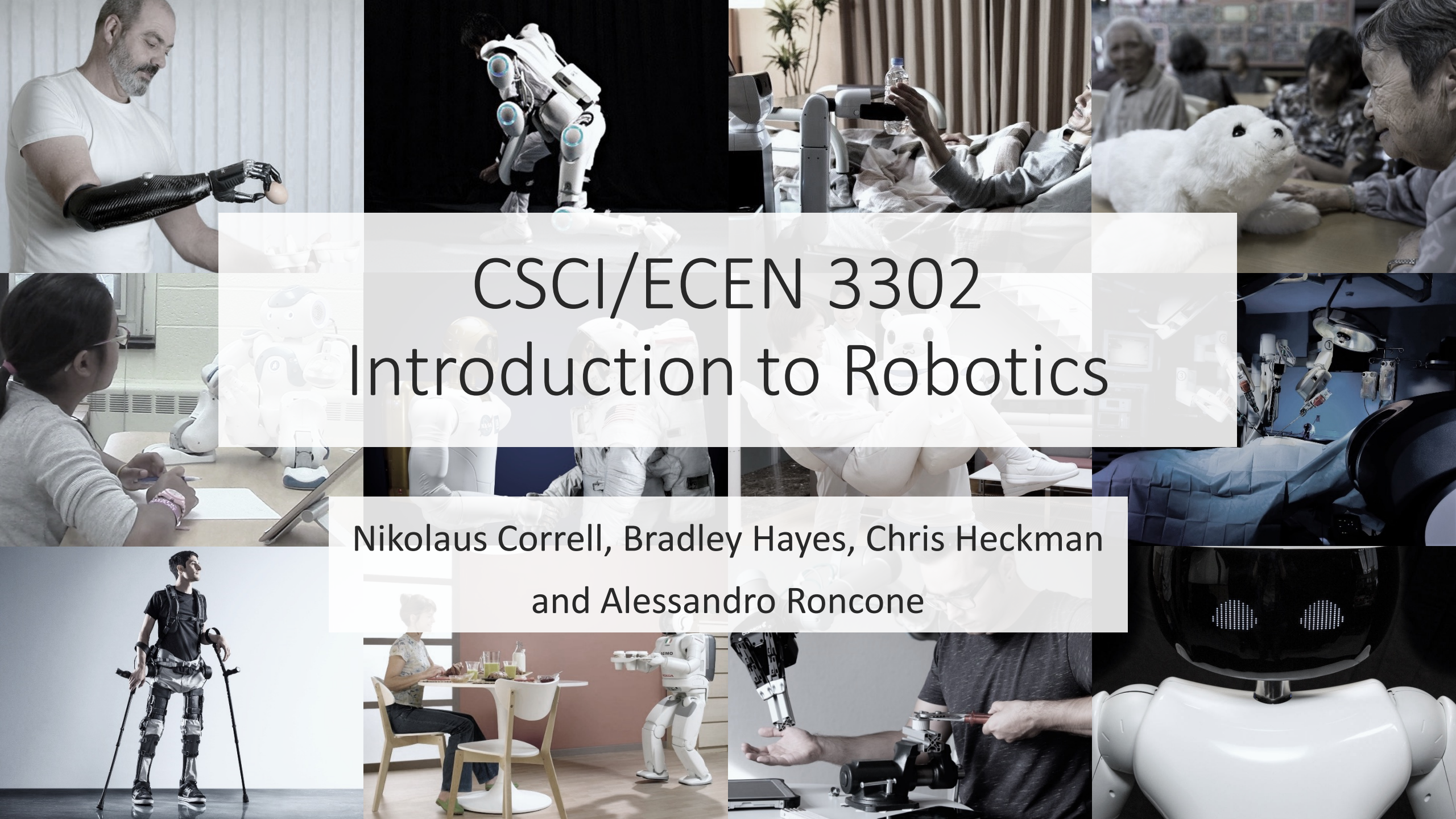


CSCI/ECEN 3302
Introduction to Robotics

Nikolaus Correll, Bradley Hayes, Chris Heckman
and Alessandro Roncone

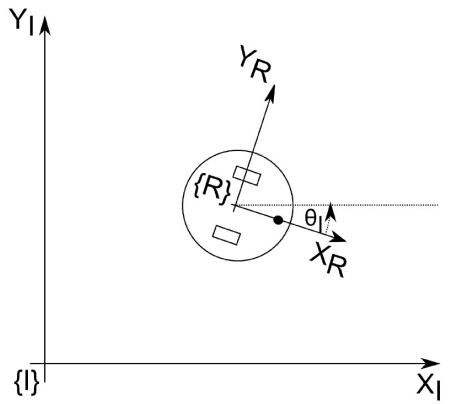
Administrivia

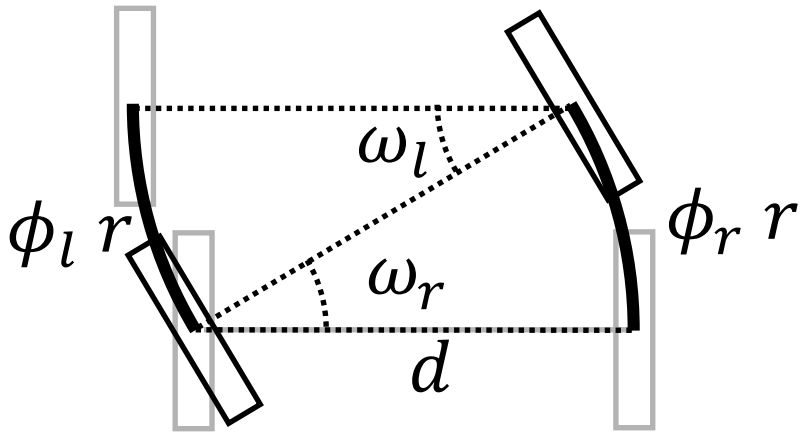
- Lab 2 – due September 20
  - Tomorrow (Wednesday):
    - Your robot should be able to run in circles
    - Focus on implementing odometry code
- HW1 – due September 19
  - More info today

Questions?

