CSCI 3302 / ECEN 3303 Introduction to Robotics

Alessandro Roncone

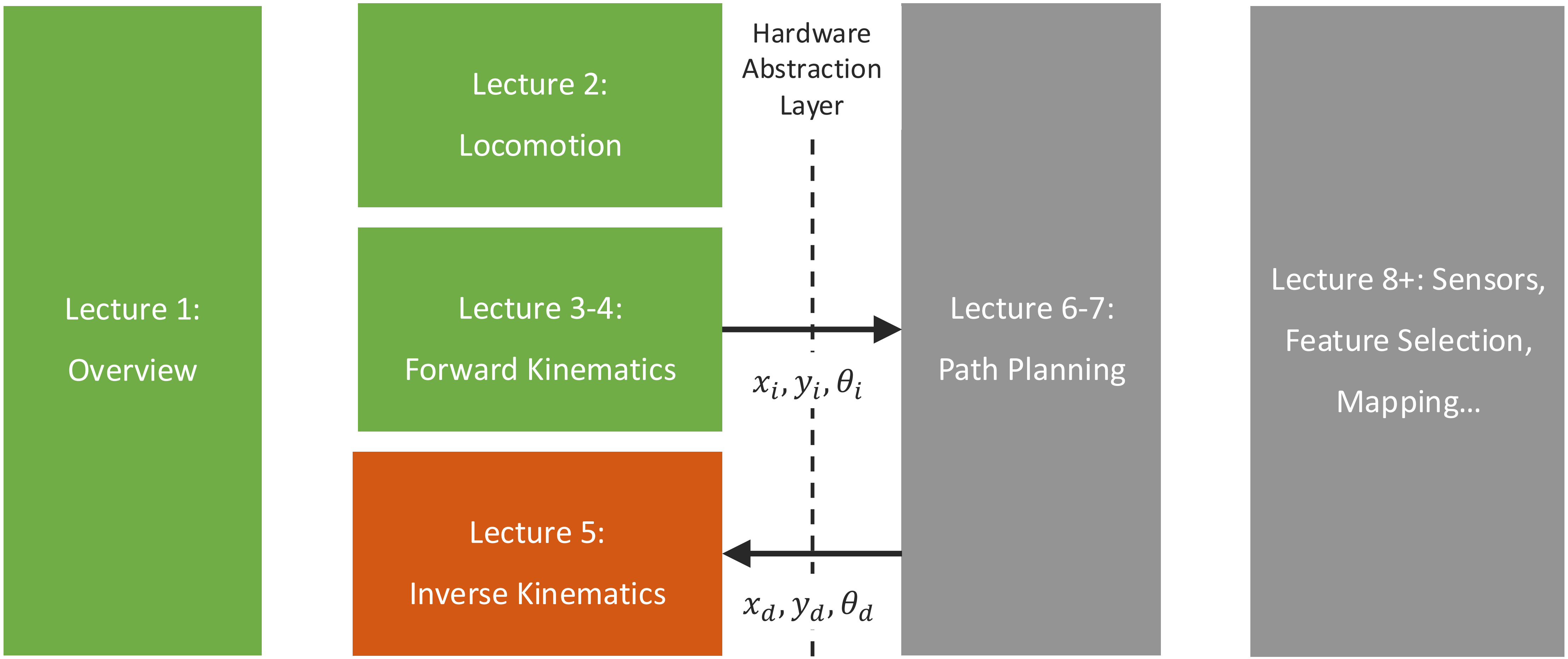
aroncone@colorado.edu https://hiro-group.ronc.one

