

Robotics - 34753

# Introduction to Robotics

**Matteo Fumagalli**

Associate Professor

Automation and Control Group

Department of Electrical Engineering

DTU Lyngby, Building 326

Contributors:

Konstantinos Poulios

Nils Axel Andersen

Ole Ravn

# Outline

- Introduction to the course
- Introduction to robotics
- Introduction to group assignment

# Information Sheet

**Matteo Fumagalli**

Associate Professor

Department of Electrical and Photonics Engineering

Email: [mafum@dtu.dk](mailto:mafum@dtu.dk)

Office: Building 326, room 120

**Nils Axel Andersen**

Associate Professor

Department of Electrical and Photonics Engineering

Email: [nian@dtu.dk](mailto:nian@dtu.dk)

Office: Building 326, room 016

**Konstantinos Poulios**

Associate Professor

Department of Civil and Mechanical Engineering

Email: [kopo@dtu.dk](mailto:kopo@dtu.dk)

Office: Building 404, room 124

**Ole Ravn**

Professor

Department of Electrical and Photonics Engineering

Email: [oravn@dtu.dk](mailto:oravn@dtu.dk)

Office: Building 326, room 012

# Information Sheet

## Teaching Assistants



Louise Mattelaer

s232831@student.dtu.dk



Amin Mohseni

s222378@student.dtu.dk

# Syllabus

## Prerequisites

- 31300/ 31301/ 41671/ 41672/ 42672, knowledge of control theory corresponding to an introductory control course

## Grading

- Written report (group) + Written 2-hours exam (individual MCQ)

## Team work

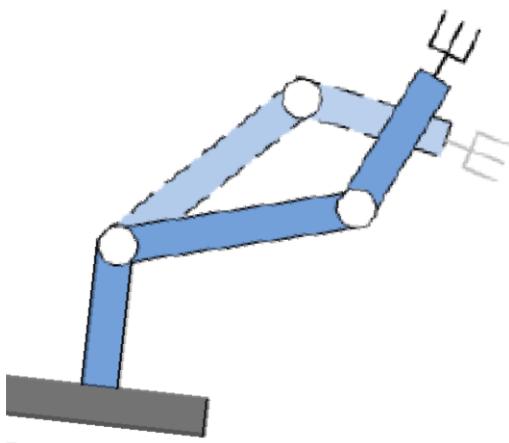
- 5 students form a group to work for the project assignment and exercise on real robot-arm

## Textbook

- “ROBOT MODELING AND CONTROL”, Mark W. Spong, Seth Hutchinson, M. Vidyasagar (first and second edition have the same relevant contents)

# Topics

## Introduction to robotics

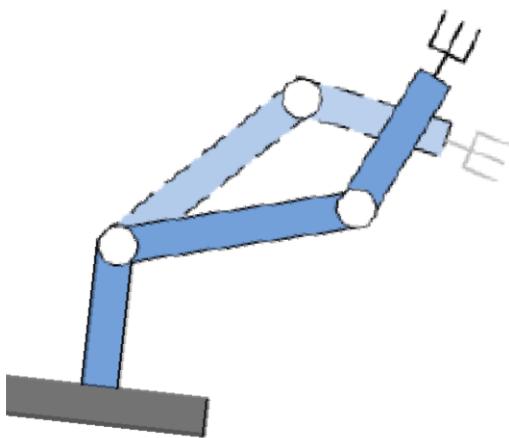


Joint  $\leftrightarrow$  End-effector  
End-effector  $\leftrightarrow$  Joint

# Topics

Introduction to robotics

Direct kinematics



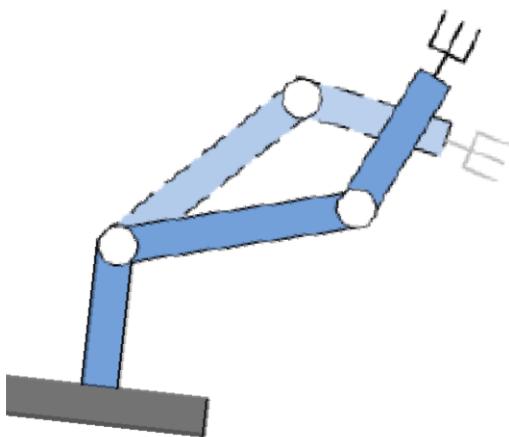
Joint  $\leftrightarrow$  End-effector  
End-effector  $\leftrightarrow$  Joint

# Topics

Introduction to robotics

Direct kinematics

Inverse Kinematics



Joint  $\leftrightarrow$  End-effector  
End-effector  $\leftrightarrow$  Joint

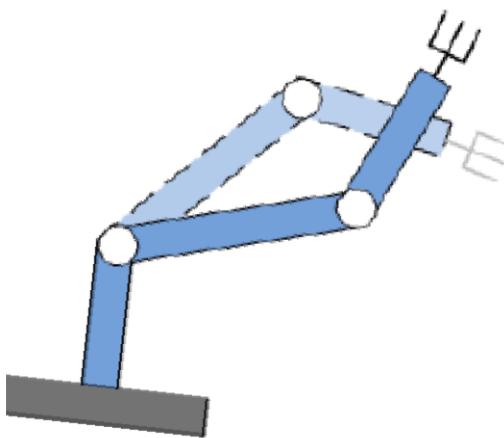
# Topics

Introduction to robotics

Direct kinematics

Inverse Kinematics

Path planning and trajectory generation



Joint  $\leftrightarrow$  End-effector  
End-effector  $\leftrightarrow$  Joint

# Topics

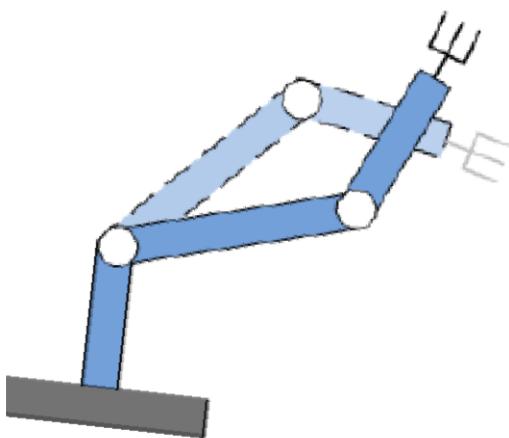
Introduction to robotics

Direct kinematics

Inverse Kinematics

Path planning and trajectory generation

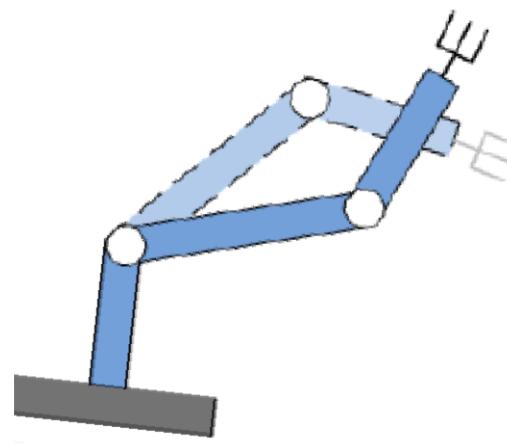
Robot dynamics



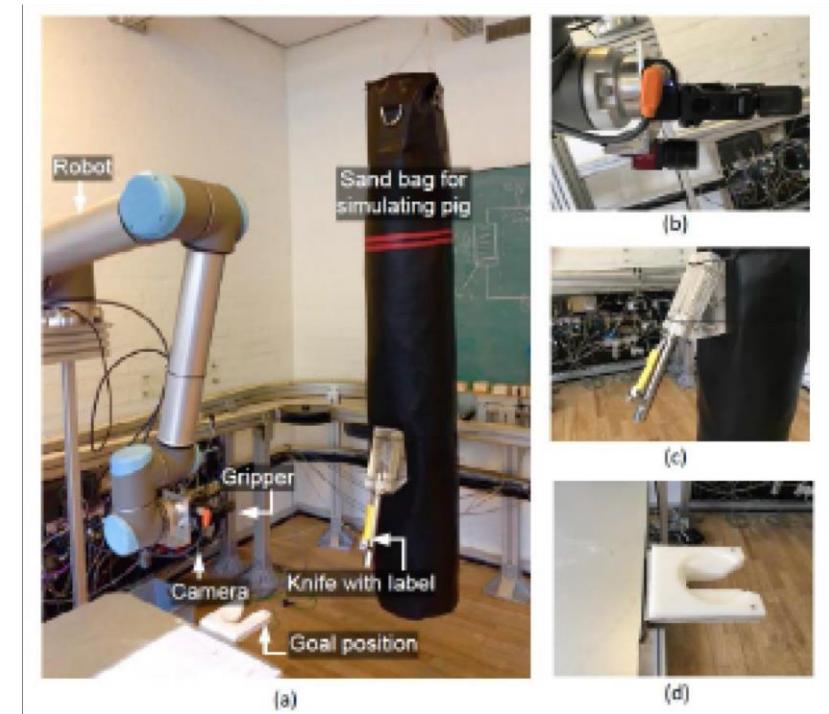
Joint  $\leftrightarrow$  End-effector  
End-effector  $\leftrightarrow$  Joint

# Topics

Introduction to robotics  
Direct kinematics  
Inverse Kinematics  
Path planning and trajectory generation  
Robot dynamics  
Robot control



Joint  $\leftrightarrow$  End-effector  
End-effector  $\leftrightarrow$  Joint



# Topics

Introduction to robotics (Matteo)

Direct kinematics (Konstantinos)

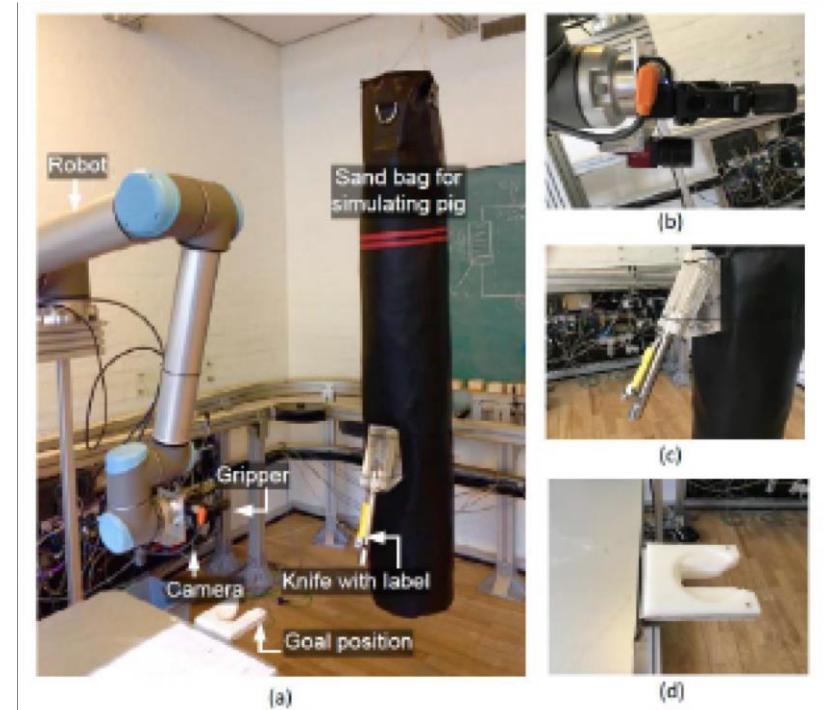
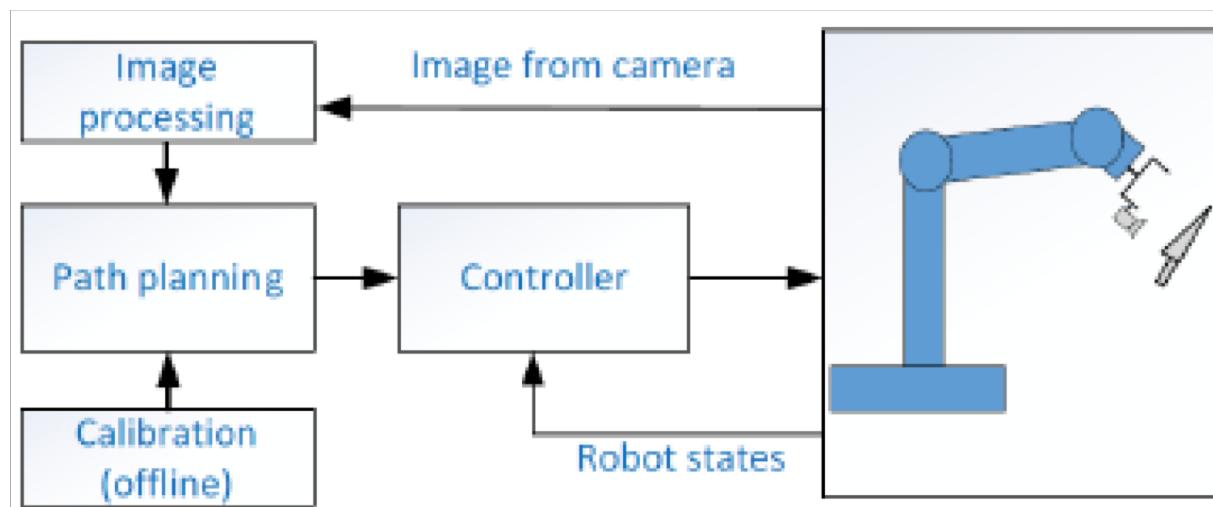
Inverse Kinematics (Konstantinos)

Path planning and trajectory generation (Konstantinos)

Robot dynamics (Konstantinos)

Robot control (Matteo)

Sensor systems in robotics (Matteo)



# Practical Assignment

- **Purpose**

- Practice the topics learned in the Robotics course 34753, on a realistic robotic application (on a real robotic system)

- **DTU CourseBot**

- 4 DoF robot

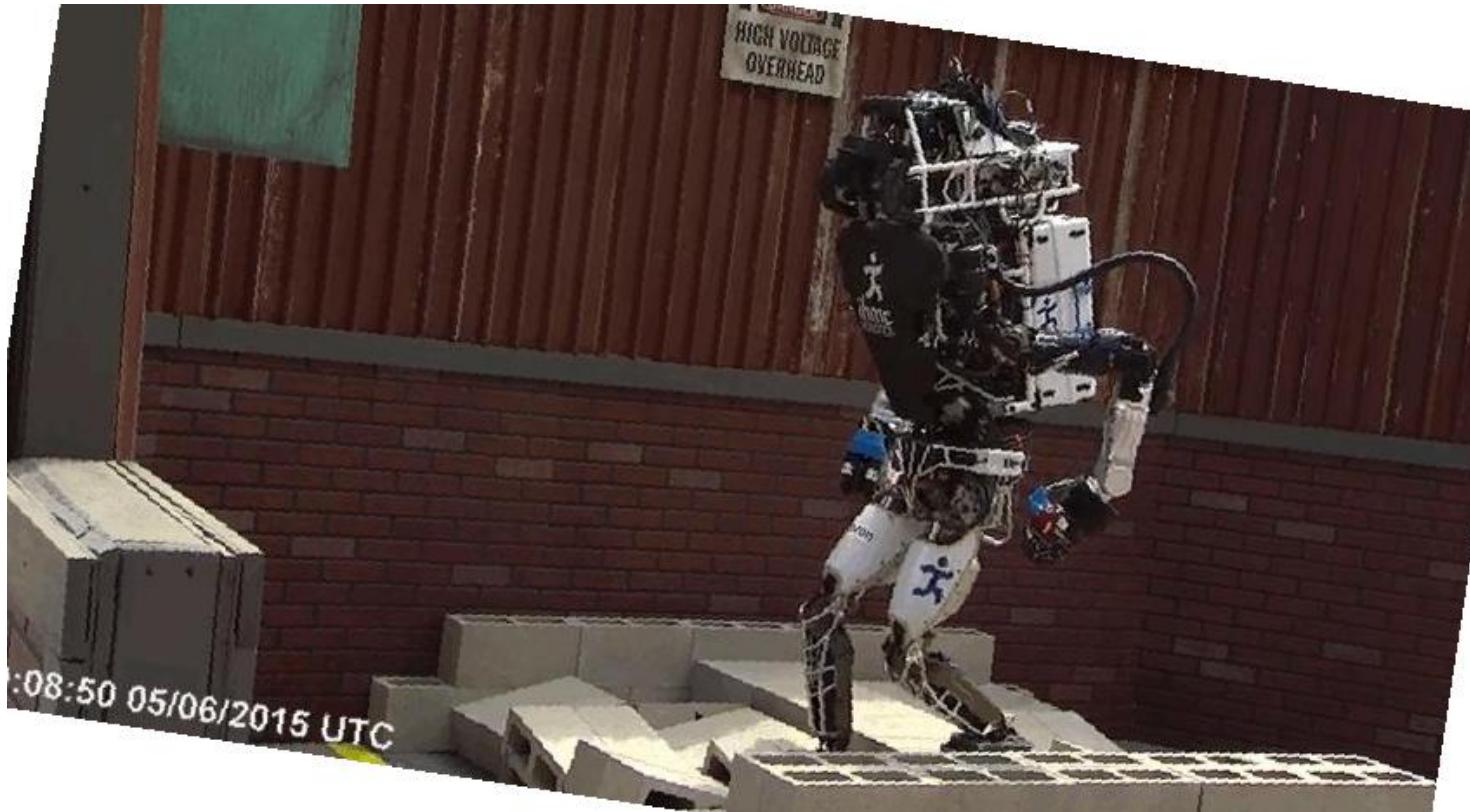
- **Solve -problems**

- Simulation, dynamics, kinematics, trajectory planning, control, real-life implementation



# Schedule

- 03/09 Introduction to robotics
- 10/09 Robot kinematics I
- 17/09 Robot kinematics II & inverse kinematics
- 24/09 Velocity Kinematics & The Jacobian Matrix
- 01/10 Side track assignments opportunities
- 08/10 Trajectory Planning & Manipulator Dynamics
- 15/10 Robot control
- 12/10 **(no lecture)**
- 29/10 Sensor systems in robotics
- 05/11 Project assignment
- 12/11 Project assignment
- 19/11 Project assignment
- 26/11 Advanced topics
- 03/12 Project assignment
- 05/12 **Deadline for handing in the compulsory project assignment**
- 16/12 Written exam**



# Introduction to Robotics

# What is a Robot?

- Origin of the word robot
  - Czech word “roboťa” – labor
  - 1920 play by Karel Čapek – Rossum’s Universal Robots
- Definition (no precise definition yet)
  - Robotics institute of American
    - “A robot (industrial robot) is a **reprogrammable, multifunctional manipulator** designed to move materials, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks.”
  - European common market
    - “an **independent acting** and **self controlling machine**, equipped with specific tools to handle or machine and whose movements are **programmable** ....”



Karel Čapek

# What is a Robot?

Hollywood's imagination



C3PO & R2-D2



Wall-E



Chappie



Terminator

# What is a Robot?

- By general agreement, a robot is:

A programmable machine that imitates the actions or appearance of an intelligent creature – usually a human.



"Real steel", 2011

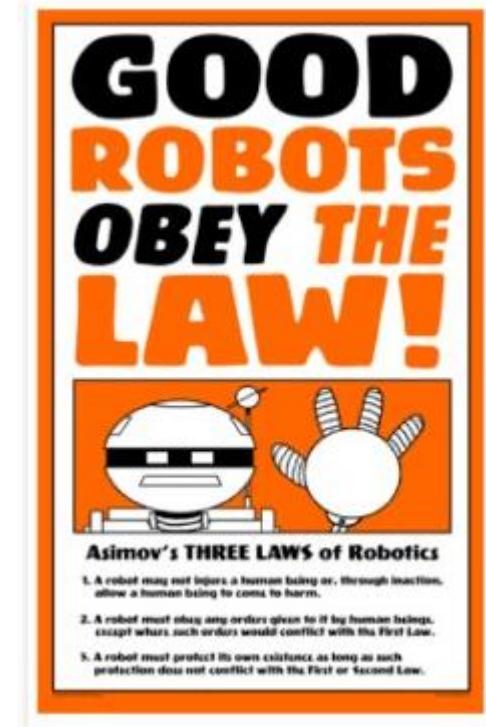
Sensing and perception

Carry out different tasks

Interact with human beings

# The Three Laws of Robotics

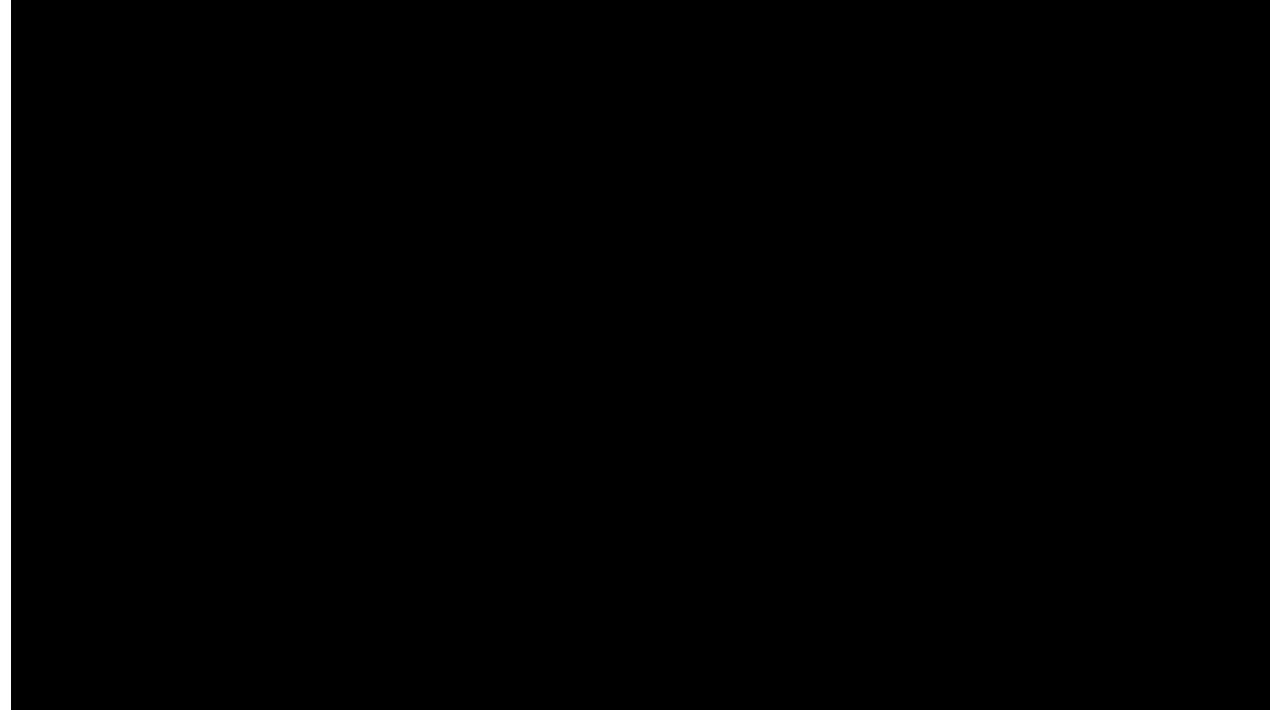
- Asimov proposed three “Laws of Robotics”
- **Law 1:** A robot may not injure a human being or through inaction, allow a human being to come to harm
- **Law 2:** A robot must obey orders given to it by human beings, except where such orders would conflict with a higher order law
- **Law 3:** A robot must protect its own existence as long as such protection does not conflict with a higher order law



# Types of Robots



Manipulator

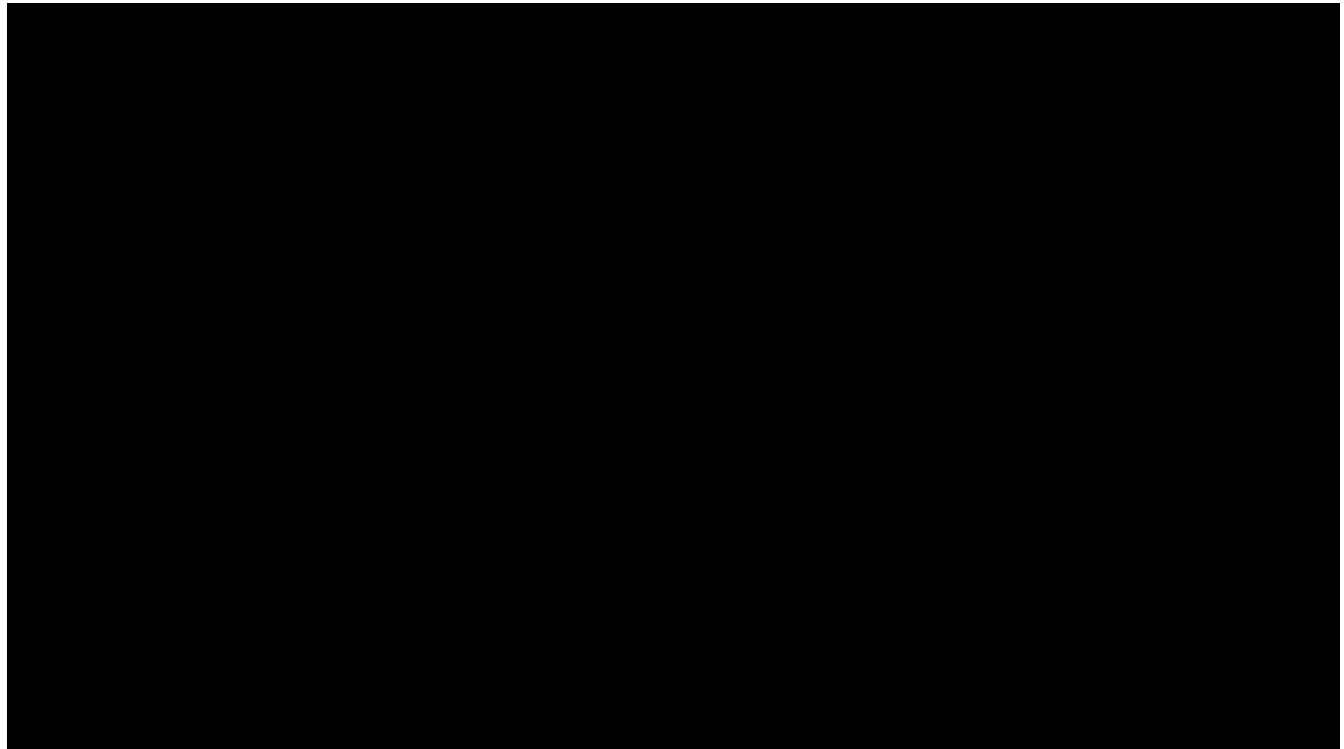


<https://www.youtube.com/watch?v=neWc5I9IdQ4>

# Types of Robots



Wheeled mobile robots



<https://www.youtube.com/watch?v=BwG5yoTbX6c>

# Types of Robots



Legged robots



<https://www.youtube.com/watch?v=bmNaLtC6vkU>

*'BigDog runs at 4 mph (1.79 m/s), climbs slopes up to 35 degrees, walks across rubble, climbs muddy hiking trails, walks in snow and water, and carries 340 lb (154kg) load.'*

# Types of Robots



Humanoid

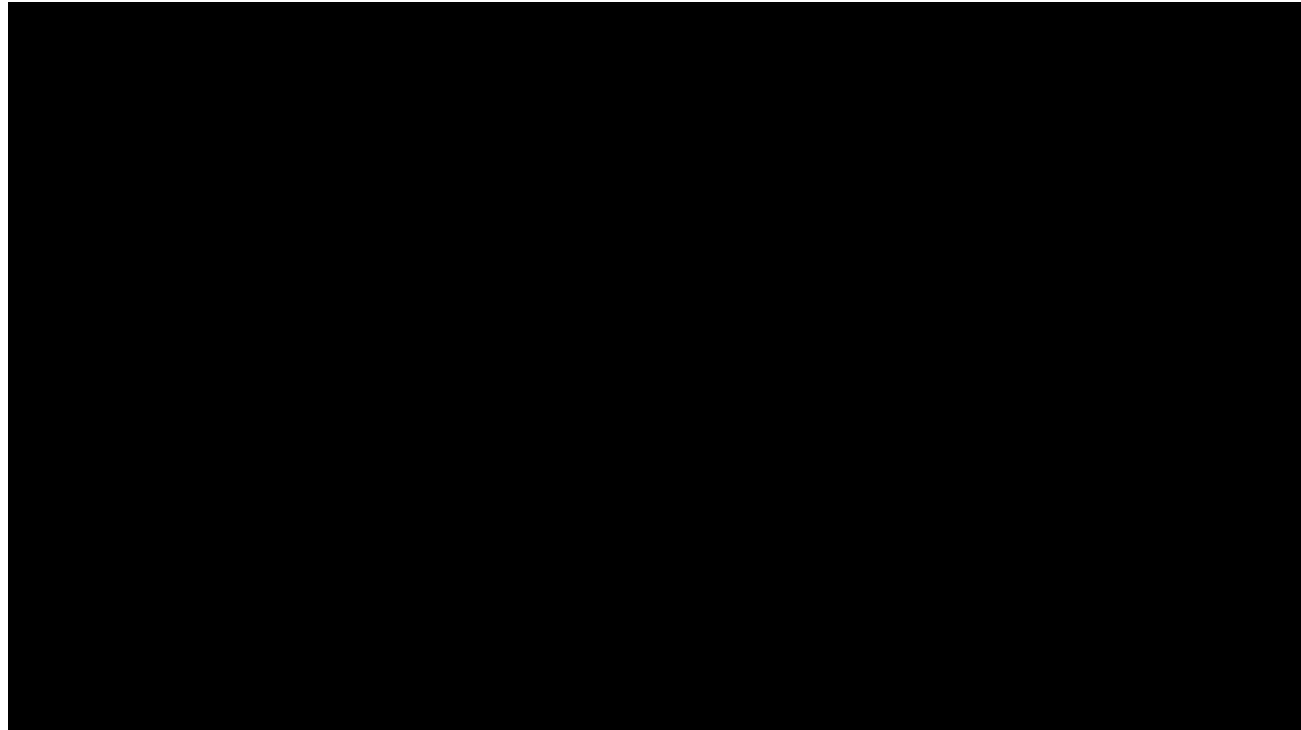


<http://www.youtube.com/watch?v=Krl-YzdVZak>

# Types of Robots



Humanoid

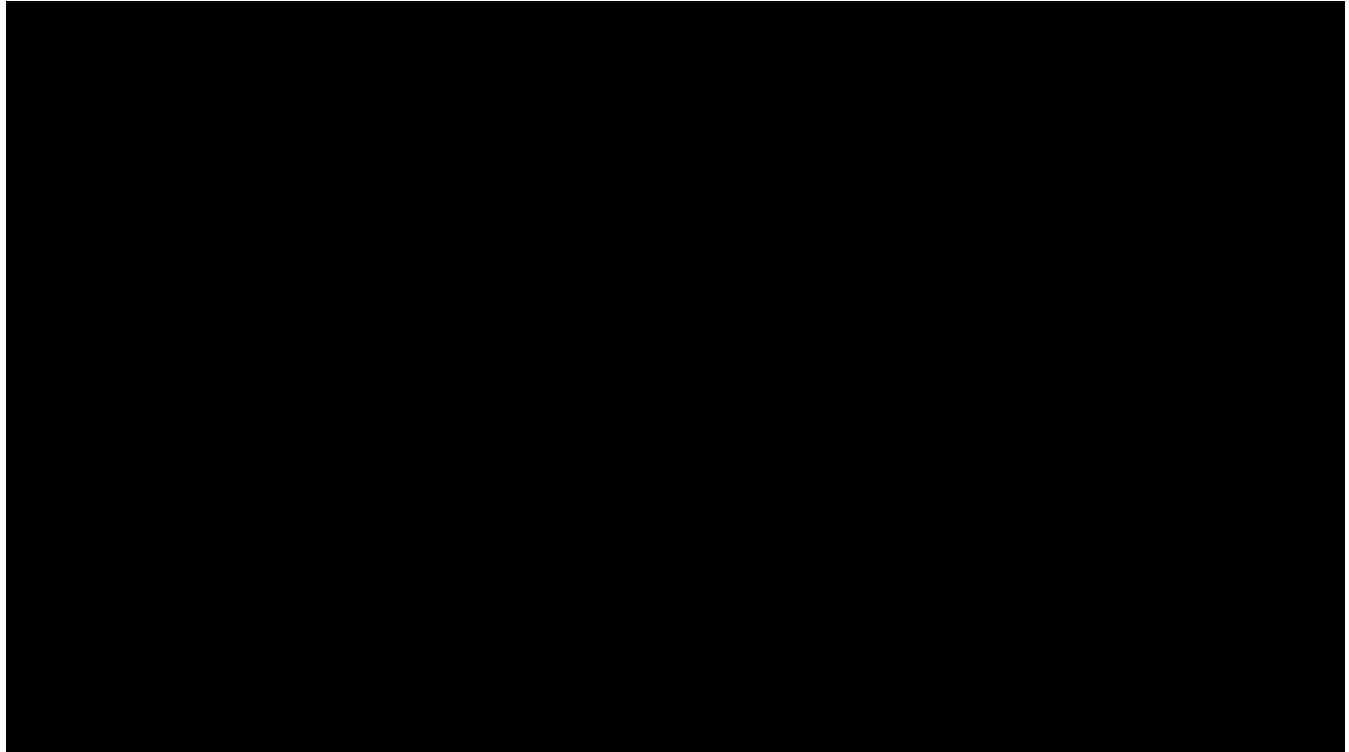


<https://www.youtube.com/watch?v=rVlhMGQgDkY>

# Types of Robots



Aerial robots

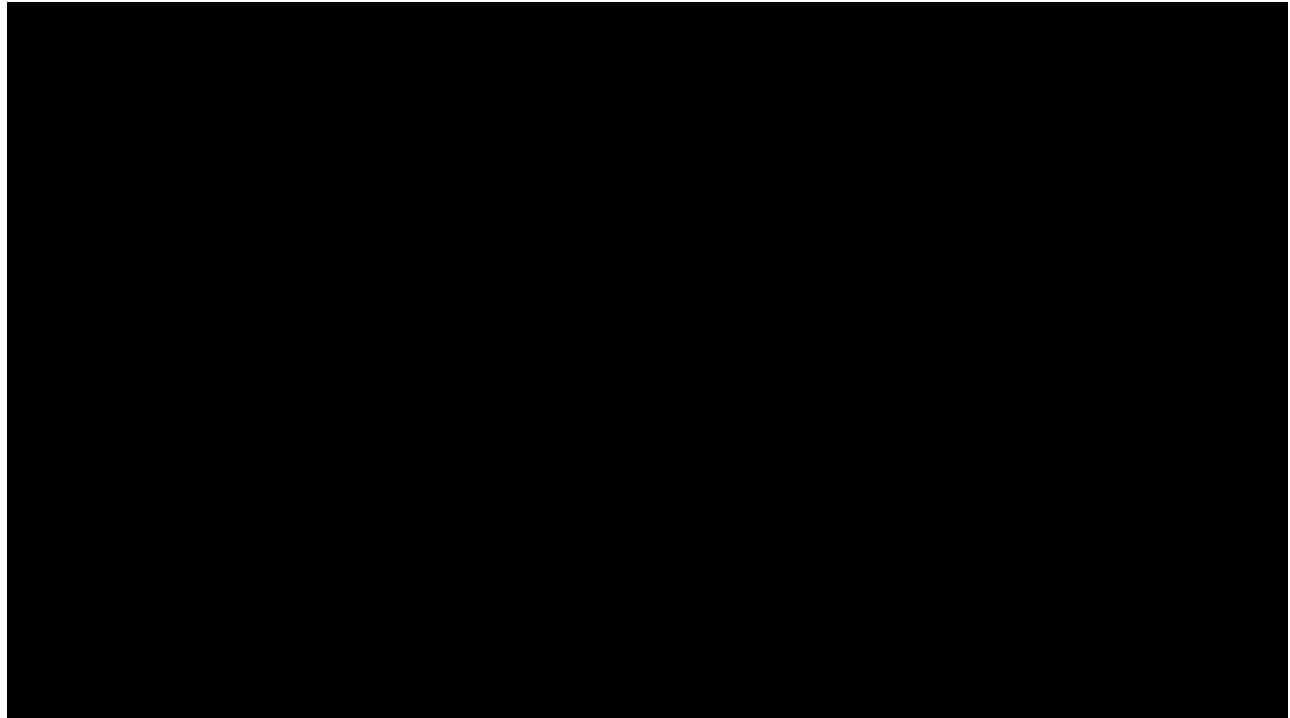


<https://www.youtube.com/watch?v=6uh7zrIbHwM>

# Types of Robots



Underwater robots



[https://www.youtube.com/watch?v=l3IlcUkIC\\_s](https://www.youtube.com/watch?v=l3IlcUkIC_s)

# Types of Robots



Manipulator



Aerial robots



Wheeled mobile robots



Humanoid



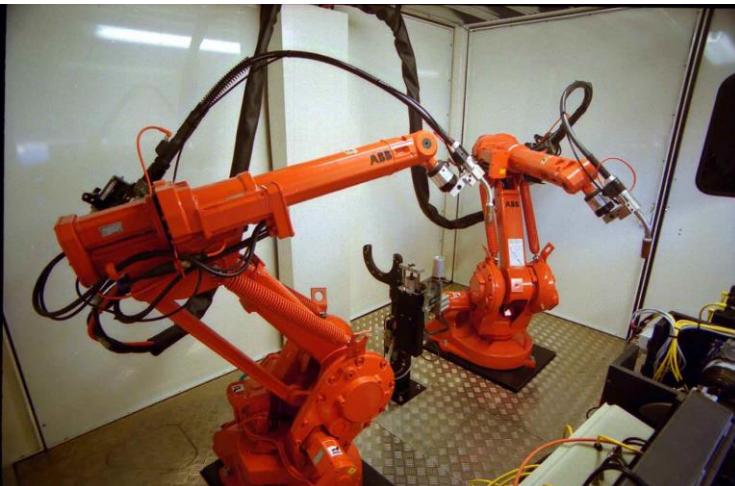
Underwater robots



Legged robots

# Why use Robots?

- Increase product quality
  - Superior Accuracies (thousands of an inch, wafer-handling: microinch)
  - Repeatable precision → consistency of products
- Increase efficiency
  - Work continuously without fatigue
  - Need no vacation



Welding Robot



The SCRUBMATE Robot

# Why use Robots?

- Increase safety
  - Operate in dangerous environment
  - Need no environmental comfort air conditioning, noise protection, etc

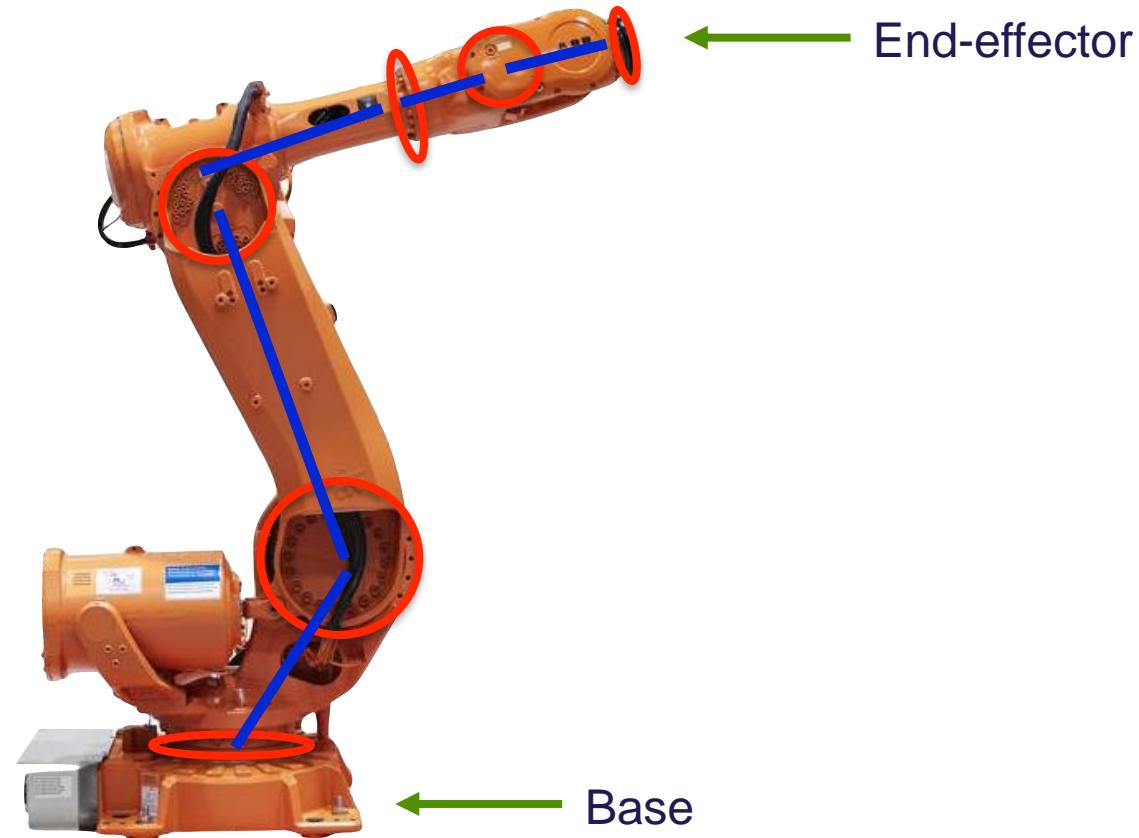


## Decontaminating Robot

Cleaning the main circulating pump housing in the nuclear power plant

# Anatomy of a Robotic Arm

- Base
- End-effector
- Links
- Joints



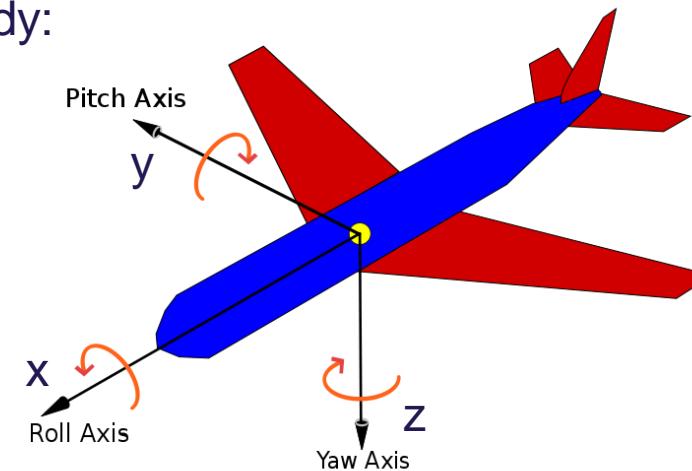
# Degrees of Freedom (DOF)

- Definition:

*A DOF of a mechanical system is the number of independent parameters that define its configuration*

- The general case, for an unrestricted rigid body:

- 3 directional DOF: (x,y,z)
- 3 rotational DOF: (roll, pitch, yaw)



# Choosing the Right Robot

- Workspace
  - Must reach a number of fixtures and workpieces
  - Margin around fixtures and workpieces in order to avoid collisions
  - Consider the shape and singularities
- Load capacity
- Speed
- Repeatability and accuracy
- External interfaces



# Degrees of Freedom (DOF)

- General positioning and orienting requires 6 degrees of freedom (DOF).
- Tasks with rotationally symmetric tool requires only 5 DOF (welding, grinding, polishing etc.)
- Positioning parts on planar surface (pick and place) requires only 4 DOF (X,Y,Z,roll).



# Kinematic Configuration and Workspace

A **configuration** of a manipulator is a complete specification of the location of every point on the manipulator. The set of all possible configurations is called the **configuration space**.

An object is said to have **n** degrees-of-freedom (DOF) if its configuration can be minimally specified by **n** parameters. Thus, the number of DOF is equal to the dimension of the configuration space.

The **workspace** of a manipulator is the total volume swept out by the end-effector as the manipulator executes all possible motions. The workspace is constrained by the geometry of the manipulator as well as mechanical constraints on the joints.

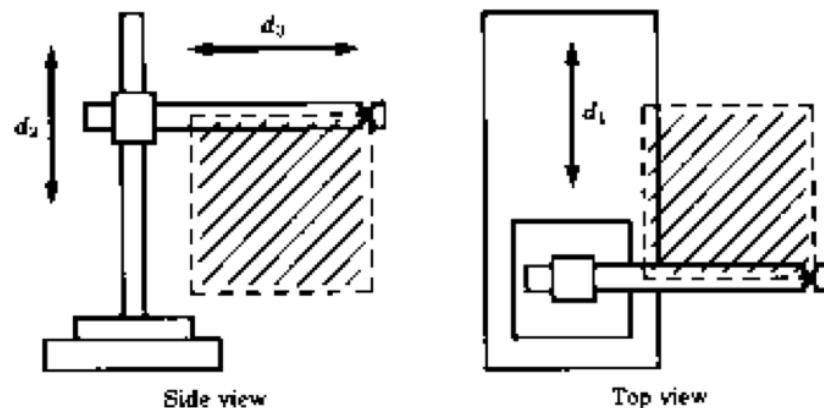
**Reachable workspace:** entire set of points reachable by the manipulator

**Dexterous workspace:** points that the manipulator can reach with an arbitrary orientation of the end-effector.

# Kinematic Configurations

- The type of joints before the wrist determine the difference
- Common kinematic configurations:

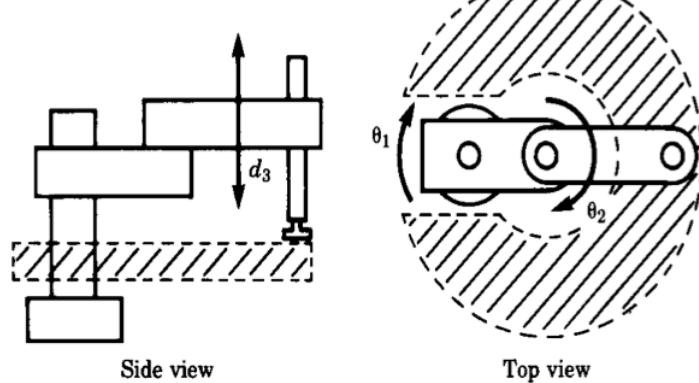
## Cartesian (TTT)



# Kinematic Configurations

- The type of joints before the wrist determine the difference
- Common kinematic configurations:

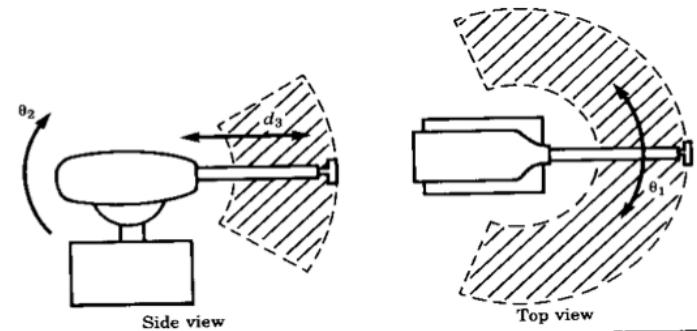
## Scara (RRT)



# Kinematic Configurations

- The type of joints before the wrist determine the difference
- Common kinematic configurations:

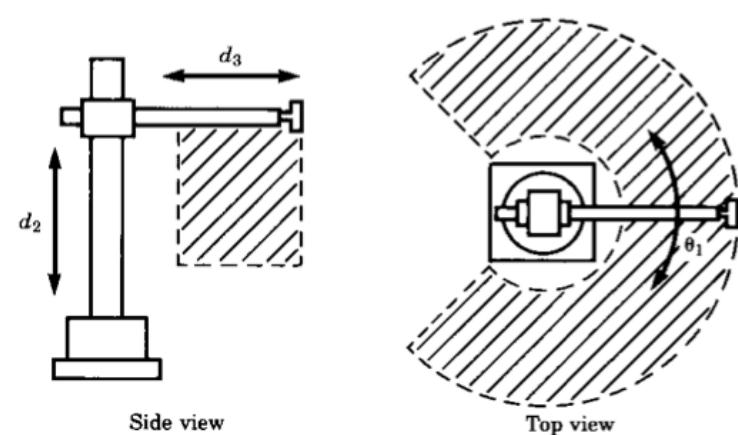
## Spherical (RRT)



# Kinematic Configurations

- The type of joints before the wrist determine the difference
- Common kinematic configurations:

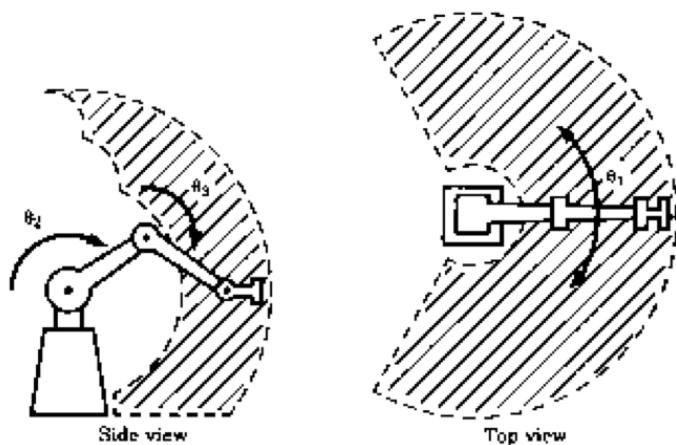
## Cylindrical (RTT)



# Kinematic Configurations

- The type of joints before the wrist determine the difference
- Common kinematic configurations:

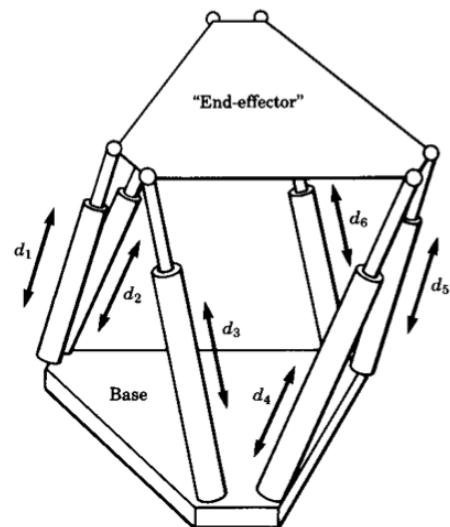
## Articulated (RRR)



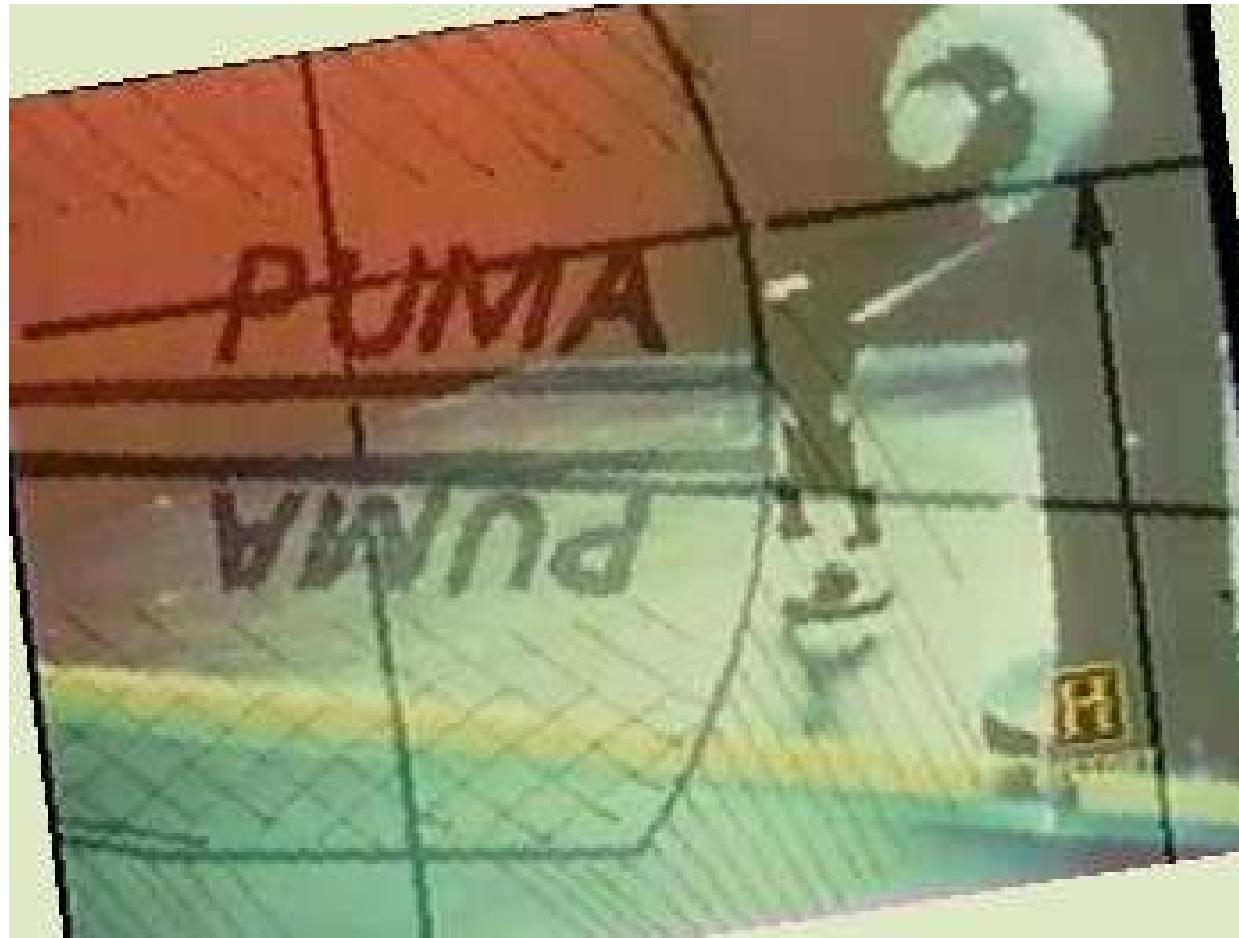
# Kinematic Configurations

- The type of joints before the wrist determine the difference
- Common kinematic configurations:

## Closed structures

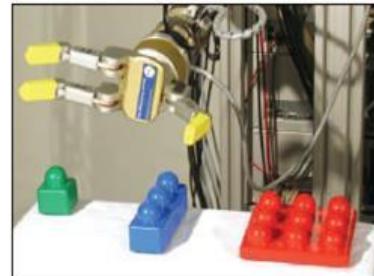


# Industrial Robot – PUMA (1978)

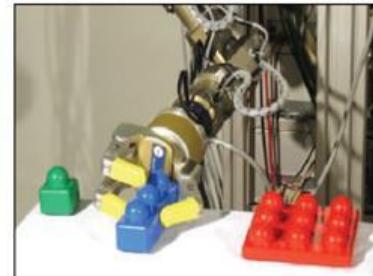


# How are They Used?

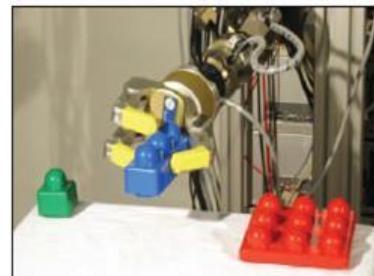
- Industrial robots
  - 70% welding and painting
  - 20% pick and place
  - 10% others
- Research focus on
  - Manipulator control
  - End-effector design
    - Compliance device
    - Dexterous robot hand
  - Visual and force feedback



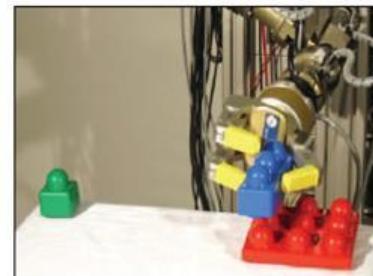
a)



b)



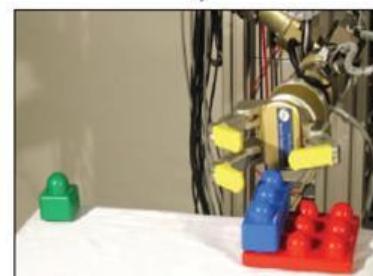
c)



d)



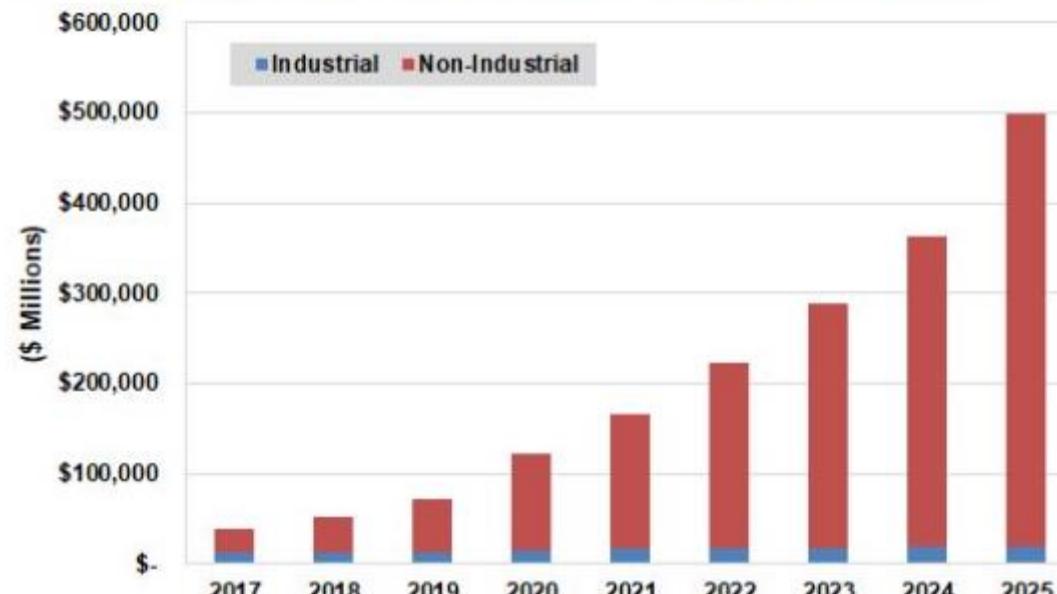
e)



f)

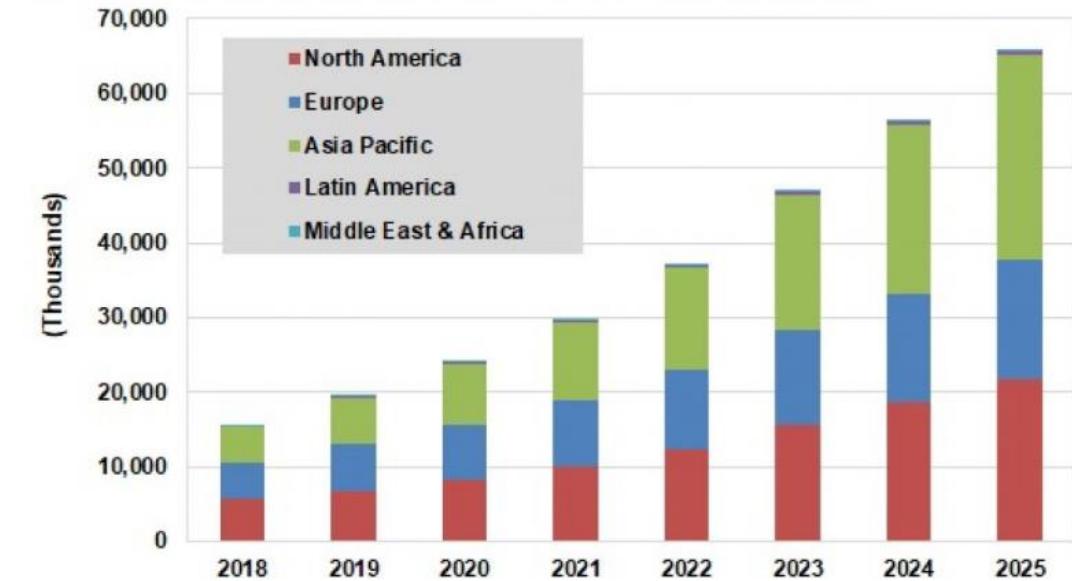
# Industrial and Non-Industrial Robots Forecast

Total Industrial and Non-Industrial Robotics Revenue, World Markets: 2017-2025



Source: Tractica

Consumer Robotics Shipments by Region, World Markets: 2018-2025



Source: Tractica

# Service Robots



floor cleaning



human guider



entertainment



lawn-mower



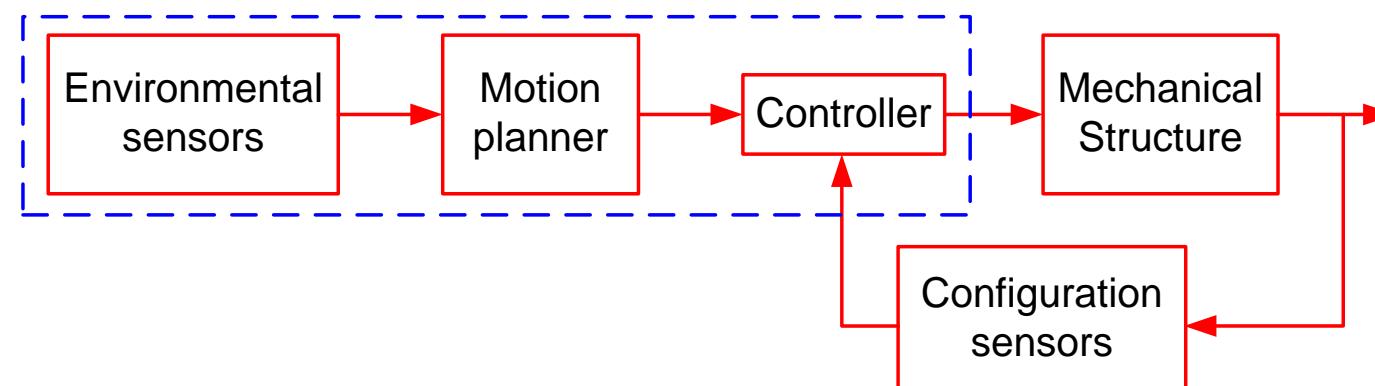
vacuum cleaner



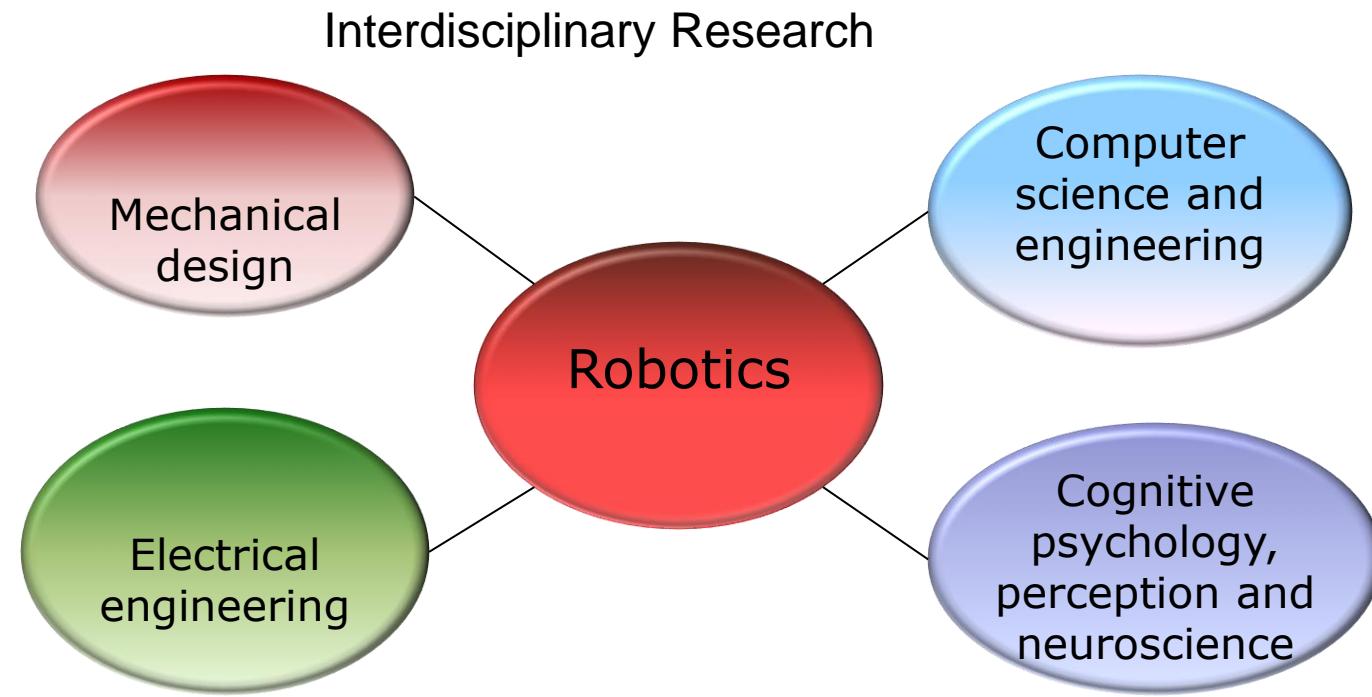
tennis ball collector

# Architecture of Robotic Systems

- Mechanical Structure
  - Kinematics model
  - Dynamics model
- Actuators: electrical, hydraulic, pneumatic, artificial muscle
- Computation and controllers
- Sensors
- Communications
- User interface



# Summary



- Open problems

- Manipulation, Locomotion
  - Human-Robot Interaction

- Control, Navigation
  - Learning & Adaptation (AI)

Robotics – 34753

# Compulsory Project Assignment

Konstantinos Poulios  
Associate Professor

Department of Civil and Mechanical Engineering  
DTU Lyngby, building 404 / room 124

# Project Assignment – Basic Info

- The project assignment is compulsory and forms part of the final grade
- Executed in groups of 5 – enroll yourself in a group on DTU Learn
- The assignment focuses mostly on kinematics and computer vision (minimal use of control theory)
- The assignment consists of 5 parts. Parts 1-4 are theoretical and part 5 is practical. Part 5 weighs significantly more.
- The practical part requires access to a real robot arm in the lab, that you need to book in advance

# Educational 4-Joint Robot Arm



- DIY robot based on Dynamixel servos
- Platform for learning kinematics and computer vision
- Certain degree of self-learning required

# Project Assignment – Structure

- Part 1 (problems 1-5): Forward, inverse, and velocity kinematics
- Part 2 (problems 6-7): Trajectory planning
- Part 3 (problems 8-9): Singularities and motor loads (static)
- Part 4 (problem 10): Dynamics
- Part 5 (problem 11): Practical robot task, choosing one of:
  - Picking smarties
  - Line following
  - Keyboard typing
  - ....
  - Your own idea approved by the course responsible

# Project Assignment – Additional Info for Part 5

- Part 5 can be implemented in Matlab or Python
- You will receive sample code in Matlab and Python that you can use as an inspiration but not as a black box. You should not use other black-box robotics frameworks, except for verification of your own implementations.
- You need to upload a video showing the physical robot performing the desired task
- If you have your own robot arm with 4 joints or more, you can use that instead, after agreement with the course responsible
- In part 5, we suggest some representative tasks for the robot, but you can come up with a complex task, and get permission by course responsible to work on it instead
- This part requires a significant amount of self-learning, especially with regard to programming aspects and APIs for control of the robot servos and computer vision

# Project Assignment – Hand-in Info

- Assignment text will be uploaded to DTU Learn in course week 2
- Be aware that revisions with corrections may be uploaded during the semester
- Hand-in date will be in the first week of December (exact date will be available on DTU Learn)
- Support answers to the problems with sufficient text and intermediate calculations. Goal is to make sure that your explanation is clear, showing the principles and methods used to solve the problems.
- Upload one report per team
- Specify own contributions in the report, do not forget to state your student number
- Specify license for your uploaded videos (preferably creative commons) and code

# Inspirational Videos

# Group Formation

Robotics – 34753

# Robot Kinematics I

Konstantinos Poulios  
Associate Professor

Department of Civil and Mechanical Engineering  
DTU Lyngby, building 404 / room 124

# Robot Kinematics I – Lecture Overview

## 1. Robot Manipulators

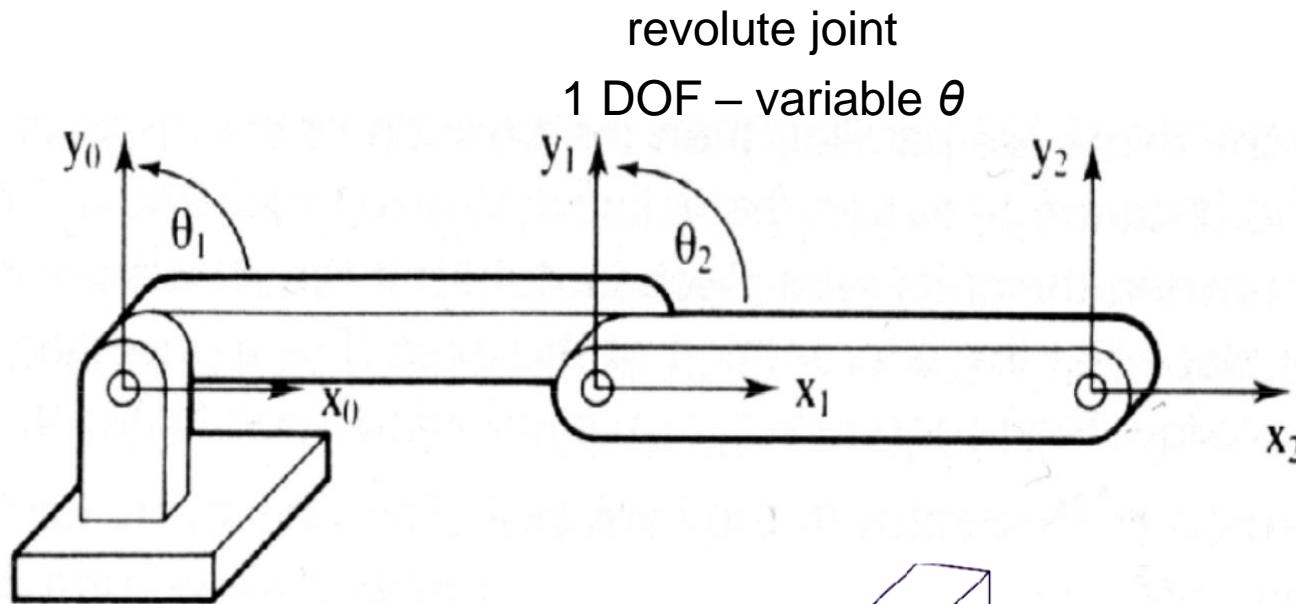
- Revolute/prismatic Joints
- Robot grippers
- Robot configurations
- Workspace

## 2. Kinematics

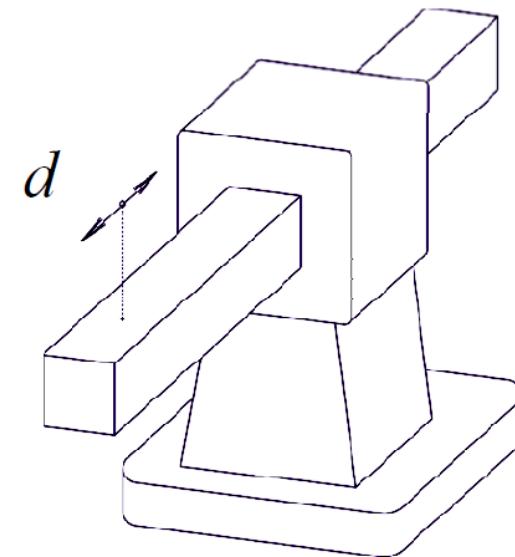
- Position representation
- Rotation/Orientation representation
- Compositions of rotations
- Rotation parametrization – Euler angles
- Homogeneous transformation matrices

