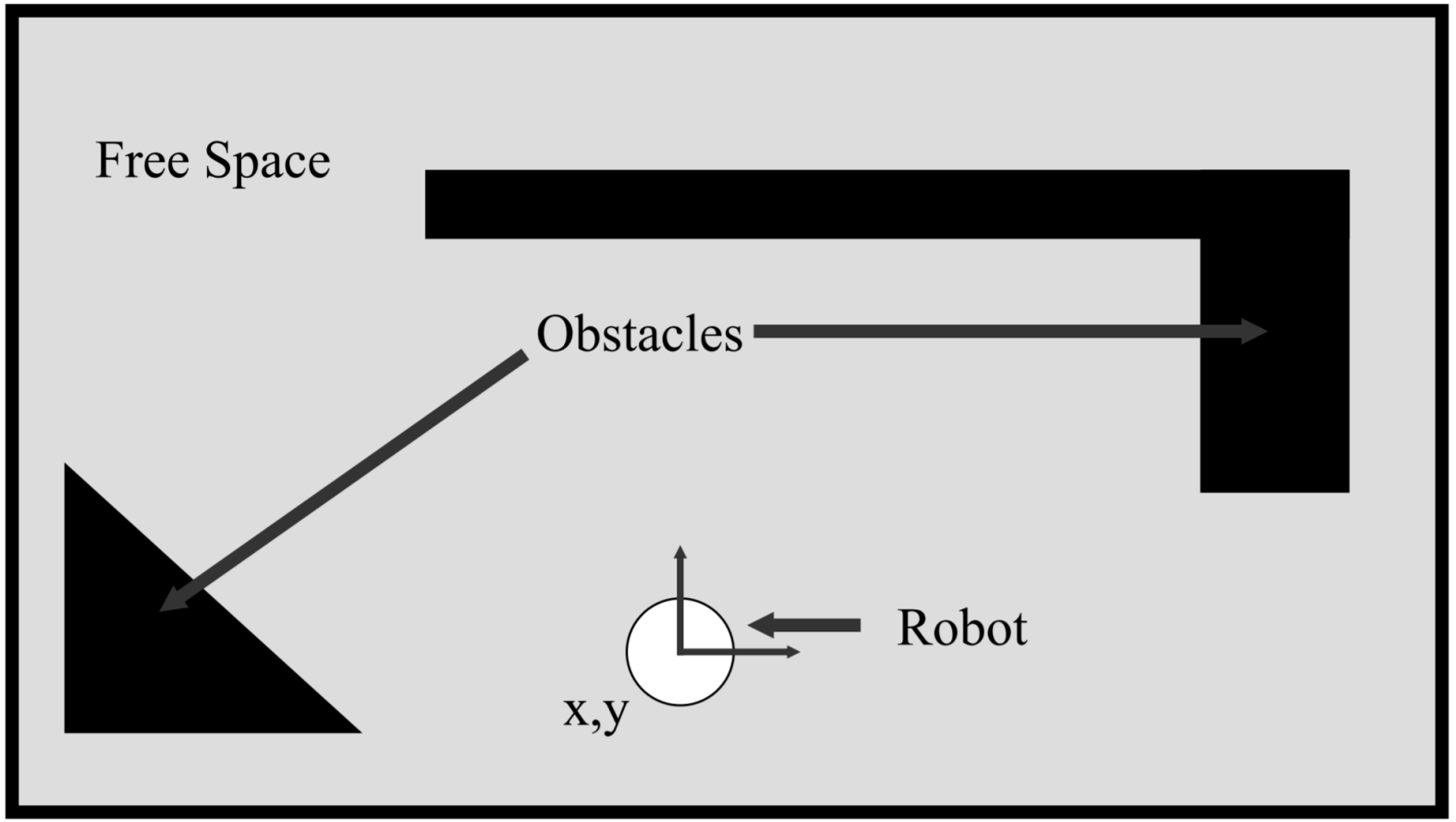


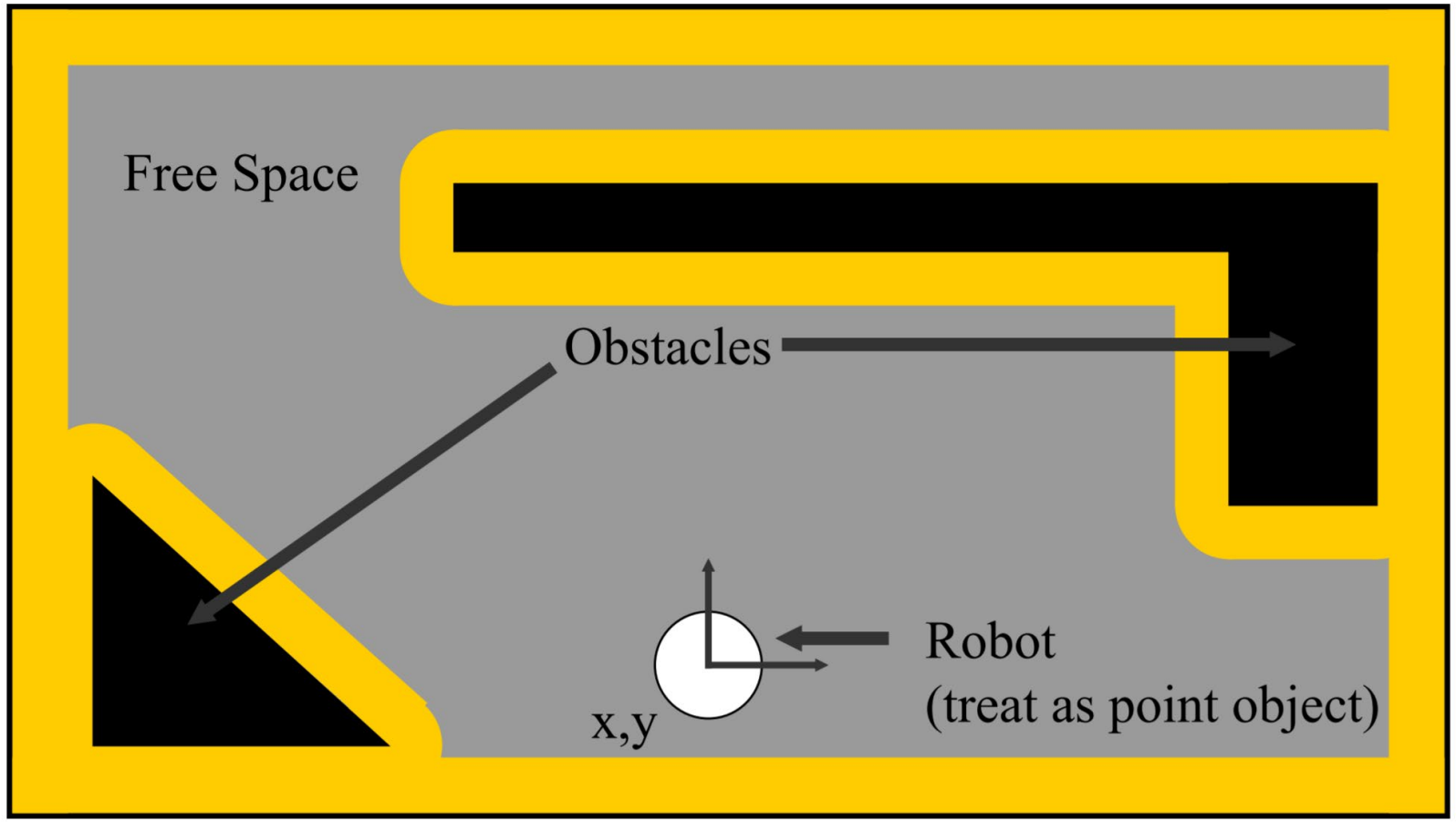
Figure 2.4: C_{obs} for a robot R that translates in x-y with a rectangle obstacle O

Example of taking a robot that can translate across (X, Y) and an obstacle and putting them in configuration space.