Chapter 3

Module 1 – MECHANISMS

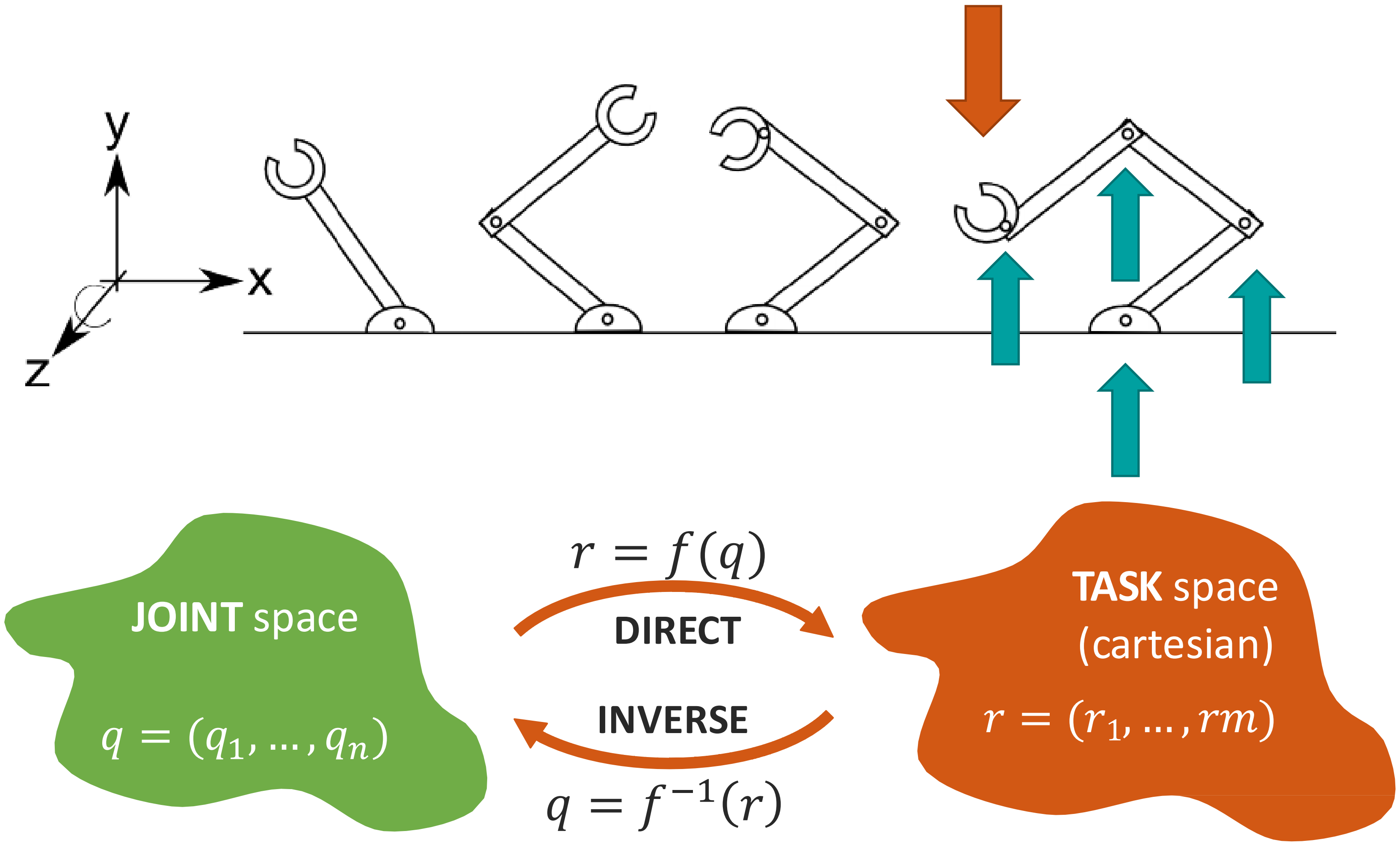
Part III – Inverse Kinematics

Roadmap



Lecture 1:
Overview
Lecture 2:
Locomotion
Lecture 3-4:
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…

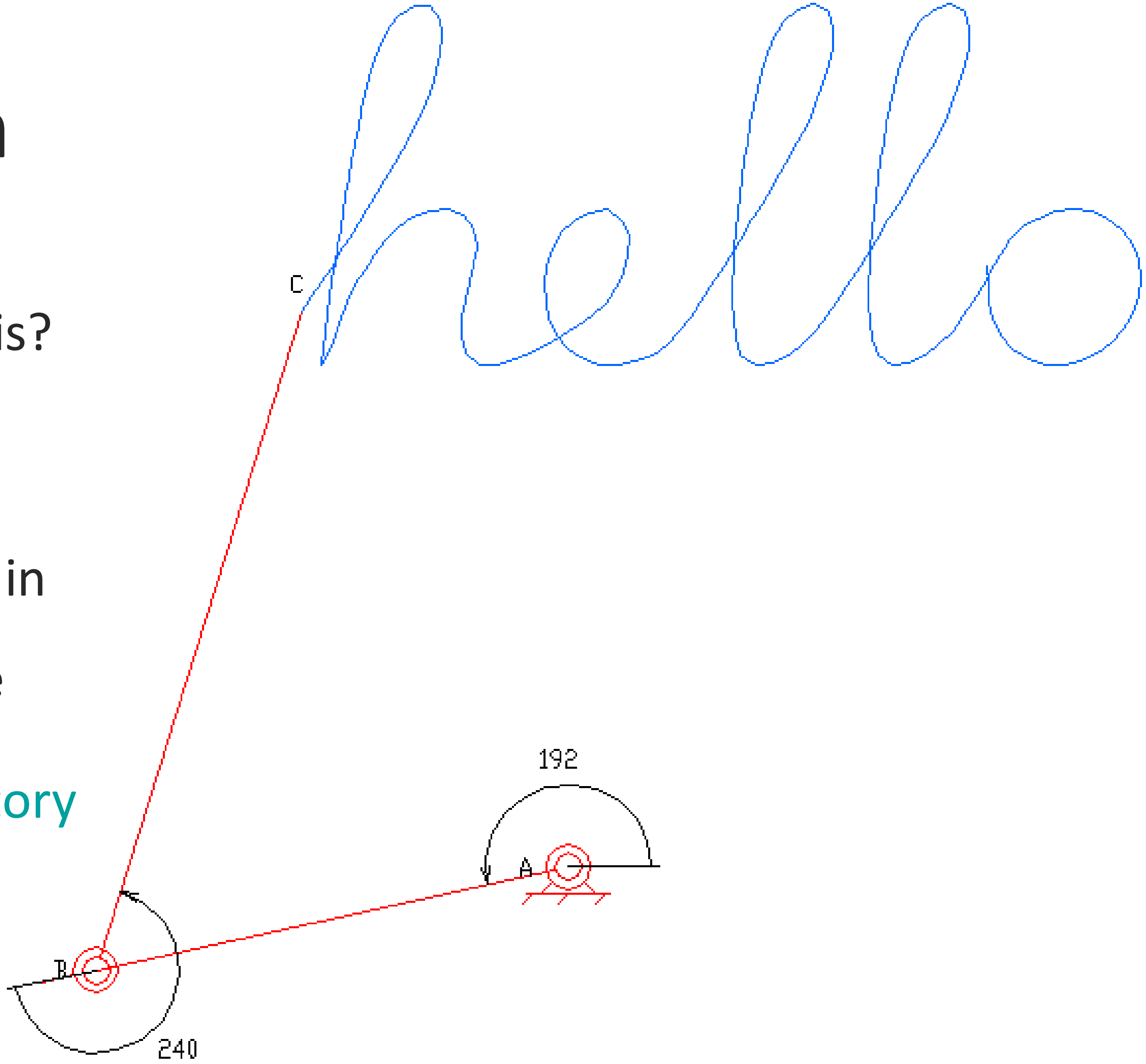
Inverse Kinematics



Motivating Problem

How can we get a robot to do this?