# Robot Joints

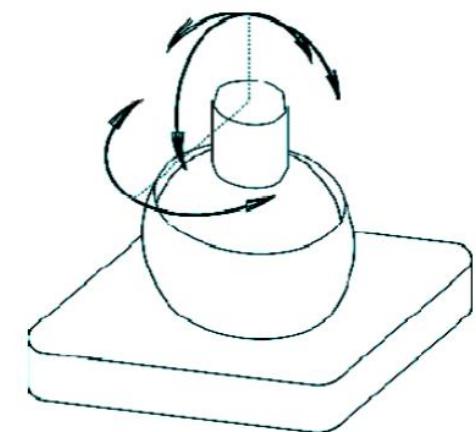
# Joints



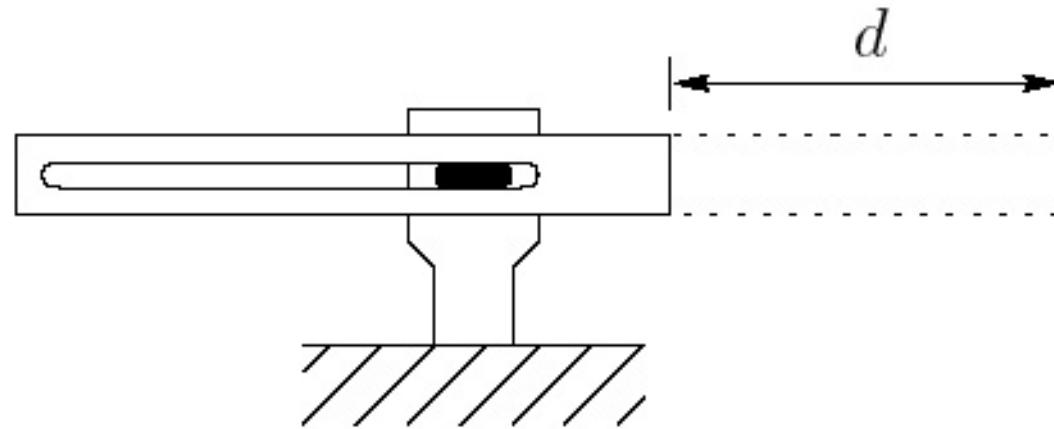
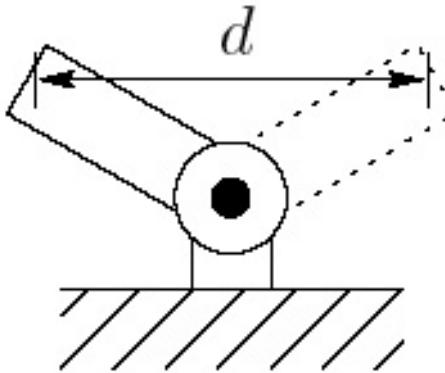
prismatic joint  
1 DOF – variable  $d$



spherical joint  
(not considered in the course)  
3 DOF – variables  $\theta_1, \theta_2, \theta_3$



# Revolute vs Prismatic Joints



- Prismatic link: covers distance **equal to** the length of the link
- Rotational link: covers distance **twice as** the length of the link

Revolute joint:

- Compact
- Simple construction
- Even transmission of constraint forces
- Easy to lubricate
- Rather insensitive to dirt
- Difficult to produce linear motion (in limited space)



<https://rozum.com/>

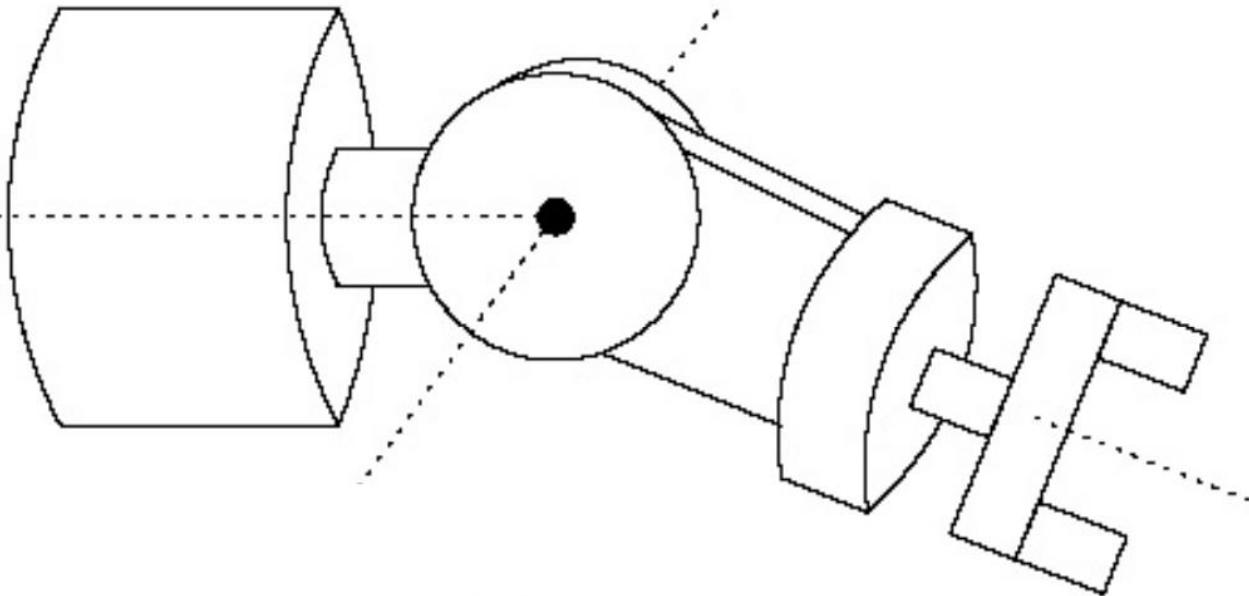
Prismatic joint:

- Larger
- More complex construction
- Uneven reaction forces
- More difficult to lubricate
- More sensitive to dirt
- Produces linear motion

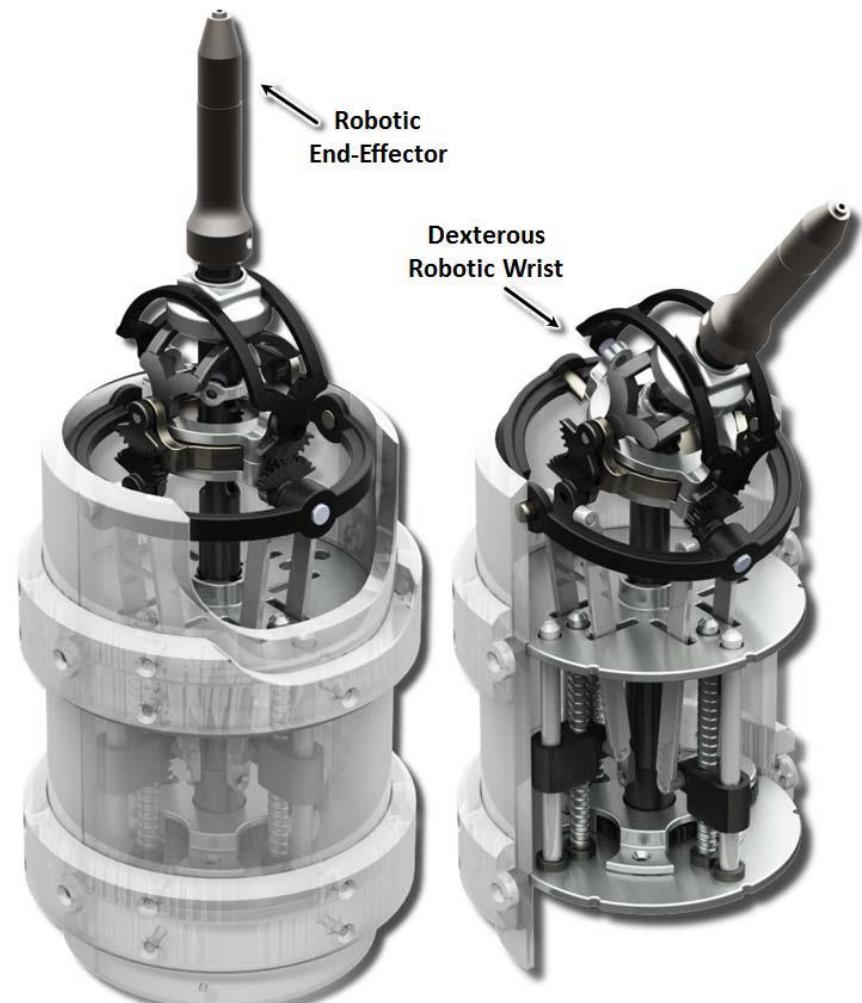


<https://medias.schaeffler.com/>

# Spherical Wrist Joint



- Three axes of rotation: roll, yaw pitch
- All three axes intersect at a single point: wrist center point



Hammond et al. 2013

# Robot Grippers

# Robot Grippers

<https://thinkbotsolutions.com>



Two-fingered  
parallel jaw  
gripper



Parallelogram  
gripper



Adaptive  
gripper

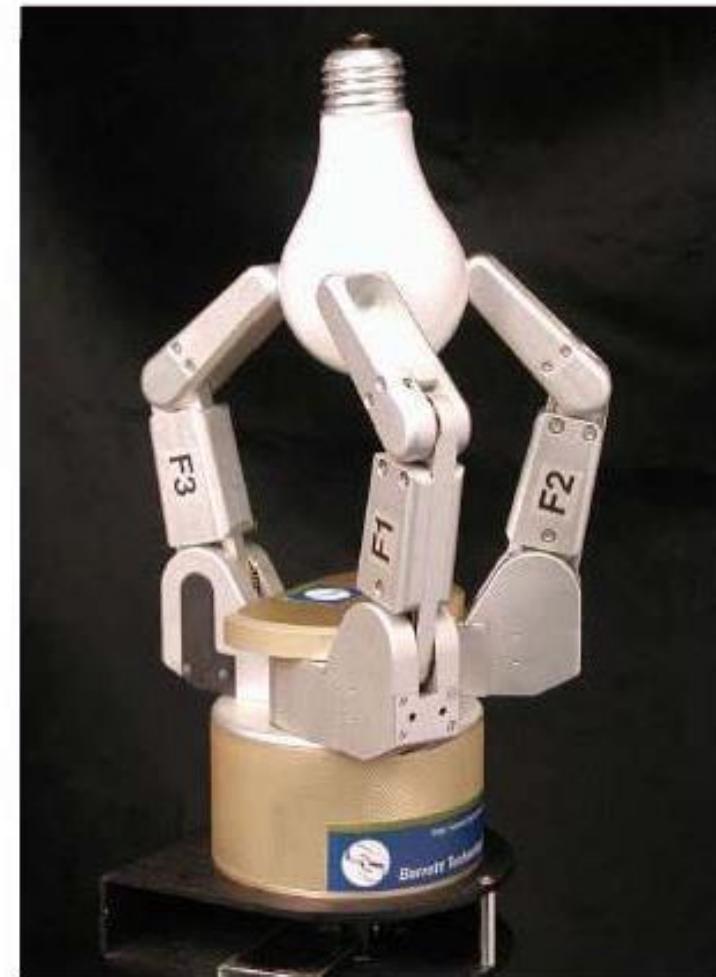


Bajaj et al. 2019

Scissor  
gripper

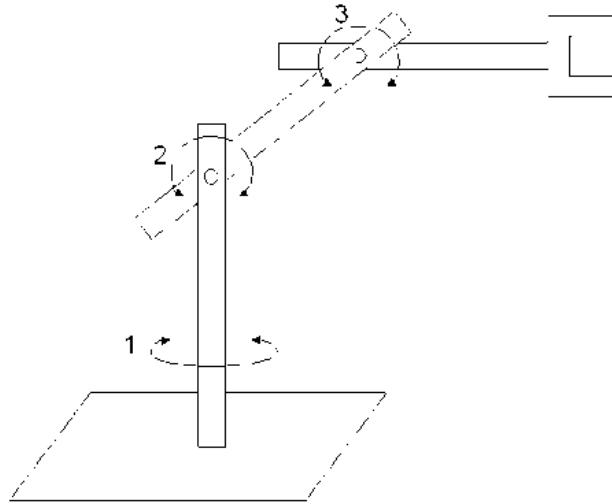
# Three-fingered Anthropomorphic Hand

- More dexterity
- Better ability to manipulate objects of various sizes and geometries

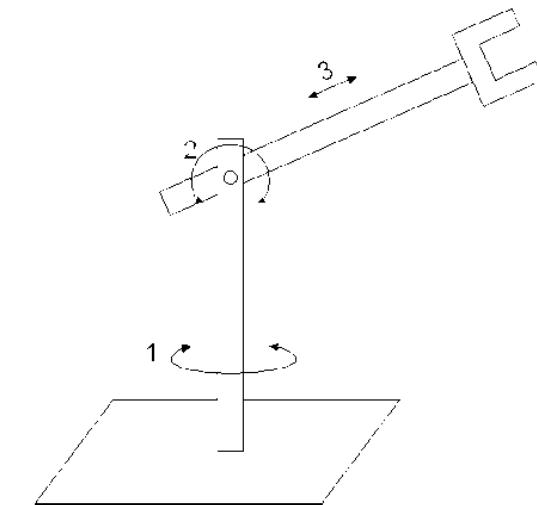


# Robot Configurations

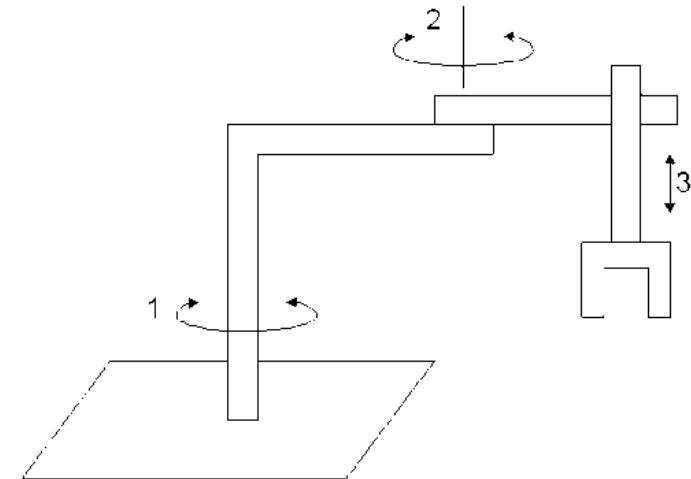
# Robot Arm Examples



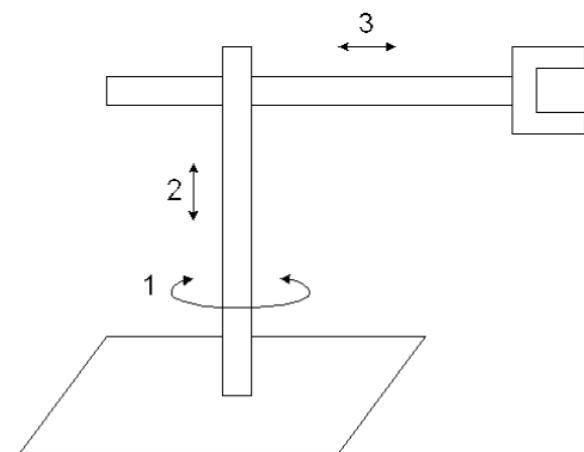
Articulated: RRR



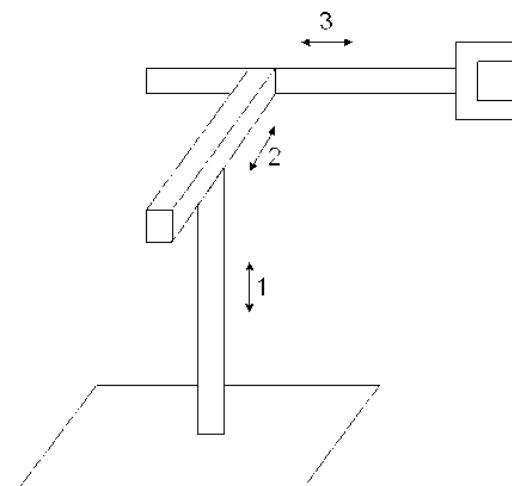
Spherical: RRP



SCARA: RRP



Cylindrical: RPP



Cartesian: PPP

# Articulated Manipulator (RRR)

- A.k.a. revolute/elbow/anthropomorphic manipulator
- $z_2 \parallel z_1$  ,  $z_1 \perp z_0$  ,  $z_2 \perp z_0$

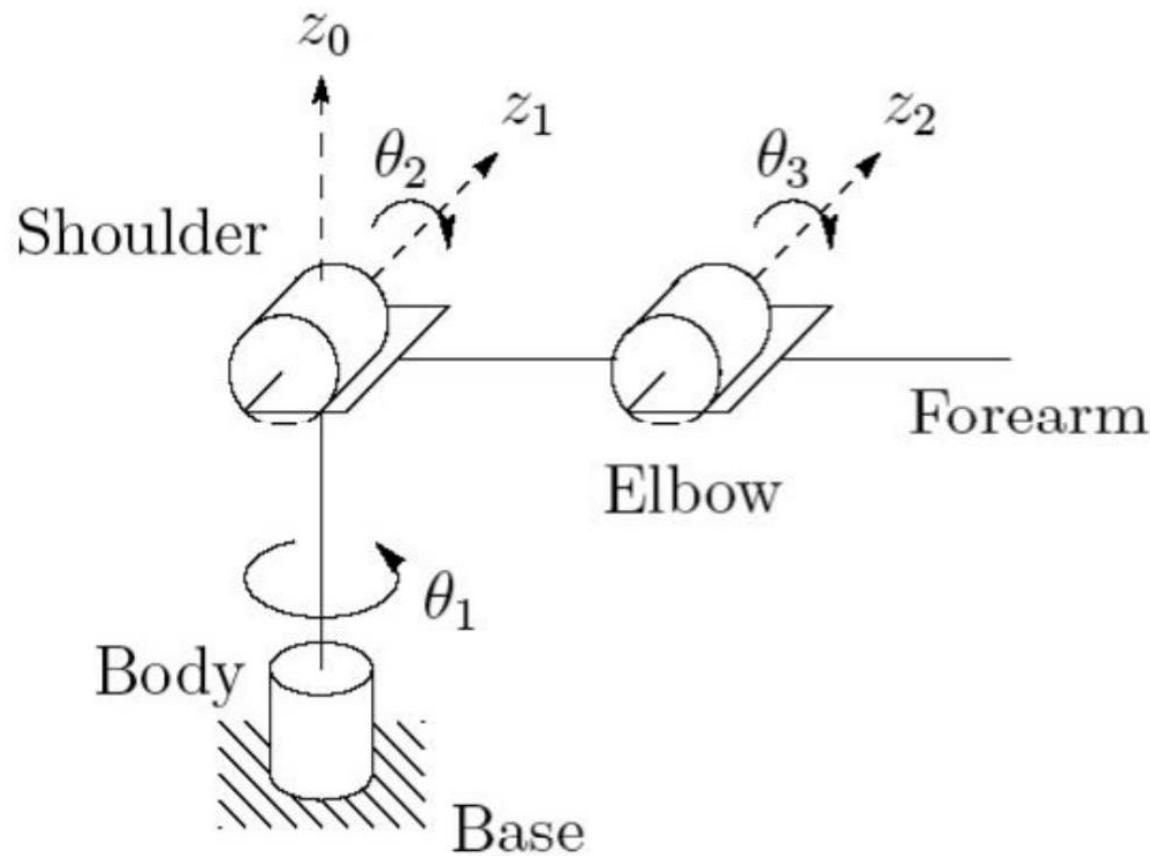
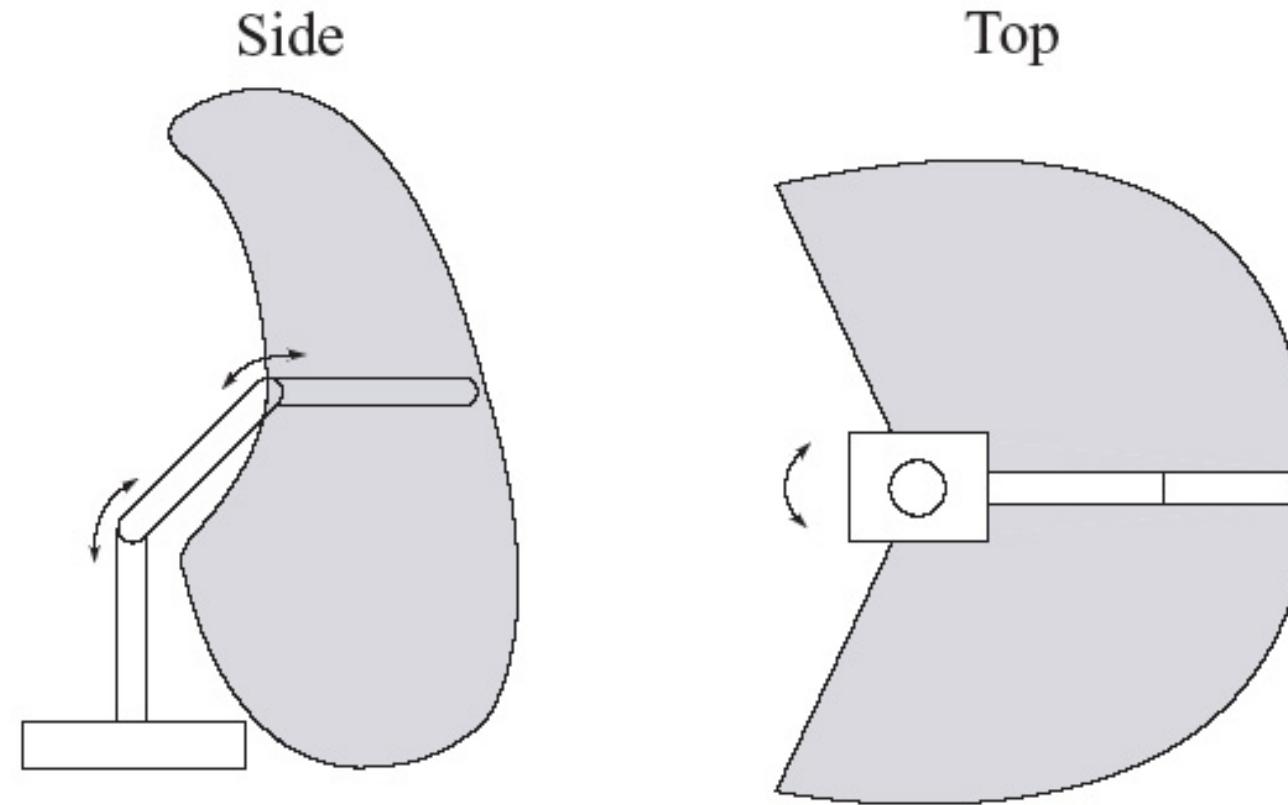


ABB IRB1400 Robot

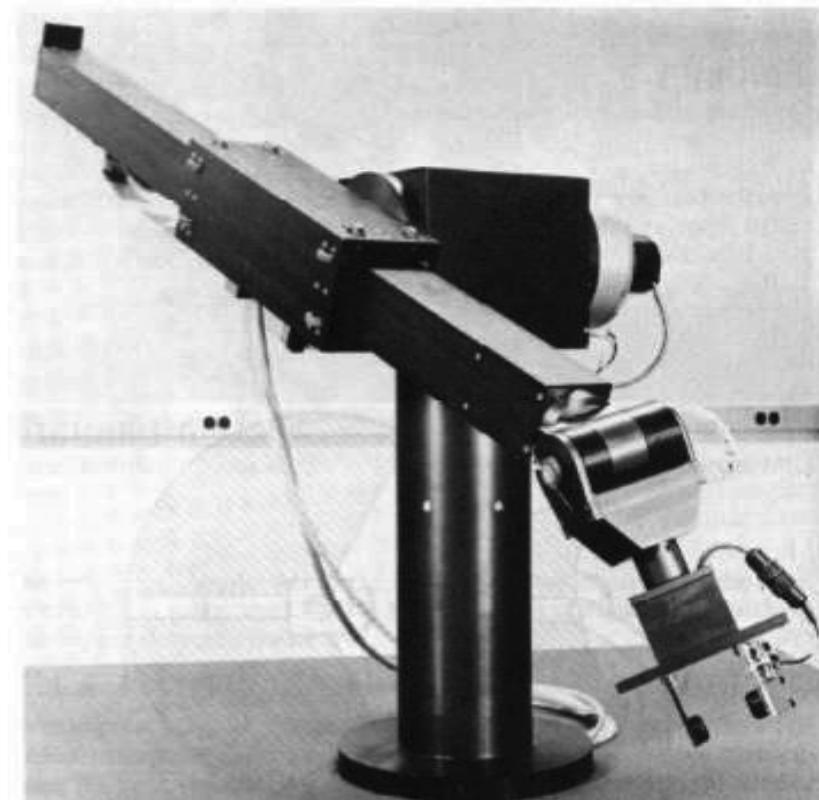
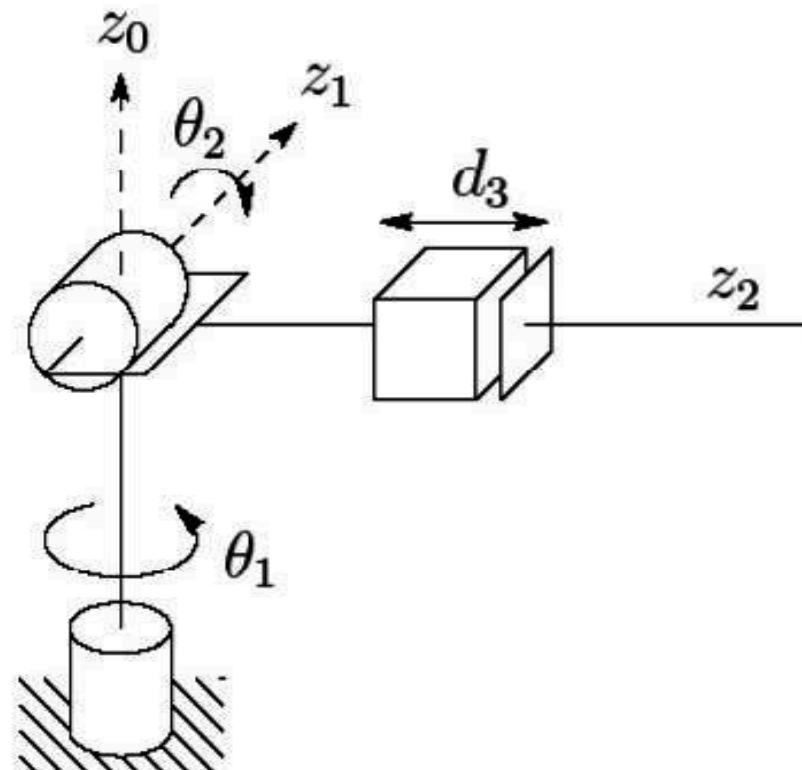
# Workspace of the Articulated Manipulator (RRR)



Provides a larger workspace than other kinematic designs

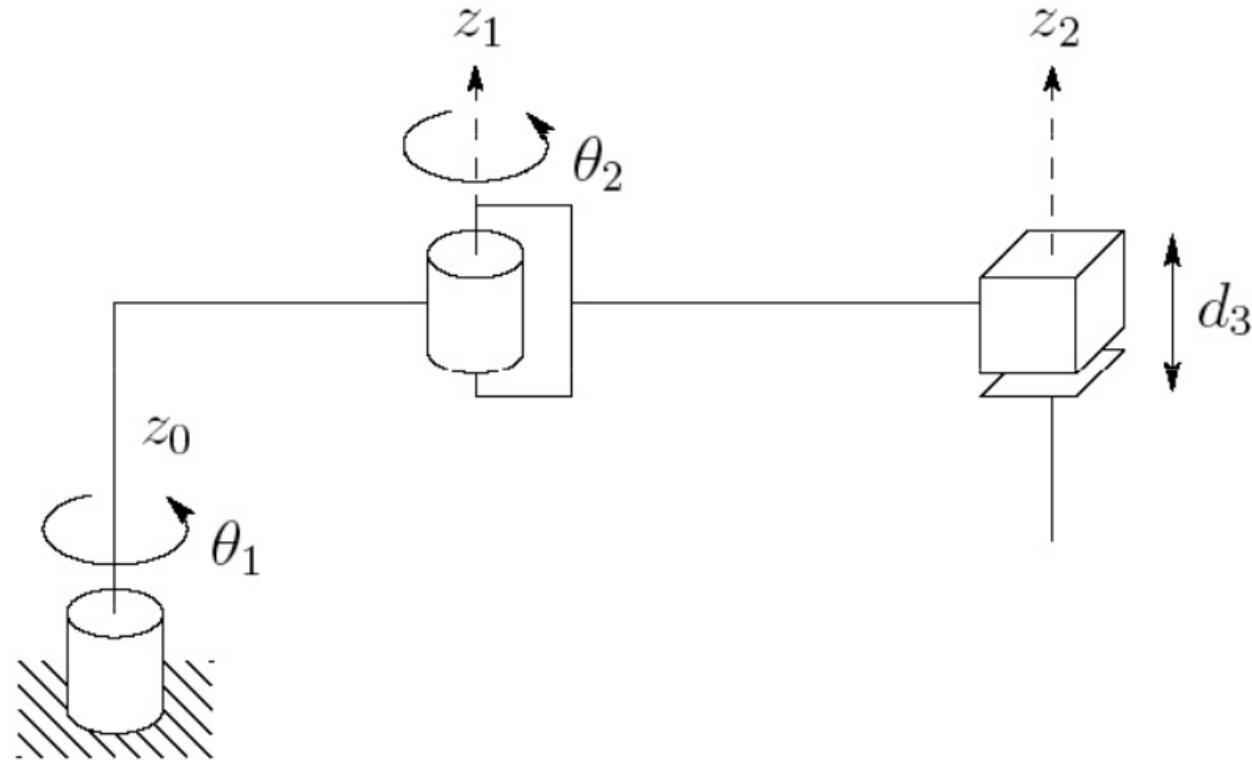
# Spherical Manipulator (RRP)

- ‘Spherical’ because the joint coordinates coincide with the spherical coordinates of the end-effector
- All three joint axes cross through a single point



# SCARA Manipulator (RRP)

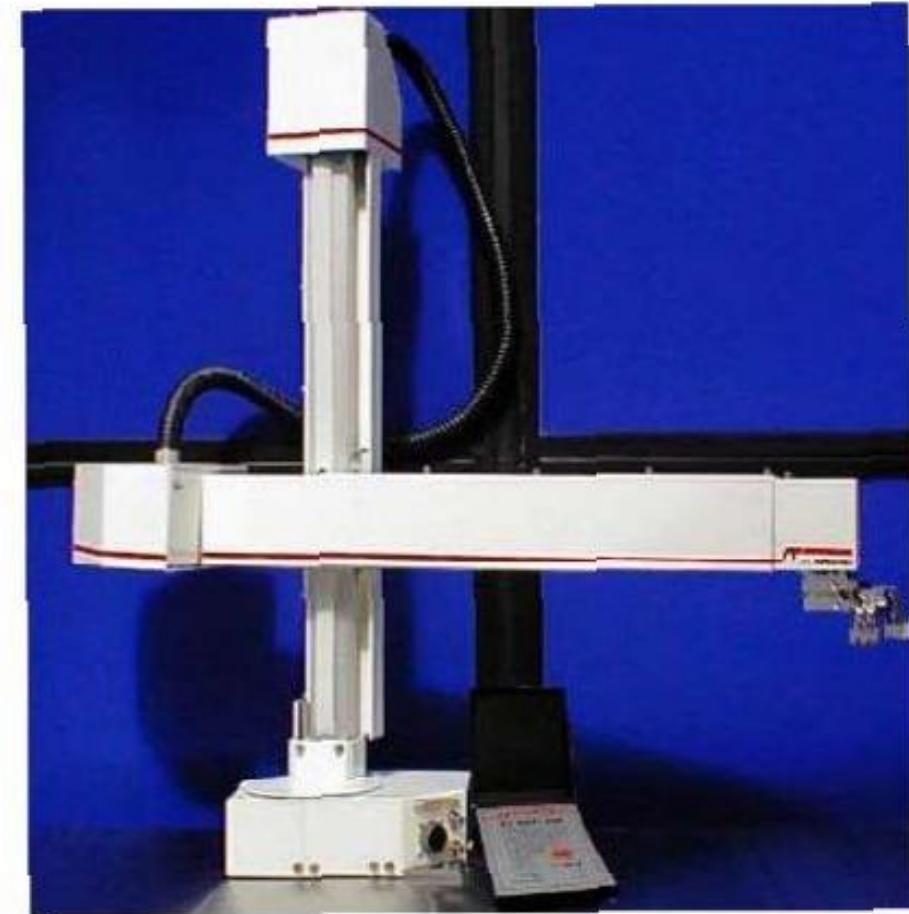
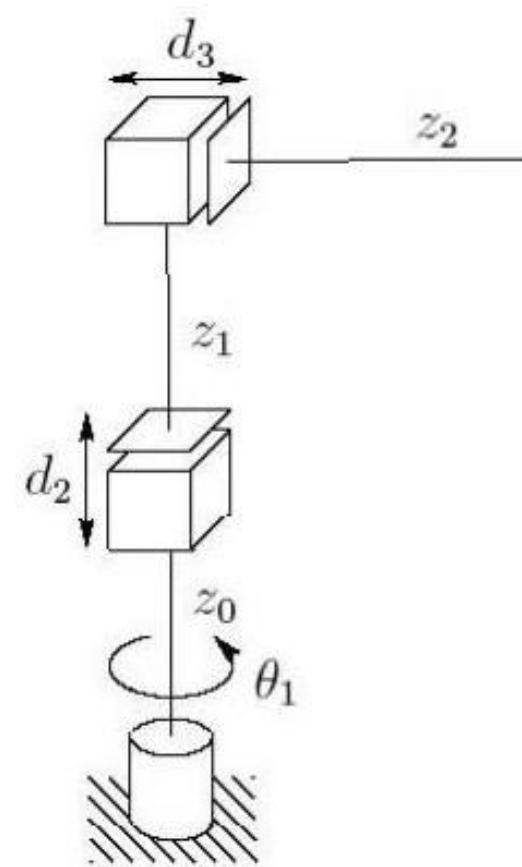
- SCARA: Selective Compliant Articulated Robot for Assembly
- $z_0 \parallel z_1 \parallel z_2$
- Ideal for table-top assembly, pick-and-place, packaging



The Adept Cobra  
Smart 600  
SCARA arm

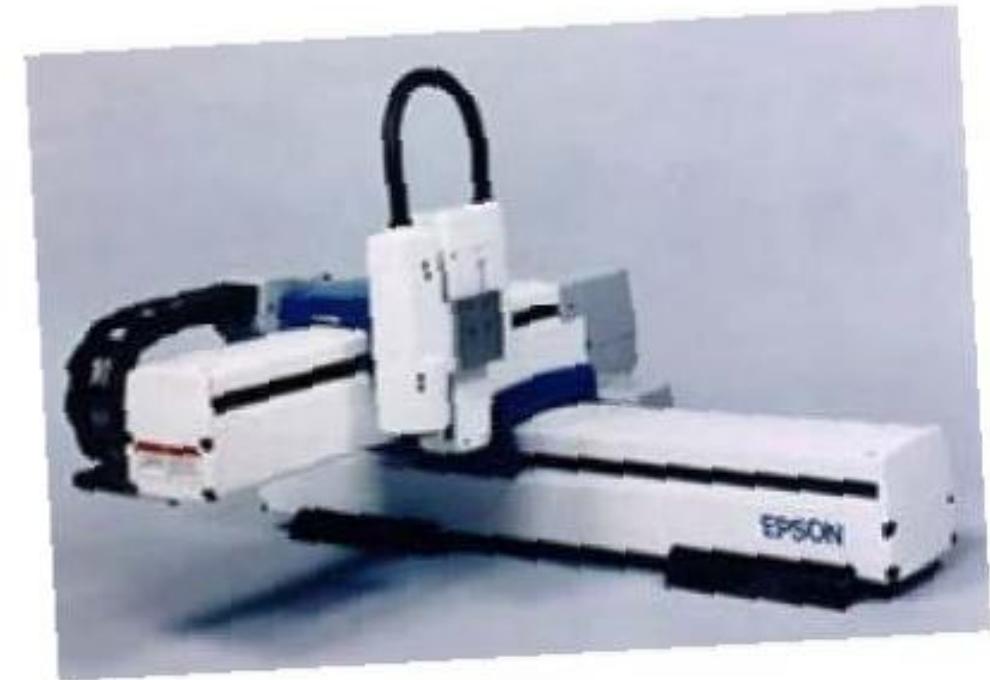
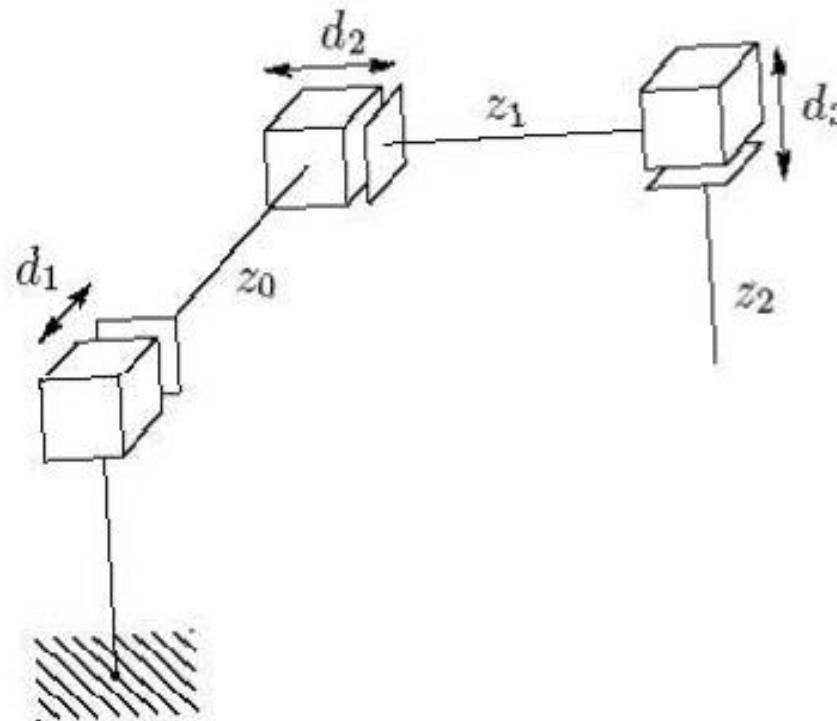
# Cylindrical Manipulator (RPP)

- Typically used in material transfer applications



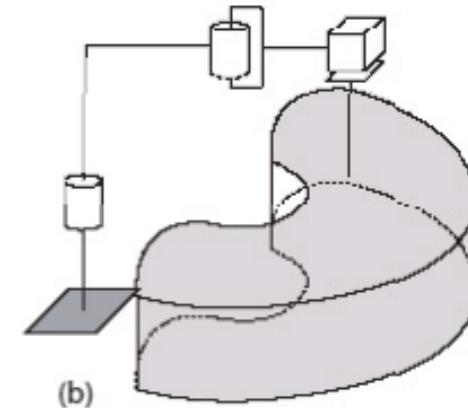
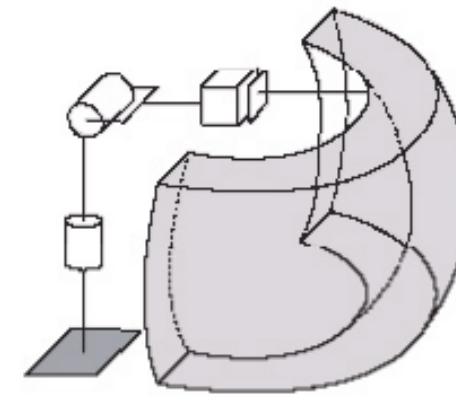
# Cartesian Manipulator (RPP)

- Simplest kinematic description
- Suitable for table-top assembly application, transfer of material or cargo



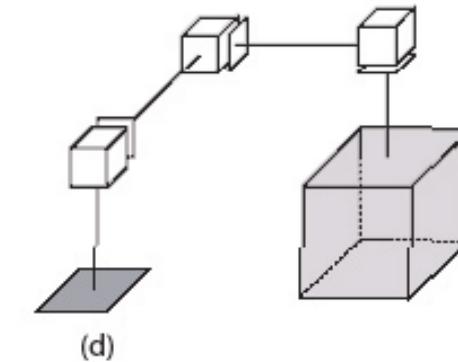
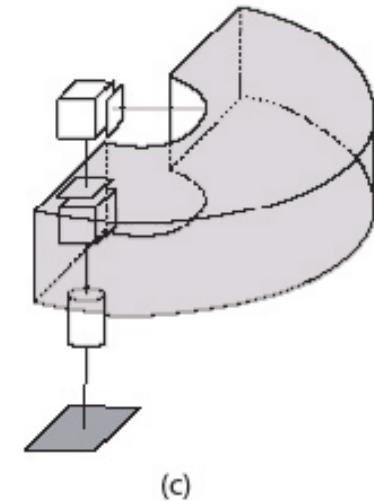
# Workspace of Common Manipulators

Spherical robot



SCARA robot

Cylindrical robot



Cartesian robot

# Parallel Manipulator

- A subset of the links form a closed chain
- Two or more kinematic chains connecting the base with the end-effector
- Greater rigidity and accuracy
- Fundamentally different to serial link robots

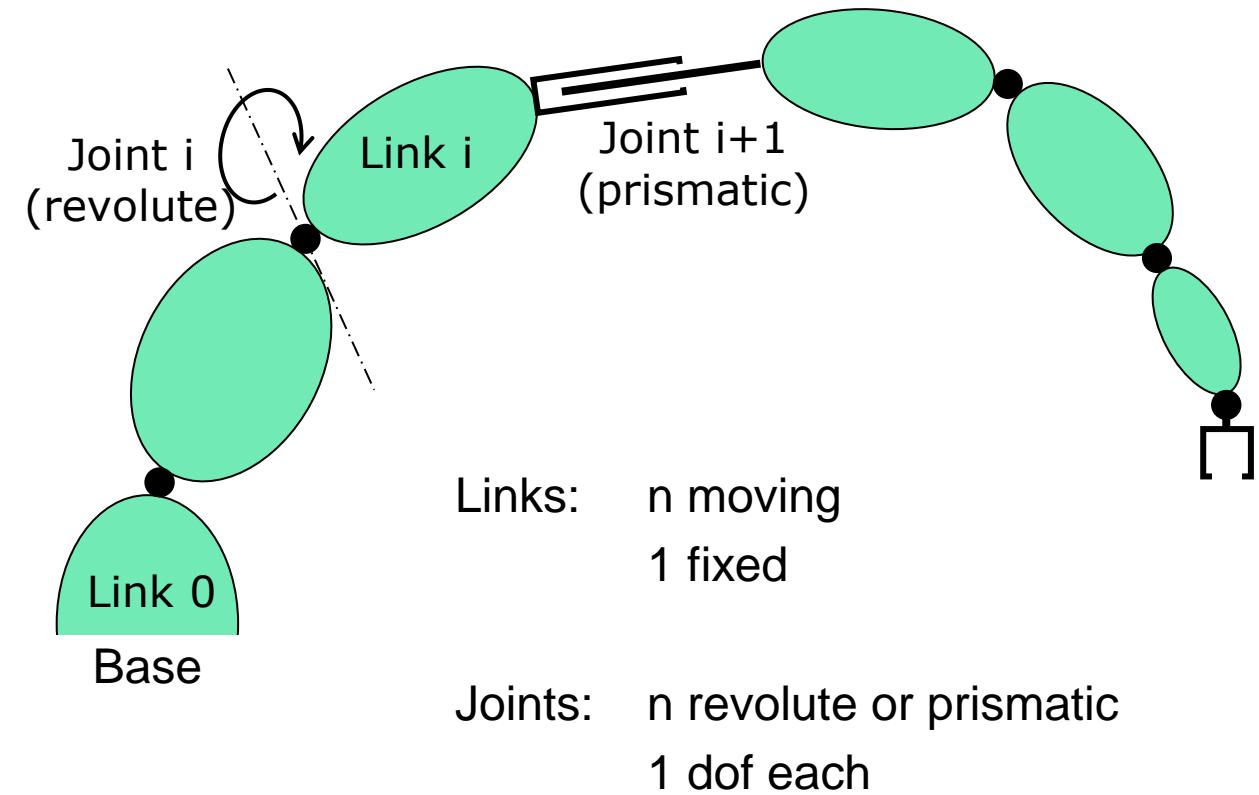


ABB IRB940 Tricept parallel robot

# Kinematics

# Kinematic Model of Robot Arm

- Serial link manipulator (a.k.a. robot arm, industrial robot)
  - An open chain of rigid bodies (links) connected by joints (revolute or prismatic)
- Manipulator specification
  - Degrees of freedom:  $n$
  - Joint space
  - Work space
  - Redundancy:  $n > m$



# Kinematics Notation and Terminology

- Definition: studying position and motion **without** considering **forces** (purely geometrical)
- Synthetic vs analytic approach
- Notation for representing position       $p \equiv p^0 \equiv p^1$
- Notation for representing coordinate frame position       $o_1^0$       versus     $o_0^1$
- Notation for representing a free vector       $v_a \equiv v_a^0 \equiv v_a^1$
- Rotation/Orientation in 2D/3D
- Position + Orientation = Placement

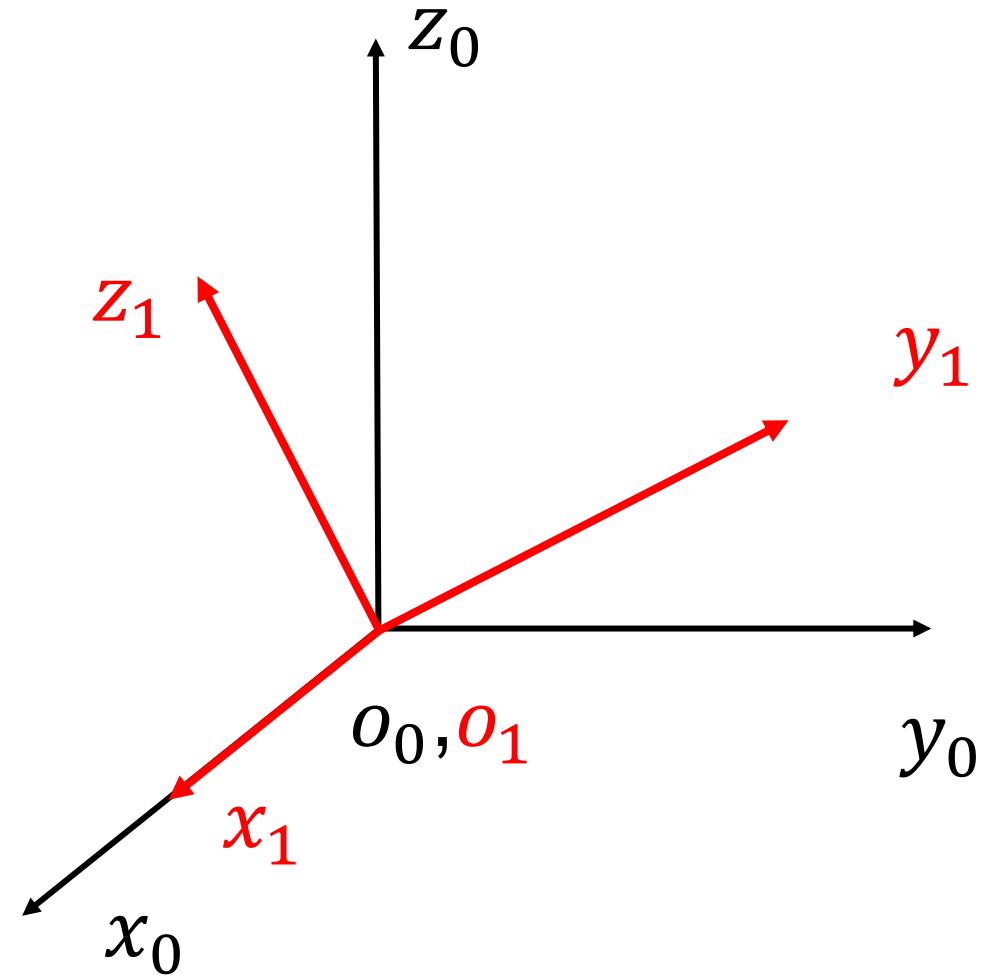
# Rotation Matrix for Frame Orientation

- Basic rotation matrix between frame  $o_0x_0y_0z_0$  and frame  $o_1x_1y_1z_1$

$$R_1^0 = \begin{bmatrix} x_0 \cdot x_1 & x_0 \cdot y_1 & x_0 \cdot z_1 \\ y_0 \cdot x_1 & y_0 \cdot y_1 & y_0 \cdot z_1 \\ z_0 \cdot x_1 & z_0 \cdot y_1 & z_0 \cdot z_1 \end{bmatrix}$$

$$= [x_1^0 \mid y_1^0 \mid z_1^0]$$

$$R_0^1 = (R_1^0)^{-1}$$



# Basic Rotation Matrices

- About x-axis with  $\theta$

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & -s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix}$$

- About y-axis with  $\theta$

$$R_{y,\theta} = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}$$

- About z-axis with  $\theta$

$$R_{z,\theta} = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Properties:

$$R_{z,\theta} R_{z,\phi} = R_{z,\theta+\phi} \quad (R_{z,\theta})^{-1} = R_{z,-\theta}$$

# Rotation Matrix for Frame Transformation

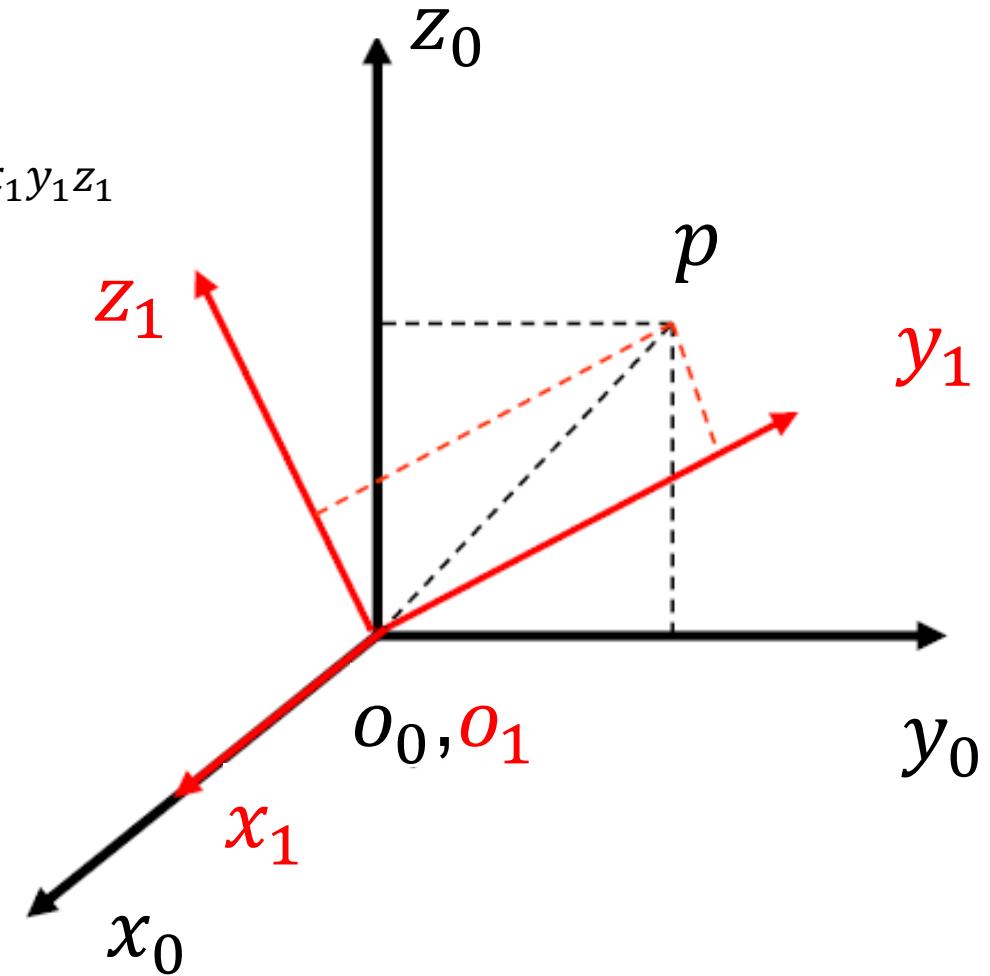
- Position vector

$$p \equiv p^0 \equiv p^1$$

- Basic rotation matrix between frame  $o_0x_0y_0z_0$  and frame  $o_1x_1y_1z_1$

$$p^0 = R_1^0 p^1$$

$$p^1 = (R_1^0)^{-1} p^0$$



# Rotation Matrix as Operator

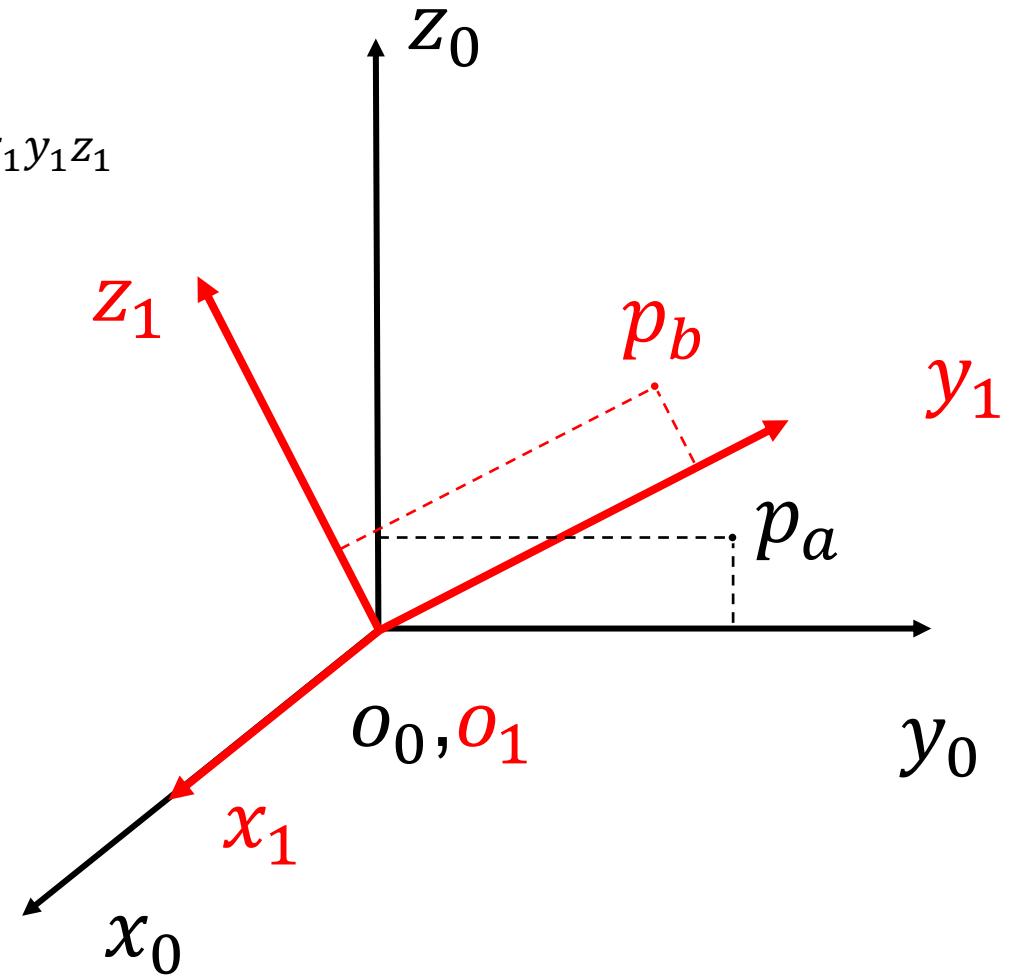
- Position vector

$$p_a^0 \rightarrow p_b^0$$

$$p_a \not\equiv p_b$$

- Basic rotation matrix between frame  $o_0x_0y_0z_0$  and frame  $o_1x_1y_1z_1$

$$p_b^0 = R_1^0 p_a^0$$



# The Three Interpretations of a Rotation Matrix

- Describe the orientation of a frame 1 w.r.t. another frame 0

$$R_1^0 = [x_1^0 \mid y_1^0 \mid z_1^0]$$

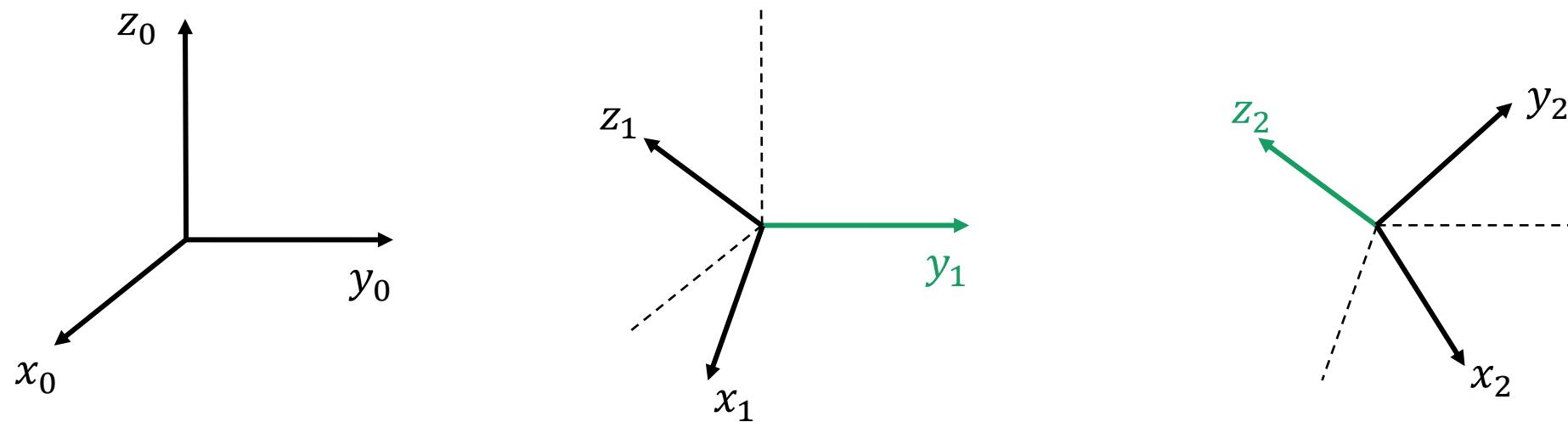
- Coordinates transformation for a fixed point from reference frame 1 to reference frame 0

$$p^0 = R_1^0 p^1$$

- Operator for rotating a point with the same rotation as the rotation between two frames

$$p_b^0 = R_1^0 p_a^0$$

# Composition of Rotations (3D)



$$R_{y,45^\circ} = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix}$$

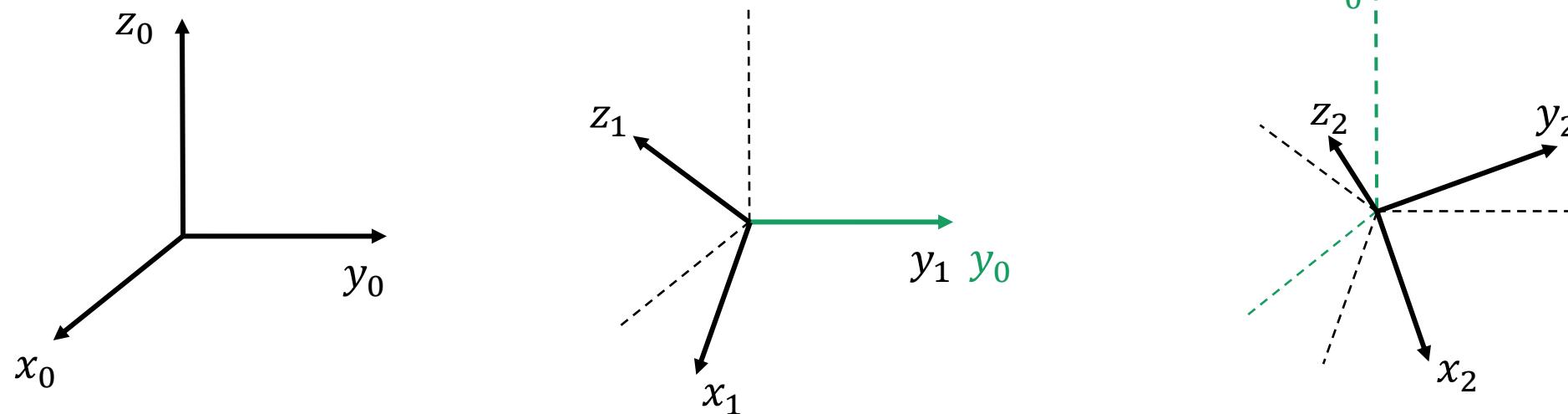
$$R_{z,45^\circ} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_2^0 = [x_2^0 \mid y_2^0 \mid z_2^0] = [R_1^0 x_2^1 \mid R_1^0 y_2^1 \mid R_1^0 z_2^1] = R_1^0 [x_2^1 \mid y_2^1 \mid z_2^1] = R_1^0 R_2^1 = R_{y,45^\circ} R_{z,45^\circ}$$

$$\Rightarrow R = \begin{bmatrix} 1/2 & -1/2 & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/2 & 1/2 & 1/\sqrt{2} \end{bmatrix}$$

current axis → post-multiplication

# Composition of Rotations (3D)



$$R_{y,45^\circ} = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix}$$

$$R_{z,45^\circ} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R' = R_{z,45^\circ} R_{y,45^\circ}$$

$$\Rightarrow R' = \begin{bmatrix} 1/2 & -1/\sqrt{2} & 1/2 \\ 1/2 & 1/\sqrt{2} & 1/2 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} \end{bmatrix}$$

fixed axis → pre-multiplication

# Parametrization of Rotations

- Express an arbitrary rotation matrix as a composition of 3 "standard" rotations

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \text{ with } \begin{aligned} r_{11}^2 + r_{21}^2 + r_{31}^2 &= 1 \\ r_{12}^2 + r_{22}^2 + r_{32}^2 &= 1 \\ r_{13}^2 + r_{23}^2 + r_{33}^2 &= 1 \\ r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} &= 0 \\ r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} &= 0 \\ r_{13}r_{12} + r_{23}r_{22} + r_{33}r_{32} &= 0 \end{aligned} \quad 6 \text{ constraints} \rightarrow 3 \text{ independent parameters}$$

- Euler angles (ZYX current)

$$R = R_{z,\phi} R_{y,\theta} R_{z,\psi} \rightarrow \phi, \theta, \psi$$

- Use of atan2
- Gimbal lock: infinitely many solutions for  $\theta = 0$  or  $\theta = \pi$

- Axis/angle

$$R = R_{k,\theta} \rightarrow k, \theta$$

- Roll( $\phi$ )/Pitch( $\theta$ )/Yaw( $\psi$ ) (XYZ fixed)

$$R = R_{z,\phi} R_{y,\theta} R_{x,\psi} \rightarrow \psi, \theta, \phi$$

# Parametrization of Rotations

- Euler angles (ZYX current)
  - Use of Atan2
  - Gimbal lock: infinitely many solutions for  $\theta = 0$  or  $\theta = \pi$

$$R = R_{z,\phi} R_{y,\theta} R_{z,\psi} \rightarrow \phi, \theta, \psi$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

$$\theta = \text{Atan2}(c_\theta, s_\theta)$$

$$\theta = \text{Atan2}\left(r_{33}, \sqrt{1 - r_{33}^2}\right) \quad \text{or} \quad \theta = \text{Atan2}\left(r_{33}, -\sqrt{1 - r_{33}^2}\right)$$

$$\phi = \text{Atan2}(c_\phi, s_\phi)$$

$$\phi = \text{Atan2}(r_{13}, r_{23})$$

$$\phi = \text{Atan2}(-r_{13}, -r_{23})$$

$$\psi = \text{Atan2}(c_\psi, s_\psi)$$

$$\psi = \text{Atan2}(-r_{31}, r_{32})$$

$$\psi = \text{Atan2}(r_{31}, -r_{32})$$

In the textbook:

$$\text{Atan2}(x, y) = \text{Atan}(y/x)$$

In the Matlab:

[atan2](#)

Four-quadrant inverse tangent

[Syntax](#)

P = atan2(Y,X)

# Homogeneous Transformations

# Homogeneous Transformation Matrix

- Coordinate transformation from  $\{1\}$  to  $\{0\}$

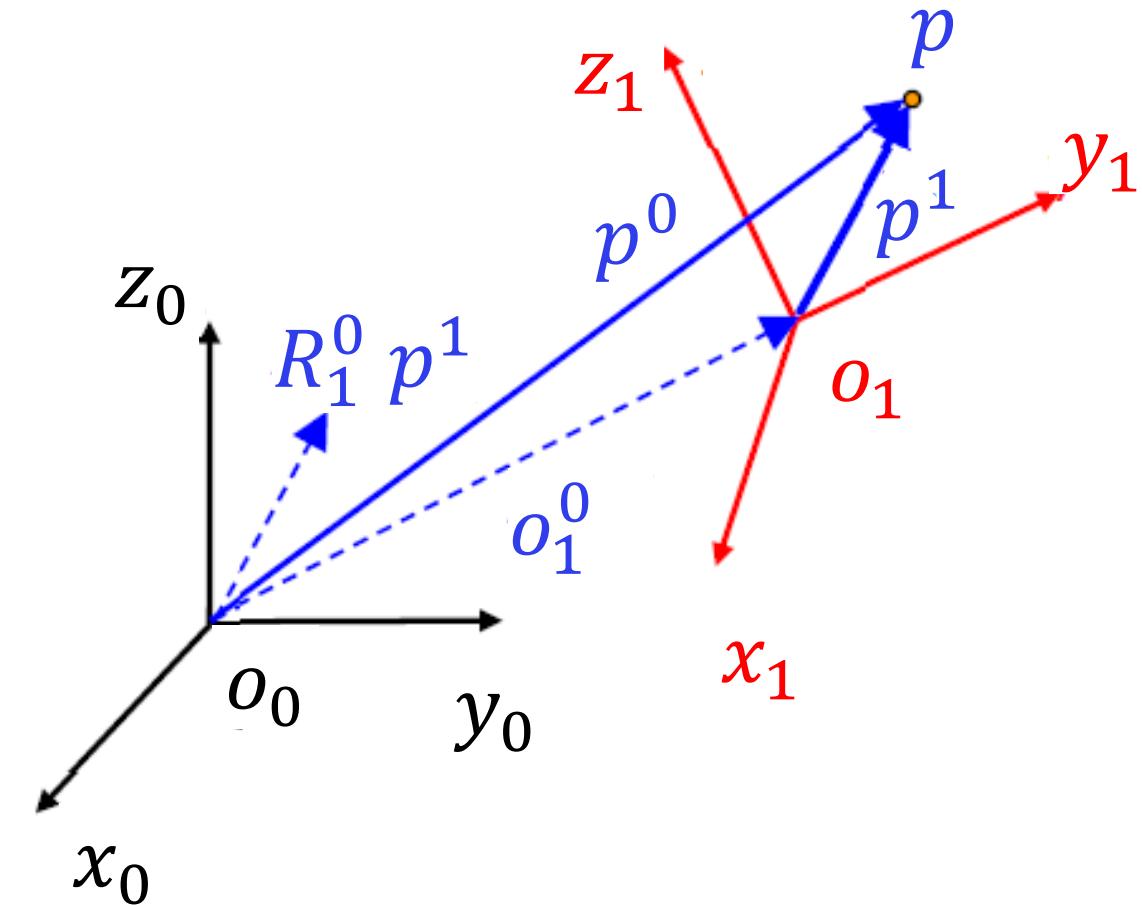
$$p^0 = R_1^0 p^1 + o_1^0$$

$$\begin{bmatrix} p^0 \\ 1 \end{bmatrix} = \begin{bmatrix} R_1^0 & o_1^0 \\ 0_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} p^1 \\ 1 \end{bmatrix}$$

- Homogeneous transformation matrix

$$H_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

rotation matrix  
position vector



# Homogeneous Transformation Matrix

- Special cases

- Translation

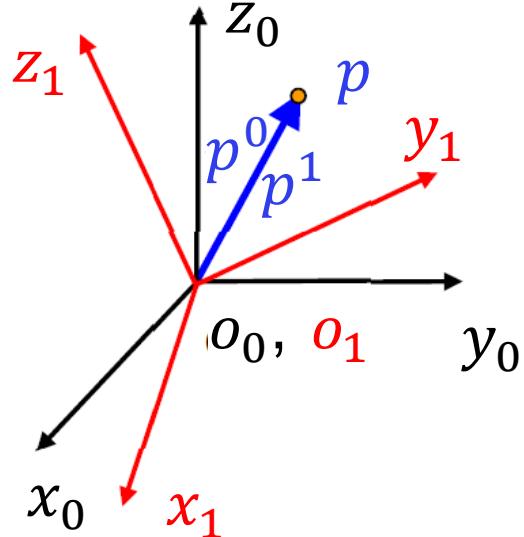
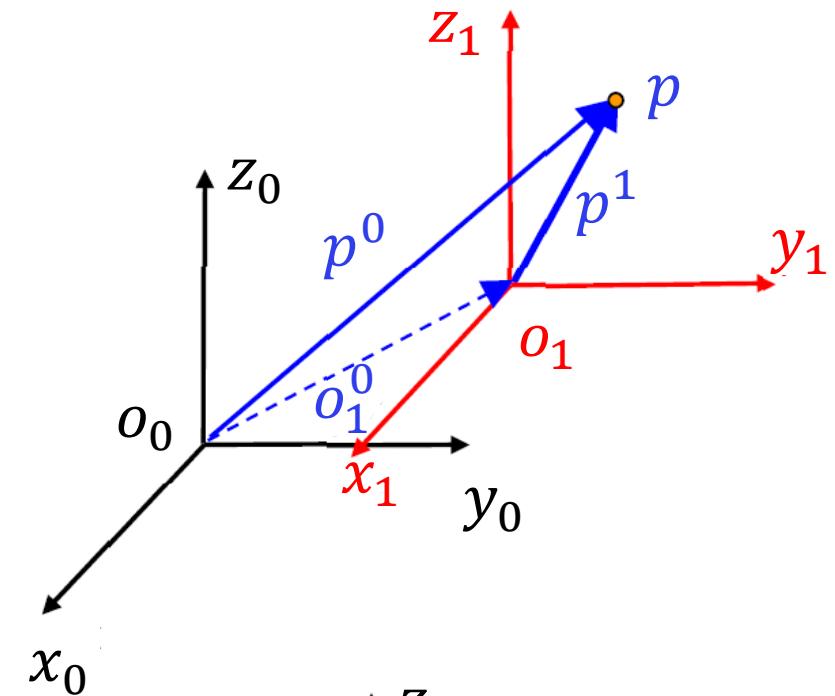
$$H_1^0 = \begin{bmatrix} I_{3 \times 3} & o_1^0 \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

- Rotation

$$H_1^0 = \begin{bmatrix} R_1^0 & 0_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

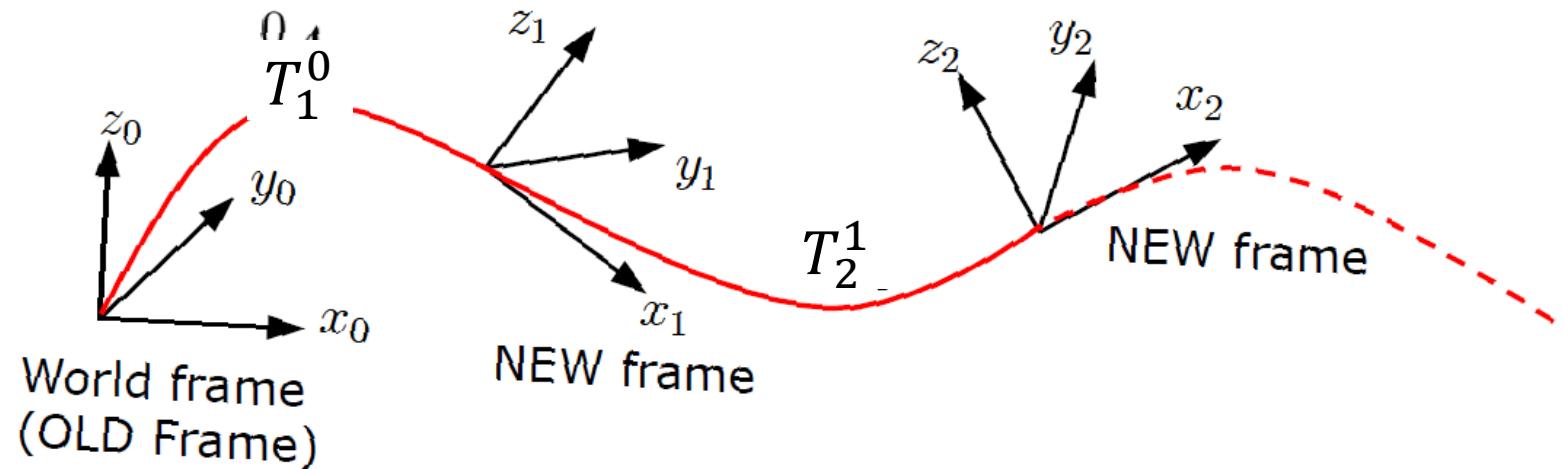
- General case

$$H_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 & o_1^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Composition of Transformation Matrices

- Composite Homogeneous Transformation Matrix
  - Rotation/Translation wrt. original/fixed/global/world frame → **Pre-Multiplication**
  - Rotation/Translation wrt. current frame → **Post-Multiplication**



# Basic Homogeneous Transformation Matrices

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{z,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Exercises

## Problem 1

What is the rotation matrix for a rotation of  $30^\circ$  about the world  $z$ -axis, followed by a rotation of  $60^\circ$  about the world  $x$ -axis, followed by a rotation of  $90^\circ$  about the world  $y$ -axis?

