



CSCI/ECEN 3302

Introduction to Robotics

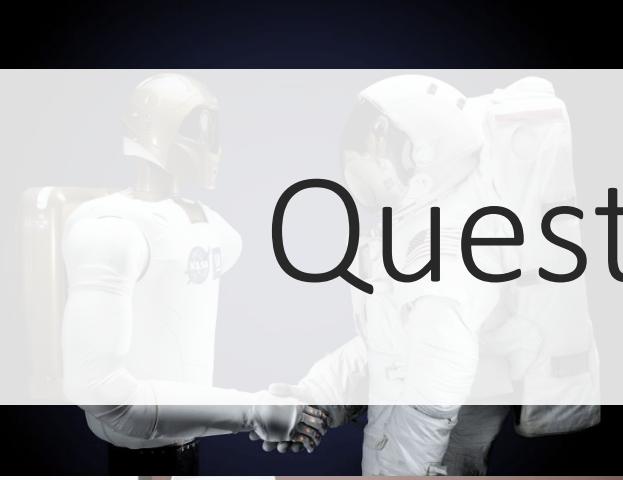
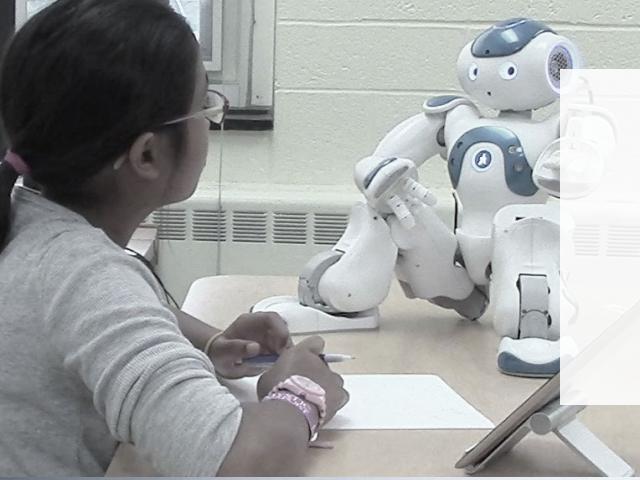
Alessandro Roncone

aroncone@colorado.edu

<https://hiro-group.ronc.one>

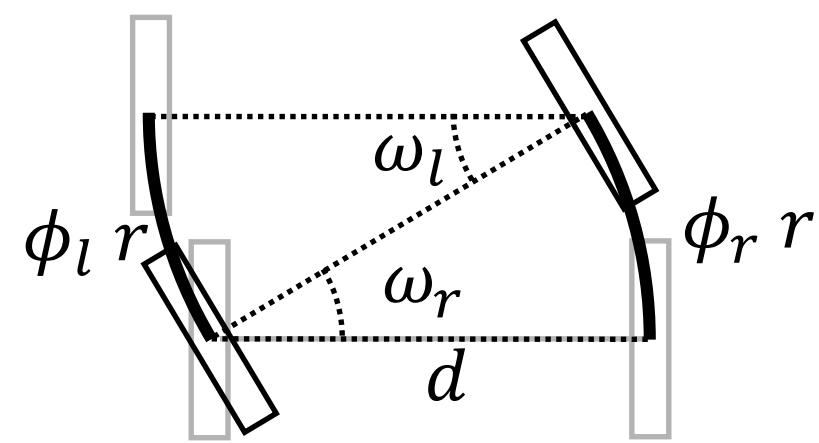
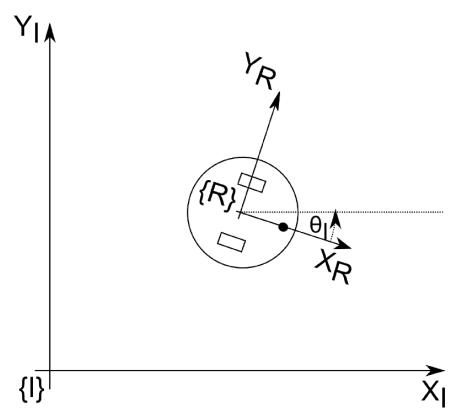
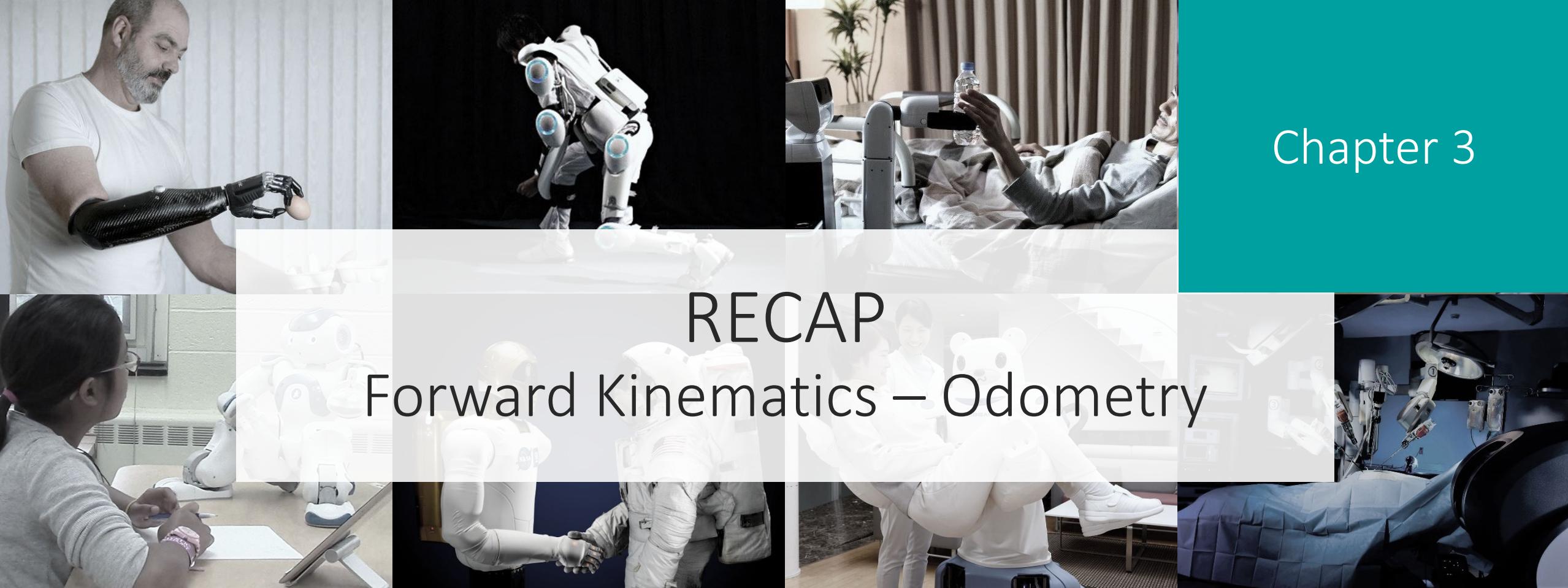
Administrivia

- Lab 2 – due **Thursday!!**
 - Tomorrow (Wednesday):
 - if you need help with Lab 2, we are happy to provide support, BUT
 - The expectation is that the majority of the class will start working on Lab 3
- HW1 – due **Sunday!!**
- Lab 3
 - We will make it 2 weeks long
 - We will work on it tomorrow and next Wed
- Lab 4 will be 1-week long!

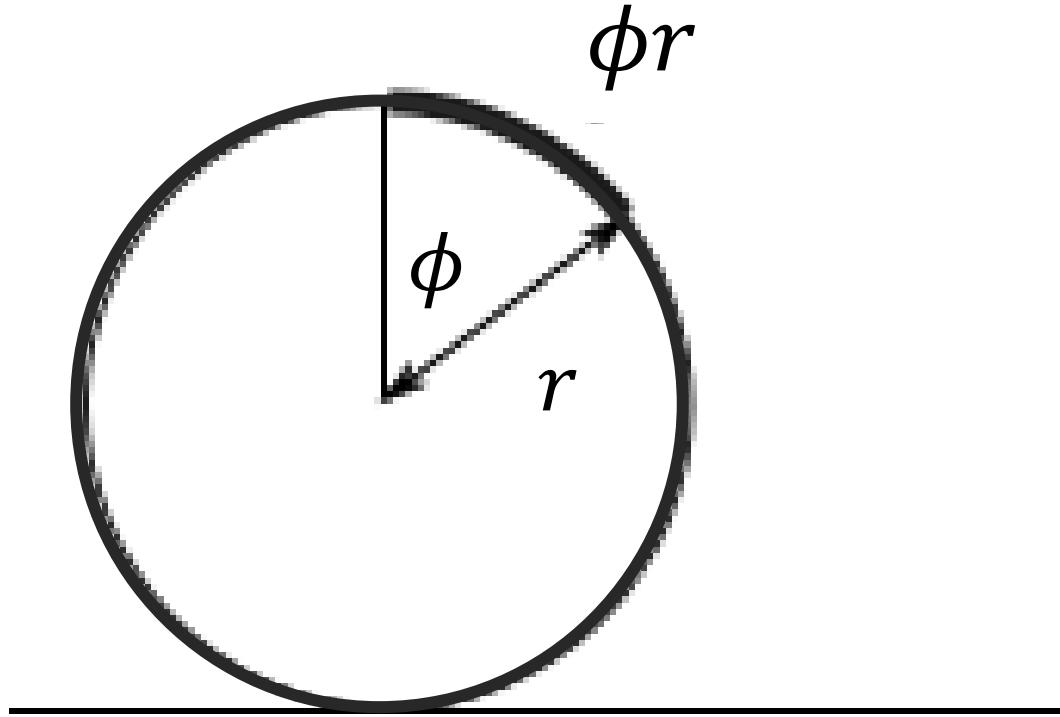


Questions?

