Introduction to Robotics

Lecture 1

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

General Information

Lecture:	Friday 10:15 c. t. - 11:45 c. t.
Room:	F-334
Web:	`http://tams.inf...burg.de/lectures/`
Exercises /RPC:	Friday 09:00 c. t. - 11:00 c. t. / Friday 09:00 c. t. - 13:00 c. t. (alternating) see website for dates
Room:	BigBlueButton/F-334/F-304
Name:	Prof. Dr. Jianwei Zhang
Office:	F-330a
E-mail:	`zhang@informatik.uni-hamburg.de`
Consultation:	by arrangement

General Information (cont.)

Name:	Shuang Li
Office:	F-330
Tel.:	+49 40 42883-2504
E-mail:	`sli@informatik.uni-hamburg.de`
Consultation:	by arrangement (eMail)

Name:	Yannick Jonetzko
Office:	F-324
Tel.:	+49 40 42883-2356
E-mail:	`jonetzko@informatik.uni-hamburg.de`
Consultation:	by arrangement (eMail)

General Information (cont.)

Secretary: Tatjana Tetsis
Office: F-311
Tel.: +49 40 42883-2430
E-mail: tetsis@informatik.uni-hamburg.de

- See website for more information

TAMS course website:
http://tams.informatik.uni-hamburg.de/lectures/2020ss/vorlesung/itr

This course is organized with Moodle:
https://lernen.min.uni-hamburg.de/

Related Courses

Lecture

- Intelligent Robotics (winter, Bestmann)
- RoboCup - Playing football with humanoid robots (Summer, Bestmann)
- Lecture Computer Vision I (winter, Frintrop)
- Lecture Computer Vision II (summer, Frintrop)
- Neural Networks (summer, Wermter)

Projects

- Masterproject intelligent robotics (winter, TAMS)
- RoboCup - Playing football with humanoid robots (winter, Bestmann)
- Human-Computer Interaction (winter, Heinecke)

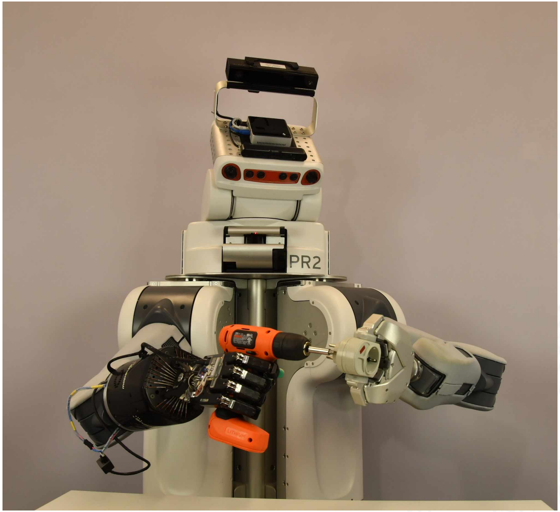
Previous Knowledge

- Linear algebra
  - Essence of linear algebra by 3Blue1Brown
- Basics in physics
  - force, torque, work...
- Related computer skills
  - Linux (RPC)
  - Python (RPC and Excercises)
  - Matlab (Excercises)
  - git (RPC)
  - access to `mafiasi.de` and pool computers

Own Hardware

If you use your own laptop, you require a `Ubuntu 18.06` (Live or Virtual Machine) and fully installed `ros-melodic-desktop-full`

PR2 robot



Content

- Mathematic concepts
  - spatial description
  - kinematics
  - dynamics
- Control concepts
  - movement execution
- Programming aspects
  - ROS, URDF, Kinematics Simulator
- Task-oriented movement and planning

Slides & Dates

24.04.	#01	[EX] Introduction, Coordinate Systems
01.05.	#02	[NO] Kinematics, Robot Description
08.05.	#03	[RPC] Robot Description, Inverse Kinematics
15.05.	#04	[EX] Differential Motion
	#05	[EX] Jacobian
22.05.	#06	[RPC] Trajectory Planning
29.05.	#07	[EX] Trajectory Generation
05.06.	No lecture	(Holiday)
12.06.	#08	[RPC] Dynamics
19.06.	#09	[EX] Robot Control
26.06.	#10	[RPC] Task-oriented Trajectory Generation and Object Representation
03.07.	#11	[EX] Path Planning
10.07.	#12	[RPC] Architectures of Sensor-Based Intelligent Systems
	#LC	[RPC] Summary, Conclusion, Outlook

Outline

Introduction
- Basic Terms
- Degree of Freedom
- Robot Classification

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

Outline (cont.)

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Background of some terms

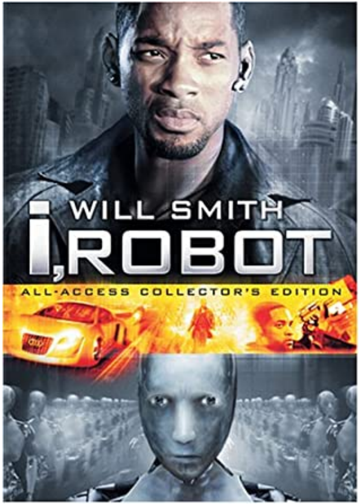
Robot became popular through a stage play by Karel Čapek in 1920, being a capable servant.

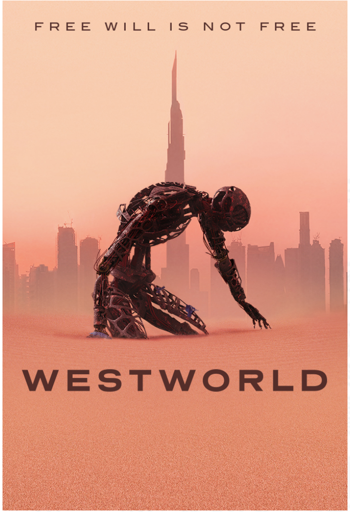
Robotics was first used by Isaac Asimov in 1942.

Three Laws of Robotics

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Obey or not



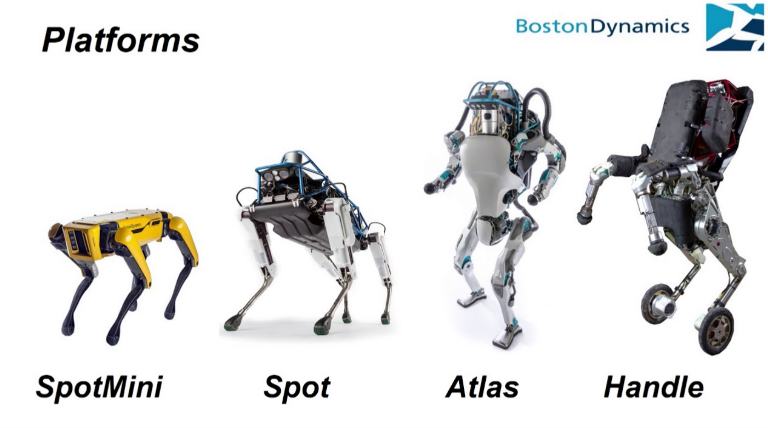


1 2

[1] https://irobot.fandom.com/wiki/I,_Robot_(film)
[2] https://www.rottentomatoes.com/tv/westworld/s03

Advanced robots [1]

Legged-robots in Boston Dynamics

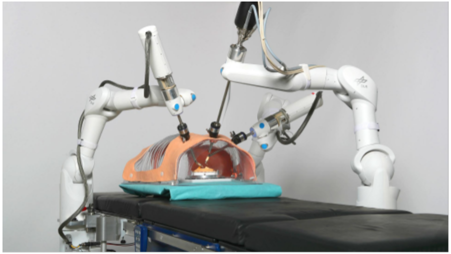


[3]

[3] https://www.youtube.com/watch?v=iZD6hkRwZKM

Advanced robots

Medical Robot



[4] [5] [6]

[4] https://www.dlr.de/content/en/articles/news/2019/02/20190507_dih-hero-a-medical-robotics-network.html

[5] https://newatlas.com/hyundai-robotic-exoskeleton/43331/

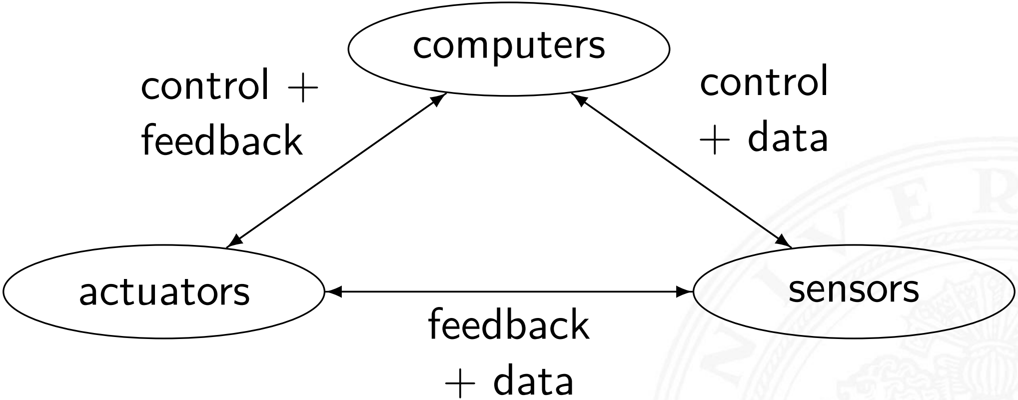
[6] https://www.youtube.com/watch?v=wOzw71j4b78&t=4s

Industrial Robot

7

[7] https://www.robotics.org/blog-article.cfm/Industrial-Robot-Sales-Broke-Records-in-2018/136

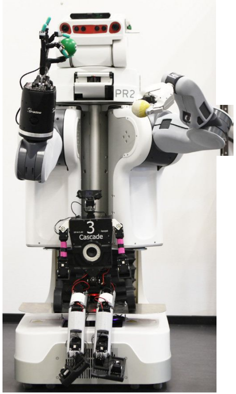
Components of a robot

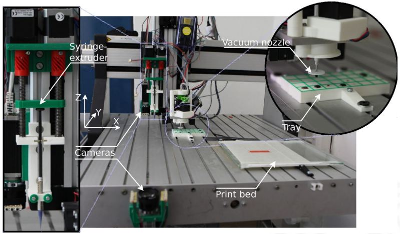


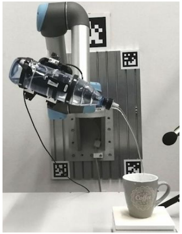
Robotics

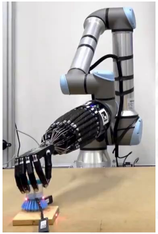
Intelligent combination of computers, sensors and actuators.

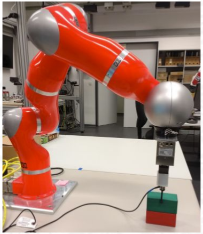
Hardwares in TAMS

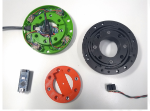










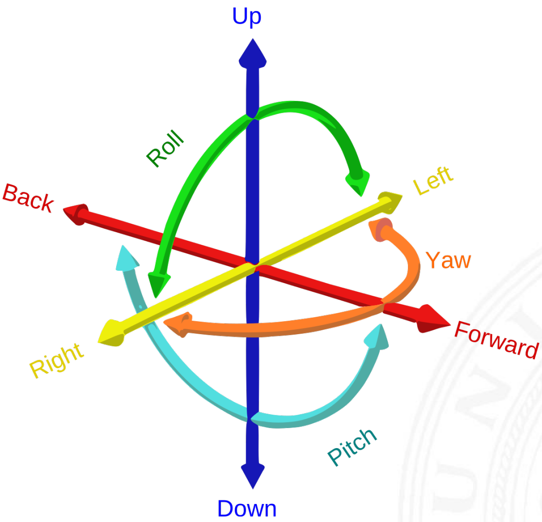


Degree of Freedom (DOF)

The number of variables to determine position of a control system in space.

- Point on a line
- Point on a plane
- Point in space
- Rigid body
  - in space
  - on a plane
- Non-rigid body
- Manipulator
  - number of independently controllable joints

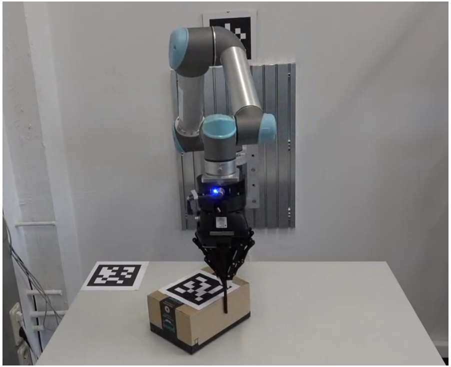
DOF of rigid body

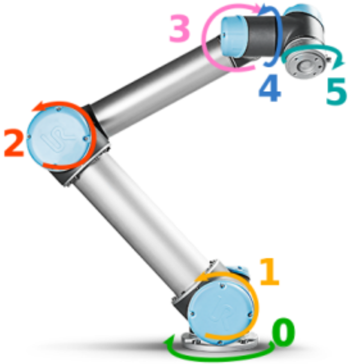


[8]

[8] https://commons.wikimedia.org/wiki/File:6DOF.svg

DOF examples

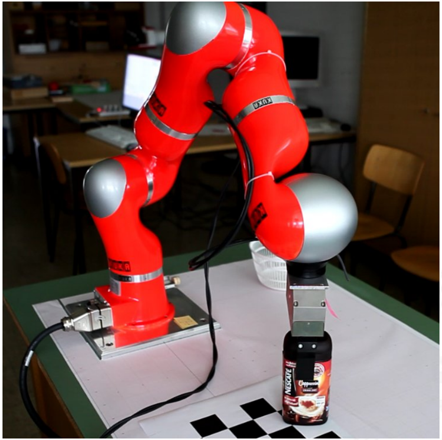




UR5 robot with Robotiq 3-finger gripper

$$\underbrace{6\text{-DOF} + 3\text{-DOF gripper}}_{9}$$

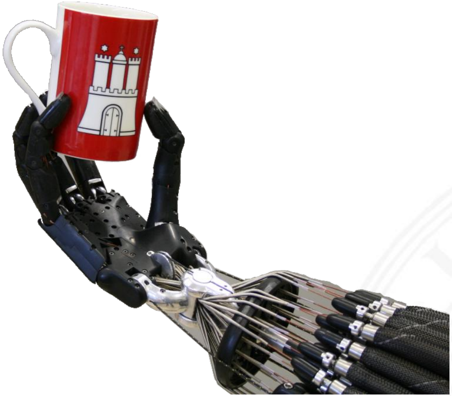
DOF examples (cont.)



KUKA LWR 4+ arm with Schunk gripper

7-DOF + 1-DOF gripper

DOF examples (cont.)



Shadow C5 Air Muscle hand

20-DOF + 4 unactuated joints

DOF examples (cont.)