## Problem 2

What is the rotation matrix for a rotation  $\phi$  about the world  $x$ -axis, followed by a rotation  $\psi$  about the current  $z$ -axis, followed by a rotation  $\theta$  about the world  $y$ -axis?

## Problem 3

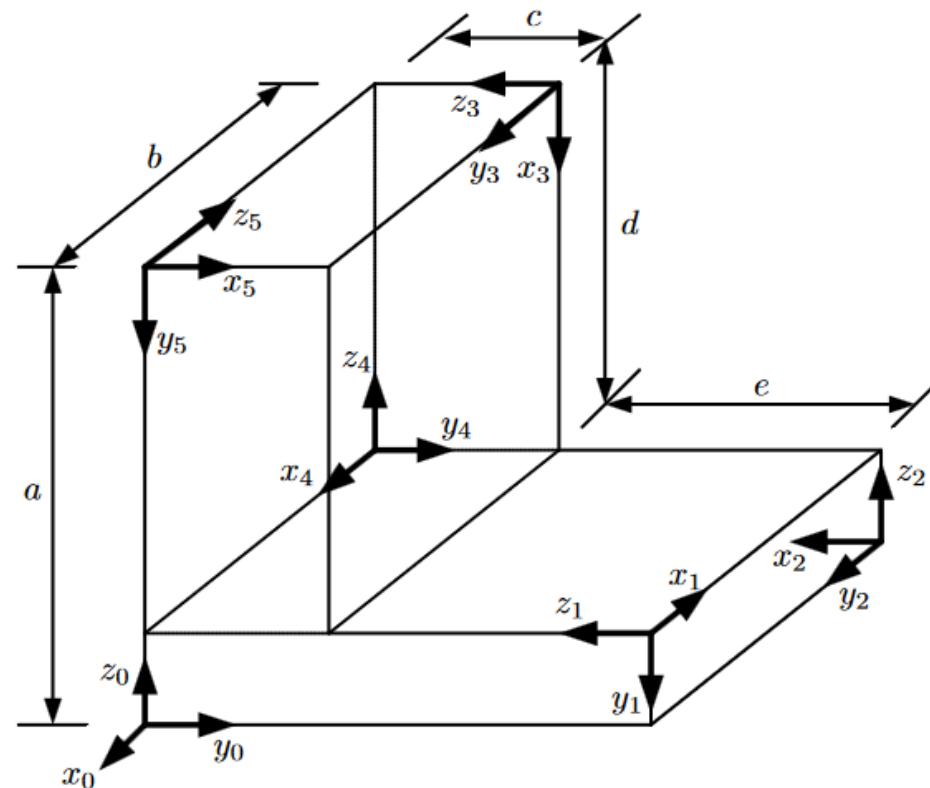
Find another sequence of rotations that is different from Prob. 2, but which results in the same rotation matrix.

## Problem 4

Determine a homogeneous transformation matrix  $H$  that represents a rotation with an angle  $\alpha$  about the world  $x$ -axis, followed by a translation with a length  $b$  along the world  $z$ -axis, followed by a rotation  $\phi$  about the current  $y$ -axis.

## Problem 5

For the figure shown below, find the  $4 \times 4$  homogeneous transformation matrices  $A_1^0, A_2^1, A_3^2, A_4^3$ , and  $A_5^4$ , as well as  $A_5^0$ .

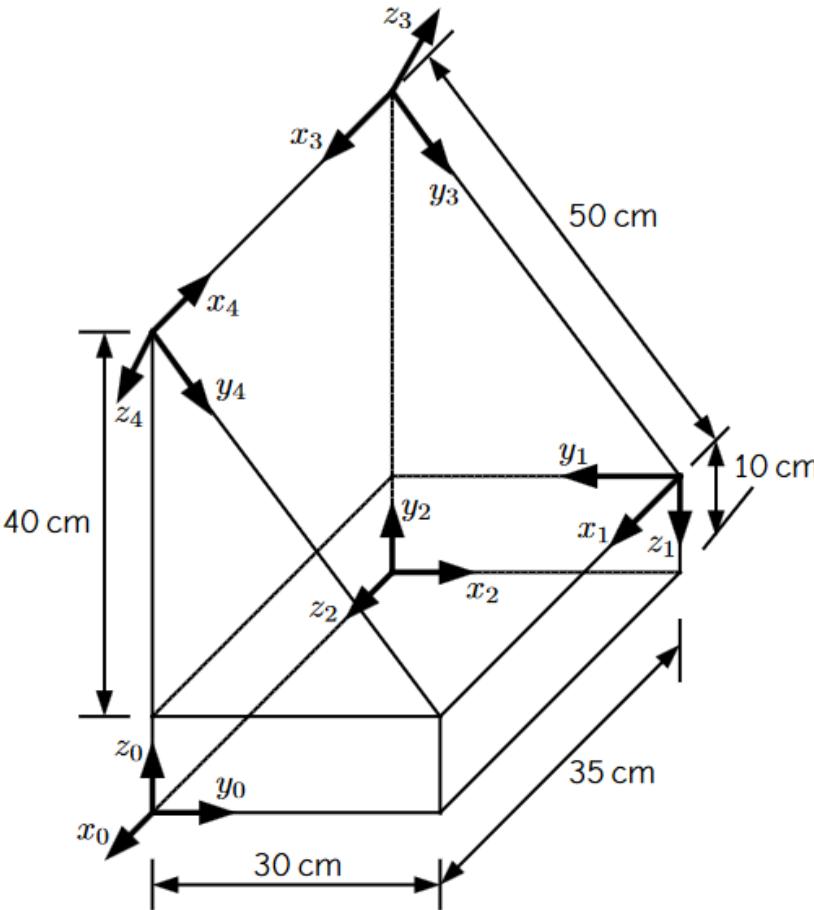


**Hint:** you can find the answers directly by observation, based on the geometric interpretation of each column in the homogeneous transformation matrix.

# Exercises

## Problem 6

For the figure shown below, find the  $4 \times 4$  homogeneous transformation matrices  $A_1^0$ ,  $A_2^1$ ,  $A_3^2$ , and  $A_4^3$ , as well as  $A_4^0$ .



**Hint:** you can find the answers directly by observation, based on the geometric interpretation of each column in the homogeneous transformation matrix.

It seems that last week it was difficult for many of you to fully grasp pre- and post-multiplication of transformation matrices.

I have prepared a 2D example which is as simple as possible to help you develop an intuitive understanding of this subject.

Think of the following transformation

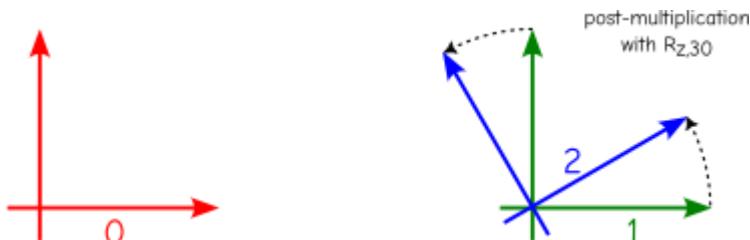
$$T_{x,100mm} R_{z,30^\circ}$$

which transforms frame 0 to frame 2 in the illustration below

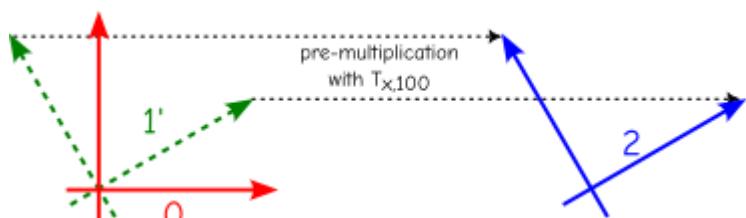


As we have learned in lecture 2, there are 2 possible ways for interpreting the product of the two homogeneous transformation matrices that leads to this result.

1. Translation along the x-axis by 100 mm first, and then rotation about the **current** z-axis by  $30^\circ$



2. Rotation about the z-axis by  $30^\circ$  first, and then translation about the **global** x-axis by 100 mm.



Can you imagine what the outcome would be if in case #1 we rotated about the global z-axis instead?

Can you imagine what the outcome would be if in case #2 we translated about the current x-axis instead?

Robotics – 34753

# Robot Kinematics II & Inverse Kinematics

Konstantinos Poulios  
Associate Professor

Department of Civil and Mechanical Engineering  
DTU Lyngby, building 404 / room 124

# Robot Kinematics II & Inverse Kinematics – Lecture Overview

## 1. Repetition

## 2. Kinematic Chains

- Denavit-Hartenberg Convention (D-H)
- Coordinate Frame Assignment
- End Effector Frame
- Examples

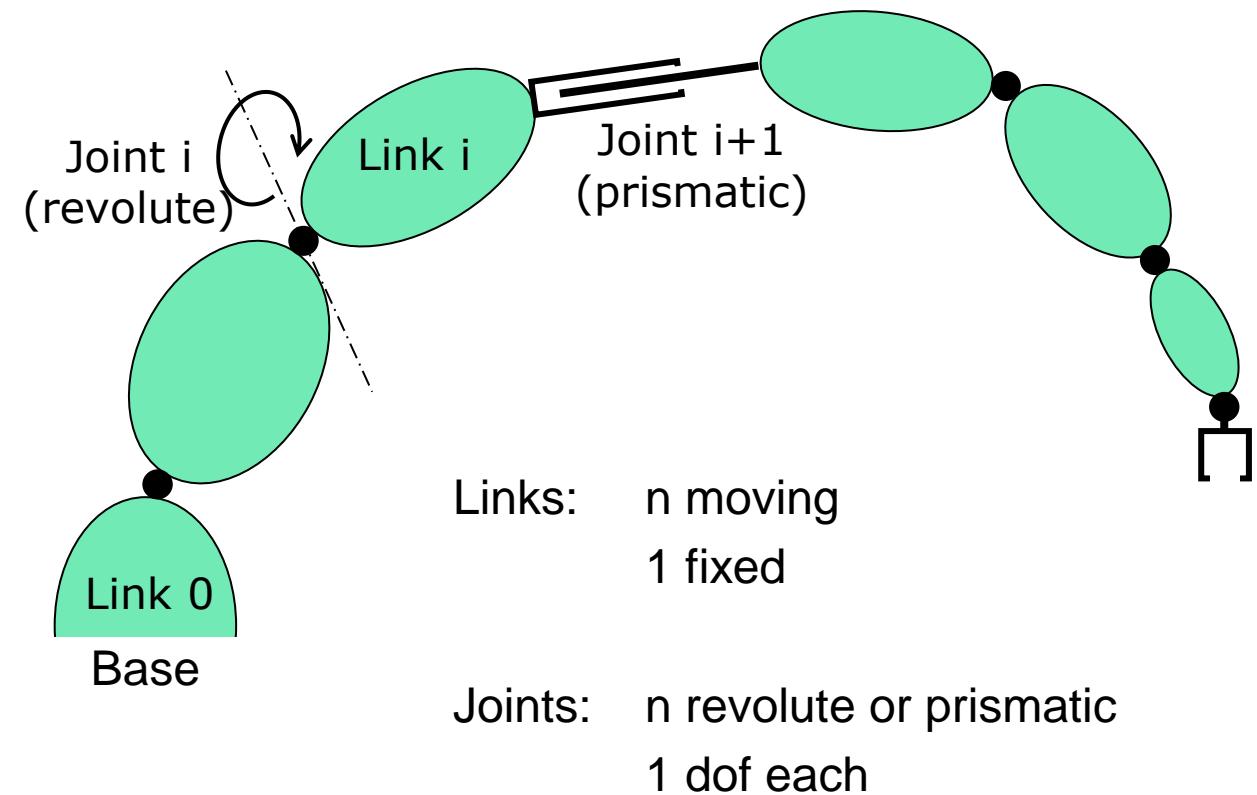
## 3. Inverse Kinematics

- Inverse Position
- Inverse Orientation

# Repetition

# Repetition

- Serial link manipulator (a.k.a. robot arm, industrial robot)
  - An open chain of rigid bodies (links) connected by joints (revolute or prismatic)
- Manipulator specification
  - Degrees of freedom:  $n$
  - Joint space
  - Work space
  - Redundancy:  $n > m$



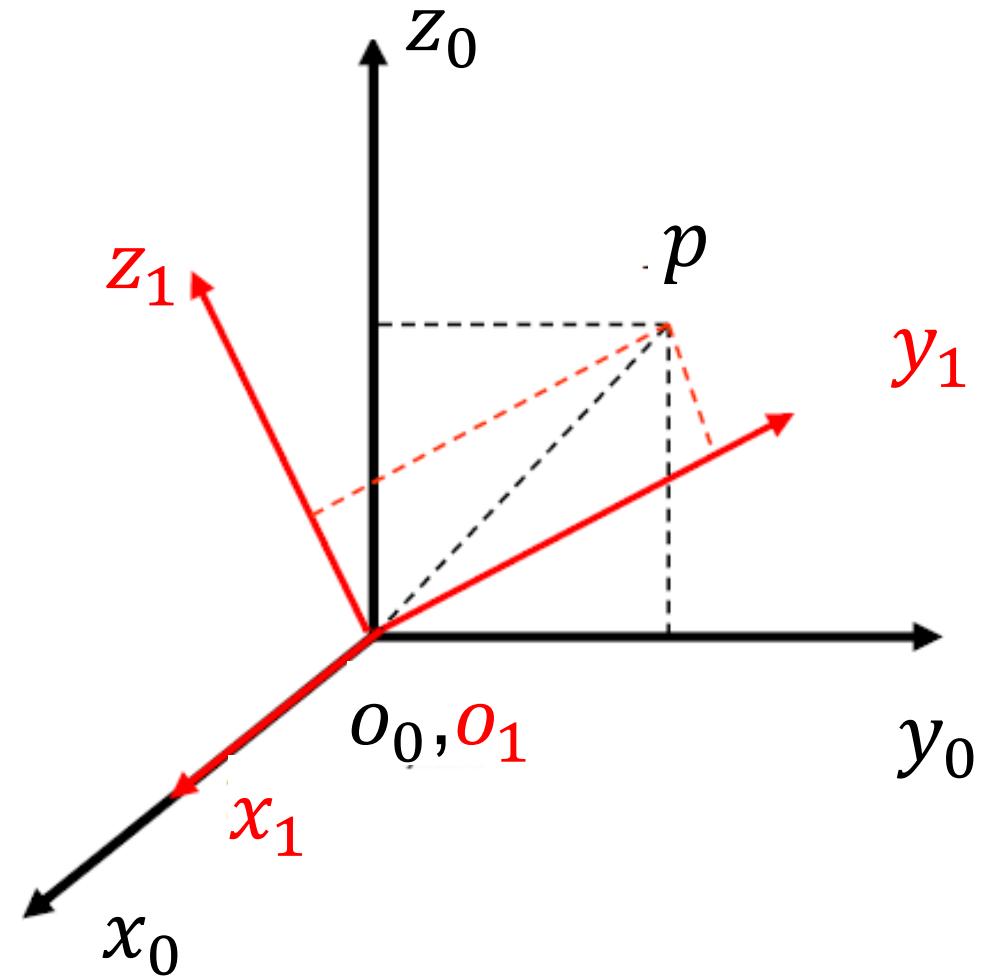
# Repetition

- Basic rotation matrix between frame  $o_0x_0y_0z_0$  and frame  $o_1x_1y_1z_1$

$$p^0 = \begin{bmatrix} x_0 \cdot x_1 & x_0 \cdot y_1 & x_0 \cdot z_1 \\ y_0 \cdot x_1 & y \cdot y_1 & y_0 \cdot z_1 \\ z_0 \cdot x_1 & z_0 \cdot y_1 & z_0 \cdot z_1 \end{bmatrix} p^1$$

$$p^0 = R_1^0 p^1$$

$$p^1 = (R_1^0)^{-1} p^0$$



# Repetition

- Basic rotation matrices

- About x-axis with  $\theta$

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & -s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix}$$

- About y-axis with  $\theta$

$$R_{y,\theta} = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}$$

- About z-axis with  $\theta$

$$R_{z,\theta} = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Repetition

- Coordinate transformation from  $\{1\}$  to  $\{0\}$

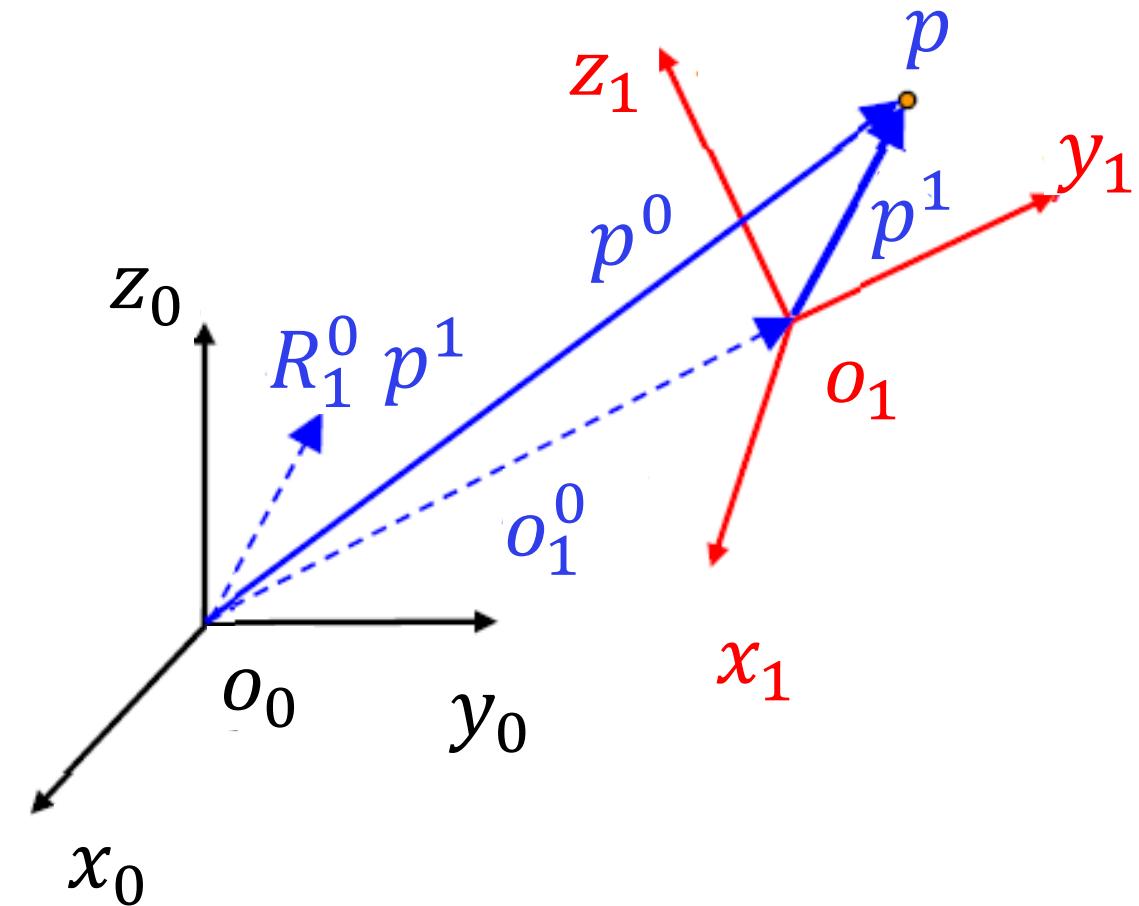
$$p^0 = R_1^0 p^1 + o_1^0$$

$$\begin{bmatrix} p^0 \\ 1 \end{bmatrix} = \begin{bmatrix} R_1^0 & o_1^0 \\ 0_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} p^1 \\ 1 \end{bmatrix}$$

- Homogeneous transformation matrix

$$T_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

rotation matrix  
position vector



# Repetition

- Special cases

- Translation

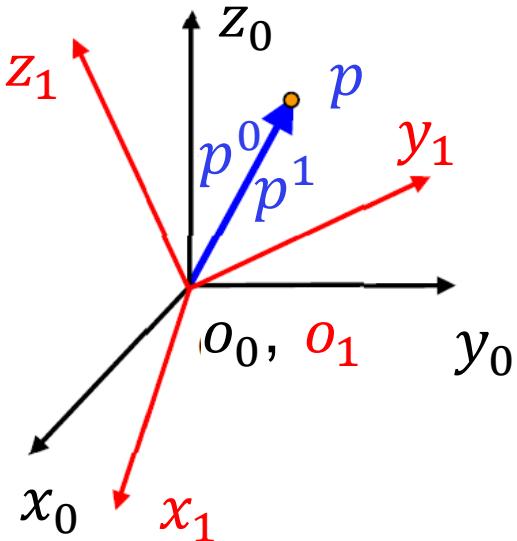
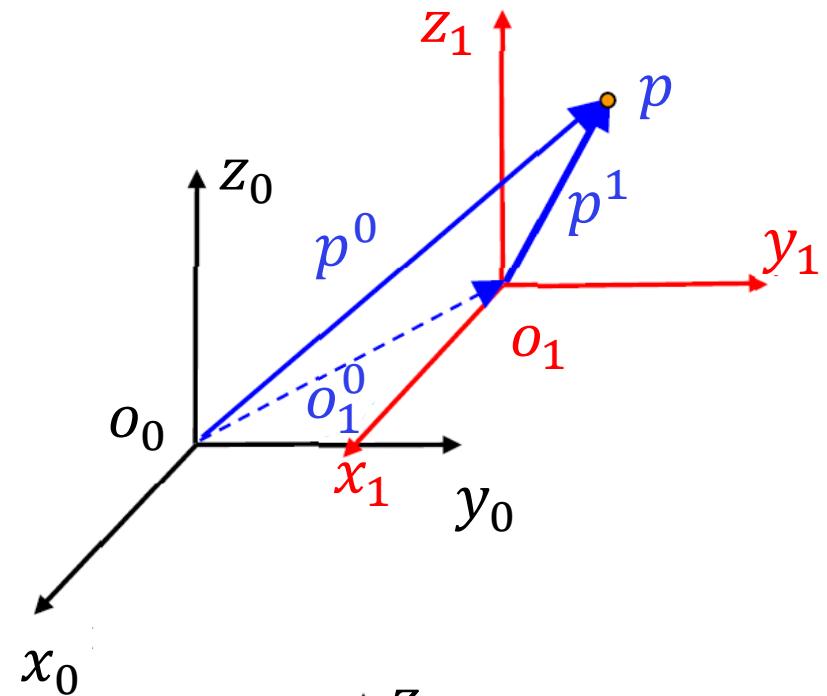
$$H_1^0 = \begin{bmatrix} I_{3 \times 3} & o_1^0 \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

- Rotation

$$H_1^0 = \begin{bmatrix} R_1^0 & 0_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

- General case

$$H_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 & o_1^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Repetition

- Basic homogeneous transformation matrices

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{z,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Repetition

- Composition of Transformations
  - Rotation/Translation wrt. original/fixed/global/world frame → **Pre-Multiplication**
  - Rotation/Translation wrt. current frame → **Post-Multiplication**
- Example with rotations
  1.  $\theta$ -rotation about **current x**
  2.  $\phi$ -rotation about **current z**
  3.  $\alpha$ -rotation about **fixed z**
  4.  $\beta$ -rotation about **current y**
  5.  $\delta$ -rotation about **fixed x**

$$R_5^0 = R_{x,\delta} R_{z,\alpha} R_{x,\theta} R_{z,\phi} R_{y,\beta}$$

5    3    1    2    4

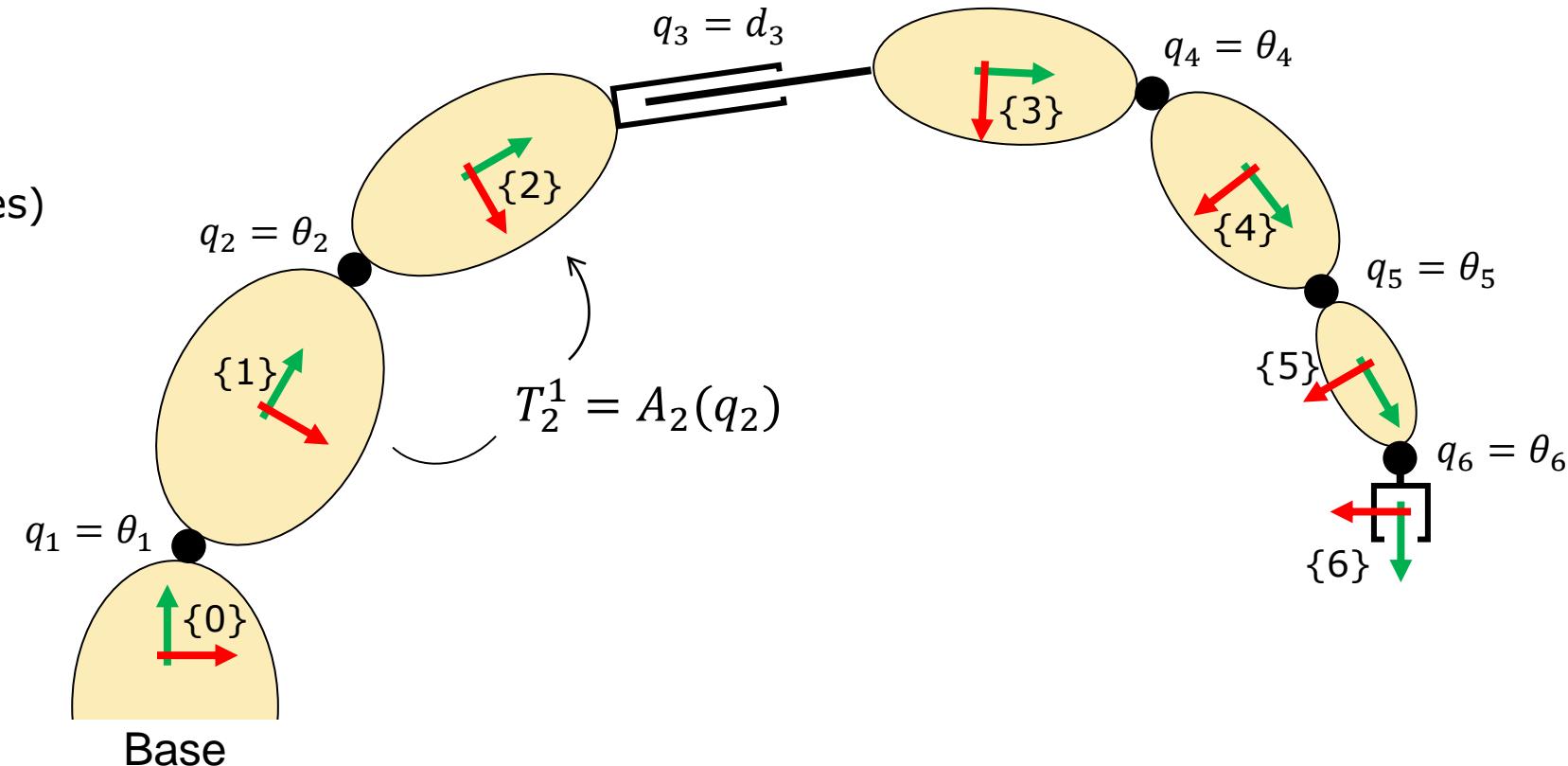
# Kinematic Chains

# Kinematic Chains

$n = 6$  joints/dofs

$n + 1 = 7$  links (bodies)

$n + 1 = 7$  frames



Note:  
 $A_i = T_i^{i-1}$

$$T_6^0 = A_1(q_1) A_2(q_2) A_3(q_3) A_4(q_4) A_5(q_5) A_6(q_6)$$

... but each homogeneous transformation matrix  $A_j(q_j)$  is a rather complex function

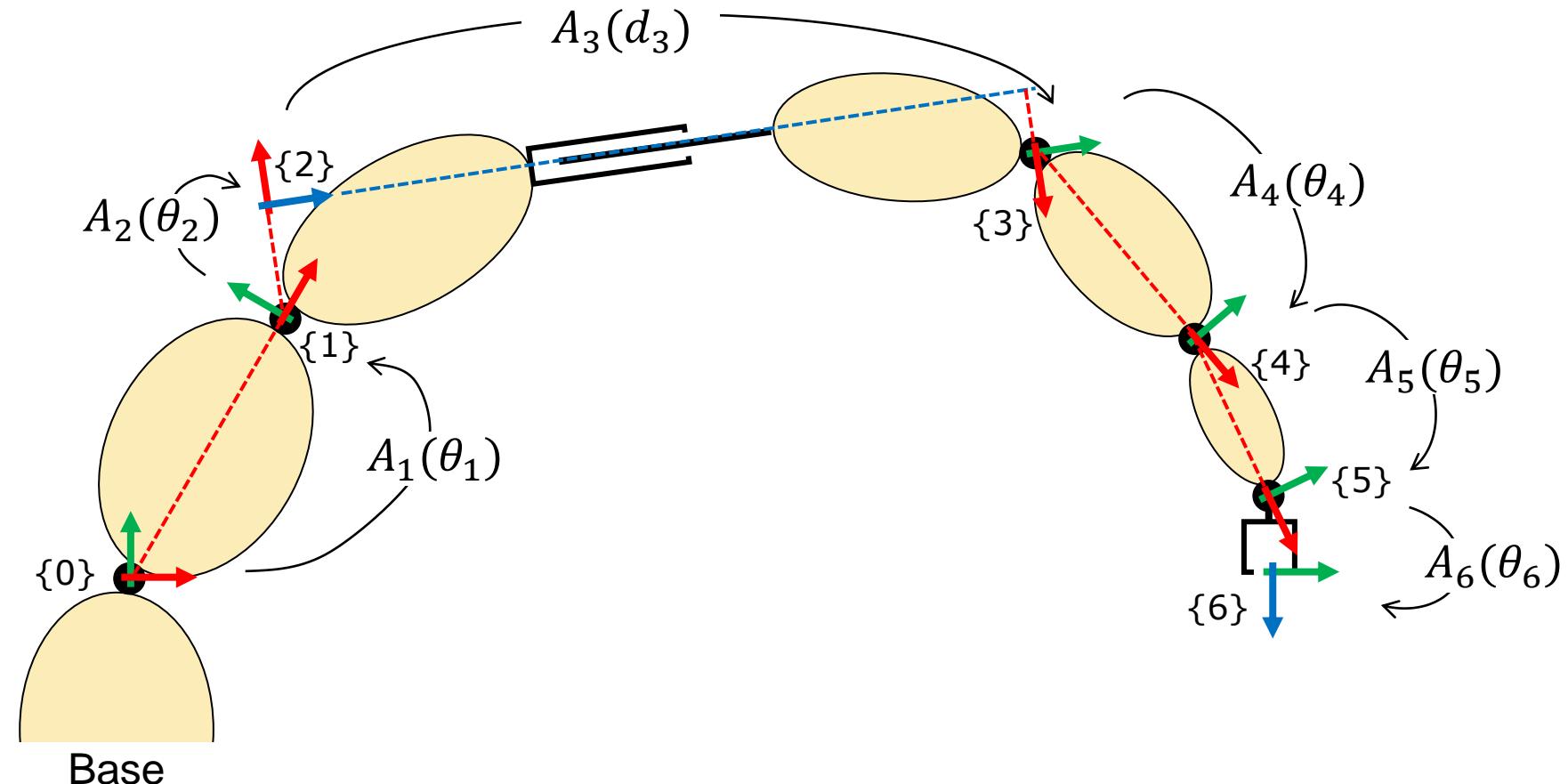
# Kinematic Chains

$n = 6$  joints/dofs

$n + 1 = 7$  links (bodies)

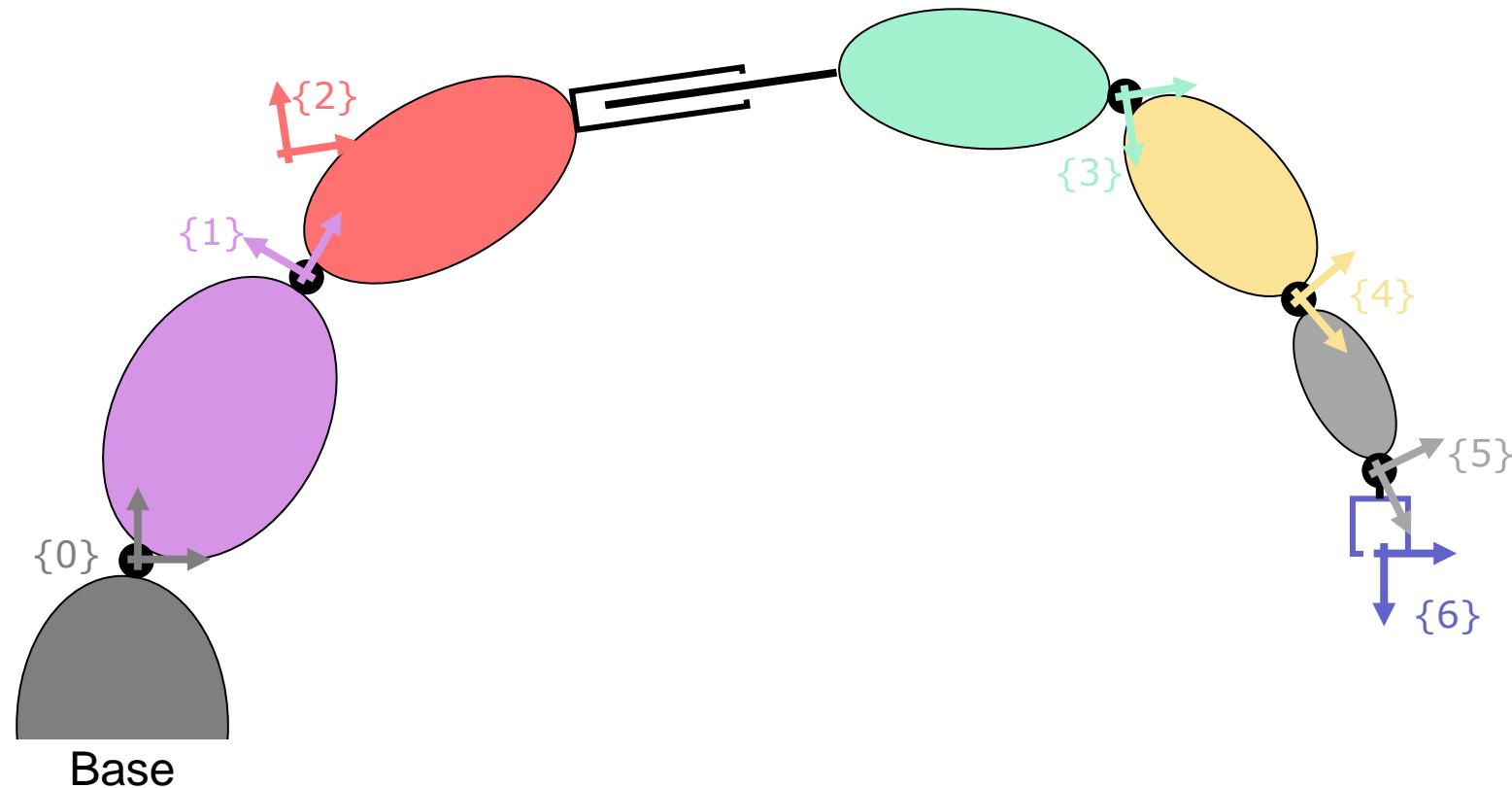
$n + 1 = 6 + 1$  frames

Last frame is special



$$T_6^0 = A_1(\theta_1) \ A_2(\theta_2) \ A_3(d_3) \ A_4(\theta_4) \ A_5(\theta_5) \ A_6(\theta_6)$$

# Kinematic Chains



# Denavit–Hartenberg Convention

- The coordinate frame  $\{i\}$  is determined from the frame  $\{i-1\}$  through a homogeneous transformation in the form:

$$A_i = \text{Rot}_{z,\theta_i} \text{Trans}_{z,d_i} \text{Trans}_{x,a_i} \text{Rot}_{x,\alpha_i}$$

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Where:

$\theta_i$	: joint angle
$d_i$	: link offset
$a_i$	: link length
$\alpha_i$	: link twist

Only 4 parameters instead of 6.  
Why?

# Denavit–Hartenberg Convention

- The coordinate frame  $\{i\}$  is determined from the frame  $\{i-1\}$  through a homogeneous transformation in the form:

$$A_i = \text{Rot}_{z,\theta_i} \text{Trans}_{z,d_i} \text{Trans}_{x,a_i} \text{Rot}_{x,\alpha_i}$$

- Where:

$\theta_i$	: joint angle
$d_i$	: link offset
$a_i$	: link length
$\alpha_i$	: link twist
- Only 4 parameters instead of 6.  
Why?

- New coordinate frame with z-axis aligned with the **next** joint axis
- Position along and rotation around the axis **not free** to choose anymore

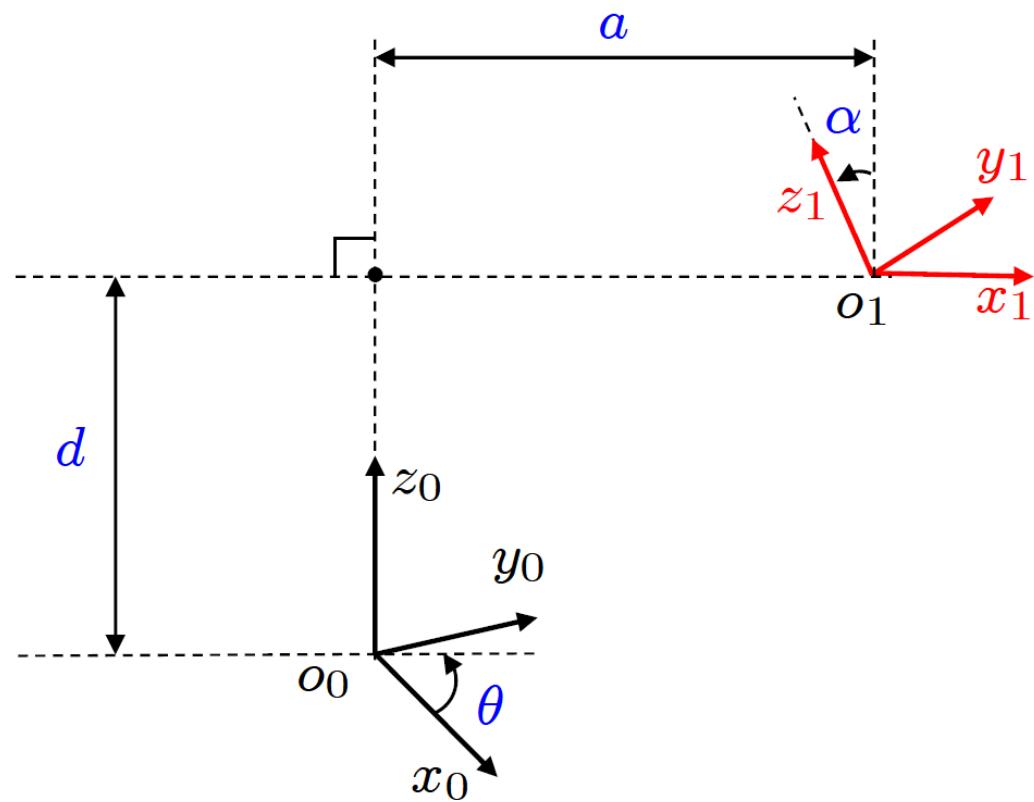
# Denavit–Hartenberg Convention

- Where:

$\theta_i$	: joint angle
$d_i$	: link offset
$a_i$	: link length
$\alpha_i$	: link twist

Only 4 parameters instead of 6  
Why?
- The missing two parameters are due to the two Denavit–Hartenberg conditions:

DH1: The axis  $x_i$  is perpendicular to the axis  $z_{i-1}$   
DH2: The axis  $x_i$  intersects the axis  $z_{i-1}$



# Physical Interpretation of $\theta, d, a, \alpha$

- $\theta_1$  : angle from  $x_0$  to  $x_1$  about  $z_0$
- $d_1$  : distance from  $o_0$  to  $x_1$  (along  $z_0$ )
- $a_1$  : distance from  $z_0$  to  $o_1$  (along  $x_1$ )
- $\alpha_1$  : angle from  $z_0$  to  $z_1$  about  $x_1$

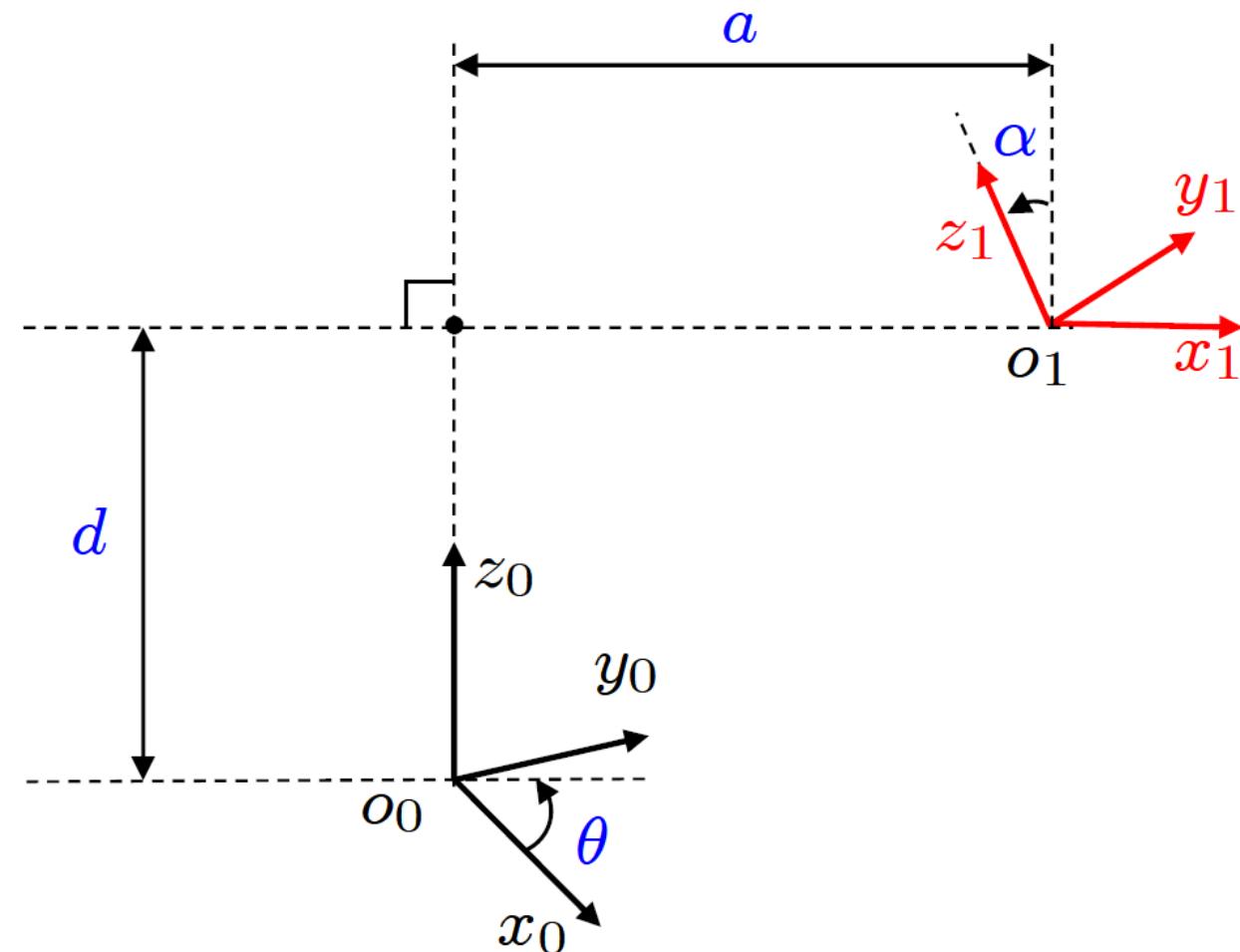
Mnemonic:

$\theta - d - a - \alpha$

about-along - along-about

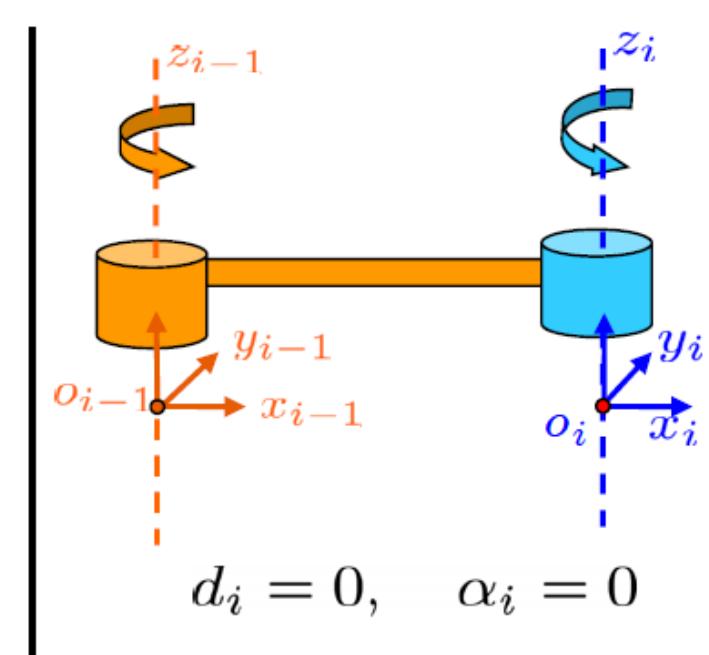
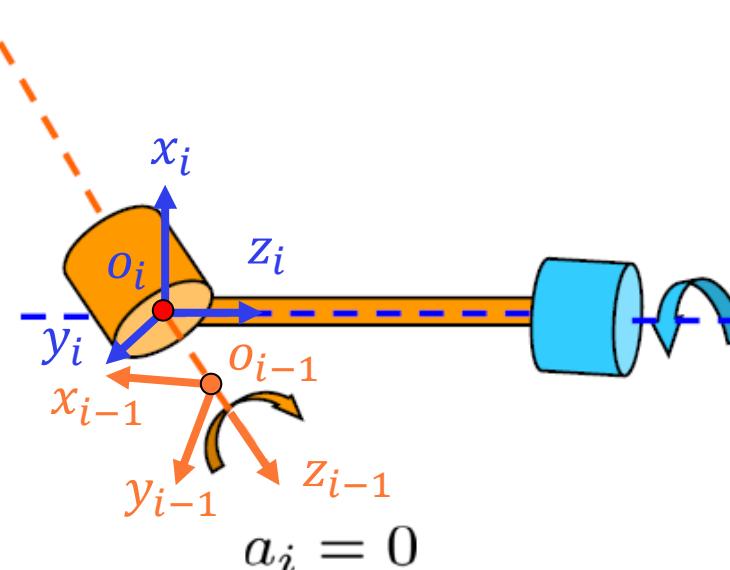
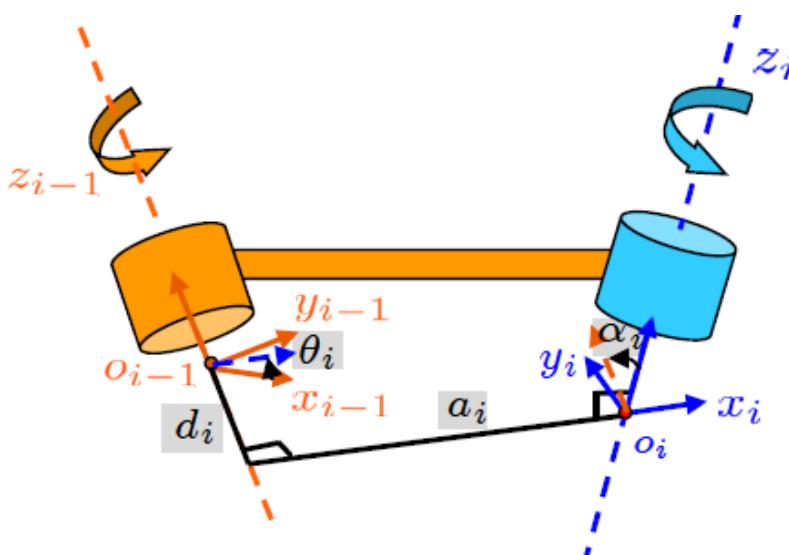
$z_{i-1}$

$x_i$



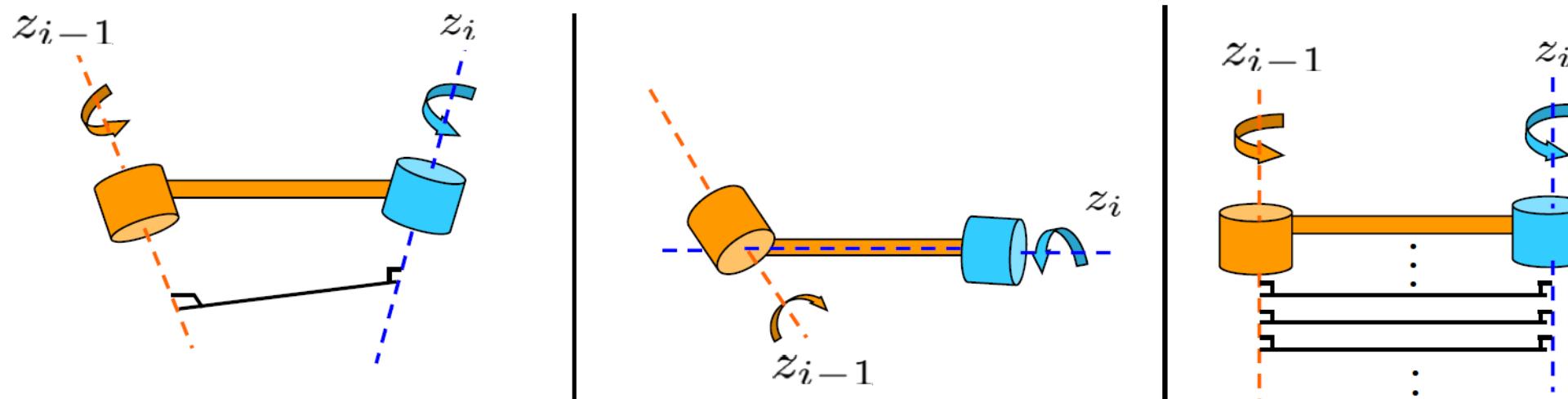
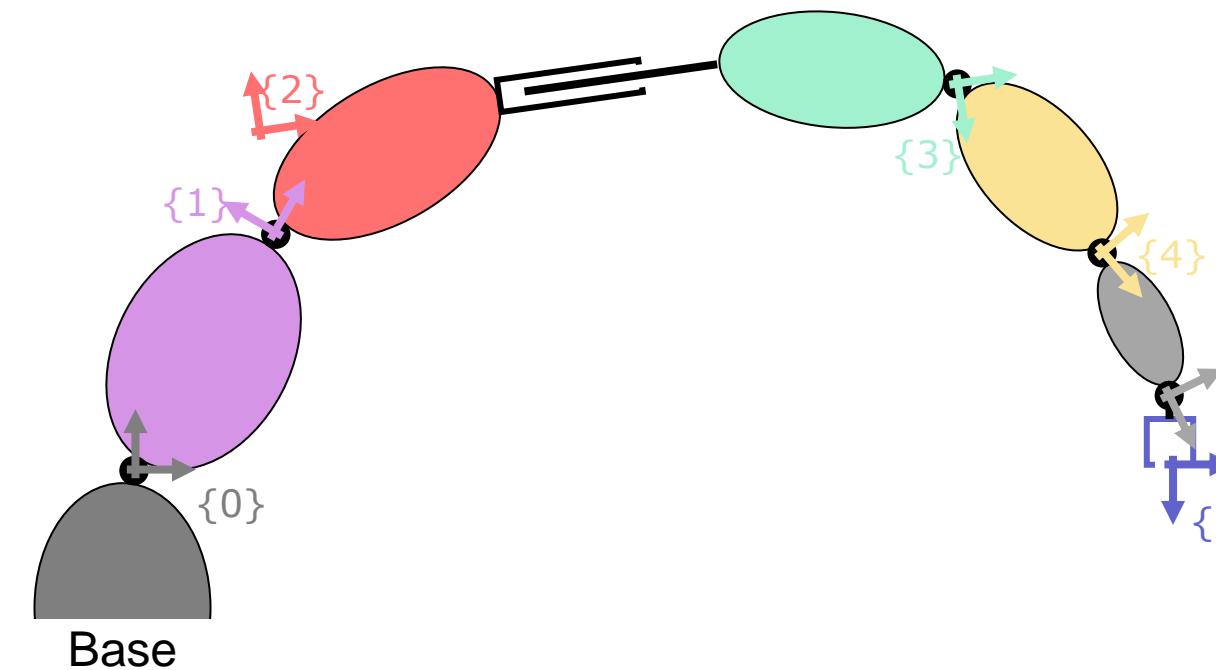
# Physical Interpretation of $\theta, d, a, \alpha$

- |            |   |   |  |
|------------|---|---|--|
| $\theta_1$ | : angle from $x_0$ to $x_1$ about $z_0$       | } | one of both is a variable:<br>$\theta_1$ for revolute, $d_1$ for prismatic |
| $d_1$      | : distance from $o_0$ to $x_1$ (along $z_0$ ) |   | always constant characteristic<br>of the manipulator                       |
| $a_1$      | : distance from $z_0$ to $o_1$ (along $x_1$ ) |   |  |
| $\alpha_1$ | : angle from $z_0$ to $z_1$ about $x_1$       |   |  |



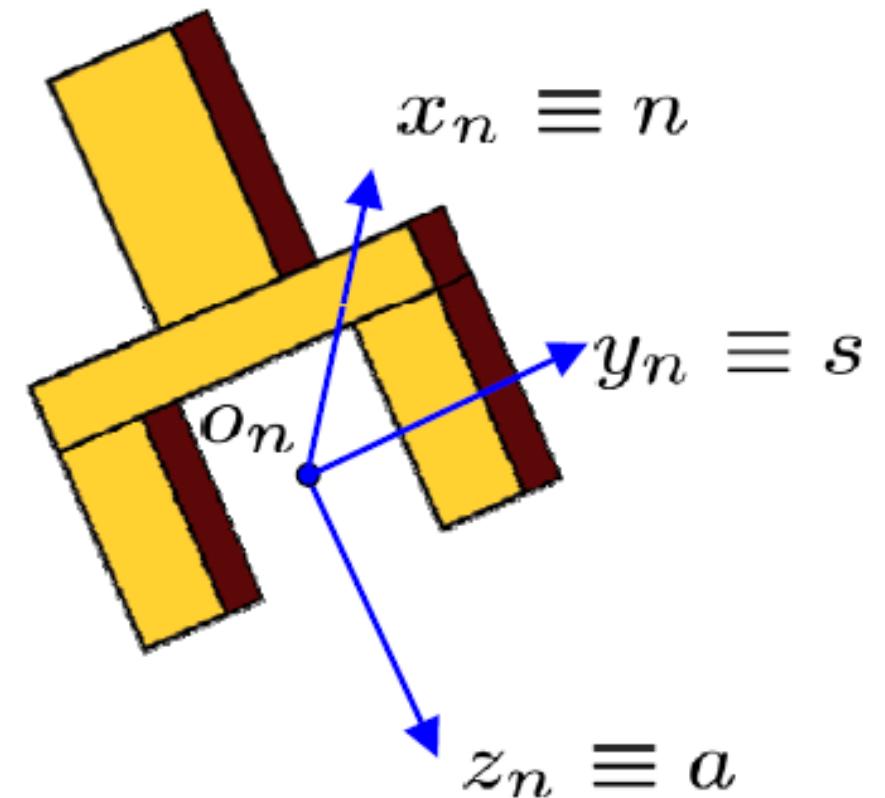
# Assignment of Coordinate Frames

- $z_i$  axis along the  $i + 1$  joint axis
- $x_i$  axis parallel to  $z_i \times z_{i-1}$
- $y_i$  axis parallel to  $z_i \times x_i$
- Origin  $o_i$  along  $z_i$  at the point of shortest distance to  $z_{i-1}$



# End Effector Coordinate Frame

- Origin  $o_n$  in the middle between the fingers
- $z_n$  axis parallel to the fingers, also denoted as  $a$  (approach)
- $y_n$  axis parallel to the closing direction of the fingers, also denoted as  $s$  (sliding)
- $x_n$  axis parallel to  $y_n \times z_n$ , also denoted as  $n$  (normal)



# Examples

# Three-Link Cylindrical Robot

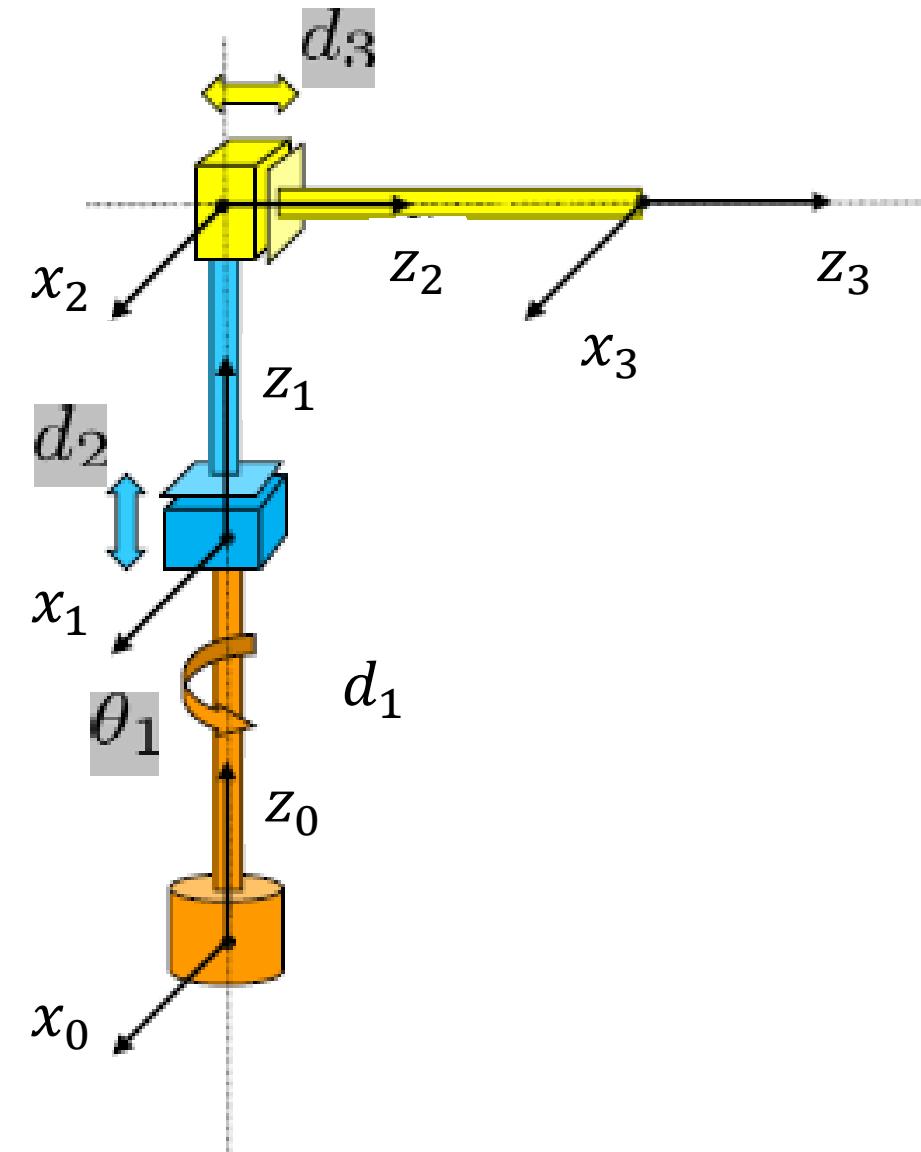
$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Joint i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^*$	$d_1$	0	0
2	0	$d_2^*$	0	-90
3	0	$d_3^*$	0	0



# SCARA Manipulator

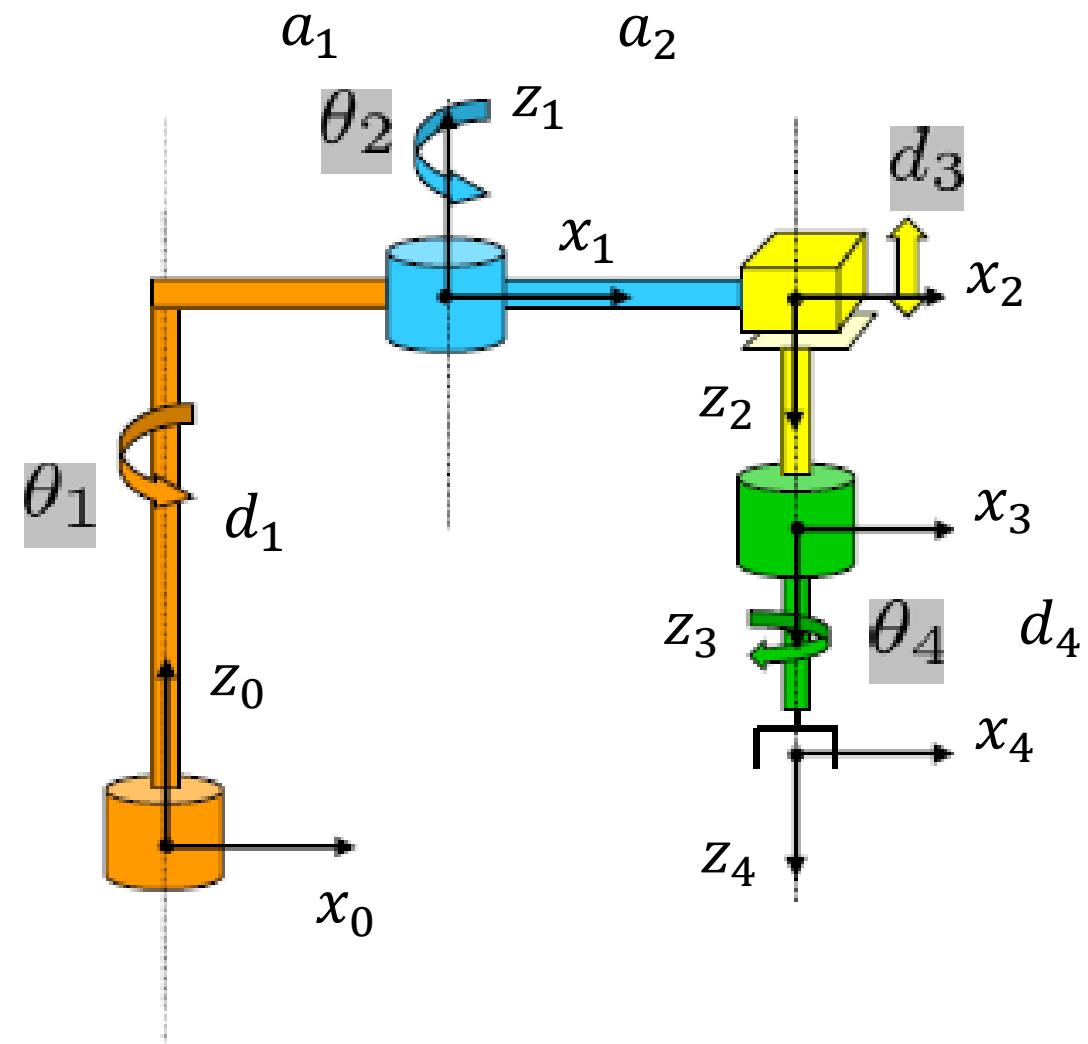
$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

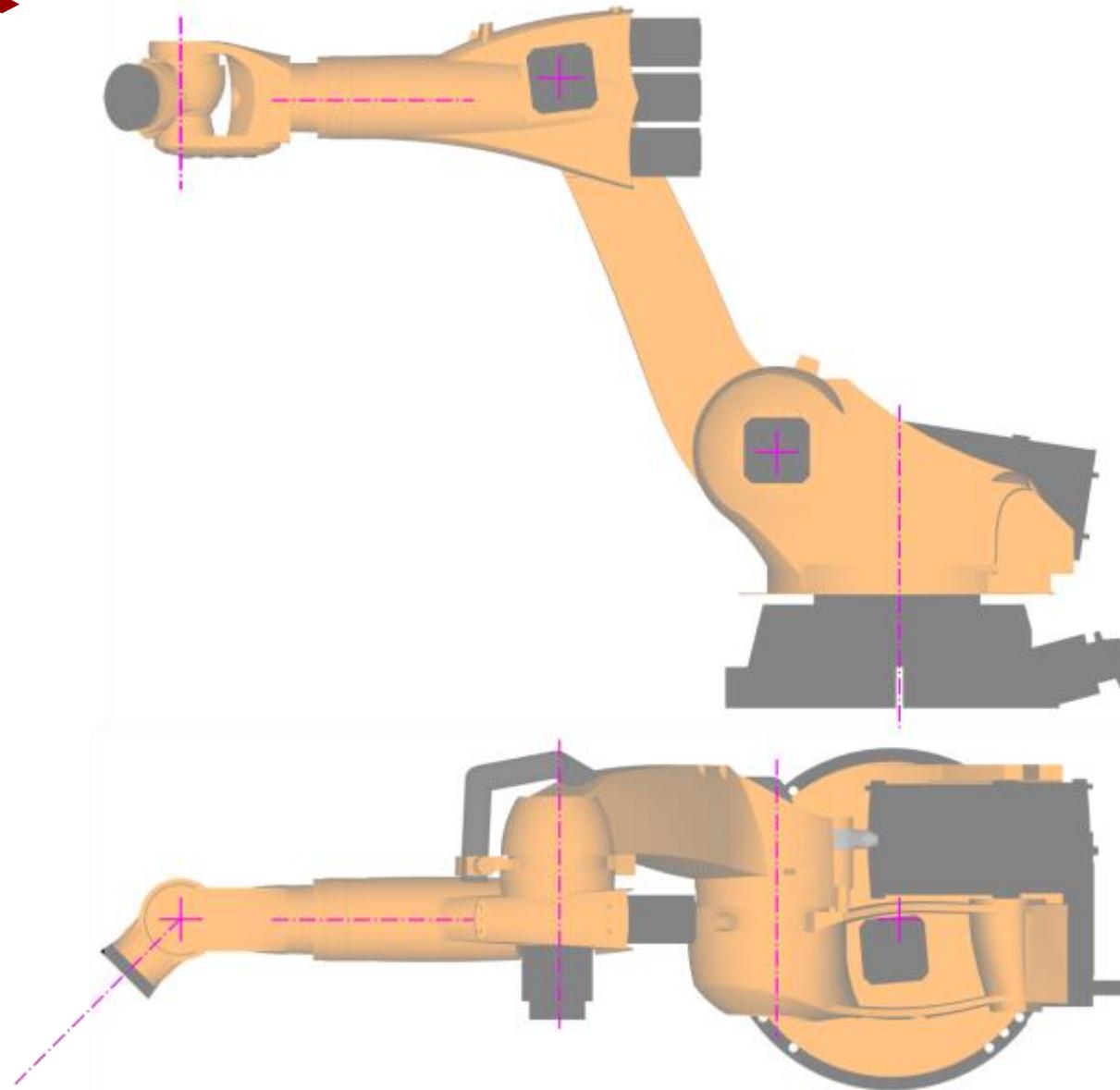
$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Joint i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^*$	$d_1$	$a_1$	0
2	$\theta_2^*$	0	$a_2$	180
3	0	$d_3^*$	0	0
4	$\theta_4^*$	$d_4$	0	0

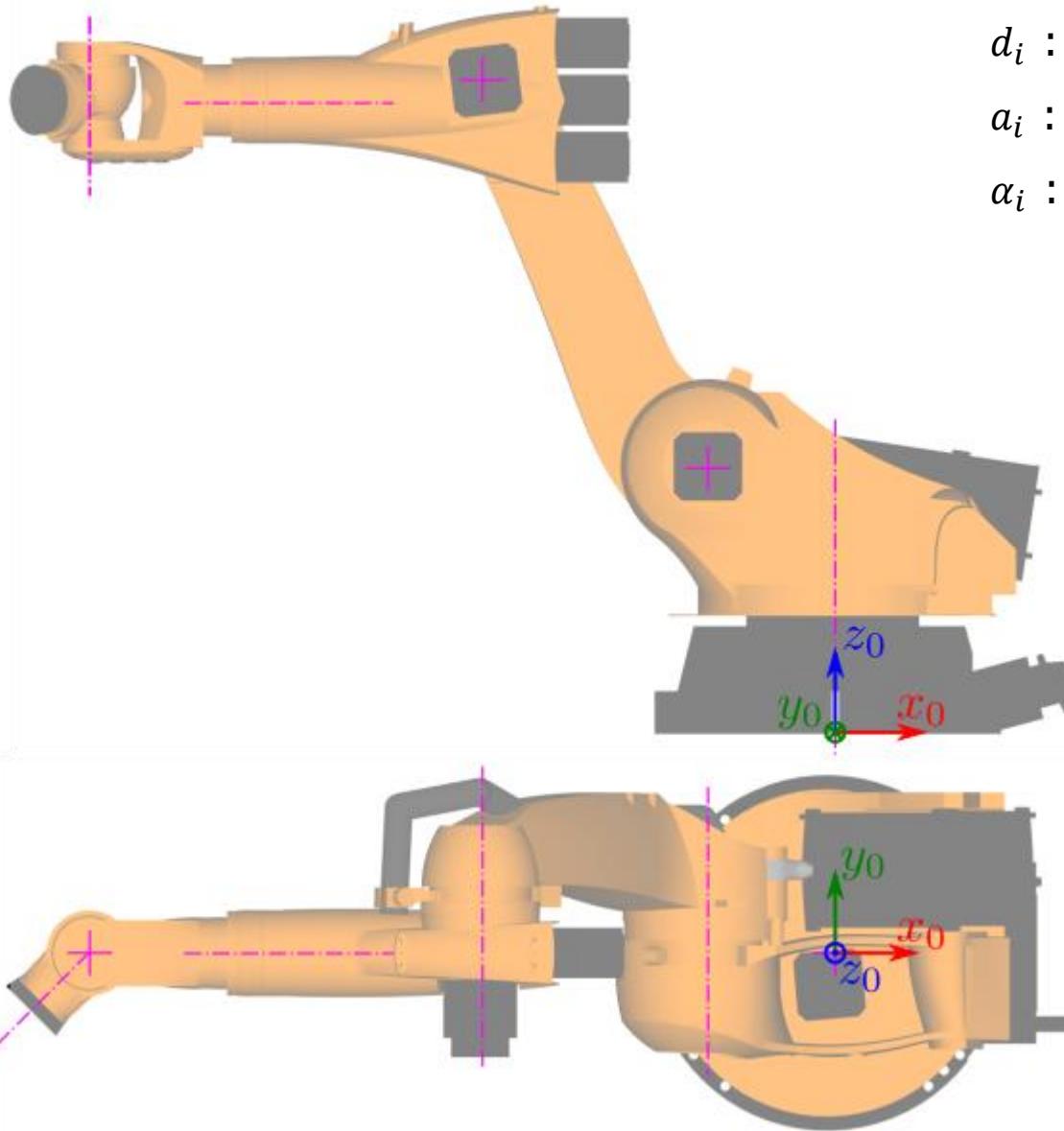


# KUKA KR 210



- Number the joints
- Establish base frame
- Establish joint axes  $Z_i$
- Locate origin  $O_i$
- Establish  $x_i$  and  $y_i$