Chapter 3



1. Wheel motion

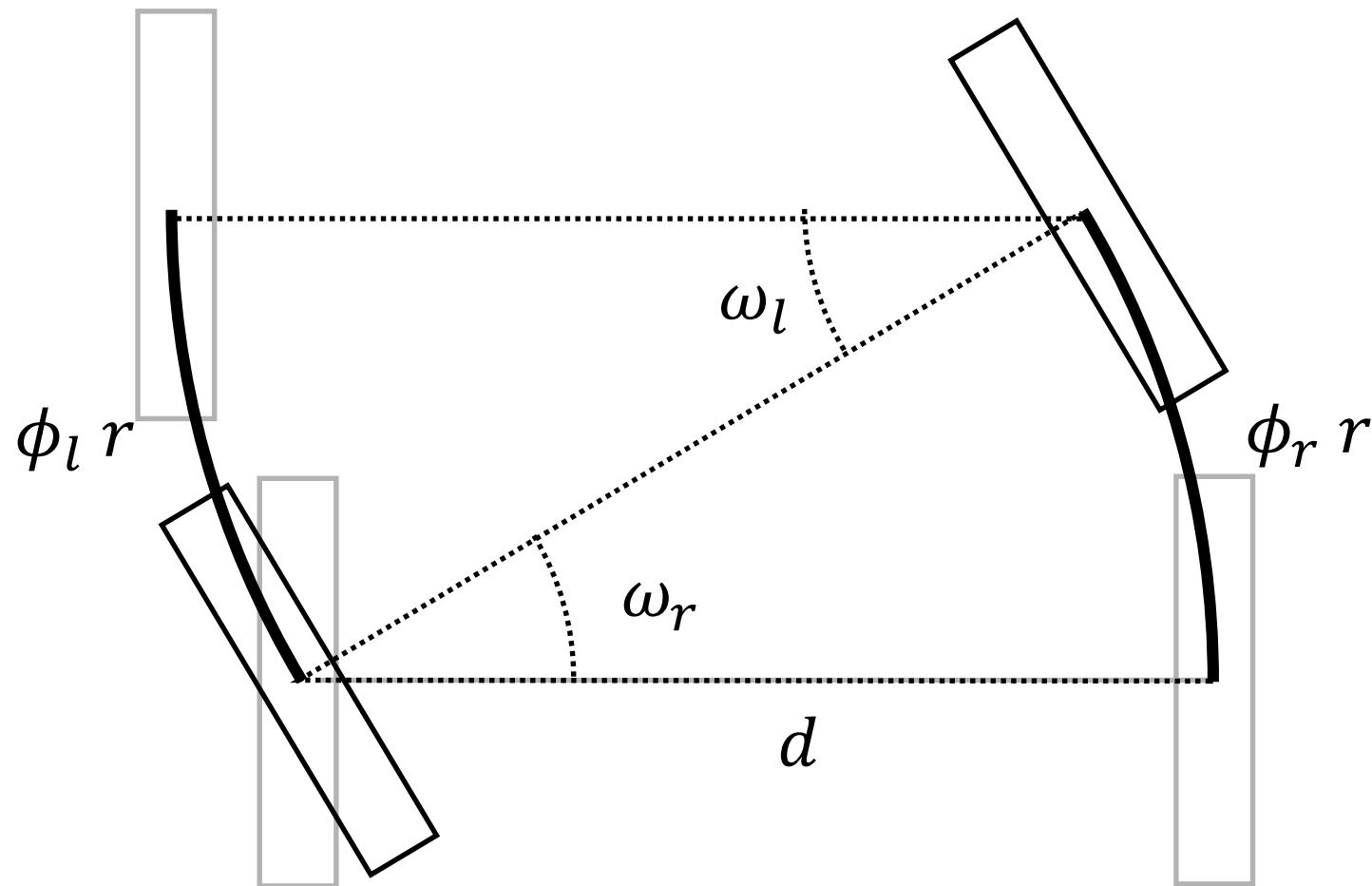


Distance traveled is angle of rotation times wheel radius:

$$x = r\phi$$

$$\dot{x} = r\dot{\phi}$$

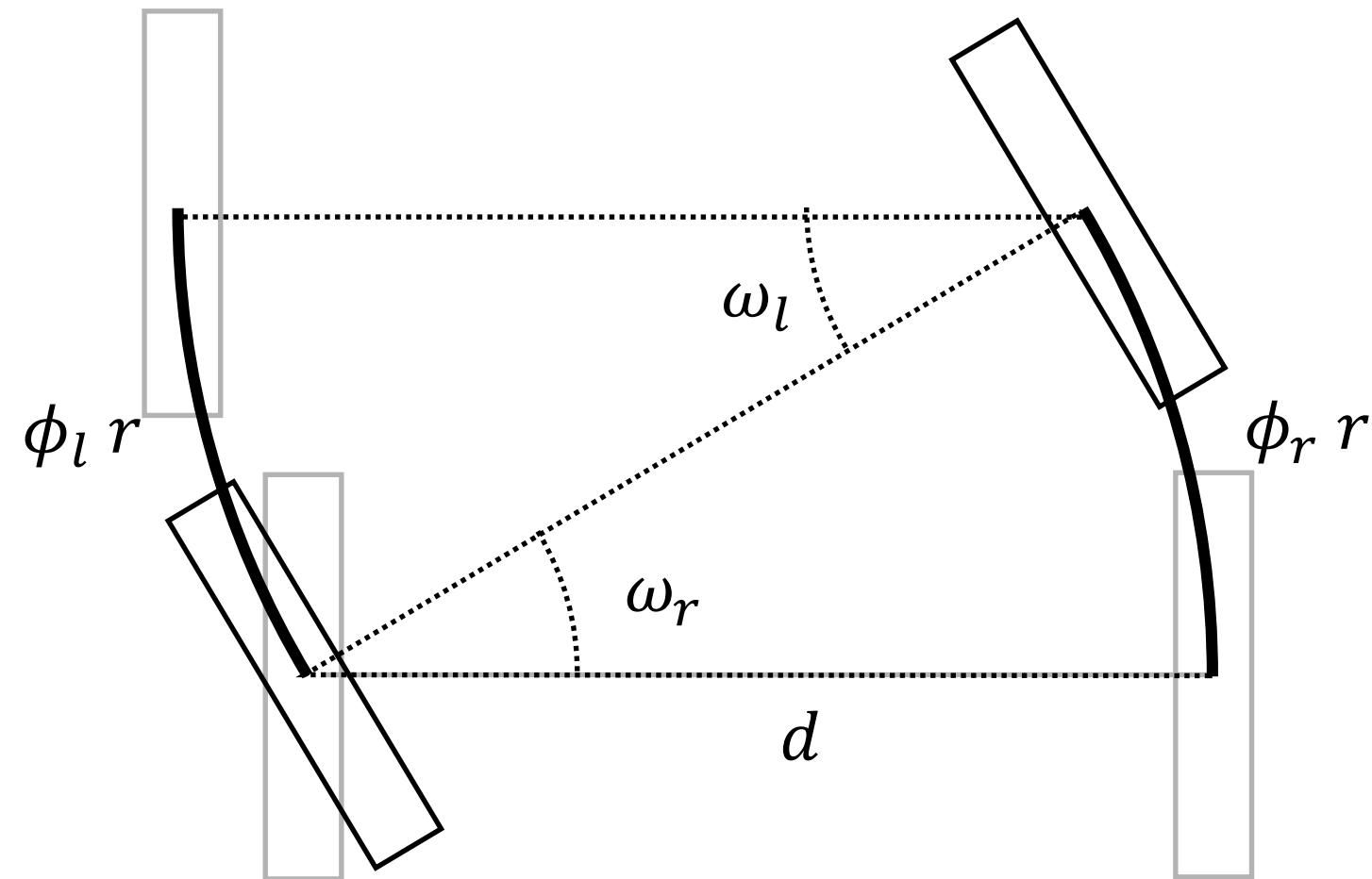
2. Wheel motions → Position Updates



What about the case where both wheels are moving at different speeds $\dot{\phi}_l$ and $\dot{\phi}_r$?