PR2 service robot with Shadow C6 electrical hand

19-DOF + 20-DOF hand

DOF examples (cont.)

Boston Dynamics Atlas (2020)

28-DOF
[10]

[9]https://studywolf.wordpress.com/2016/08/

[10]https://medium.com/its42/the-reality-of-the-state-of-affairs-in-robotics-fyi-apart-from-the-hyperbole-it-is-sad-2c24a7f560ba

Robot classification by input power source

by input power source

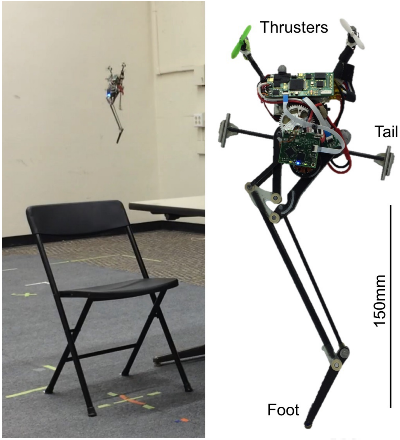
- electrical
- hydraulic
- pneumatic

Robot classification by fileld of work

by field of work

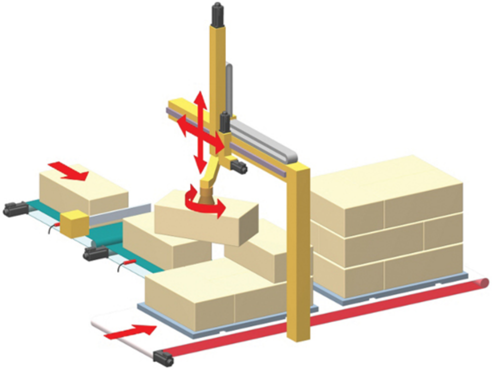
- stationary
  - arms with n DOF
  - multi-finger hand
- mobile
  - portal robot
  - mobile platform
  - running machines and flying robots
  - anthropomorphic robots (humanoids)

Salto Robot [2]



Robot classification by mechanical structure

by mechanical structure



11

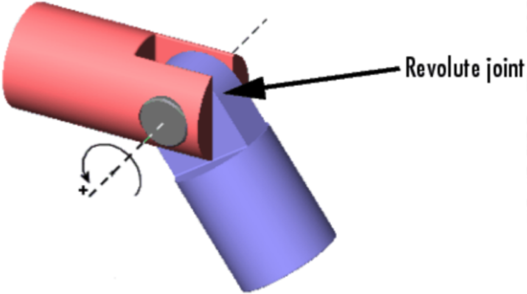
[11] https://www.machinedesign.com/mechanical-motion-systems/article/21831692/the-difference-between-cartesian-sixaxis-and-scara-robots

Type of a joint

- rotatory
  - revolute
- translatory
  - prismatic
- combinations
  - spherical
  - cylindrical
  - planar

Type of a joint

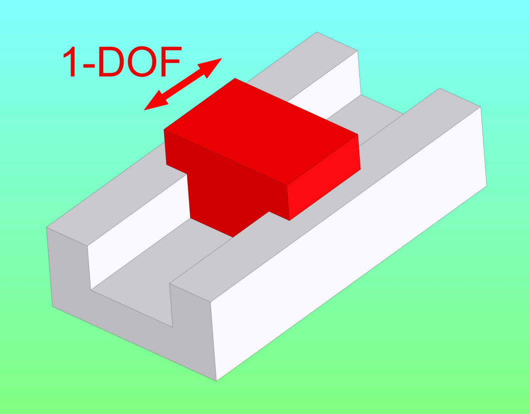
revolute joint



12

[12] https://www.youtube.com/playlist?list=PLrIVgT56nVQ4pm5QFeQ8Z288_VwopKmfW

prismatic joint



13

[13] https://www.youtube.com/playlist?list=PLrIVgT56nVQ4pm5QFeQ8Z288_VwopKmfW

Type of a joint

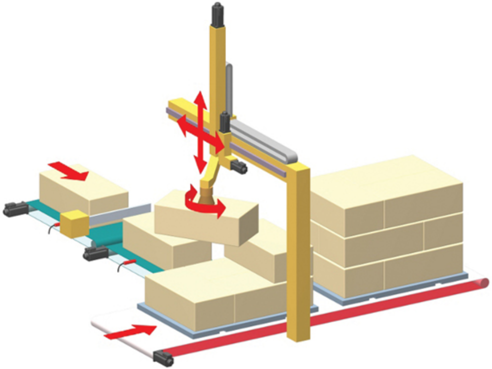
joints with more than one degree of freedom

14

[14] https://www.youtube.com/playlist?list=PLrIVgT56nVQ4pm5QFeQ8Z288_VwopKmfW

Robot classification by mechanical structure

by mechanical structure



Robot classification by mechanical structure

by mechanical structure

- cartesian
- cylindrical
- spherical / polar
- Articulated Robot
- SCARA (Selective Compliance Assembly Robot Arm)

Robot classification by mechanical structure

Selective Compliance Assembly Robot Arm

Task

Please find SCARA robots in the Fanuc industrial robot part.

15

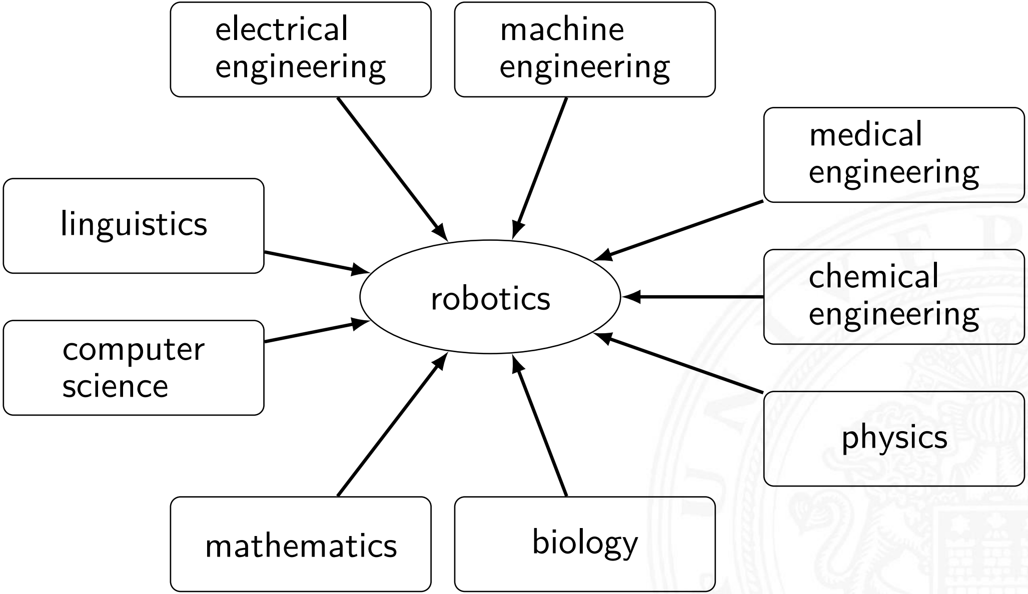
[15] https://www.youtube.com/watch?v=97KX-j8Onu0&t=30s

Robot classification by usage

by usage

- object manipulation
- object processing
- transport
- assembly
- quality testing
- deployment in non-accessible areas
- agriculture and forestry
- underwater
- building industry
- service robot in medicine, housework, ...

An interdisciplinary field

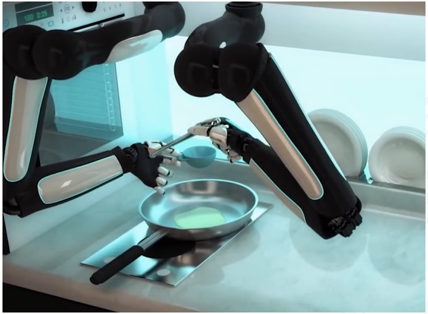


Robotics is Fun!

- A dream of mankind:

  Computers are the most ingenious product of human laziness to date.

computers $\Rightarrow$ robots



16

[16] https://www.youtube.com/watch?v=P1Irm1HlwnQ

Outline

Introduction

Spatial Description and Transformations

- Rigid Body Configuration
- Concatenation of rotation matrices
- Homogenous Transformation
- Transformation Equation

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Outline (cont.)

Path Planning

Task/Manipulation Planning

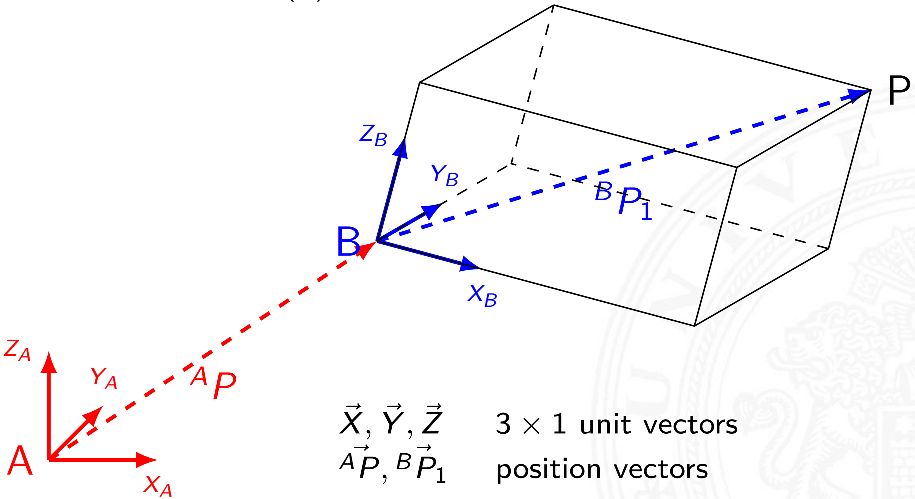
Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

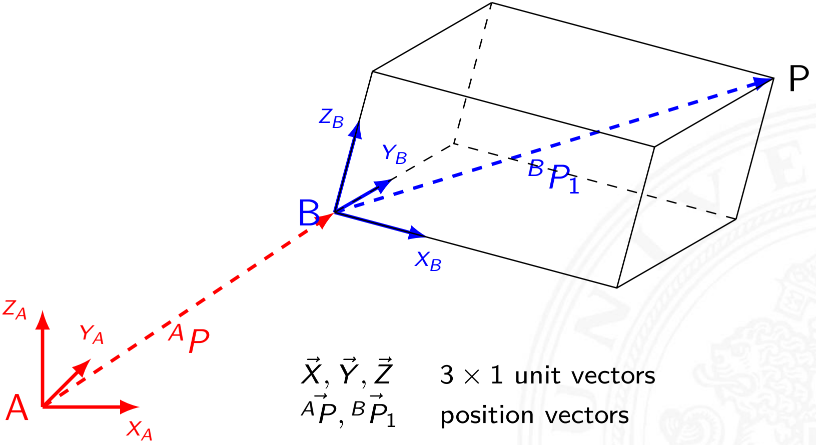
The **pose** of objects, in other words their **position** and **orientation** in Euclidian space can be described through specification of a cartesian coordinate system (B) in relation to a base coordinate system (A).



$\vec{X}, \vec{Y}, \vec{Z}$	3×1 unit vectors
$^A\vec{P}, {}^B\vec{P}_1$	position vectors

Position:

- translation along the axes of the base coordinate system (A)



$\vec{X}, \vec{Y}, \vec{Z}$ 3 × 1 unit vectors
$^A\vec{P}, {^B\vec{P}_1}$ position vectors

- given by position vector $\vec{^A\mathbf{P}} = [^Ap_x, ^Ap_y, ^Ap_z]^T \in \mathcal{R}^3$

Specification of position and orientation

`Orientation` (in space):

- given by Rotation matrix $R_B = [\vec{X}_B \ \vec{Y}_B \ \vec{Z}_B] \in \mathcal{R}^{3\times3}$
- given by Rotation matrix ${}^AR_B = [{}^A\vec{X}_B \ {}^A\vec{Y}_B \ {}^A\vec{Z}_B] \in \mathcal{R}^{3\times3}$

- ${}^AR_B$: the orientation of B with respect to A.
  (Latex: `$^{A}R_{B}$`)
- ${}^A\vec{X}_B, {}^A\vec{Y}_B, {}^A\vec{Z}_B$ are projection of $\vec{X}_B$, $\vec{Y}_B$, $\vec{Z}_B$ in A.

Rotation matrix

Dot product

In terms of the geometric definition, the dot product of two unit vectors $\vec{a}$ and $\vec{b}$ means the projection of the $\vec{a}$ in $\vec{b}$.
$\vec{a} \cdot \vec{b} = \|a\| \|b\| \cos(\theta)$

$$
{}^{A}\vec{X}_B = \begin{bmatrix} \vec{X}_B \cdot \vec{X}_A \\ \vec{X}_B \cdot \vec{Y}_A \\ \vec{X}_B \cdot \vec{Z}_A \end{bmatrix} \quad \text{and} \quad {}^{A}R_B = \begin{bmatrix} {}^{A}\vec{X}_B & {}^{A}\vec{Y}_B & {}^{A}\vec{Z}_B \end{bmatrix}
$$

$$
{}^{A}R_B = \begin{bmatrix} \vec{X}_B \cdot \vec{X}_A & \vec{Y}_B \cdot \vec{X}_A & \vec{Z}_B \cdot \vec{X}_A \\ \vec{X}_B \cdot \vec{Y}_A & \vec{Y}_B \cdot \vec{Y}_A & \vec{Z}_B \cdot \vec{Y}_A \\ \vec{X}_B \cdot \vec{Z}_A & \vec{Y}_B \cdot \vec{Z}_A & \vec{Z}_B \cdot \vec{Z}_A \end{bmatrix}
$$

Inverse of rotation matrix

$$ {}^{A}R_{B} = \begin{bmatrix} \vec{X}_B \cdot \vec{X}_A & \vec{Y}_B \cdot \vec{X}_A & \vec{Z}_B \cdot \vec{X}_A \\ \vec{X}_B \cdot \vec{Y}_A & \vec{Y}_B \cdot \vec{Y}_A & \vec{Z}_B \cdot \vec{Y}_A \\ \vec{X}_B \cdot \vec{Z}_A & \vec{Y}_B \cdot \vec{Z}_A & \vec{Z}_B \cdot \vec{Z}_A \end{bmatrix} $$

${}^{B}X_{A}^{T}$

the projection of $\vec{X}_A$ in B

$$ {}^{A}R_{B} = \begin{bmatrix} {}^{A}\vec{X}_B & {}^{A}\vec{Y}_B & {}^{A}\vec{Z}_B \end{bmatrix} = \begin{bmatrix} {}^{B}\vec{X}_A^T \\ {}^{B}\vec{Y}_A^T \\ {}^{B}\vec{Z}_A^T \end{bmatrix} = \begin{bmatrix} {}^{B}\vec{X}_A & {}^{B}\vec{Y}_A & {}^{B}\vec{Z}_A \end{bmatrix}^T = {}^{B}R_{A}^{T} $$

Inverse of rotation matrix (cont.)

$$
{}^{A}R_{B} = \begin{bmatrix} {}^{A}\vec{X}_{B} & {}^{A}\vec{Y}_{B} & {}^{A}\vec{Z}_{B} \end{bmatrix} = \begin{bmatrix} {}^{B}\vec{X}_{A}^{T} \\ {}^{B}\vec{Y}_{A}^{T} \\ {}^{B}\vec{Z}_{A}^{T} \end{bmatrix} = \begin{bmatrix} {}^{B}\vec{X}_{A} & {}^{B}\vec{Y}_{A} & {}^{B}\vec{Z}_{A} \end{bmatrix}^{T} = {}^{B}R_{A}^{T}
$$

The inverse of a rotation matrix is simply its transpose:

$$
{}^{A}R_{B}^{-1} = {}^{B}R_{A} = {}^{B}R_{A}^{T} \quad \text{and} \quad {}^{A}R_{B}{}^{B}R_{A} = I
$$

whereas I is the identity matrix.