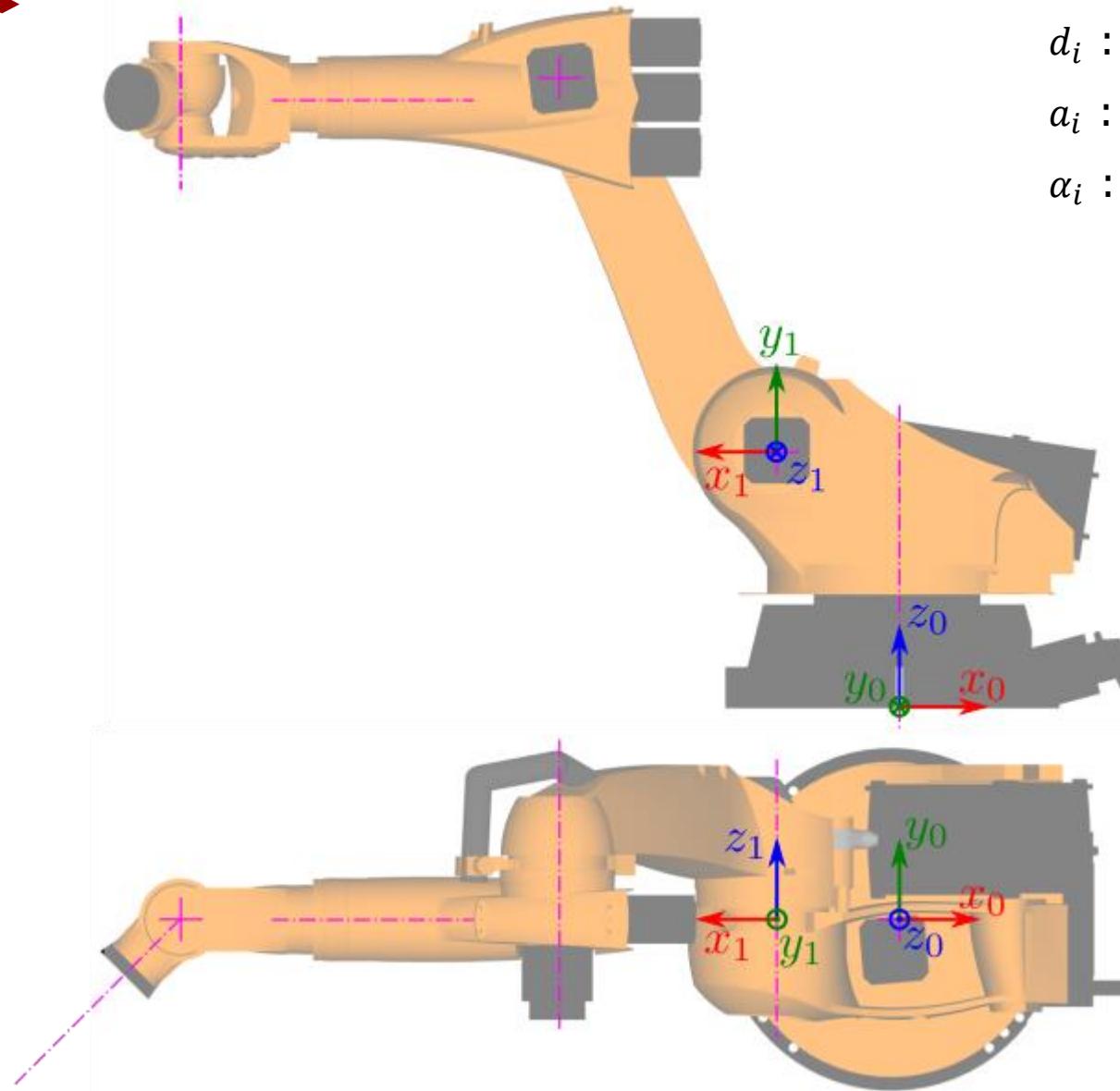
# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$   
 $d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )  
 $a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )  
 $\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1				
2				
3				
4				
5				
6				

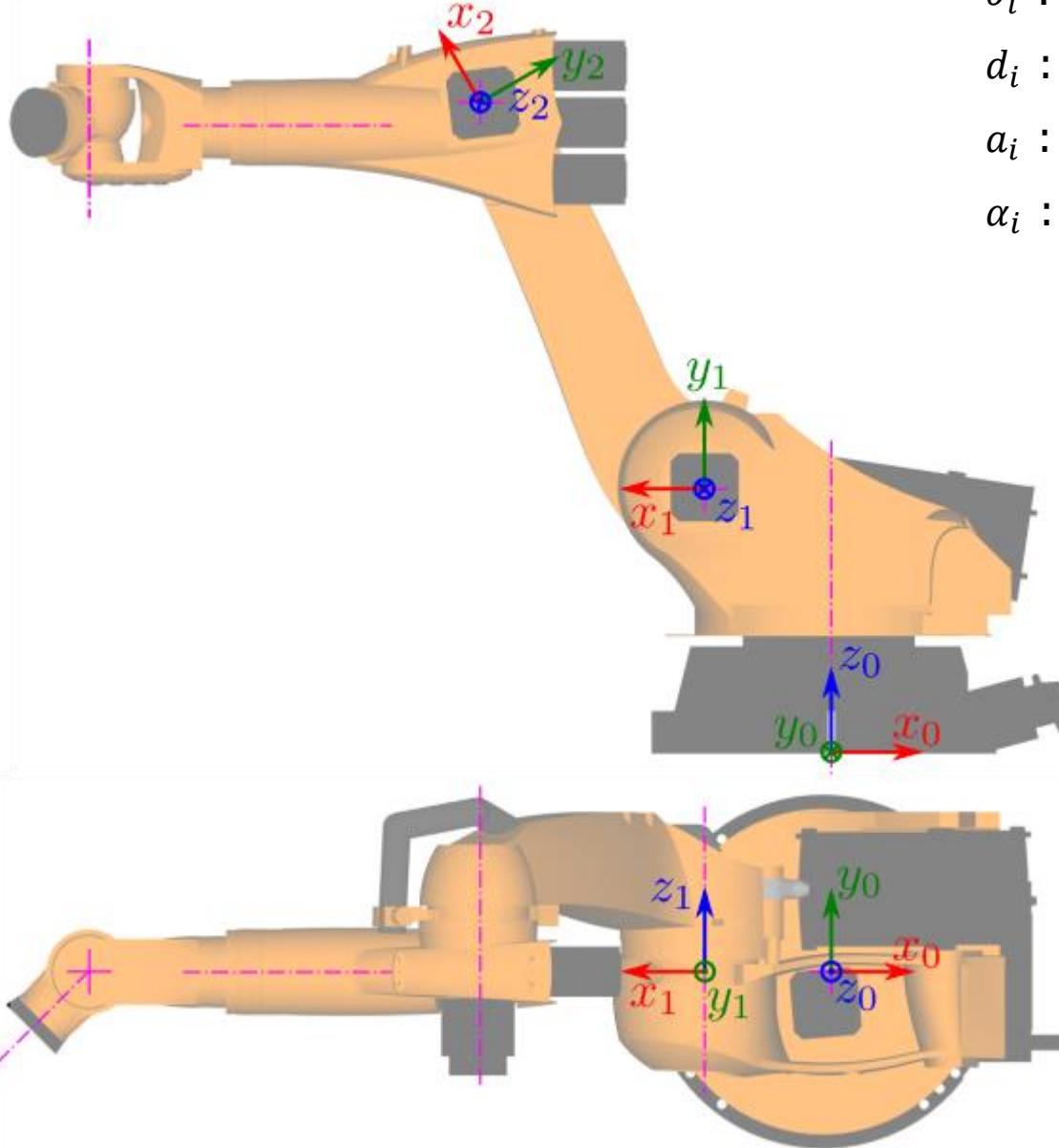
# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$   
 $d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )  
 $a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )  
 $\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^* = 180$	200	100	90
2				
3				
4				
5				
6				

# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

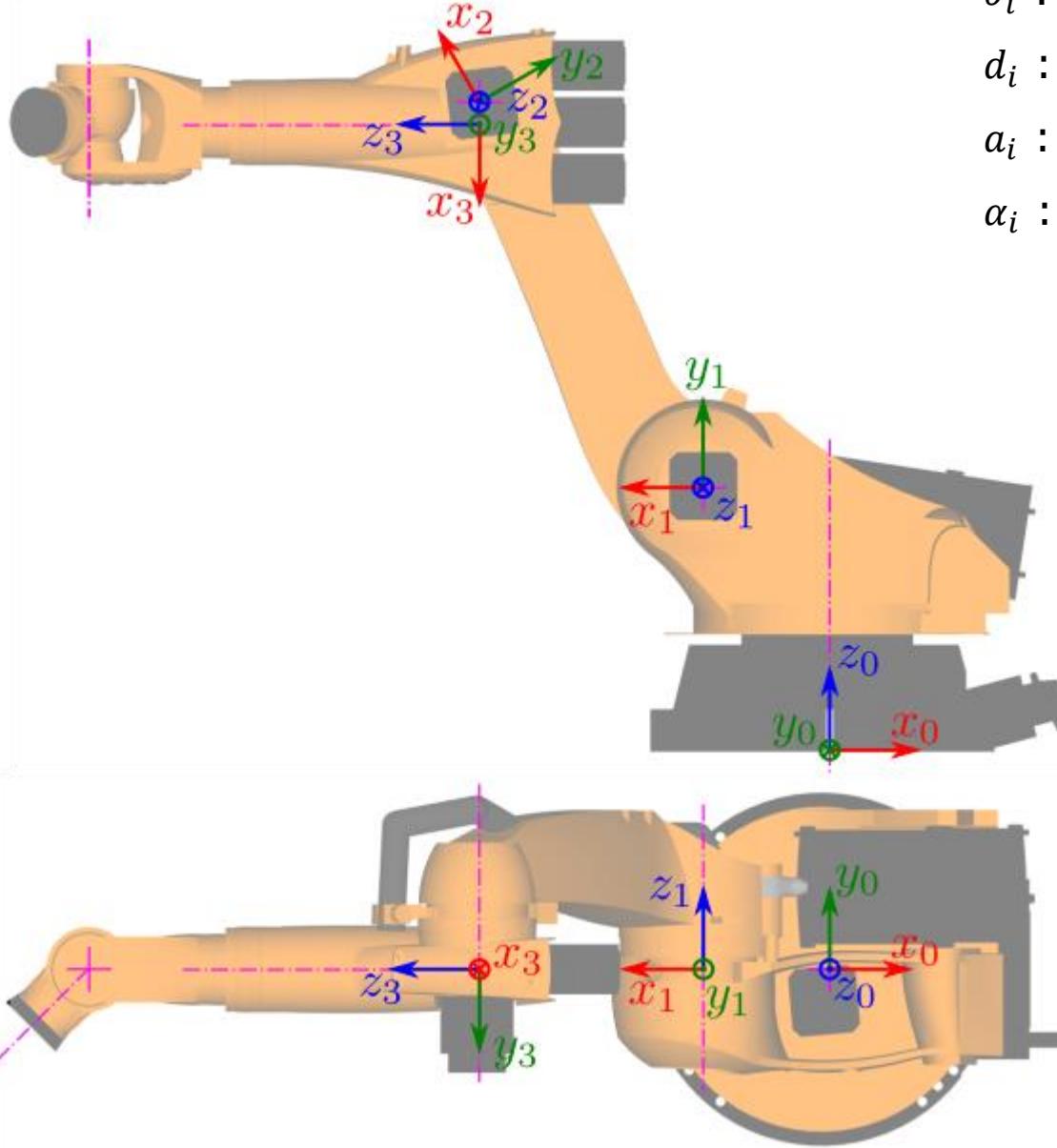
$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^* = 180$	200	100	90
2	$\theta_2^* = 60$	0	300	0
3				
4				
5				
6				

# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

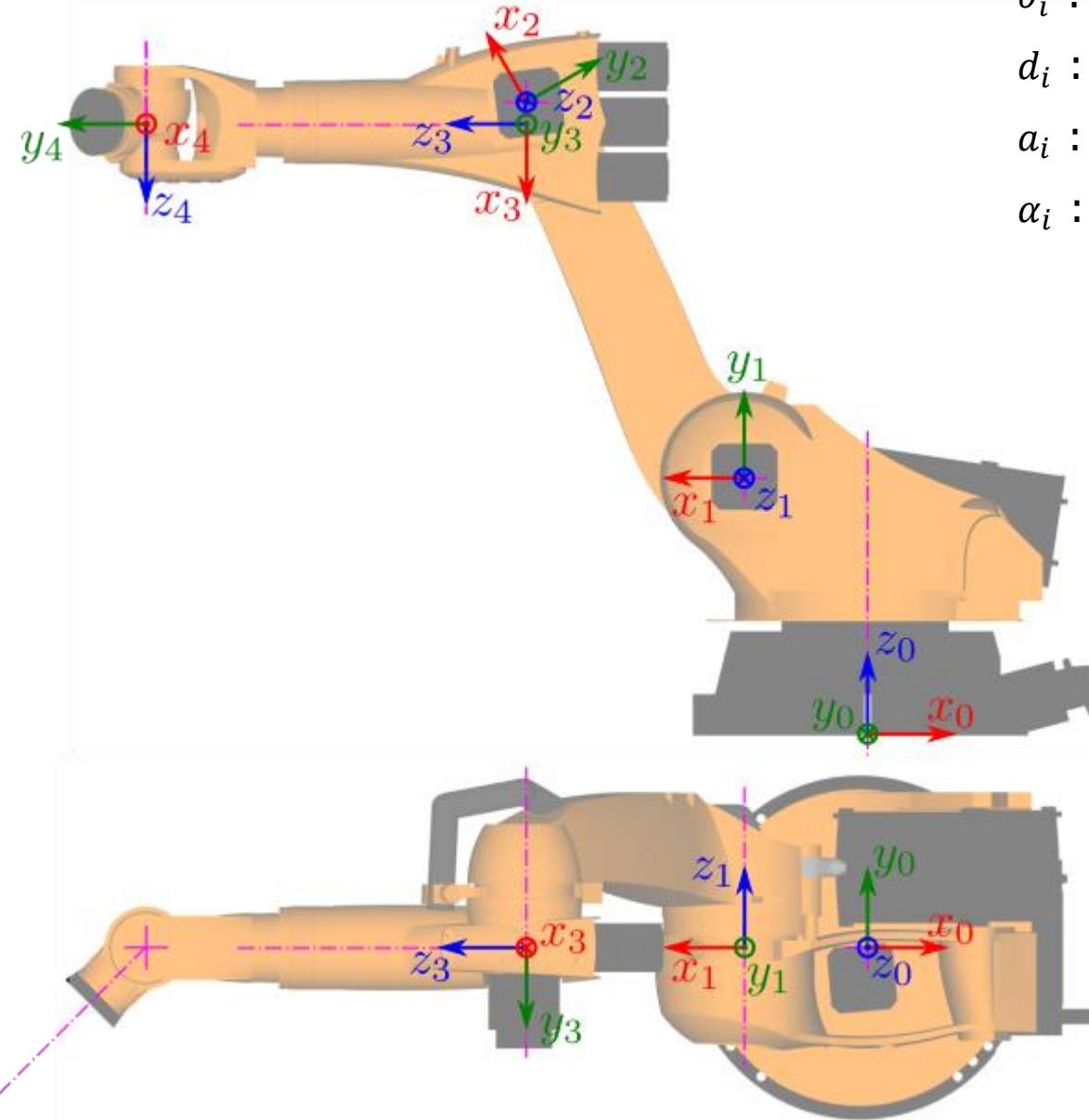
$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^* = 180$	200	100	90
2	$\theta_2^* = 60$	0	300	0
3	$\theta_3^* = -150$	0	20	-90
4				
5				
6				

# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

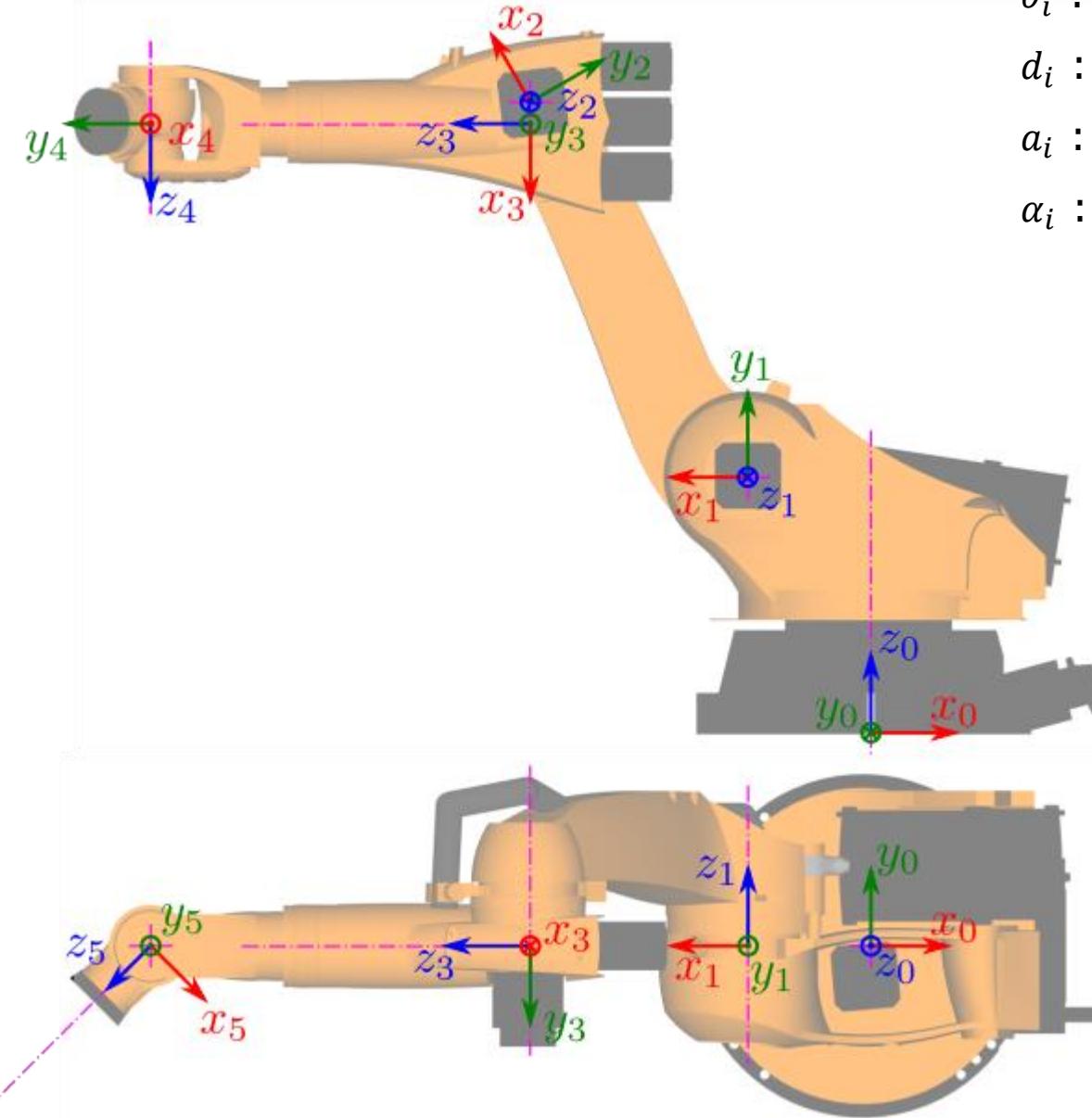
$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^* = 180$	200	100	90
2	$\theta_2^* = 60$	0	300	0
3	$\theta_3^* = -150$	0	20	-90
4	90	$d_4^* = 300$	0	90
5				
6				

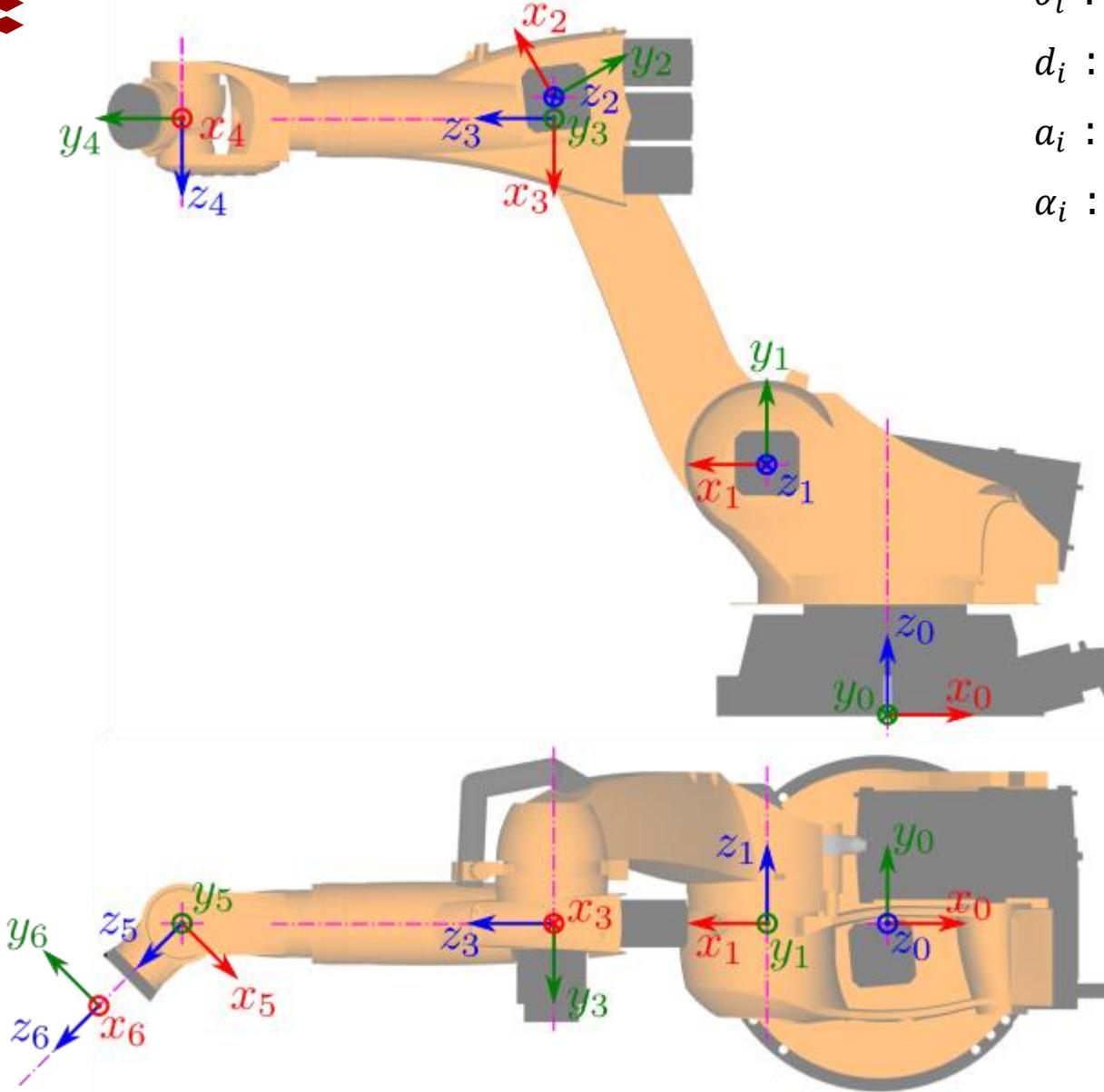
# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$   
 $d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )  
 $a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )  
 $\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^* = 180$	200	100	90
2	$\theta_2^* = 60$	0	300	0
3	$\theta_3^* = -150$	0	20	-90
4	90	$d_4^* = 300$	0	90
5	$\theta_5^* = -45$	0	0	-90
6				

# KUKA KR 210



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

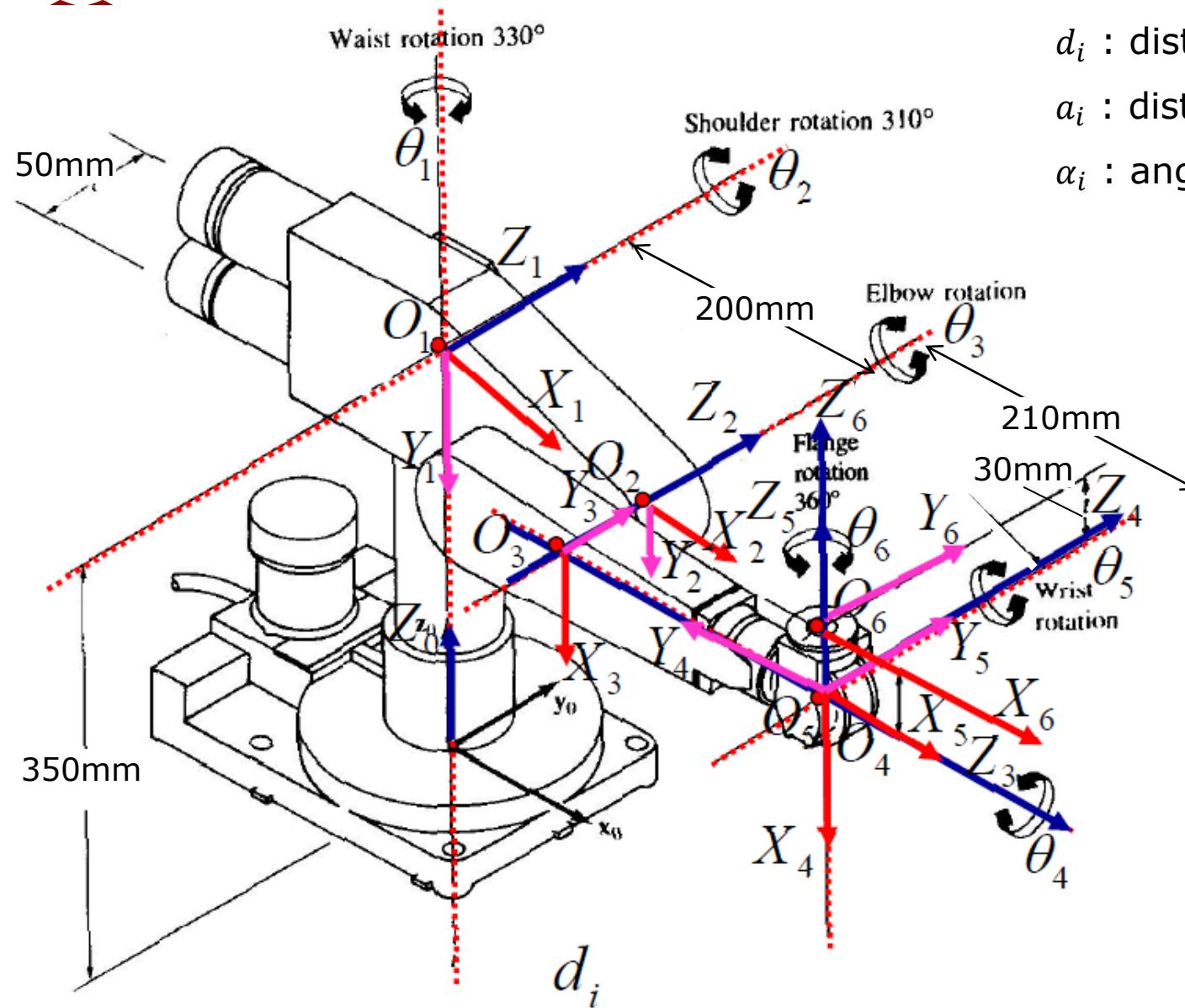
$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^* = 180$	200	100	90
2	$\theta_2^* = 60$	0	300	0
3	$\theta_3^* = -150$	0	20	-90
4	90	$d_4^* = 300$	0	90
5	$\theta_5^* = -45$	0	0	-90
6	$\theta_6^* = 90$	50	0	0

# PUMA 260



$\theta_i$  : angle from  $x_{i-1}$  to  $x_i$  about  $z_{i-1}$

$d_i$  : distance from  $o_{i-1}$  to  $x_i$  (along  $z_{i-1}$ )

$a_i$  : distance from  $z_{i-1}$  and  $o_i$  (along  $x_i$ )

$\alpha_i$  : angle from  $z_{i-1}$  to  $z_i$  about  $x_i$

Frame i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^*$	350	0	-90
2	$\theta_2^*$	0	200	0
3	$\theta_3^*$	-50	0	0
4	$\theta_4^*$	210	0	90
5	$\theta_5^*$	0	0	90
6	$\theta_6^*$	30	0	0

# Inverse Kinematics

# Inverse Kinematics

- **Forward kinematics:**

Known joint degrees of freedom  $q_1, q_2, \dots q_n$

→ find the pose of the end effector as:

$$H = T_n^0 = A_1(q_1) A_2(q_2) A_3(q_3) \dots A_n(q_n)$$

- **Inverse kinematics:**

Known homogeneous transformation  $H$  for end effector

→ solve the nonlinear system of equations

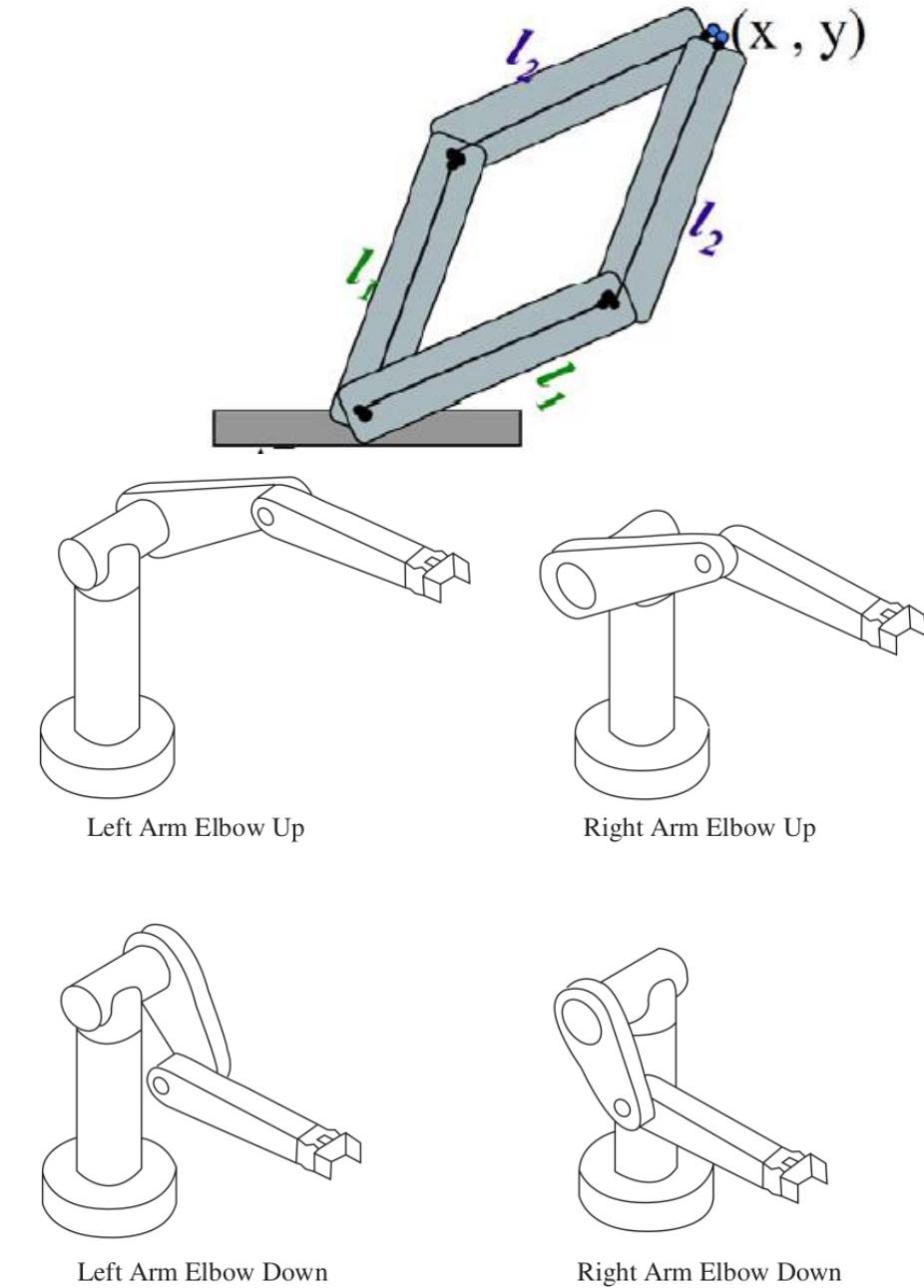
$$A_1(q_1) A_2(q_2) A_3(q_3) \dots A_n(q_n) = H$$

for  $q_1, q_2, \dots q_n$

→ 12 nonlinear equations → too difficult to find analytical solutions for

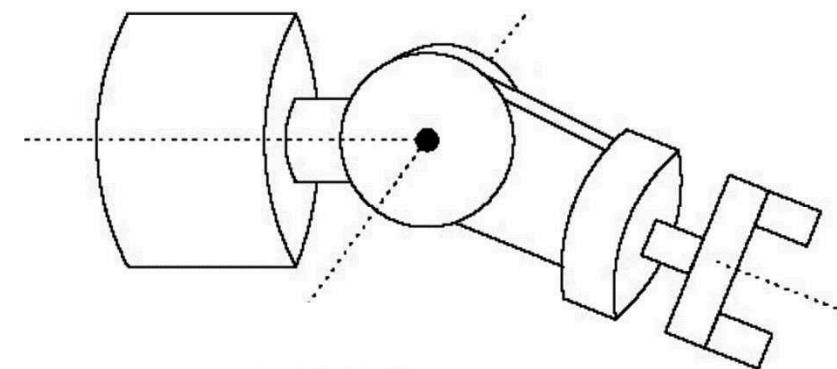
# Inverse Kinematics

- 12 nonlinear equations  
→ too difficult to find analytical solutions
- Not unique solutions
  - Redundant manipulator
  - Elbow-up/elbow-down solutions
- Kinematic decoupling
  - Inverse position: geometric approach
  - Inverse orientation Euler angles



# Inverse Kinematics – Kinematic Decoupling

- Decoupling of position and orientation
  - Assume a manipulator where the last 3 dofs correspond to a spherical wrist
  - First solve for the position of the wrist center
  - Then solve for the orientation of the end-effector

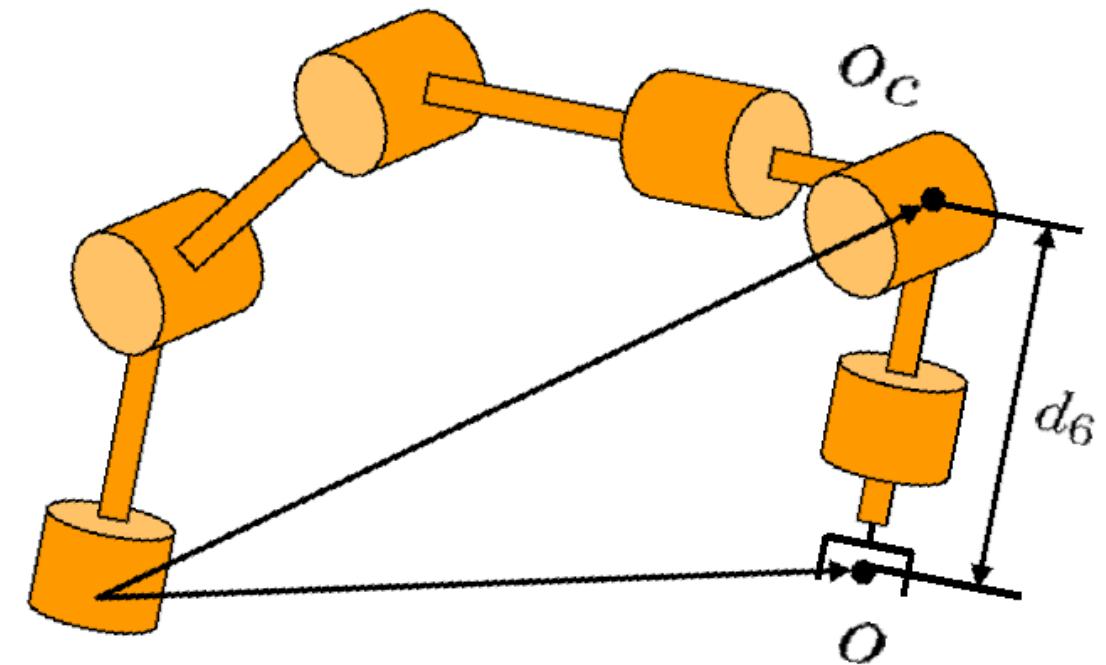


# Inverse Kinematics – Kinematic Decoupling

Kinematic decoupling for 6-DoF manipulator with spherical wrist:

- Inverse position kinematics  
→ wrist center
- Inverse orientation kinematics  
→ wrist orientation

Axes  $Z_3, Z_4, Z_5$  intersect at  $o_c$ :  
their rotations will not affect the position of  $o_c$



$$\begin{cases} R_6^0(q_1, \dots, q_6) = R \\ o_6^0(q_1, \dots, q_6) = o \end{cases}$$

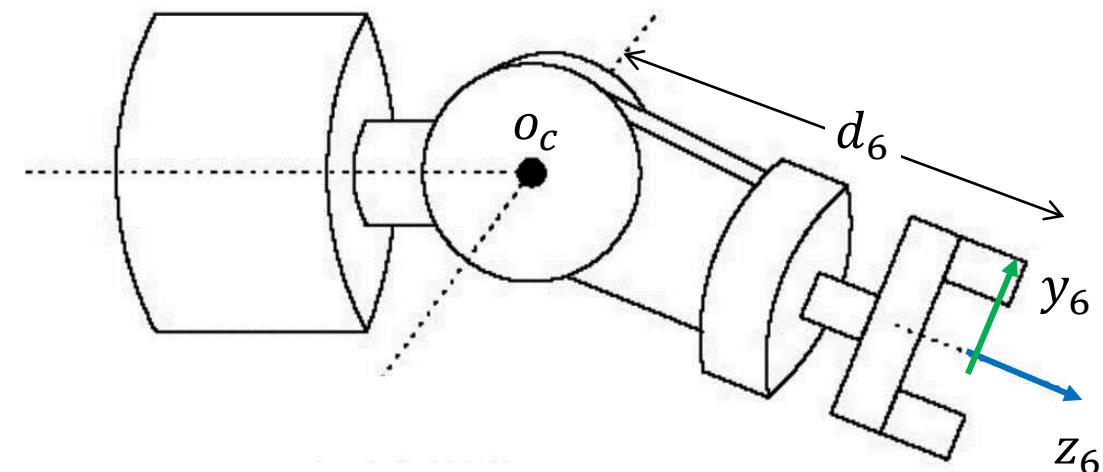
# Kinematic Decoupling

- Intended homogeneous transformation matrix:

$$H = A_1 A_2 A_3 A_4 A_5 A_6 = T_6^0 = \begin{bmatrix} R_6^0 & o_6^0 \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

Based on the wrist configuration we know:

$$o_c^0 = o_6^0 - d_6 R_6^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$



- We find  $q_1, q_2, q_3$  that satisfy  $o_c^0$
- We apply forward kinematics to calculate

$$R_3^0 = R_1^0(q_1) R_2^1(q_2) R_3^2(q_3)$$

- We find  $q_4, q_5, q_6$  from solving:

$$R_6^0 = R_3^0 R_6^3(q_4, q_5, q_6) \Rightarrow R_6^3(q_4, q_5, q_6) = (R_3^0)^T R_6^0$$

# Kinematic Decoupling

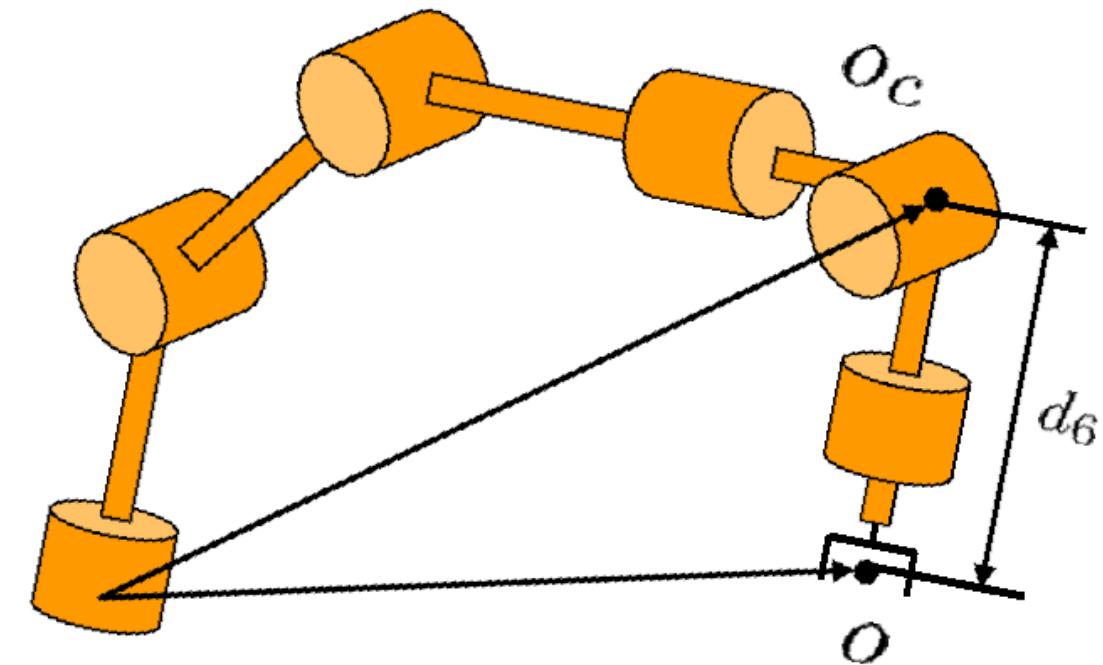
Kinematic decoupling for 6-DoF manipulator with spherical wrist:

Inverse position

$$o_c^0(q_1, q_2, q_3) = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

q<sub>1</sub>, q<sub>2</sub>, q<sub>3</sub>

Axes z<sub>3</sub>, z<sub>4</sub>, z<sub>5</sub> intersect at o<sub>c</sub>:  
their rotations will not affect the position of o<sub>c</sub>



Inverse orientation

$$R_6^3(q_4, q_5, q_6) = (R_3^0(q_1, q_2, q_3))^T R_6^0$$

q<sub>4</sub>, q<sub>5</sub>, q<sub>6</sub>

$$\left\{ \begin{array}{l} R_6^0(q_1, \dots, q_6) = R \\ o_6^0(q_1, \dots, q_6) = o_c \end{array} \right.$$

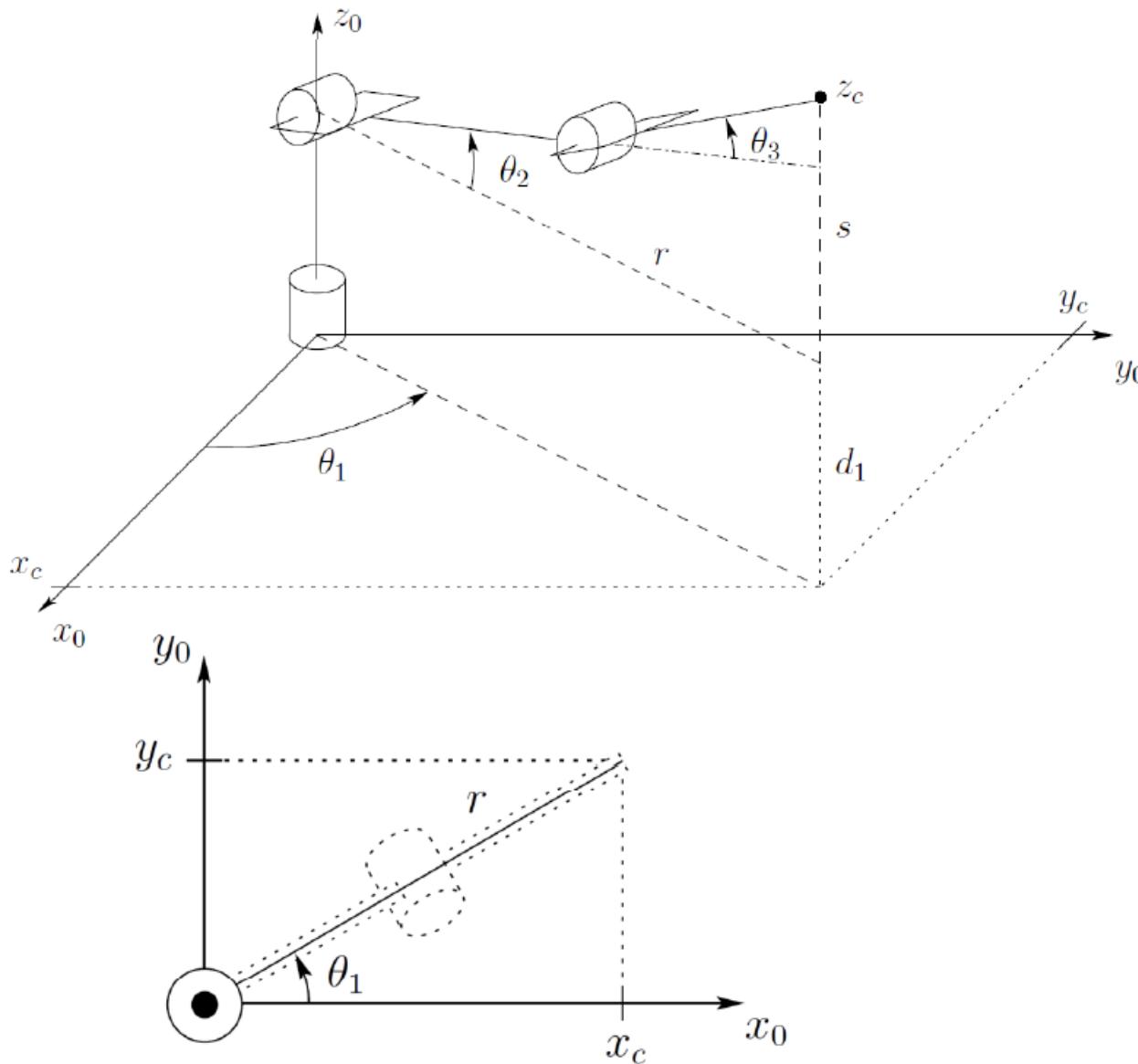
# Inverse position

- We find  $q_1, q_2, q_3$  that satisfy:

$$o_c^0(q_1, q_2, q_3) = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

- But How?
- By a graphical method:
  - Projecting the manipulator onto the  $xy$ -plane of a link frame
  - Applying trigonometry on the projected geometry

# Inverse Kinematics of Articulated Manipulator

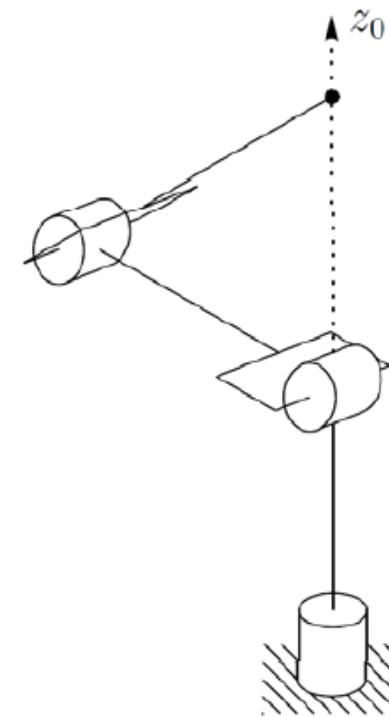


The angle  $\theta_1$  is easy to determine:

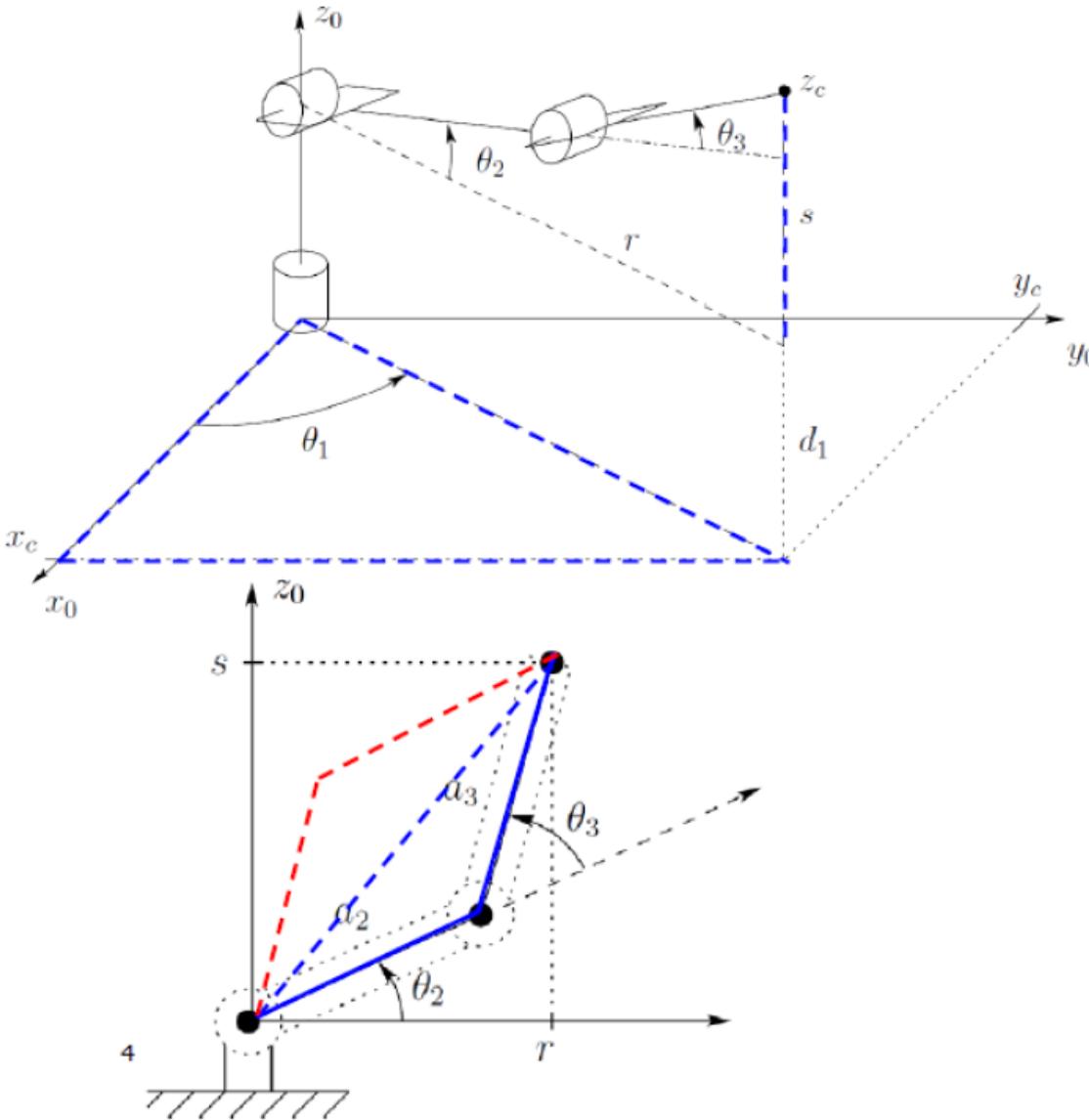
$$\theta_1 = \text{Atan2}(x_c, y_c)$$

when  $(x_c, y_c) \neq (0,0)$

Otherwise for  $(x_c, y_c) = (0,0)$  there is a singularity w.r.t. determining  $\theta_1$ :



# Inverse Kinematics of Articulated Manipulator



From the law of cosines:

$$\cos(\theta_3) = \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2a_3}$$

where:

$$r^2 = x_c^2 + y_c^2$$

$$s = z_c - d_1$$

There are two solutions:

$$\theta_3 = \text{Atan2}\left(c_3, \pm\sqrt{1 - c_3^2}\right)$$

(elbow-down or elbow-up)

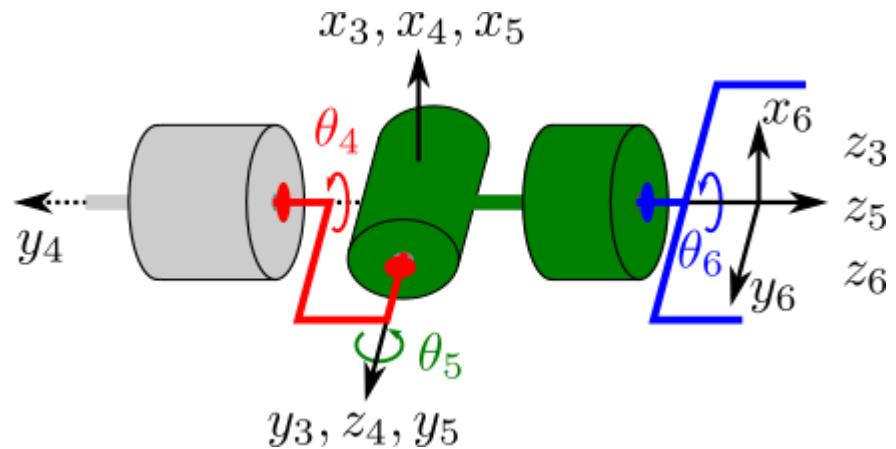
Then one can also find:

$$\theta_2 = \text{Atan2}(r, s) - \text{Atan2}(a_2 + a_3c_3, a_3s_3)$$

# Inverse Orientation

- Find  $\theta_4, \theta_5, \theta_6$  that satisfy

$$R_6^3(\theta_4, \theta_5, \theta_6) = (R_3^0)^T R_6^0$$



$$\begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \end{bmatrix} = (R_3^0)^T R_6^0$$

→ Same problem as finding the Euler angles in Lecture 2:

$$\theta_5 = \theta = \text{Atan2}\left(r_{33}, \sqrt{1 - r_{33}^2}\right)$$

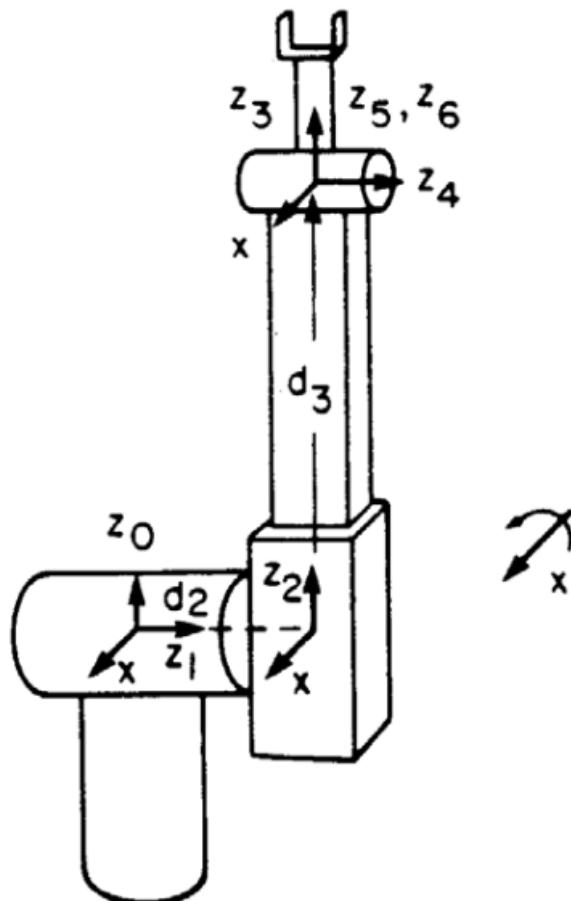
$$\theta_4 = \phi = \text{Atan2}(r_{13}, r_{23})$$

$$\theta_6 = \psi = \text{Atan2}(-r_{31}, r_{32})$$

# Exercises

## Problem 1

Given the Stanford arm in the figure below, with  $d_2 = 0.1 \text{ m}$ , answer the following questions.



### Question 1

Find the link parameters for the robotic arm ( $d_3$  is a prismatic joint variable, other joints are rotational joints, the link coordinate frames have been established as shown in the figure).

Joint i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1				
2				
3				
4				
5				
6				

### Question 2

Find the forward kinematic model for the arm and represent it in homogeneous matrix form.

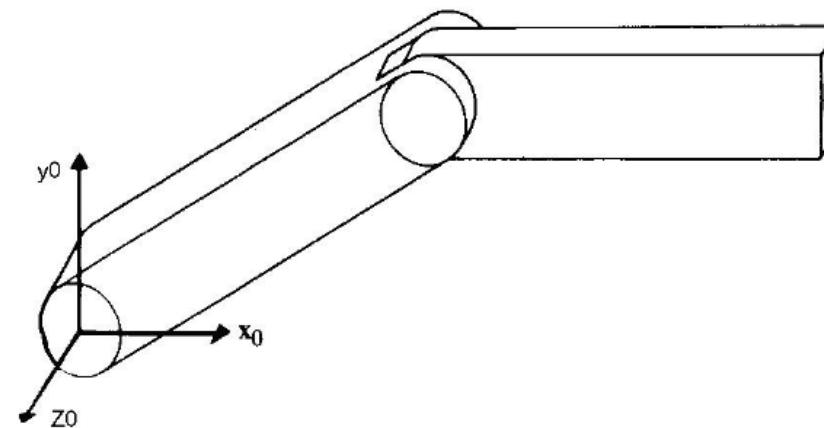
### Question 3

Represent the orientation of the end-effector with Yaw-Pitch-Roll angles.

# Exercises

## Problem 2

A two degree-of-freedom manipulator is shown in the figure below. Given that the length of each link is 1 m, establish its link coordinate frames and find  $T_1^0$ ,  $T_2^1$  and the kinematics matrix. For coordinate frame 2, assume a revolute joint at the tip of the robotic arm with its axis parallel to  $z_0$ .



### Question 1

Find the forward kinematics solution for this manipulator, i.e. the homogeneous transformation matrix for the end-effector as a function of the joint angles.

### Question 2

Find the inverse kinematics solution for this manipulator assuming the position of the robot tip is known, i.e. elements  $r_{14}$  and  $r_{24}$  in the homogeneous transformation matrix. (Hint: use trigonometry and the law of cosines)

Robotics – 34753

# Velocity Kinematics & The Jacobian Matrix

Konstantinos Poulios  
Associate Professor

Department of Civil and Mechanical Engineering  
DTU Lyngby, building 404 / room 124

# Velocity Kinematics & The Jacobian Matrix – Lecture Overview

## 1. Repetition

## 2. Angular Velocities

- Skew Symmetric Matrices
- Addition of Angular Velocities

## 3. Linear Velocities

- Fixed Point on a Link

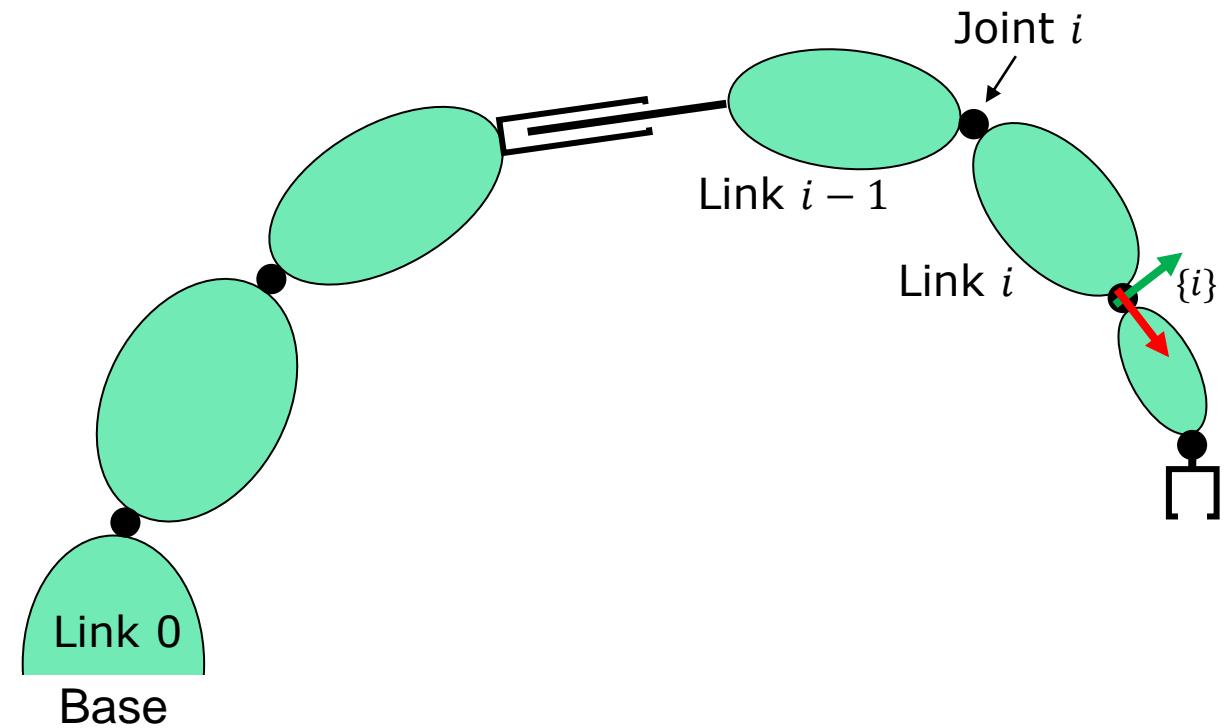
## 4. The Manipulator Jacobian

- Singularities
- Inverse Jacobian

# Repetition

# Repetition

- Kinematic Model
    - Joint  $i$  is fixed to link  $i - 1$
    - Joint  $i$  actuation  $\rightarrow$  motion of link  $i$  and frame  $o_i x_i y_i z_i$



# Repetition

- D-H convention: 4 basic transformations → Homogeneous transformation

$$A_i = \text{Rot}_{z,\theta_i} \text{Trans}_{z,d_i} \text{Trans}_{x,a_i} \text{Rot}_{x,\alpha_i}$$

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$\theta_i$  : joint angle  
 $d_i$  : link offset  
 $a_i$  : link length  
 $\alpha_i$  : link twist

versus

Six parameters  
 $\phi, \theta, \psi,$   
 $d_x, d_y, d_z$

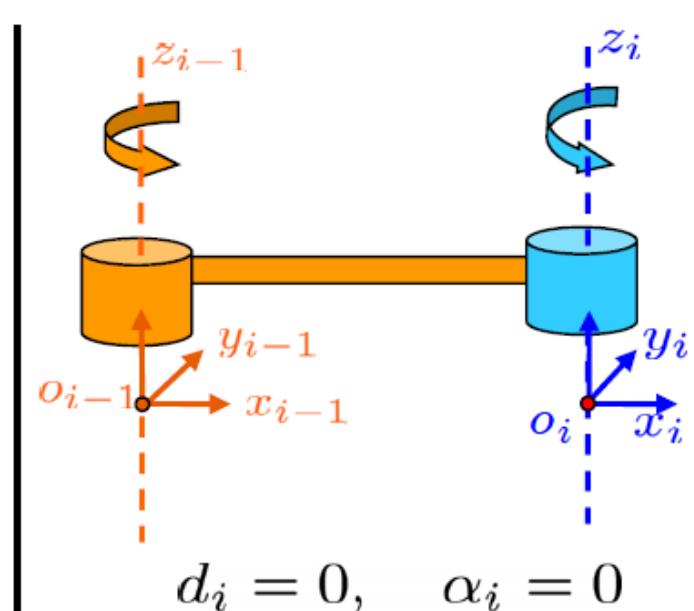
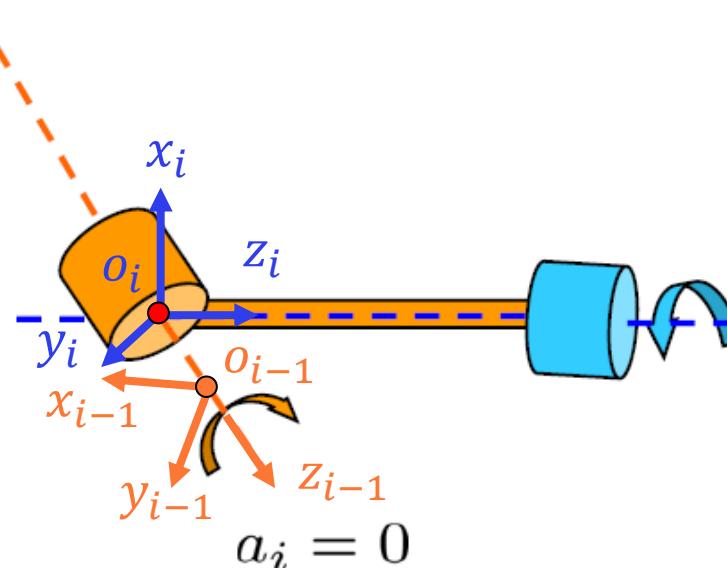
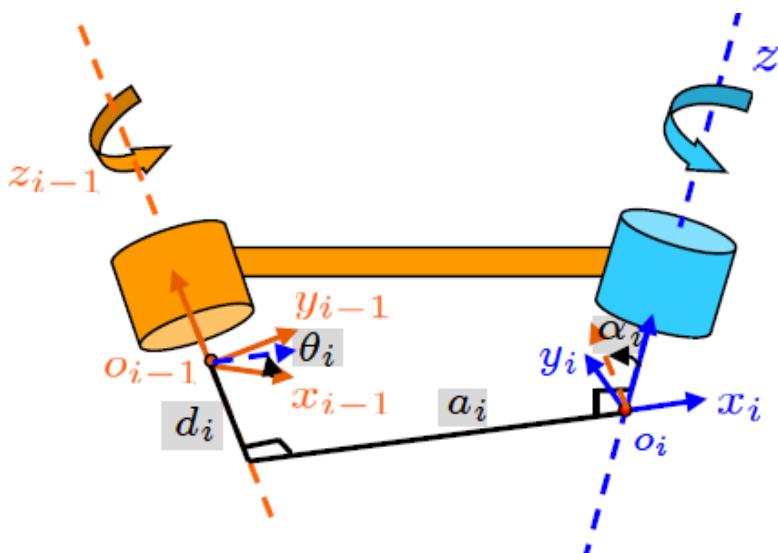
# Repetition

Physical interpretation of  $\theta, d, a, \alpha$ :

- $\theta_1$  : angle from  $x_0$  to  $x_1$  about  $z_0$
- $d_1$  : distance from  $o_0$  to  $x_1$  (along  $z_0$ )
- $a_1$  : distance from  $z_0$  to  $o_1$  (along  $x_1$ )
- $\alpha_1$  : angle from  $z_0$  to  $z_1$  about  $x_1$



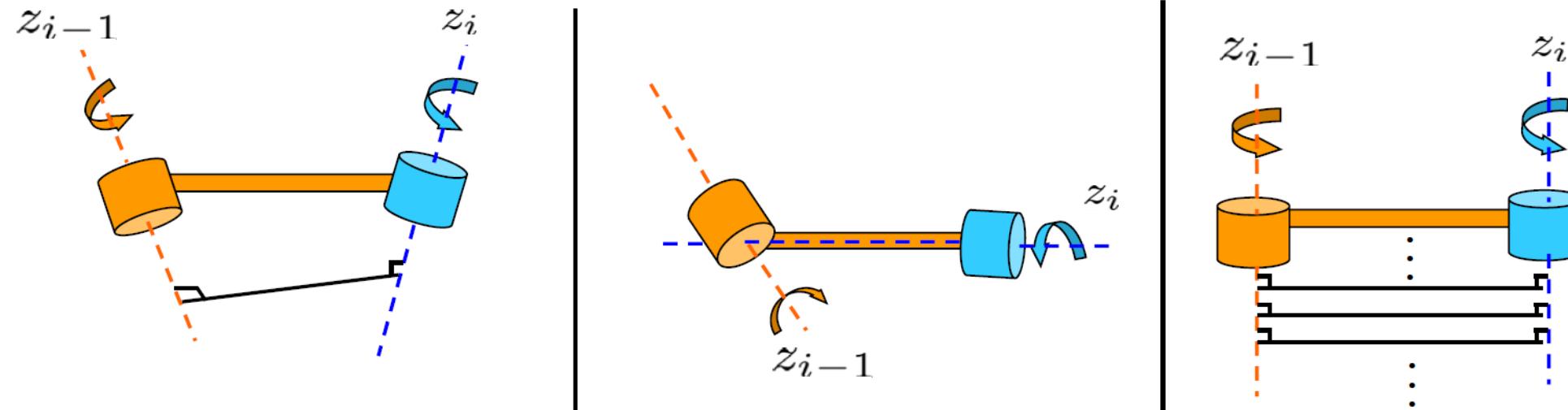
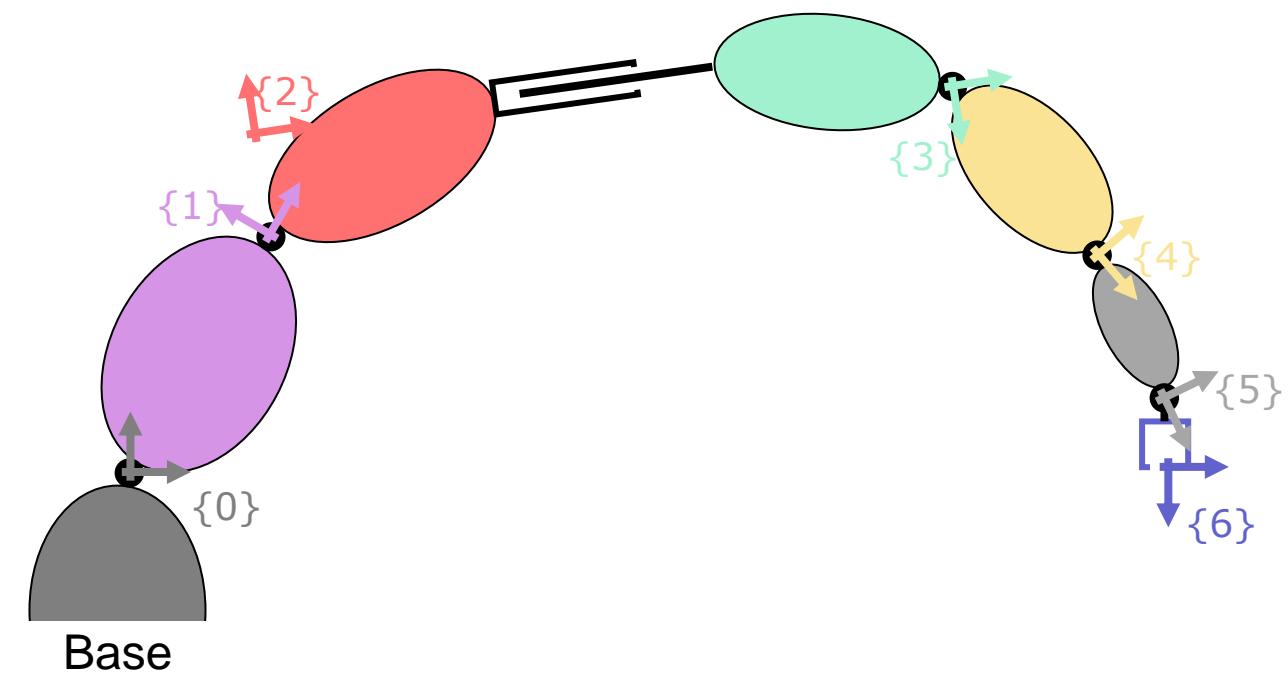
one of both is a variable:  
 $\theta_1$  for revolute,  $d_1$  for prismatic  
 always constant  
 characteristic of the manipulator