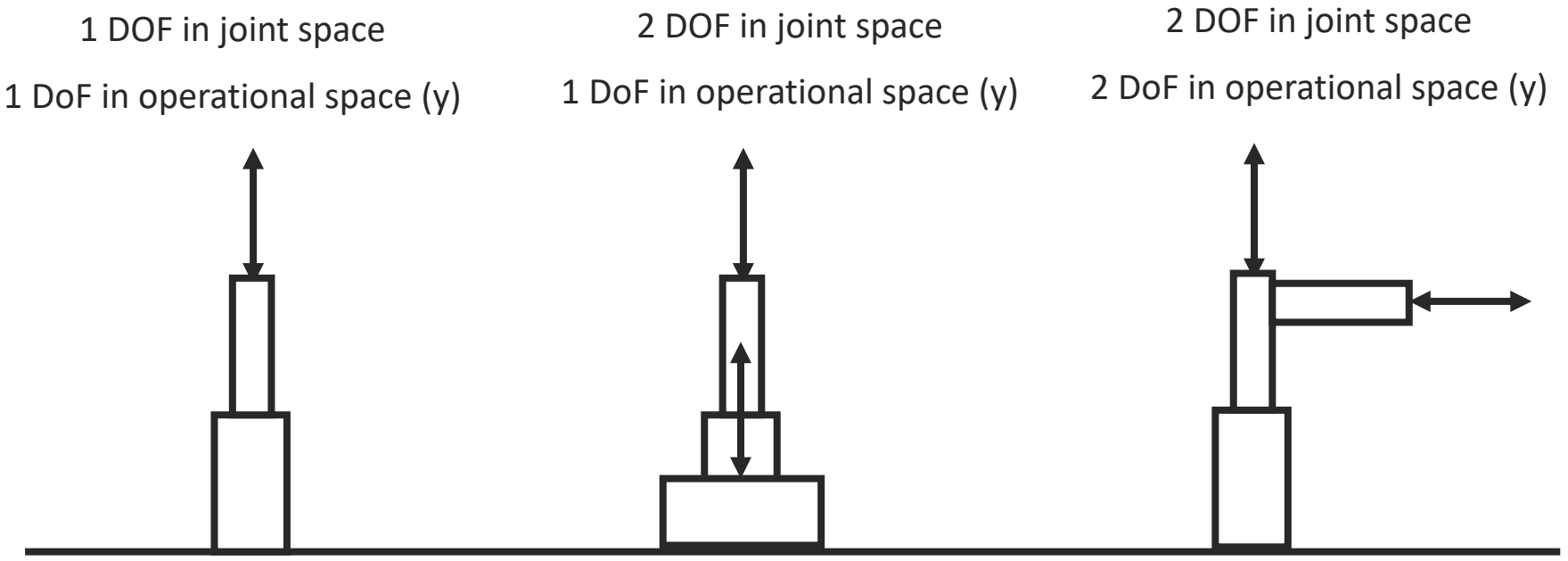
Chapter 3

RECAP
Forward Kinematics – Odometry



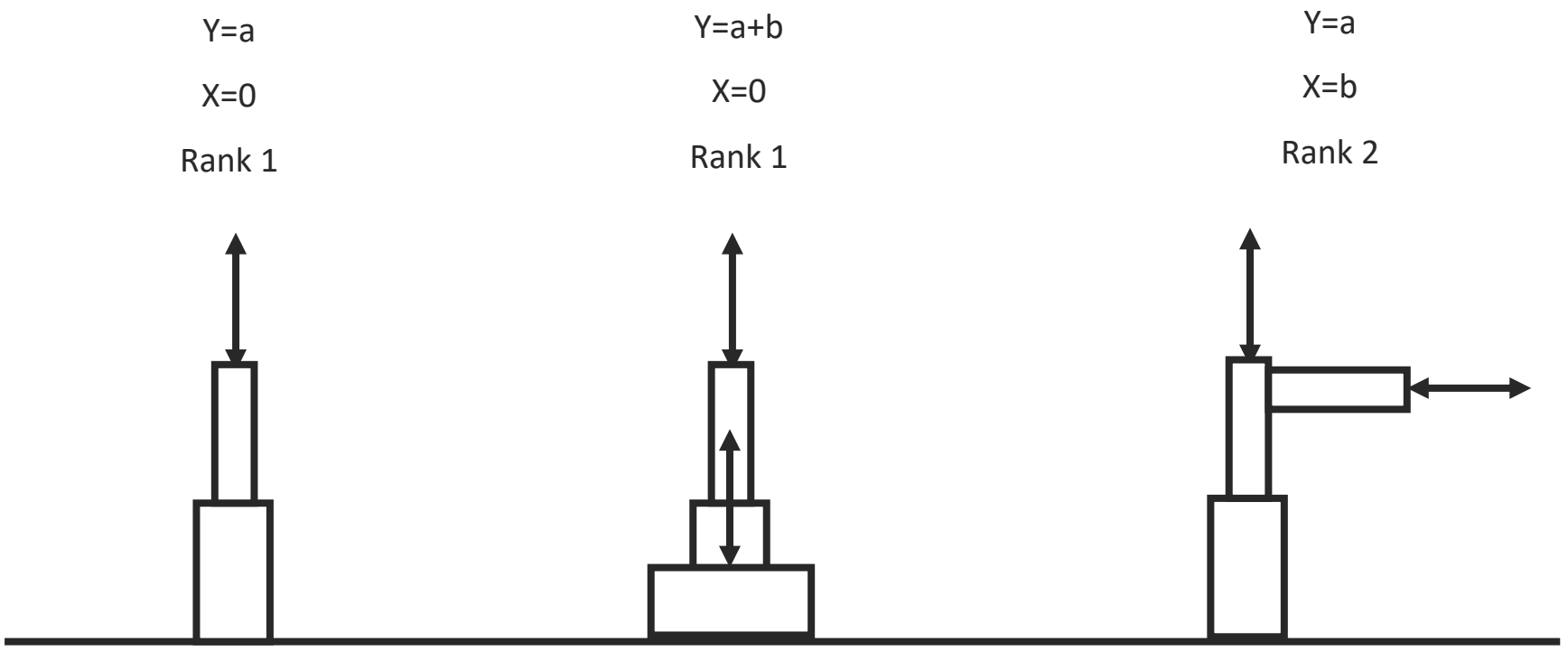


Recap: Degrees of Freedom



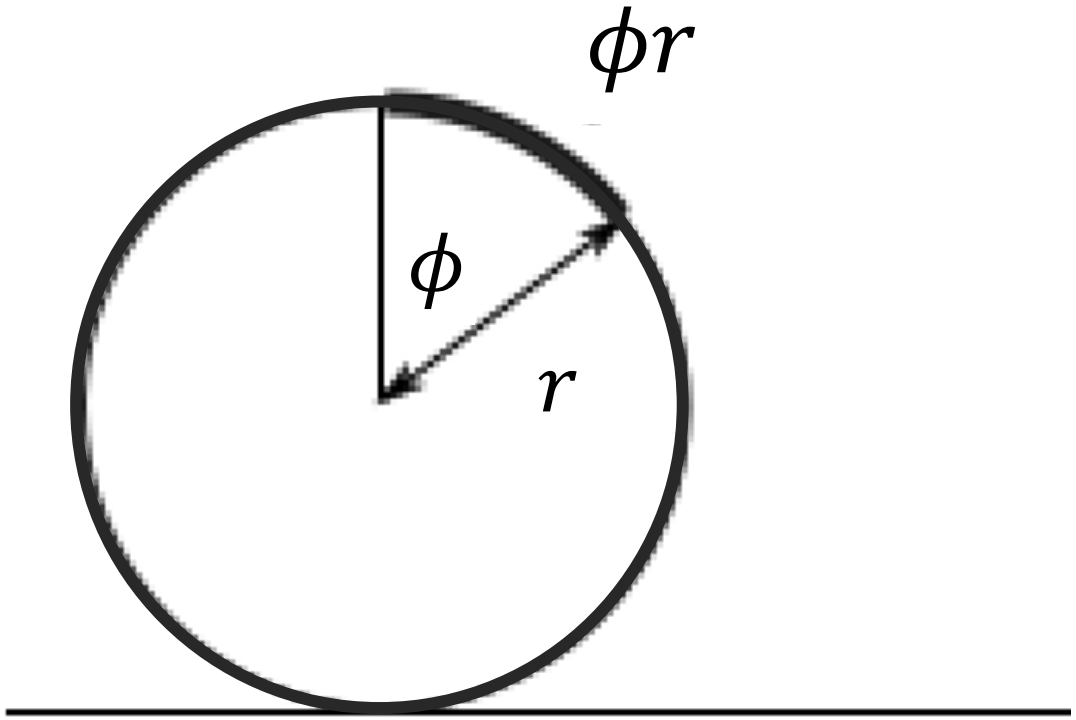
DoF in joint space aka “mobility” of the system

Degrees of Freedom



DoF in joint space aka “mobility” of the system

Recap: Wheel motion

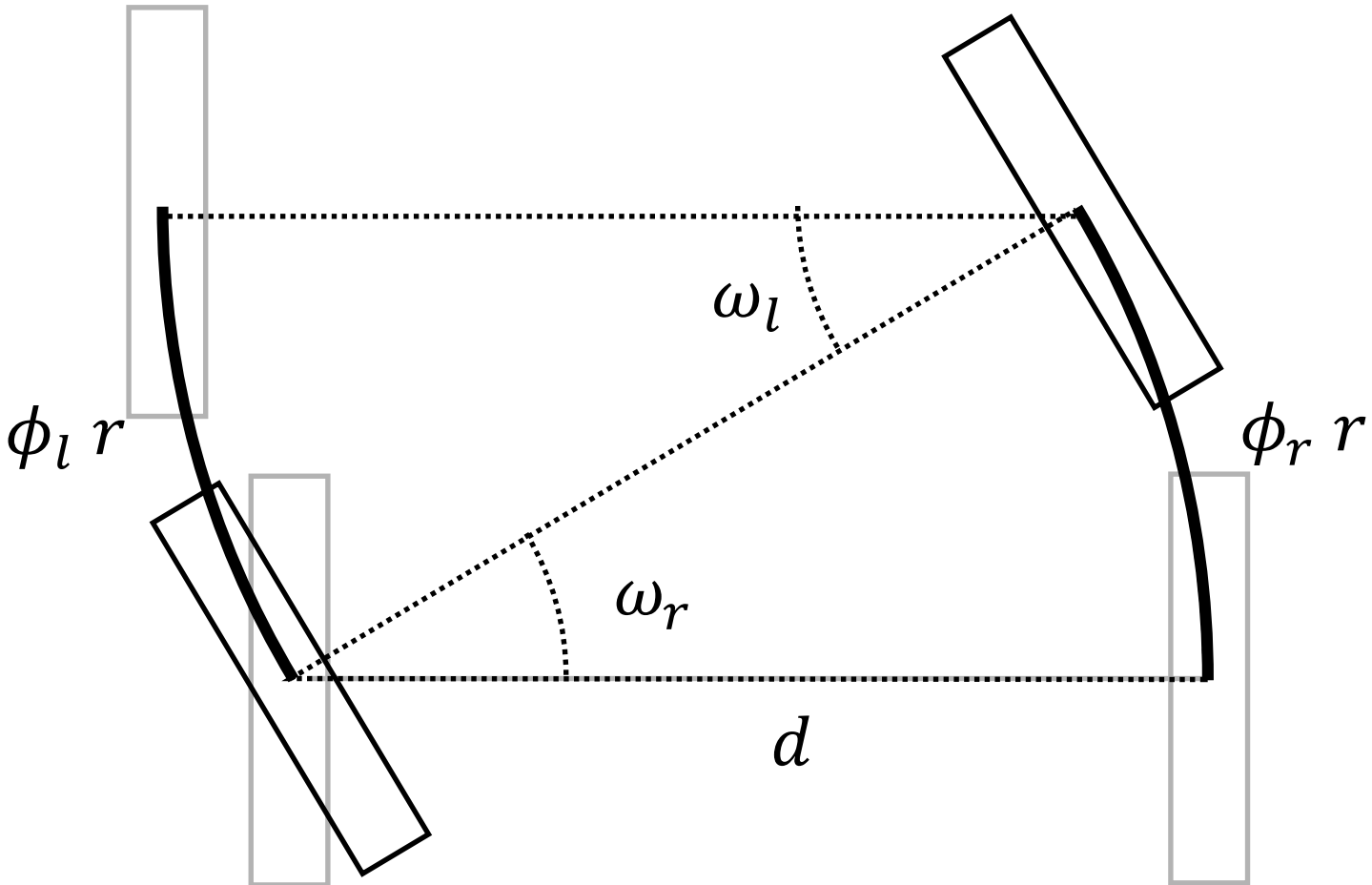


Distance traveled is angle of rotation times wheel radius:

$$x = r\phi$$

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

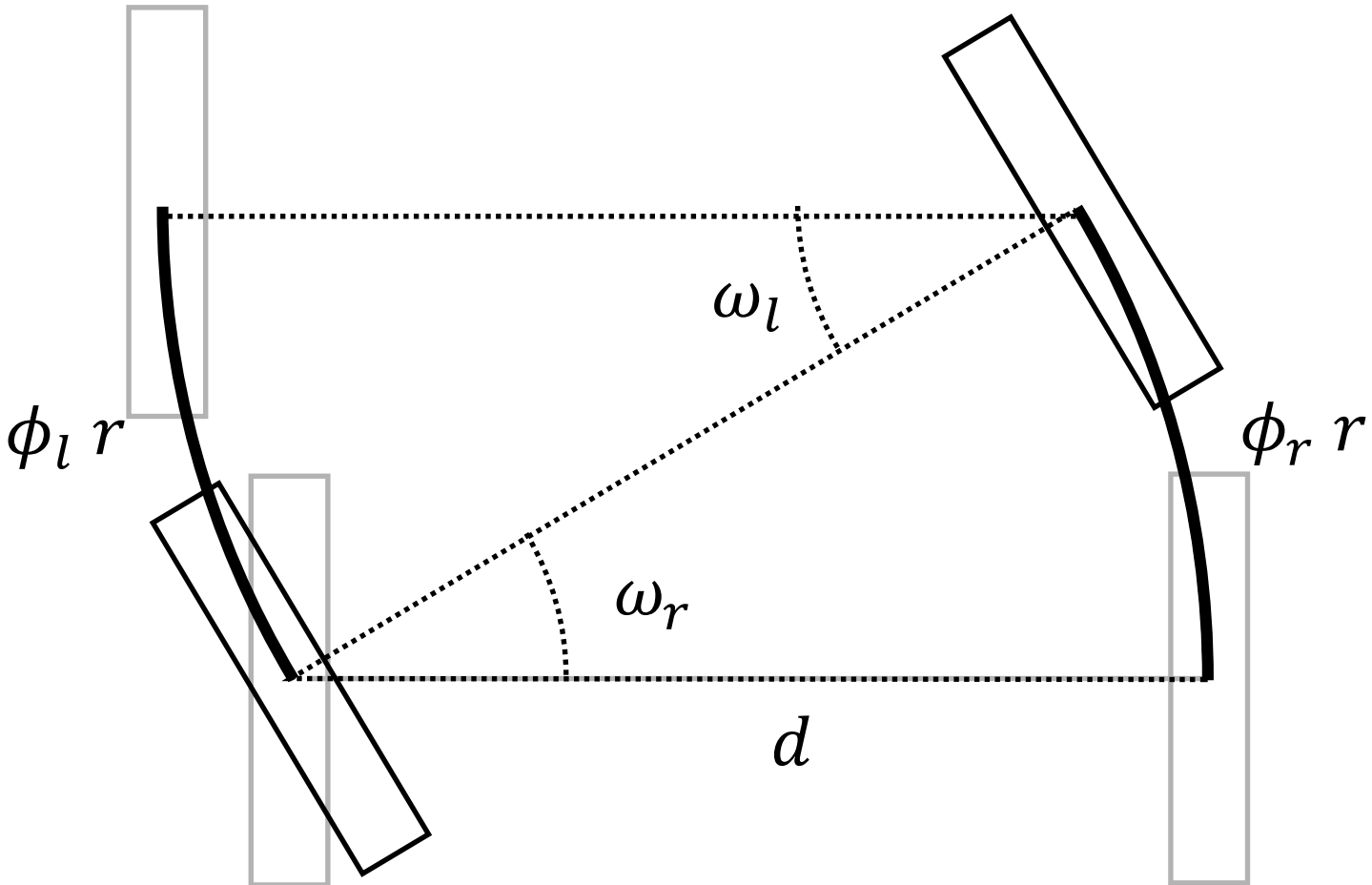
Recap: Wheel motions → Position Updates



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

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

Recap: Wheel motions → Position Updates

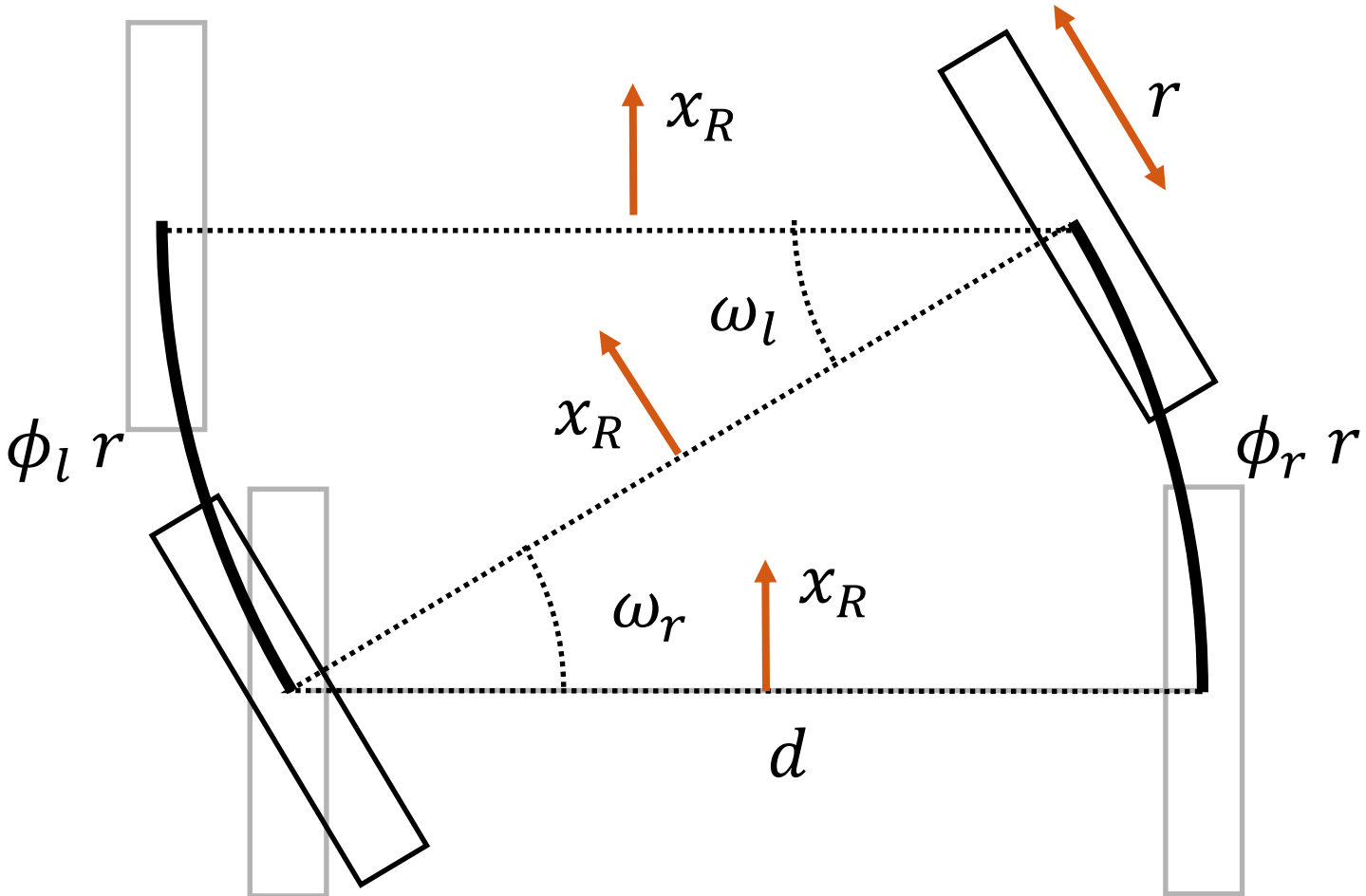


$$\dot{\omega}_r = \frac{\dot{\phi}_r r}{d}$$

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

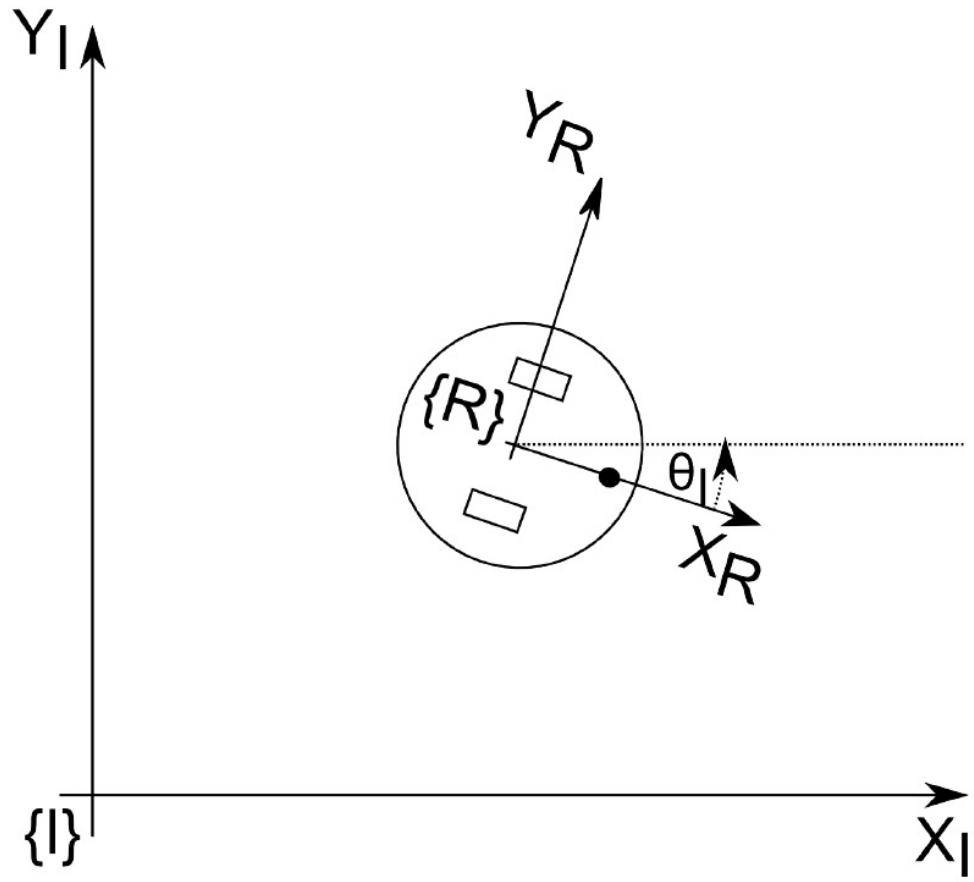
$$\dot{\theta} = \frac{\dot{\phi}_r r}{d} - \frac{\dot{\phi}_l r}{d}$$

Recap: Forward Kinematics of mobile robot



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

Recap: Rotating into inertial (”I”) frame



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

$$\dot{x_{I,y}} = -sin(\theta)\dot{y_R}$$

$$\dot{x_I} = cos(\theta)\dot{x_R} - sin(\theta)\dot{y_R}$$

$$\dot{y_I} = sin(\theta)\dot{x_R} + cos(\theta)\dot{y_R}$$

$$\dot{\theta_I} = \dot{\theta_R}$$

$$\begin{pmatrix} \dot{x_I} \\ \dot{y_I} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{r\dot{\phi_l}}{2} + \frac{r\dot{\phi_r}}{2} \\ 0 \\ \frac{\dot{\phi_r} r}{d} - \frac{\dot{\phi_l} r}{d} \end{pmatrix}$$

Recap: Speeds → How can we compute positions?

$$\begin{pmatrix} x_I(T) \\ y_I(T) \\ \theta(T) \end{pmatrix} = \int_0^T \begin{pmatrix} \dot{x_I}(t) \\ \dot{y_I}(t) \\ \dot{\theta}(t) \end{pmatrix} dt \approx \sum_{k=0}^{k=T} \begin{pmatrix} cos(\theta)\ (\ \frac{r\dot{\phi_l}}{2} + \frac{r\dot{\phi_r}}{2}\ ) \\ sin(\theta)\ (\ \frac{r\dot{\phi_l}}{2} + \frac{r\dot{\phi_r}}{2}\ ) \\ \frac{\dot{\phi_r} r}{d} - \frac{\dot{\phi_l} r}{d} \end{pmatrix} \Delta t$$

We do this in lab 2 – adding pose up during every `robot_step`

Homework: integrate wheel positions

- Which terms are dependent on time?
- Which terms are constant?

$$\dot{x_I} = cos(\theta)\left(\frac{r\dot{\phi_l}}{2} + \frac{r\dot{\phi_r}}{2}\right)$$

$$\dot{y_I} = sin(\theta)\left(\frac{r\dot{\phi_l}}{2} + \frac{r\dot{\phi_r}}{2}\right)$$

$$\dot{\theta} = \frac{\dot{\phi_r} r}{d} - \frac{\dot{\phi_l} r}{d}$$

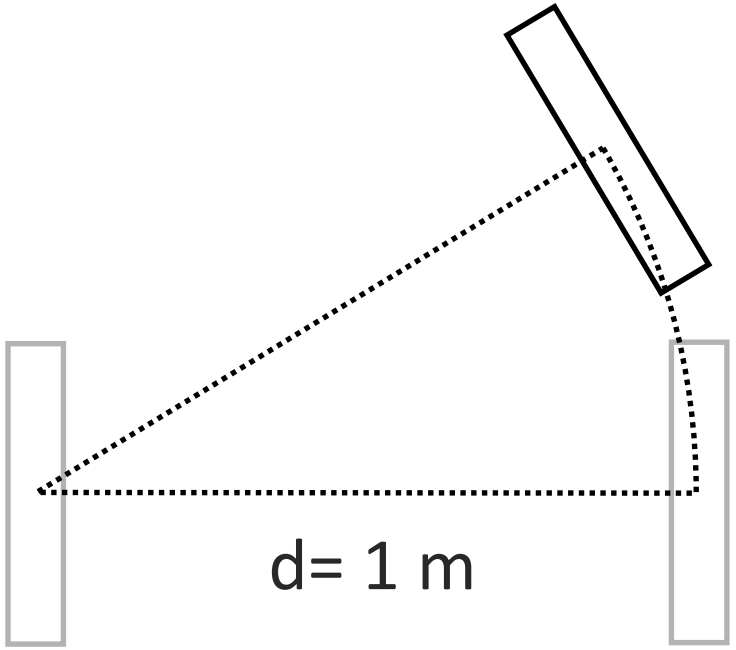
Start with the angle

$$\int \dot{\theta} dt = \int \frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d} dt = \left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) t$$

$$\int_0^t \dot{\theta} dt = \left|\left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) t\right|^t - \left|\left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) t\right|^0$$

$$\int_0^t \dot{\theta} dt = \ \theta(t) = \left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) t$$