Specification of position and orientation

- Position:
  - given through ${}^{A}\vec{P} \in \mathcal{R}^3$
- Orientation:
  - given through the projection of $\vec{X}_B, \vec{Y}_B, \vec{Z}_B \in \mathcal{R}^3$ of B to the origin system A
  - summarized to rotation matrix ${}^{A}R_B = [{}^{A}\vec{X}_B \; {}^{A}\vec{Y}_B \; {}^{A}\vec{Z}_B] \in \mathcal{R}^{3\times 3}$

$$
{}^{A}R_B = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix}
$$

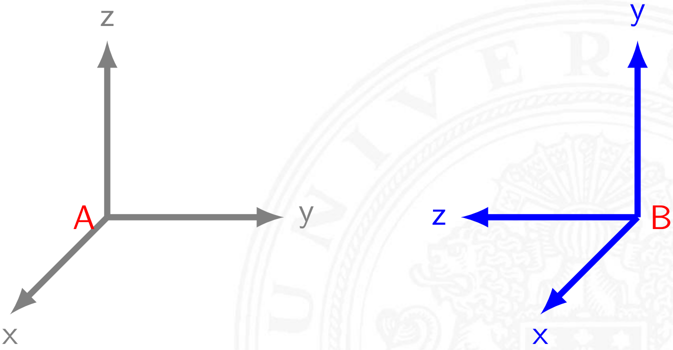
  - redundant, since there are 9 parameters for 3 degrees of freedom

Example of rotation matrix

Write the Rotation matrix of ${}^{A}R_{B}$.

$${}^{A}R_{B} = [{}^{A}\vec{X}_{B} \ {}^{A}\vec{Y}_{B} \ {}^{A}\vec{Z}_{B}]$$

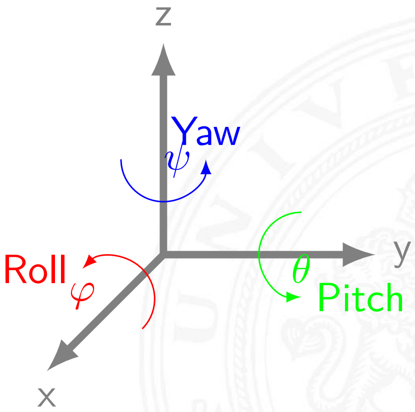
$${}^{A}R_{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$



Rotation by rotation matrix

Sequential multiplication of the rotation matrices by order of rotation.

1. rotation φ (*phi*) around the *x*-axis
   $R_{x,\varphi}$ – Roll
2. rotation θ (*theta*) around the *y*-axis
   $R_{y,\theta}$ – Pitch
3. rotation ψ (*psi*) around the *z*-axis
   $R_{z,\psi}$ – Yaw



Rotatory transformation

(shortened representation: S : sin, C : cos)
The rotation matrix corresponding to a rotation around the x-axis with angle φ (*phi*):

$$R_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi & C\varphi \end{bmatrix}$$

Rotatory transformation (cont.)

The rotation matrix corresponding to a rotation around the y-axis with angle θ (*theta*):

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

Rotatory transformation (cont.)

The rotation matrix corresponding to a rotation around the z-axis with angle ψ (*psi*):

$$R_{z,\psi} = \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Concatenation of rotation matrices

$$R_{\psi,\theta,\varphi} = R_{z,\psi} R_{y,\theta} R_{x,\varphi}$$

$$= \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi & C\varphi \end{bmatrix}$$

$$= \begin{bmatrix} C\psi C\theta & C\psi S\theta S\varphi - S\psi C\varphi & C\psi S\theta C\varphi + S\psi S\varphi \\ S\psi C\theta & S\psi S\theta S\varphi + C\psi C\varphi & S\psi S\theta C\varphi - C\psi S\varphi \\ -S\theta & C\theta S\varphi & C\theta C\varphi \end{bmatrix}$$

Remark: Matrix multiplication is not commutative:

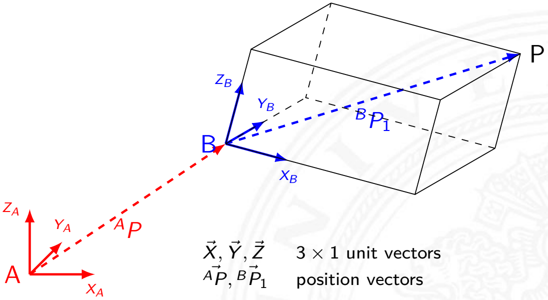
$$AB \neq BA$$

Concatenation of rotation matrices

- `Several rotations` can be multiplied. The following applies:
  - If the rotations are performed in relation to the current, newly defined (or changed) coordinate system, the newly added transformation matrices need to be multiplicatively appended on the right-hand side.
  - If all of them are performed in relation to the fixed reference coordinate system, the transformation matrices need to be multiplicatively appended on the left-hand side.

Mapping: changing descriptions from frame to frame.
For example, change the reference frame of ${}^{B}\vec{P}_1$?

$$
{}^{A}\vec{P}_1 = \begin{bmatrix} {}^{B}\vec{X}_A \cdot {}^{B}\vec{P}_1 \\ {}^{B}\vec{Y}_A \cdot {}^{B}\vec{P}_1 \\ {}^{B}\vec{Z}_A \cdot {}^{B}\vec{P}_1 \end{bmatrix}
= \begin{bmatrix} {}^{B}\vec{X}_A^T \\ {}^{B}\vec{Y}_A^T \\ {}^{B}\vec{Z}_A^T \end{bmatrix} \cdot {}^{B}\vec{P}_1
= {}^{A}R_B {}^{B}\vec{P}_1
$$



$\vec{X}, \vec{Y}, \vec{Z}$	3×1 unit vectors
${}^{A}\vec{P}, {}^{B}\vec{P}_1$	position vectors

Summary: three common uses of a rotation matrix

Three common uses of a rotation matrix:

- represent an orientation
- rotate a vector or frame
- change the frame of reference of a vector or frame

Homogenous transformation

- `Homogeneous transformation matrix:`

$$T = \begin{bmatrix} R & \vec{p} \\ P & S \end{bmatrix}$$

where P depicts the perspective transformation and S the scaling.

- In robotics, $P = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ and $S = 1$. Other values are used for computer graphics.

Homogenous transformation (cont.)

- Combination of $\vec{p}$ and R to $T = \begin{bmatrix} R & \vec{p} \\ \vec{0} & 1 \end{bmatrix} \in \mathcal{R}^{4\times4}$
- Concatenation of several T through matrix multiplication
  - ${}^{A}T_{B}\,{}^{B}T_{C} = {}^{A}T_{C}$
- not commutative, in other words ${}^{B}T_{C}\,{}^{A}T_{B} \neq {}^{A}T_{B}\,{}^{B}T_{C}$

Homogenous transformation (cont.)

They are represented as four vectors using the elements of homogeneous transformation.

$$T = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & p_x \\ r_{12} & r_{22} & r_{32} & p_y \\ r_{13} & r_{23} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Inverse transformation

The inverse of a rotation matrix is simply its transpose:

$$R^{-1} = R^T \text{ and } RR^T = I$$

whereas I is the identity matrix.

The inverse of (1) is:

$$T^{-1} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -\mathbf{p}^\mathbf{T} \cdot \mathbf{r}_1 \\ r_{21} & r_{22} & r_{23} & -\mathbf{p}^\mathbf{T} \cdot \mathbf{r}_2 \\ r_{31} & r_{32} & r_{33} & -\mathbf{p}^\mathbf{T} \cdot \mathbf{r}_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

whereas $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ and $\mathbf{p}$ are the four column vectors of (1) and $\cdot$ represents the dot product of vectors.

Translatory transformation

A translation with a vector $[p_x, p_y, p_z]^T$ is expressed through a transformation:

$$T_{(p_x, p_y, p_z)} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotatory transformation

The transformation corresponding to a rotation around the x-axis with angle φ (*phi*):

$$T_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\varphi & -S\varphi & 0 \\ 0 & S\varphi & C\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotatory transformation (cont.)

The transformation corresponding to a rotation around the y-axis with angle θ (*theta*):

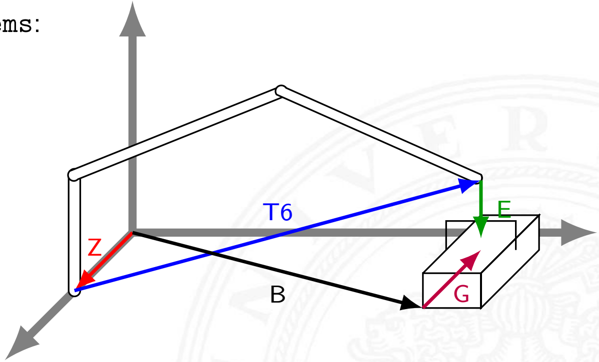
$$T_{y,\theta} = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotatory transformation (cont.)

The transformation corresponding to a rotation around the z-axis with angle ψ (*psi*):

$$T_{z,\psi} = \begin{bmatrix} C\psi & -S\psi & 0 & 0 \\ S\psi & C\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

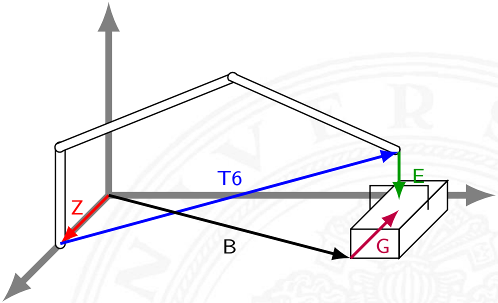
- `Transform of Coordinate systems`:
  frame: a reference S
  typical frames:
  - robot base
  - end effector
  - table (world)
  -
  - object
  - camera
  - ...



Relative transformations

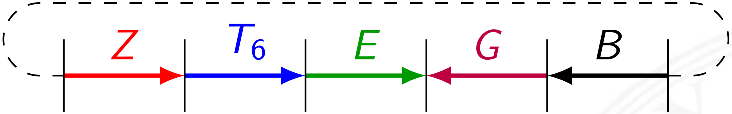
One has the following transformations:

- Z :
  World → Manipulator base
- T_6 :
  Manipulator base → Manipulator end
- E :
  Manipulator end → End effector
- B :
  World → Object
- G :
  Object → End effector



There are two descriptions for the desired end effector pose, one in relation to the object and the other in relation to the manipulator. Both descriptions should equal to each other for grasping:

$$Z T_6 E = B G$$



In order to find the manipulator transformation:

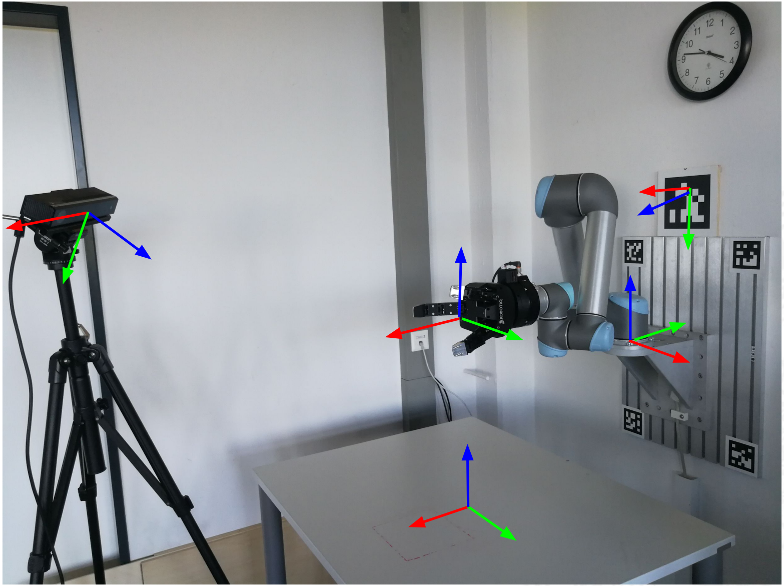
$$T_6 = Z^{-1} B G E^{-1}$$

In order to determine the pose of the object:

$$B = Z T_6 E G^{-1}$$

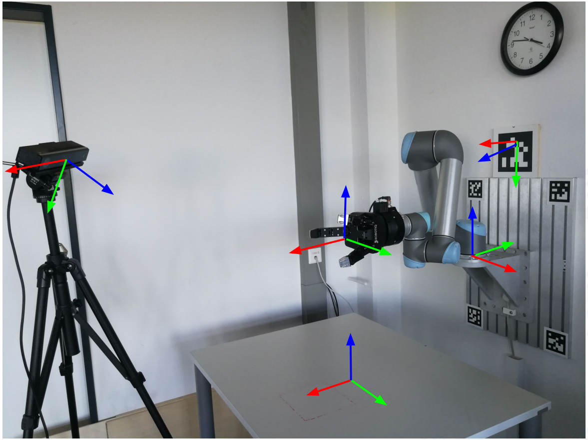
This is also called kinematic chain.

Example: coordinate transformation



Example: coordinate transformation

Given $T_{Base-Apriltag}$, $T_{Camera-Apritag}$, $T_{Camera-Object}$, calculate $T_{Base-Object}$.



$$T_{Base-Object} = T_{Base-Apriltag} \, T^{-1}_{Camera-Apritag} \, T_{Camera-Object}$$

Summary of homogeneous transformations

- `A homogeneous transformation` depicts the position and orientation of a coordinate frame in space.
- `If the coordinate frame` is defined in relation to a solid object, the position and orientation of the solid object is unambiguously specified.
- Three common uses of a transformation matrix: to represent a rigid-body configuration; to change the frame of reference of a vector or a frame; to displace a vector or a frame.

Summary of homogeneous transformations (cont.)