$$\dot{x}_r = \frac{r\dot{\phi}_l}{2} + \frac{r\dot{\phi}_r}{2}$$

2. Wheel motions → Position Updates

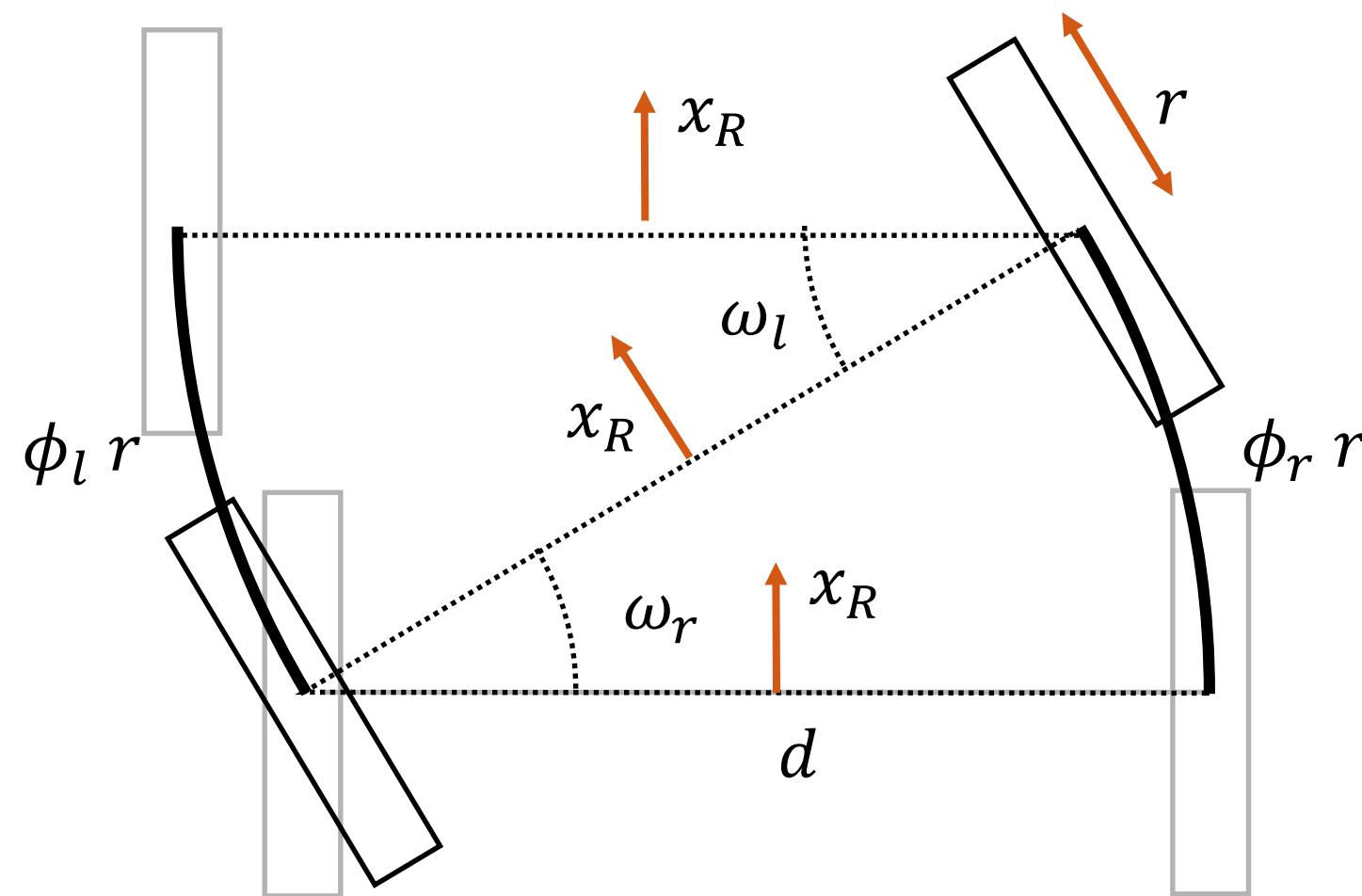


$$\dot{\omega}_r = \frac{\dot{\phi}_r r}{d}$$

$$\dot{\omega}_l = \frac{\dot{\phi}_l r}{d}$$

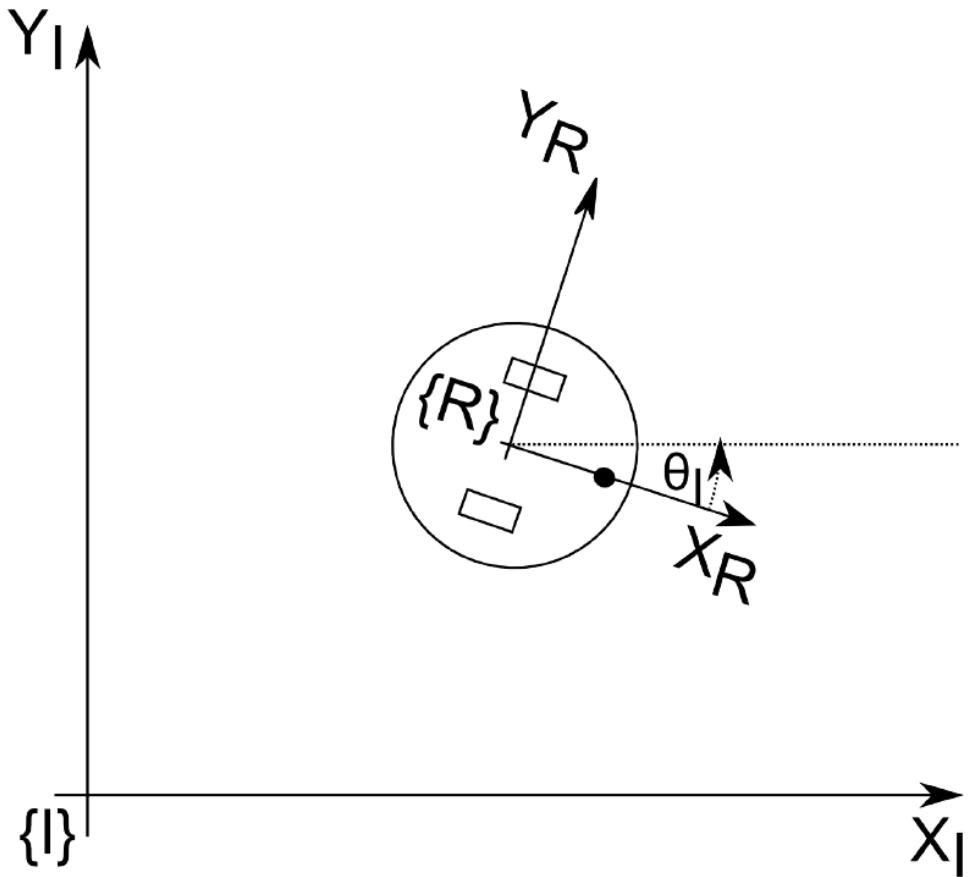
$$\dot{\theta} = \frac{\dot{\phi}_r r}{d} - \frac{\dot{\phi}_l r}{d}$$

2. Forward Kinematics of mobile robot



$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} \frac{r\dot{\phi}_l}{2} + \frac{r\dot{\phi}_r}{2} \\ 0 \\ \frac{\dot{\phi}_r r}{d} - \frac{\dot{\phi}_l r}{d} \end{bmatrix}$$

3. Forward Kinematics + Odometry



$$\dot{x}_{I,x} = \cos(\theta) \dot{x}_R$$

$$\dot{x}_{I,y} = -\sin(\theta) \dot{y}_R$$

$$\dot{x}_I = \cos(\theta) \dot{x}_R - \sin(\theta) \dot{y}_R$$

$$\dot{y}_I = \sin(\theta) \dot{x}_R + \cos(\theta) \dot{y}_R$$

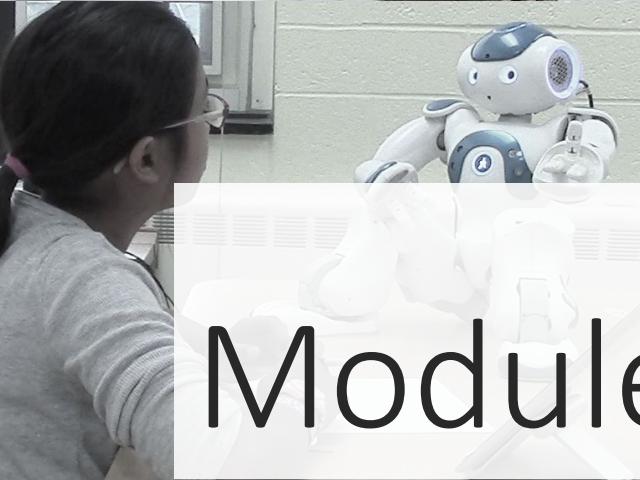
$$\dot{\theta}_I = \dot{\theta}_R$$

$$\begin{pmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{r\dot{\phi}_l}{2} + \frac{r\dot{\phi}_r}{2} \\ 0 \\ \frac{\dot{\phi}_r r}{d} - \frac{\dot{\phi}_l r}{d} \end{pmatrix}$$

4. Speeds → How can we compute positions?

$$\begin{pmatrix} x_I(T) \\ y_I(T) \\ \theta(T) \end{pmatrix} = \int_0^T \begin{pmatrix} \dot{x}_I(t) \\ \dot{y}_I(t) \\ \dot{\theta}(t) \end{pmatrix} dt \approx \sum_{k=0}^{k=T} \begin{pmatrix} \Delta x_I(k) \\ \Delta y_I(k) \\ \Delta \theta(k) \end{pmatrix} \Delta t$$