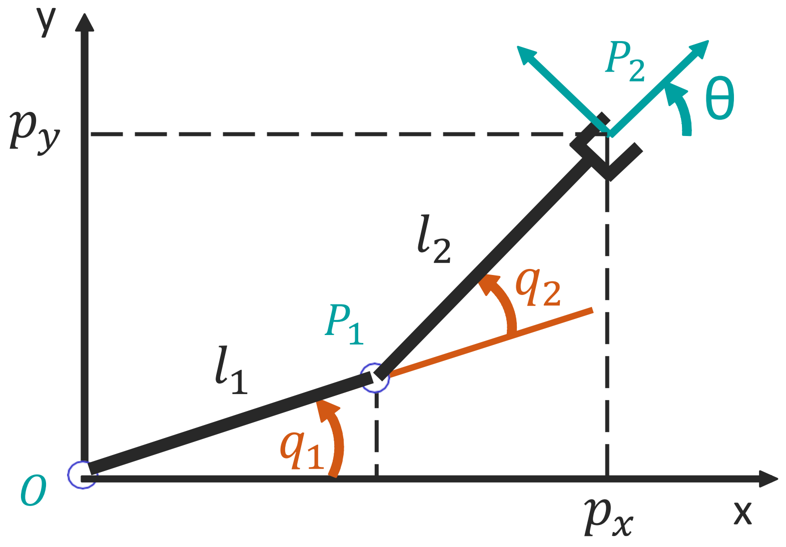
- We need to define a trajectory in end-effector/operational space
- We need to convert this trajectory to joint/configuration space



Section 3.4 in the book

Inverse Kinematics

IK for manipulator

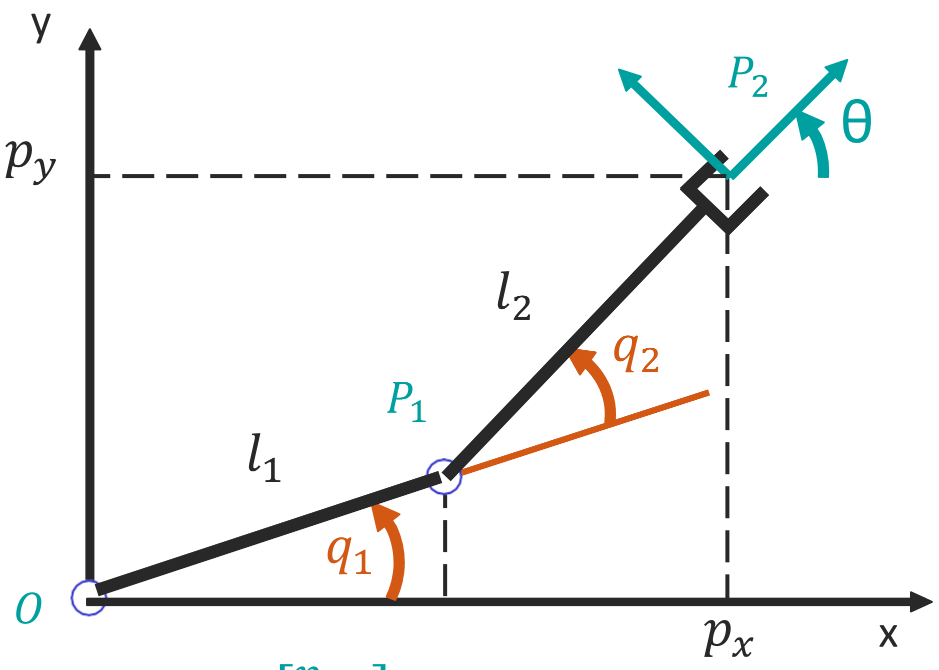


Given point (x, y, θ) Find angles (q_1, q_2)

IK for e-puck

Given: (x, y, θ) Find: (ϕ_l, ϕ_r)

Example: Inverse Kinematics of a 2R arm



Given $r = \begin{bmatrix} p_{2,x} \\ p_{2,y} \\ \theta \end{bmatrix}$ ⇐ Operational-space DOFs

Find $q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ ⇐ Joint-space DOFs

$$p_{1,x} = l_1 \cos(q_1) \rightarrow q_1 = \left[\cos^{-1}\frac{x_1}{l_1}, -\cos^{-1}\frac{x_1}{l_1}\right]$$

$$p_{2,x} = x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$

$$p_{2,y} = y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$

$$\Downarrow$$

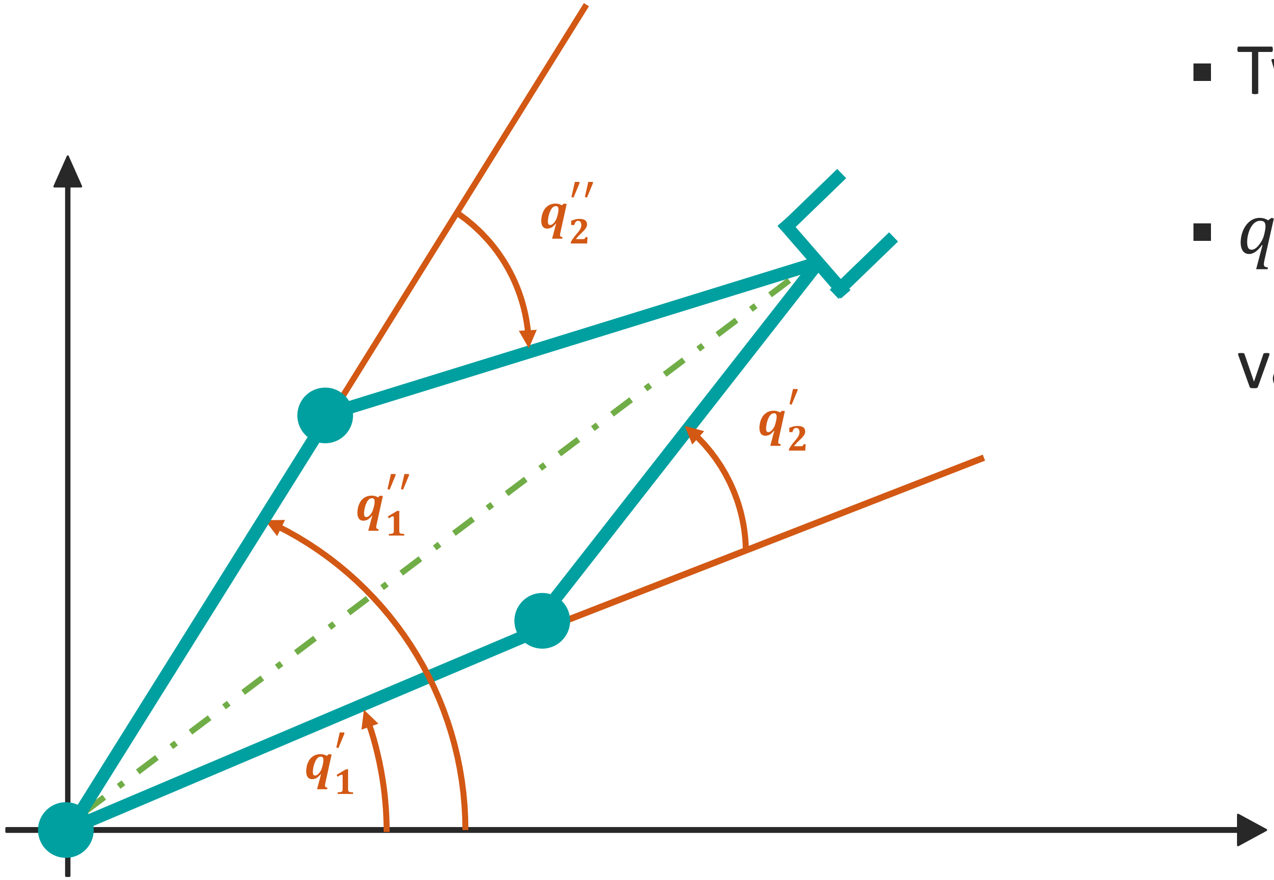
$$q_1 = \cos^{-1}\frac{x^2 y + y^3 - \sqrt{4x^4 - x^6 + 4x^2y^2 - 2x^4y^2 - x^2y^4}}{2(x^2 + y^2)}$$