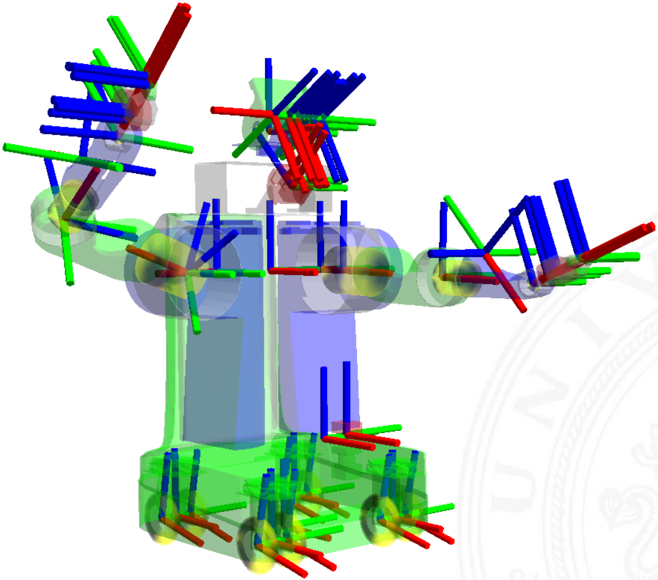
- `Several translations` and rotations can be multiplied.
  - right-hand multiplication $\rightarrow$ in relation to thecurrent, newly defined (or changed) coordinate system, .
  - left-hand multiplication $\rightarrow$ in relation to the fixed reference coordinate system.

Coordinates of a manipulator

- Joint coordinates:
  A vector $\mathbf{q}(t) = (q_1(t), q_2(t), ..., q_n(t))^T$
  (a robot configuration)
- End effector coordinates
  (Object coordinates):
  - A vector $\mathbf{p} = [p_x, p_y, p_z]^T$
  - Rotation matrix:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Outlook



Outlook (cont.)

- Can we use less of 9 parameters to represent the orientation?
- How to construct the transformation matrix of the manipulator's end-effector relative to the base of the manipulator?

Suggestions

- Read (available on google & library):
  - J. F. Engelberger, *Robotics in service*. MIT Press, 1989
  - K. Fu, R. González, and C. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill series in CAD/CAM robotics and computer vision, McGraw-Hill, 1987
  - R. Paul, *Robot Manipulators: Mathematics, Programming, and Control: the Computer Control of Robot Manipulators*. Artificial Intelligence Series, MIT Press, 1981
  - J. Craig, *Introduction to Robotics: Pearson New International Edition: Mechanics and Control*. Always learning, Pearson Education, Limited, 2013
- Repeat your linear algebra knowledge, especially regarding elementary algebra of matrices.

Introduction to Robotics

Lecture 2

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

- More on presentation of a rigid body
- Denavit-Hartenberg convention
- Definition of joint coordinate systems
- Example DH-Parameter of a single joint
- Example DH-Parameter for a manipulator
- Example featuring Mitsubishi PA10-7C

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Outline (cont.)

Robot Control

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Review of last lecture

- Degree of freedom
  - The number of variables to determine position of a control system in space.
- Robot classification
  - mechanical structure
- Rotation matrix
  - ${}^{A}R_{B}^{-1} = {}^{B}R_{A} = {}^{B}R_{A}^{T} \quad \text{and} \quad {}^{A}R_{B}{}^{B}R_{A} = I$
- Homogeneous transformation matrix
  - $T = \begin{bmatrix} R & \vec{p} \\ 0 & 1 \end{bmatrix}$
  - Transformation equation

In order to find the desired end effector pose:

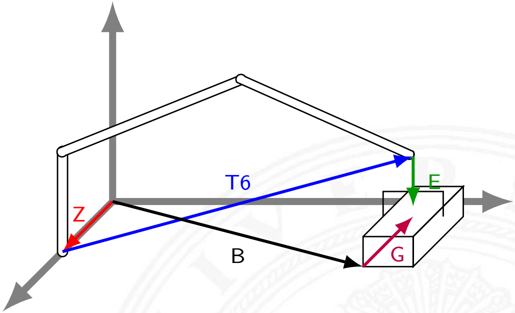
$$Z\,T_6\,E = B\,G$$

In order to find the manipulator transformation T_6:

$$T_6 = Z^{-1}B\,G\,E^{-1}$$

In order to determine the pose of the object B:

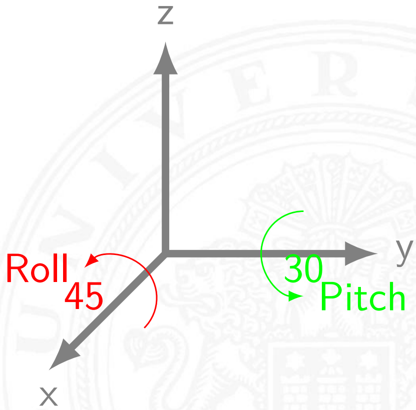
$$B = Z\,T_6\,E\,G^{-1}$$



Review of last lecture

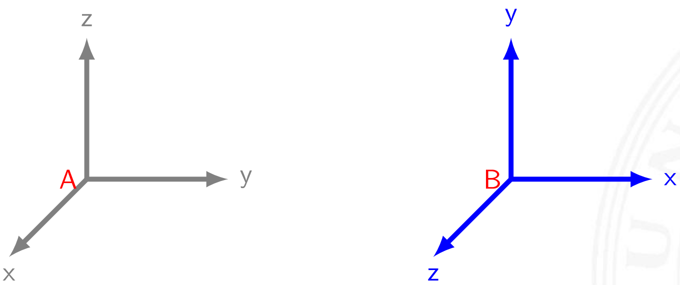
A vector $^{A}\vec{P}$ is rotated about $\hat{Y}$ by 30 degrees and is subsequently rotated about $\hat{X}$ by 45 degrees. Give the rotation matrix that accomplishes these rotations in the given order.

$$
\begin{aligned}
R &= R_{x,45} R_{y,30} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos45 & -sin45 \\ 0 & sin45 & cos45 \end{bmatrix} \begin{bmatrix} cos30 & 0 & sin30 \\ 0 & 1 & 0 \\ -sin30 & 0 & cos30 \end{bmatrix} \\
&= \begin{bmatrix} 0.866 & 0 & 0.5 \\ 0.353 & 0.707 & -0.612 \\ -0.353 & 0.707 & 0.612 \end{bmatrix}
\end{aligned}
$$



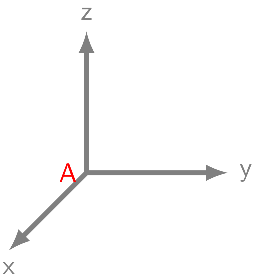
More on presentation of orientation: Euler angles

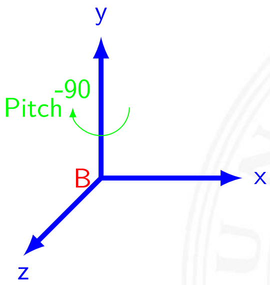
- Euler angles φ, θ, ψ



More on presentation of orientation: Euler angles (cont.)

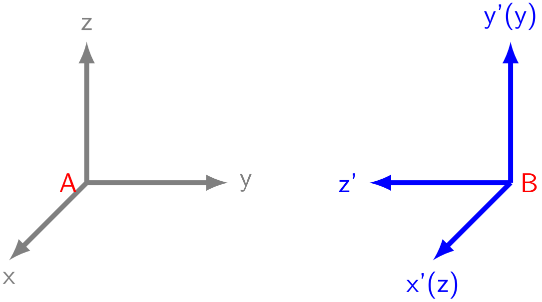
- Euler-angles φ, θ, ψ



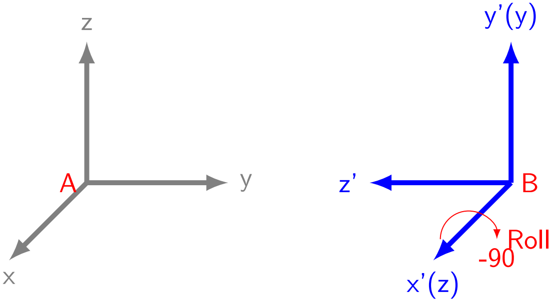


More on presentation of orientation: Euler angles (cont.)

- Euler-angles φ, θ, ψ

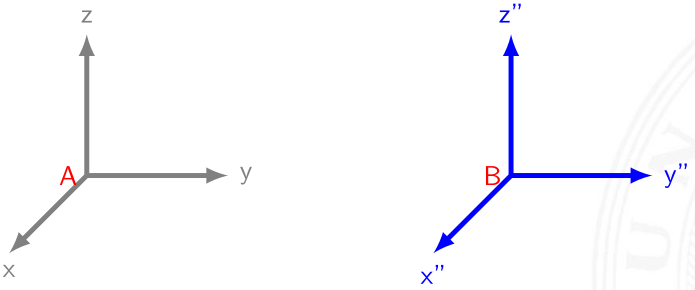


- Euler-angles φ, θ, ψ



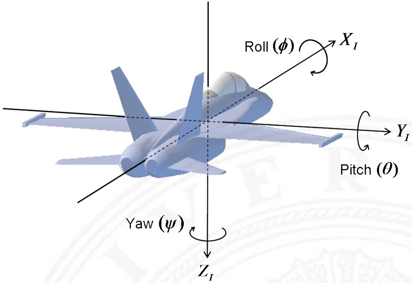
More on presentation of orientation: Euler angles (cont.)

- Euler-angles φ, θ, ψ



More on presentation of orientation: Euler angles (cont.)

- Euler-angles φ, θ, ψ
  - rotations are performed **successively** around the axes, e. g. *ZYX* or *ZXZ* (12 possibilities!)
  - order depends on reference coordinates
  - Intrinsic rotations
  - Extrinsic (fix angle) rotations
- Roll-Pitch-Yaw
  - X-Y-Z fixed angles
  - used in aviation and maritime



Converting Euler Angles to a Rotation Matrix

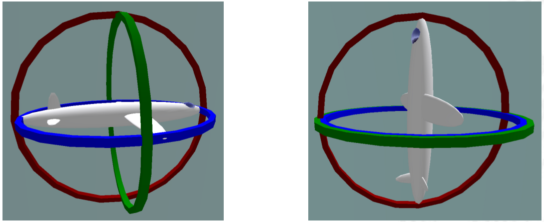
$$R_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\varphi & -S\varphi \\ 0 & S\varphi & C\varphi \end{bmatrix}$$

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

$$R_{z,\psi} = \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

More on presentation of orientation

- Rotation matrix
  - implicit, easy to use linear algebra to perform computation
- Euler angles
  - Gimbal lock!
  - When two gimbals rotate around the same axis, the system loses one degree of freedom.



17

More on presentation of orientation (cont.)

- Rotation matrix
  - implicit, easy to use linear algebra to perform computation, singularity-free
- Euler angles φ, θ, ψ
  - explicit, but gimbal lock/singularity happens
- Equivalent angle-axis representation $R_{k,\theta}$
  - the angle for a rotation about an axis vector
- Quaternion $[x, y, z, w]$
  - 4D vectors that represent 3D rigid body orientations
  - Unit quaternion: $x^2 + y^2 + z^2 + w^2 = 1$

Tools

python: Numpy, pyquaternion
c++: Eigen

[17] https://en.wikipedia.org/wiki/Gimbal_lock

Manipulator

- A manipulator is considered as set of **links** connected by **joints**
  - serial robots (vs.parallel robots)
- Types of joints
  - revolute joints
  - prismatic joints

Forward kinematics

- Movement depiction of the mechanical systems as fixed body chains
- Translate a series of joint parameters $\Longrightarrow$ cartesian pose of the end effector

Purpose

Absolute determination of the position of the end effector (TCP) in the cartesian coordinate system

Tool Center Point (TCP) description

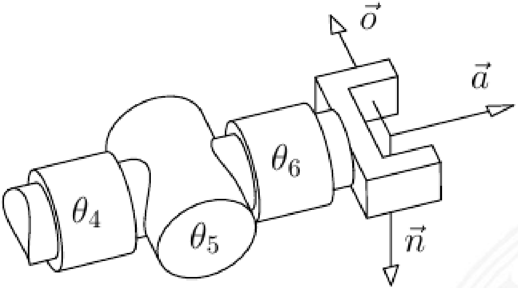
Using a vector $\vec{p}$, the TCP position is depicted.

Three unit vectors:

- $\vec{a}$: (approach vector),
- $\vec{o}$: (orientation vector),
- $\vec{n}$: (normal vector)

specify the orientation of the TCP.

Tool Center Point (TCP) description (cont.)



Thus, the transformation T consists of the following elements:

$$T = \begin{bmatrix} \vec{n} & \vec{o} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Transformation regulation, which describes the relation between joint coordinates of a robot $\mathbf{q}$ and the environment coordinates of the end effector $\mathbf{x}$
- Solely determined by the geometry of the robot
  - Base frame
  - Relation of frames to one another
    $\Longrightarrow$ Formation of a recursive chain
  - Joint coordinates:

$$q_i = \begin{cases} \theta_i & : \text{rotational joint} \\ d_i & : \text{translation joint} \end{cases}$$

Kinematic equations

- In each link, a coordinate frame is attached
- A homogeneous matrix ${}^{i-1}T_i$ depicts the relative translation and rotation between two consecutive joints
  - joint transition
- For a manipulator consisting of six joints:
  - ${}^{0}T_1$: depicts position and orientation of the first link with respect to the base

    $\vdots$
  - ${}^{5}T_6$: depicts position and orientation of the 6th link in regard to link 5

The resulting product is defined as:

$$T_6 = {}^{0}T_1 \, {}^{1}T_2 \, {}^{2}T_3 \, {}^{3}T_4 \, {}^{4}T_5 \, {}^{5}T_6$$

Kinematic description

- Calculation of $T_6 = \prod_{i=1}^{n} T_i$, T_i short for ${}^{i-1}T_i$
  - T_6 defines, how n joint transitions describe 6 cartesian DOF
- Definition of one coordinate system (CS) per segment i
  - generally arbitrary definition
- Determination of one transformation T_i per segment $i = 1..n$
  - generally 6 parameters (3 rotational + 3 translational) required
  - different sets of parameters and transformation orders possible

Solution

Denavit-Hartenberg (DH) convention

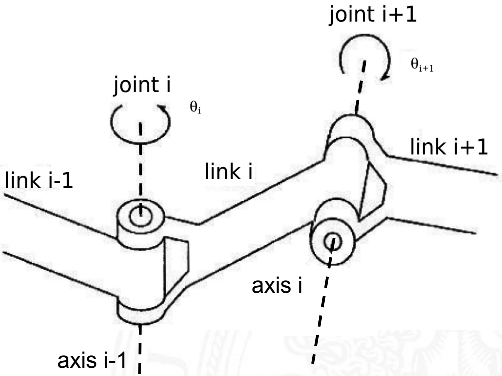
Denavit-Hartenberg convention

- first published by Denavit and Hartenberg in 1955
- established principle
- determination of a transformation matrix T_i using **four** parameters
  - link length, link twist, link offset and joint angle (a_i, α_i, d_i, θ_i)

Parameters for description of two arbitrary links

Two parameters for the description of the link structure i

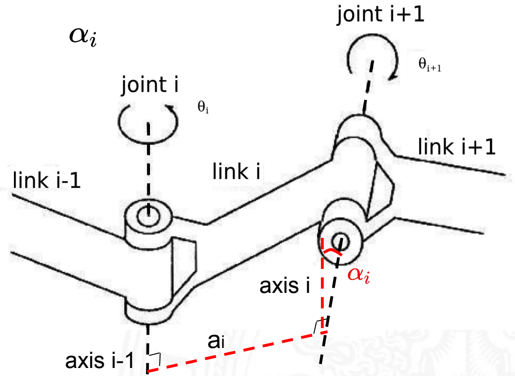
- link length a_i
- link twist α_i



Parameters for description of two arbitrary links

Two parameters for the description of the link structure i

- link length a_i: shortest distance between the axis $i-1$ and the axis i
- link twist α_i: rotation angle from axis $i-1$ to axis i in the right-hand sense about a_i

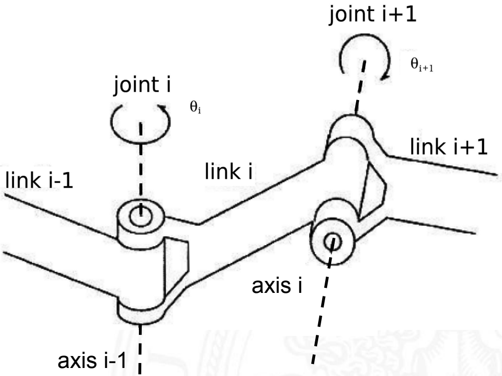


a_i and α_i are constant values due to construction

Parameters for describing two arbitrary links (cont.)