Chapter 3

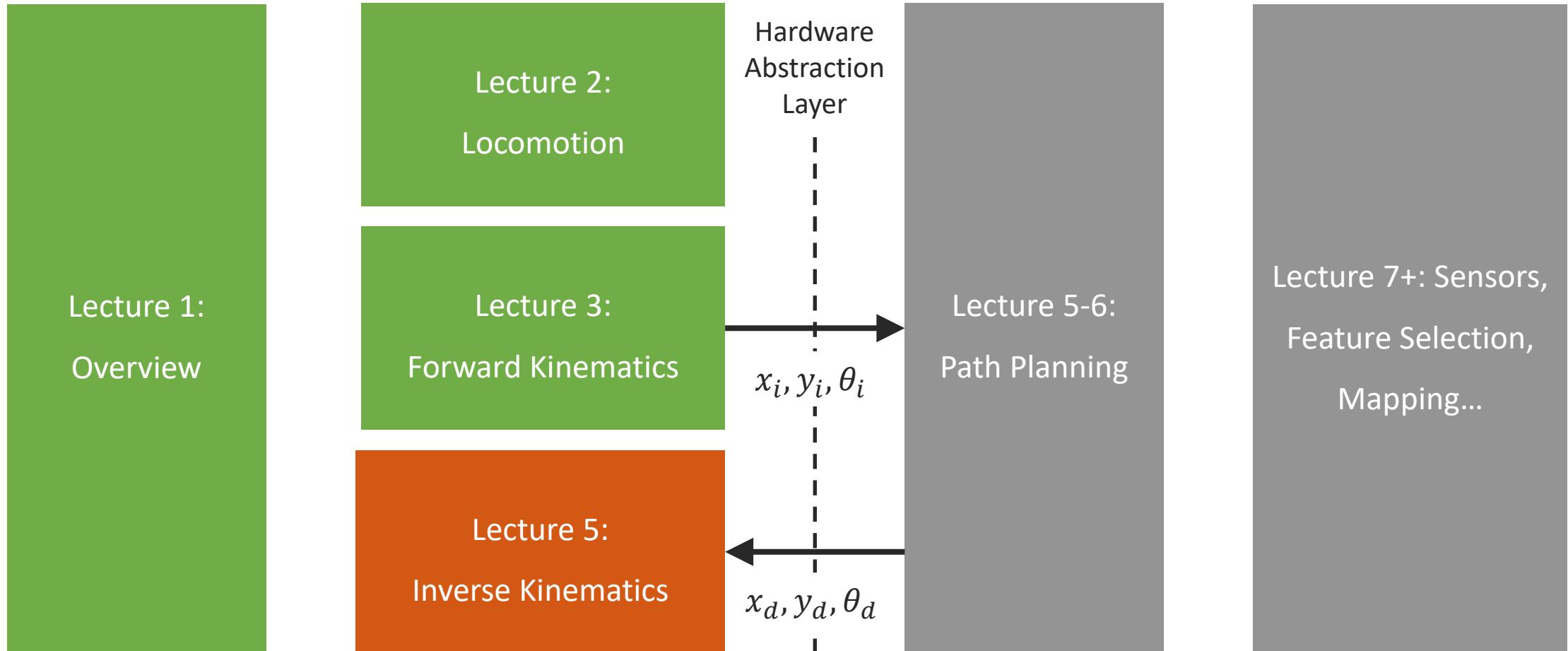


Module 1 – MECHANISMS

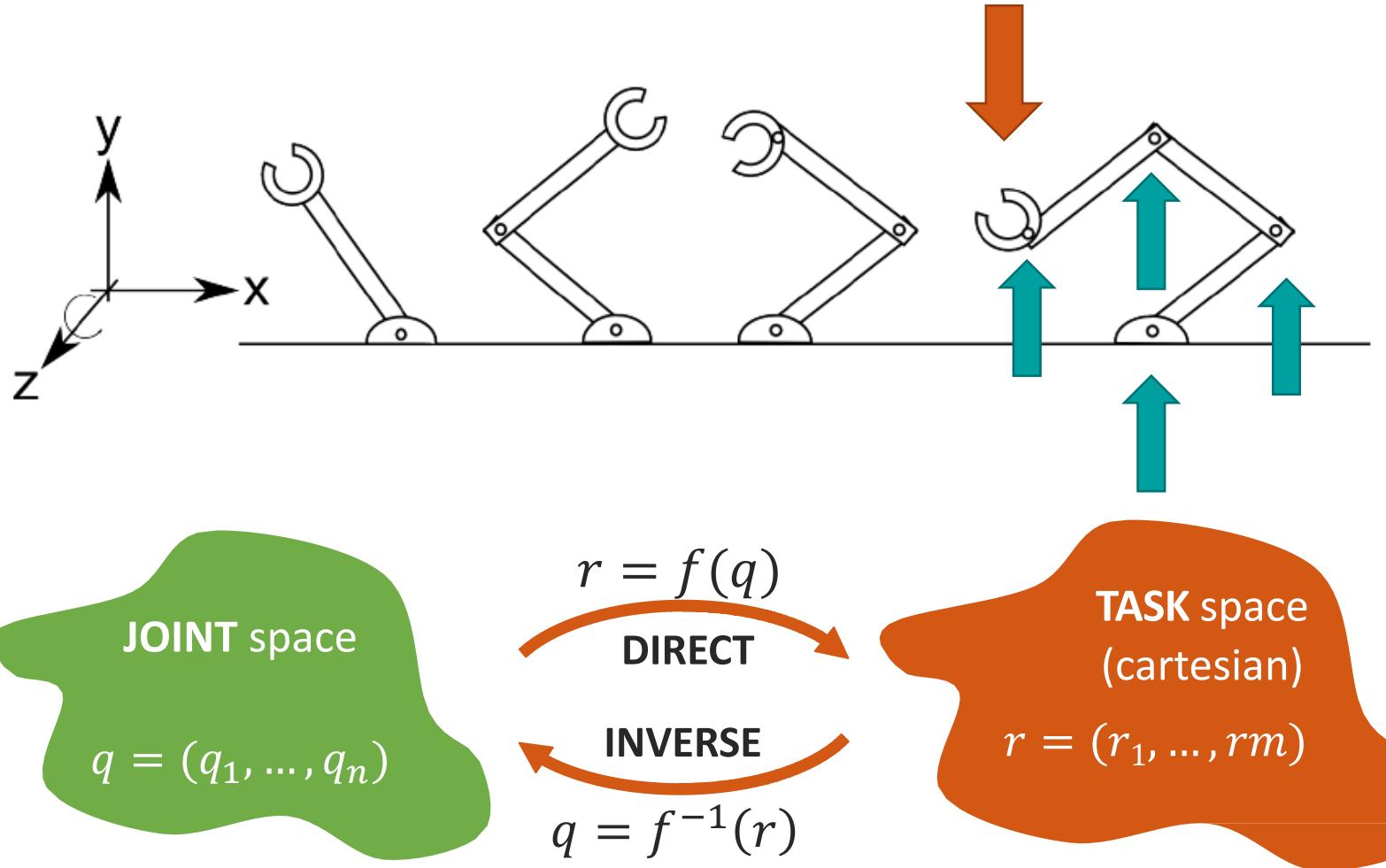
Part III – Inverse Kinematics



Roadmap



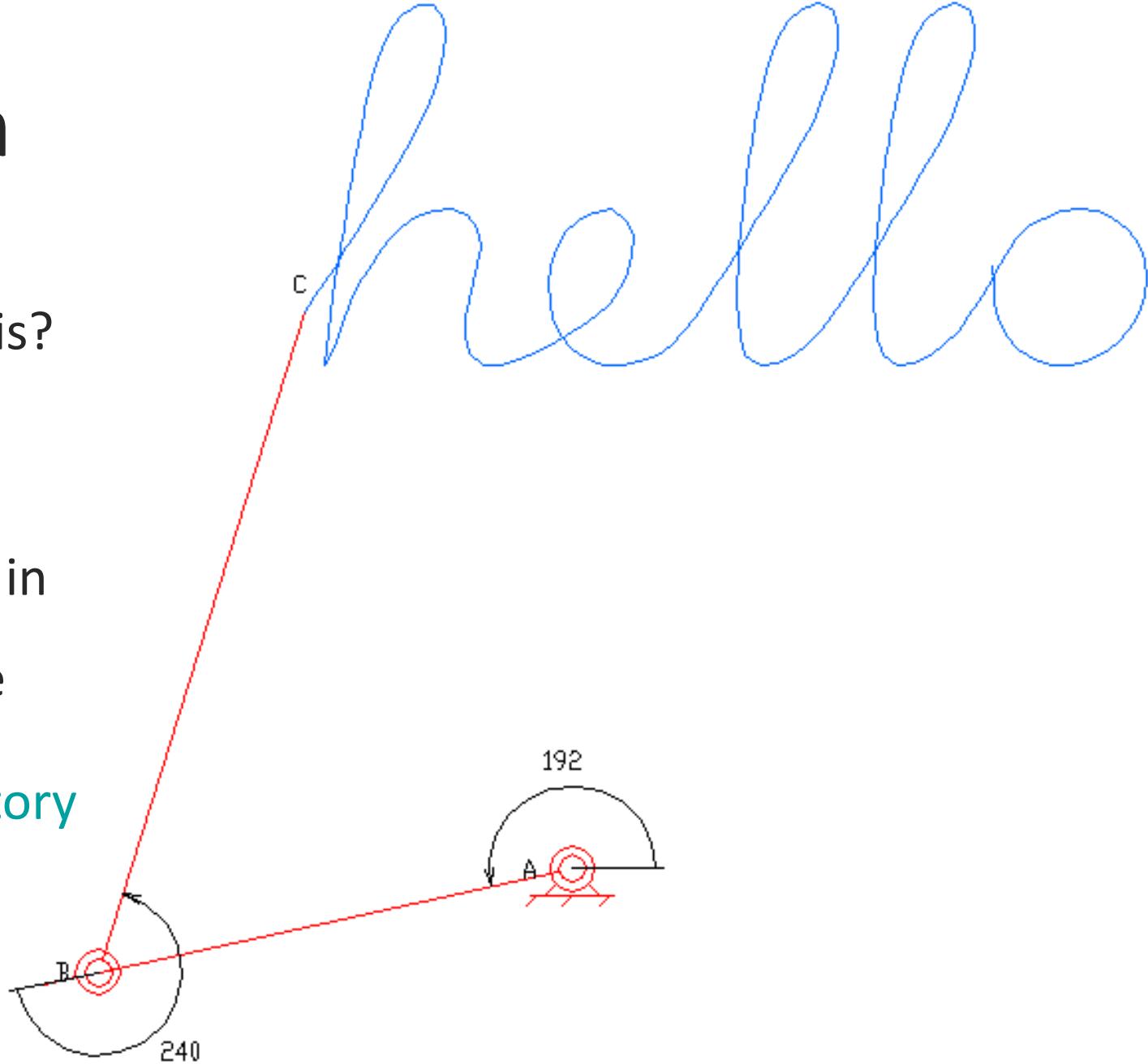
Inverse Kinematics



Motivating Problem

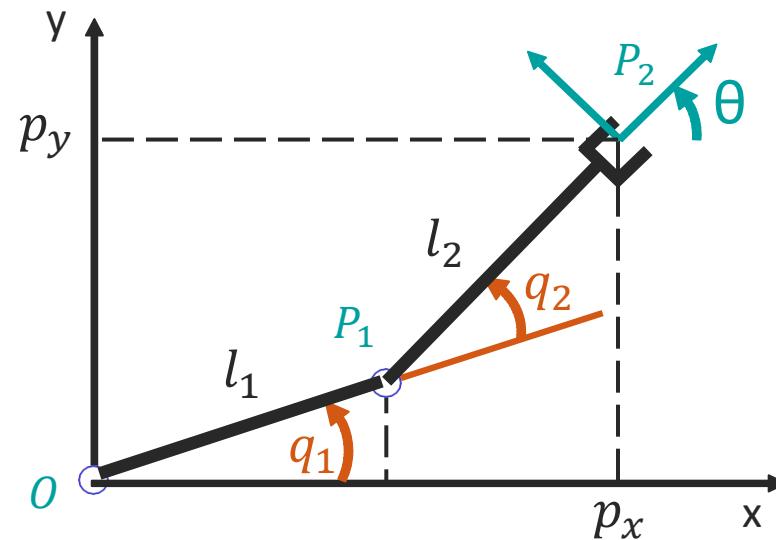
How can we get a robot to do this?

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



Inverse Kinematics

IK for manipulator



Given point (x, y, θ)

Find angles (q_1, q_2)

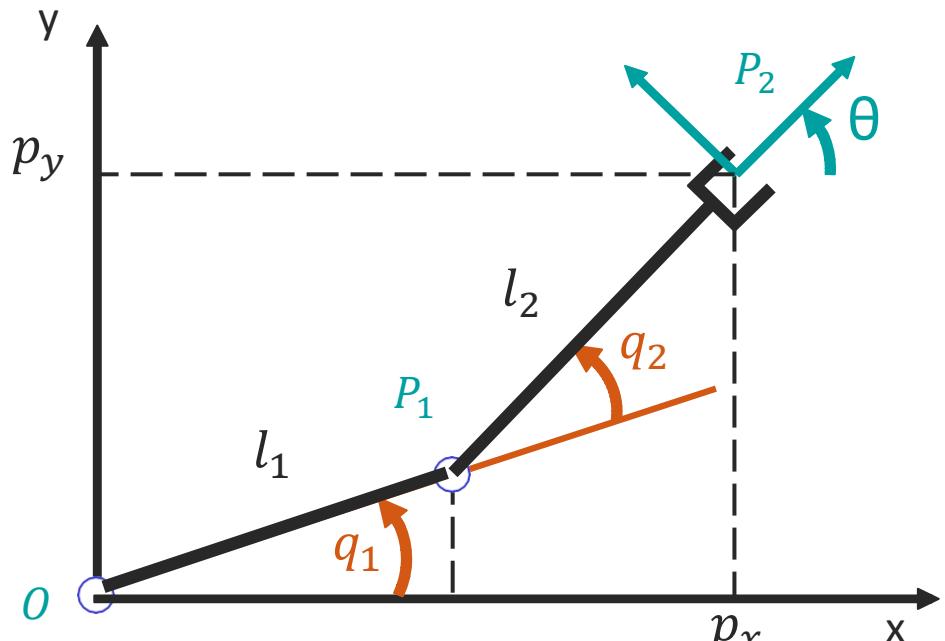