# Repetition

Assignment of Coordinate Frames:

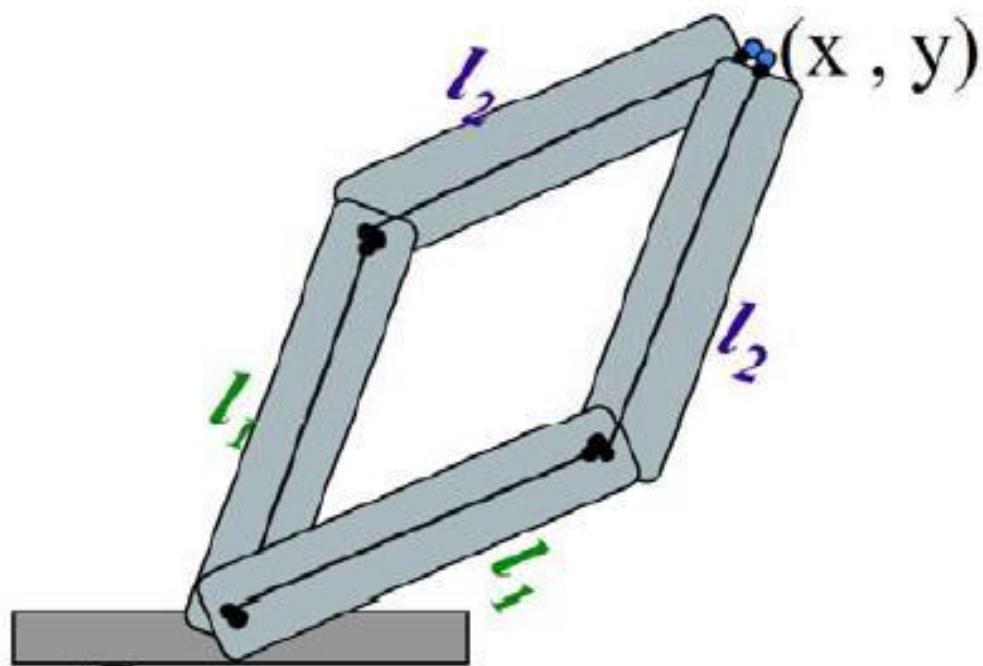
- $z_i$  axis along the  $i + 1$  joint axis
- $x_i$  axis parallel to  $z_i \times z_{i-1}$
- $y_i$  axis parallel to  $z_i \times x_i$
- Origin  $o_i$  along  $z_i$  at the point of shortest distance to  $z_{i-1}$



# Repetition

Inverse kinematics:

- 12 nonlinear equations  
→ too difficult to find analytical solutions
- Not unique solutions
  - Redundant manipulator
  - Elbow-up/elbow-down solutions
- Kinematic decoupling
  - Inverse position: geometric approach
  - Inverse orientation Euler angles

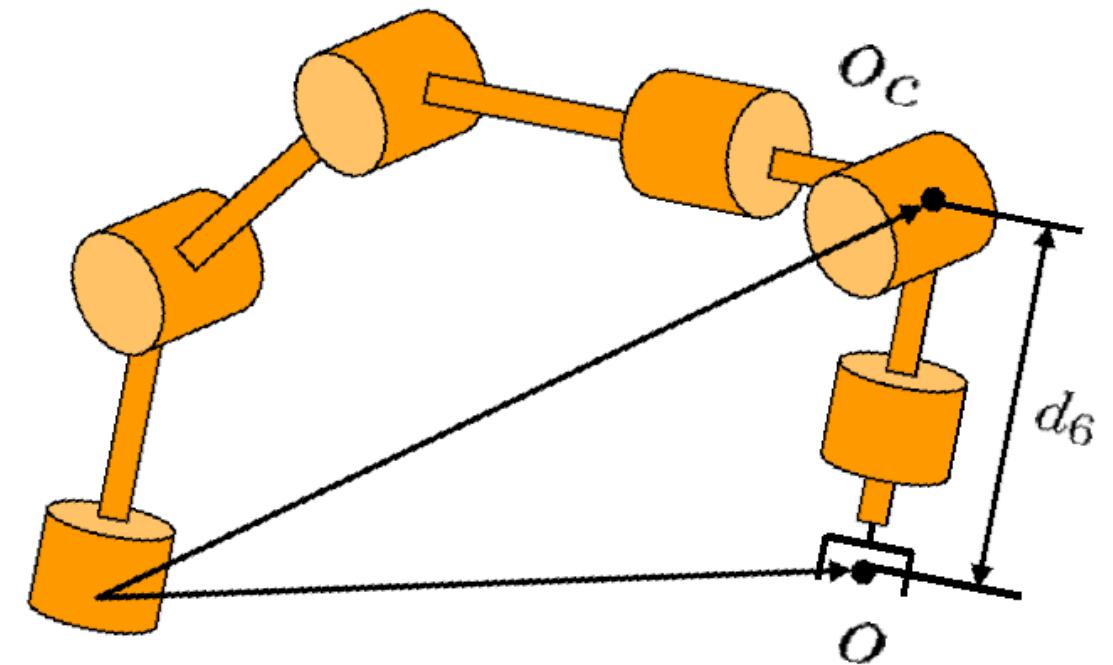


# Repetition

Kinematic decoupling for 6-DoF manipulator with spherical wrist:

- Inverse position kinematics  
→ wrist center
- Inverse orientation kinematics  
→ wrist orientation

Axes  $z_3, z_4, z_5$  intersect at  $o_c$ :  
their rotations will not affect the  
position of  $o_c$



$$\begin{cases} R_6^0(q_1, \dots, q_6) = R \\ o_6^0(q_1, \dots, q_6) = \underline{\quad} \end{cases}$$

# Repetition

Kinematic decoupling for 6-DoF manipulator with spherical wrist:

Inverse position

$$o_c^0(q_1, q_2, q_3) = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

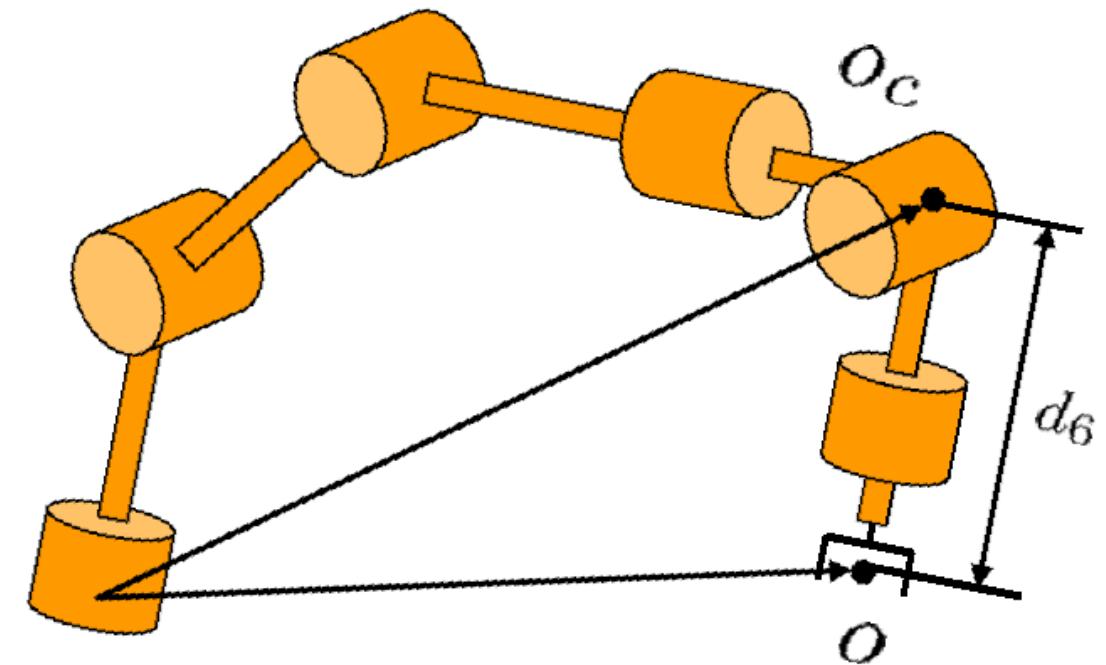
$\xrightarrow{\hspace{1cm}} q_1, q_2, q_3$

Inverse orientation

$$R_6^3(q_4, q_5, q_6) = (R_3^0(q_1, q_2, q_3))^T R_6^0$$

$\xrightarrow{\hspace{1cm}} q_4, q_5, q_6$

Axes  $z_3, z_4, z_5$  intersect at  $o_c$ : their rotations will not affect the position of  $o_c$

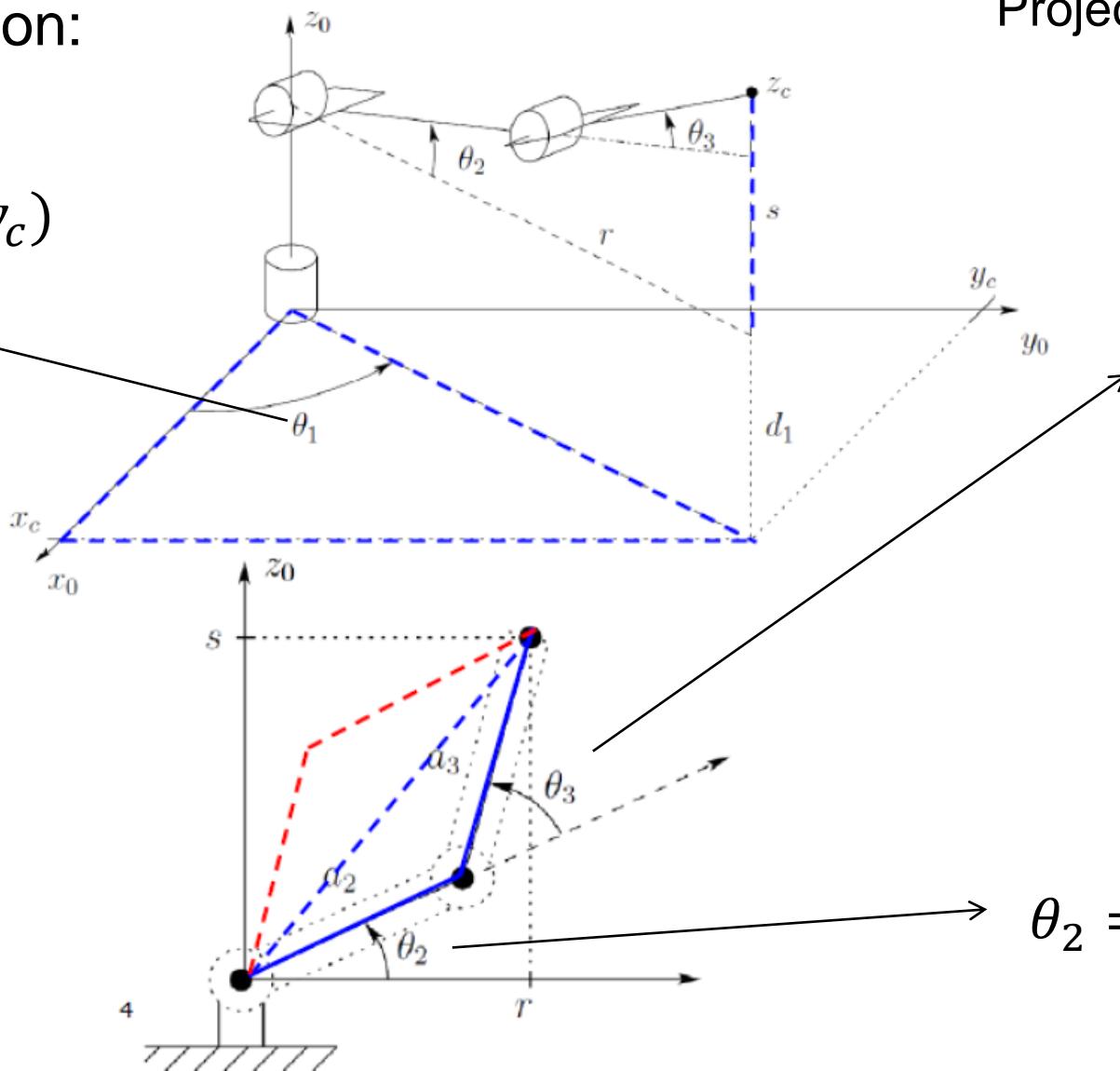


$$\left\{ \begin{array}{l} R_6^0(q_1, \dots, q_6) = R \\ o_c^0(q_1, \dots, q_6) = o \end{array} \right.$$

# Repetition

Inverse position:

$$\theta_1 = \text{Atan2}(x_c, y_c)$$



Projecting the manipulator onto  
an  $x_i y_i$  plane

$$\theta_3 = \text{Atan2}\left(c_3, \pm\sqrt{1 - c_3^2}\right)$$

$$c_3 = \frac{r^2 + s^2 - a_2^2 - a_3^2}{2a_2a_3}$$

$$r^2 = x_c^2 + y_c^2$$

$$s = z_c - d_1$$

$$\begin{aligned} \theta_2 = & \text{Atan2}(r, s) \\ & -\text{Atan2}(a_2 + a_3 c_3, a_3 s_3) \end{aligned}$$

# Repetition

Inverse orientation:

- Find  $\theta_4, \theta_5, \theta_6$  that satisfy

$$\begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 \\ s_4 c_5 c_6 + c_4 s_6 & -s_4 c_5 s_6 + c_4 c_6 & s_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \end{bmatrix} = (R_3^0)^T R_6^0$$

$$\theta_5 = \theta = \text{Atan2}\left(r_{33}, \sqrt{1 - r_{33}^2}\right)$$

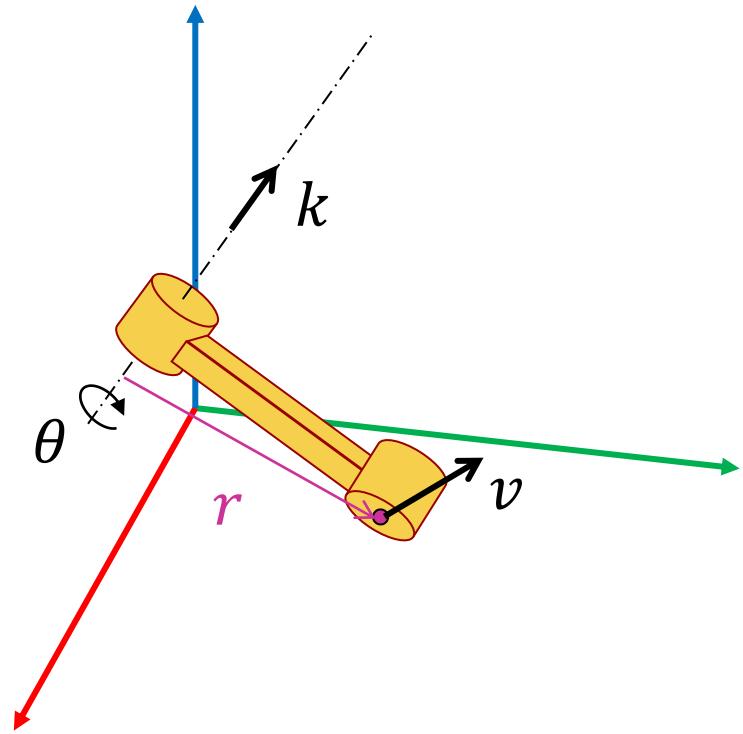
$$\theta_4 = \phi = \text{Atan2}(r_{13}, r_{23})$$

$$\theta_6 = \psi = \text{Atan2}(-r_{31}, r_{32})$$

# Angular Velocities

# Angular Velocity as a Vector

- Axis of rotation  $k$  (unit vector)
- Rotation angle  $\theta \rightarrow$  Rotation matrix  $R_{k,\theta}$
- Time derivative  $\dot{\theta}$
- Angular velocity (vector)  
$$\omega = \dot{\theta}k$$
- Linear velocity of a point at a distance  $r$   
$$v = \omega \times r$$



# Cross Product through a Skew Symmetric Matrix

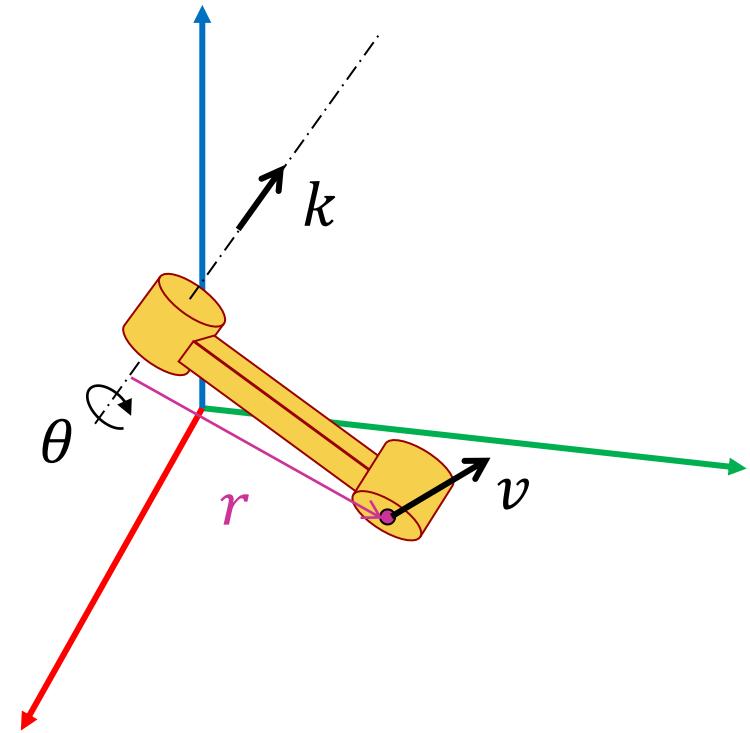
- Angular velocity (vector)

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \dot{\theta} k$$

- Linear velocity of a point  $p$

$$v = \omega \times r = \underbrace{\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}}_{S(\omega)} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$

skew symmetric matrix



# Properties of Skew Symmetric Matrices

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \rightarrow S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Example:

- $S(\omega) = \begin{bmatrix} 0 & 3 & -1 \\ -3 & 0 & -4 \\ 1 & 4 & 0 \end{bmatrix}$

Properties:

- $S + S^T = 0$
- $S(\alpha a + \beta b) = \alpha S(a) + \beta S(b)$
- $S(R a) = RS(a)R^T$ , for any orthonormal matrix  $R$
- $x^T S(a)x = 0$ , for any vector  $x$

- $\omega = ?$
- $\dot{\theta} = ?$
- $k = ?$

# Derivative of a Rotation Matrix

$$RR^T = I \quad \Rightarrow \quad \frac{dR}{d\theta} R^T + R \frac{dR^T}{d\theta} = 0$$

Which means

- $\frac{dR}{d\theta} R^T = S \quad \Rightarrow \quad \frac{dR}{d\theta} = S R(\theta)$

Example:

- $R_{z,\theta} = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- $\frac{dR_{z,\theta}}{d\theta} R_{z,\theta}^T = \begin{bmatrix} -s_\theta & -c_\theta & 0 \\ c_\theta & -s_\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_\theta & s_\theta & 0 \\ -s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

# Derivative of a Rotation Matrix

$$RR^T = I \quad \Rightarrow \quad \frac{dR}{d\theta} R^T + R \frac{dR^T}{d\theta} = 0$$

Which means

$$\bullet \frac{dR}{d\theta} R^T = S \quad \Rightarrow \quad \frac{dR}{d\theta} = S R(\theta)$$

Time derivative

$$\bullet \dot{R}_{k,\theta} = \frac{dR}{d\theta} \frac{d\theta}{dt} = S(k) R_{k,\theta} \dot{\theta} = \dot{\theta} S(k) R_{k,\theta} \quad \Rightarrow \quad \dot{R}_{k,\theta} = S(\dot{\theta} k) R_{k,\theta}$$

$\underbrace{\phantom{S(\dot{\theta} k) R_{k,\theta}}}_{\omega}$

$\omega$

# Addition of Angular Velocities

General case:

$$\dot{R}(t) = S(\omega(t))R(t)$$

$\omega(t)$ : instantaneous angular velocity vector

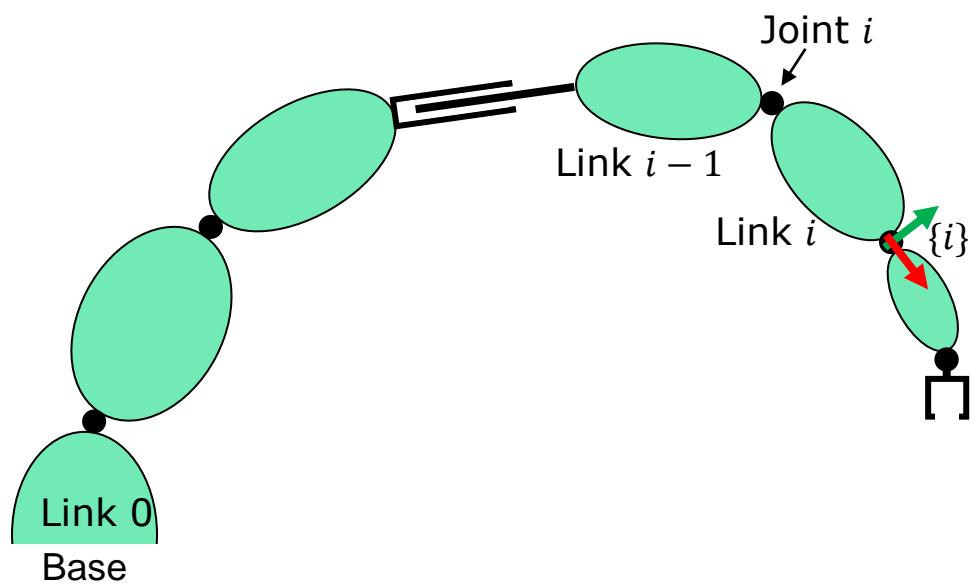
Angular velocity vector of link  $i$  w.r.t link  $i - 1$   
expressed in frame  $i - 1$ :

$$\omega_{i-1,i}^{i-1}$$

With this notation the angular velocity of the  
end-effector is

$$\omega_{0,n}^0 = \omega_{0,1}^0 + R_1^0 \omega_{1,2}^1 + \cdots + R_{n-1}^0 \omega_{n-1,n}^{n-1}$$

$$\omega_{0,n}^0 = \omega_{0,1}^0 + \omega_{1,2}^0 + \cdots + \omega_{n-1,n}^0$$



# Linear Velocities

# Linear Velocity of a Fixed Point on a Link

Frame  $i$  attached to link  $i$  with

$$H_i^0 = \begin{bmatrix} R_i^0 & o_i^0 \\ 0 & 1 \end{bmatrix}$$

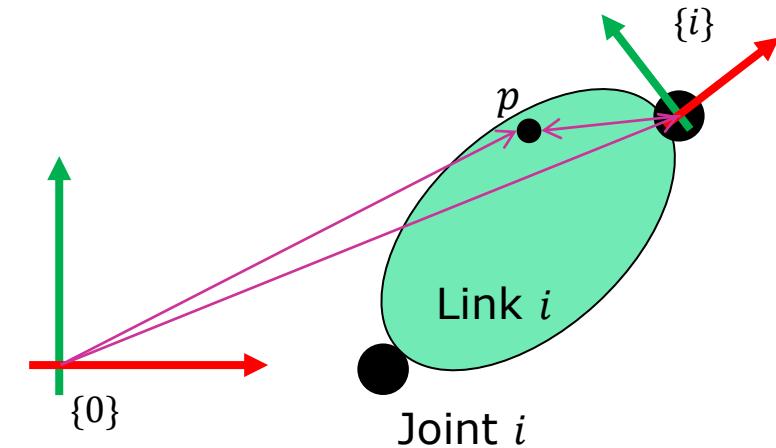
Consider a point  $p = p^i$  on link  $i$  fixed in frame  $i$ . The point coordinates wrt. frame 0:

$$p^0 = R_i^0 p^i + o_i^0$$

Its linear velocity:

$$\dot{p}^0 = \dot{R}_i^0 p^i + \dot{o}_i^0 = \underbrace{S(\omega_i^0) R_i^0}_{\omega \times r} p^i + \dot{o}_i^0$$

where  $r = p^0 - o_i^0$



Based on the fact that  $p^i$  fixed, i.e.  $\dot{p}^i = 0$

# The Manipulator Jacobian

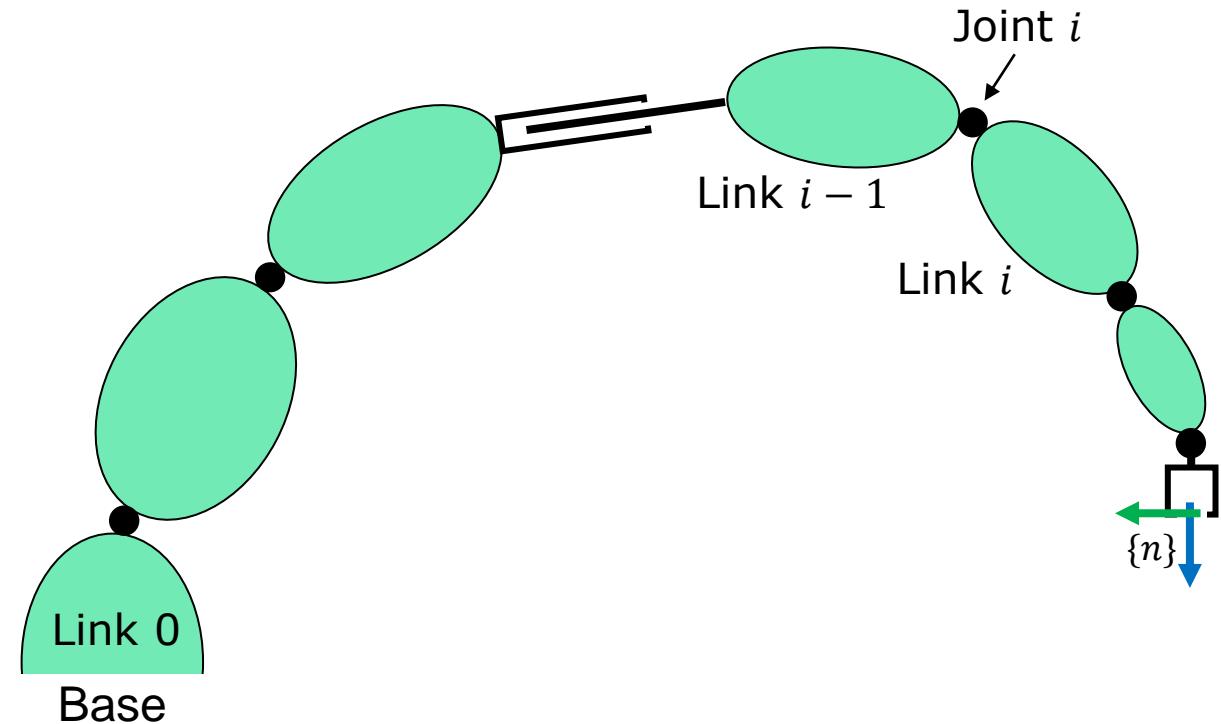
# The Manipulator Jacobian

- Joint variables  $q_i(t) \rightarrow$  joint velocities  $\dot{q}_i$
- End effector position  $o_n^0(t) \rightarrow$  linear velocity  $v_n^0 = \dot{o}_n^0$
- End effector orientation  $R_n^0(t) \rightarrow$  angular velocity  $\omega_n^0$  such that  $S(\omega_n^0) = \dot{R}_n^0(R_n^0)^T$

For a given configuration  $q$   
we seek a relationship in the form:

$$\xi = \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_{11} & \cdots & J_{1n} \\ \vdots & \ddots & \vdots \\ J_{61} & \cdots & J_{6n} \end{bmatrix} \dot{q} = J \dot{q}_i$$

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$



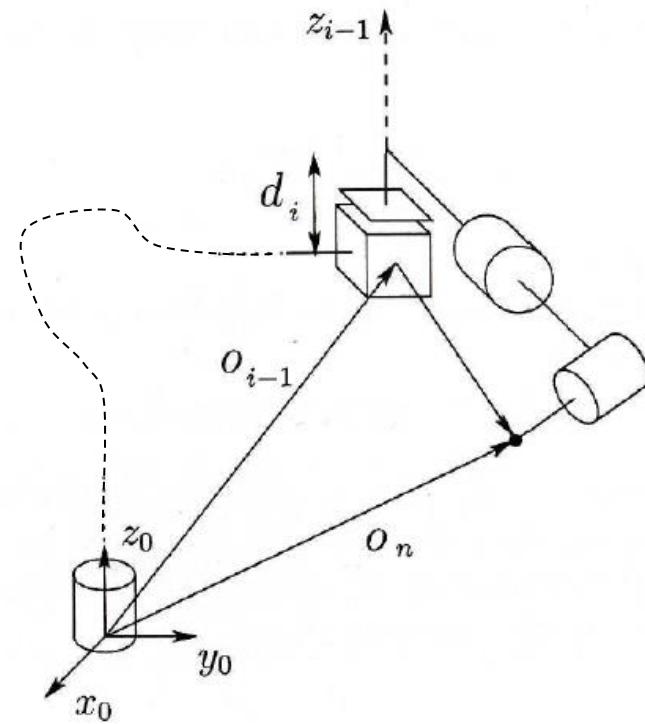
# Derivation of the Jacobian

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

Prismatic Joint

$$v_n^0 = \dot{o}_n^0 = \dot{d}_i z_{i-1}$$

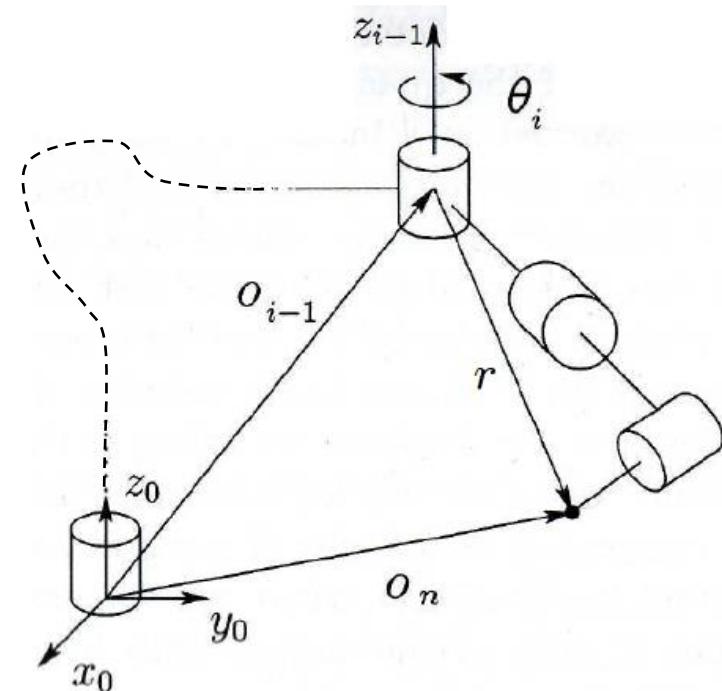
i.e.  $J_{v_i} = z_{i-1}$



Revolute Joint

$$v_n^0 = \dot{o}_n^0 = \omega \times r = \dot{\theta}_i z_{i-1} \times (o_n - o_{i-1})$$

i.e.  $J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$



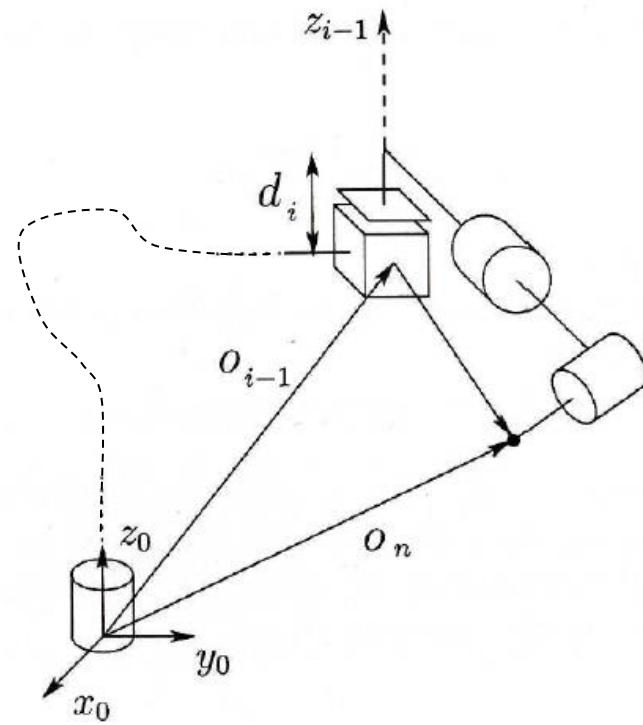
# Derivation of the Jacobian

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

Prismatic Joint

$$\omega_n^0 = 0$$

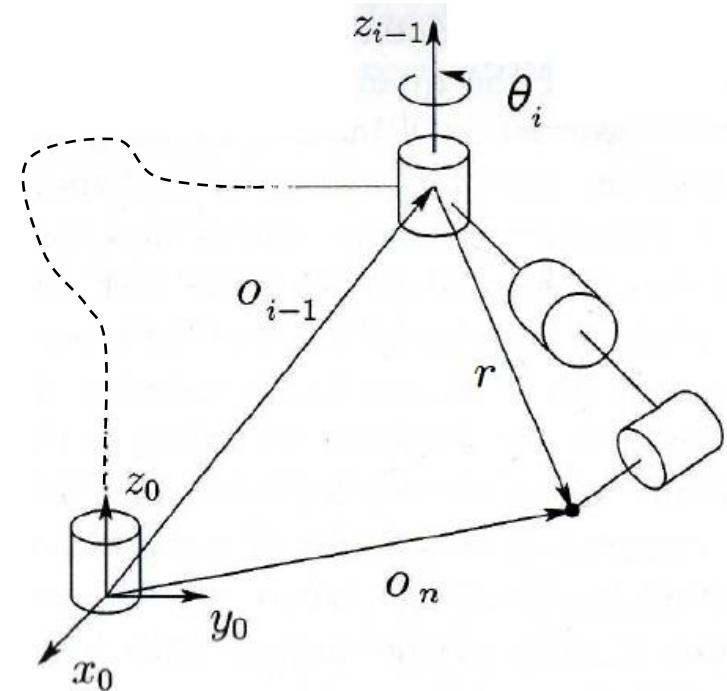
i.e.  $J_{\omega_i} = 0$



Revolute Joint

$$\omega_n^0 = \dot{\theta}_i z_{i-1}^0$$

i.e.  $J_{\omega_i} = z_{i-1}$



# Derivation of the Jacobian

$$\begin{bmatrix} v_n^0 \\ w_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

Prismatic Joint

$$J_{v_i} = z_{i-1}$$

$$J_{\omega_i} = 0$$

Revolute Joint

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$$

$$J_{\omega_i} = z_{i-1}$$

$$J = [J_1 \quad \cdots \quad J_i \quad \cdots \quad J_n]$$

where

$$J_i = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}, & i \text{ prismatic} \\ \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}, & i \text{ revolute} \end{cases}$$

Where do we get  $z_{i-1}$  and  $o_{i-1}$  from:

$$T_{i-1} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & o_x \\ r_{21} & r_{22} & r_{23} & o_y \\ r_{31} & r_{32} & r_{33} & o_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Example of a Jacobian

Jacobian matrix of a two-link planar manipulator:

- $J = \begin{bmatrix} z_0 \times (o_n - o_0) & z_1 \times (o_n - o_1) \\ z_0 & z_1 \end{bmatrix}$

- $o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$     $o_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ 0 \end{bmatrix}$     $o_2 = \begin{bmatrix} a_1 c_1 + a_2 c_{12} \\ a_1 s_1 + a_2 s_{12} \\ 0 \end{bmatrix}$

- $z_0 = z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

$$\Rightarrow J = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Prismatic Joint

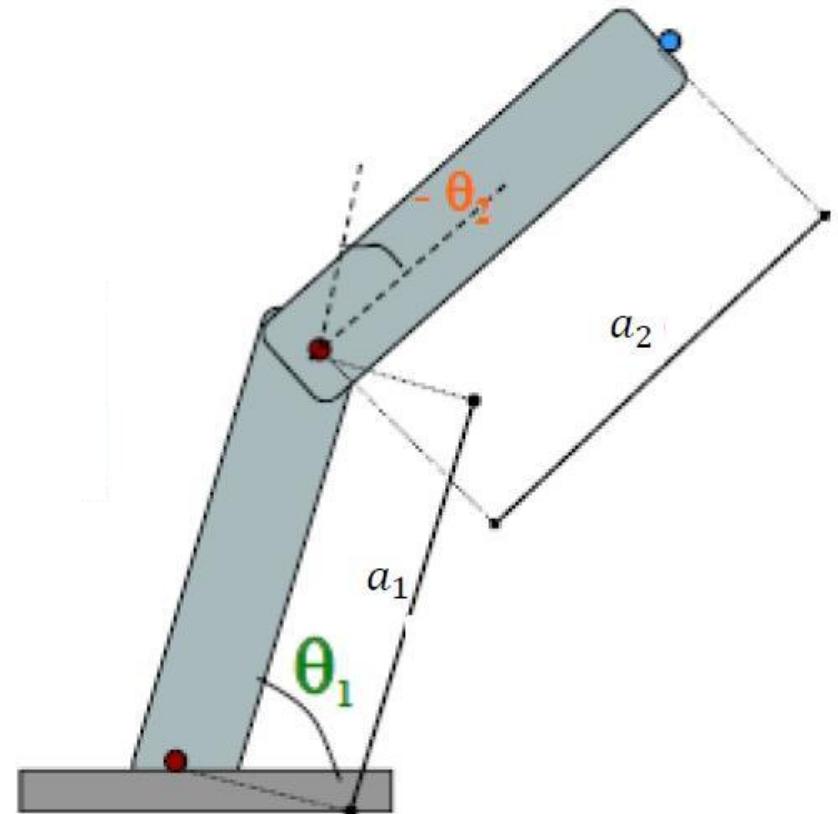
$$J_{v_i} = z_{i-1}$$

$$J_{\omega_i} = 0$$

Revolute Joint

$$J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$$

$$J_{\omega_i} = z_{i-1}$$

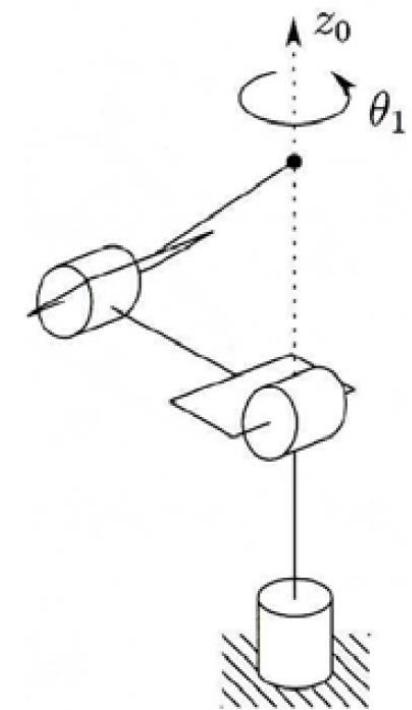
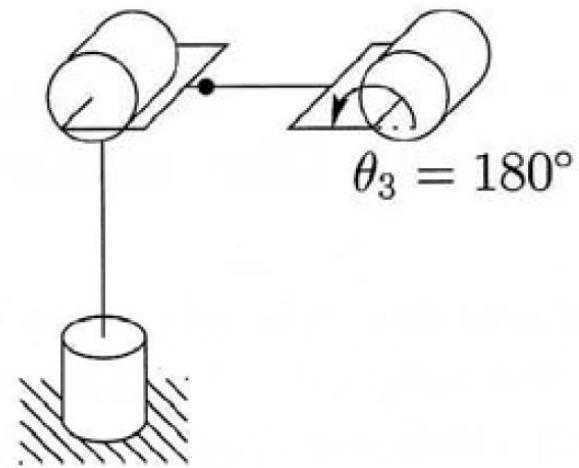
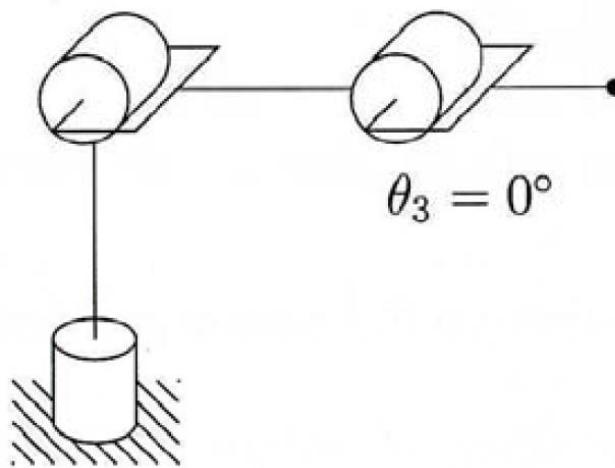


# Singularities

# Singularities

- Singular configuration

$$q^s = [q_1^s, q_2^s, \dots, q_n^s] \quad \text{for} \quad \text{rank}[J(q^s)] < \max_q \{ \text{rank}[J(q)] \}$$



# Singularity Implications

- Certain directions of motion may be unattainable
- Infinite joint velocity may correspond to bounded end-effector velocities
- Bounded joint torques may correspond to infinite end-effector torques/forces
- Often at the boundaries of the workspace
- Positions that may become unreachable by small perturbations of the robot link geometry

# Static force/torque relationships

- Joint torques/forces vector  $\tau$  can be found as

$$\tau = J^T(q)F$$

where  $F$  is a 6-component vector with forces/torques applied to the end-effector

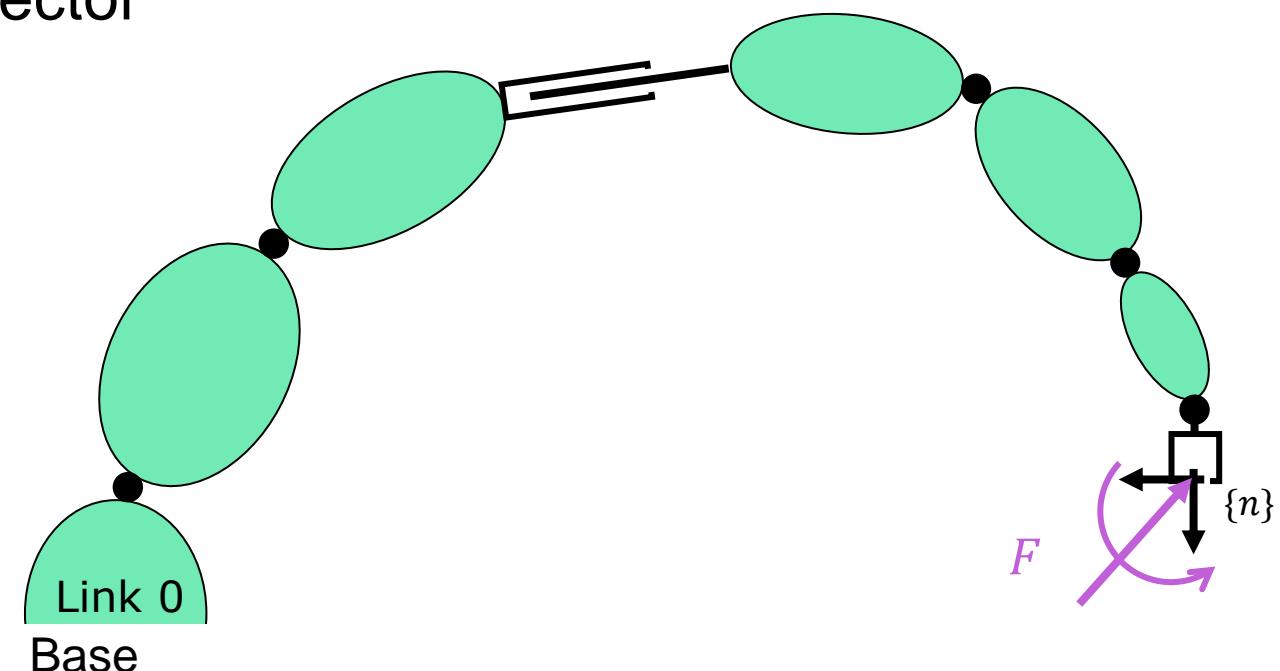
- Remember that

$$\xi = J(q)\dot{q}$$

- Work conjugate pairs

$$(\xi, F)$$

$$(\dot{q}, \tau)$$



# Inverse Velocity

# Inverse Velocity

- Inverse Jacobian
  - Pseudoinverse

$$A^+ = \begin{cases} A^T [AA^T]^{-1}, & m < n \\ A^{-1}, & m = n \\ [A^T A]^{-1} A^T, & m > n \end{cases}$$

→ overactuated arm  
→ fully actuated arm  
→ underactuated arm

- Inverse velocity

$$\dot{q} = J^+ \xi$$

in general

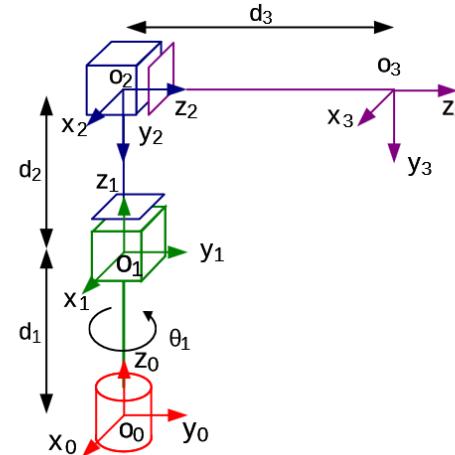
$$\dot{q} = J^+ \xi + (I_{n \times n} - J^+ J)b_{n \times 1}$$

# Exercises

# Exercises

## Problem 3

Find the  $6 \times 3$  Jacobian matrix corresponding to the end-effector frame 3 of the cylindrical manipulator shown below.



## Problem 1

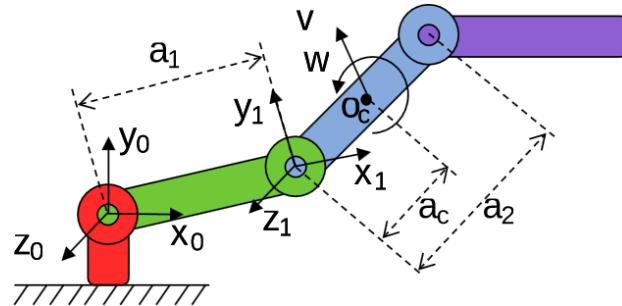
Two frames  $o_0x_0y_0z_0$  and  $o_1x_1y_1z_1$  are related by the time-independent homogeneous transformation

$$H_1^0 = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A particle has velocity  $v^1 = \{3, 1, 0\}^T$  relative to frame  $o_1x_1y_1z_1$ . What is the velocity of the particle in frame  $o_0x_0y_0z_0$ ?

## Problem 2

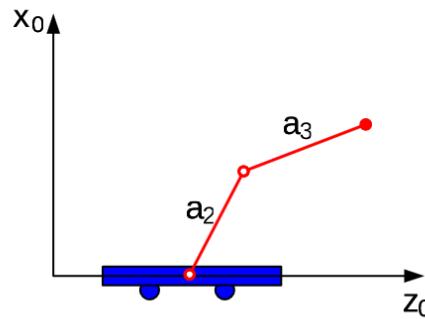
Consider the three-link planar manipulator shown below.



Compute the linear velocity  $v$  and angular velocity  $\omega$  at the center  $o_c$  of link 2, where  $a_c = a_2/2$ , as functions of the joint variables  $\{\theta_1, \theta_2\}$  and joint velocities  $\{\dot{\theta}_1, \dot{\theta}_2\}$ .

## Problem 4

The planar manipulator with a mobile platform moving in Z direction shown below can be interpreted as a PRR manipulator. The robot operates only in the  $XZ$  plane in the working space.



Define the coordinate frames 1 and 2 according to the Denavit-Hartenberg convention. Define frame 3 according to the usual convention for the end-effector, assuming that the grip opens and closes in the out of plane direction. Determine the transformation matrix  $T_3^0$ , and find the Jacobian matrix for this mobile robot.

Note: the end-effector frame does not follow the Denavit-Hartenberg convention, hence there is some ambiguity in how you define angle  $\theta_3$ .

Robotics – 34753

# Trajectory Planning & Manipulator Dynamics

Konstantinos Poulios

Associate Professor

Department of Civil and Mechanical Engineering  
DTU Lyngby, building 404 / room 124

# Trajectory Planning & Manipulator Dynamics – Lecture Overview

## 1. Repetition

## 2. Trajectory Planning

- Point to Point Motion
- Constant Jerk Curve

## 3. Manipulator Dynamics

- Euler-Lagrange Equations
- Kinetic Energy
- Potential Energy
- Generalized Forces

# Repetition

# Repetition

Rotation by means of a skew symmetric matrix:

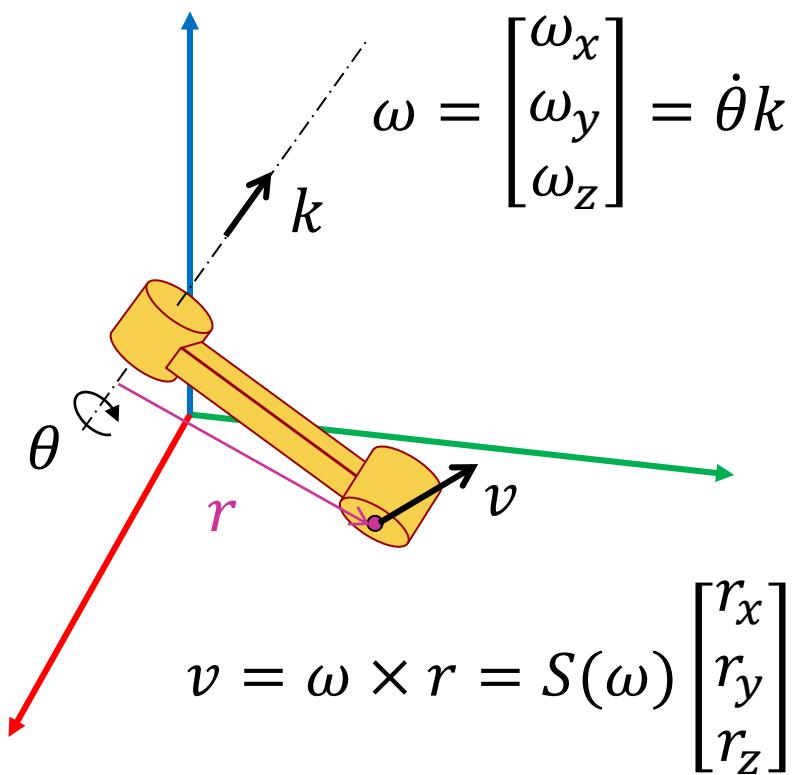
$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \rightarrow S(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Properties:

- $S + S^T = 0$
- $S(\alpha a + \beta b) = \alpha S(a) + \beta S(b)$
- $S(R a) = RS(a)R^T$ , for any orthonormal matrix  $R$
- $x^T S(a)x = 0$ , for any vector  $x$

Derivative of a rotation matrix:

$$\frac{dR}{d\theta} = S R(\theta) \quad \text{and} \quad \dot{R}_{k,\theta} = S(\dot{\theta} k) R_{k,\theta}$$



# Repetition

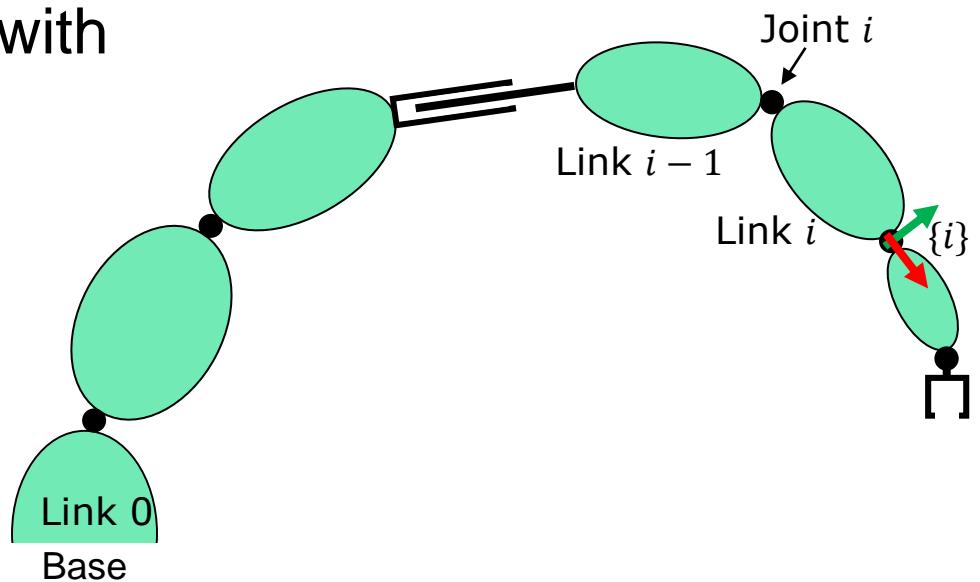
- Angular velocities (relative to the previous link and relative to the world frame)

$$\omega_{0,n}^0 = \omega_{0,1}^0 + R_1^0 \omega_{1,2}^1 + \cdots + R_{n-1}^0 \omega_{n-1,n}^{n-1}$$

$$\omega_{0,n}^0 = \omega_{0,1}^0 + \omega_{1,2}^0 + \cdots + \omega_{n-1,n}^0$$

- Linear velocities (of arbitrary points on a link with respect to the world frame)

$$\dot{p}^0 = S(\omega_i^0)R_i^0 p^i + \dot{o}_i^0$$



# Repetition

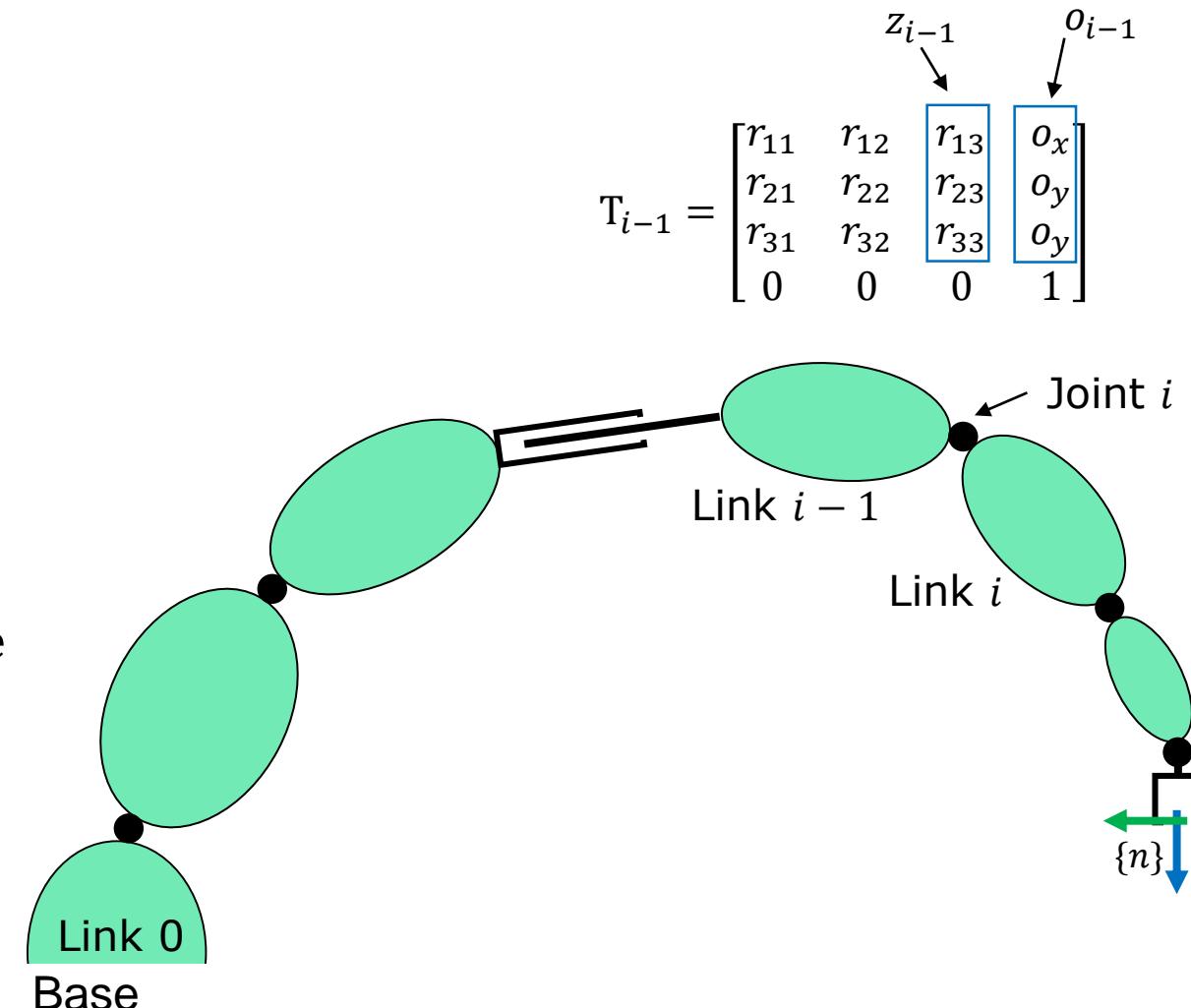
The **manipulator Jacobian** connects joint velocities to (end-effector) frame velocities:

$$\xi = \begin{bmatrix} v_n^0 \\ w_n^0 \end{bmatrix} = J \dot{q} = \begin{bmatrix} J_{11} & \cdots & J_{1n} \\ \vdots & \ddots & \vdots \\ J_{61} & \cdots & J_{6n} \end{bmatrix} \dot{q} = \begin{bmatrix} J_v(q(t)) \\ J_\omega(q(t)) \end{bmatrix} \dot{q}$$

Calculation of the manipulator Jacobian:

$$J = [J_1 \quad \cdots \quad J_i \quad \cdots \quad J_n]$$

where  $J_i = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}, & i \text{ prismatic} \\ \begin{bmatrix} z_{i-1} \times (o_n - o_{i-1}) \\ z_{i-1} \end{bmatrix}, & i \text{ revolute} \end{cases}$

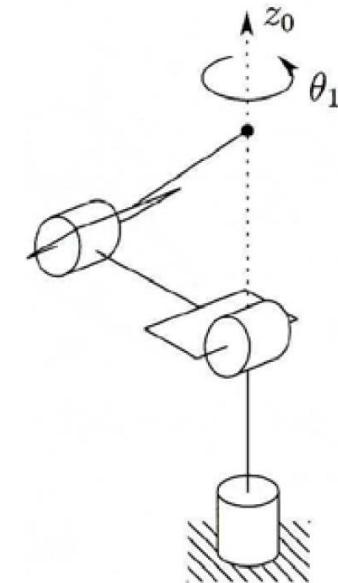
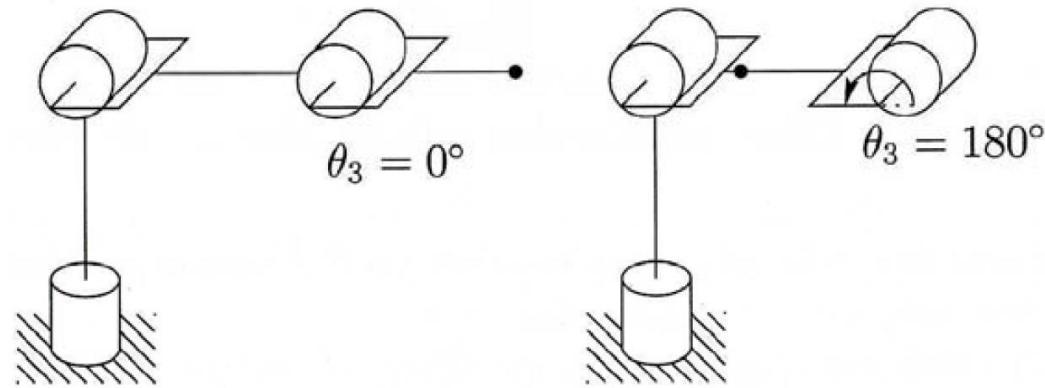


# Repetition

## Singularities

- Singular configuration

$$q^s = [q_1^s, q_2^s, \dots, q_n^s] \quad \text{for} \quad \text{rank}[J(q^s)] < \max_q \{ \text{rank}[J(q)] \}$$



- Inverse Jacobian
  - Pseudoinverse

$$A^+ = \begin{cases} A^T [AA^T]^{-1}, & m < n \\ A^{-1}, & m = n \\ [A^T A]^{-1} A^T, & m > n \end{cases}$$

# Repetition

Static force/torque relationships

- Joint torques/forces vector  $\tau$  can be found as

$$\tau = J^T(q)F$$

where  $F$  is a 6-component vector with forces/torques applied to the end-effector

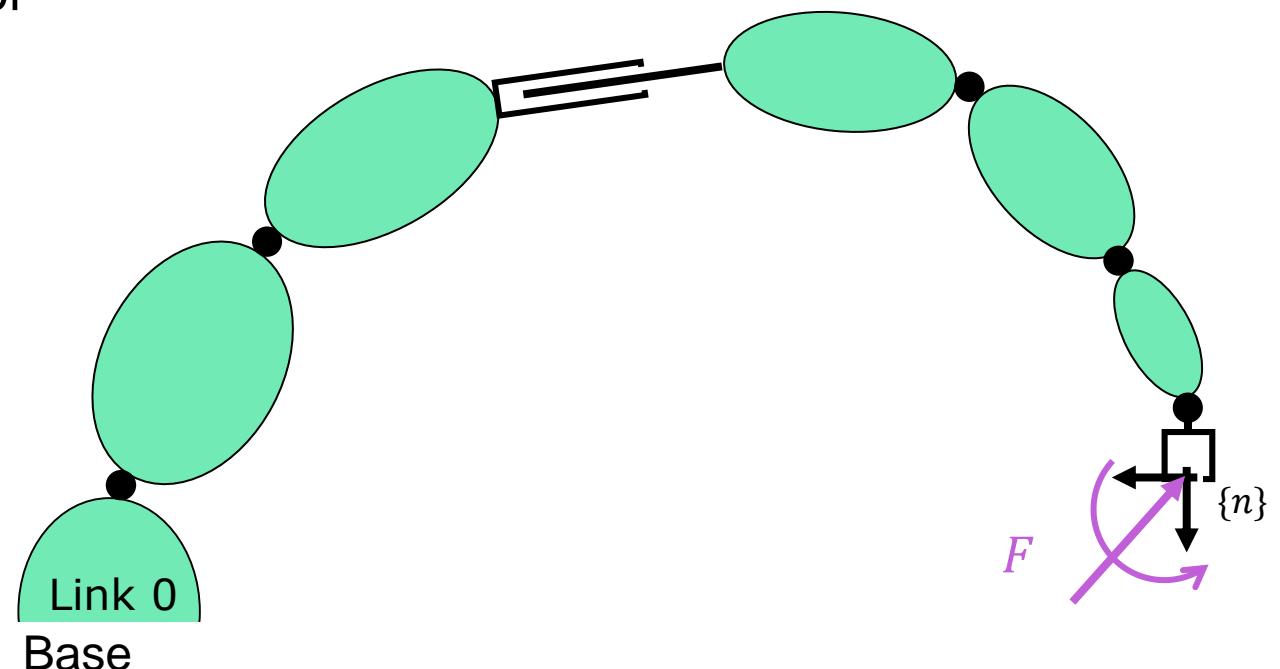
- Remember that

$$\xi = J(q)\dot{q}$$

- Work conjugate pairs

$$(\xi, F)$$

$$(\dot{q}, \tau)$$

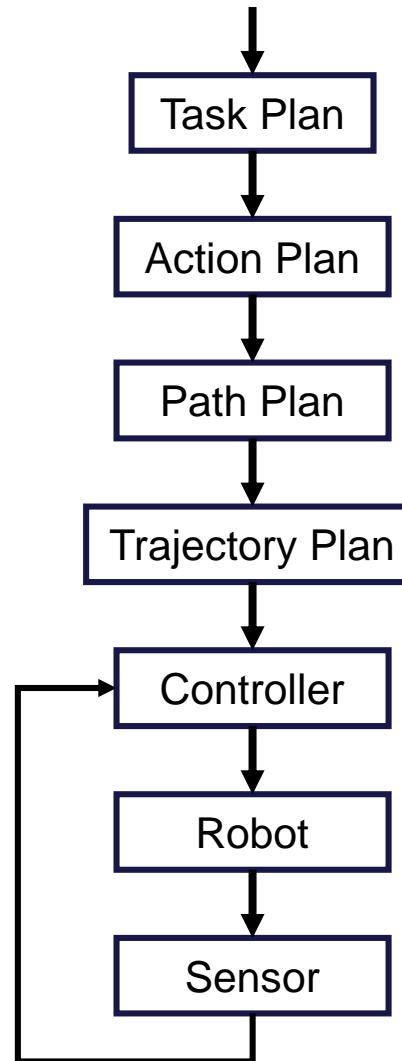


# Trajectory Planning

# Path versus Trajectory

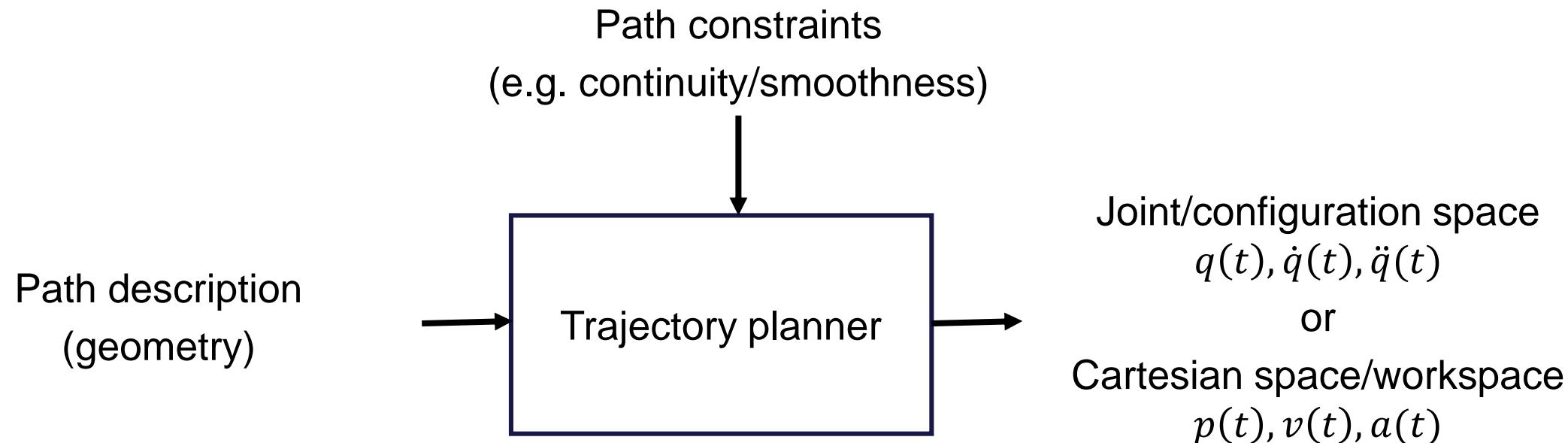
- **Path:** Purely geometric description of the route that the robot manipulator will follow
- **Trajectory:** Time-aware description of a followed path, defined as  $q(t) \rightarrow$  velocities/accelerations

# Robot Motion Planning



- **Path planning:**
  - Geometric path, e.g. point-wise in configuration space or in workspace
  - Avoidance of obstacles, shortest path ...
- **Trajectory planning:**
  - Interpolate or approximate the desired path from point to point by a class of **polynomial functions of time**
  - Satisfy position, velocity and acceleration constraints
  - Define  $q(t)$  piecewise to control the manipulator from the initial to the final configuration

# Trajectory Planning



# Trajectory Planning

- Decomposition of a path into segments with fast and slow velocity profiles

$$t_{in} \rightarrow t_f$$

$$t_{in}: q(t_{in}) = s_{in}, \quad \dot{q}(t_0) = v_{in}, \quad \ddot{q}(t_{in}) = a_{in}$$

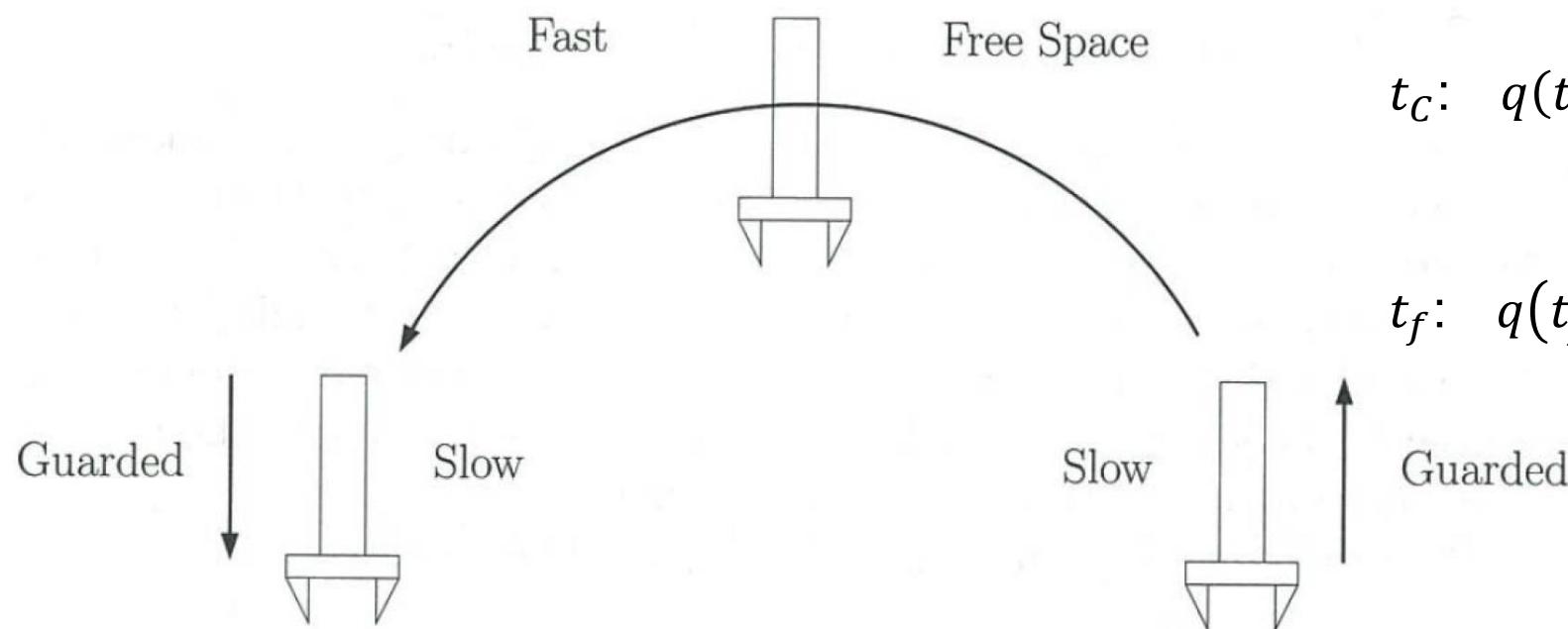
$$t_A: q(t_A) = s_A, \quad \dot{q}(t_A) = v_A, \quad \ddot{q}(t_A) = a_A$$

$$t_B: q(t_B) = s_B, \quad \dot{q}(t_B) = v_B, \quad \ddot{q}(t_B) = a_B$$

$$t_C: q(t_C) = s_C, \quad \dot{q}(t_C) = v_C, \quad \ddot{q}(t_C) = a_C$$

.....