Two for relative distance and angle of adjacent links

- link offset d_i
- joint angle θ_i

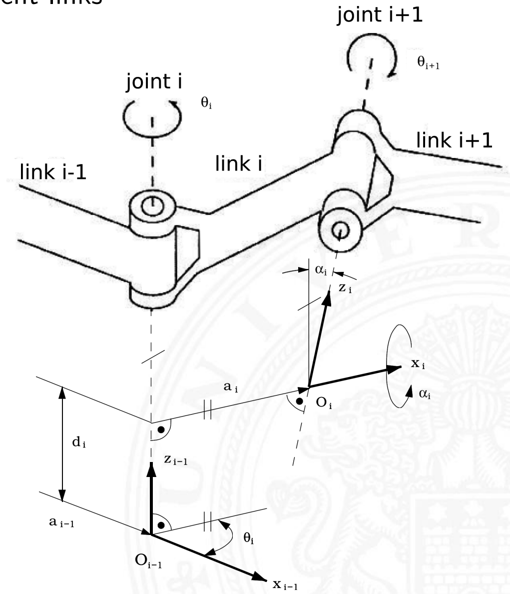


Parameters for describing two arbitrary links (cont.)

Two for relative distance and angle of adjacent links

- link offset d_i: the distance along the common axis $i-1$ from link $i-1$ to the link i
- joint angle θ_i: the amount of rotation about the common axis $i-1$ between the link $i-1$ and the link i

θ_i and d_i are variable
- rotational: θ_i variable, d_i fixed
- translational: d_i variable, θ_i fixed

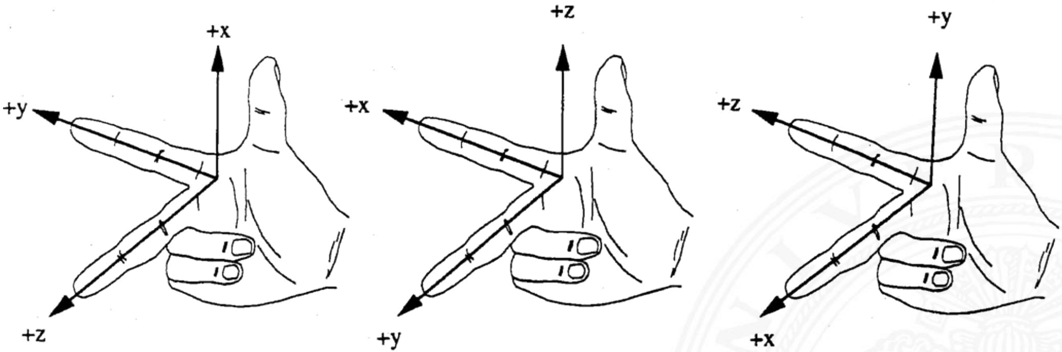


DH Parameters summary

Four DH parameters:
link length, link twist, link offset and joint angle
$(a_i, \alpha_i, d_i, \theta_i)$

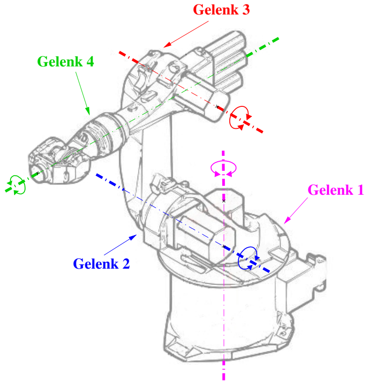
- 3 fixed link parameters
- one joint variable
  - revolute: θ_i variable
  - prismatic: d_i variable
- a_i, α_i: describe the link i
- d_i, θ_i: describe the link's connection

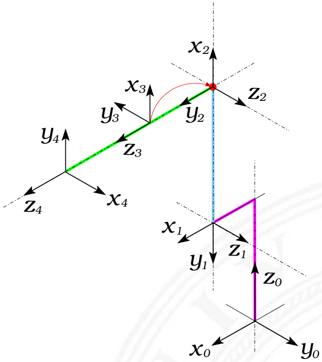
Right-Handed Coordinate System



Configuration 1 Configuration 2 Configuration 3

Definition of joint coordinate systems (classic)

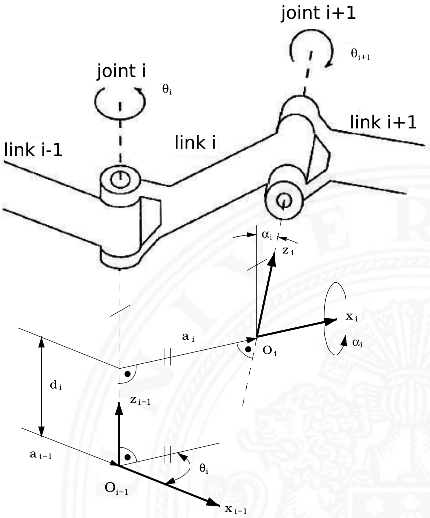




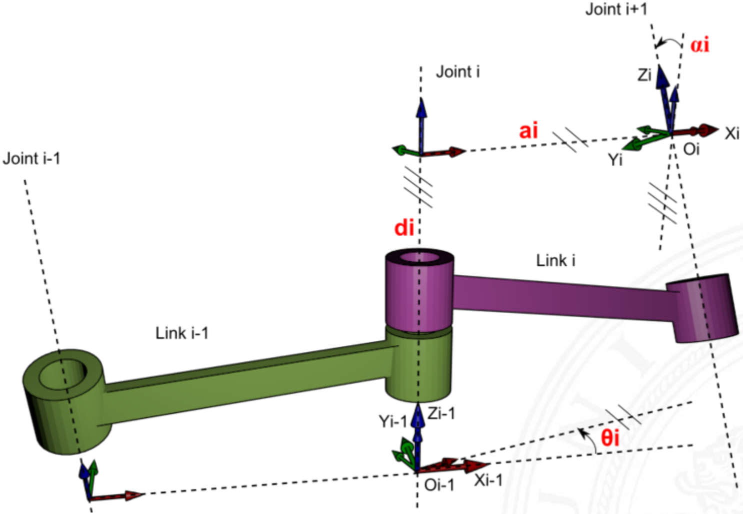
- axis z_{i-1} is set along the axis of motion of the i^{th} joint
- axis x_i is parallel to the common normal of z_{i-1} and z_i $(x_i \parallel (z_{i-1} \times z_i))$.
- axis y_i concludes a right-handed coordinate system
- CS_0 is the stationary origin at the base of the manipulator

DH Parameters

- link length a_i: distance from z_{i-1}-axis to z_i-axis measured along x_i-axis
- link twist α_i: angle from z_{i-1}-axis to z_i-axis measured around x_i-axis
- link offset d_i: distance from x_{i-1} to x_i measured along z_{i-1}-axis
- joint angle θ_i: joint angle from x_{i-1} to x_i measured around z_{i-1}-axis



Classic Parameters



Transformation order

$$T_i = R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(d_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) \rightarrow CS_i$$

Frame transformation for two links (classic)

Creation of the relation between frame i and frame $(i-1)$ through the following rotations and translations:

- `Rotate around` z_{i-1} `by angle` θ_i
- `Translate along` z_{i-1} `by` d_i
- `Translate along` x_i `by` a_i
- `Rotate around` x_i `by angle` α_i

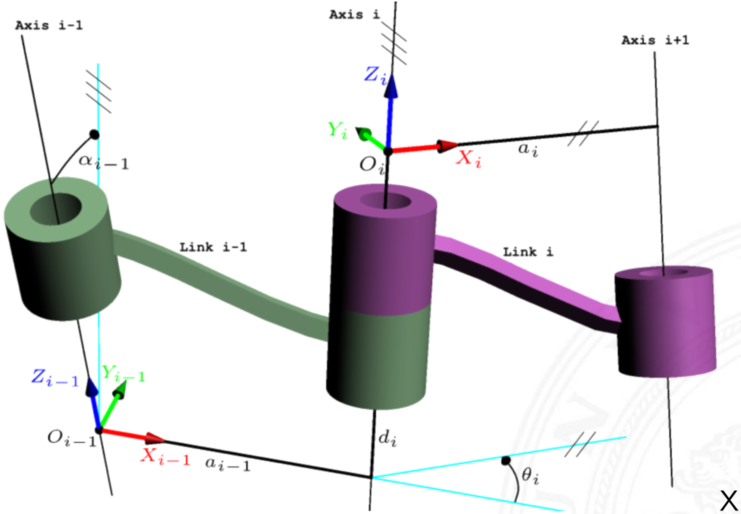
Using the product of four homogeneous transformations, which transform the coordinate frame $i-1$ into the coordinate frame i, the matrix A_i can be calculated as follows:

$$T_i = R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(d_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) \rightarrow CS_i$$

$$T_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dots & 0 \\ \dots & 0 \\ \dots & d_i \\ \dots & 1 \end{bmatrix} \begin{bmatrix} \dots & a_i \\ \dots & 0 \\ \dots & 0 \\ \dots & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Modified Parameters



Transformation order

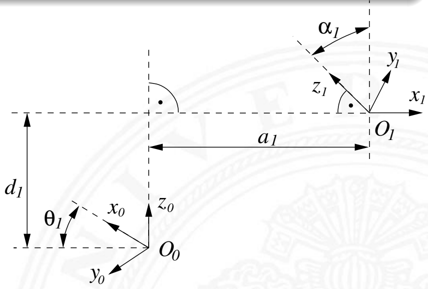
$$T_i = R_{x_{i-1}}(\alpha_{i-1}) \cdot T_{x_{i-1}}(a_{i-1}) \cdot R_{z_i}(\theta_i) \cdot T_{z_i}(d_i) \rightarrow CS_i$$

Definition of joint coordinate systems: Exceptions

Beware

The Denavit-Hartenberg convention is ambiguous!

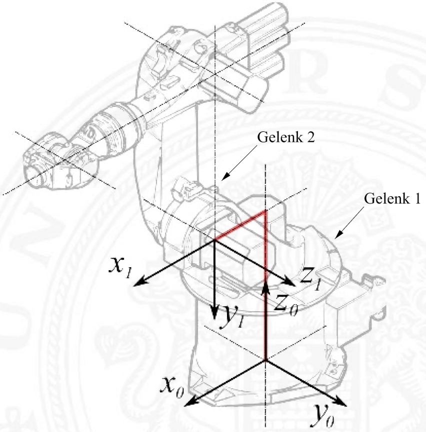
- z_{i-1} is parallel to z_i
  - arbitrary shortest normal
  - usually $d_i = 0$ is chosen
- z_{i-1} intersects z_i
  - usually $a_i = 0$ such that CS lies in the intersection point
- orientation of CS_n ambigous, as no joint $n+1$ exists
  - x_n must be a normal to z_{n-1}
  - usually z_n is chosen to point in the direction of the approach vector $\vec{a}$ of the tcp



Example DH-Parameter of a single joint

Determination of DH-Parameter (θ, d, a, α) for calculation of joint transformation:
$T_1 = R_z(\theta_1) T_z(d_1) T_x(a_1) R_x(\alpha_1)$
joint angle rotate by θ_1 around z_0, such that x_0 is parallel to x_1

$$R_z(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

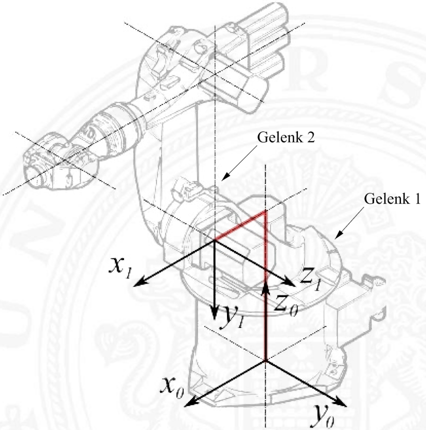


for the shown joint configuration $\theta_1 = 0°$

Example DH-Parameter of a single joint (cont.)

link offset translate by d_1 along z_0 until the intersection of z_0 and x_1

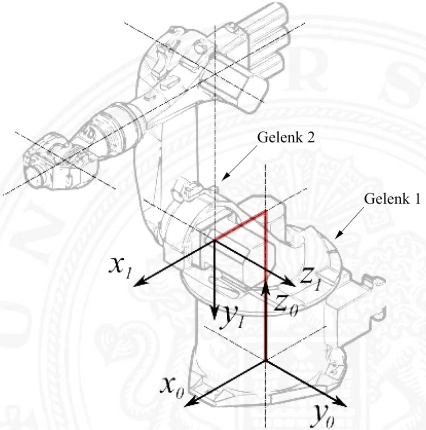
$$T_z(d_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Example DH-Parameter of a single joint (cont.)

link length translate by a_1 along x_1 such that the origins of both CS are congruent

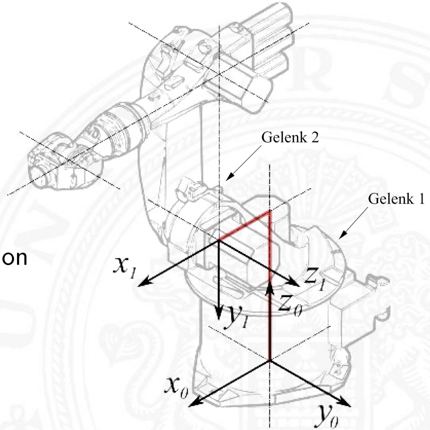
$$T_x(a_1) = \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Example DH-Parameter of a single joint (cont.)