$$q_2 = -\cos^{-1}\frac{1}{2}(-2 + x^2 + y^2)$$

r=f(q)

$$p_{2,x} = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$
$$p_{2,y} = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$
$$\theta = q_1 + q_2$$

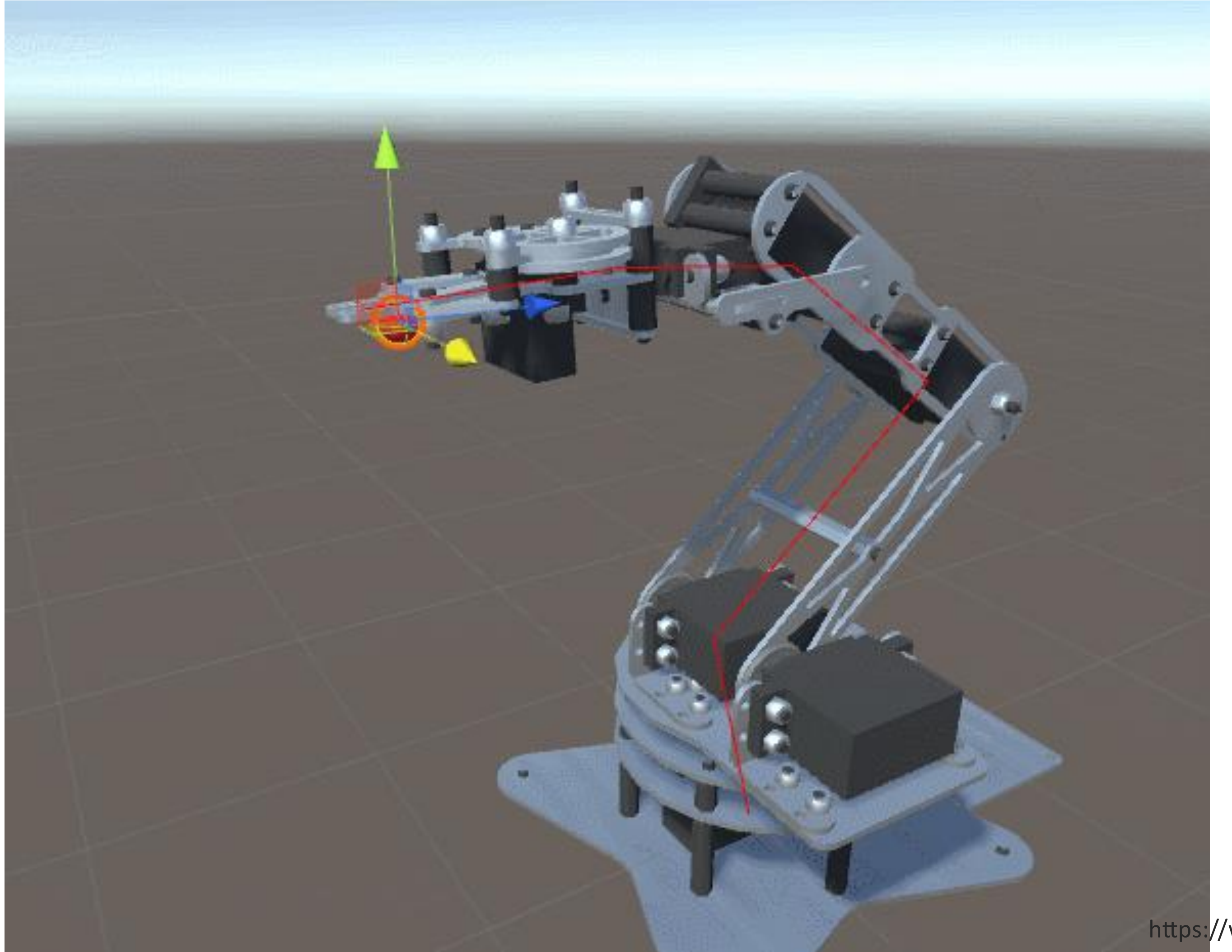
Example: Inverse Kinematics of a 2R arm



- Two solutions!
- q_2' and q_2'' have same absolute value, but different sign

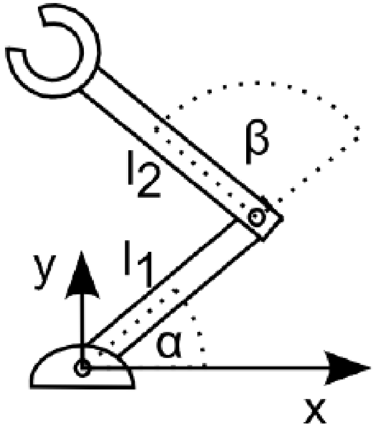
Great video about IK: https://www.youtube.com/watch?v=IKOGwoJ2HLk