$$t_f: q(t_f) = s_f, \quad \dot{q}(t_f) = v_f, \quad \ddot{q}(t_f) = a_f$$



# Trajectory Planning

- Interpolation between two points in configuration space

$$t_A \rightarrow t_B$$

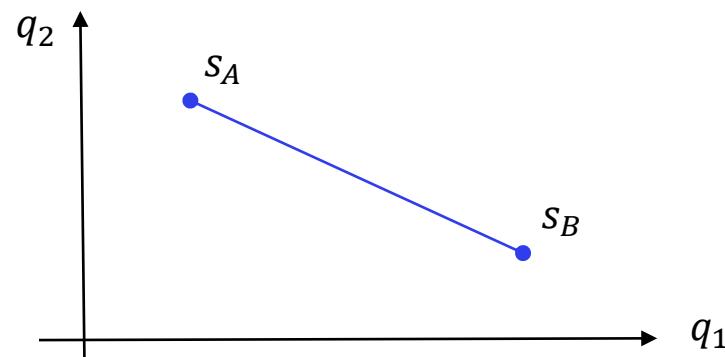
$$t_A: \quad q(t_A) = s_A, \quad \dot{q}(t_A) = v_A, \quad \ddot{q}(t_A) = a_A$$

$$t_B: \quad q(t_B) = s_B, \quad \dot{q}(t_B) = v_B, \quad \ddot{q}(t_B) = a_B$$

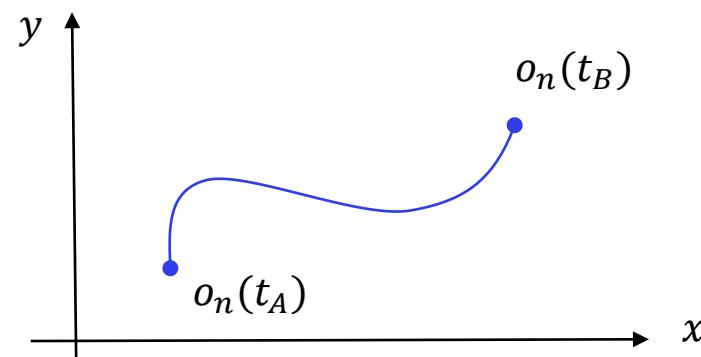
- E.g. linear interpolation

$$q(t) = \frac{t_B - t}{t_B - t_A} s_A + \frac{t - t_A}{t_B - t_A} s_B$$

Path in a 2D configuration space

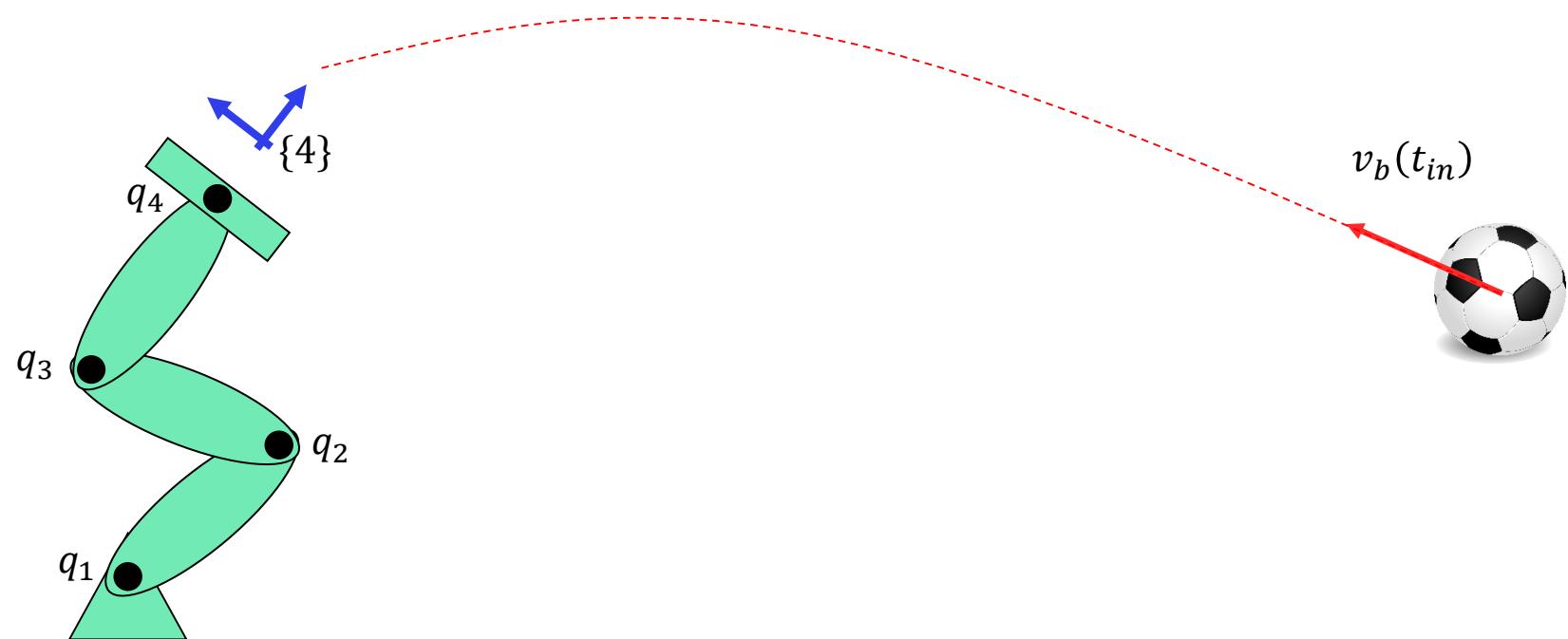


End effector path in workspace



# Trajectory Planning Example

- Soccer ball catcher



$$q(t_{in}) = \begin{bmatrix} 0.25\pi \\ 0.6\pi \\ -0.52\pi \\ 0 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} 0 & -1/\sqrt{2} & 1/\sqrt{2} & 0.3 \\ 0 & 1/\sqrt{2} & 1/\sqrt{2} & 1.2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

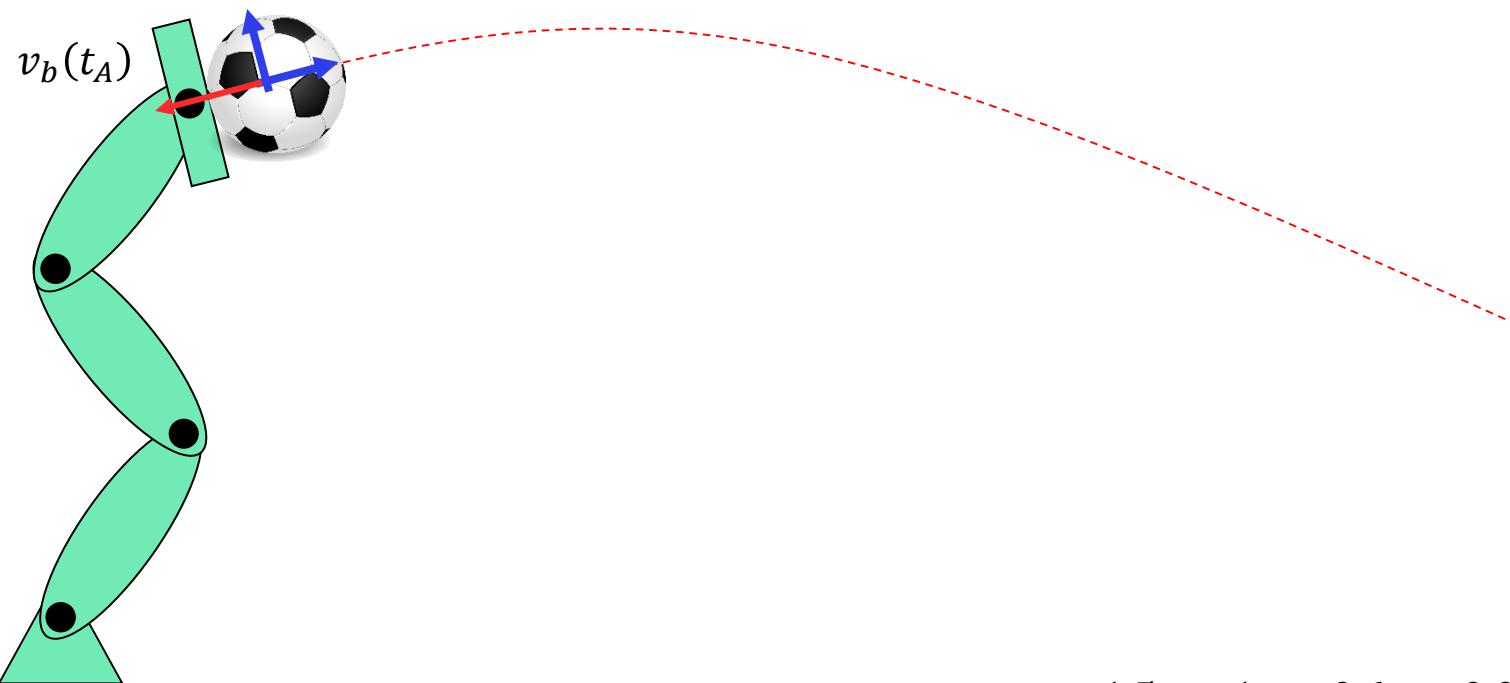
$$\dot{q} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} -1.2 & -0.9 & -0.7 & -0.1 \\ 0.3 & -0.02 & 0.4 & 0.1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{aligned} v_4^0 &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ \omega_4^0 &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

# Trajectory Planning Example

- Soccer ball catcher



$$q(t_A) = \begin{bmatrix} 0.33\pi \\ 0.4\pi \\ -0.4\pi \\ -0.2\pi \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} 0 & -0.15 & 0.989 & 0.4 \\ 0 & 0.989 & 0.15 & 1.5 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\dot{q} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix}$$

$$J_4 = \begin{bmatrix} -1.5 & -1 & -0.6 & -0.02 \\ 0.4 & 0.1 & 0.4 & 0.15 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{aligned} v_4^0 &= v_b(t_A) = \begin{bmatrix} -9.89 \\ -1.5 \\ 0 \end{bmatrix} \\ \omega_4^0 &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

# Trajectory Planning Example

$$\dot{q} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \end{bmatrix} \quad J_4 = \begin{bmatrix} -1.5 & -1 & -0.6 & -0.02 \\ 0.4 & 0.1 & 0.4 & 0.15 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad v_4^0 = v_b(t_A) = \begin{bmatrix} -9.89 \\ -1.5 \\ 0 \end{bmatrix}$$

$$\omega_4^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\dot{q} = J_4^T (J_4 J_4^T)^{-1} \begin{pmatrix} v_4^0 \\ \omega_4^0 \end{pmatrix} = \begin{bmatrix} -0.51 & 0.966 & 0 & 0 & 0 & -0.401 \\ -0.494 & -2.91 & 0 & 0 & 0 & 0.629 \\ 0.411 & 2.452 & 0 & 0 & 0 & -0.073 \\ 0.592 & -0.508 & 0 & 0 & 0 & 0.845 \end{bmatrix} \begin{bmatrix} -9.89 \\ -1.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3.596 \\ 9.246 \\ -7.746 \\ -5.096 \end{bmatrix}$$

Check that indeed

$$J_4 \dot{q} = \begin{bmatrix} -1.5 & -1 & -0.6 & -0.02 \\ 0.4 & 0.1 & 0.4 & 0.15 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3.596 \\ 9.246 \\ -7.746 \\ -5.096 \end{bmatrix} = \begin{bmatrix} -9.89 \\ -1.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

# Trajectory Planning – Interpolation Functions

In the previous example, we know the state at time  $t_{in}$ :

$$q(t_{in}) = \begin{bmatrix} 0.25\pi \\ 0.6\pi \\ -0.52\pi \\ 0 \end{bmatrix} \quad \text{and} \quad \dot{q}(t_{in}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

and then, at the second time instant  $t_A$  we need to reach a state with

$$q(t_A) = \begin{bmatrix} 0.33\pi \\ 0.4\pi \\ -0.4\pi \\ -0.2\pi \end{bmatrix} \quad \text{and} \quad \dot{q}(t_A) = \begin{bmatrix} 3.596 \\ 9.246 \\ -7.746 \\ -5.096 \end{bmatrix}$$

How do we control the joint variables  $q_1, q_2, q_3$  and  $q_4$  from  $t_{in}$  to  $t_A$ ?

Naive linear interpolation?

$$q_i(t) = \frac{t_A-t}{t_A-t_{in}} q_i(t_{in}) + \frac{t-t_{in}}{t_A-t_{in}} q_i(t_A) = c_0 + c_1 t$$

# Trajectory Planning – Interpolation Functions

Cubic interpolation:

$$q(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$\Rightarrow \dot{q}(t) = c_1 + 2c_2 t + 3c_3 t^2$$

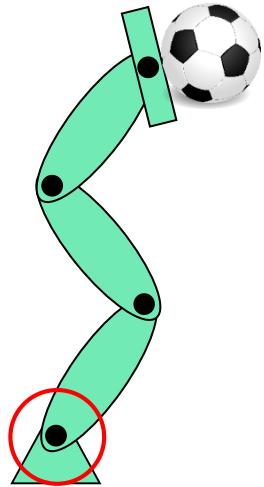
can satisfy two initial and two final conditions:

$$\begin{aligned} q(t_{in}) &= q_{in} & \text{and} & \quad \dot{q}(t_{in}) = v_{in} \\ q(t_A) &= q_A & \text{and} & \quad \dot{q}(t_A) = v_A \end{aligned}$$

These four conditions can be solved for  $c_0$ ,  $c_1$ ,  $c_2$  and  $c_3$ :

$$\begin{bmatrix} 1 & t_{in} & t_{in}^2 & t_{in}^3 \\ 0 & 1 & 2t_{in} & 3t_{in}^2 \\ 1 & t_A & t_A^2 & t_A^3 \\ 0 & 1 & 2t_A & 3t_A^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} q_{in} \\ v_{in} \\ q_A \\ v_A \end{bmatrix}$$

# Trajectory Planning – Example



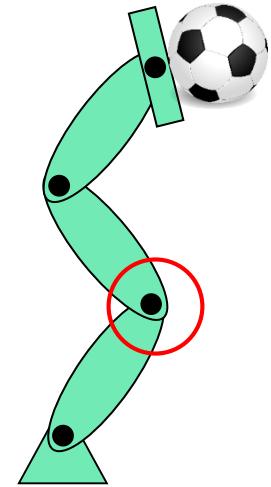
Assume  $t_{in} = 0$  and  $t_A = 1$ , then:

$$q(0) = \begin{bmatrix} 0.25\pi \\ 0.6\pi \\ -0.52\pi \\ 0 \end{bmatrix}, \dot{q}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, q(1) = \begin{bmatrix} 0.33\pi \\ 0.4\pi \\ -0.4\pi \\ -0.2\pi \end{bmatrix}, \dot{q}(1) = \begin{bmatrix} 3.60 \\ 9.25 \\ -7.75 \\ -5.10 \end{bmatrix}$$

For  $i = 1$ :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0.25\pi \\ 0 \\ 0.33\pi \\ 3.60 \end{bmatrix} \Rightarrow \begin{array}{l} c_0 = 0.785 \\ c_1 = 0 \\ c_2 = -2.842 \\ c_3 = 3.093 \end{array}$$

# Trajectory Planning – Example



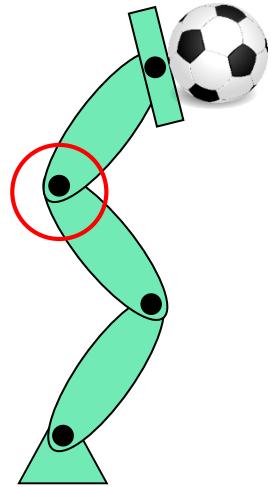
Assume  $t_{in} = 0$  and  $t_A = 1$ , then:

$$q(0) = \begin{bmatrix} 0.25\pi \\ 0.6\pi \\ -0.52\pi \\ 0 \end{bmatrix}, \dot{q}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, q(1) = \begin{bmatrix} 0.33\pi \\ 0.4\pi \\ -0.4\pi \\ -0.2\pi \end{bmatrix}, \dot{q}(1) = \begin{bmatrix} 3.60 \\ 9.25 \\ -7.75 \\ -5.10 \end{bmatrix}$$

For  $i = 2$ :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0.6\pi \\ 0 \\ 0.4\pi \\ 9.25 \end{bmatrix} \Rightarrow \begin{aligned} c_0 &= 1.885 \\ c_1 &= 0 \\ c_2 &= -11.131 \\ c_3 &= 10.503 \end{aligned}$$

# Trajectory Planning – Example



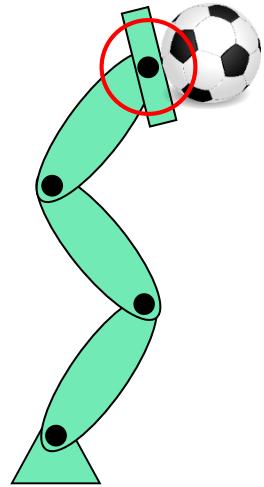
Assume  $t_{in} = 0$  and  $t_A = 1$ , then:

$$q(0) = \begin{bmatrix} 0.25\pi \\ 0.6\pi \\ -0.52\pi \\ 0 \end{bmatrix}, \dot{q}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, q(1) = \begin{bmatrix} 0.33\pi \\ 0.4\pi \\ -0.4\pi \\ -0.2\pi \end{bmatrix}, \dot{q}(1) = \begin{bmatrix} 3.60 \\ 9.25 \\ -7.75 \\ -5.10 \end{bmatrix}$$

For  $i = 3$ :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -0.52\pi \\ 0 \\ -0.4\pi \\ -7.75 \end{bmatrix} \Rightarrow \begin{aligned} c_0 &= -1.634 \\ c_1 &= 0 \\ c_2 &= 8.877 \\ c_3 &= -8.500 \end{aligned}$$

# Trajectory Planning – Example



Assume  $t_{in} = 0$  and  $t_A = 1$ , then:

$$q(0) = \begin{bmatrix} 0.25\pi \\ 0.6\pi \\ -0.52\pi \\ 0 \end{bmatrix}, \dot{q}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, q(1) = \begin{bmatrix} 0.33\pi \\ 0.4\pi \\ -0.4\pi \\ -0.2\pi \end{bmatrix}, \dot{q}(1) = \begin{bmatrix} 3.60 \\ 9.25 \\ -7.75 \\ -5.10 \end{bmatrix}$$

For  $i = 4$ :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0.2\pi \\ -5.10 \end{bmatrix} \Rightarrow \begin{array}{l} c_0 = 0 \\ c_1 = 0 \\ c_2 = 3.211 \\ c_3 = -3.839 \end{array}$$

# Trajectory Planning – Example

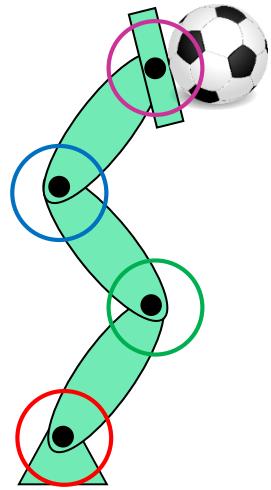
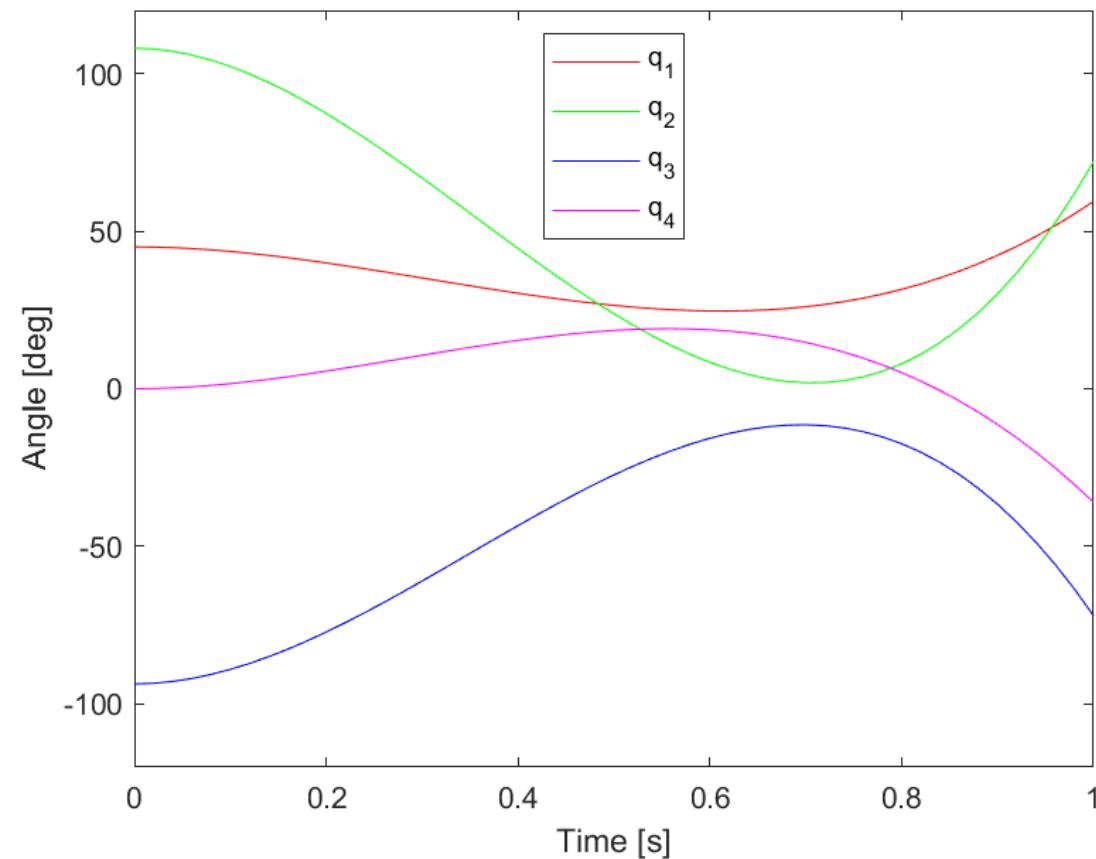
Hence, the interpolation functions for the four joints between  $t_{in} = 0$  and  $t_A = 1$  sec are:

$$q_1 = 0.785 - 2.842 t^2 + 3.093 t^3$$

$$q_2 = 1.885 - 11.131 t^2 + 10.503 t^3$$

$$q_3 = -1.634 + 8.877 t^2 - 8.5 t^3$$

$$q_4 = 3.211 t^2 - 3.839 t^3$$



# Trajectory Planning – Interpolation Functions

Quintic interpolation:

$$\begin{aligned} q(t) &= c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \\ \Rightarrow \dot{q}(t) &= c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 \\ \Rightarrow \ddot{q}(t) &= 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 \end{aligned}$$

can satisfy three initial and three final conditions:

$$\begin{array}{lll} q(t_{in}) = q_{in} & \& \dot{q}(t_{in}) = v_{in} & \& \ddot{q}(t_{in}) = a_{in} \\ q(t_A) = q_A & \& \dot{q}(t_A) = v_A & \& \ddot{q}(t_A) = a_A \end{array}$$

These six conditions can be solved for  $c_0, c_1, c_2, c_3, c_4$  and  $c_5$ :

$$\left[ \begin{array}{cccccc} 1 & t_{in} & t_{in}^2 & t_{in}^3 & t_{in}^4 & t_{in}^5 \\ 0 & 1 & 2t_{in} & 3t_{in}^2 & 4t_{in}^3 & 5t_{in}^4 \\ 0 & 0 & 2 & 6t_{in} & 12t_{in}^2 & 20t_{in}^3 \\ 1 & t_A & t_A^2 & t_A^3 & t_A^4 & t_A^5 \\ 0 & 1 & 2t_A & 3t_A^2 & 4t_A^3 & 5t_A^4 \\ 0 & 0 & 2 & 6t_A & 12t_A^2 & 20t_A^3 \end{array} \right] \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} q_{in} \\ v_{in} \\ a_{in} \\ q_A \\ v_A \\ a_A \end{bmatrix}$$

# Manipulator Dynamics

# Kinetic and Potential Energy

- Kinematics < Statics < **Dynamics**
- **Rigid bodies** < Flexible couplings < Deformable bodies
- Kinetic energy of rigid bodies:

$$\mathcal{K} = \sum_{i=1}^n \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} \omega_i^T J_i \omega_i$$

$$\mathcal{K} = \frac{1}{2} \dot{q}^T D(q) \dot{q}$$

- Potential energy of rigid bodies:

$$\mathcal{P} = - \sum_{i=1}^n m_i g^T r_{c_i}$$

# Inertia Tensor

- Kinetic energy of rigid bodies:

$$\mathcal{K} = \sum_{i=1}^n \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} \omega_i^T \mathcal{J}_i \omega_i$$

- Inertia tensor in global frame (inertial frame):

$$\begin{aligned}\mathcal{J}_i &= R_i(q_1, q_2, \dots) I_i R_i^T(q_1, q_2, \dots) \\ \mathcal{J}_i &= R_i I_i R_i^T\end{aligned}$$

- Inertia tensor in local frame:

$$I_i = \begin{bmatrix} I_{xx} & I_{xy} & I_{zx} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{zx} & I_{yz} & I_{zz} \end{bmatrix}$$

e.g. for a  $a \times b \times c$  box:  $I_{xx} = \frac{m}{12} (b^2 + c^2)$ ,  $I_{yy} = \frac{m}{12} (a^2 + c^2)$ ,  $I_{zz} = \frac{m}{12} (a^2 + b^2)$

# Kinetic and Potential Energy

- Location of the center of gravity of link  $i$  expressed in the world frame:

$$\mathbf{r}_{c_i}$$

- Inertia tensor (3x3) of link  $i$  w.r.t its center of mass  $c_i$  and axes parallel to frame  $i$ :

$$\mathbf{I}_i$$

- Inertia tensor (3x3) of link  $i$  w.r.t. its center of mass  $c_i$  and axes parallel to the world frame axes:

$$\mathcal{J}_i = \mathbf{R}_i \mathbf{I}_i \mathbf{R}_i^T$$

- Jacobian of the center of mass  $c_i$  of link  $i$ :

$$\mathbf{J}_{v,c_i} \quad \text{and} \quad \mathbf{J}_{\omega_i}$$

- Translational and rotational velocities:

$$\mathbf{v}_{c_i} = \mathbf{J}_{v,c_i} \dot{\mathbf{q}}$$

$$\boldsymbol{\omega}_i = \mathbf{J}_{\omega_i} \dot{\mathbf{q}}$$

# Kinetic and Potential Energy

- The inertia matrix of the robotic manipulator ( $n$  by  $n$ ):

$$D(q) = \sum_{i=1}^n m_i J_{v,c_i}^T J_{v,c_i} + J_{\omega_i}^T R_i I_i R_i^T J_{\omega_i}$$

- The kinetic energy of the manipulator is then written in compact form:

$$\mathcal{K} = \frac{1}{2} \dot{q}^T D(q) \dot{q}$$

- The potential energy is also known as

$$\mathcal{P} = - \sum_{i=1}^n m_i \mathbf{g}^T r_{c_i}(q)$$

# Euler-Lagrange Equations of Motion

- The Lagrangian function:

$$\mathcal{L}(q, \dot{q}) = \mathcal{K} - \mathcal{P} = \frac{1}{2} \dot{q}^T D(q) \dot{q} - \mathcal{P}(q) = \sum_{i,j} \frac{1}{2} d_{ij}(q) \dot{q}_i \dot{q}_j + \sum_i m_i g^T r_{c_i}(q)$$

- Equations of motions for  $k = 1, \dots, n$ :

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k$$

$$\sum_{j=1}^n d_{kj}(q) \ddot{q}_j + \sum_{i,j} c_{ijk}(q) \dot{q}_i \dot{q}_j + g_k(q) = \tau_k$$

where

$$c_{ijk}(q) = \frac{1}{2} \left( \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right)$$

- $\dot{q}_i^2$  terms express centrifugal effects
- $\dot{q}_i \dot{q}_j$  terms express Coriolis effects

# Equations of Motion in Matrix Form

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau$$

Where:

$$D(q) = \sum_{i=1}^n m_i J_{v,c_i}^T J_{v,c_i} + J_{\omega_i}^T R_i I_i R_i^T J_{\omega_i}$$

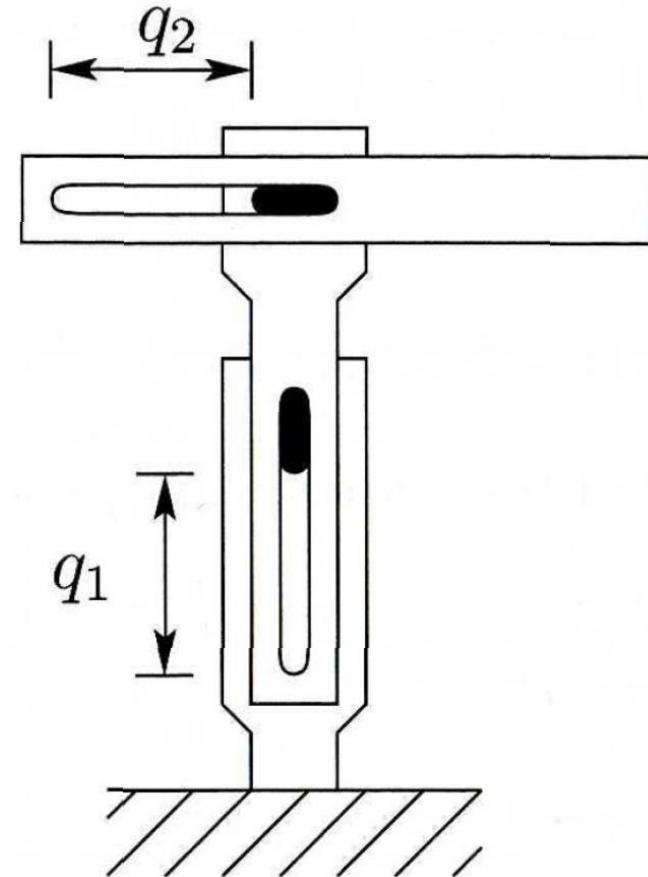
$$[C(q, \dot{q})]_{kj} = \sum_{i=1}^n c_{ijk}(q) \dot{q}_i = \sum_{i=1}^n \frac{1}{2} \left( \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right) \dot{q}_i$$

$$g = \begin{bmatrix} g_1(q) \\ \dots \\ g_n(q) \end{bmatrix} \text{ where } g_k = \frac{\partial \mathcal{P}}{\partial q_k}$$

# Example

Two-link Cartesian manipulator:

- Forward kinematics?
- Manipulator Jacobian?
- Kinetic and potential energies?
- Euler-Lagrange equation?



# Example

Two-link Cartesian manipulator:

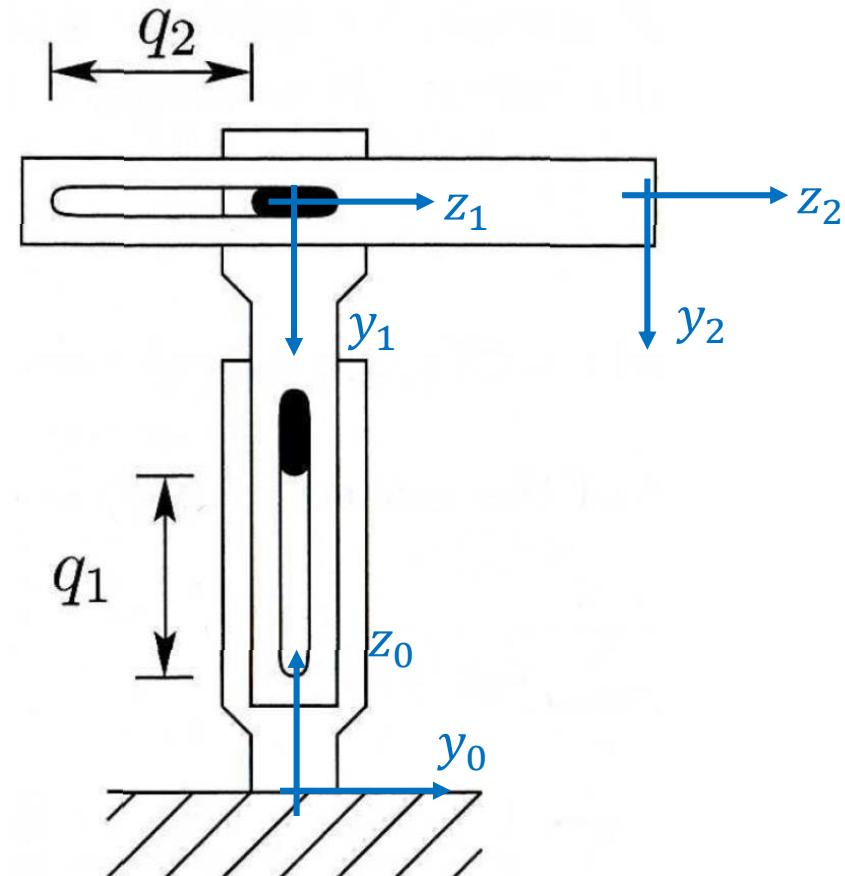
- Denavit-Hartenberg parameters

Link	$\theta$	$d$	$a$	$\alpha$
1	0	$q_1$	0	-90
2	0	$q_2$	0	0

- Jacobian matrices for centers of mass

$$J_{v,c_1} = [z_0^0, 0] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$J_{v,c_2} = [z_0^0, z_1^0] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$



# Example

Two-link Cartesian manipulator:

- Linear velocity of centers of mass

$$v_{c_1} = J_{v,c_1} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

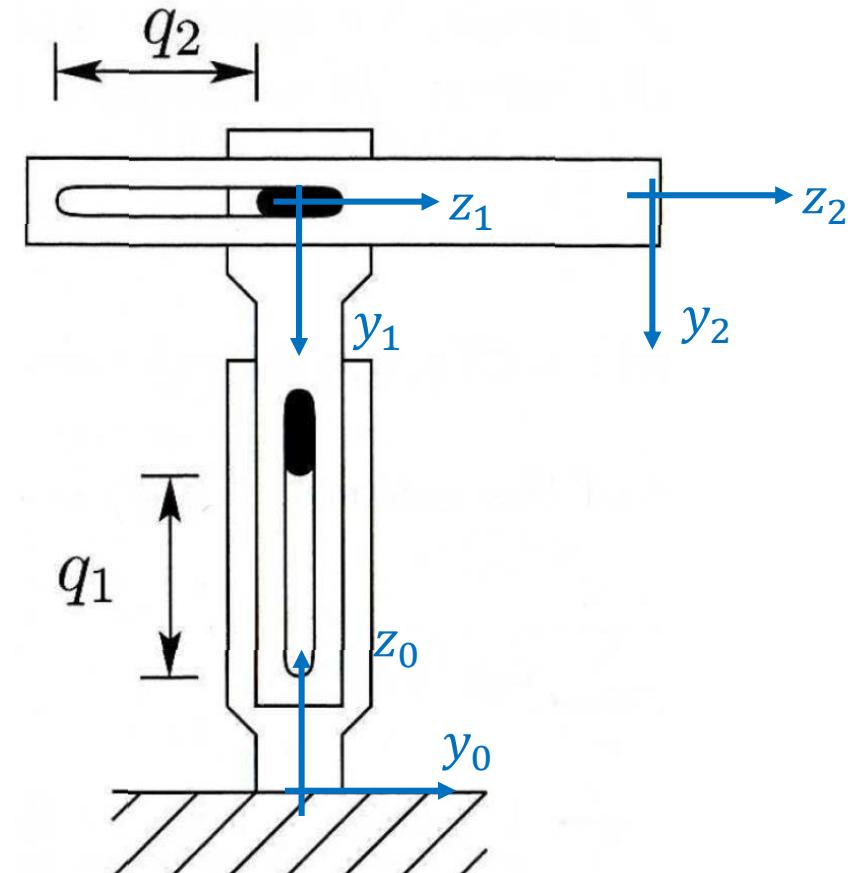
$$v_{c_2} = J_{v,c_2} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

- Kinetic energy

$$\mathcal{K} = \frac{1}{2} \dot{q}^T (m_1 J_{v,c_1}^T J_{v,c_1} + m_2 J_{v,c_2}^T J_{v,c_2}) \dot{q}$$

$$\mathcal{K} = \frac{1}{2} [\dot{q}_1 \quad \dot{q}_2] \underbrace{\begin{bmatrix} m_1 + m_2 & 0 \\ 0 & m_2 \end{bmatrix}}_D \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

*D*



# Example

Two-link Cartesian manipulator:

- Potential energy

$$\mathcal{P} = g(m_1 + m_2)q_1$$

- Gravity vector

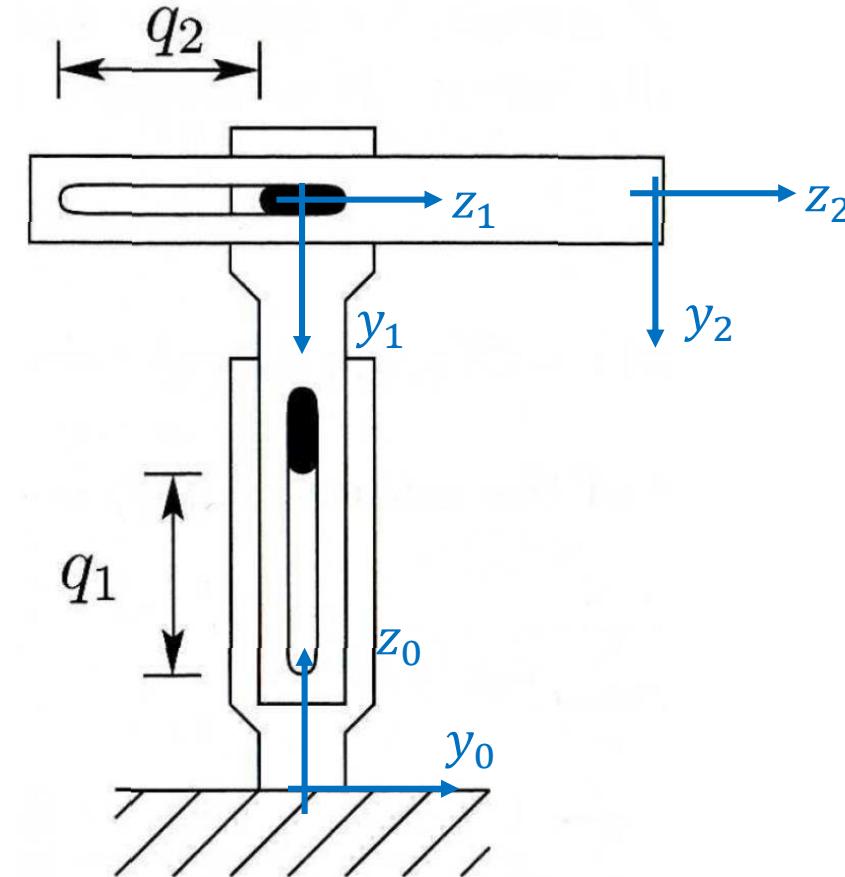
$$g_1 = \frac{\partial \mathcal{P}}{\partial q_1} = g(m_1 + m_2)$$

$$g_2 = \frac{\partial \mathcal{P}}{\partial q_2} = 0$$

- Christoffel symbols

$$c_{ijk} = 0$$

because inertia matrix  $D$  is constant and therefore independent of  $q$ .



# Example

Two-link Cartesian manipulator:

- Euler-Lagrange equation

$$\sum_{j=1}^n d_{kj}(q)\ddot{q}_j + \sum_{i,j} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k$$

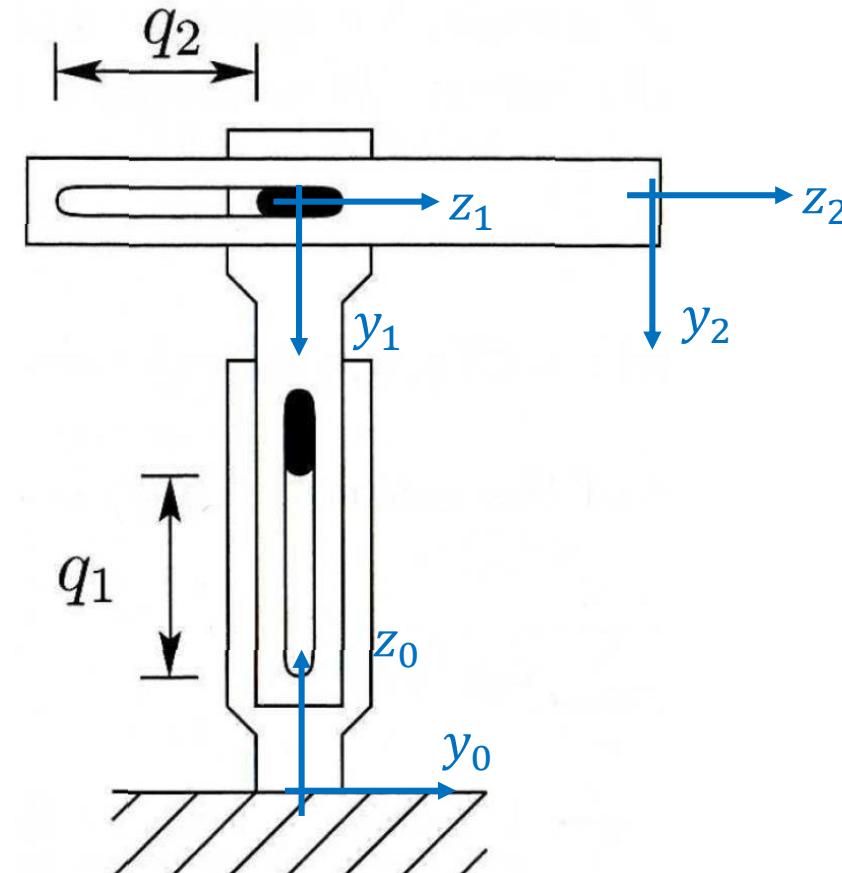
with

$$d_{11} = m_1 + m_2 \quad d_{22} = m_2 \quad d_{12} = d_{21} = 0$$

$$g_1 = g(m_1 + m_2) \quad g_2 = 0$$

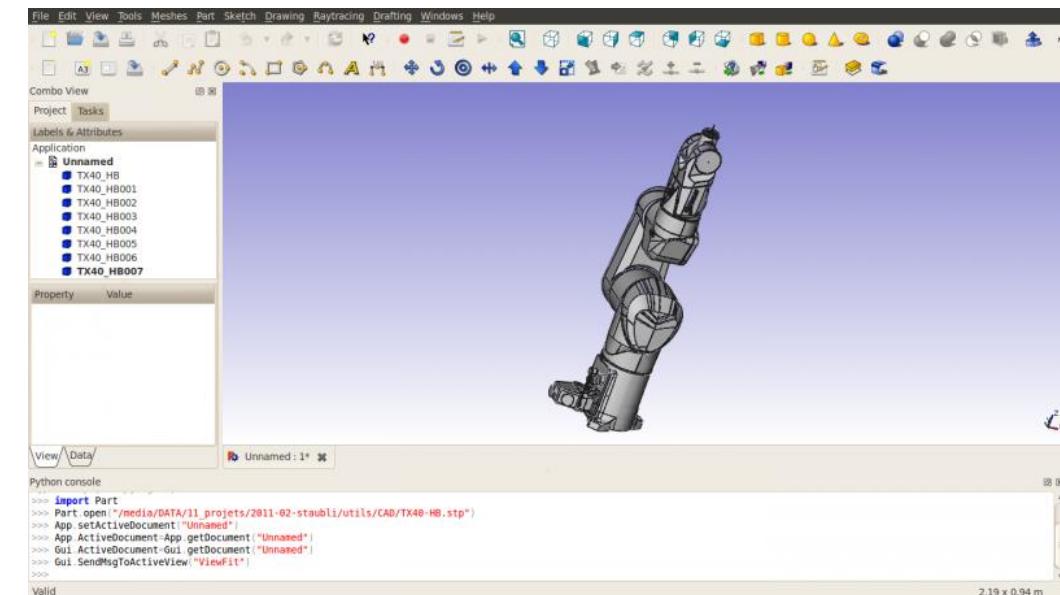


$$(m_1 + m_2)\ddot{q}_1 + g(m_1 + m_2) = f_1$$
$$m_2\ddot{q}_2 = f_2$$



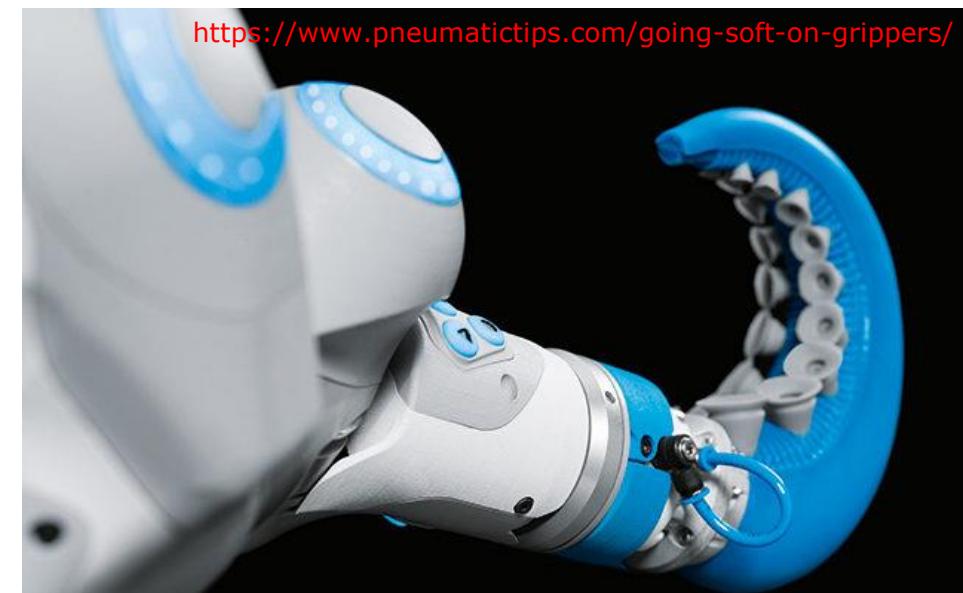
# Student Projects

- Programming of a general serial link robotic manipulator module for FreeCAD (open source)



<https://www.pneumatictips.com/going-soft-on-grippers/>

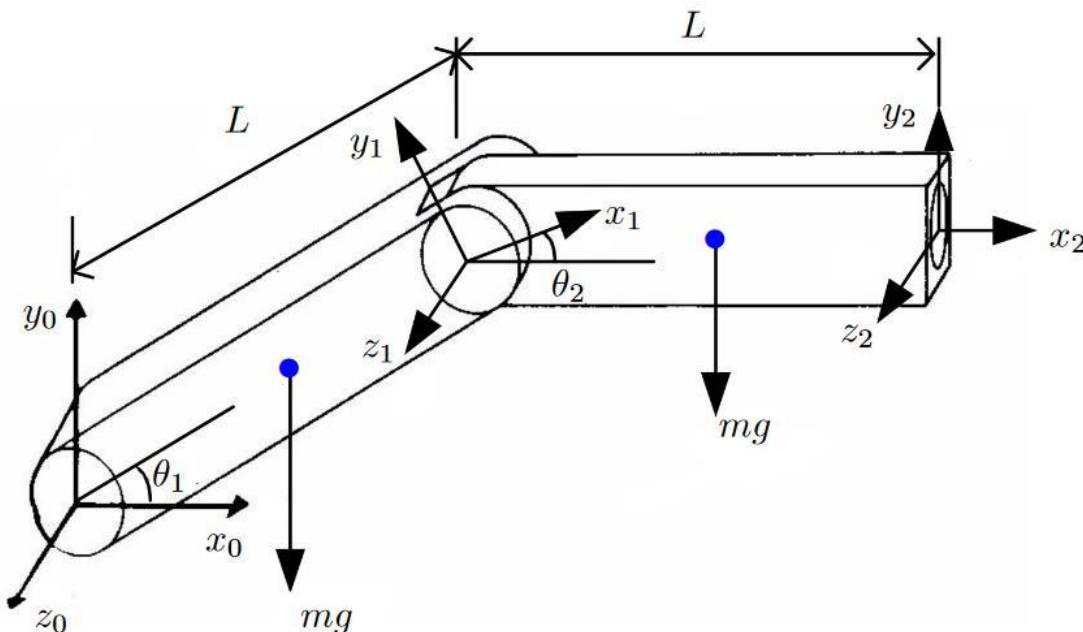
- Soft robotics design using computational tools (continuum mechanics + optimization)



# Exercise

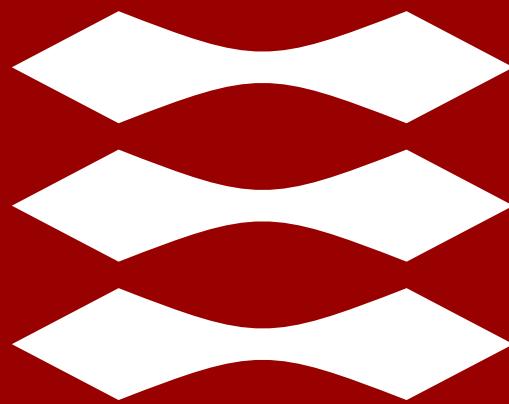
## Problem 1

Consider a manipulator of two revolute joints, as shown in the figure below.



Given that the length of each link is  $L$  and the mass of each link is  $m$ , derive the dynamic model of this two-link robot arm using the Euler-Lagrange method. Approximate the geometry of the two arms as parallelepipeds of dimensions  $L \times 0.1L \times 0.1L$  and assume that their mass is uniformly distributed.

**DTU**



Robotics - 34753

# Recap on linear control

Contributors:

Niels Axel Andersen

Hayan Wu

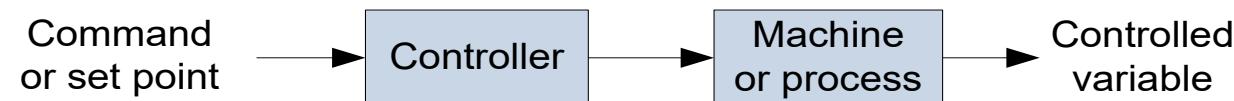
**Matteo Fumagalli**  
Associate Professor  
Automation and Control Group  
Department of Electrical Engineering  
DTU Lyngby, Building 326

# Outline

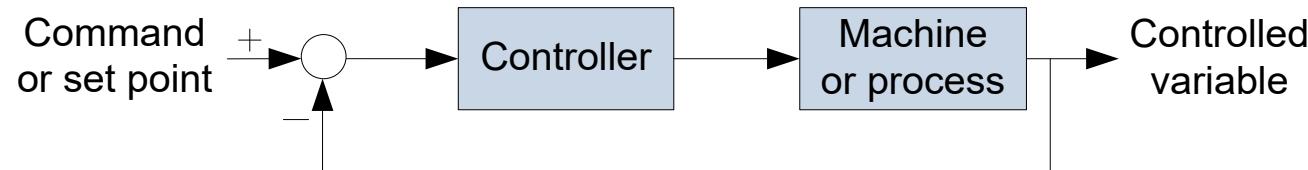
- Feedback control system?
- The Laplace transformation?
- Transfer function?
- System block diagram?
- First/second order systems?
- Basic controllers?

# Open loop and closed loop control systems

- An open-loop control system utilizes an actuating device to control the process directly without using feedback.



- A closed-loop control system uses a measurement of the output and feedback of this signal to compare it with the desired output (reference or command).



# Why do we need feedback?

- + Most systems are subject to disturbances that perturb the controlled variable. A major reason for the use of feedback is to **reduce** the effect of these **disturbances**.
- + The closed-loop system is more **accurate** than the open-loop system. It can be designed to provide extreme accuracy in the steady state.
- + Its **response time** can be adjust by appropriate design
- **Stability** problem, the controlled system may fluctuate continuously at some periodic rate and never reaches the desired steady-state condition

# How to develop a feedback control system?

- Choose a way to adjust the variable to be controlled, e.g. the mechanical load will be positioned with an electric motor
- Select suitable sensors to complete the loop
- Determine what is required for the system to operate, e.g. accuracy in steady state and response time
- System stability analysis
- Modify the system regarding stability and other desired operating conditions

# Tools for linear system analysis and design

- **Modeling**: mathematical description of the system
  - the differential equations of the system
- **Laplace transformation**: convert the differential equations to algebraic equations, and convert from the time domain to frequency domain
- **System analysis** based on e.g. Nyquist stability criterion or on the root locus method
- Analyse open loop system to **predict** the behavior of closed loop system

# Laplace Transform

Definition: The Laplace Transform of a signal  $f(t)$ ,  $t \geq 0$   
is the function of the complex variable  $s \in \mathbb{C}$

$$F(s) := \mathcal{L}[f(t)](s) = \int_0^{+\infty} f(t)e^{-st}dt$$

# Laplace Transform

Typical Laplace Transform of canonical functions

$f(t)$	$\mathcal{L}(s)$
$imp(t)$	1
$step(t)$	$\frac{1}{s}$
$cos(\omega t)$	$\frac{s}{s^2 + \omega^2}$
$sin(\omega t)$	$\frac{\omega}{s^2 + \omega^2}$
$e^{at}$	$\frac{1}{s - a}$

# Laplace Transform

Properties of the Laplace Transform:

LINEARITY

$$\mathcal{L}[\alpha_1 f_1(t) + \alpha_2 f_2(t)](s) = \alpha_1 \mathcal{L}[f_1(t)](s) + \alpha_2 \mathcal{L}[f_2(t)](s)$$

DERIVATION OVER TIME

$$\mathcal{L}[\dot{f}_1(t)](s) = s\mathcal{L}[f_1(t)](s) - f_1(0)$$

The properties allow us to compare differential and linear relations in time domain, to the equivalent in Laplace domain

# Laplace Transform

## Inverse Laplace transform

Consider  $F=N/D$ , with  $d>n$

$$F(s) = \frac{N(s)}{D(s)} = \frac{n_0 + n_1s + n_2s^2 + \dots + n_ns^n}{d_0 + d_1s + d_2s^2 + \dots + d.ds^d}$$

it is possible to define the inverse transform function

$$f(t) = \mathcal{L}^{-1}[F(s)](t) , \quad t \geq 0$$

it can be determined using the Heaviside method

# Laplace Transform

Consider the Laplace transform of a function  $f(t)$  with  $d > n$  (Strictly proper)

It is possible to calculate the value of  $f(t)$  in  $t = 0$  and the limit of  $f(t)$ , for  $t \rightarrow \infty$ , by using the laplace transform of  $f(t)$ , WITHOUT calculating  $\mathcal{L}^{-1}[F(s)](t)$  explicitly

## THEOREM OF THE INITIAL VALUE

$$f(0) = \lim_{s \rightarrow \infty} sF(s)$$

And if the roots of  $D(s)$  are equal to 0 or they have real part greater than 0:

## THEOREM OF THE FINAL VALUE

$$f(\infty) = \lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s)$$

# Transfer Function

A function  $G(s)$  that describes the input-output relation of a linear system subject to an input  $U(s)$  :

$$Y(s) = G(s)U(s)$$

NOTE: a TF allows to describe differential equations as algebraic ones, and it allows to evaluate the motion of the output  $y(t)$ ,  $t \geq 0$ , due to an input  $u(t)$ ,  $t \geq 0$  as:

$$u(t), t \geq 0 \xrightarrow{\mathcal{L}} U(s) \xrightarrow{\cdot G(s)} Y(s) \xrightarrow{\mathcal{L}^{-1}} y(t)$$

# Transfer Function

$$G(s) = C(sI - A)^{-1}B + D = \frac{N(s)}{D(s)}$$

the roots of  $D(s)$  are called POLES; the roots of  $N(s)$  are called ZEROS

Poles and zeros are singularities of  $G(s)$

Poles are the eigenvalues of the state matrix  $A$ . This means that the number of poles of a transfer function of a system of the n-th order is n

$$G(s) = \frac{\mu}{s^g} \frac{\prod_i (1 + T_i s) \prod_i \left(1 + 2\frac{\zeta_i}{\sigma_{ni}}s + \frac{s^2}{\sigma_{ni}^2}\right)}{\prod_i (1 + \tau_i s) \prod_i \left(1 + 2\frac{\xi_i}{\omega_{ni}}s + \frac{s^2}{\omega_{ni}^2}\right)}$$

# Transfer Function

$$G(s) = \frac{\mu}{s^g} \frac{\prod_i (1 + T_i s) \prod_i \left(1 + 2\frac{\zeta_i}{\sigma_{ni}} s + \frac{s^2}{\sigma_{ni}^2}\right)}{\prod_i (1 + \tau_i s) \prod_i \left(1 + 2\frac{\xi_i}{\omega_{ni}} s + \frac{s^2}{\omega_{ni}^2}\right)}$$

Where:

$g \in \mathbb{Z}$  is the type of the transfer function. In general, it indicates the number of singularity ( $s = 0$ ) of the transfer function. More precisely, if  $g > 0$  it indicates the number of poles that are equal to zero. If  $g < 0$ , it indicates the number of zeros that are equal to zero.

$\mu \in \mathbb{R}$  is the gain of the transfer function.

# Transfer Function

$$G(s) = \frac{\mu}{s^g} \frac{\prod_i (1 + T_i s) \prod_i \left(1 + 2\frac{\zeta_i}{\sigma_{ni}} s + \frac{s^2}{\sigma_{ni}^2}\right)}{\prod_i (1 + \tau_i s) \prod_i \left(1 + 2\frac{\xi_i}{\omega_{ni}} s + \frac{s^2}{\omega_{ni}^2}\right)}$$

Where:

$T_i \in \mathbb{R}$  and  $\tau_i \in \mathbb{R}$  are the time constants of the zeros and poles that are real and not null ( $s \neq 0$ )

$\sigma_{n,i} \in \mathbb{R}^+$  and  $\omega_{n,i} \in \mathbb{R}^+$  are the natural frequencies of the complex, conjugated poles

$\zeta_i \in (-1, 1)$  and  $\xi_i \in (-1, 1)$  are the damping coefficients of the complex conjugated poles and zeroes that are

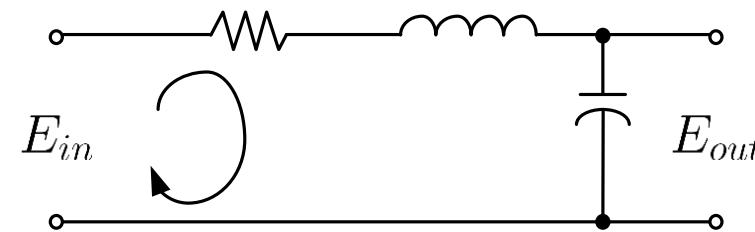
# Transfer Function

- The transfer function describe the cause and effect relationship between input and output

The transfer function of a linear system is the Laplace transform of its response to a unit impulse input

---

*Example: An R-L-C filter*



- applying Kirchhoff's law:  $E_i = iR + L\frac{di}{dt} + \frac{1}{C} \int idt, \quad E_o = \frac{1}{C} \int idt$
- applying the Laplace transform:  $E_i(s) = I(s)(R + sL + \frac{1}{sC}), \quad E_o(s) = I(s)\frac{1}{sC}$
- transfer function:  $\frac{E_o(s)}{E_i(s)} = \frac{1}{s^2LC + sCR + 1}$

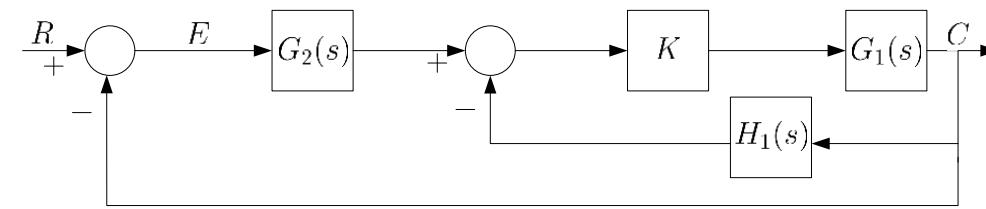
# System block diagram

- The interconnection of components to form a system is conveniently shown by blocks arranged in some sort of diagram.

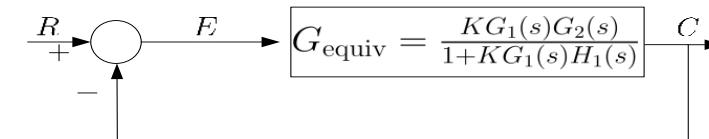
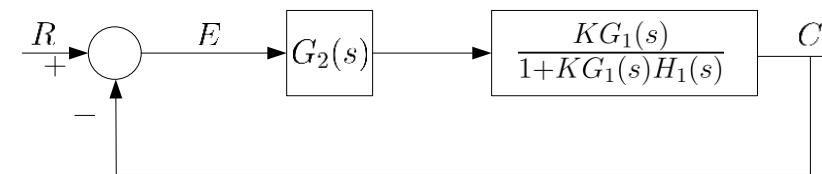
---

*Example: manipulation of a system block diagram*

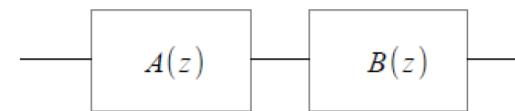
Reduction of minor feedback loop to one equivalent cascade block



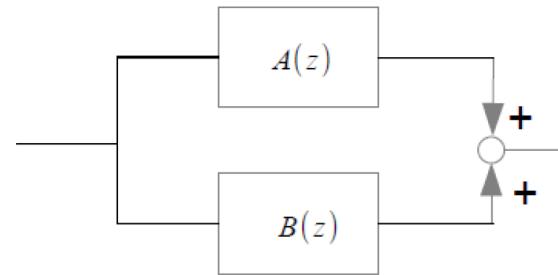
Cascaded blocks combined to give one equivalent block



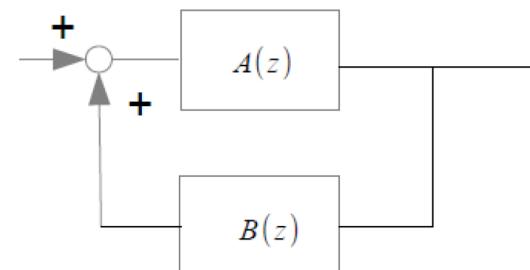
# System block diagram



$$D(z) = A(z) B(z)$$

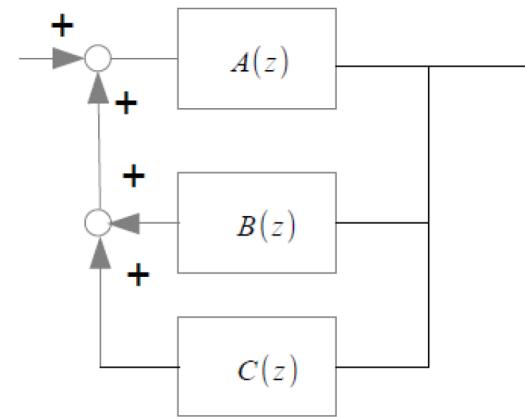


$$D(z) = A(z) + B(z)$$

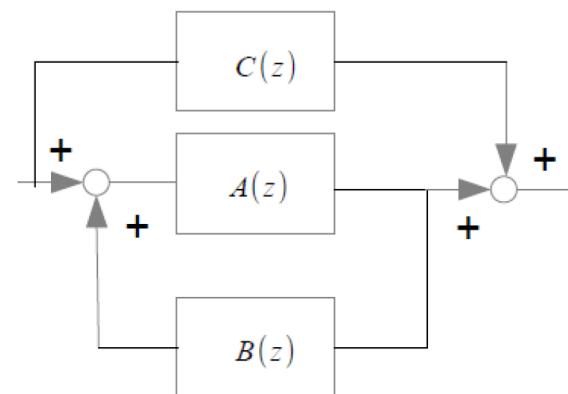


$$D(z) = \frac{A(z)}{1 - A(z)B(z)}$$

# System block diagram

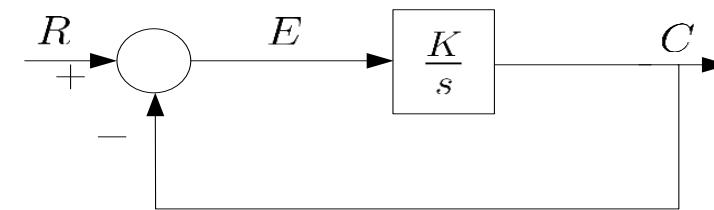


$$D(z) = \frac{A(z)}{1 - A(z)(B(z) + C(z))}$$



$$D(z) = C(z) + \frac{A(z)}{1 - A(z)B(z)}$$

# First order system



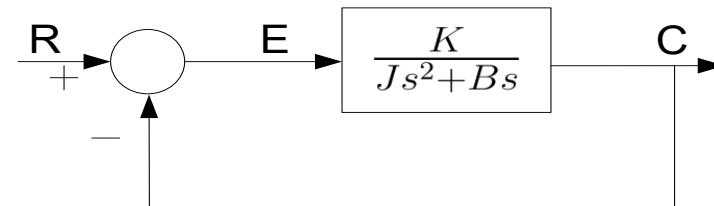
$K$  : a positive real number serving as the gain of the closed-loop system

Closed loop system:  $\frac{C}{R} = \frac{1}{sT+1}$

$T = \frac{1}{K}$  is defined as the **time constant** of the system

The closed-loop pole for this system is located at  $s = -\frac{1}{T} = -K$ .  
Since  $K > 0$  the closed loop system is guaranteed to be stable.

# Second order system



Transfer function of a general second-order system is

$$\frac{C}{R} = \frac{K}{Js^2 + Bs + K} = \frac{\omega_N^2}{s^2 + 2\xi\omega_N + \omega_N^2}$$

*Note: This is second order because the highest power of s = 2*

$$\xi = \frac{B}{2\sqrt{KJ}} \rightarrow \text{damping ratio, will determine how much the system oscillates as the response decays toward steady state}$$

$$\omega_N = \sqrt{\frac{K}{J}} \rightarrow \text{the undamped natural frequency, will determine how fast the system oscillates during any transient response}$$

# Basic controllers

- Proportional control
  - adjustable gain (amplifier)

$$u(t) = K_p e(t)$$

$$\frac{U(s)}{E(s)} = K_p$$

$e(t)$ : control error

- Integral control
  - eliminates bias (steady-state error)
  - may cause oscillations

$$u(t) = K_i \int_0^t e(t) dt$$

$$\frac{U(s)}{E(s)} = \frac{K_i}{s}$$

- Differential control
  - effective in transient periods, provides faster response
  - never used alone

$$u(t) = K_d \frac{d}{dt} e(t)$$

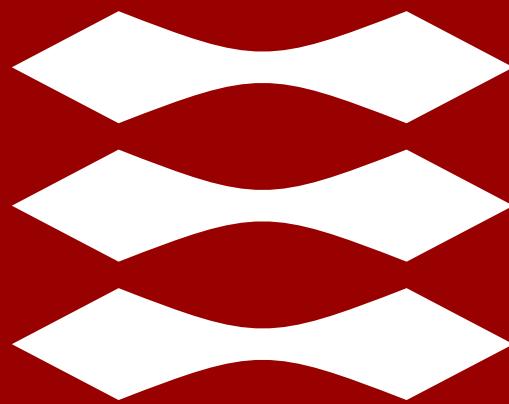
$$\frac{U(s)}{E(s)} = K_d s$$

- Integral and differential control are typically used in combination with at least proportional control

# References

- “Automatic Control Systems”, G. J. Thaler, *West Publishing Company*, 1989.
- “An Introduction to Control Systems”, K. Warwick, *World Scientific Publishing Company*, 1996.
- “Control Engineering”, O. Jannerup og P. H. Sørensen, 3. edition (2004) or 4. edition (2006), Polyteknisk Forlag. (*In danish*)
- “The Control Handbook”, William S. Levine, editor, CRC press, 1996.
- “Feedback Control of Dynamical Systems”, G. F. Franklin, J. D. Powell, and A. Emami-Naeini, Prentice-Hall, Upper Saddle River, NJ, 4th edition, 2002.
- “PID Controllers: Theory, Design, and Tuning”, K. J. Åström and T. H̄agglund, International Society for Measurement and Control, Seattle, WA, 2nd edition, 1995
- <http://www.facstaff.bucknell.edu/mastascu/eControlHTML/CourseIndex.html>
- .....