link twist rotate z_0 by α_1 around x_1, such that z_0 lines up with z_1

$$R_{x(\alpha_1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_1) & -\sin(\alpha_1) & 0 \\ 0 & \sin(\alpha_1) & \cos(\alpha_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



for the shown joint configuration, $\alpha_1 = -90^\circ$ due to construction

- total transformation of CS_0 to CS_1 (general case)

$$
\begin{aligned}
{}^0T_1 &= R_z(\theta_1)\cdot T_z(d_1)\cdot T_x(a_1)\cdot R_x(\alpha_1)\\
&= \begin{bmatrix} cos\theta_1 & -sin\theta_1 cos\alpha_1 & sin\theta_1 sin\alpha_1 & a_1 cos\theta_1 \\ sin\theta_1 & cos\theta_1 cos\alpha_1 & -cos\theta_1 sin\alpha_1 & a_1 sin\theta_1 \\ 0 & sin\alpha_1 & cos\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

- rotary case: variable θ_1 and fixed d_1, a_1 *und* $(\alpha_1 = -90^\circ)$

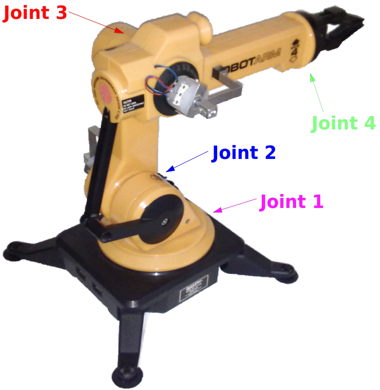
$$
\begin{aligned}
{}^0T_1 &= R_z(\theta_1)\cdot T_z(d_1)\cdot T_x(a_1)\cdot R_x(-90^\circ)\\
&= \begin{bmatrix} cos\theta_1 & 0 & -sin\theta_1 & a_1 cos\theta_1 \\ sin\theta_1 & 0 & cos\theta_1 & a_1 sin\theta_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

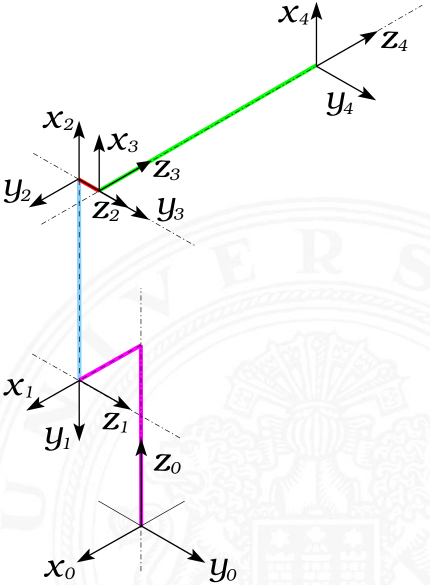
Procedure for predefined structure

- Fixed origin: CS_0 is the fixed frame at the base of the manipulator
- Determination of axes and consecutive numbering from 1 to n
- Positioning O_i on rotation- or shear-axis i,
  z_i points aways from z_{i-1}
- Determination of normal between the axes; setting x_i (in direction to the normal)
- Determination of y_i (right-hand system)
- Read off Denavit-Hartenberg parameters
- Calculation of overall transformation

Example DH-Parameter for Quickshot

- Definition of CS corresponding to DH convention
- Determination of DH-Parameter





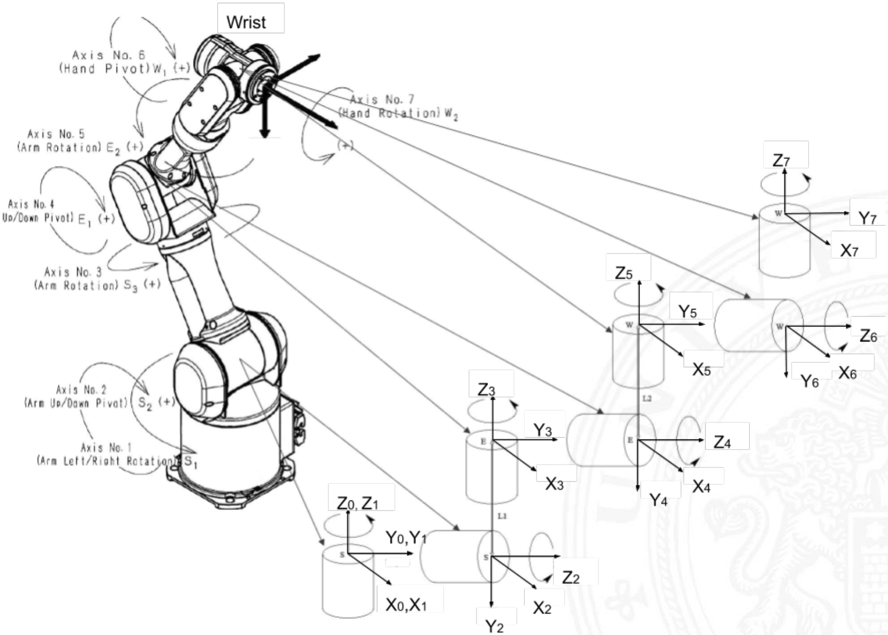
Example Transformation matrix T_6

$$
\begin{aligned}
T_6 &= T_1 \cdot T_2 \cdot T_3 \cdot T_4 \\
&= \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 20\cos\theta_1 \\ \sin\theta_1 & 0 & \cos\theta_1 & 20\sin\theta_1 \\ 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 160\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 160\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \\
&\qquad \begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ 0 & 1 & 0 & 28 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 250 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_1\cos\theta_4(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) - \sin\theta_1\sin\theta_4 & \dots & \dots & \dots \\ \sin\theta_1\cos\theta_4(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) + cos\theta_1\sin\theta_4 & \dots & \dots & \dots \\ -\cos\theta_4(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Sum-of-Angle formula

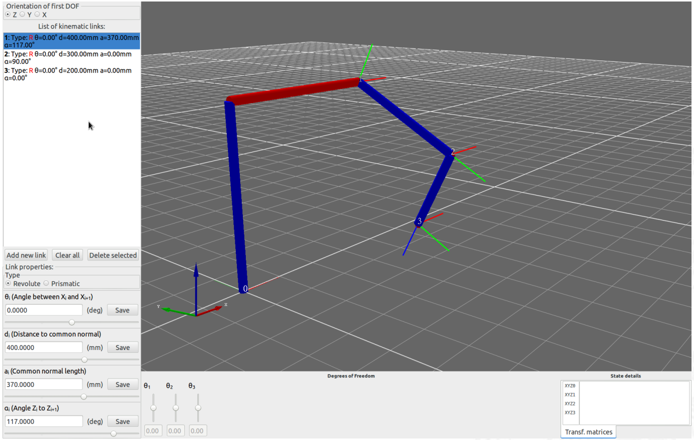
$$C_{23} = C_2 C_3 - S_2 S_3,$$
$$S_{23} = C_2 S_3 + S_2 C_3$$

Mitsubishi PA10-7C



Robotic arm kinematic GUI from MRPT

Download link



[18]

[18] Mobile Robot Programming Toolkit, https://www.mrpt.org/MRPT_in_GNU/Linux_repositories

Programming implementation

Write your own FK function!

- Robotics toolbox in Matlab
  - the implementation of book "Robotics, Vision & Control" by Peter Corke
- PythonRobotics
  - Python code collection of robotics algorithms, especially for autonomous navigation
- Robotics library
  - C++ framework for robot kinematics, dynamics, motion planning, control
- pybotics
  - provides a simple and clear interface to simulate and evaluate common robot concepts

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

Introduction to Robotics

Lecture 3

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

- Recapitulation of DH-Parameter
- URDF

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

Task/Manipulation Planning

Outline (cont.)

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

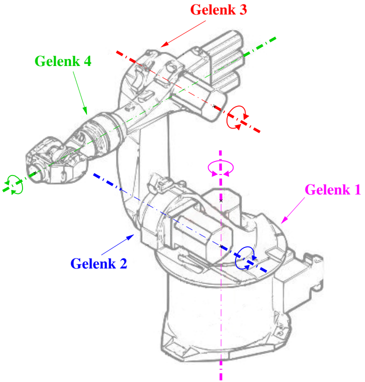
Recapitulation of DH-Parameter

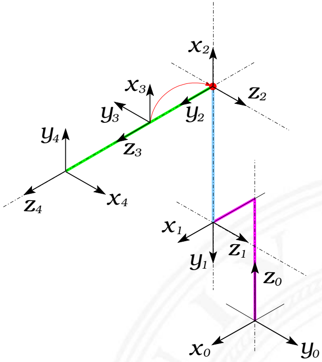
- universal minimal robot description
- based on frame transformations
- four parameters per frame transformation
- serial chain of transformations
- unique description of T_6

Drawbacks

- ambiguous convention
- only kinematic chain described
- missing information on geometry, physical constraints, dynamics, collisions, inertia, sensors, ...

Definition of joint coordinate systems





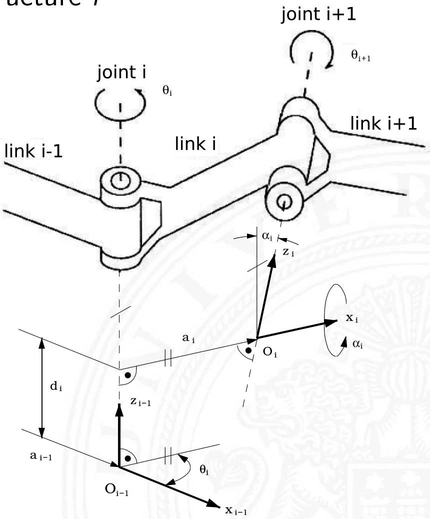
- CS_0 is the stationary origin at the base of the manipulator
- axis z_{i-1} is set along the axis of motion of the i^th joint
- axis x_i is the common normal of $z_{i-1} \times z_i$
- axis y_i concludes a right-handed coordinate system

Parameters for description of two arbitrary links

Two parameters for the description of the link structure i

- a_i: shortest distance between the z_{i-1}-axis and the z_i-axis
- α_i: rotation angle around the x_i-axis, which aligns the z_{i-1}-axis to the z_i-axis

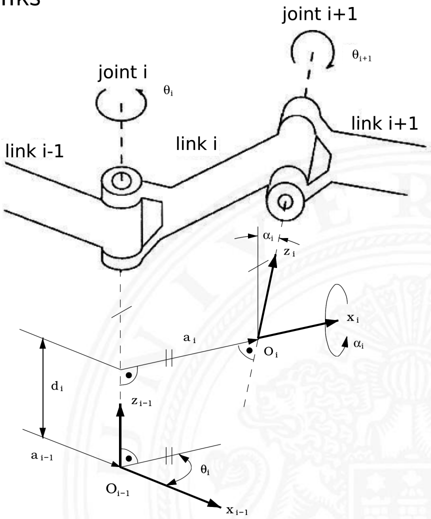
a_i and α_i are constant values due to construction



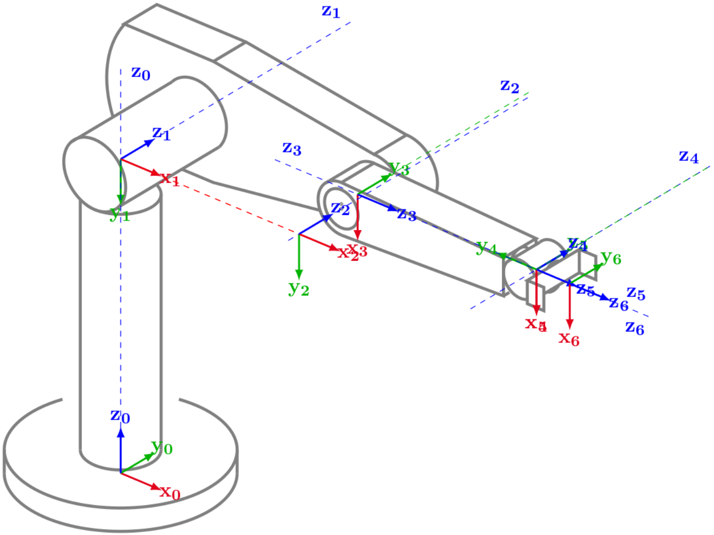
Two for relative distance and angle of adjacent links

- d_i : distance origin O_{i-1} of the $(i\text{-}1)^{st}$ CS to intersection of z_{i-1}-axis with x_i-axis
- θ_i : joint angle around z_{i-1}-axis to align x_{i-1}- parallel to x_i-axis into x_{i-1}, y_{i-1}-plane

θ_i and d_i are variable
- rotational: θ_i variable, d_i fixed
- translational: d_i variable, θ_i fixed



Example featuring PUMA 560



DH parameters of PUMA 560

Joint	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	d_1	θ_1^*
2	a_2	0	0	θ_2^*
3	a_3	$\frac{\pi}{2}$	d_3	θ_3^*
4	0	$-\frac{\pi}{2}$	d_4	θ_4^*
5	0	$\frac{\pi}{2}$	0	θ_5^*
6	0	0	d_6	θ_6^*

In order to transfer the manipulator-endpoint into the base coordinate system, T_6 is calculated as follows:

Universal Robot Description Format

Documentation

http://wiki.ros.org/urdf
http://wiki.ros.org/urdf/XML
http://wiki.ros.org/urdf/Tutorials

- robot description format used in ROS[19]
- hierarchical description of components
- XML format representing robot model
  - kinematics and dynamics
  - visual
  - collision
  - configuration

[19] http://ros.org

URDF: XML Tree Structure

- 1^{st}-level structure

```
<robot name="samplerobot">
</robot>
```

- 2^{nd}-level structure

  `link`, `joints`, `sensors`, `transmissions`, `gazebo`, `model_state`

- 3^{rd}-level structure

  `visual`, `inertia`, `collision`, `origin`, `parent`, ...

- 4^{th}-level structure

  ⋮

URDF: XML Tree Structure (cont.)