End-effector Position Control

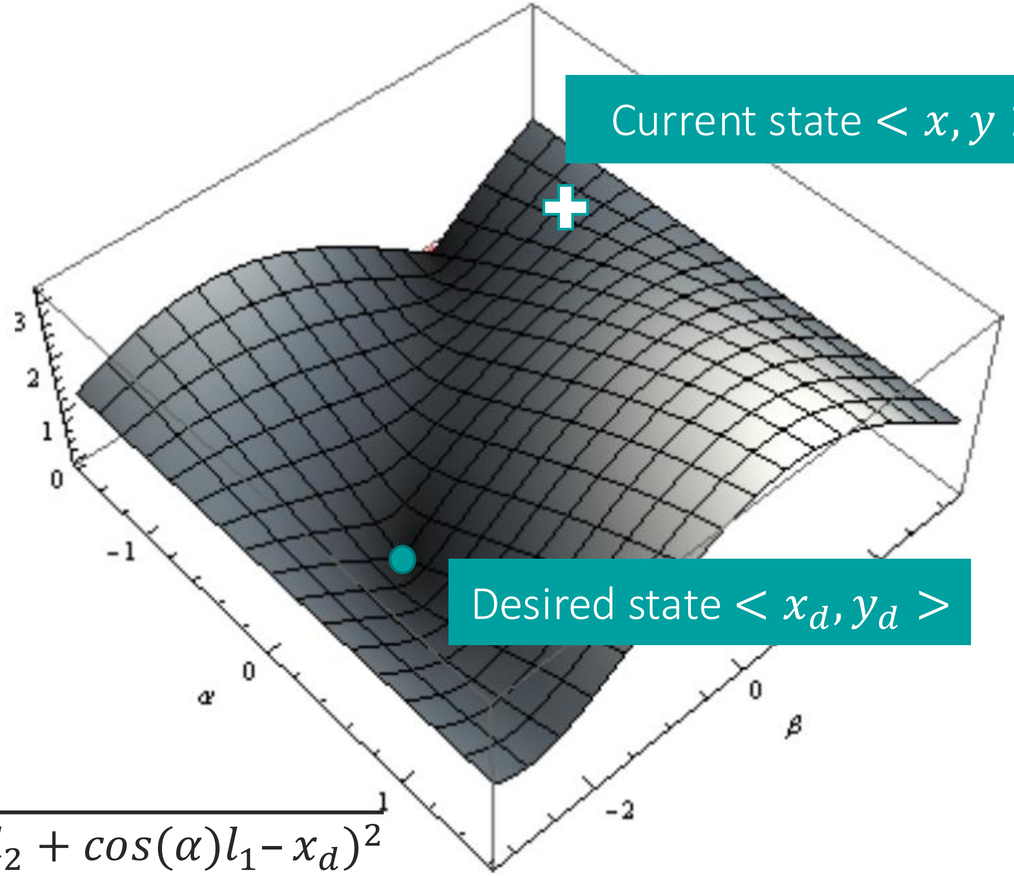


https://www.alanzucconi.com/2017/04/10/robotic-arms/

Easier ways to solve the IK problem



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



Current state $< x, y >$

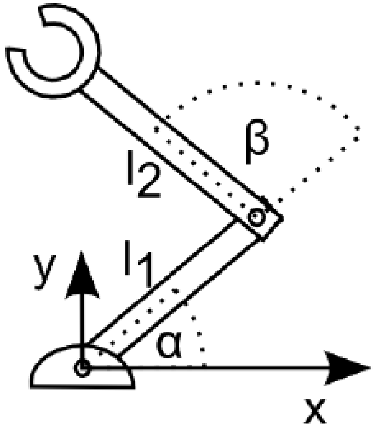
Desired state $< x_d, y_d >$

Just the Euclidean distance between two vectors!

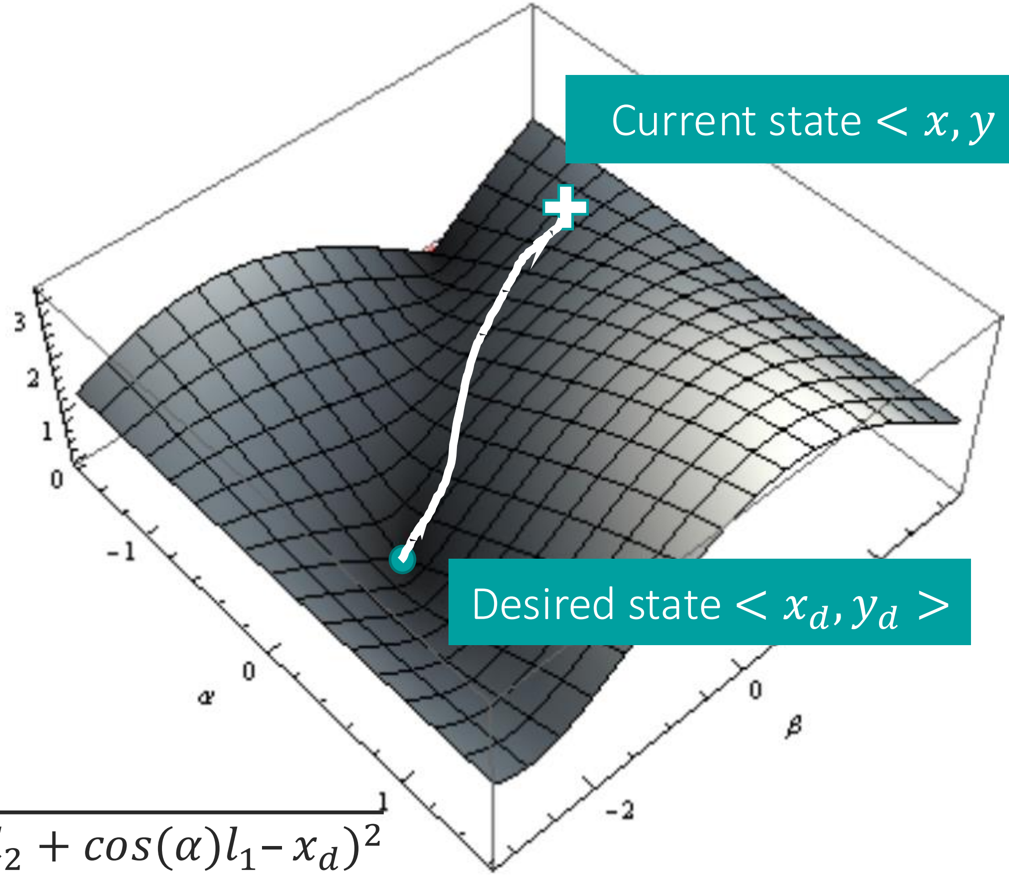
$$f_{x,y}(\alpha, \beta) = \sqrt{(\sin(\alpha + \beta)l_2 + \sin(\alpha)l_1 - y_d)^2 + (\cos(\alpha + \beta)l_2 + cos(\alpha)l_1 - x_d)^2}$$