IK for e-puck



Given: (x, y, θ)

Find: (ϕ_l, ϕ_r)

Example: Inverse Kinematics of a 2R arm



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

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

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

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

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

⇓

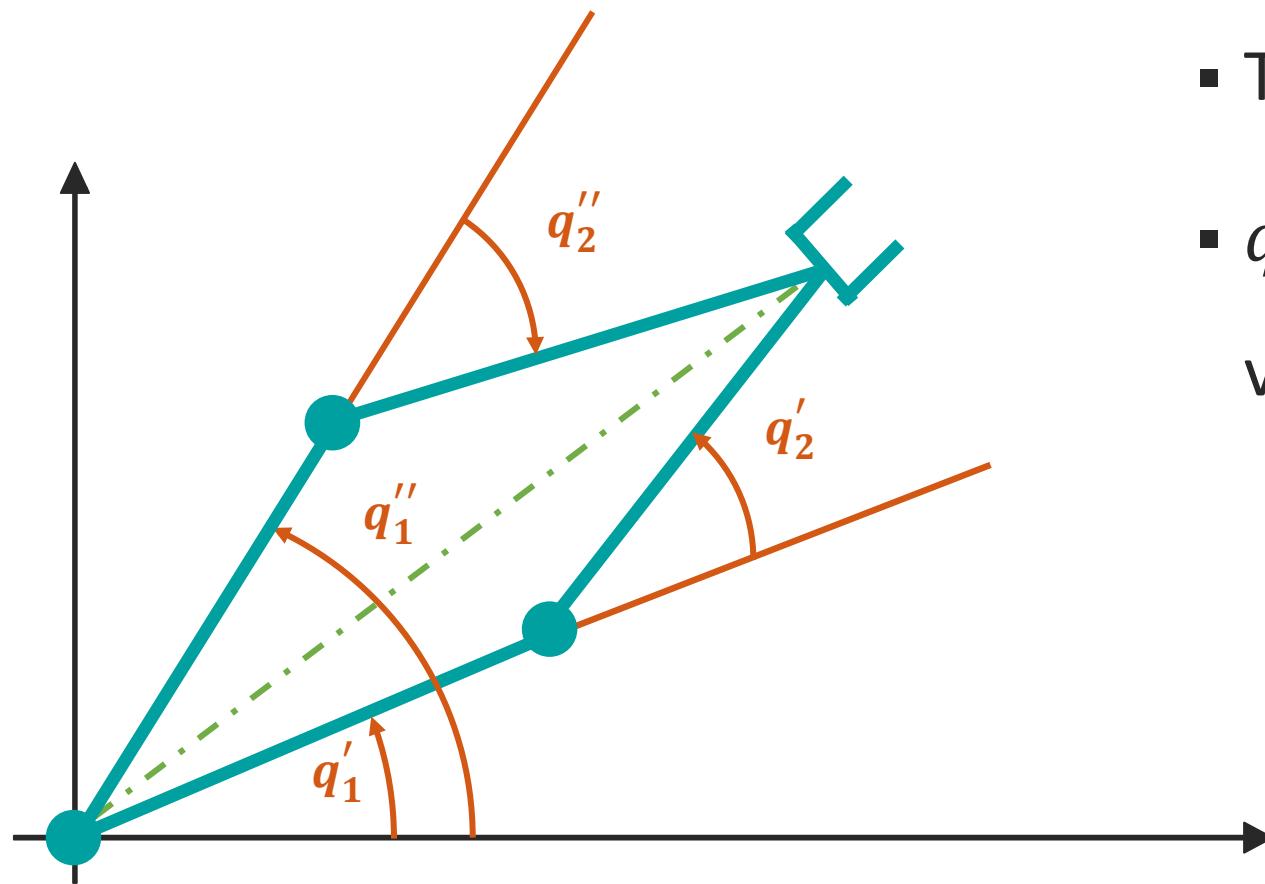
$$q_1 = \cos^{-1} \frac{x^2 y + y^3 - \sqrt{4x^4 - x^6 + 4x^2y^2 - 2x^4y^2 - x^2y^4}}{2(x^2 + y^2)}$$