Example: left turn with 30 degree rotation



$$r\dot{\varphi}_r = \frac{\pi}{6}\frac{m}{s}$$

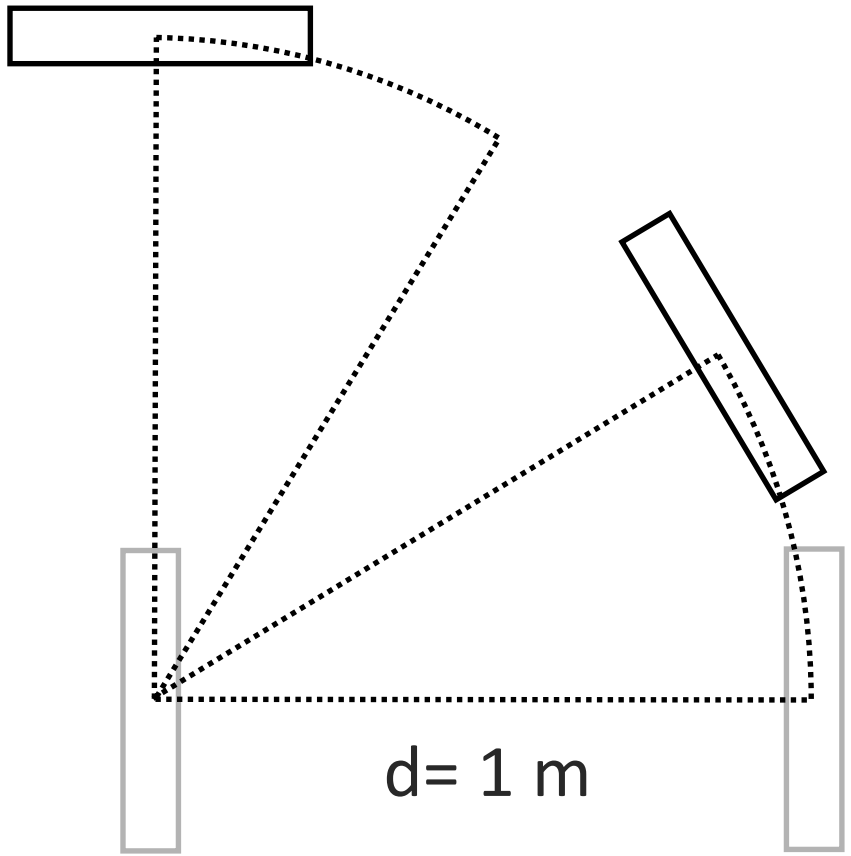
$$r\dot{\varphi}_l = 0\,\frac{m}{s}$$

$$t = 1s$$

$$\int_0^t \dot{\theta}\,dt = \left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right)t \quad (r\dot{\varphi}_l = 0)$$

$$\theta = \frac{\pi}{6}rad = 30\ \text{deg}$$

Example: left turn with 90 degree rotation



$$r\dot{\varphi}_r = \frac{\pi}{6}\frac{m}{s}$$

$$r\dot{\varphi}_l = 0\,\frac{m}{s}$$

$$t = 3s$$

$$\int_0^t \dot{\theta}\,dt = \left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right)t \quad (r\dot{\varphi}_l \to 0)$$

$$\theta = \frac{1}{2}\pi\ rad = 90\ \text{deg}$$

What about position?

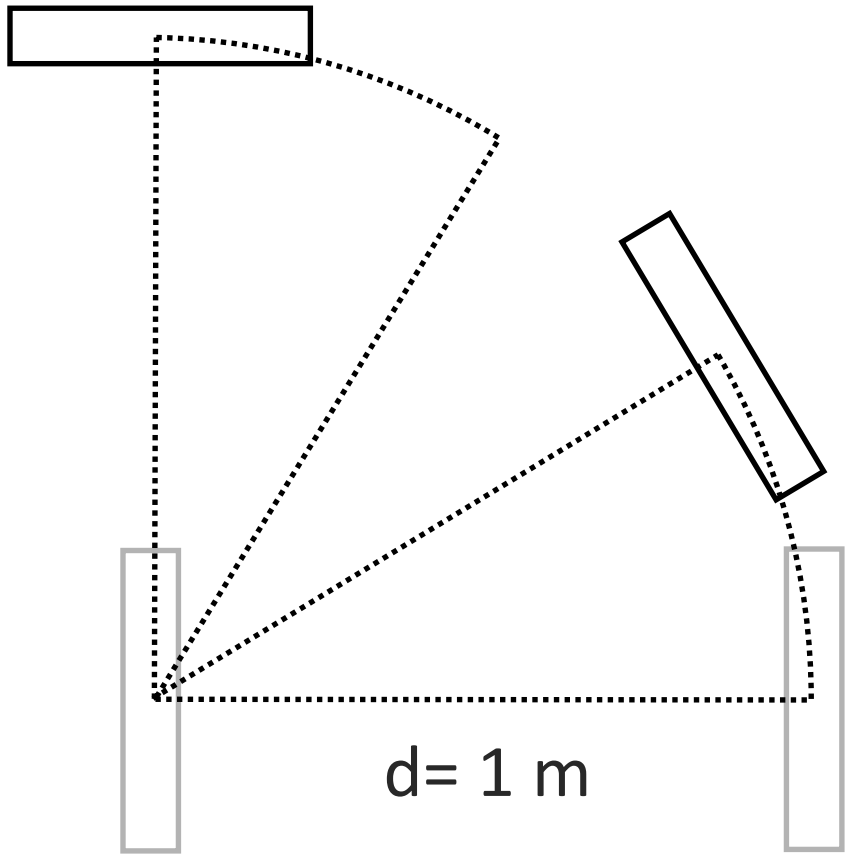
$$\int \dot{x}_I dt = \int \cos\theta(t) \left(\frac{r\dot{\varphi}_r}{2} + \frac{r\dot{\varphi}_l}{2}\right) dt = \int \cos u \left(\frac{r\dot{\varphi}_r}{2} + \frac{r\dot{\varphi}_l}{2}\right) dt$$

with:

$$u = \theta(t) = \left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) t \qquad \frac{du}{dt} = \left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) \rightarrow dt = \frac{d}{r\dot{\varphi}_r - r\dot{\varphi}_l} du$$

$$\int \dot{x}_I dt = \int \cos u \left(\frac{r\dot{\varphi}_r}{2} + \frac{r\dot{\varphi}_l}{2}\right) \frac{d}{r\dot{\varphi}_r - r\dot{\varphi}_l} du = sin\left(\left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right) t\right) \frac{(r\dot{\varphi}_r + r\dot{\varphi}_l)d}{2(r\dot{\varphi}_r - r\dot{\varphi}_l)}$$

Example: left turn with 90 degree rotation



$$r\dot{\varphi}_r = \frac{\pi}{6}\frac{m}{s}$$

$$r\dot{\varphi}_l = 0\,\frac{m}{s}$$

$$t = 3s$$

$$\sin\left(\left(\frac{r\dot{\varphi}_r}{d} - \frac{r\dot{\varphi}_l}{d}\right)t\right)\frac{(r\dot{\varphi}_r + r\dot{\varphi}_l)d}{2(r\dot{\varphi}_r - r\dot{\varphi}_l)}$$