EE → End-Effector

Motion Planning in EE Space



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



Current state $< x, y >$

Desired state $< x_d, y_d >$

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

Coordinate Descent

IDEA: We can minimize a multivariate function by tweaking one parameter at a time

$$x^0 = (x_1^0, \dots, x_n^0)$$

At iteration 0, all n variables are set to their initial values

$$x_i^{k+1} = argmin_{y \in \mathbb{R}}\, f(x_1^{k+1}, \dots, x_{i-1}^{k+1}, y, x_{i+1}^{k}, \dots, x_n^{k})$$

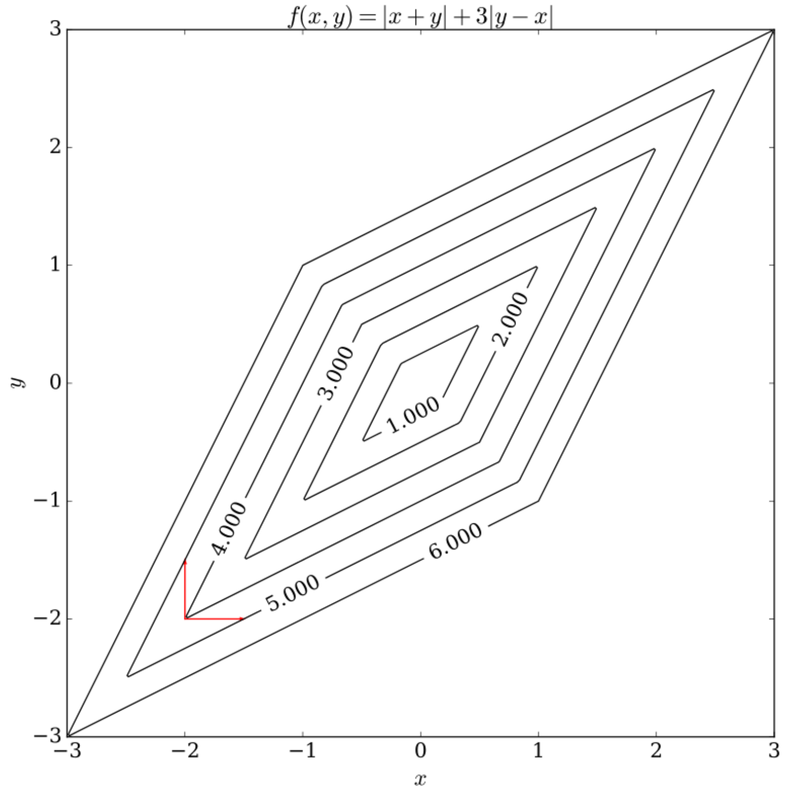
At iteration $k + 1$, $n - 1$ variables are set to their previous values, and only one (position i) is updated to a value minimizing f

$$F(x^0) \geq F(x^1) \geq F(x^2) \geq \cdots$$

Each iteration reduces our error or remains stationary

Coordinate Descent

IDEA: We can minimize a multivariate function by tweaking one parameter at a time