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

$$r=f(q)$$

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

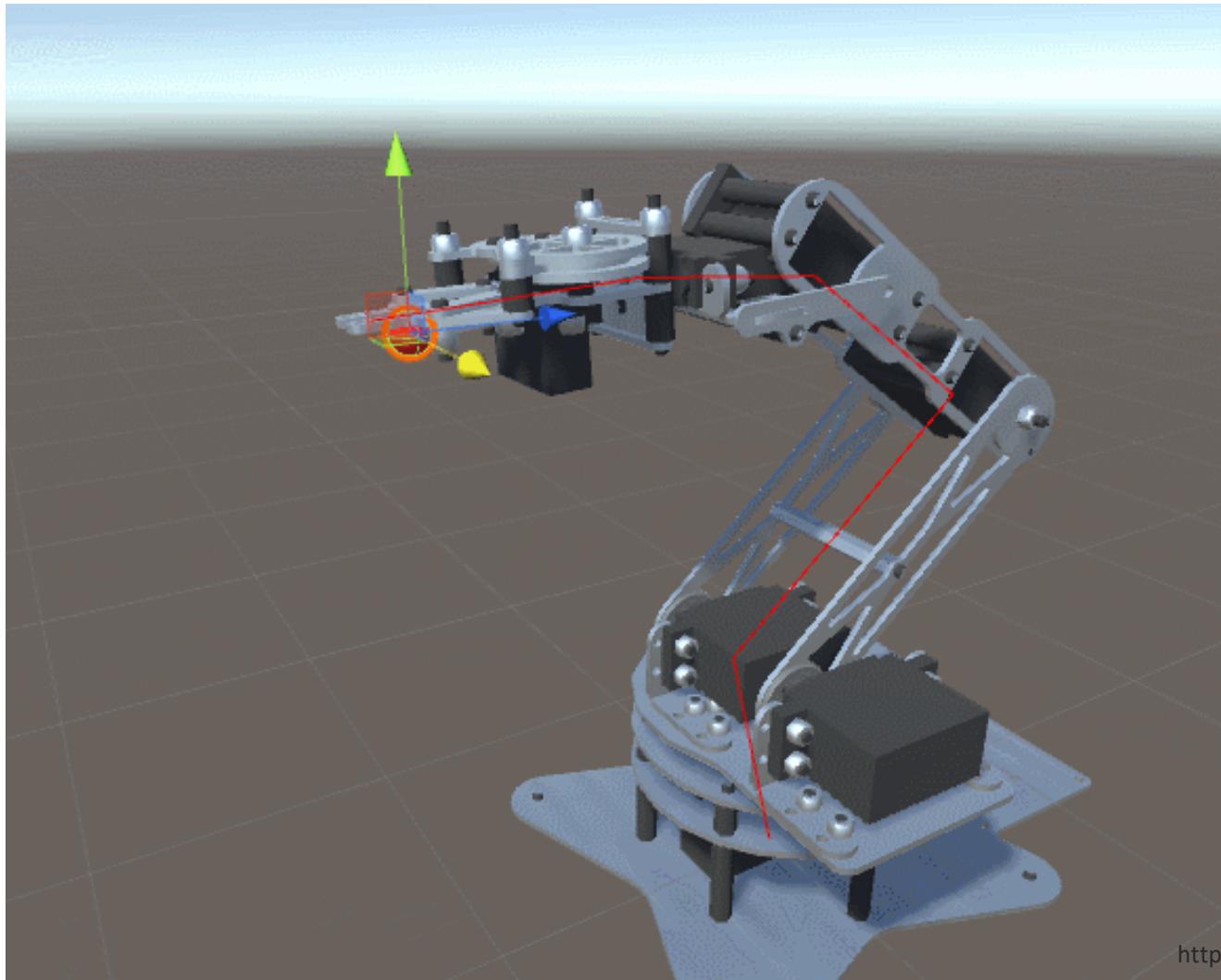
Example: Inverse Kinematics of a 2R arm



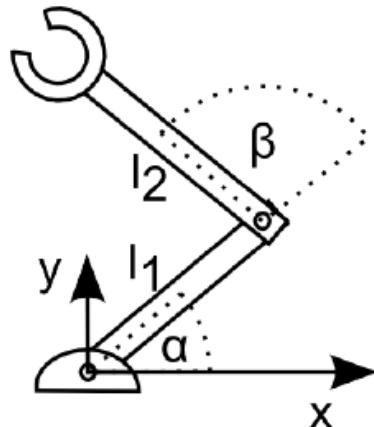
- Two solutions!
- q_1' and q_2'' have same absolute value, but different sign

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

End-effector Position Control



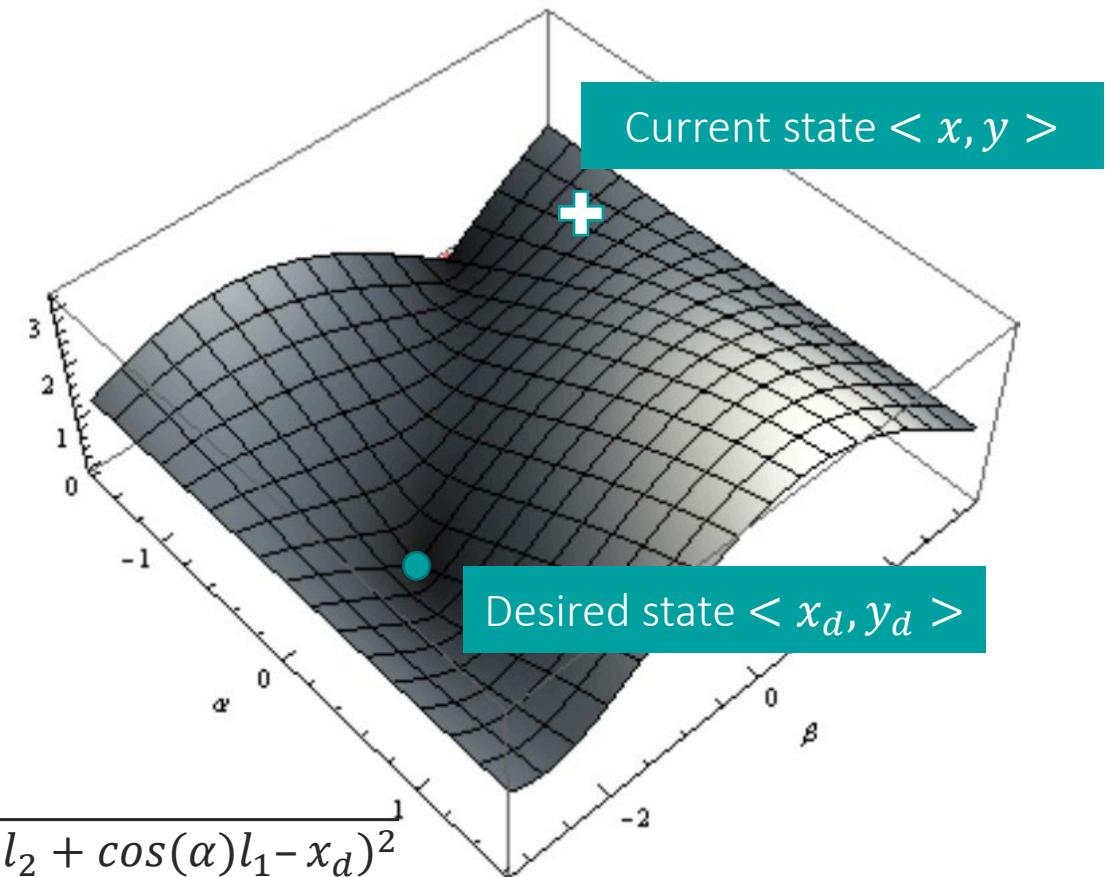
Easier ways to solve the IK problem



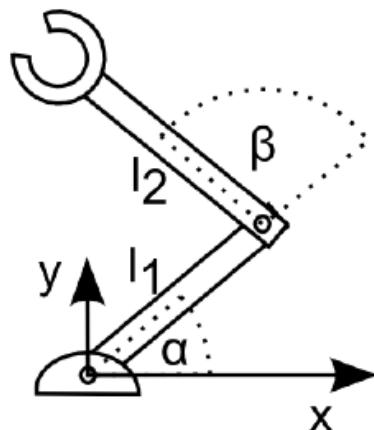
$$x = l_1 \cos(\alpha) + l_2 \cos(\alpha + \beta)$$
$$y = l_1 \sin(\alpha) + l_2 \sin(\alpha + \beta)$$

Just the Euclidean distance
between two vectors!