$$\sin\left(\left(\frac{r\dot{\varphi}_r}{d}\right)t\right)\frac{d}{2}$$

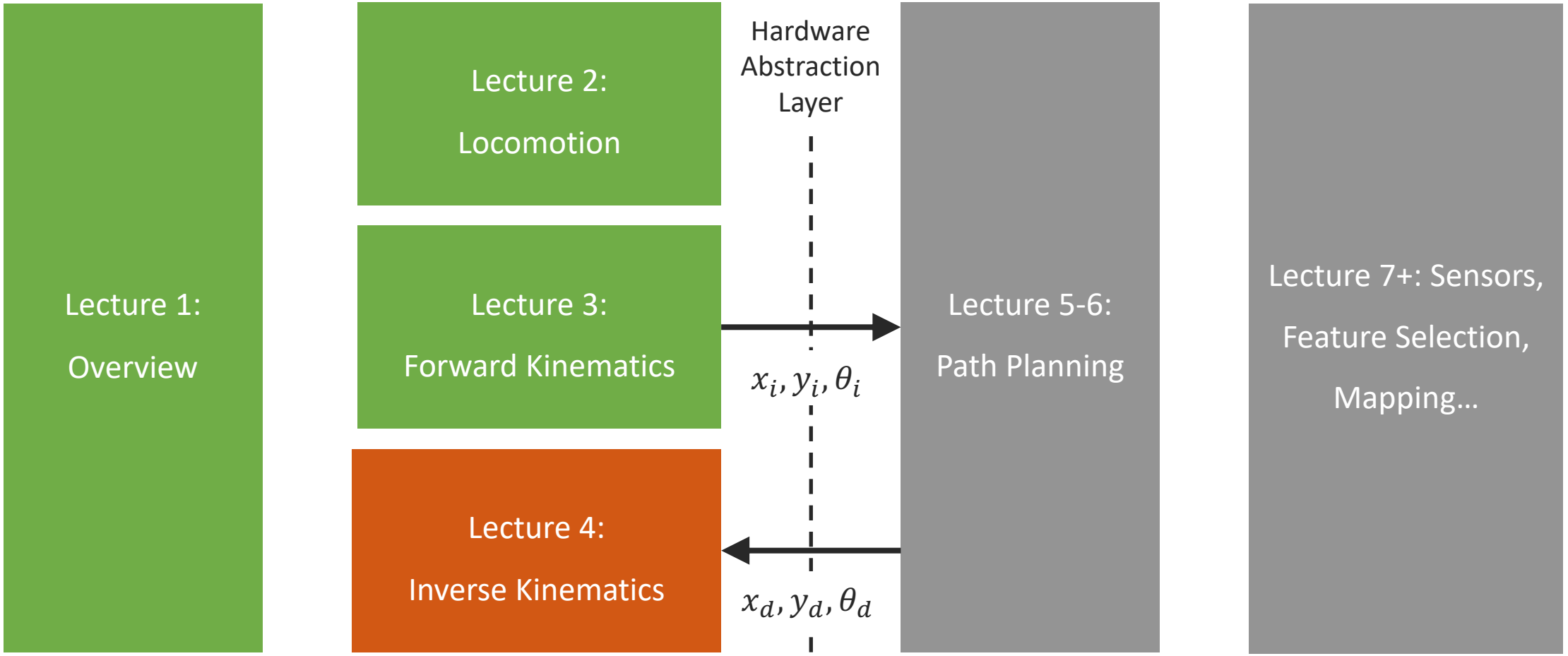
$$x_I = \frac{1}{2}m$$

Chapter 3

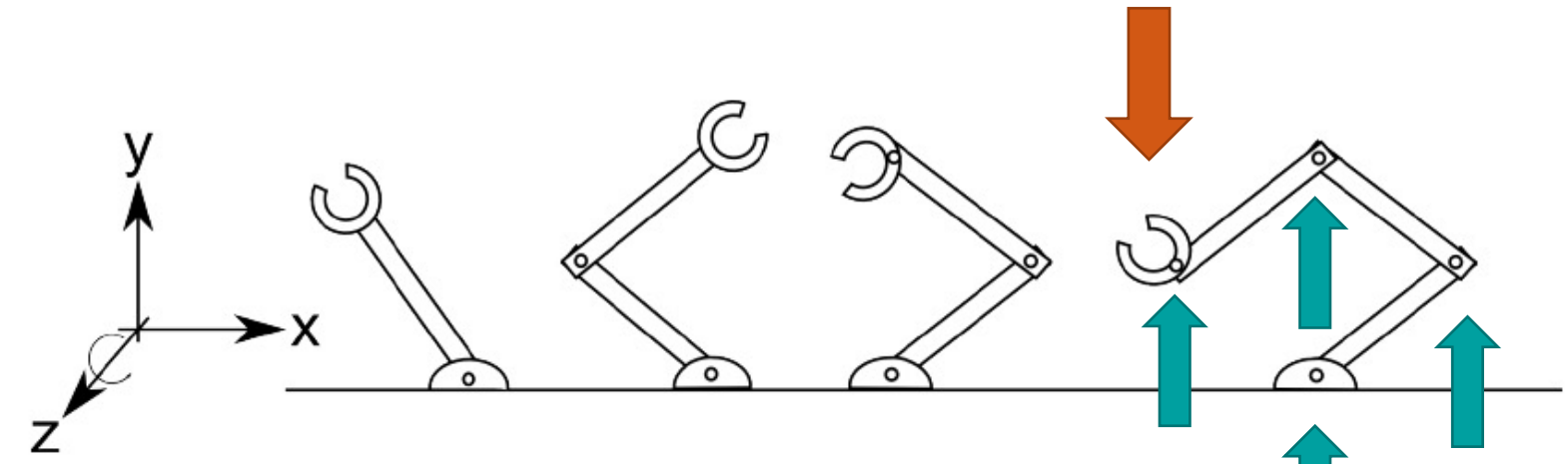
Module 1 – MECHANISMS

Part III – Inverse Kinematics

Roadmap


Lecture 1:
Overview
Lecture 2:
Locomotion
Lecture 3:
Forward Kinematics
Lecture 4:
Inverse Kinematics
Hardware
Abstraction
Layer
x_i, y_i, θ_i
x_d, y_d, θ_d
Lecture 5-6:
Path Planning
Lecture 7+: Sensors,
Feature Selection,
Mapping…

Inverse Kinematics



JOINT space

$q = (q_1, \ldots, q_n)$

$r = f(q)$

FORWARD

INVERSE

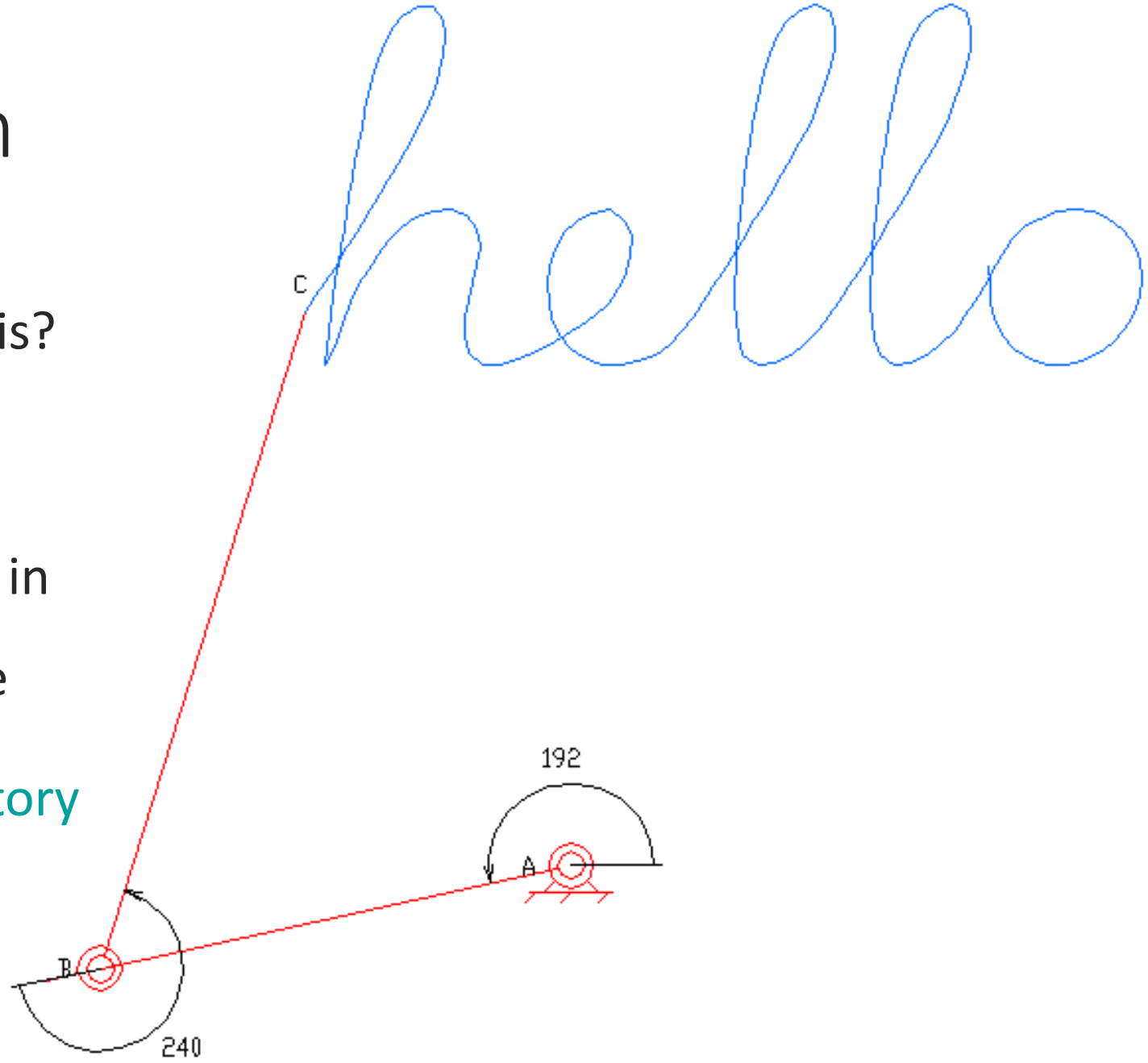
$q = f^{-1}(r)$

TASK space
(cartesian)

$r = (r_1, \ldots, rm)$

Motivating Problem

How can we get a robot to do this?

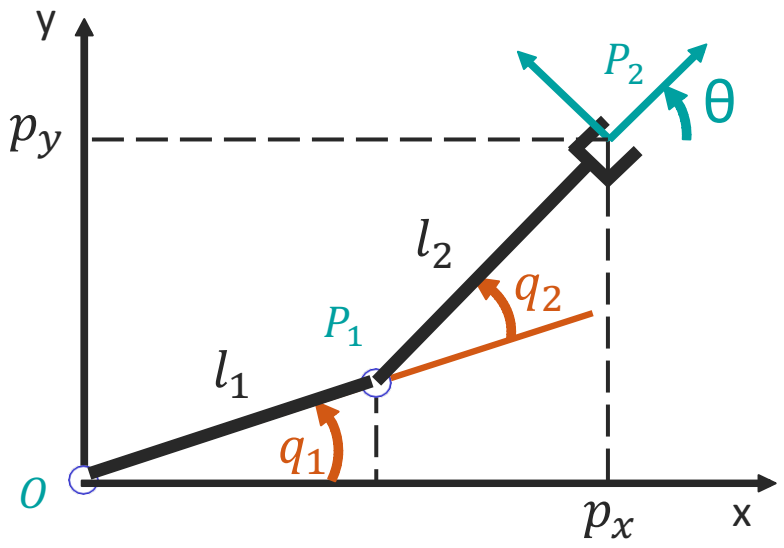


- We need to define a trajectory in end-effector/operational space
- We need to convert this trajectory to joint/configuration space

Section 3.3.2 in the book

Inverse Kinematics

IK for manipulator

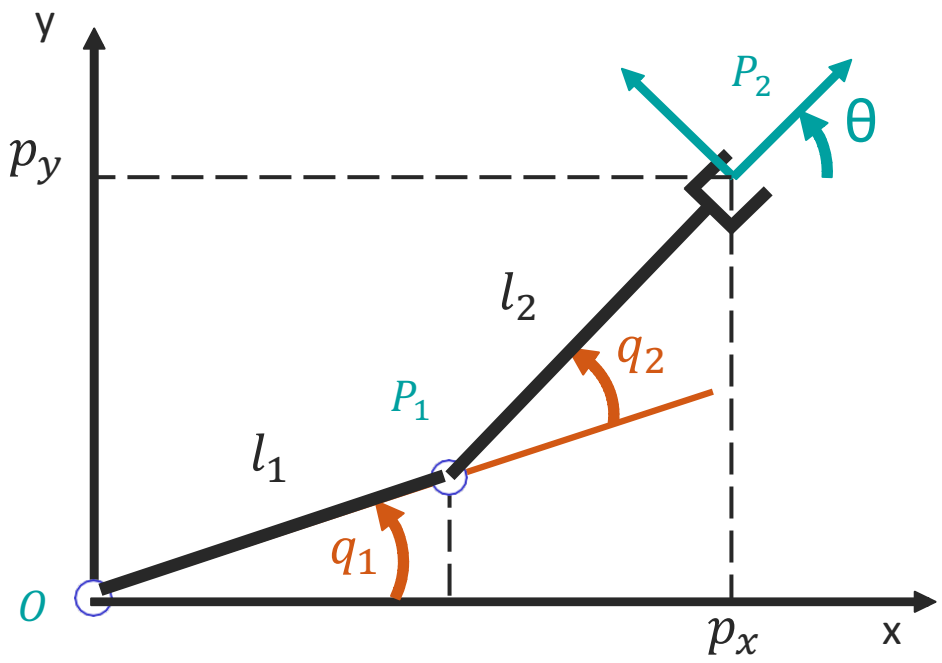


Given point (x, y, θ) Find angles (q_1, q_2)