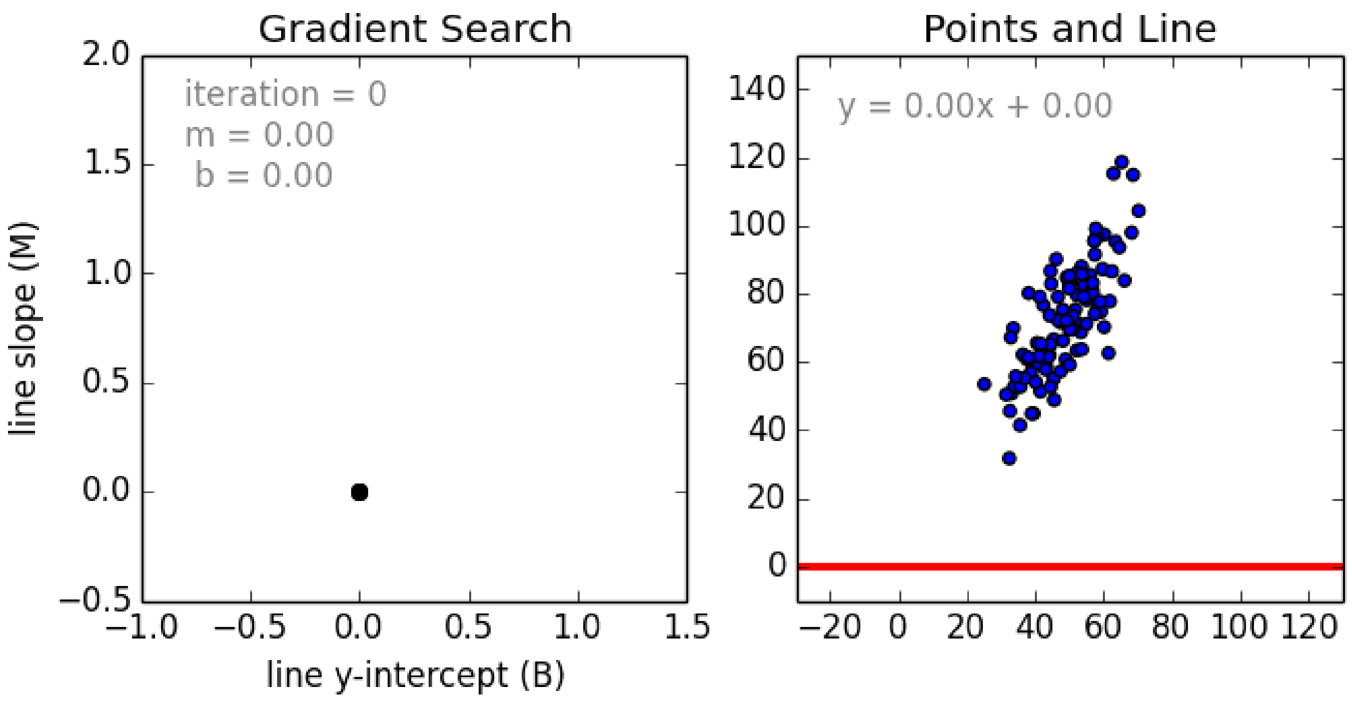


- Choose an initial parameter vector $\boldsymbol{x}$
- Until convergence is reached (or for some fixed number of iterations):
  - Choose an index i from 1 to n
  - Choose a step size α
  - Update x_i to $x_i - \alpha \frac{\partial F}{\partial x_i}(\boldsymbol{x})$

Trouble when all axis-aligned movements increase loss function!

Gradient Descent

- Avoids pitfalls of univariate optimization
- Requires a gradient (can be analytic or empirical)
- Takes steps that optimize across all variables



Gradient Descent for Solving IK

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

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

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

Gradient Descent for Solving IK

- Gradient Definition:

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

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

- Update Rule:

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

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

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

Distance from goal
$f: \mathbb{R}^3 \to \mathbb{R}$

Joint angles
α_i

Learning rate
L

How do we Move the Robot?

- Linear equations dictate end-effector position:

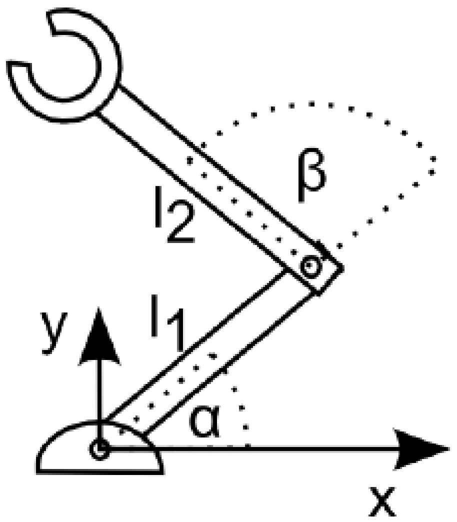
$$\left.\begin{aligned} x_e(\alpha,\beta) &= l_1\cos(\alpha) + l_2\cos(\alpha+\beta) \\ y_e(\alpha,\beta) &= l_1\,sin(\alpha) + l_2\sin(\alpha+\beta) \end{aligned}\right\} \text{Forward Kinematics Equations}$$

- Relationship between position change and angle change:

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

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

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



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 , $v_e = \boldsymbol{J} \cdot \dot{\boldsymbol{q}}$

- $\dot{q} = \boldsymbol{J}^{-\mathbf{1}} \cdot \left[v_{e,d} + K\left(p_{e,d} - p\right)\right]$
- With a single equation, you are automatically performing gradient descent!