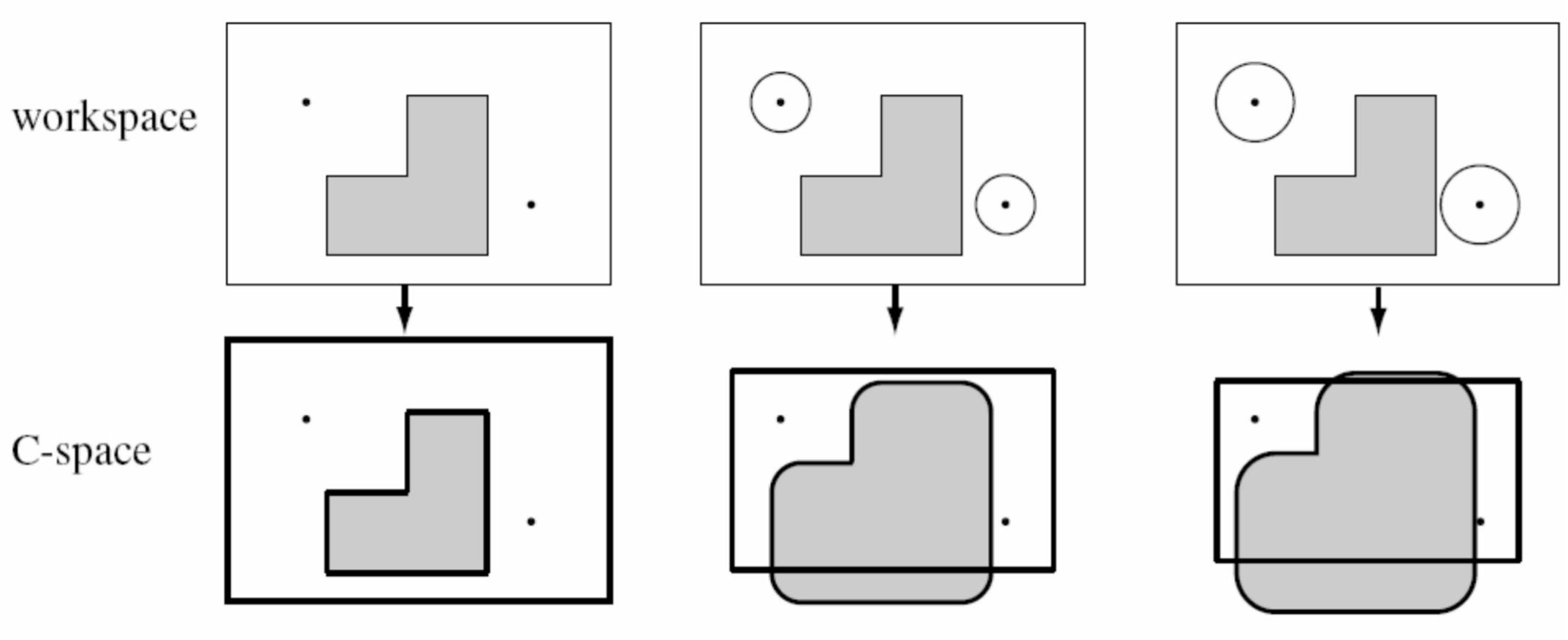
Configuration Space



Configuration Space

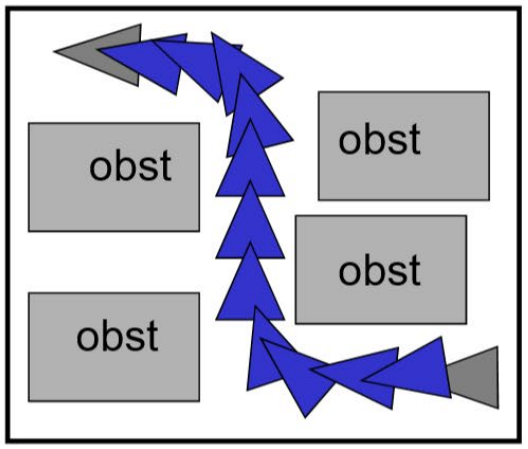


Configuration Space



Configuration Space

Each step has to check for collisions between the robot's body and obstacles!



The robot's path is just a path through space – as a single point, collision checks are easy!