IK for e-puck

Given: (x, y, θ) Find: $(\phi_l(t), \phi_r(t))$

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

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

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

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

⇓

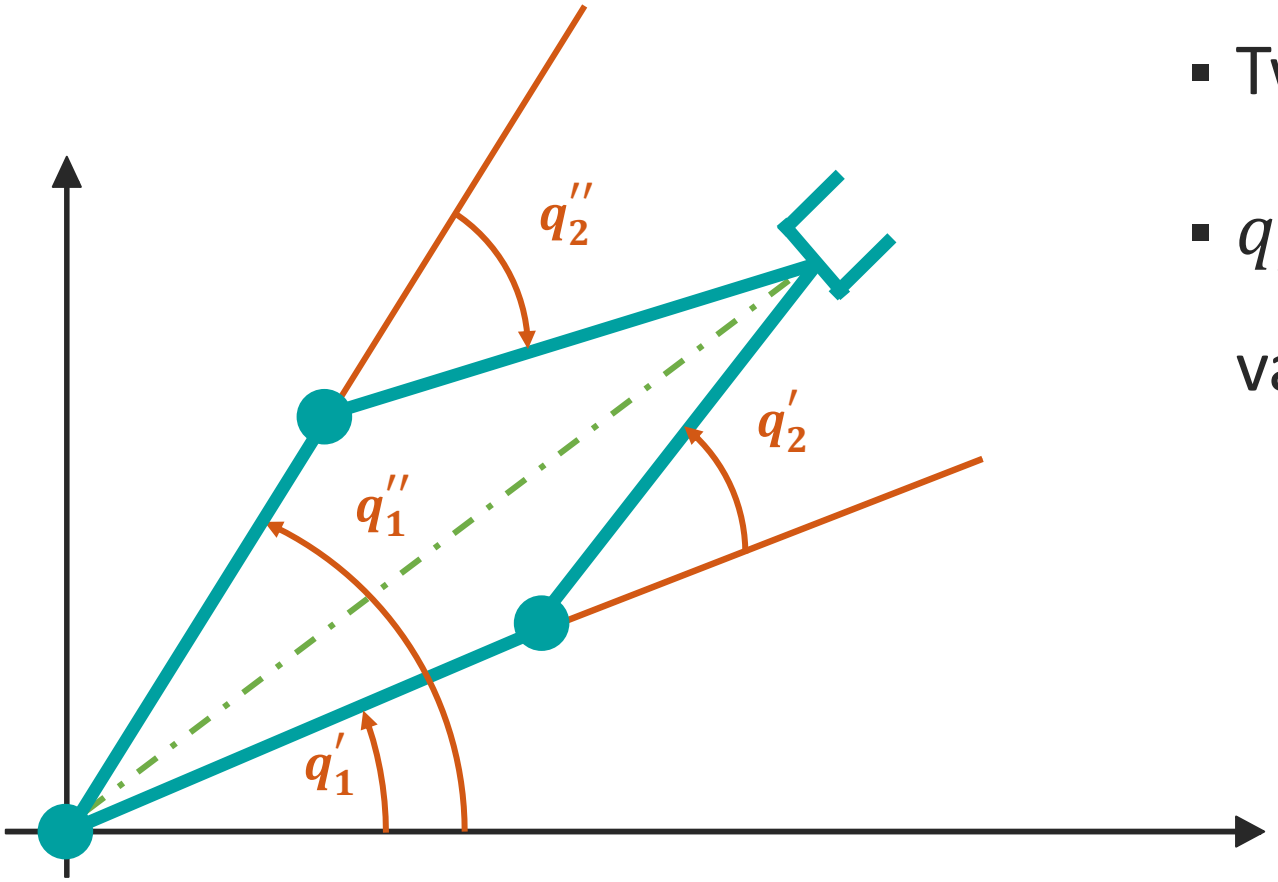
$$q_1 = \cos^{-1}\frac{x^2y + y^3 - \sqrt{4x^4 - x^6 + 4x^2y^2 - 2x^4y^2 - x^2y^4}}{2(x^2 + y^2)}$$

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

r=f(q)

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

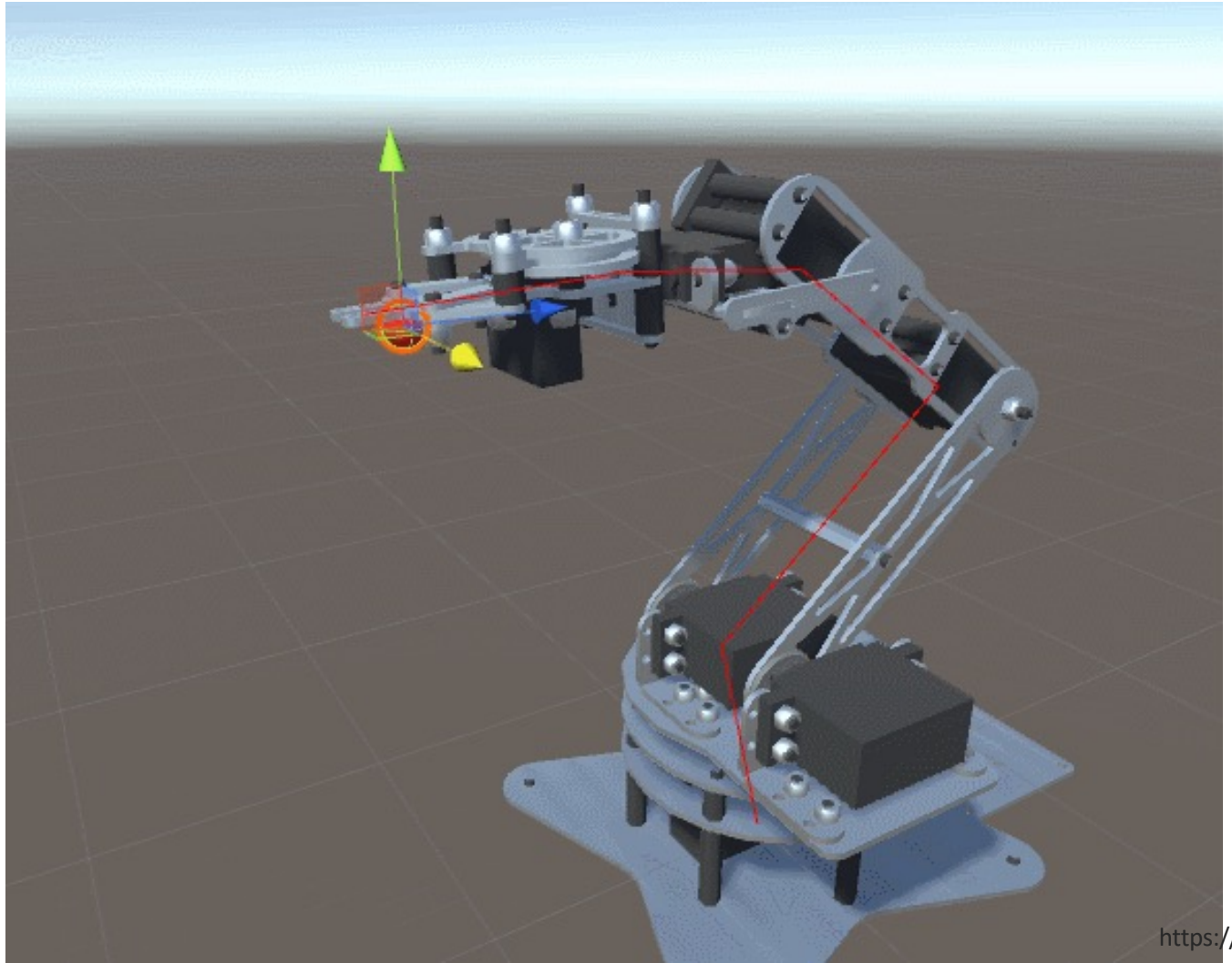
Example: Inverse Kinematics of a 2R arm



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

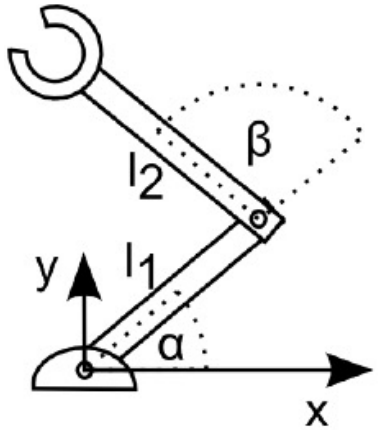
Great video about IK: https://www.youtube.com/watch?v=IKOGwoJ2HLk