**DTU**



ROBOTICS 34753

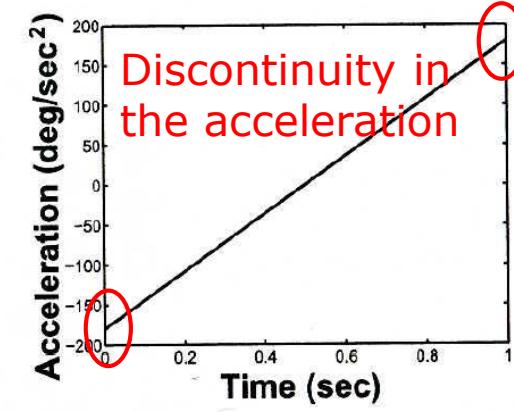
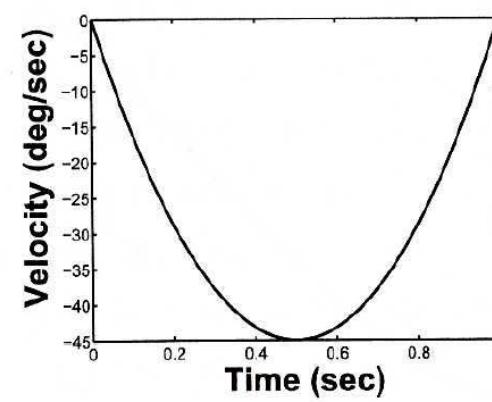
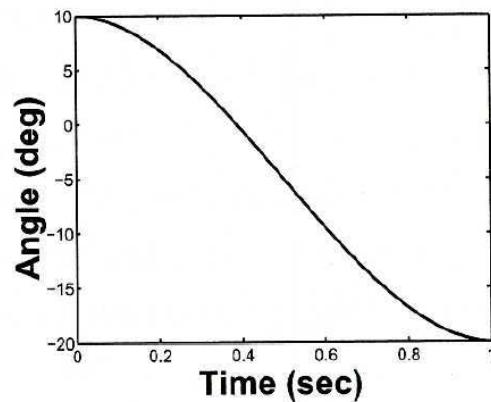
# Independent Joint Control

# Outline

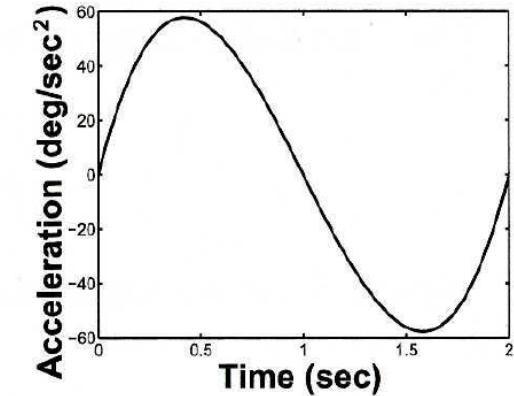
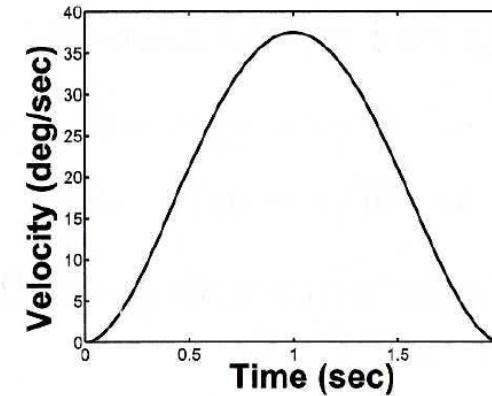
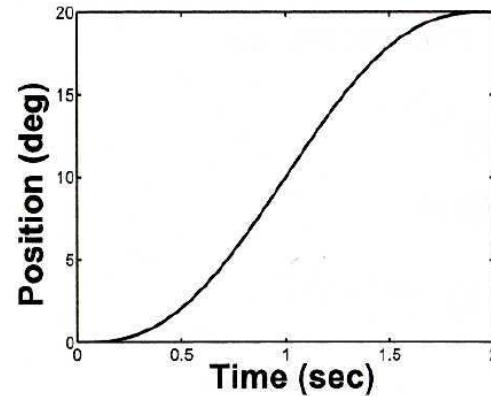
- Review
  - Trajectory Planning
  - Dynamics
- Actuator Dynamics
- Dynamical Model of a Robot with One Joint
- Controller Design

# Review

- Trajectory planning
  - 3<sup>rd</sup> order polynomial



- 5<sup>th</sup> order polynomial



# Review

- Dynamics: Euler-Lagrange Equations
  - Euler-Lagrange formulation

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \quad \mathcal{L} = \mathcal{K} - \mathcal{P}$$

- Kinetic energy

$$\mathcal{K} = \frac{1}{2} \dot{q}^T \left[ \sum_{i=1}^n m_i J_{v_i}^T(q) J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I R_i(q)^T J_{\omega_i}(q) \right] \dot{q}$$

- Potential energy

$$\mathcal{P} = \sum_{i=1}^n \mathcal{P}_i = \sum_{i=1}^n m_i g^T r_{ci}$$

# Review

- Equation of motion

$$\sum_{j=1}^n d_{kj} \ddot{q}_j + \sum_{i,j} c_{ijk}(q) \dot{q}_i \dot{q}_j + g_k(q) = \tau_k$$



$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau$$

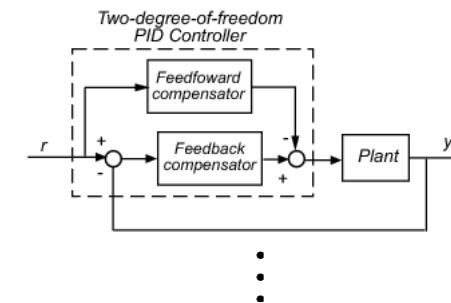
$$D(q) = \sum_{i=1}^k (m_i J_{v_i}^T(q) J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I R_i(q)^T J_{\omega_i}(q))$$

**Element of  $C(q, \dot{q})$ :**  $c_{kj} = \sum_{i=1}^n c_{ijk}(q) \dot{q}_i = \sum_{i=1}^n \frac{1}{2} \left( \frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right) \dot{q}_i$

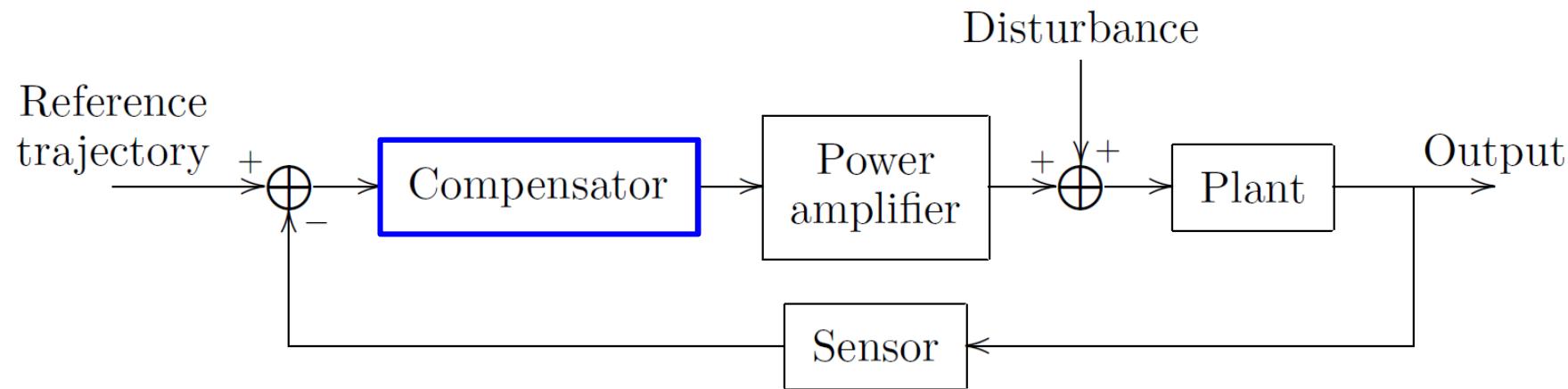
$$g(q) = [g_1, \dots, g_n(q)]^T$$

# Development control systems for robots

- Modeling robot behaviors
  - Kinematics
  - Dynamics
  - ...
- Choice of actuators, gears, sensors and their allocation
- Choice of control architecture
  - Linear vs. nonlinear
  - Delay compensation
  - Adaptive online parameter estimation
  - Robustness
  - ...



# Basic structure of a feedback system



Design objective: choose the **compensator** to drive the plant's output to **follow** a desired reference.

# Actuator Dynamics

# Permanent magnet DC motor

- A current-carrying conductor in magnetic field
  - Force

$$F = i_a \times \phi$$

$i_a$ : current

$\phi$  : magnetic flux

- Torque

$$\tau_m = K_1 \phi i_a$$

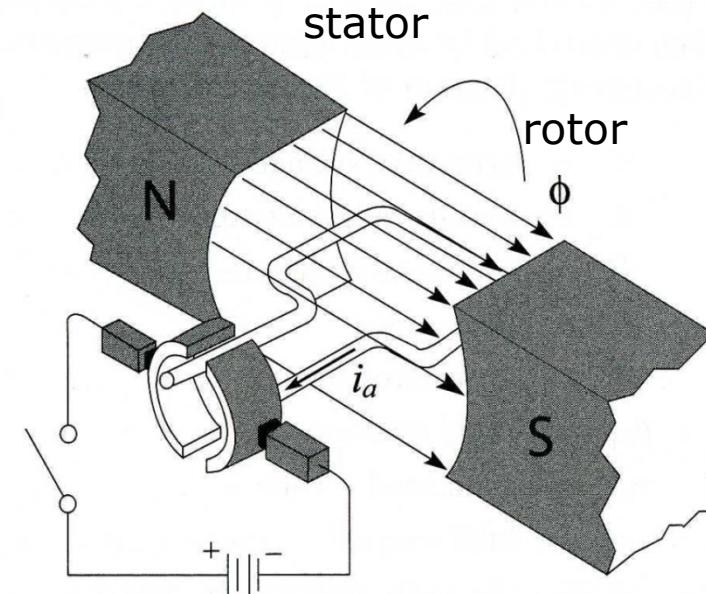
$K_1$ : physical constant

- Back emf (electromotive force)

$$V_b = K_2 \phi \omega_m$$

$K_2$ : proportionality constant

$\omega_m$ : angular velocity



# Permanent magnet DC motor

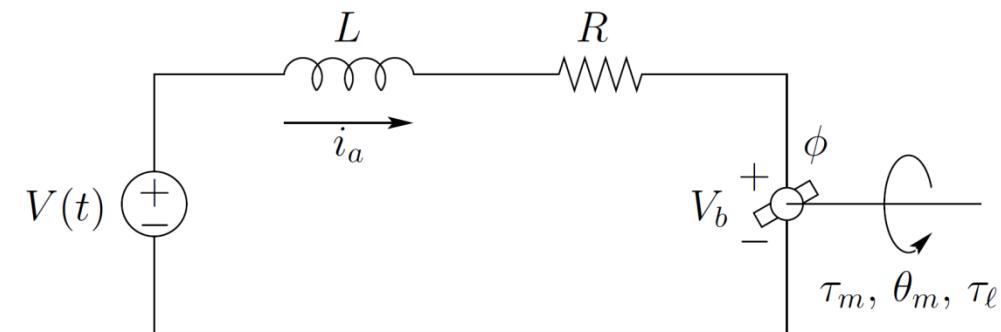
- Relations between

$$i_a, V, \omega_m, \tau_m$$

$$L \frac{di_a}{dt} + Ri_a = V - V_b$$

$$\tau_m = K_1 \phi i_a = K_m i_a$$

$$V_b = K_2 \phi \omega_m = K_b \omega_m$$



$i_a$  : armature current

$V$  : armature voltage

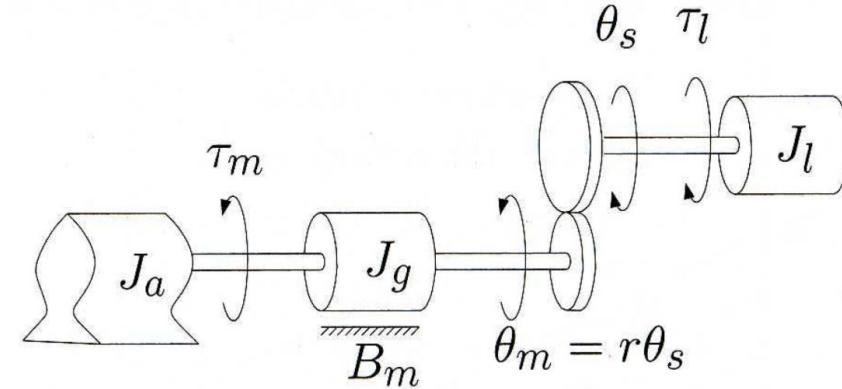
$\omega_m$  : angular velocity

$\tau_m$  : generated torque

# Independent joint model

- Single link with actuator/gear train
  - Each link: independent **SISO** system
  - Large gear reduction: reduce the nonlinear coupling among the links
  - Equation of motion

$$\begin{aligned} J_m \left( \frac{d^2\theta_m}{dt^2} \right) + B_m \frac{\theta_m}{dt} &= \tau_m - \tau_l/r \\ &= K_m i_a - \tau_l/r \end{aligned}$$



$B_m$ : coefficient of motor friction

$J_a$ : actuator inertia

$J_g$ : gear inertia

$J_l$ : load inertia

$$J_m = J_a + J_g$$

# Independent joint model

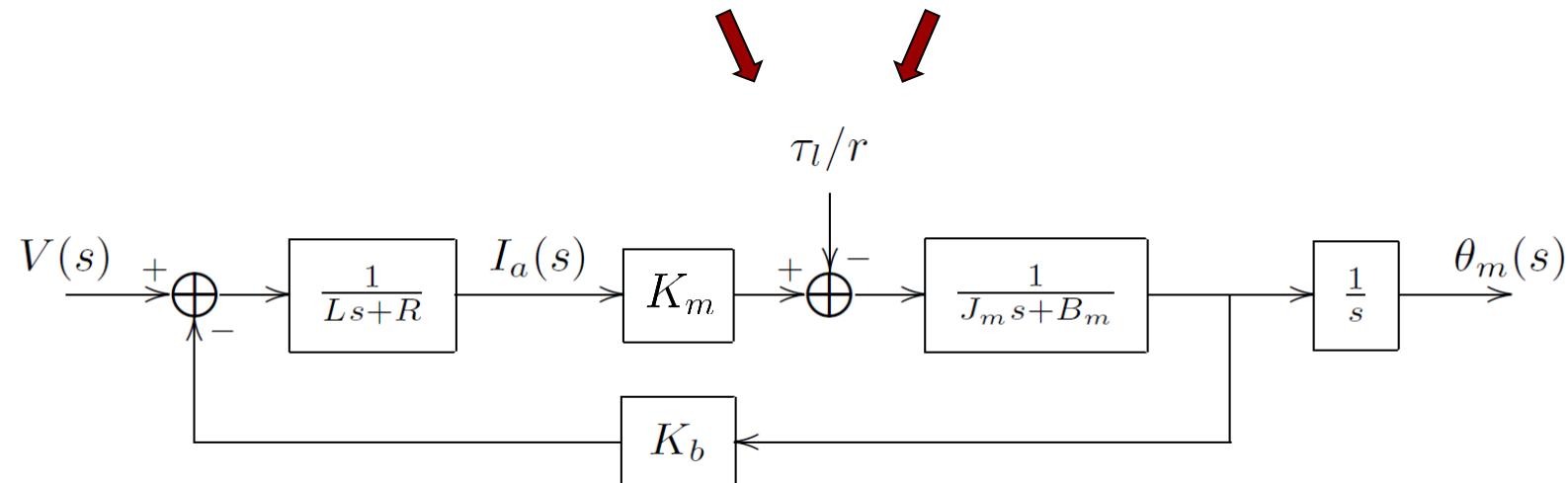
- Augmenting the mechanical and electrical models

$$J_m \left( \frac{d^2\theta_m}{dt^2} \right) + B_m \frac{\theta_m}{dt} = K_m i_a - \tau_l/r$$

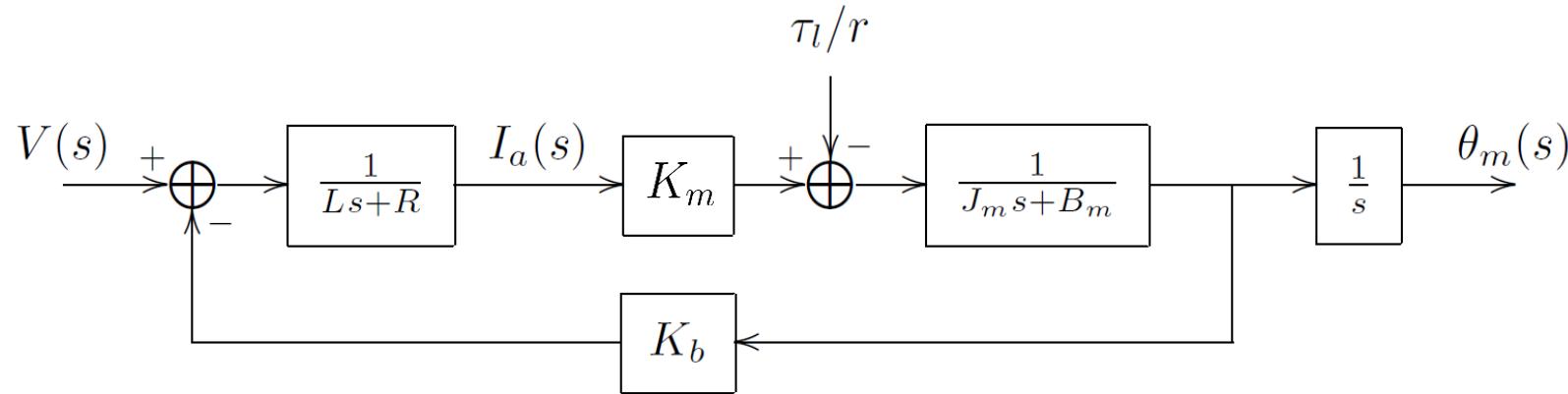
$$L \frac{di_a}{dt} + Ri_a = V - V_b$$

$$(J_m s^2 + B_m s)\theta_m(s) = K_m I_a(s) - \tau_l(s)/r$$

$$(Ls + R)I_a(s) = V(s) - K_b s \theta_m(s)$$



# Independent joint model



- Transfer function  $V(s) \rightarrow \Theta_m(s)$

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m}{s[(LS + R)(J_ms + B_m) + K_bK_m]}$$

- Reduction of the model based on

$$\frac{\Theta_m(s)}{V(s)} = \frac{K_m/R}{s[J_ms + B_m + K_bK_m/R]}$$

- Transfer function  $\tau_l(s) \rightarrow \Theta_m(s)$

$$\frac{\Theta_m(s)}{\tau_l(s)} = \frac{-(LS + R)/r}{s[(LS + R)(J_ms + B_m) + K_bK_m]}$$

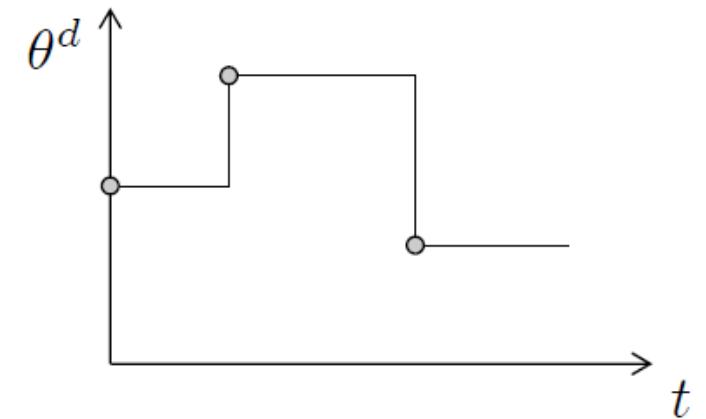
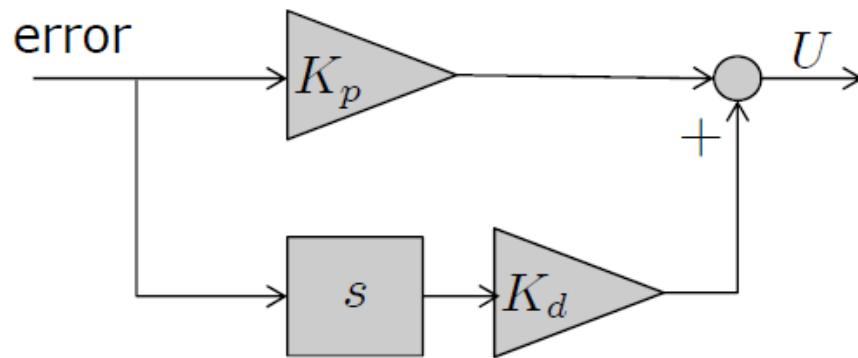
$$\frac{L}{R} \approx 0$$

$$\frac{\Theta_m(s)}{\tau_l(s)} = \frac{-1/r}{s[J_ms + B_m + K_bK_m/R]}$$

# **Controller Design: PD and PID controller**

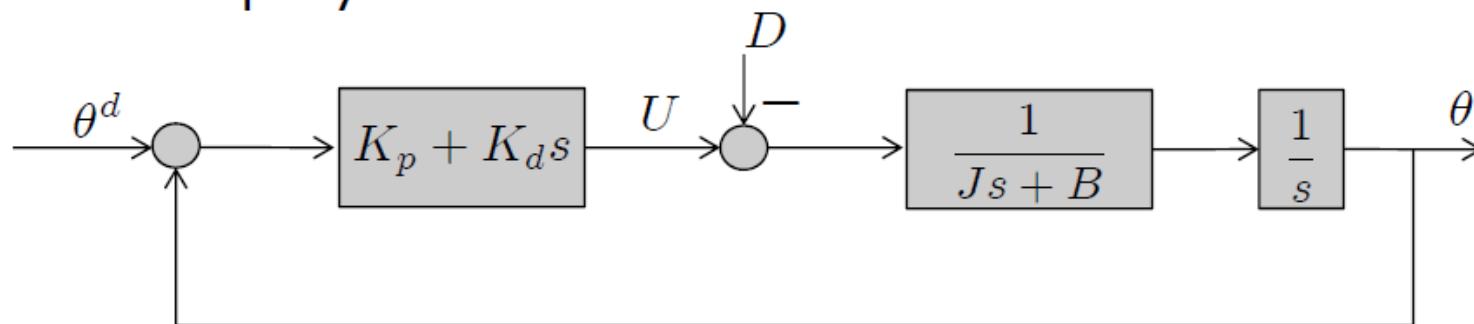
# Set-Point tracking with PD controller

- PD controller



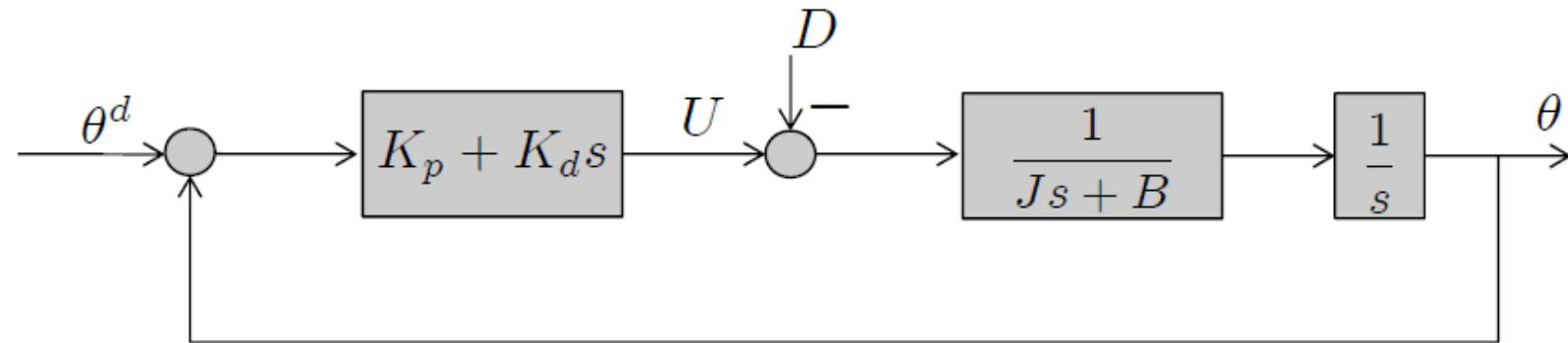
$$U(s) = (K_p + K_d s)(\theta^d(s) - \theta(s))$$

- Closed-loop system



# Set-Point tracking with PD controller

- Relationship between output and input

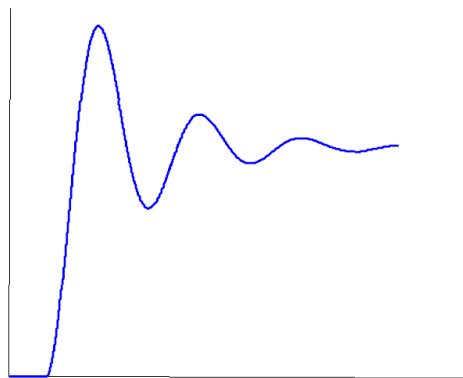


$$[(\theta^d - \theta)(K_p + K_D s) - D(s)] \frac{1}{Js+B} \frac{1}{s} = \theta$$

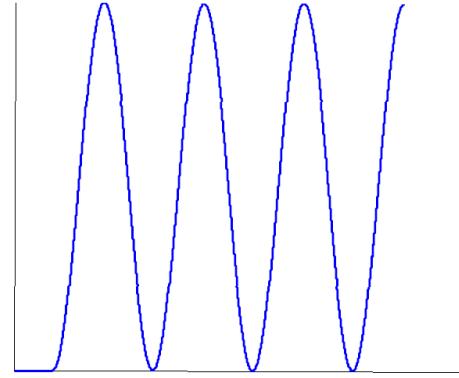
$$\theta = \frac{(K_p+K_D s)\theta^d(s)-D(s)}{J s^2+(B+K_D)s+K_p}$$

- Type I system (the number of  $1/s$  in the open-loop transfer function is 1)
- Stable for all positive values  $K_p, K_d$

# System stability



Asymptotically stable

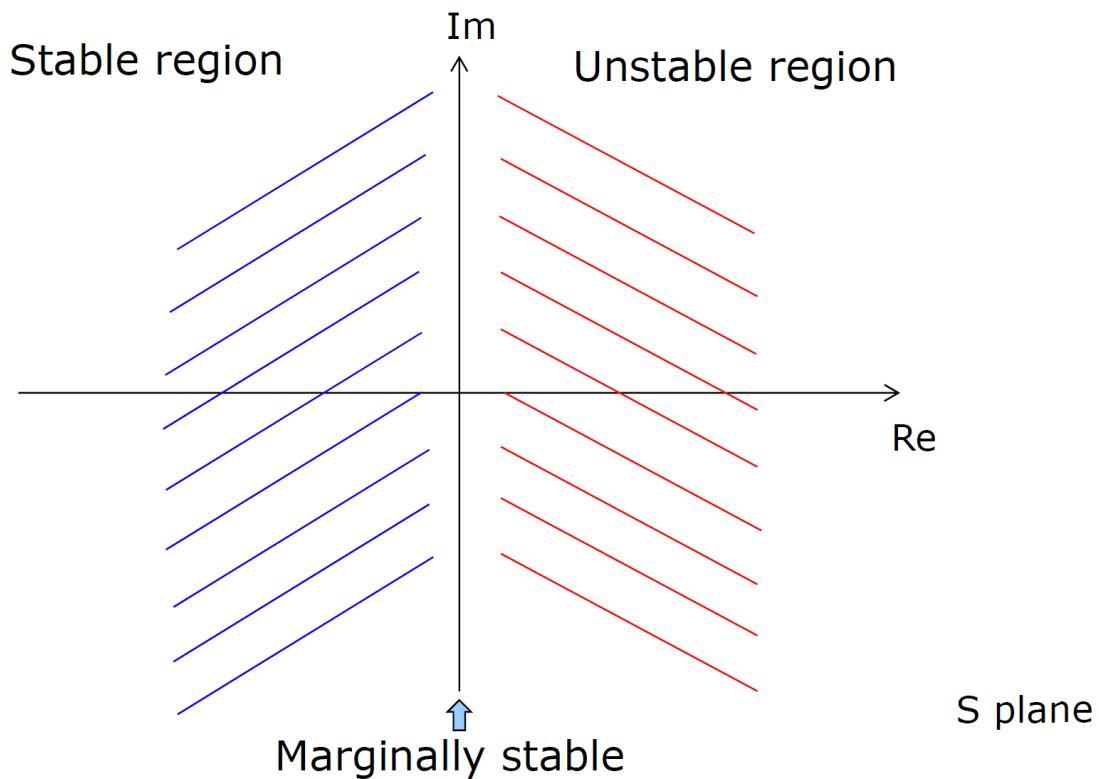


Marginally stable



Unstable

## S-plane for stability analysis



## Root of characteristic polynomial (PD controller)

- Characteristic equation

$$Js^2 + (B + K_d)s + K_p = 0$$

- If roots are on the left half plane -> stable

$$s = \frac{-(B + K_d) \pm \sqrt{(B + K_d)^2 - 4JK_p}}{2J}$$

- For  $(B + K_d)^2 - 4JK_p < 0$  , complex conjugate → stable
- For  $(B + K_d)^2 - 4JK_p = 0$  -> stable
- For  $(B + K_d)^2 - 4JK_p > 0$  :
  - $-(B + K_d) - \sqrt{(B + K_d)^2 - 4JK_p} < 0$  → stable
  - $-(B + K_d) + \sqrt{(B + K_d)^2 - 4JK_p} < 0$  → for  $K_p > 0$  → stable

For all positive values  $K_p, K_d$  → the system with PD controller is stable

## Steady-state error

- Tracking error

$$\theta(t) = \theta^d(t) - \theta(t)$$

$$E(s) = \theta^d(s) - \theta(s)$$

- Assume

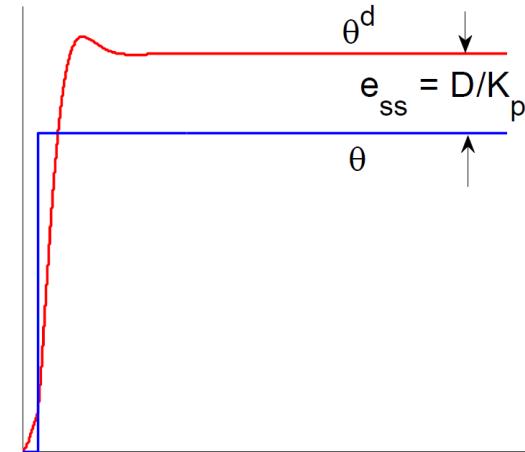
- A step reference input  $\theta^d(s) = \frac{\Omega^d}{s}$

- A constant disturbance  $D(s) = \frac{D}{s}$

- Steady-state error

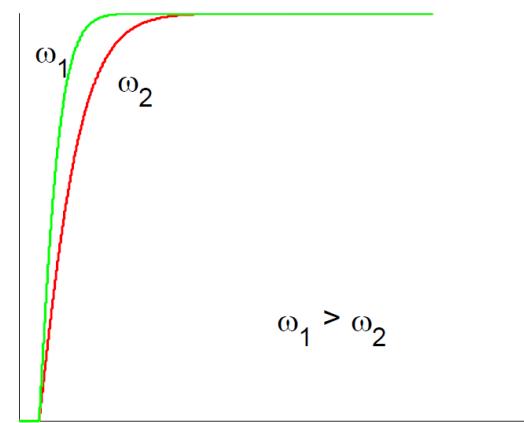
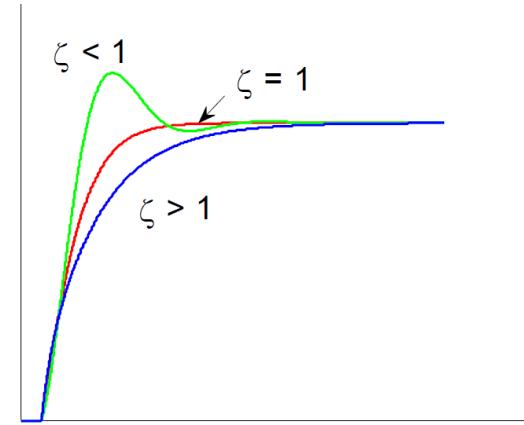
- $\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} sE(s)$   $\leftarrow$  *Final value theorem*

- $e_{ss} = \lim_{s \rightarrow 0} sE(s) = \frac{D}{K_p}$   $\Rightarrow$   $\because D \propto \frac{1}{r}$   $\therefore r \uparrow, e_{ss} \downarrow$



## Natural frequency and damping ratio

- For a standard second order system, step response is determined by the closed loop natural frequency  $\omega$  and damping ratio  $\zeta$
- $s^2 + \frac{B + K_d}{J}s + \frac{K_p}{J} \leftrightarrow s^2 + 2\omega\zeta s + \omega^2$   
     $\Rightarrow K_p = J\omega^2, \quad K_d = 2\omega J\zeta - B$
- Usually choose  $\zeta = 1 \rightarrow$  critically damped response

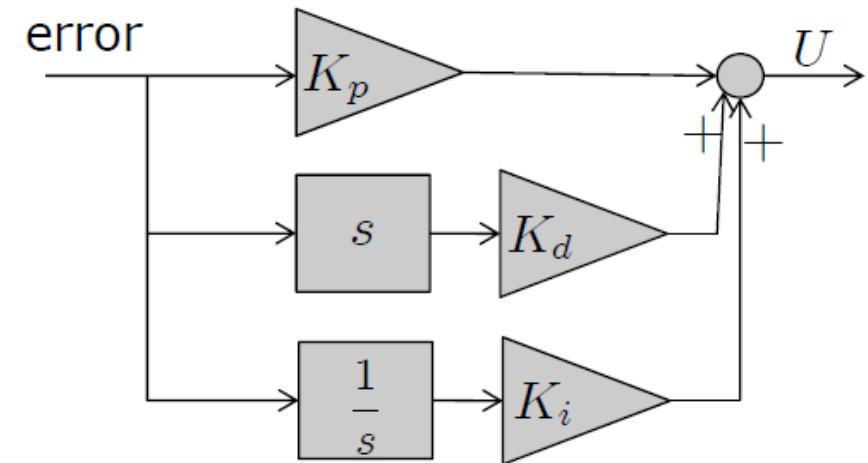


# PID controller

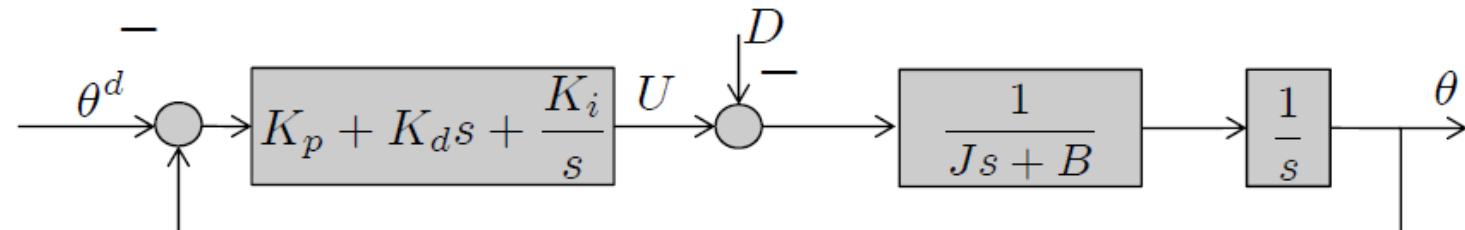
- Control output

$$U(s) = (K_p + K_d s + \frac{K_i}{s})(\theta^d(s) - \theta(s))$$

$$K_p, K_d, K_i > 0$$



- Closed-loop system with PDI controller

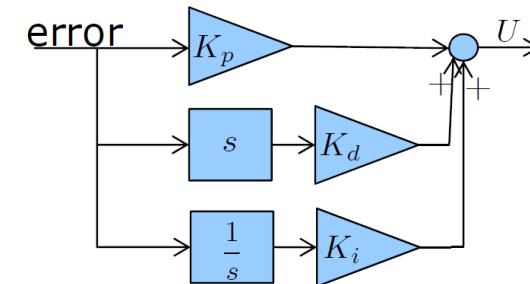


# PID Controller

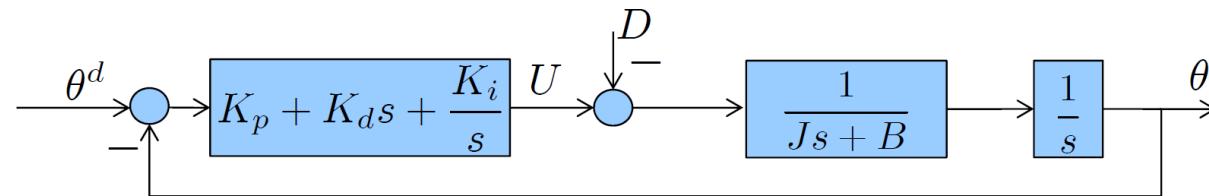
- Control output

$$U(s) = \left( K_p + K_d s + \frac{K_i}{s} \right) (\theta^d(s) - \theta(s))$$

$$K_p, K_d, K_i > 0$$



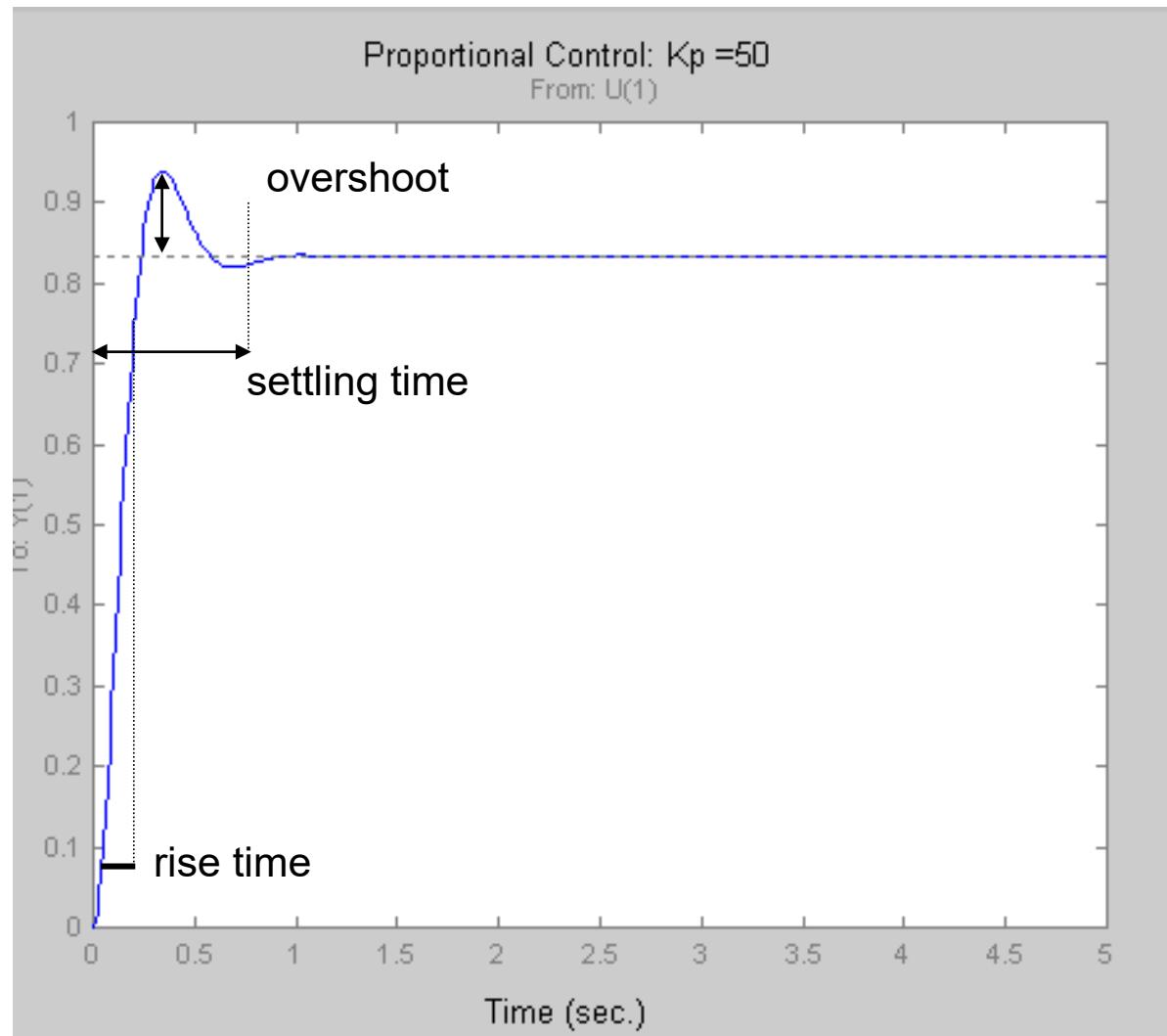
- Closed-loop system with PID controller



$$\theta(s) = \frac{(K_d s^2 + K_p s + \frac{K_i}{s}) \theta^d(s) - sD}{J s^3 + (B + K_d) s^2 + K_p s + K_i} \quad \Rightarrow \text{third-order system}$$

- Stable for  $K_i < \frac{(B + K_d)K_p}{J}$     $\leftarrow$  *Routh-Hurwitz criterion*
- Steady-state error:  $e_{ss} = 0$

# Evaluating the response



↑ steady-state error

**ss error** -- difference from the system's desired value

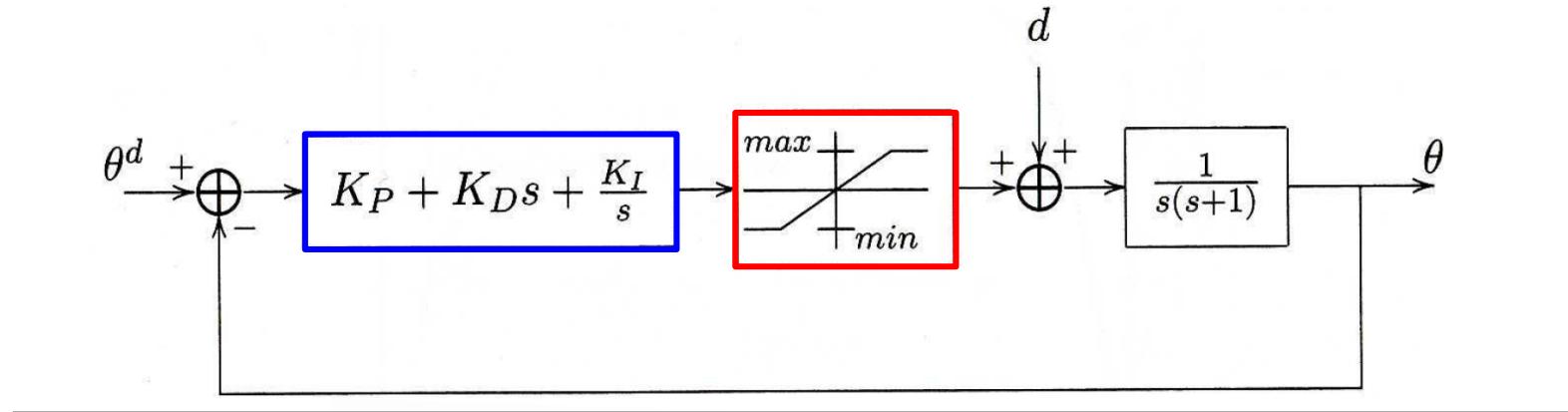
**overshoot** -- % of final value exceeded at first oscillation

**rise time** -- time to span from 10% to 90% of the final value

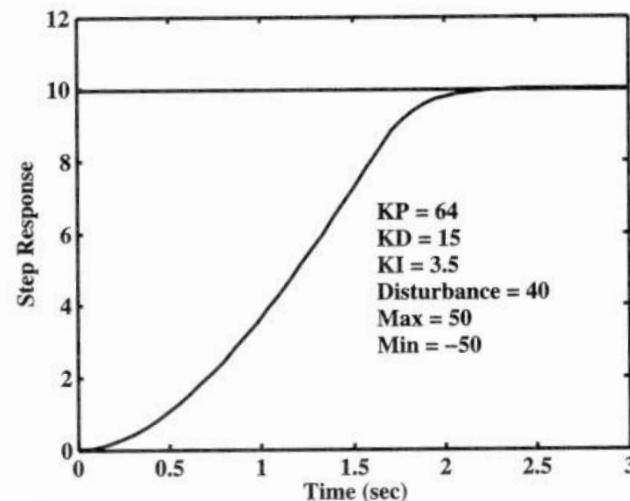
**settling time** -- time to reach within 2% of the final value

# Input saturation

- Saturation: limits on the maximum torque (or current) input



Example:

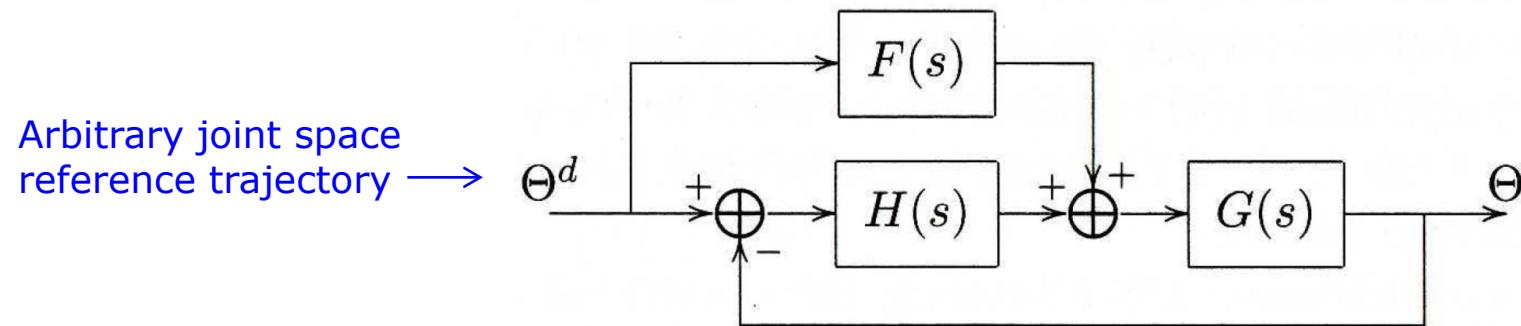


Effect of saturation:  
much slower rise time

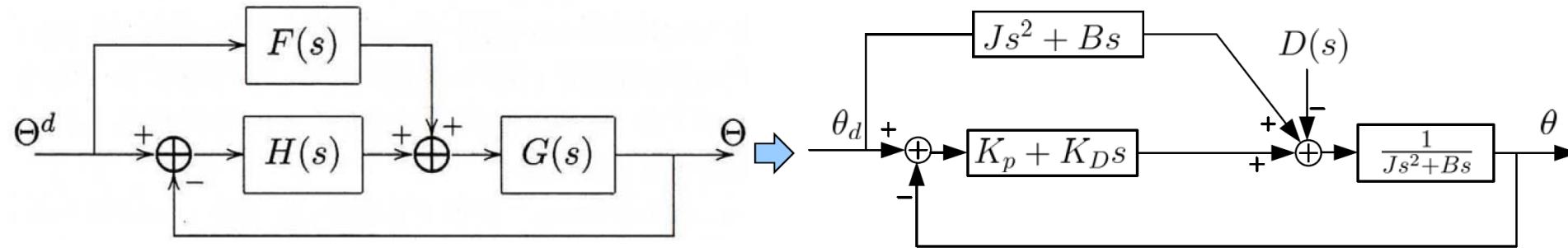
# Controller Design: Feedforward control

# Feedforward control

- Track **time varying** trajectories
- Transfer function  $F(s)$ : new parameter for design
- Choice of  $F(s)$  stable, proper

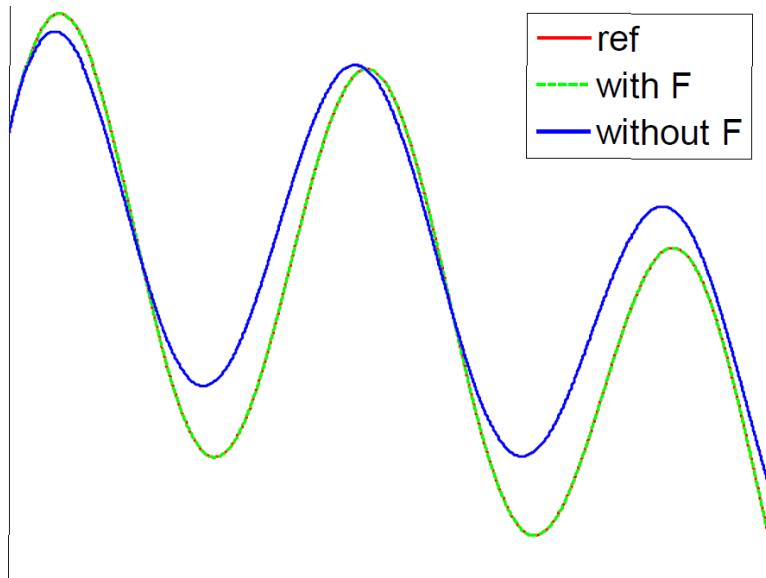


# Feedforward control for tracking time varying trajectories

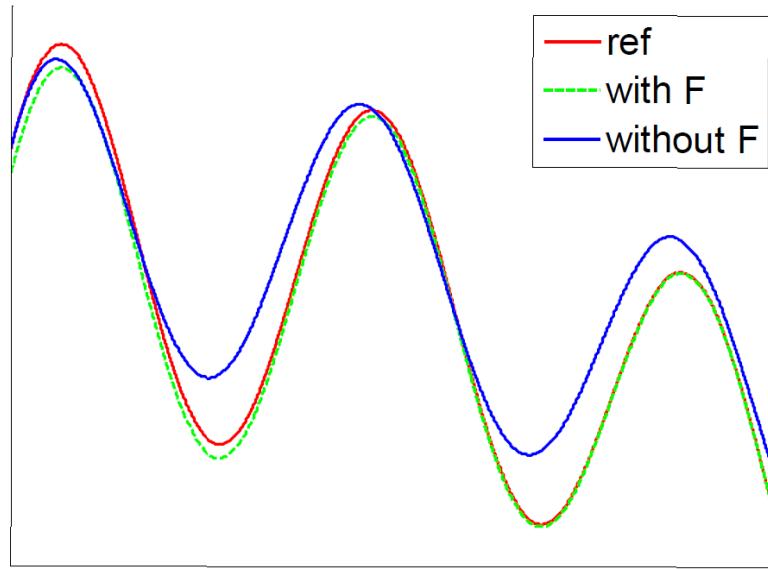


- Assume:  $G(s) = \frac{q(s)}{p(s)}$ ,  $H(s) = \frac{c(s)}{d(s)}$ ,  $F(s) = \frac{a(s)}{b(s)}$
- Transfer function:  $\frac{\theta}{\theta^d} = \frac{(bc + ad)q}{(pd + cq)b}$
- Stable if  $pd + cq$  and  $b$  are Hurwitz  $\rightarrow$  the feedforward transfer function itself must be stable
- Choose  $F(s) = \frac{1}{G(s)}$ :  $\theta = \theta^d \rightarrow$  tracking error:  $Err = \theta^d - \theta = 0$

# With vs Without Feedforward transfer function



Without disturbance = 0



With disturbance

The closed-loop system will track any desired trajectory. The steady state error is only determined by the disturbance.



$$\Sigma_{in} = \frac{1}{m^2} J_2 \ddot{\theta}_m + \frac{1}{m} \tilde{\Sigma}_L$$

$$J_n \ddot{\theta}_n + D_n \dot{\theta}_n = \Sigma_n - \frac{1}{m^2} J_2 \ddot{\theta}_n - \frac{1}{m} \Sigma_L$$

$$\left( J_n + \frac{J_2}{m^2} \right) \ddot{\theta}_n + D_n \dot{\theta}_n = \Sigma_n - \frac{1}{m} \Sigma_L$$



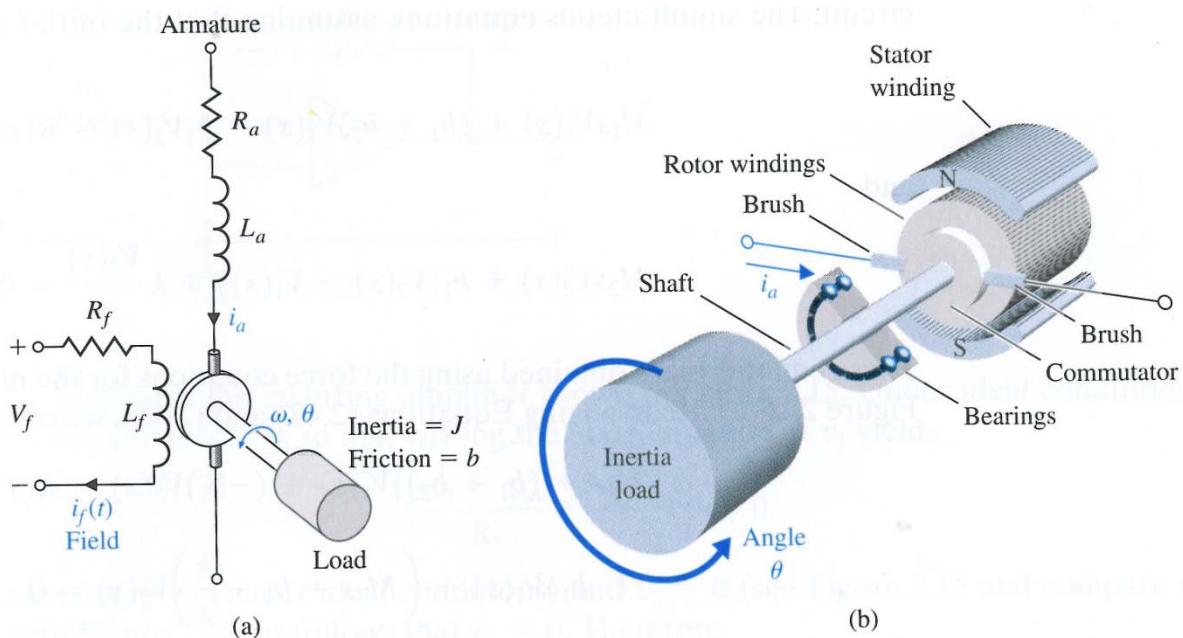
REFLECTED  
WAVE (R<sub>I</sub>)

$$\uparrow_m \rightarrow \downarrow R_I$$

## DC Motor

The DC motor is a power actuator device that delivers energy to a load. The DC motor converts direct current (DC) electrical energy into rotational mechanical energy. A major fraction of the torque generated in the rotor (armature) of the motor is available to drive an external load. Because of features such as high torque, speed controllability over a wide range, portability, well-behaved speed-torque characteristics, and adaptability to various types of control methods, DC motors are widely used in numerous control applications, including robotic manipulators, tape transport mechanisms, disk drives, machine tools, and servo-valve actuators.

The transfer function of the DC motor will be developed for a linear approximation to an actual motor, and second-order effects, such as hysteresis and the voltage drop across the brushes, will be neglected. The input voltage may be applied to the field or armature terminals.



The air-gap flux of the motor is proportional to the field current, provided the field is unsaturated, so that

$$\phi = K_f i_f,$$

where  $i_f$  is the field current,  $K_f$  is the proportional constant, and  $\phi$  is the field flux. The torque derived by the motor is assumed to be related linearly to  $\phi$  and the armature current as follows:

$$T_m(t) = K_1 \phi i_a(t) = K_1 K_f i_f(t) i_a(t),$$

where  $i_a$  is the armature current, and  $K_1$  is a constant. It is clear from this equation that, to have a linear system, one current must be maintained constant while the other current becomes the input current. The armature-controlled DC motor uses the armature current as the control variable. The stator field can be established by a field coil and current or a permanent magnet. When a constant field current is established in a field coil, the motor torque in *Laplace transform* notation is

$$T_m(s) = (K_1 K_f I_f) I_a(s) = K_m I_a(s),$$

where  $K_m$  is a function of the permeability of the magnetic material.

The armature current is related to the input voltage applied to the armature by

$$V_a(s) = (R_a + L_a s) I_a(s) + V_b(s),$$

where  $V_b(s)$  is the back electromotive-force voltage proportional to the motor speed.

Therefore, we have

$$V_b(s) = K_b \omega(s),$$

where  $\omega(s) = s\theta(s)$  is the transformation of the angular speed.

The motor torque is equal to the torque delivered to the load. This relation may be expressed as

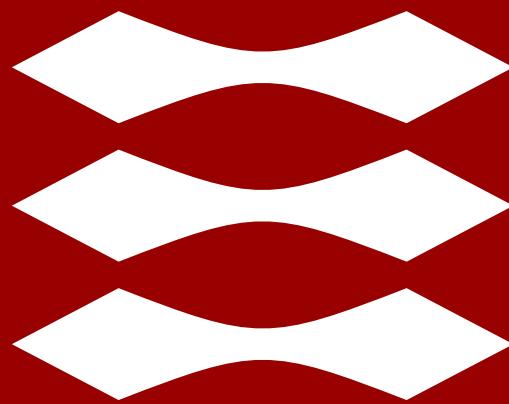
$$T_m(s) = T_L(s) + T_d(s),$$

where  $T_L(s)$  is the load torque and  $T_d(s)$  is the disturbance torque. The load torque for rotating inertia is written as:

$$T_L(s) = J s^2 \theta(s) + b s \theta(s).$$

- Calculate the transfer function  $G(s) = \frac{\theta(s)}{V_a(s)}$  using the equations presented above, and letting  $T_d(s) = 0$ ;
- Build a Simulink/Matlab model of the DC motor including a controller (P, PD or PID). ( $J = 2, K_m = 10, R_a = 1, L_a = 1, b = 0.5, K_b = 0.1$ )

**DTU**

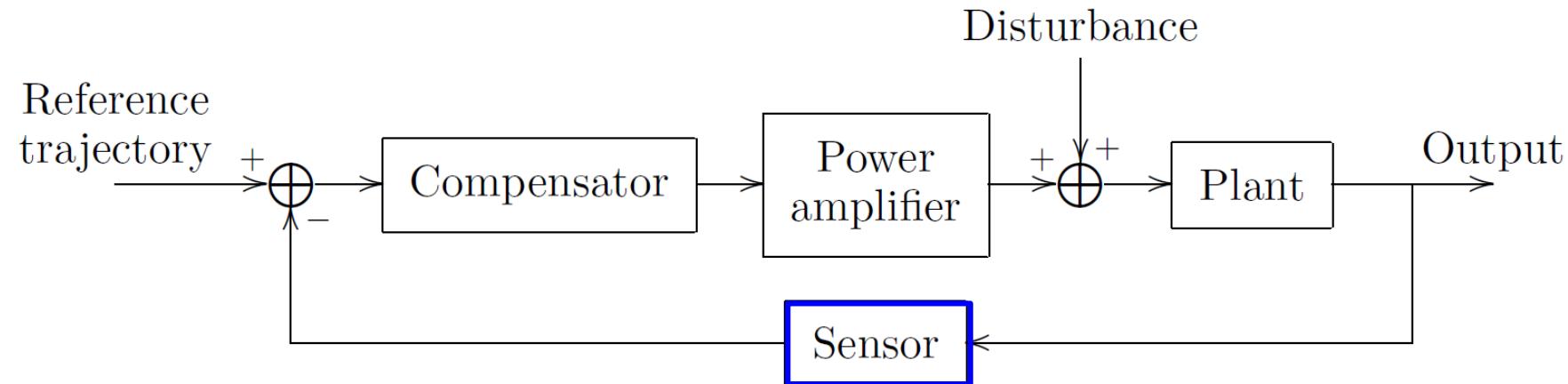


Robotics - 34753

# Sensors for robotics

**Matteo Fumagalli**  
Associate Professor  
Automation and Control Group  
Department of Electrical Engineering  
DTU Lyngby, Building 326

# Basic structure of a feedback system



# Outline

- An overview of Robot Sensor
- Computer Vision
- Vision-based Robot Control

# Robot Sensors

## An Overview

# What is sensing?

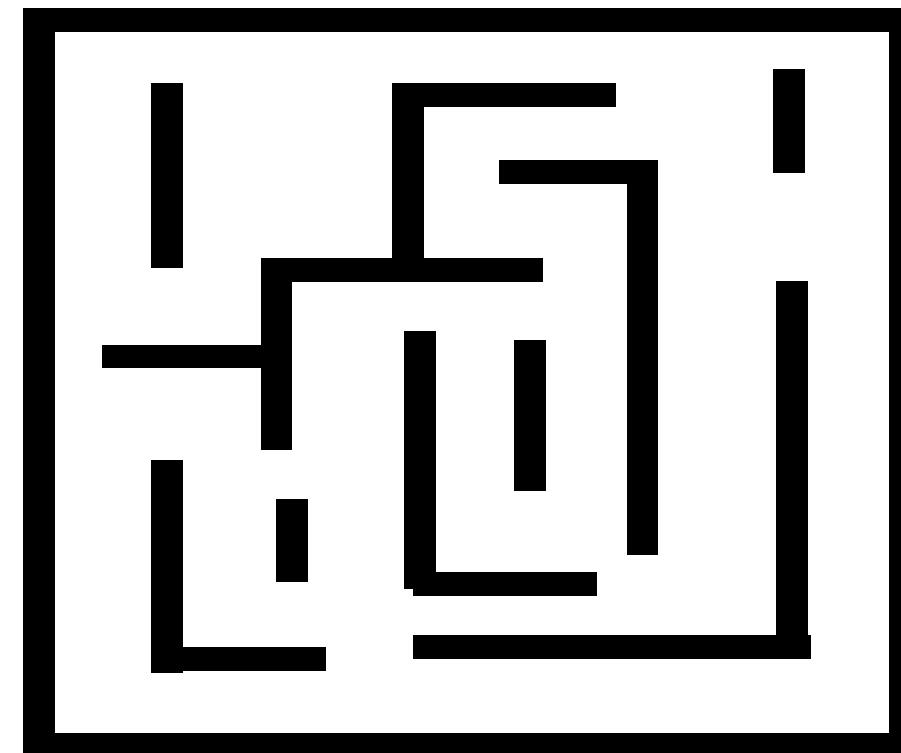
- Collect information about the world
  - Sensor - an electrical/mechanical/chemical device that maps an environmental attribute to a quantitative measurement
  - Each sensor is based on a ***transduction principle*** - conversion of energy from one form to another



## Acoustic      electrical energy

# Robotic systems need sensors for ...

**Where am I?**



**Localization**

# Robotic systems need sensors for ...

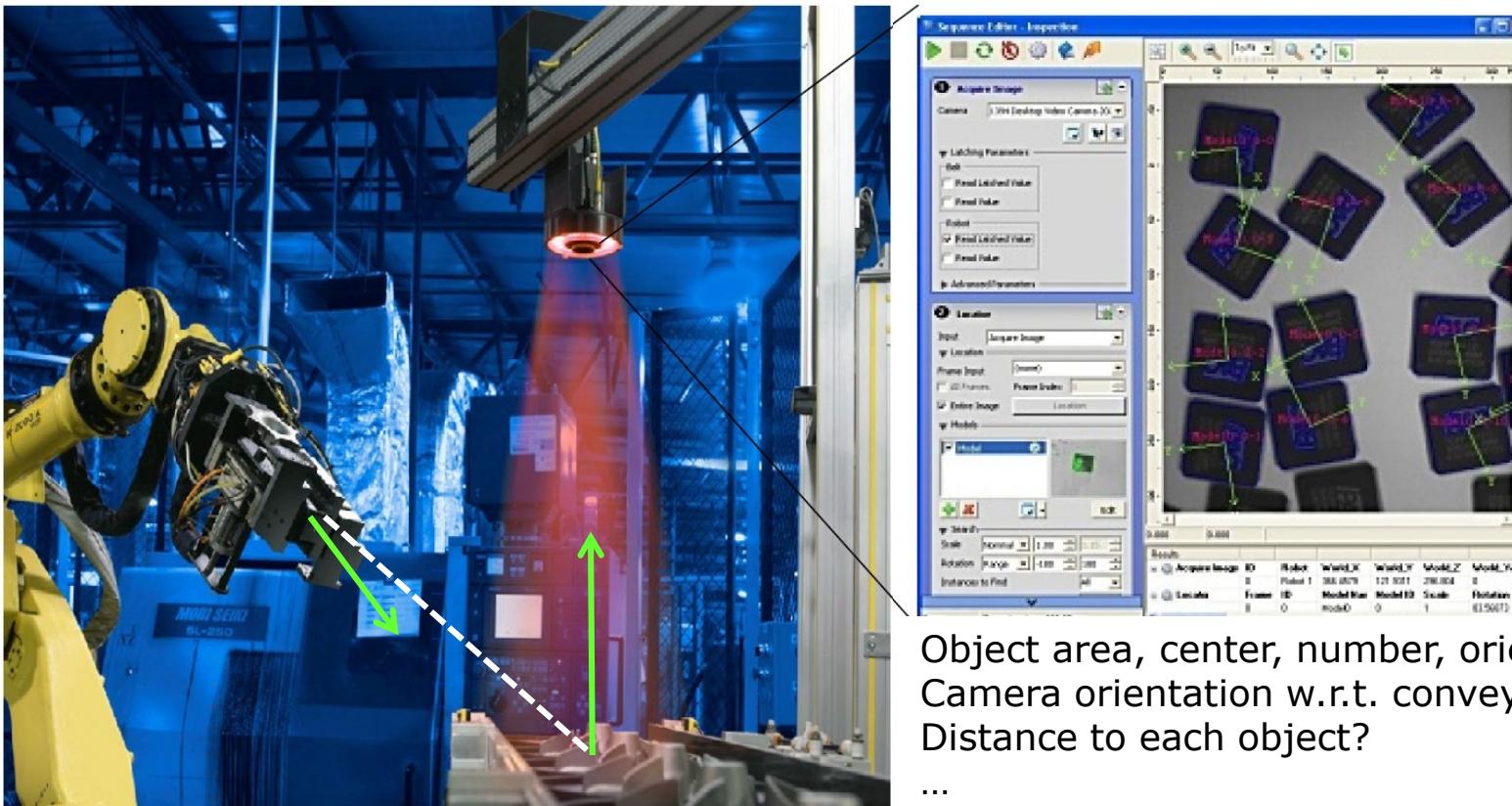
**Will I hit anything?**



**Obstacle detection**

# Robotic systems need sensors for ...

- Provides “awareness” of surroundings
- Gives the robot capability to goal-seek
- ...



# Sensing for specific tasks

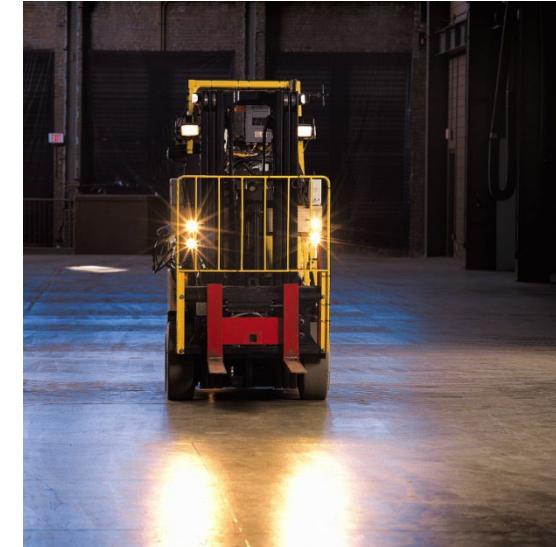
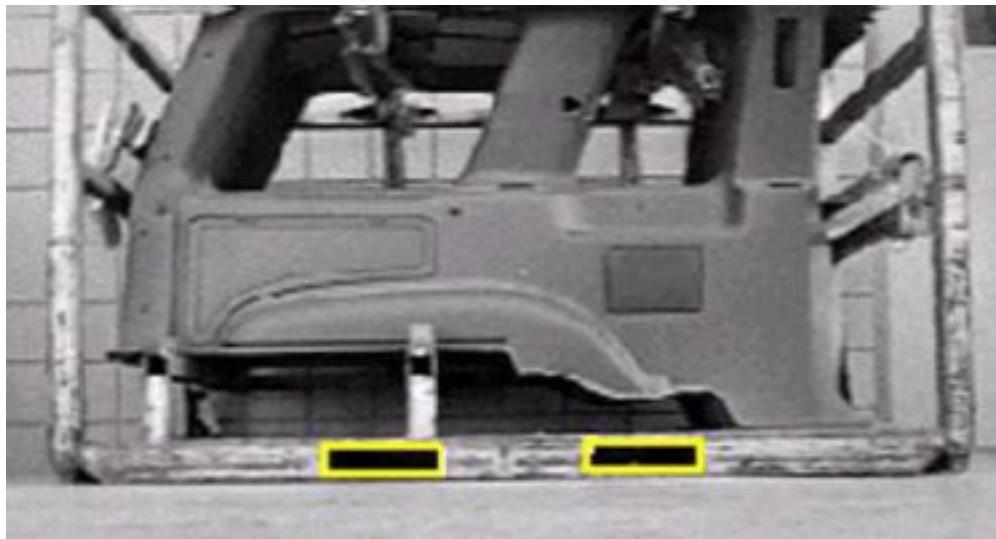
**Where is the crop-line?**



**Autonomous  
harvesting**

# Sensing for specific tasks

**Where are the fork-holes ?**



**Autonomous material handling**

# Sensing for specific tasks

**Where is the face?**



**Face detection & tracking**

# Human sensing and organs

- Vision: eyes (optics, light)
- Hearing: ears (acoustics, sound)
- Touch: skin (mechanics, heat)
- Odor: nose (vapor-phase chemistry)
- Taste: tongue (liquid-phase chemistry)



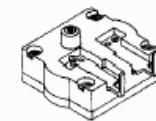
Counterpart?

# What sensors are out there?

Required Measurement	Typical Sensors
Angle of a joint	Encoder
Speed of rotation	Tachometer
Absolute position	GPS
Distance to an object	Optical sensor Ultrasonic proximity sensor Laser distance sensor
Quality of a part Size of a part or features	Machine vision
Direction of travel	Compass
Rate of tilt	Gyroscope
...	