- Filename: robotname.urdf
- XML prolog:

```
<?xml version="1.0" encoding="utf-8"?>
```

- XML element types

```
<tag attribute="value"/>
```

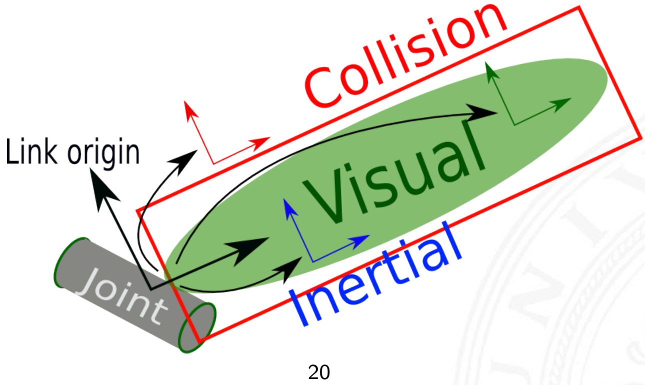
```
<tag attribute="value">
text or element(s)
</tag>
```

- XML comments

```
<!-- Comments are placed within these tags -->
```

URDF: Link

Link describes geometrical properties of a rigid body.



URDF: Link (cont.)


```
<link name="sample_link">
<!-- describes the mass and inertial properties of
the link -->
<inertial/>

<!-- describes the visual appearance of the link.
can be described using geometric primitives or
meshes -->
<visual/>

<!-- describes the collision space of the link.
is described like the visual appearance -->
<collision/>
</link>
```

[20] http://wiki.ros.org/urdf/XML/link

Geometric primitives for describing visual appearance of the link

```
<link name="base_link">
  <visual>
    <origin xyz="0 0 0.01" rpy="0 0 0"/>
    <geometry>
      <box size="0.2 0.2 0.02"/>
    </geometry>
    <material name="cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>
</link>
```

- Geometric primitives: `<box>`, `<cylinder>`, `<sphere>`
- Materials: `<color>`, `<texture>`

URDF: Link – visual – meshes

3D meshes for describing visual appearance of the link

```
<link name="base_link">
  <visual>
    <origin xyz="0 0 0.01" rpy="0 0 0"/>
    <geometry>
      <mesh filename="meshes/base_link.dae"
    </geometry>
  </visual>
  <collision>
     <origin xyz="0 0 0.01" rpy="0 0 0"/>
     <geometry>
          <cylinder radius="1" length="0.5"/>
     </geometry>
  </collision>
</link>
```

- the <collision> element can be simpler from the <visual> in order to reduce computation time

URDF: Link – inertial

Parameters describing the physical properties of the link

```
<link name="base_link">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <mass value="1">
    <inertia ixx="100" ixy="0" ixz="0"
             iyy="100" iyz="0" izz="100" />
  </inertial>
</link>
```

- center of gravity `<origin xyz>`
- object mass `<mass value>`
- inertia tensor `<intertia>`

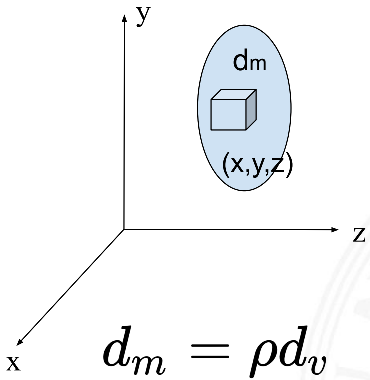
URDF: Inertia

Inertia tensor describes the distribution of mass of the link

- orientation and position of the inertia CS described by <origin> tag
- $$ {}^{A}I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & Izz \end{bmatrix} $$
    - diagonal values describe main inertial axes `ixx`, `iyy`, `izz`
    - `ixy`, `ixz`, `iyz` are 0 for symmetric primitives
    - rotations around largest and smallest inertial axis are most stable

URDF: Inertia (cont.)

- moments of inertia:
  $I_{xx} = \int (y^2 + z^2) dm \; I_{yy} = \int (x^2 + z^2) dm \; I_{zz} = \int (x^2 + y^2) dm$

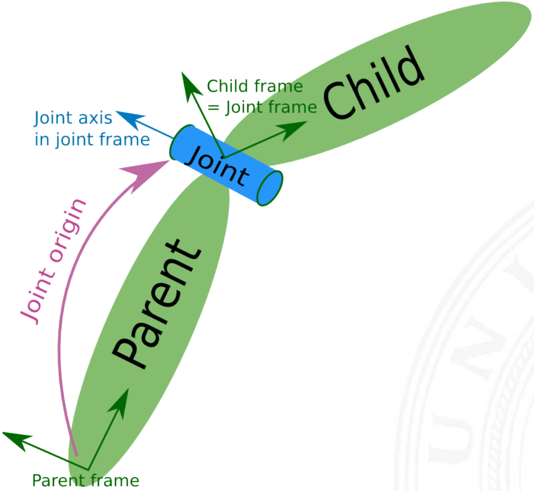


$$d_m = \rho d_v$$

- Products of Inertia:
  $I_{xy} = I_{yx} = \int xydm \; I_{zy} = I_{yz} = \int yzdm \; I_{xz} = I_{zx} = \int zydm$

URDF: Joint

Joint describes geometrical connections of two links.



21

URDF: Joint (cont.)

```
<joint name="base_link_to_cyl" type="revolute">
  <!-- describes joint position and orientation -->
  <origin xyz="0 0 0.07" rpy="0 0 0"/>

  <!-- describes the related links -->
  <parent link="base_link"/>
  <child link="base_cyl"/>

  <!-- describes the axis of rotation-->
  <axis xyz="0 0 1"/>

  <!-- describes the joint limits-->
  <limit velocity="1.5707963267"
         lower="-3.1415926535" upper="3.1415926535"/>
</joint>
```

[21] http://wiki.ros.org/urdf/XML/joint

URDF: Joint (cont.)

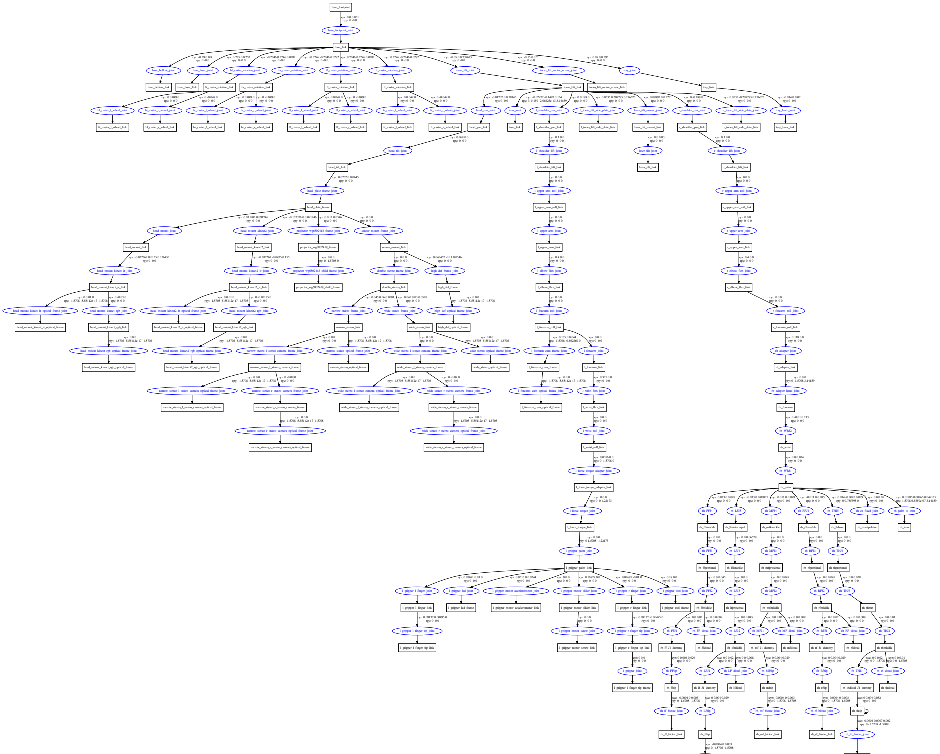
`type`	`revolute`, `continuous`, `prismatic`, `fixed`, `floating`, `planar`
`parent_link`	link which the joint is connected to
`child_link`	link which is connected to the joint
`axis`	joint axis relative to the joint CS. Represented using a normalized vector
`limit`	joint limits for motion (`lower`, `upper`), `velocity` and `effort`
`dynamics`	damping, friction
`calibration`	rising, falling
`mimic`	joint, multiplier, offset
`safety_controller`	soft_lower_limit, soft_upper_limit, k_position, k_velocity

URDF: Other elements

- sensor
  - position and orientation relative to link
  - sensor properties
    - update rate
    - resolution
    - minimum / maximum angle
- transmissions
  - relation of motor to joint motion
- gazebo
  - simulation properties
- model state
  - description of different robot configurations

Complex Hierachy

Full URDF hierarchy of the TAMS PR2 with the Shadow Hand.



base_footprint
base_footprint_joint
base_link
base_bellow_joint
base_bellow_link
base_laser_joint
base_laser_link
bl_caster_rotation_joint
bl_caster_rotation_link
br_caster_rotation_joint
br_caster_rotation_link
fl_caster_rotation_joint
fl_caster_rotation_link
fr_caster_rotation_joint
fr_caster_rotation_link
torso_lift_joint
torso_lift_link
torso_lift_motor_screw_joint
torso_lift_motor_screw_link
tray_joint
tray_link
bl_caster_l_wheel_joint
bl_caster_l_wheel_link
bl_caster_r_wheel_joint
bl_caster_r_wheel_link
br_caster_l_wheel_joint
br_caster_l_wheel_link
br_caster_r_wheel_joint
br_caster_r_wheel_link
fl_caster_l_wheel_joint
fl_caster_l_wheel_link
fl_caster_r_wheel_joint
fl_caster_r_wheel_link
fr_caster_l_wheel_joint
fr_caster_l_wheel_link
fr_caster_r_wheel_joint
fr_caster_r_wheel_link
head_pan_joint
head_pan_link
imu_joint
imu_link
l_shoulder_pan_joint
l_shoulder_pan_link
l_torso_lift_side_plate_joint
l_torso_lift_side_plate_link
laser_tilt_mount_joint
laser_tilt_mount_link
r_shoulder_pan_joint
r_shoulder_pan_link
r_torso_lift_side_plate_joint
r_torso_lift_side_plate_link
tray_laser_joint
tray_laser_link
head_tilt_joint
head_tilt_link
laser_tilt_joint
laser_tilt_link
l_shoulder_lift_joint
l_shoulder_lift_link
r_shoulder_lift_joint
r_shoulder_lift_link
head_plate_frame_joint
head_plate_frame
l_upper_arm_roll_joint
l_upper_arm_roll_link
r_upper_arm_roll_joint
r_upper_arm_roll_link
head_mount_joint
head_mount_link
head_mount_kinect2_joint
head_mount_kinect2_link
projector_wg6802418_frame_joint
projector_wg6802418_frame
sensor_mount_frame_joint
sensor_mount_link
l_upper_arm_joint
l_upper_arm_link
r_upper_arm_joint
r_upper_arm_link
head_mount_kinect_ir_joint
head_mount_kinect_ir_link
head_mount_kinect2_ir_joint
head_mount_kinect2_ir_link
projector_wg6802418_child_frame_joint
projector_wg6802418_child_frame
double_stereo_frame_joint
double_stereo_link
high_def_frame_joint
high_def_frame
l_elbow_flex_joint
l_elbow_flex_link
r_elbow_flex_joint
r_elbow_flex_link
head_mount_kinect_ir_optical_frame_joint
head_mount_kinect_ir_optical_frame
head_mount_kinect_rgb_joint
head_mount_kinect_rgb_link
head_mount_kinect2_ir_optical_frame_joint
head_mount_kinect2_ir_optical_frame
head_mount_kinect2_rgb_joint
head_mount_kinect2_rgb_link
narrow_stereo_frame_joint
narrow_stereo_link
wide_stereo_frame_joint
wide_stereo_link
high_def_optical_frame_joint
high_def_optical_frame
l_forearm_roll_joint
l_forearm_roll_link
r_forearm_roll_joint
r_forearm_roll_link
head_mount_kinect_rgb_optical_frame_joint
head_mount_kinect_rgb_optical_frame
head_mount_kinect2_rgb_optical_frame_joint
head_mount_kinect2_rgb_optical_frame
narrow_stereo_l_stereo_camera_frame_joint
narrow_stereo_l_stereo_camera_frame
narrow_stereo_optical_frame_joint
narrow_stereo_optical_frame
wide_stereo_l_stereo_camera_frame_joint
wide_stereo_l_stereo_camera_frame
wide_stereo_optical_frame_joint
wide_stereo_optical_frame
l_forearm_cam_frame_joint
l_forearm_cam_frame
l_forearm_joint
l_forearm_link
rh_adapter_joint
rh_adapter_link
narrow_stereo_l_stereo_camera_optical_frame_joint
narrow_stereo_l_stereo_camera_optical_frame
narrow_stereo_r_stereo_camera_frame_joint
narrow_stereo_r_stereo_camera_frame
wide_stereo_l_stereo_camera_optical_frame_joint
wide_stereo_l_stereo_camera_optical_frame
wide_stereo_r_stereo_camera_frame_joint
wide_stereo_r_stereo_camera_frame
l_forearm_cam_optical_frame_joint
l_forearm_cam_optical_frame
l_wrist_flex_joint
l_wrist_flex_link
rh_adapter_hand_joint
rh_forearm
narrow_stereo_r_stereo_camera_optical_frame_joint
narrow_stereo_r_stereo_camera_optical_frame
wide_stereo_r_stereo_camera_optical_frame_joint
wide_stereo_r_stereo_camera_optical_frame
l_wrist_roll_joint
l_wrist_roll_link
rh_WRJ2
rh_wrist
l_force_torque_adapter_joint
l_force_torque_adapter_link
rh_WRJ1
rh_palm
l_force_torque_joint
l_force_torque_link
rh_FFJ4
rh_ffknuckle
rh_LFJ5
rh_lfmetacarpal
rh_MFJ4
rh_mfknuckle
rh_RFJ4
rh_rfknuckle
rh_THJ5
rh_thbase
rh_ee_fixed_joint
rh_manipulator
rh_palm_to_imu
rh_imu
l_gripper_palm_joint
l_gripper_palm_link
rh_FFJ3
rh_ffproximal
rh_LFJ4
rh_lfknuckle
rh_MFJ3
rh_mfproximal
rh_RFJ3
rh_rfproximal
rh_THJ4
rh_thproximal
l_gripper_l_finger_joint
l_gripper_l_finger_link
l_gripper_led_joint
l_gripper_led_frame
l_gripper_motor_accelerometer_joint
l_gripper_motor_accelerometer_link
l_gripper_motor_slider_joint
l_gripper_motor_slider_link
l_gripper_r_finger_joint
l_gripper_r_finger_link
l_gripper_tool_joint
l_gripper_tool_frame
rh_FFJ2
rh_ffmiddle
rh_LFJ3
rh_lfproximal
rh_MFJ2
rh_mfmiddle
rh_RFJ2
rh_rfmiddle
rh_THJ3
rh_thhub
l_gripper_l_finger_tip_joint
l_gripper_l_finger_tip_link
l_gripper_motor_screw_joint
l_gripper_motor_screw_link
l_gripper_r_finger_tip_joint
l_gripper_r_finger_tip_link
rh_FFJ1
rh_ff_J1_dummy
rh_FF_distal_joint
rh_ffdistal
rh_LFJ2
rh_lfmiddle
rh_MFJ1
rh_mf_J1_dummy
rh_MF_distal_joint
rh_mfdistal
rh_RFJ1
rh_rf_J1_dummy
rh_RF_distal_joint
rh_rfdistal
rh_THJ2
rh_thmiddle
l_gripper_joint
l_gripper_l_finger_tip_frame
rh_FFtip
rh_fftip
rh_LFJ1
rh_lf_J1_dummy
rh_LF_distal_joint
rh_lfdistal
rh_MFtip
rh_mftip
rh_RFtip
rh_rftip
rh_THJ1
rh_thdistal_J1_dummy
rh_th_distal_joint
rh_thdistal
rh_ff_biotac_joint
rh_ff_biotac_link
rh_LFtip
rh_lftip
rh_mf_biotac_joint
rh_mf_biotac_link
rh_rf_biotac_joint
rh_rf_biotac_link
rh_thtip
rh_th_biotac_joint

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators
- Workspace
- Algebraic solvability of manipulator
- Geometrical solvability of manipulator
- Popular inverse kinematics solutions

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Outline (cont.)