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



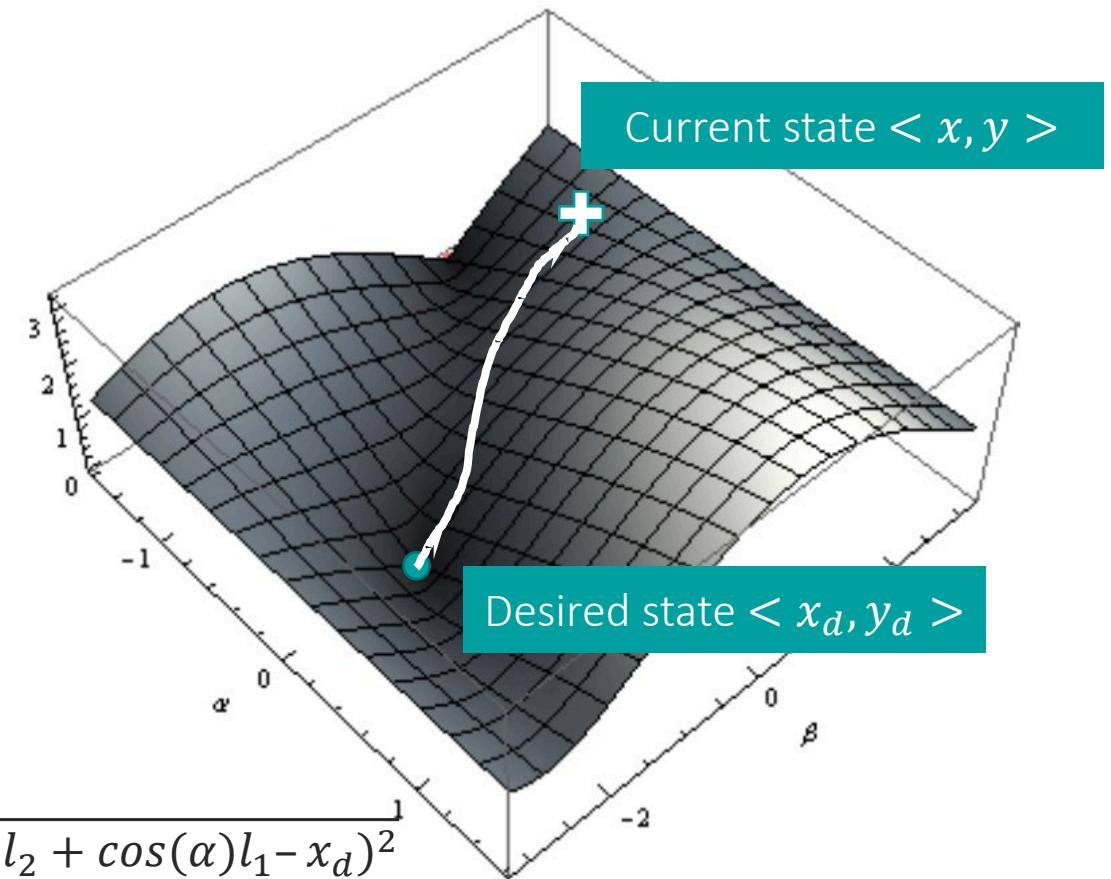
Motion Planning in EE Space



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

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

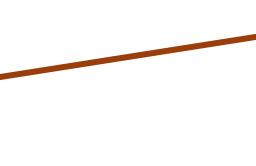
Optimization methods tend to **require very little knowledge** about the parameter space or domain

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

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

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



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

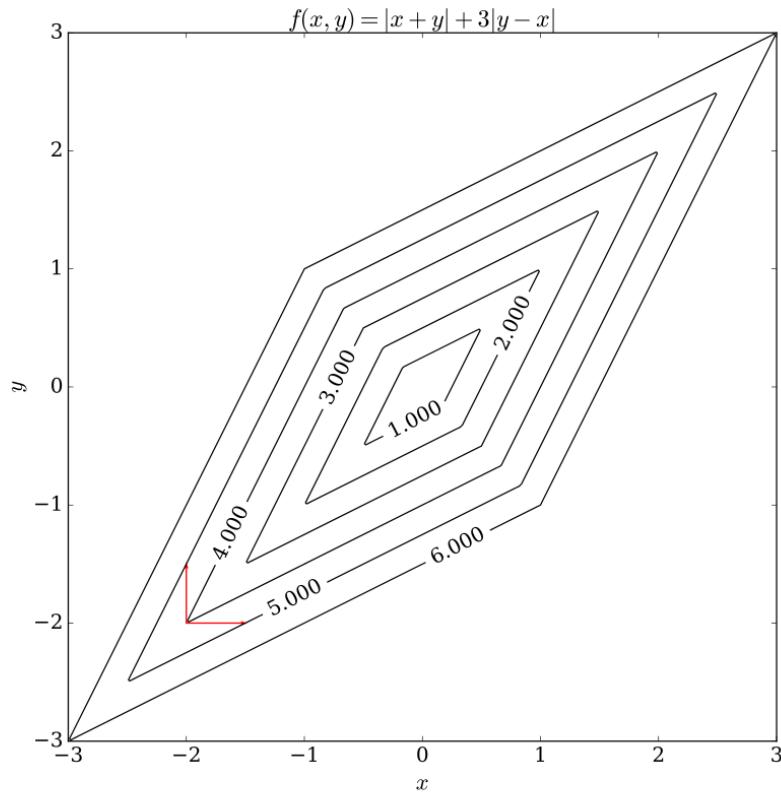
$$F(x^0) \geq F(x^1) \geq F(x^2) \geq \dots$$



Each iteration reduces our error or remains stationary

Coordinate Descent

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

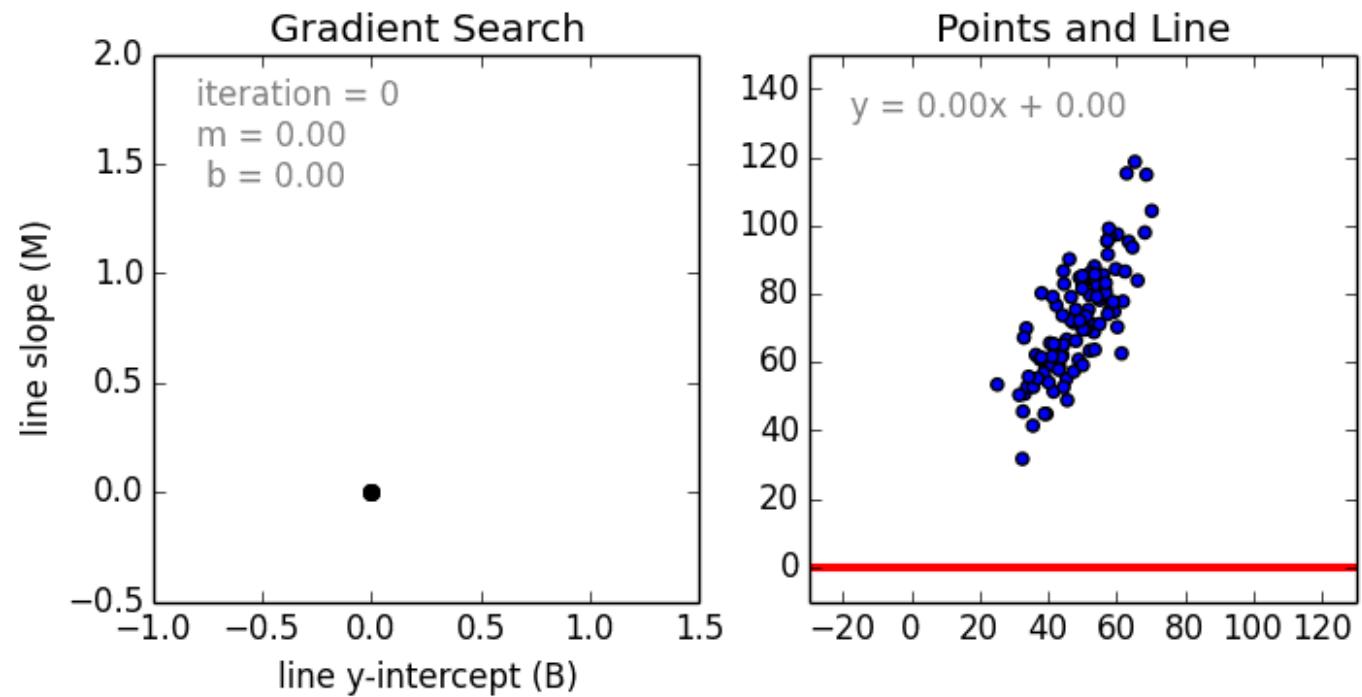


- Choose an initial parameter vector \boldsymbol{x}
- Until convergence is reached (or for some fixed number of iterations):
 - Choose an index i from 1 to n
 - Choose a step size α
 - Update x_i to $x_i - \alpha \frac{\partial F}{\partial x_i}(\boldsymbol{x})$

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

$$f: \mathbb{R}^3 \rightarrow \mathbb{R}$$

Joint angles

$$\alpha_i$$

Learning rate

$$L$$

How do we Move the Robot?