Solar Cell



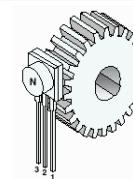
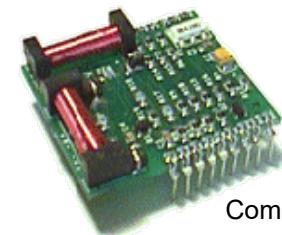
Pressure Switch



IR Reflection Sensor



Gyro

Hall Effect  
Magnetic Field  
Sensors

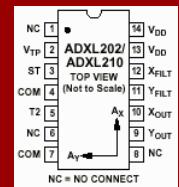
Compass



Gas Sensor

Pendulum Resistive  
Tilt Sensors

kinect



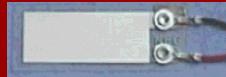
Accelerometer



Gyro



Pendulum Resistive  
Tilt Sensors



Piezo Bend Sensor



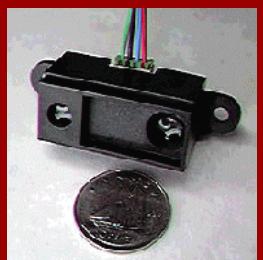
Metal Detector



Gas Sensor



Gieger-Muller  
Radiation Sensor



Digital Infrared Ranging



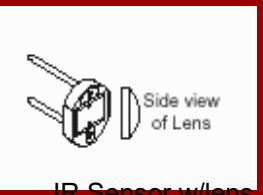
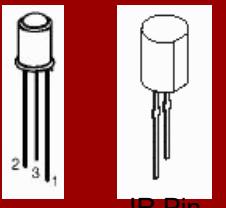
CDS Cell  
Resistive Light Sensor



Resistive Bend Sensors



UV Detector



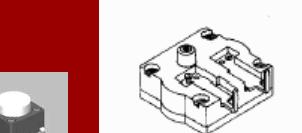
IR Sensor w/lens



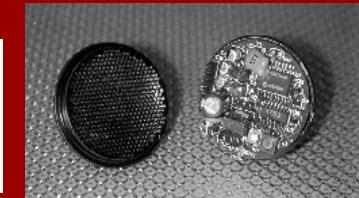
Limit Switch



Mechanical Tilt Sensors



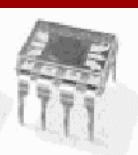
Touch Switch



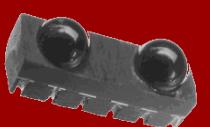
Miniature Polaroid Sensor



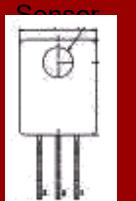
Diode



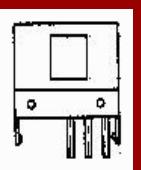
IR Reflection



IR Amplifier Sensor



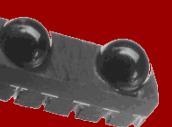
Lite-On IR  
Remote Receiver



Radio Shack  
Remote Receiver



IR Modulator  
Receiver



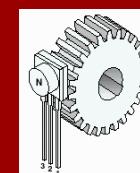
IRDA Transceiver



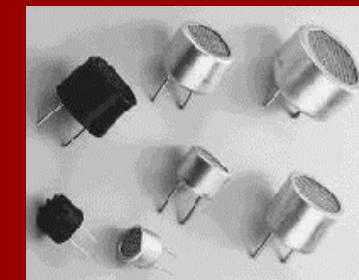
Solar Cell



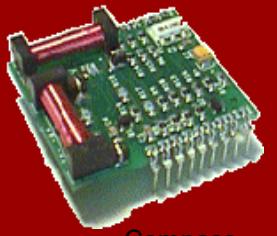
Magnetic Reed Switch



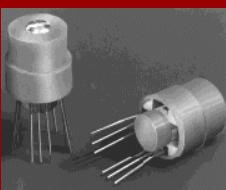
Hall Effect  
Magnetic Field  
Sensors



Piezo Ultrasonic  
Transducers



Compass



Compass

# Sensor Fusion

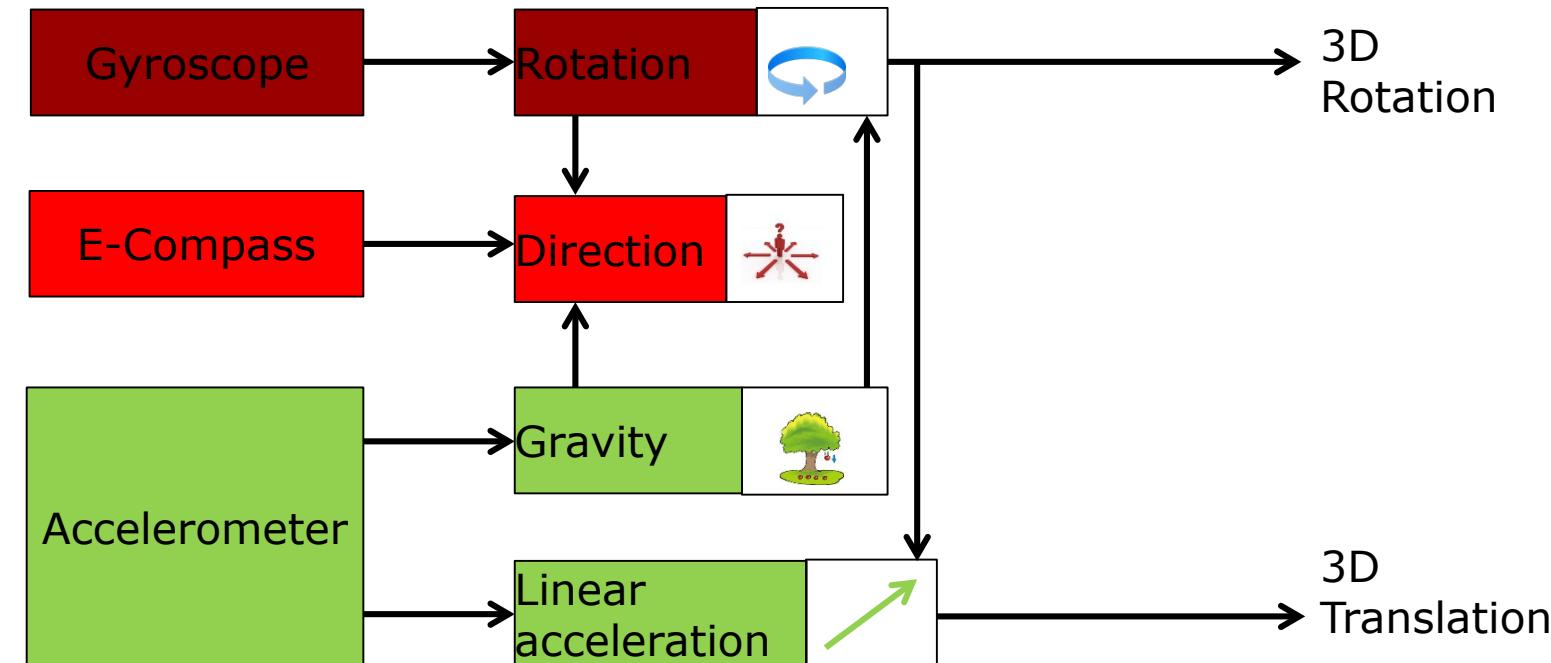
- One sensor is (usually) not enough
  - Real sensors are noisy
  - Limited accuracy
  - Unreliable - failure/redundancy
  - Limited point of view of the environment
    - return an incomplete description of the environment
  - The sensor of choice may be expensive - might be cheaper to combine two inexpensive sensors

# Sensor/Data Fusion

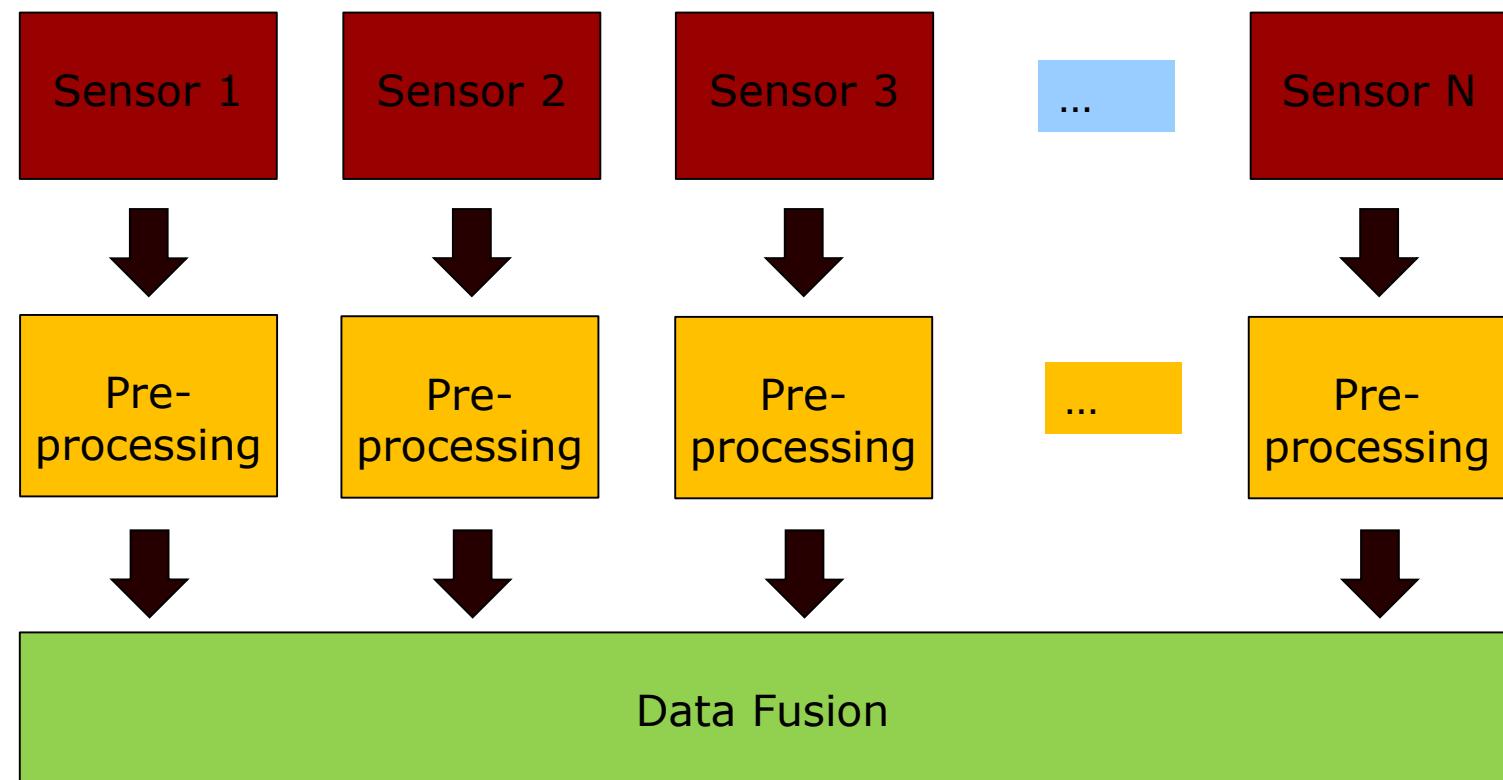
- Combine data from different sources
  - Measurements from different **sensors**
  - Measurements from different **positions**
  - Measurements from different **times**
- Often a mathematical technique that takes into account uncertainties in data sources
  - Kalman filtering
  - Neural networks
  - ...

# Sensor/Data Fusion

- Example: combining data from different sensors to calculate accurate position and orientation information



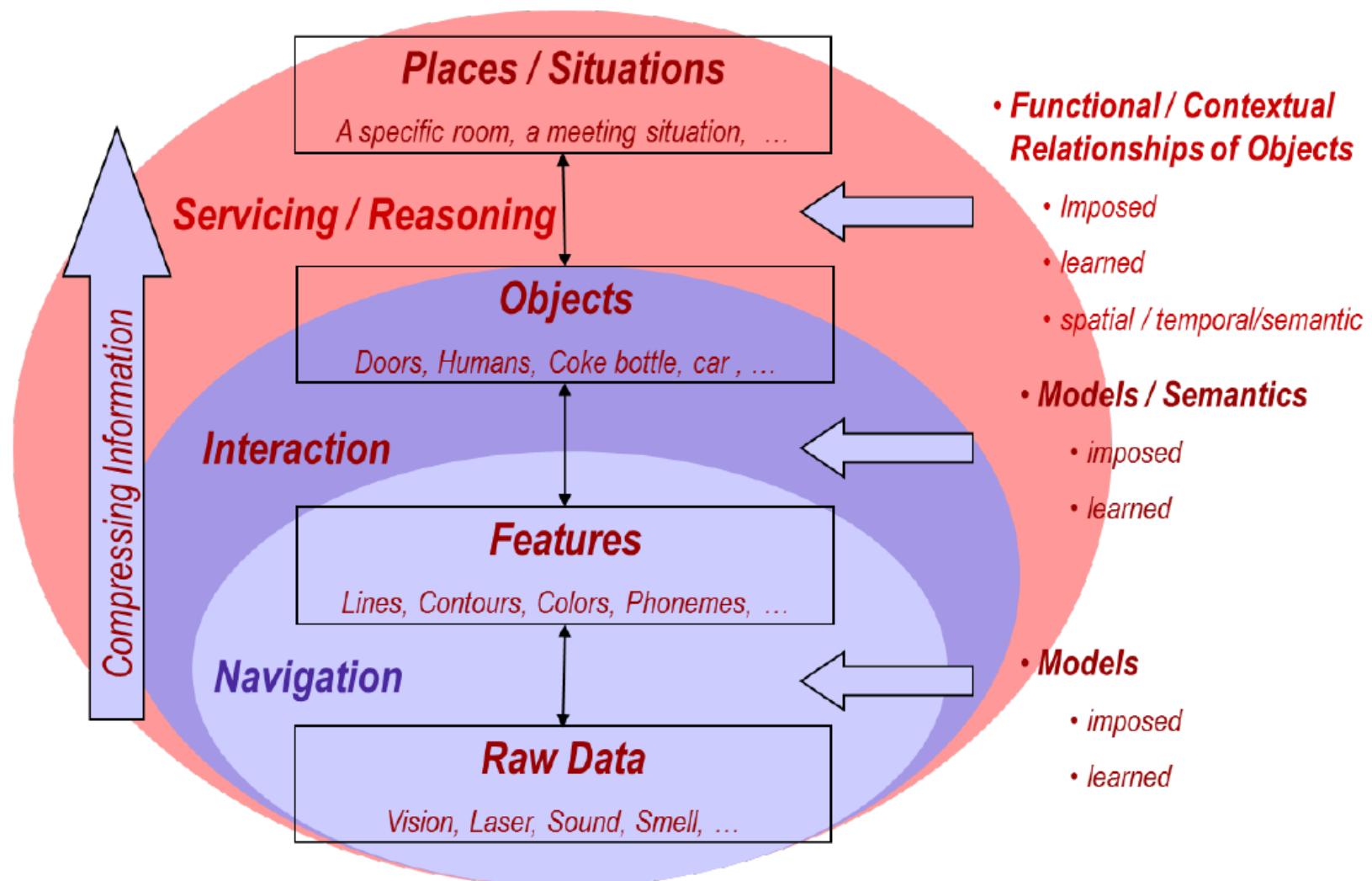
# General Processing



# Preprocessing

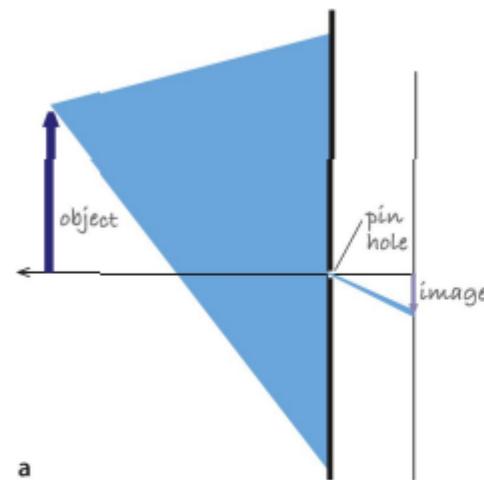
- ‘Cleanup’ the sensor readings before using them
- Noise reduction - filtering
- Re-calibration
- ‘Basic’ stuff - e.g. edge detection in vision
- Change (transform) data representation
- ...

# Information Processing & Integration

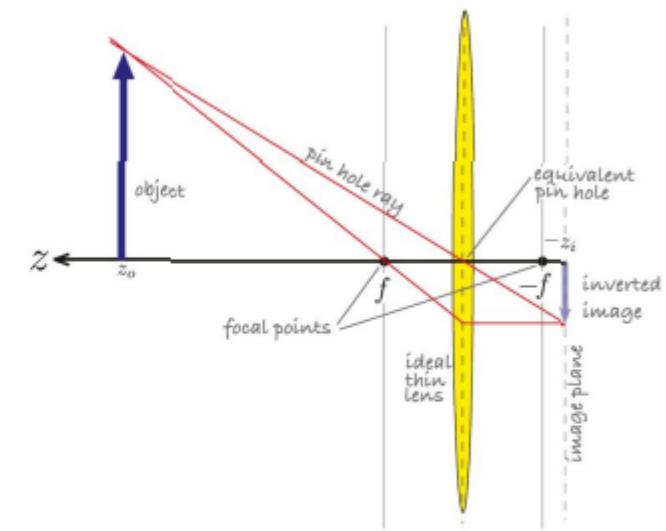


# Computer Vision

# Geometry of Image Formation



pin-hole camera



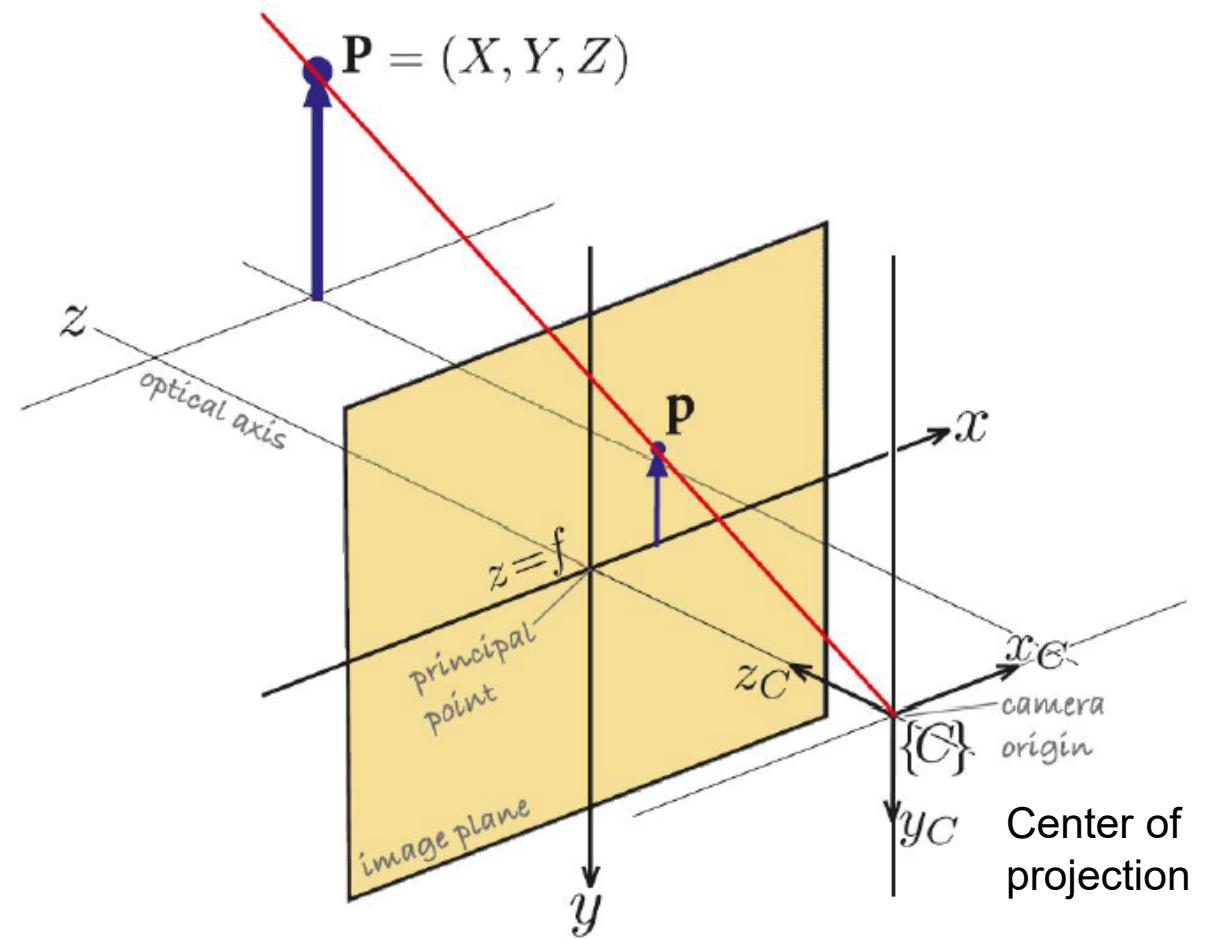
thin lens

# Geometry of Image Formation

- Image: 2D array consisting of *pixels*
- Pinhole camera model

- Perspective projection

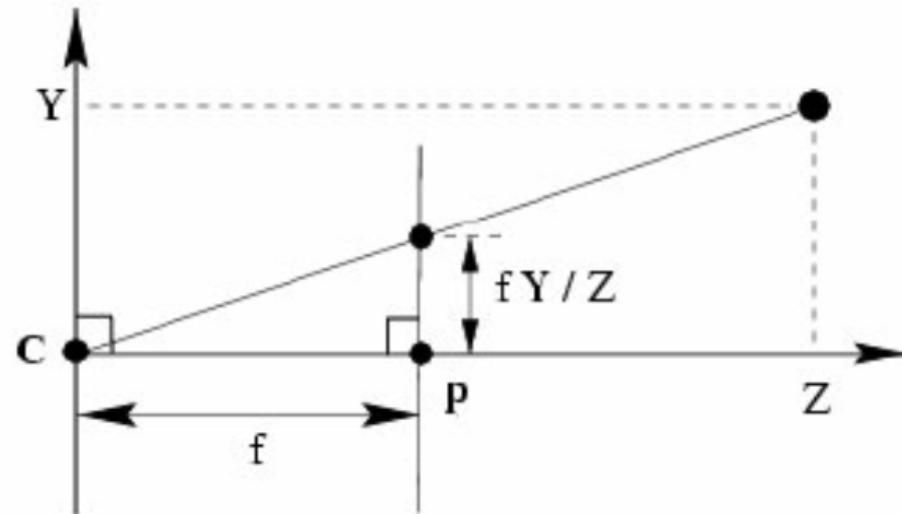
$$k \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix}$$



# Geometry of Image Formation

- Perspective projection

$$k = \frac{\lambda}{z}$$



- Image plane coordinate

$$u = \lambda \frac{x}{z}, \quad v = \lambda \frac{y}{z}$$

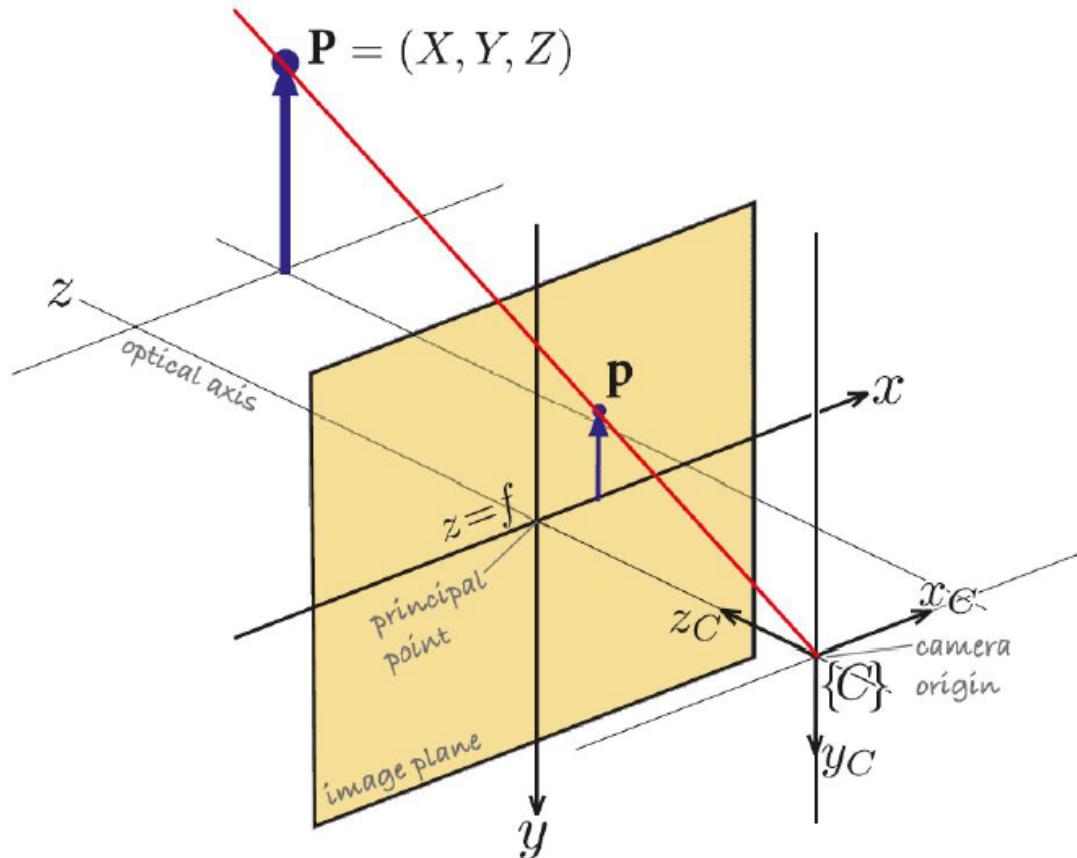
# Image plane and Sensor Array

r, c = pixel coordinates

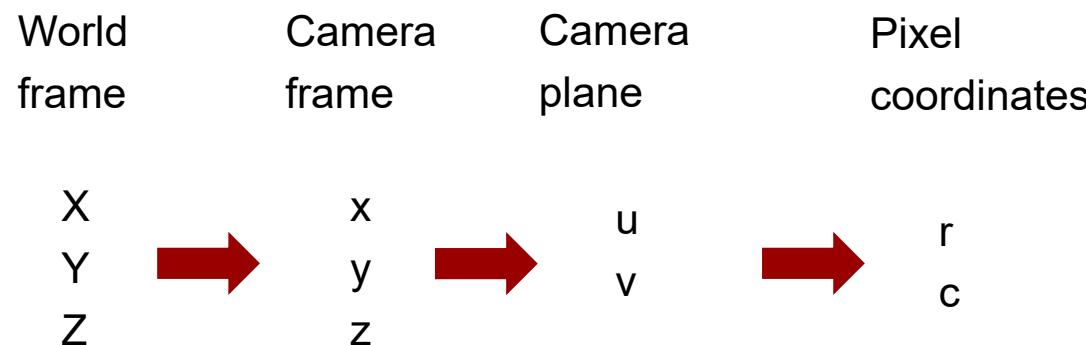
or, oc = coordinates of the pixel that contains the principal point

s<sub>x</sub>, s<sub>y</sub> = horizontal and vertical dimension of a pixel

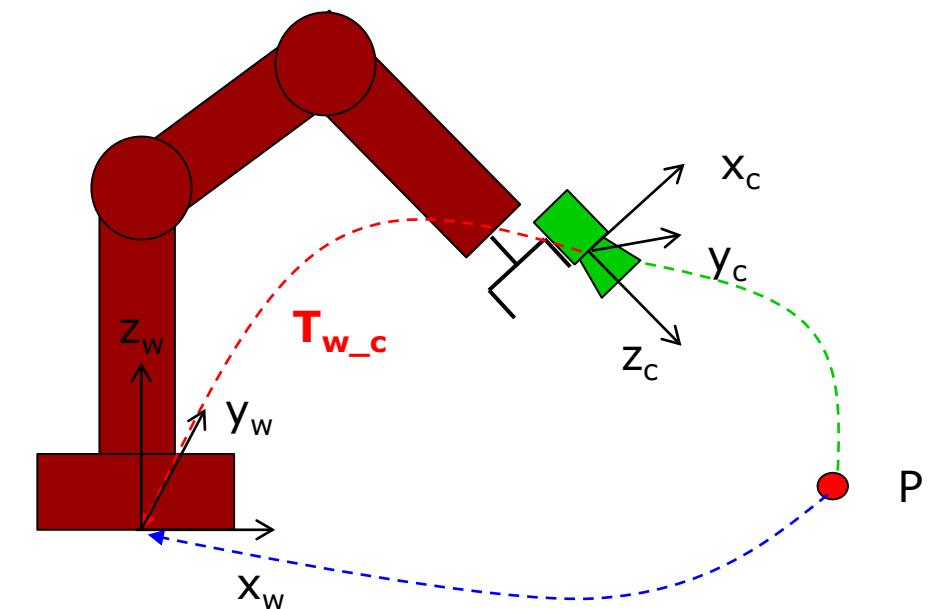
$$-\frac{u}{s_x} = (r - o_r), \quad \frac{v}{s_y} = (c - o_c)$$



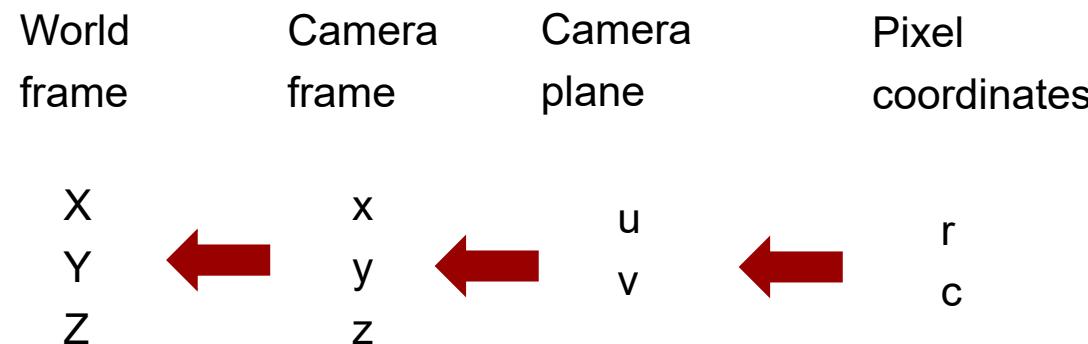
# Forward Projection



How can we describe how a 3D point projects into the camera plane?  
On which pixels will we have the projection of an object in world frame?

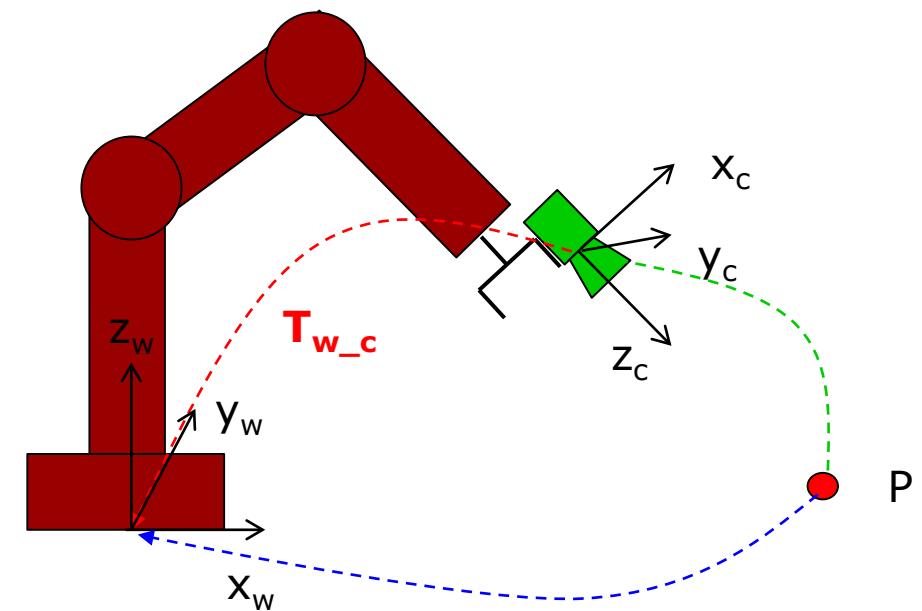


# Backward Projection



Can we reconstruct the position of an object/feature in 3D, from An image in 2D?

- Challenging problem
- Need stereo-vision or motion



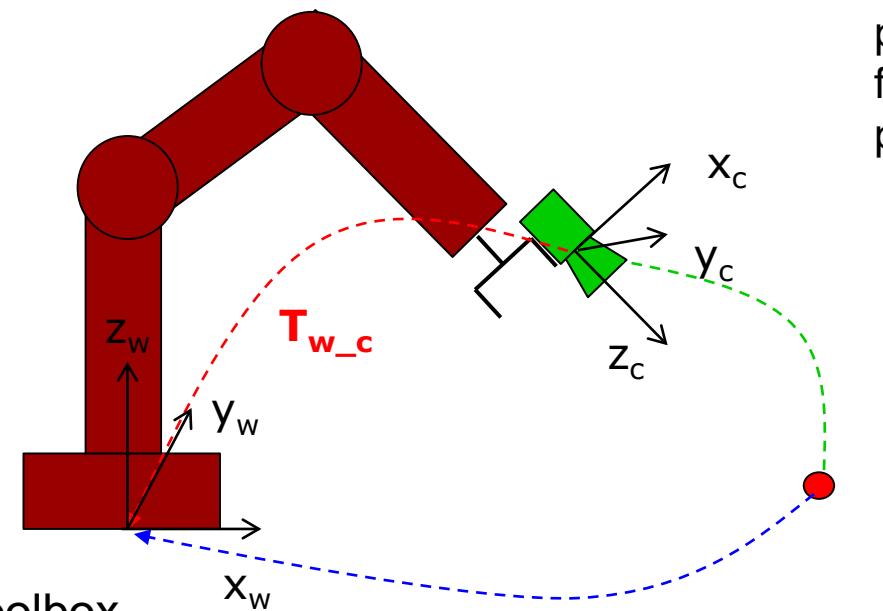
# Intrinsic & Extrinsic Camera Parameters

- Intrinsic parameters:  $f_x, o_r, f_y, o_c$
- Extrinsic parameters: camera → world frame

$$f_x = \frac{\lambda}{s_x}, \quad f_y = \frac{\lambda}{s_y}$$

Once we know the intrinsic camera parameters, we can compute  $u, v$ , from  $(x, y, z)$  – coordinates of a 3D point relative to the camera frame

$$r = -f_x \frac{x}{z} + o_r, \quad c = -f_y \frac{y}{z} + o_c$$



- Camera calibration toolbox

# Intrinsic & Extrinsic Camera Parameters

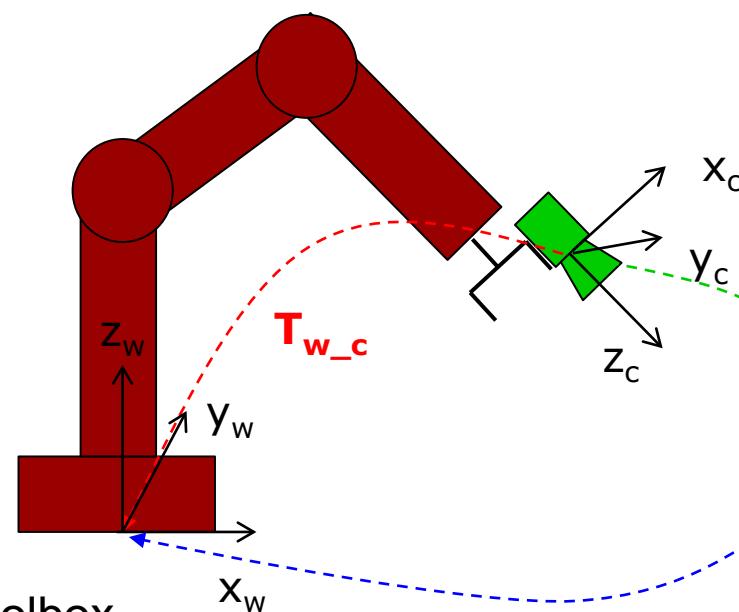
- Intrinsic parameters:  $f_x, o_r, f_y, o_c$
- Extrinsic parameters: camera → world frame

$$x^c = Rx^w + C$$

$$f_x = \frac{\lambda}{s_x}, \quad f_y = \frac{\lambda}{s_y}$$

Once we know the intrinsic camera parameters, we can compute  $u, v$ , from  $(x, y, z)$  – coordinates of a 3D point relative to the camera frame

$$r = -f_x \frac{x}{z} + o_r, \quad c = -f_y \frac{y}{z} + o_c$$

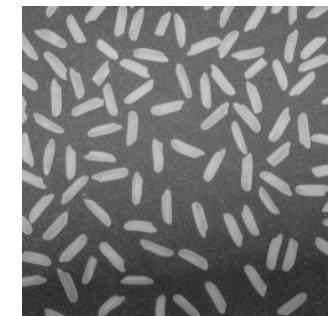


- Camera calibration toolbox

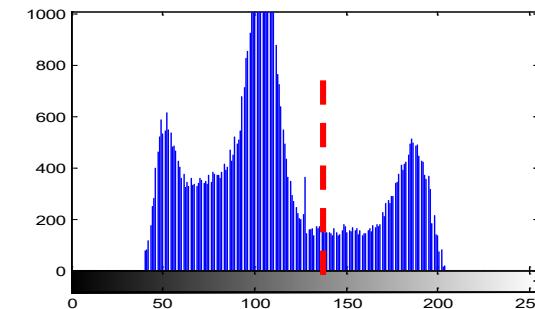
We need to know the extrinsic camera parameters, in order to define the position of an object in the image plane, given the position of the object in world frame

# Segmentation by thresholding

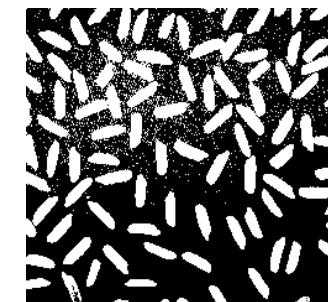
- An gray image: 256 gray levels
- Histogram: occurrence of each gray level value in the image
- Partitioning the image into homogeneous regions: objects & background



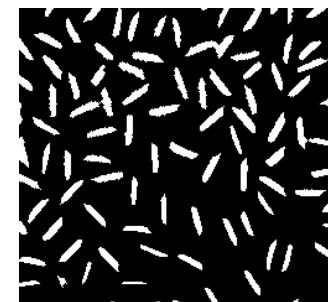
Input image



Histogram



Thresholding



Erosion



Dilation

# Image Features

A corner is defined as the intersection of one or more edges

A corner has high localization accuracy +

Corner detectors are good for VO +

It's less distinctive than a blob -

Methods: Harris, Shi-Tomasi, SUSAN, FAST, etc



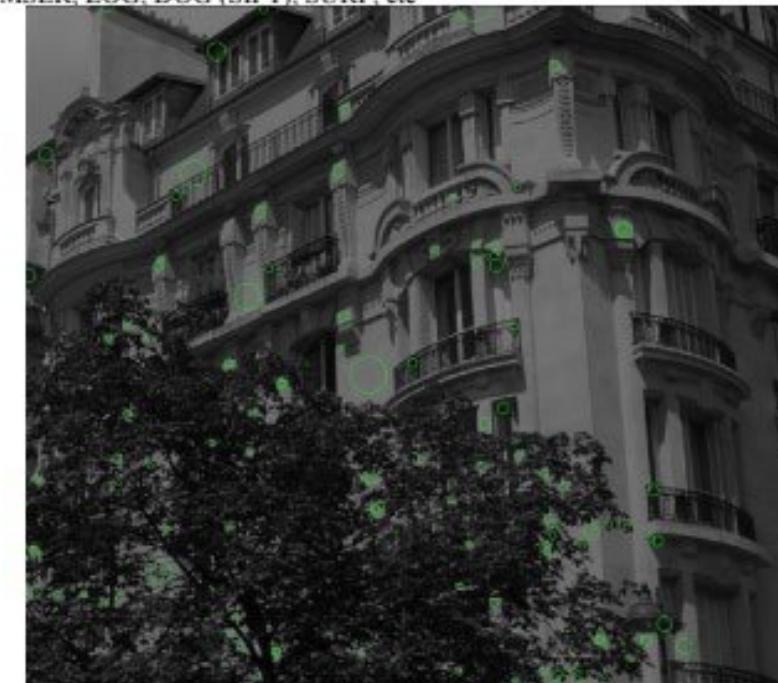
A blob is any other image pattern, which is not a corner, that differs significantly from its neighbors in intensity and texture (e.g., a patch of pixels with similar color, a circle, line, etc.)

Blob detectors are better for place recognition +

It's more distinctive than a corner +

Has less localization accuracy than a corner -

Methods: MSER, LOG, DOG (SIFT), SURF, etc

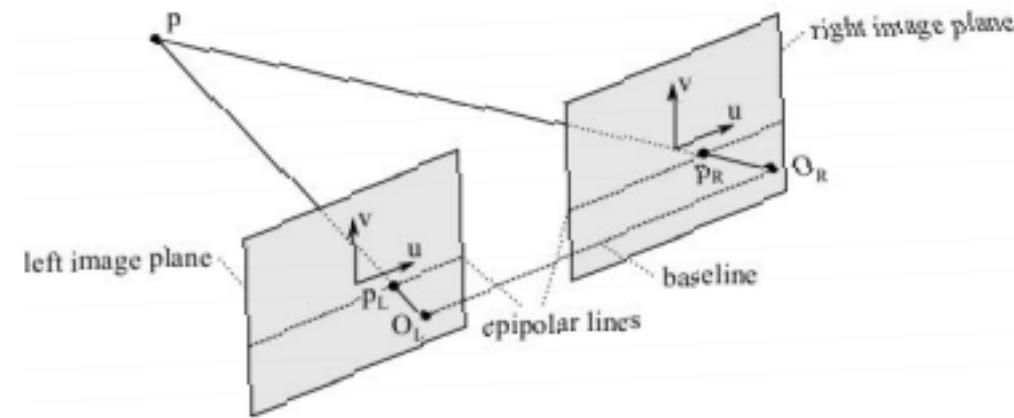


# Depth from Stereo

Both cameras are identical (i.e., same focal length) and are aligned with the x-axis

## Overview

- Camera Calibration
- Image Rectification
- Stereo matching
- Disparity - Triangulation



## Problems

- Camera Calibration errors
- Poor image resolution
- Occlusions
- Violation of brightness consistency
- Low contrast image regions

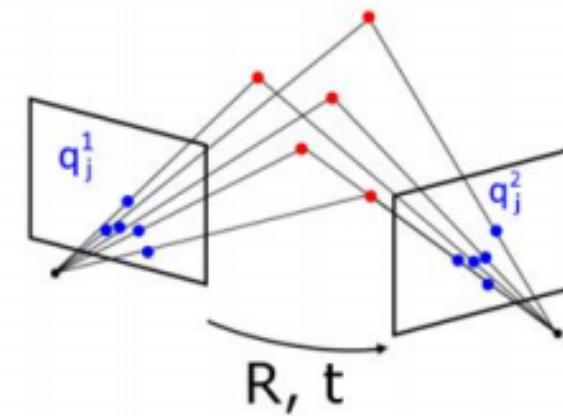
# Depth from Stereo

## Feature extraction

- Detection of feature locations  $q_j \in \mathbb{R}^2$  in both keyframes
- Generation of matched feature sets from different viewpoints
- Removal of outlier features

## Relative Pose estimation

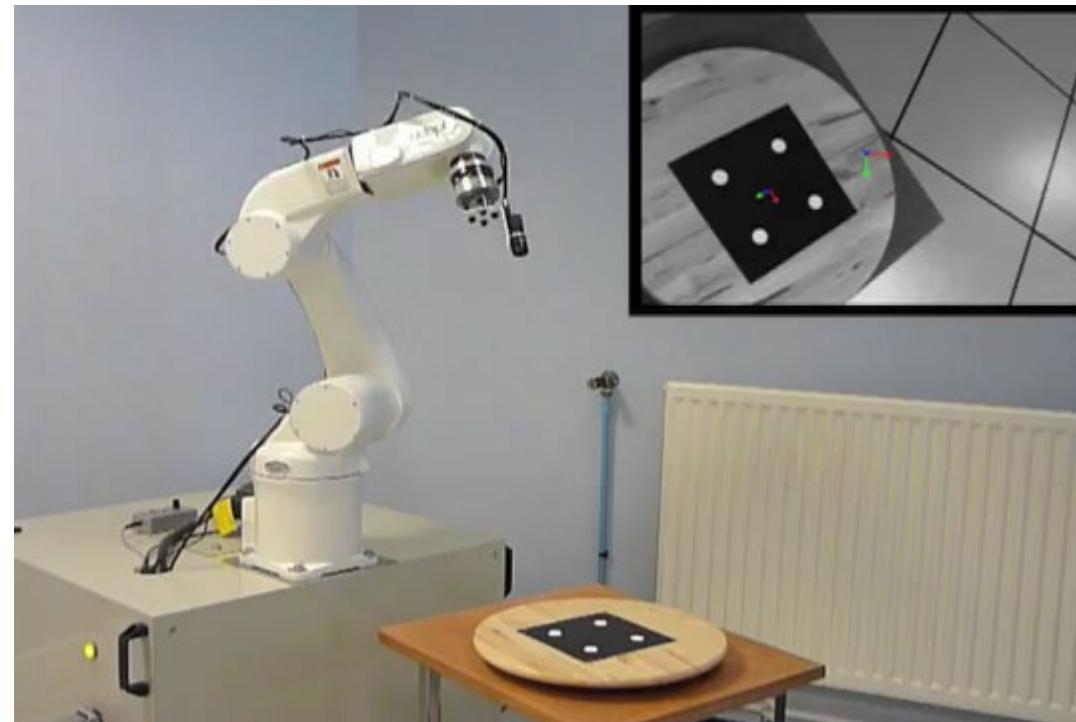
- Frame geometry expressed through Essential matrix  $E$  ( $E = R \times t$ )
- Estimation of  $E$



# Vision-Based Robot Control

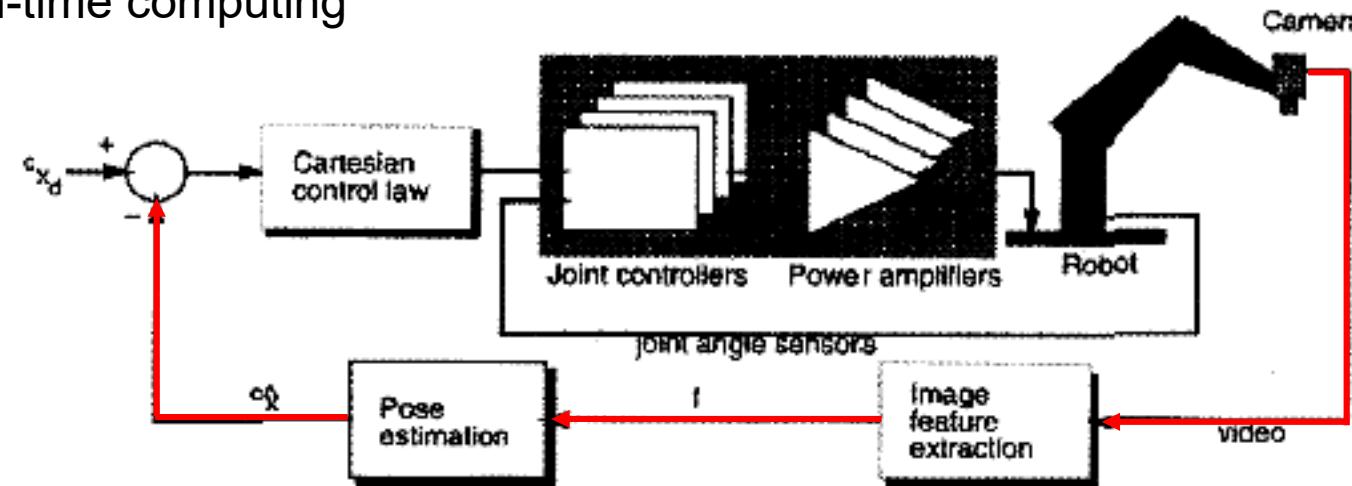
# Visual servoing

How can we track an object, given a camera stream?



# What is visual servoing?

- Utilize visual information in the feedback;
- Better accuracy than ‘Look and Move’ structure
- Fusion of many elemental areas
  - Image processing
  - Kinematics, dynamics
  - Control theory
  - Real-time computing



Figures from S. Hutchinson: A Tutorial on Visual Servo Control

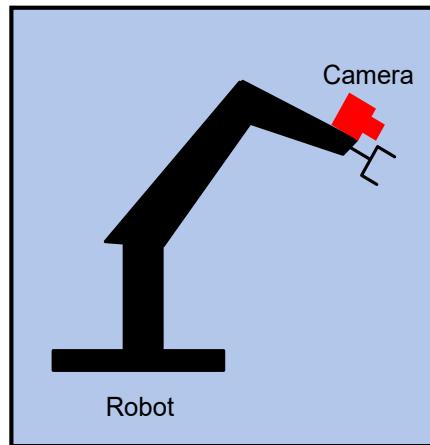
# Example: align and track



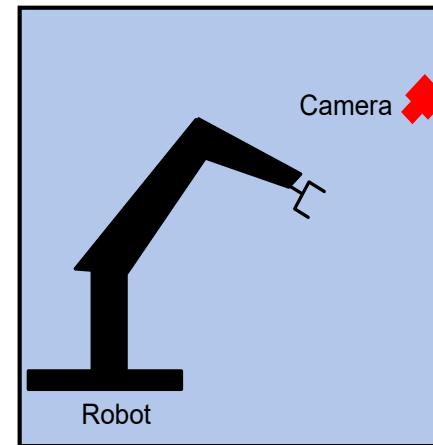
Figures from V. Lippiello, et. al.: Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration

# Visual Servoing

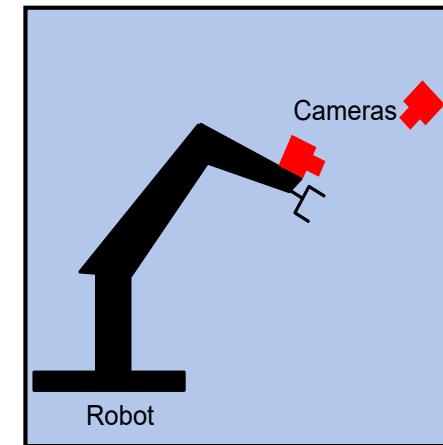
- Camera configurations



Eye in hand



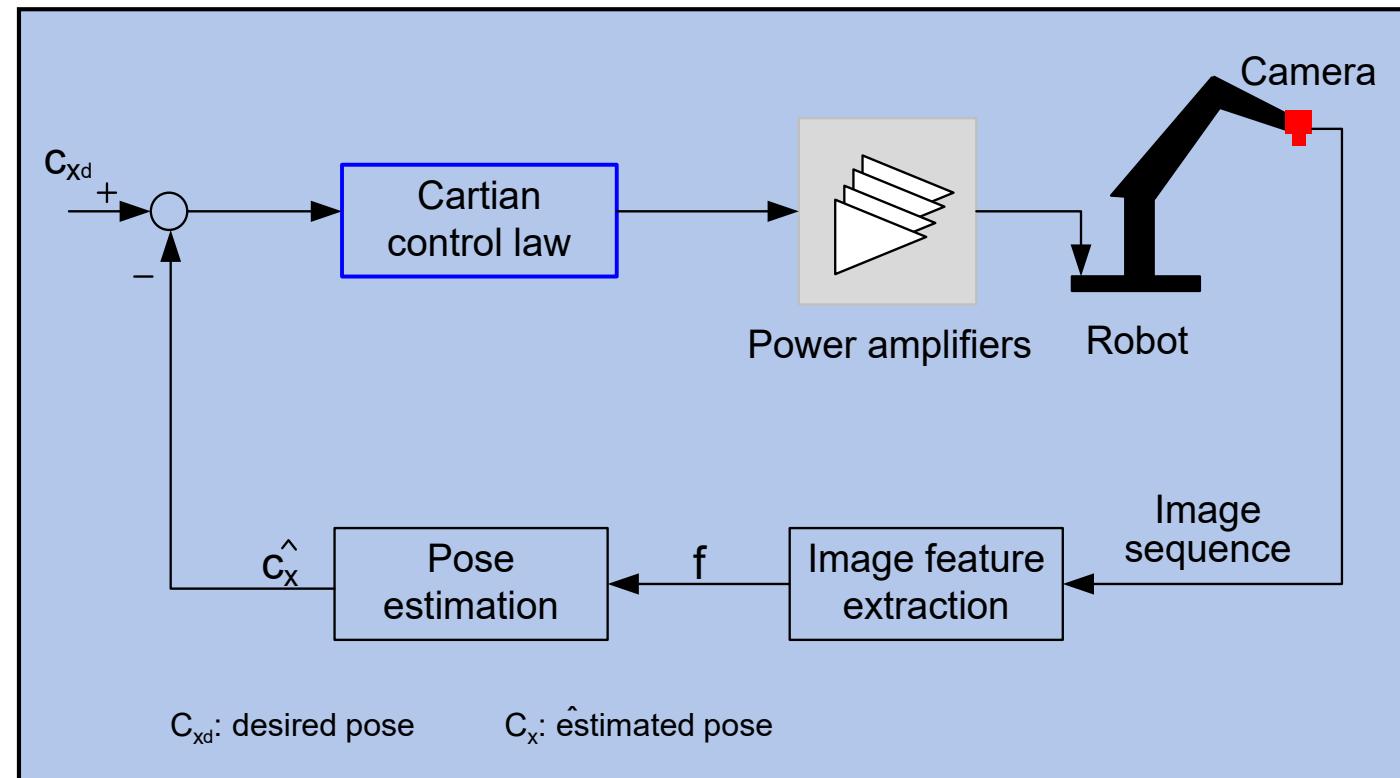
Eye to hand



Eye in hand/Eye to hand cooperation

# Visual Servoing

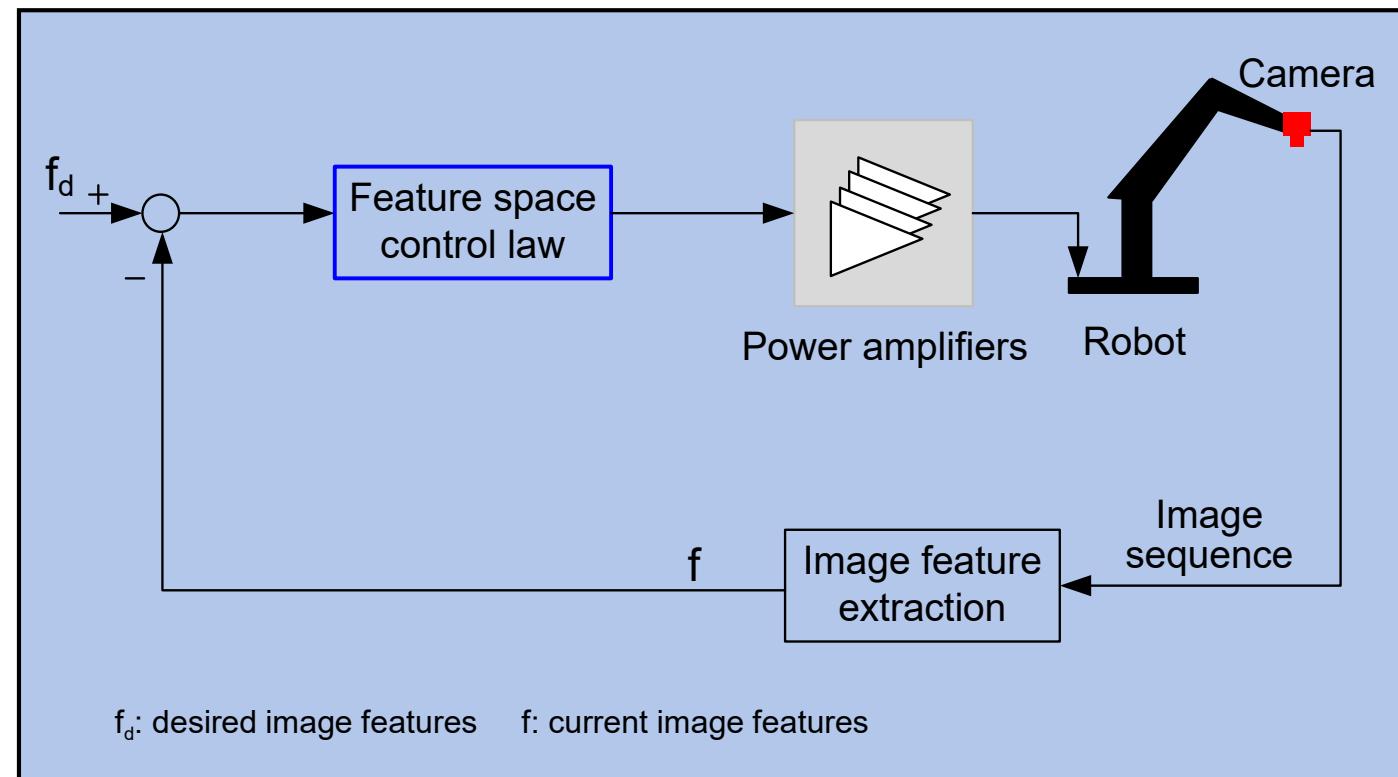
- Servoing architecture: position-based visual servo (PBVS)



Figures modified from S. Hutchinson: A tutorial on visual servo control

# Visual Servoing

- Servoing architecture: image-based visual servo (IBVS)



Figures modified from S. Hutchinson: A tutorial on visual servo control

# Next semester

- Follow up course on Autonomous robot systems (34754)
- Unmanned Autonomous Systems (3-weeks, 34757)
  
- Are you looking for a Master or Bachelor thesis in Robotics?
  - Contact me via email ([mafum@elektro.dtu.dk](mailto:mafum@elektro.dtu.dk))
  - Possibilities with:
    - Drones
    - Physical interaction control
    - Underwater robots

# PERSPECTIVE PROJECTION

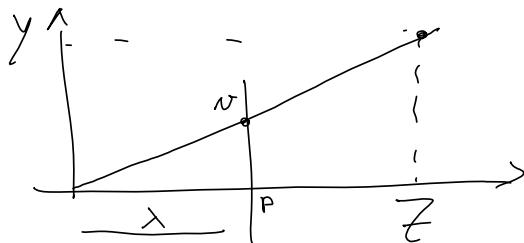


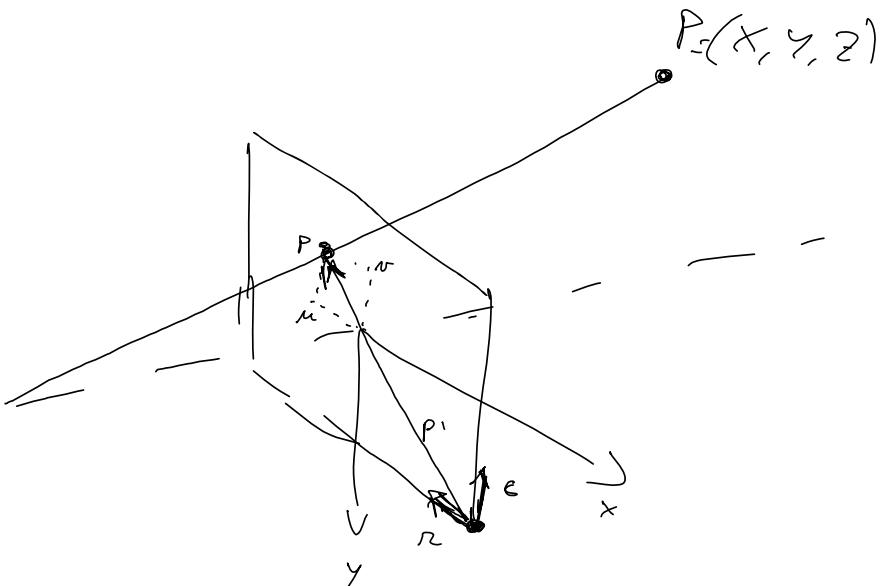
IMAGE  
PLANE

$$k \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ \lambda \end{bmatrix} \rightarrow \begin{array}{l} k_x = u \\ k_y = v \\ k_z = \lambda \end{array} \rightarrow k = \frac{\lambda}{z}$$

$$\Rightarrow \lambda \frac{x}{z} = u$$

$$\lambda \frac{y}{z} = v$$

IMAGE PLANE VS SENSOR ARRAY



$$P = (u, v)$$

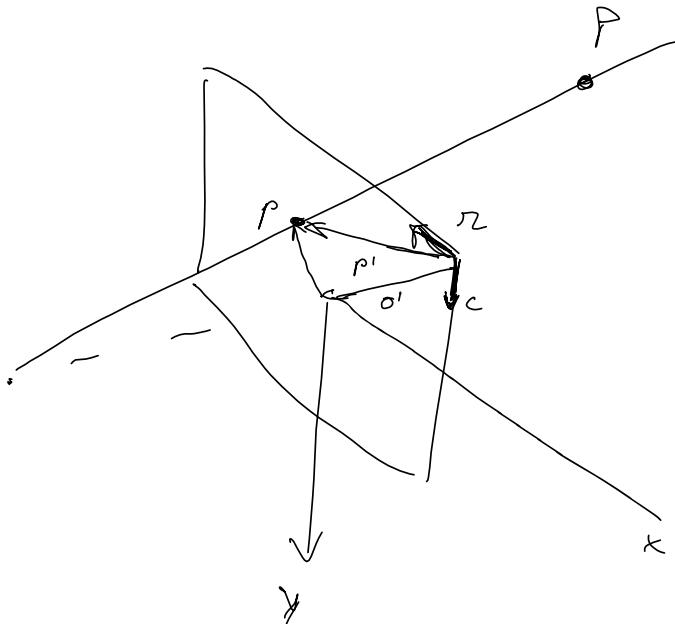
CENTER OF IMAGE PLANE :  $(\sigma_u, \sigma_c)$

$s_x, s_y$  = PIXEL DIMENSION

$$P' = (r, c)$$

$$P = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -s_x(P'_x - \sigma_r) \\ -s_y(P'_y - \sigma_c) \end{bmatrix}$$

# CHANGING OF SENSOR ARRAY FRAME



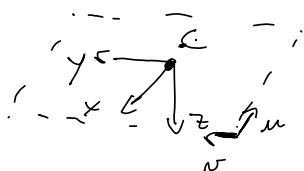
$$P = (u, v)$$

$$P' = (r, c)$$

$$P = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -s_x(r - o_x) \\ s_y(c - o_y) \end{bmatrix}$$

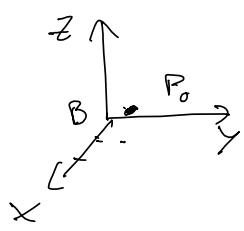
Ex : ROBOT MAS CAMERA AT END-EFFECTOR

$$T_c^B = \begin{bmatrix} 1 & 0 & 0 & 0.3 \\ 0 & -1 & 0 & 0.1 \\ 0 & 0 & -1 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$T_o^B = T_c^B \cdot T_o^c$$

$$T_o^c = \begin{bmatrix} R_o^c & P_o^c \\ 0^T & 1 \end{bmatrix}$$



$$P_o^c = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} -0.2 \\ 0 \\ 0.8 \end{bmatrix}$$

FIND THE PROJECTION ON SENSOR  
CAMERA FRAME

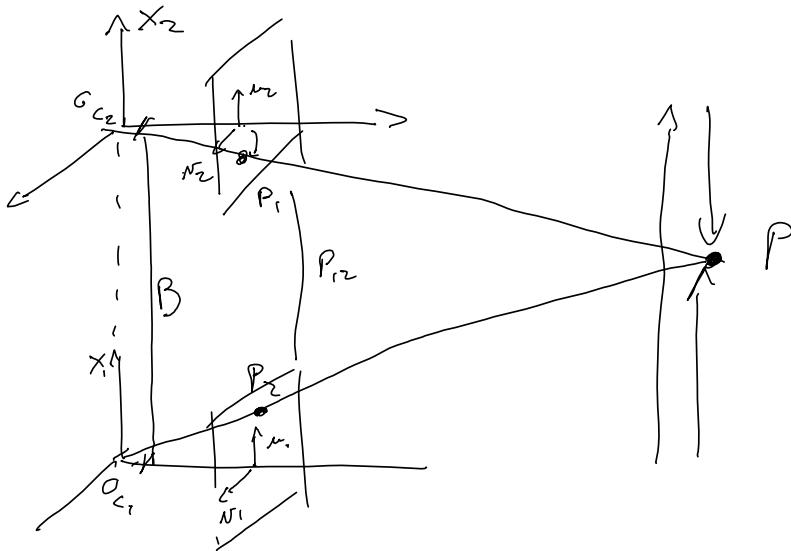
$$u_{pix} = \frac{x}{z} = 500 \cdot \frac{-0.2}{0.8} = -(r - G_n) = -125$$

$$v_{pix} = \frac{y}{z} = 500 \cdot \frac{0}{0.8} = C - O_x = 0$$

$$n - O_n = 125 \quad n = 125 + O_n = 445$$

$$c - O_c = 0 \quad c = O_c \quad 240$$

$E \times 2$



$$T_{C_2}^{C_1} = \begin{bmatrix} 1 & B \\ 0 & 1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \quad B = 0.1 \text{ m}$$

$$\lambda = 1 \text{ m}$$

$$P_1 = (0.05, 0)$$

$$P_2 = (-0.05, 0)$$

X COORD FGR CAN 1 IS EQUAL TO X COORD  
MINUS B

$$\frac{(m_1 - m_2)}{\lambda} = \frac{B}{z} \quad z = \frac{B}{\left(\frac{m_1 - m_2}{\lambda}\right)}$$



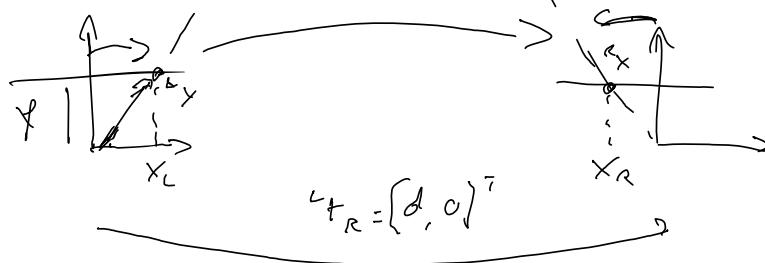
$$x = \lambda_L^L x$$

$$\begin{matrix} & \nearrow \\ / & \backslash \end{matrix} \quad \lambda_R^R x$$

/ \

/ \

/



$$\lambda_R^R x = (d, 0)^T$$

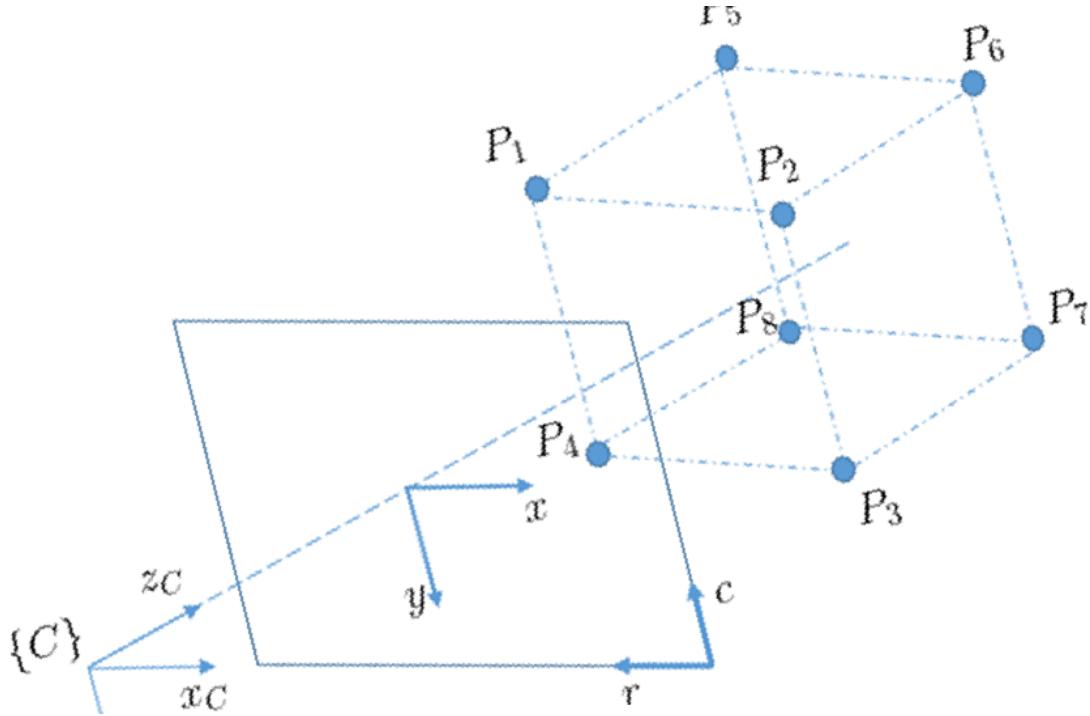
$$x = \lambda_L^L \frac{x}{z}$$

$$x = \lambda_R^R \frac{x}{z}$$



#### Q4.1

A cubic object is placed in front of a camera with frame  $\{C\}$ , as in figure:



The camera parameters are:

$$f = 8 \text{ mm}$$

$$\alpha_x = \frac{1}{s_x} = 79.2 \text{ pix/mm}$$

$$\alpha_y = \frac{1}{s_y} = 120.5 \text{ pix/mm}$$

The frame of reference of the camera is placed in the positive x-y quadrant with respect to frame  $\{C\}$ , at the corner of the CCD, and the distance between the CCD frame and the focal axis is  $o_r = 250$  pix and  $o_c = 250$  pix.

The corners of the cubic object are placed in the following positions w.r.t. the camera frame  $\{C\}$ :

$$P_1 = [-0.1; -0.1; 1] \text{ m}$$

$$P_2 = [0.2; -0.1; 1] \text{ m}$$

$$P_3 = [0.2; 0.2; 1] \text{ m}$$

$$P_4 = [-0.1; 0.2; 1] \text{ m}$$

$$P_5 = [-0.1; -0.1; 2] \text{ m}$$

$$P_6 = [0.2; -0.1; 2] \text{ m}$$

$$P_7 = [0.2; 0.2; 2] \text{ m}$$

$$P_8 = [-0.1; 0.2; 2] \text{ m}$$

Find the projection  $\hat{P}_i$  in pixels on the CCD of all eight points. Choose the correct answer:

Vælg en svarmulighed

$\hat{P}_1 = [313; 346]$  pix

$\hat{P}_2 = [123; 346]$  pix

$\hat{P}_3 = [123; 57]$  pix

$\hat{P}_4 = [313; 57]$  pix

$\hat{P}_5 = [282; 282]$  pix

$\hat{P}_6 = [187; 282]$  pix

$\hat{P}_7 = [187; 187]$  pix

$\hat{P}_8 = [282; 187]$  pix

$\hat{P}_1 = [313; 346]$  pix

$\hat{P}_2 = [123; 346]$  pix

$\hat{P}_3 = [123; 57]$  pix

$\hat{P}_4 = [313; 57]$  pix

$$\begin{aligned}\hat{P}_5 &= [282; 298] \text{ pix} \\ \hat{P}_6 &= [187; 298] \text{ pix} \\ \hat{P}_7 &= [187; 154] \text{ pix} \\ \hat{P}_8 &= [282; 154] \text{ pix}\end{aligned}$$

○  $\hat{P}_1 = [346; 346] \text{ pix}$

$$\hat{P}_2 = [57; 346] \text{ pix}$$

$$\hat{P}_3 = [57; 57] \text{ pix}$$

$$\hat{P}_4 = [346; 57] \text{ pix}$$

$$\hat{P}_5 = [298; 298] \text{ pix}$$

$$\hat{P}_6 = [154; 298] \text{ pix}$$

$$\hat{P}_7 = [154; 154] \text{ pix}$$

$$\hat{P}_8 = [298; 154] \text{ pix}$$

○  $\hat{P}_1 = [313; 313] \text{ pix}$

$$\hat{P}_2 = [123; 313] \text{ pix}$$

$$\hat{P}_3 = [123; 123] \text{ pix}$$

$$\hat{P}_4 = [313; 123] \text{ pix}$$

$$\hat{P}_5 = [282; 282] \text{ pix}$$

$$\hat{P}_6 = [187; 282] \text{ pix}$$

$$\hat{P}_7 = [187; 187] \text{ pix}$$

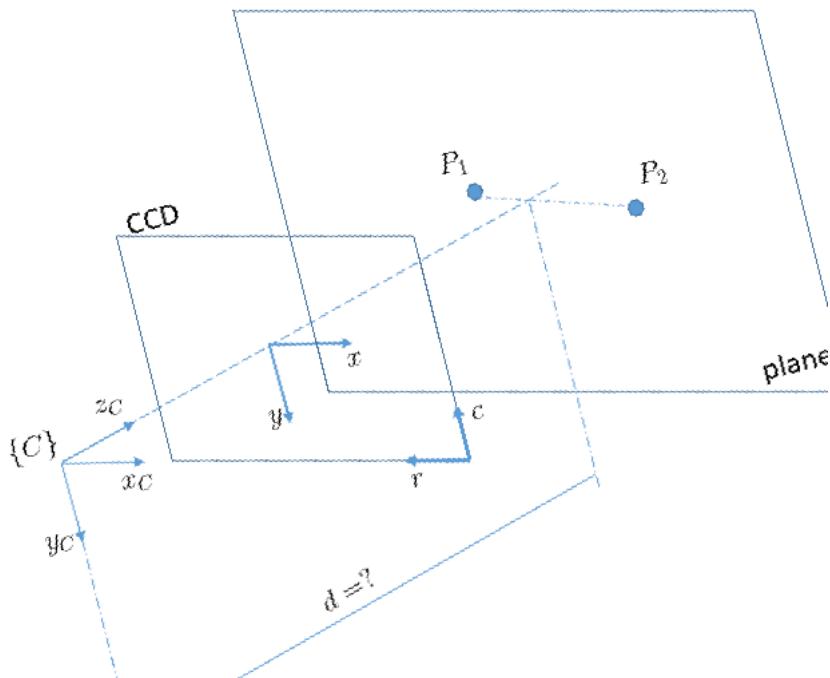
$$\hat{P}_8 = [282; 187] \text{ pix}$$

#### Q4.2

Two points,  $P_1$  and  $P_2$ , are projected on the CCD at pixels  
 $r_1 = 313$  pix,  $c_1 = 250$  pix  
and  
 $r_2 = 155$  pix,  $c_2 = 250$  pix,  
respectively.

Both points lie on a plane parallel to the CCD plane, and the distance  $d$  of the plane where the two points lie, and the camera frame  $\{C\}$  is unknown.

The distance between the two points is  $\|P_1 - P_2\| = 0.3$  m.



The camera parameters are:

$$f = 8 \text{ mm}$$

$$\alpha_x = \frac{1}{s_x} = 79.2 \text{ pix/mm}$$

$$\alpha_y = \frac{1}{s_y} = 120.5 \text{ pix/mm}$$

The frame of reference of the camera is placed in the positive x-y quadrant with respect to frame  $\{C\}$ , at the corner of the CCD, and the distance between the CCD frame and the focal axis is  $o_r = 250$  pix and  $o_c = 250$  pix.

Find, within reasonable precision, the distance  $d$  from the camera origin to the plane of the two points (z coordinate of the points w.r.t. frame  $\{C\}$ )

Vælg en svarmulighed

- $d = 0.78 \text{ m}$
- $d = 1.25 \text{ m}$
- $d = 1.8 \text{ m}$
- $d = 0.95 \text{ m}$
- $d = 1.2 \text{ m}$
- $d = 1 \text{ m}$
- $d = 3 \text{ m}$