End-effector Position Control

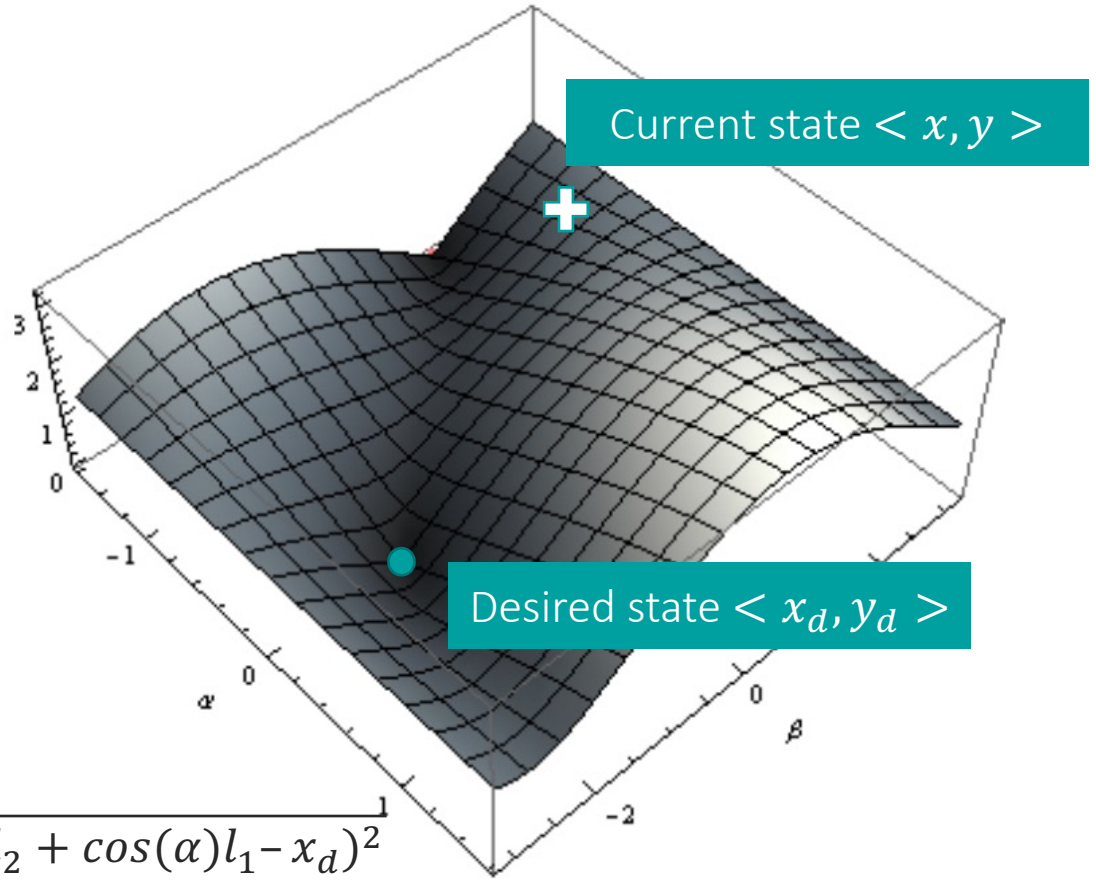


https://www.alanzucconi.com/2017/04/10/robotic-arms/

Easier ways to solve the IK problem



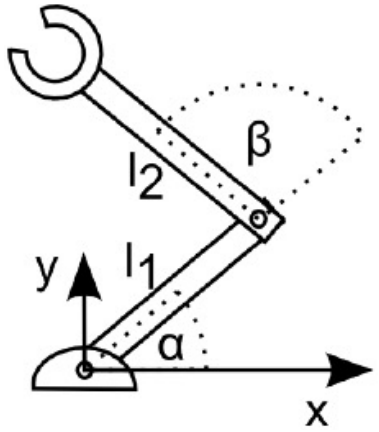
$$x = l_1 \cos(\alpha) + l_2 \cos(\alpha + \beta)$$
$$y = l_1 \sin(\alpha) + l_2 \sin(\alpha + \beta)$$



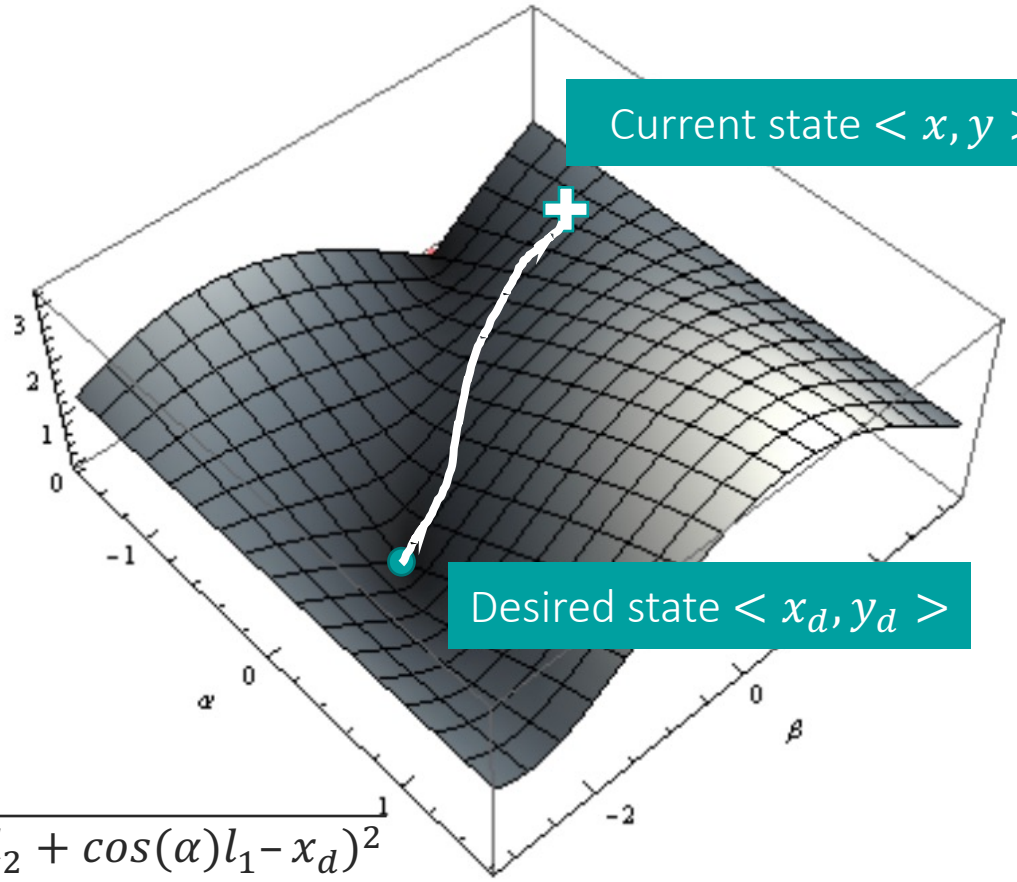
Just the Euclidean distance between two vectors!

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

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

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

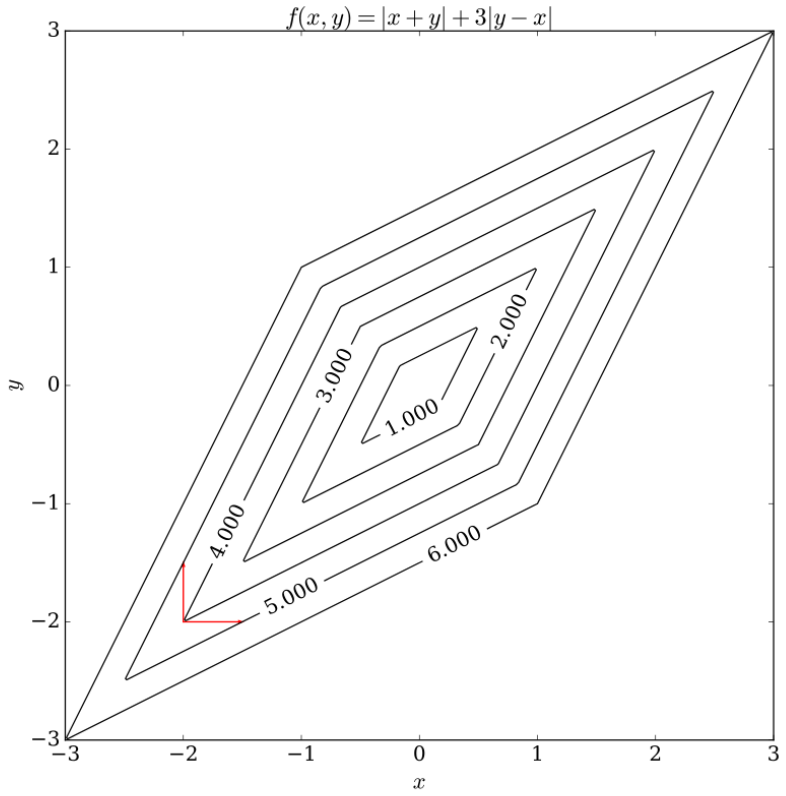
At iteration $k+1$, $n-1$ variables are set to their previous values, and only one (position i) is updated to a value minimizing f

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

Each iteration reduces our error or remains stationary

Coordinate Descent

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

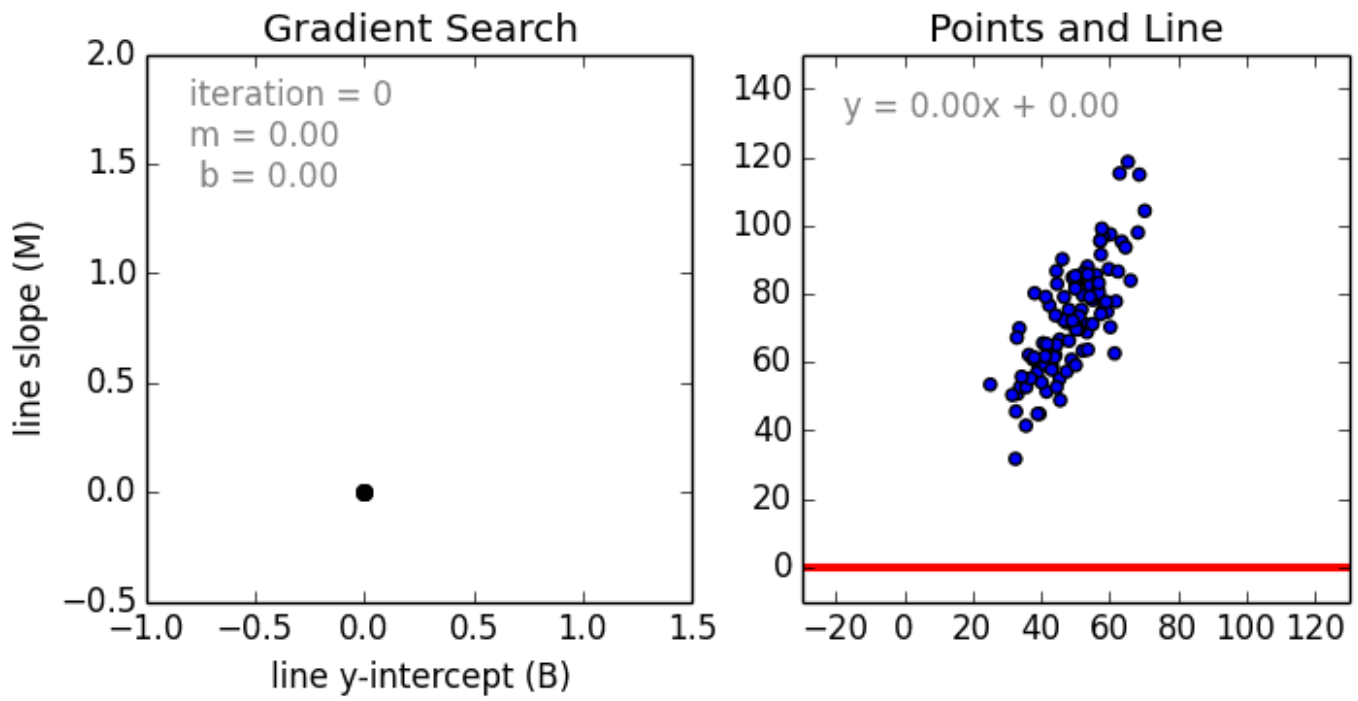


- Choose an initial parameter vector $\boldsymbol{x}$
- Until convergence is reached (or for some fixed number of iterations):
  - Choose an index i from 1 to n
  - Choose a step size α
  - Update x_i to $x_i - \alpha \frac{\partial F}{\partial x_i}(\boldsymbol{x})$