- Linear equations dictate end-effector position:

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

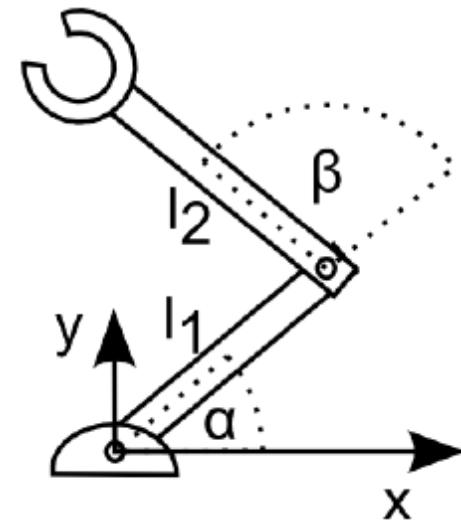
Forward Kinematics Equations

- 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 = $J \cdot \dot{q}$



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

- $\frac{dp_e}{dt} = J \frac{dq}{dt}$, or in other words , $v_e = J \cdot \dot{q}$
- $\dot{q} = J^{-1} \cdot [v_{e,d} + K(p_{e,d} - p)]$

\dot{q} : Change in Configuration space

K: gain

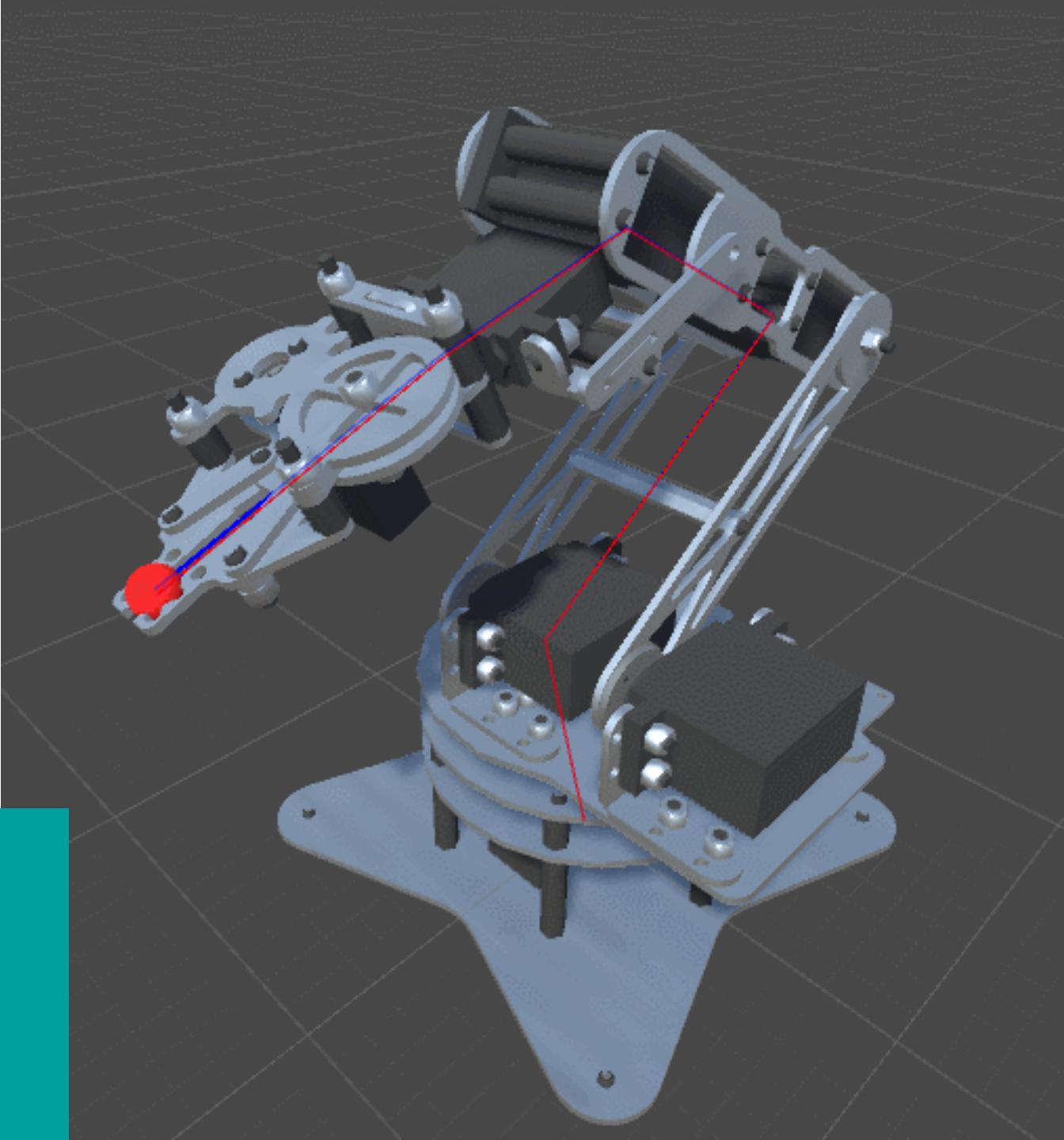
$v_{e,d}$: Desired velocity

$p_{e,d}$: Desired position

Convergence Problems!

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

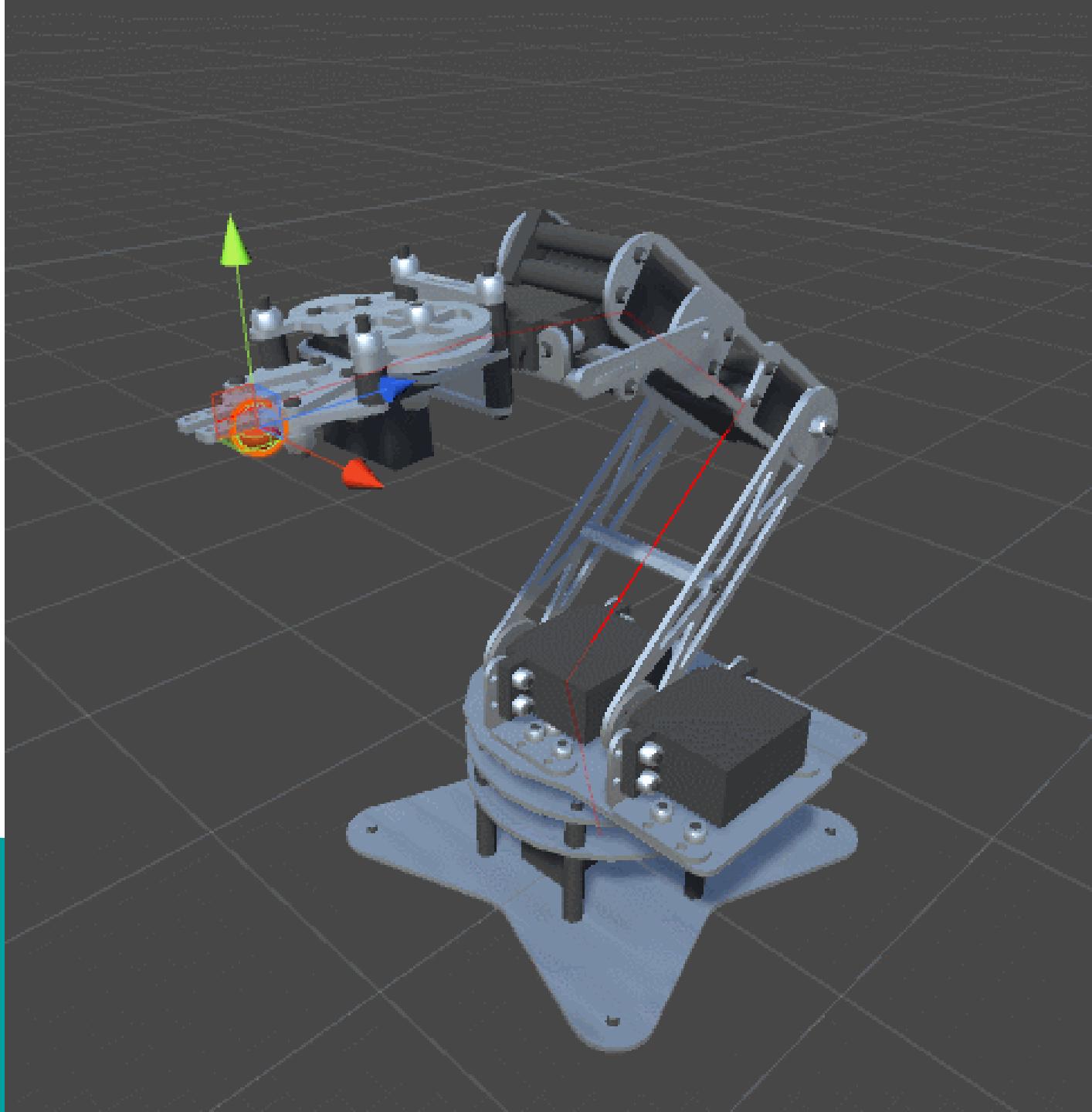
$$\alpha_0 \leftarrow \alpha_0 - L \nabla f_{\alpha_0}(\alpha_0, \alpha_1, \alpha_2)$$



Joint Angle

Limitation Problems!

How do we fix this?



Fast IK Planning