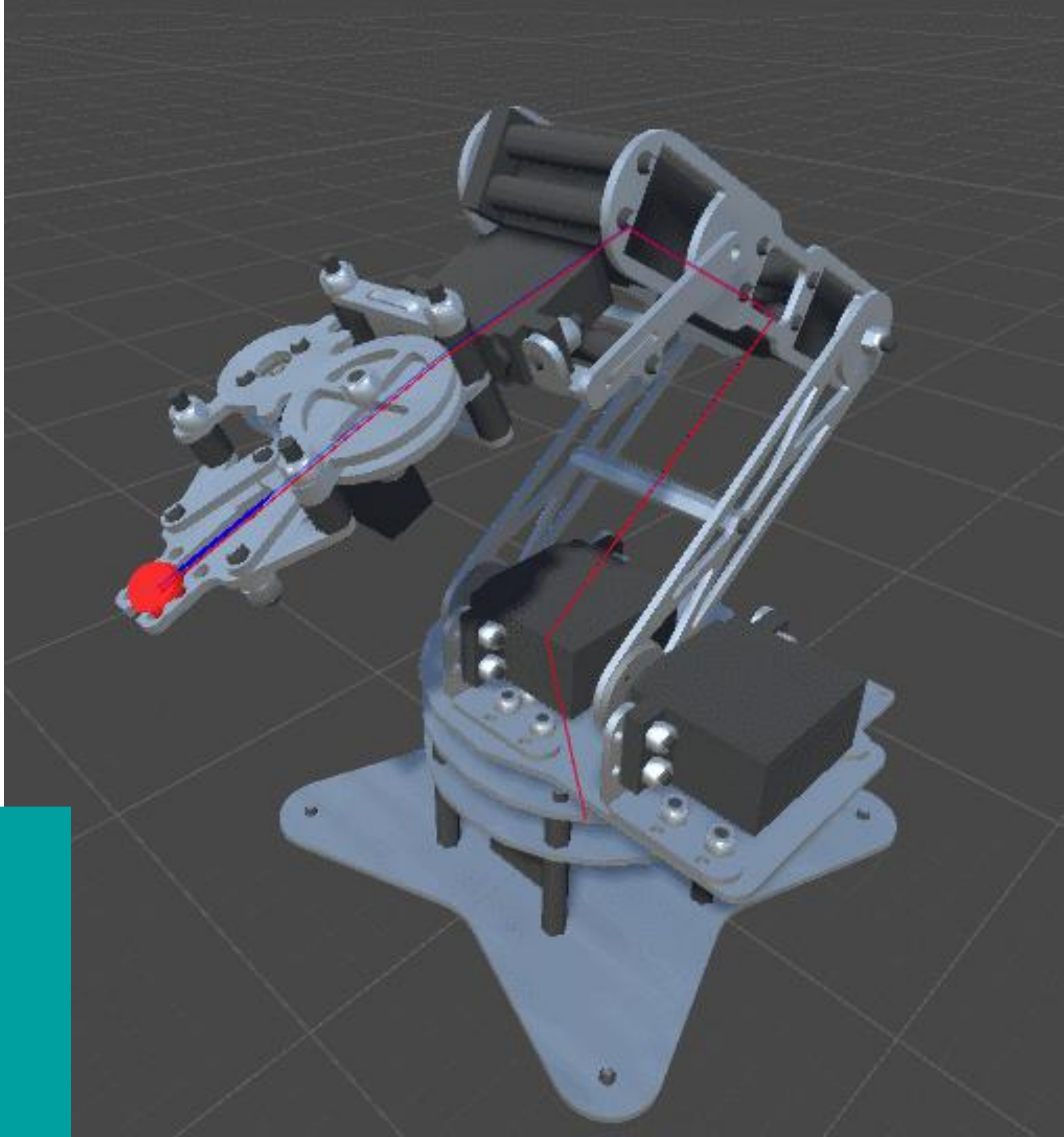
$\dot{q}$: Change in C-space

K: gain

$v_{e,d}$: Desired velocity

$p_{e,d}$: Desired position

Convergence Problems!

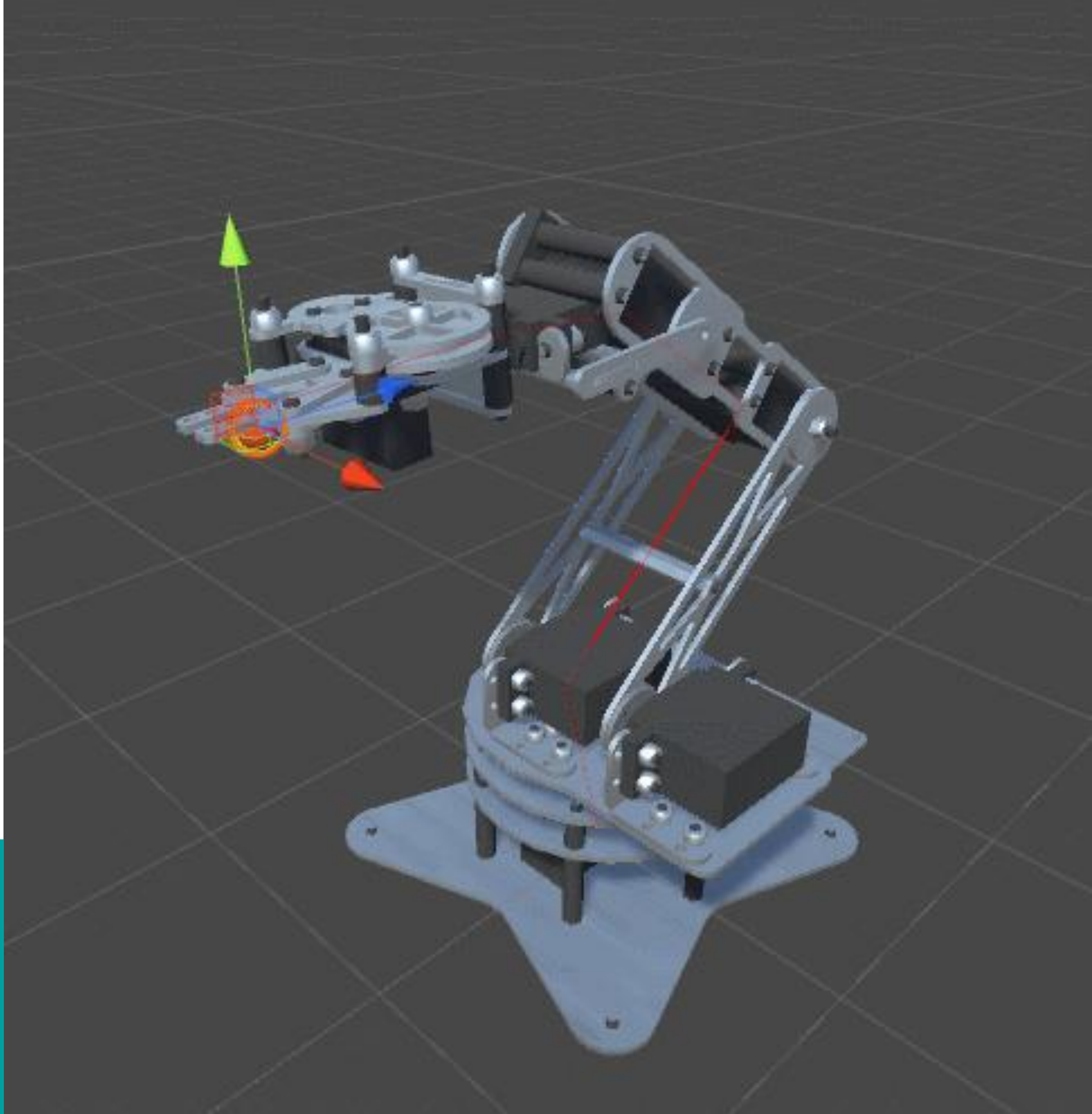


$$\nabla f_{\alpha_0} = (\alpha_0, \alpha_1, \alpha_2) = \frac{f(\alpha_0 + \Delta_x, \alpha_1, \alpha_2) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta x}$$

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

Joint Angle

Limitation Problems!



How do we fix this?

Fast IK Planning