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

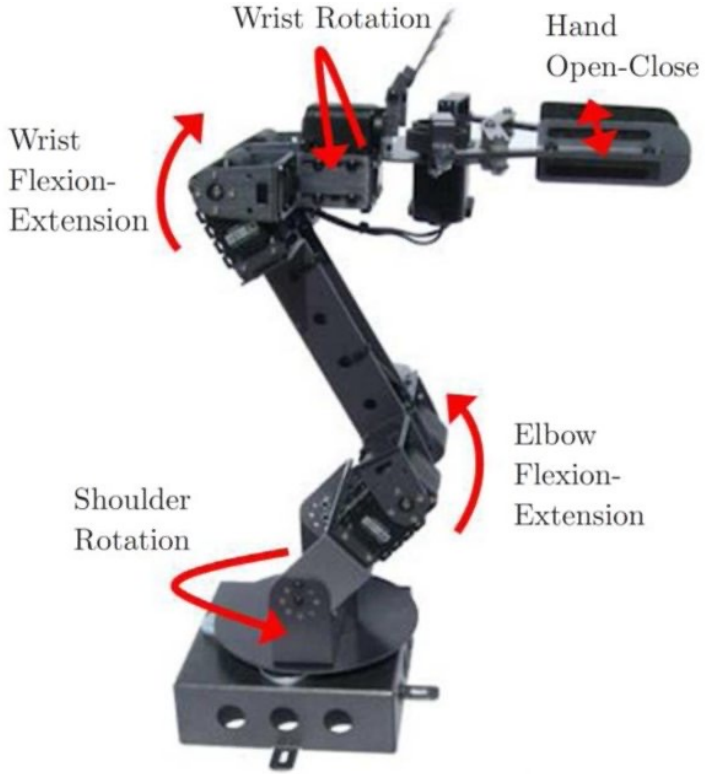


Gradient Descent for Solving IK

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

Joint angles
α_i

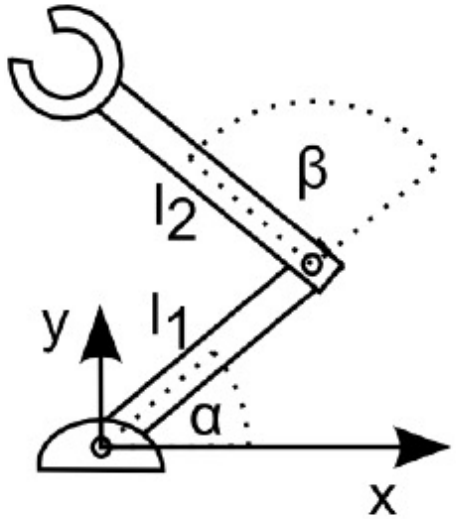
Learning rate
L

Analytically computing the gradient

- 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:

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

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

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

∂ : partial derivative

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 Configuration space

K: gain

$v_{e,d}$: Desired velocity

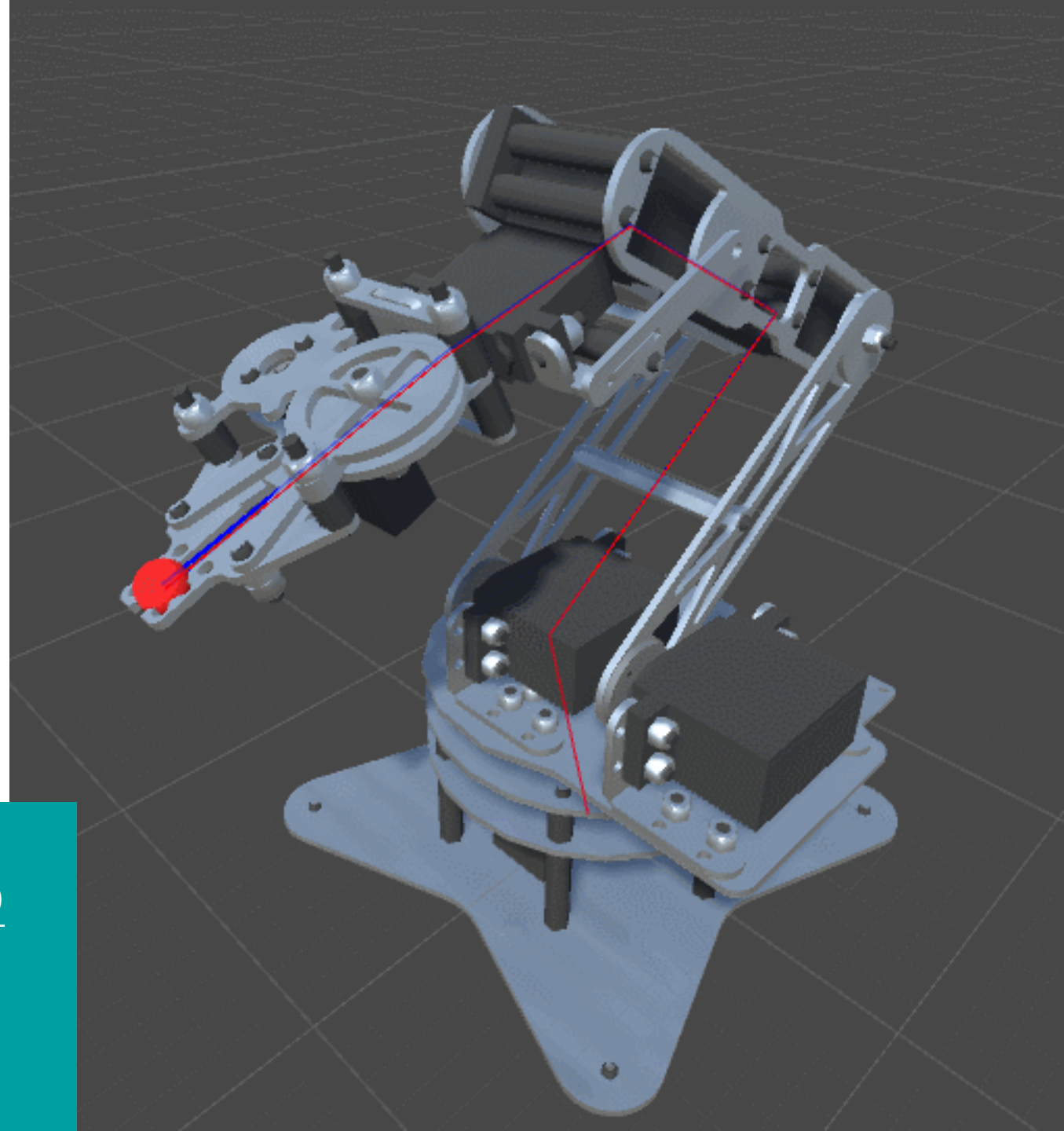
$p_{e,d}$: Desired position

Convergence

Problems!

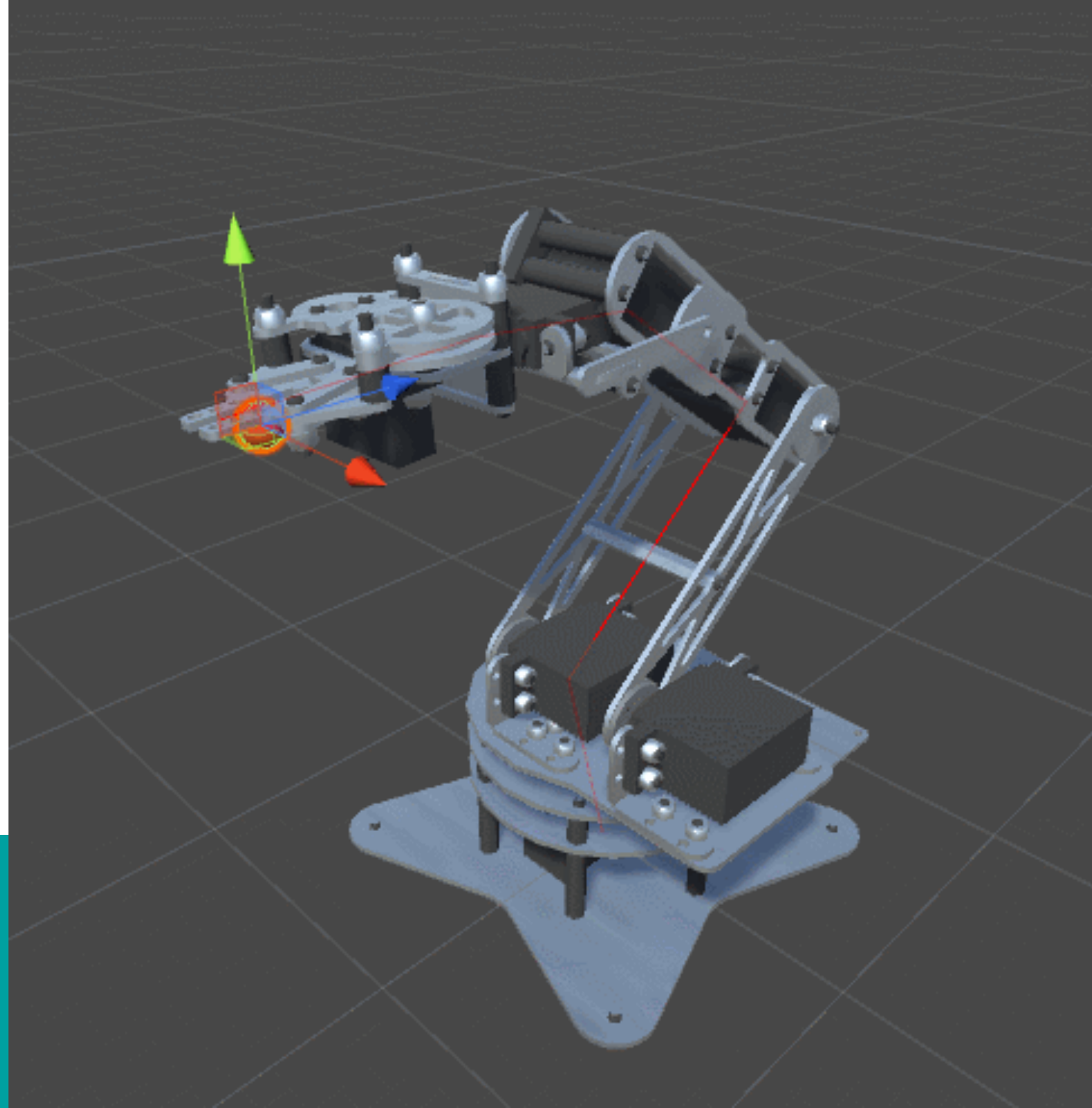
$$\nabla f_{\alpha_0} = (\alpha_0, \alpha_1, \alpha_2) = \frac{f(\alpha_0 + \Delta_x, \alpha_1, \alpha_2) - f(\alpha_0, \alpha_1, \alpha_2)}{\Delta x}$$

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



Joint Angle Limitation Problems!

How do we fix this?



Fast IK Planning