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

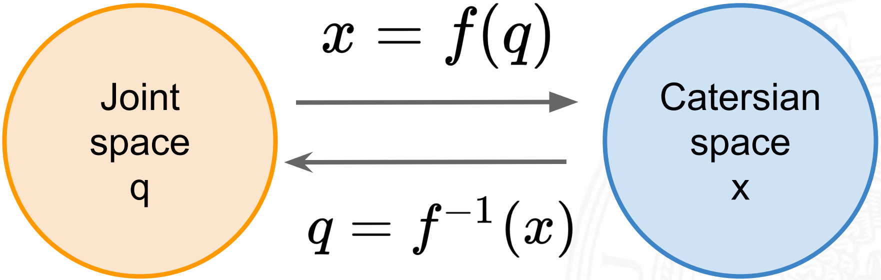
Inverse kinematics for manipulators

- **Forward Kinematics**: given robot configurations(joint angles), find position & orientations of the end-effector

Set of problems

- In the majority of cases the control of robot manipulators takes place in the *joint space*,
- The informations about objects are mostly given in the *cartesian space*.

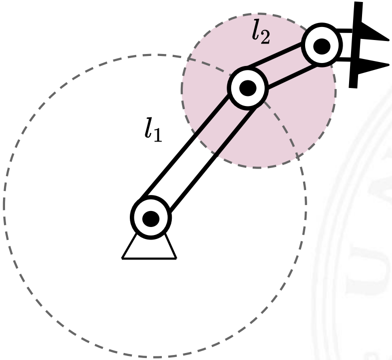
- **Inverse Kinematics**: give position & orientations of the end-effector, find robot configurations(joint angles)



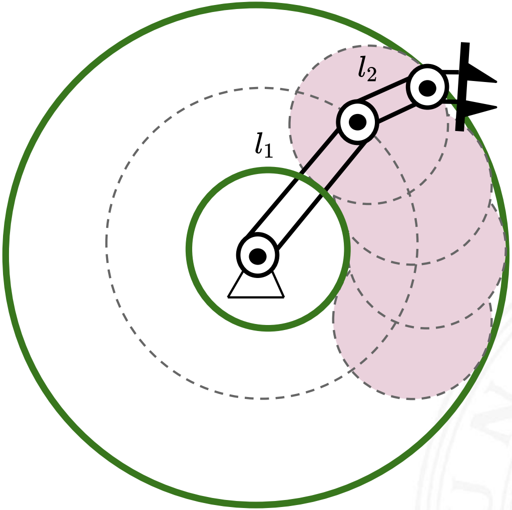
Existence of solutions: Workspace

`Workspace`: the volume of space that is reachable for the tool of the manipulator.

- `reachable` workspace
- `dexterous` workspace

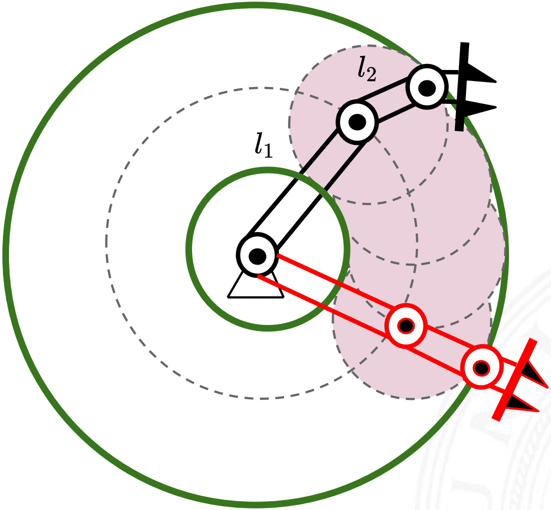


Existence of solutions: Workspace (cont.)



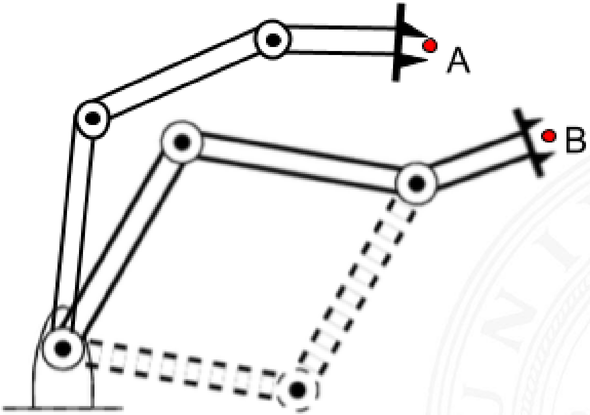
if $l_1 \neq l_2$, the reachable workspace becomes a ring of outer radius $|l_1 + l_2|$,and inner radius $|l_1 - l_2|$.

Existence of solutions: Workspace (cont.)



Does the workspace change if joint limits are considered?
For example, $q_1 \in [0, \pi], q_2 \in [0, \pi]$.

Multiple solutions



The solution using the example of PUMA 560

$$T_6 = T'T'' = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$n_x = C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1(S_4C_5C_6 + C_4S_6) \quad (2)$$
$$n_y = S_1[C_{23}(C_4C_5C_6 - S_4S_6 - S_{23}S_5S_6] + C_1(S_4C_5C_6 + C_4S_6) \quad (3)$$
$$n_z = -S_{23}[C_4C_5C_6 - S_4S_6] - C_{23}S_5C_6 \quad (4)$$

The solution using the example of PUMA 560 (cont.)

$$o_x = \ldots \tag{5}$$
$$o_y = \ldots \tag{6}$$
$$o_z = \ldots \tag{7}$$
$$a_x = \ldots \tag{8}$$
$$a_y = \ldots \tag{9}$$
$$a_z = \ldots \tag{10}$$
$$p_x = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2) \tag{11}$$
$$p_y = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + s_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2) \tag{12}$$
$$p_z = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2 \tag{13}$$

Remark

- Non-linear equations
- Existence of solutions
- Multiple solutions
- `Different solution strategy:` closed solutions vs. numerical solutions

Different methods for solution finding

Closed form (analytical):
An expression is said to be a closed-form experession if it can be expressed analytically in terms of a bounded number of certain 'well-known' functions.

- $+ - \times \div$
- nth roots
- exponent and logarithm
- trigonometric and inverse trigonometric functions
- Do not include infinite series, continued fractions, integrals or limits.

Different methods for solution finding (cont.)

Closed form (analytical):

- `algebraic solution`
  - + accurate solution by means of equations
  - – solution is not geometrically representative
- `geometrical solution`
  - + case-by-case analysis of possible robot configurations
  - – robot specific

Numerical form:

- `iterative methods`
  - + the methods are transferable
  - – computationally intensive, for several exceptions the convergence can not be guaranteed

Algebraic solution

Algebraic Approach manipulates the given equations into a form whose solution is known.

- Method1: **Transcendental equations**

  $1.\ \sin\theta = a \Rightarrow \theta = A\tan 2(a, \pm\sqrt{1-a^2})$

  $2.\ \cos\theta = b \Rightarrow \theta = \pm A\tan 2(\sqrt{1-b^2}, b)$

  $3.\{\begin{array}{c} \sin\theta = a \\ \cos\theta = b \end{array} \Rightarrow \theta = A\tan 2(a, b)$

  $4. a\cos\theta + b\sin\theta = 0 \Rightarrow \theta = A\tan 2(a, -b)\ or\ \theta = A\tan 2(-a, b)$

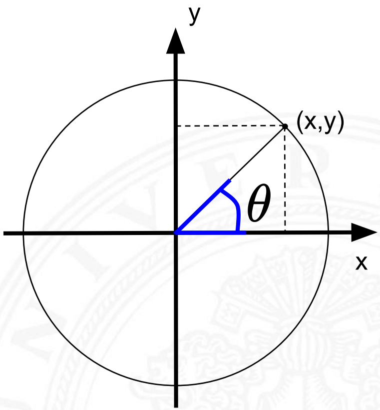
  $5. a\cos\theta + b\sin\theta = c \Rightarrow \theta = A\tan 2(b, a) \pm A\tan 2(\sqrt{a^2+b^2-c^2}, c)$

  $6.\{\begin{array}{c} a\cos\theta - b\sin\theta = c \\ a\sin\theta + b\cos\theta = d \end{array} \Rightarrow \theta = A\tan 2(ad - ba, ac - bd)$

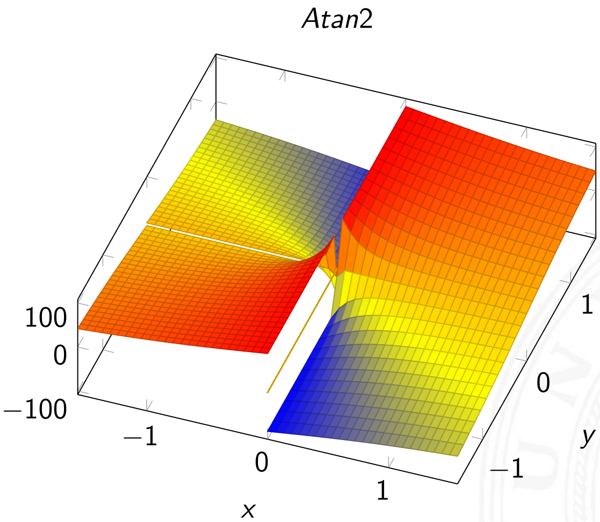
Algebraic solution (cont.)

We define the function *Atan*2 as:

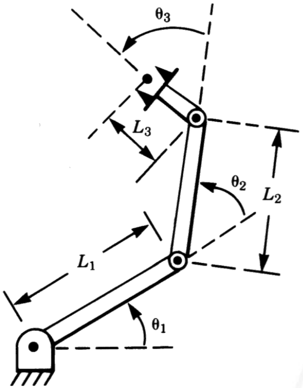
$$\theta = Atan2(y, x) = \begin{cases} Atan(\frac{y}{x}) & \text{for } +x \\ Atan(\frac{y}{x}) + \pi & \text{for } -x, +y_0 \\ Atan(\frac{y}{x}) - \pi & \text{for } -x, -y \\ \frac{\pi}{2} & \text{for } x = 0, +y \\ \frac{-\pi}{2} & \text{for } x = 0, -y \\ NaN & \text{for } x = 0, y = 0 \end{cases}$$



Algebraic solution (cont.)



Example: a planar 3 DOF manipulator



Joint	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	l_1	0	θ_2
3	0	l_2	0	θ_3

The algebraical solution for the 3 DOF planar

$$T_6 = {}^0T_3 = \begin{bmatrix} C_{123} & -S_{123} & 0 & l_1 C_1 + l_2 C_{12} \\ S_{123} & C_{123} & 0 & l_1 S_1 + l_2 S_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with $C_{ij[k]} = \cos(\theta_i + \theta_j[+\theta_k])$

Specification for the TCP: (x, y, ϕ). For such kind of vectors applies:

$${}^0T_3 = \begin{bmatrix} C_\phi & -S_\phi & 0 & x \\ S_\phi & C_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The algebraical solution for the 3 DOF planar (cont.)

Resultant, four equations can be derived:

$$C_\phi = C_{123} \tag{14}$$

$$S_\phi = S_{123} \tag{15}$$

$$x = l_1 C_1 + l_2 C_{12} \tag{16}$$

$$y = l_1 S_1 + l_2 S_{12} \tag{17}$$

The algebraical solution for the 3 DOF planar (cont.)

Square and add (20) ($x = l_1 C_1 + l_2 C_{12}$) and (21) ($y = l_1 S_1 + l_2 S_{12}$)

$$x^2 + y^2 = l_1^1 + l_2^2 + 2 l_1 l_2 C_2$$

using

$$C_{12} = C_1 C_2 - S_1 S_2, S_{12} = C_1 S_2 + S_1 C_2$$

giving

$$C_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

for goal in workspace

$$S_2 = \pm\sqrt{1 - C_2^2}$$

solution

$$\theta_2 = atan2(S_2, C_2)$$

The algebraical solution for the 3 DOF planar (cont.)

solve (20) $(x = l_1 C_1 + l_2 C_{12})$ and (21) $(y = l_1 S_1 + l_2 S_{12})$ for θ_1

$$\theta_1 = atan2(y, x) - atan2(k_2, k_1)$$

where $k_1 = l_1 + l_2 C_2$ and $k_2 = l_2 S_2$.

solve θ_3 from (19) $(C_\phi = C_{123})$ and (18) $(S_\phi = S_{123})$

$$\theta_1 + \theta_2 + \theta_3 = atan2(S_\phi, C_\phi) = \phi$$

Analytical solvability of manipulator

The closed solution exists if specific constraints (sufficient constraints) for the arm geometry are satisfied:

If 3 sequent axes intersect in a given point

or if 3 sequent axes are parallel to each other

- manipulators should be designed regarding these constraints
- most of them are
  - PUMA 560: axes 4, 5 & 6 intersect in a single point
  - Mitsubishi PA10, KUKA LWR, PR2
  - 3-DOF planar (RPC)

Algebraical solution (polynomial conversion)

Method2: **Reduction to polynomial**

The following substitutions are used for the polynomial conversion of transcendental equations:

$$u = tan\frac{\theta}{2}$$

$$\cos\theta = \frac{1-u^2}{1+u^2}$$

$$\sin\theta = \frac{2u}{1+u^2}$$

Algebraical solution (polynomial conversion) (cont.)

Example:
The following transcendental equation is given:

$$a\cos\theta + b\sin\theta = c$$

$\Rightarrow \theta = A\tan 2(b, a) \pm A\tan 2(\sqrt{a^2 + b^2 - c^2}, c)$
After the polynomial conversion:

$$a(1-u^2) + 2bu = c(1+u^2)$$

The solution for u:

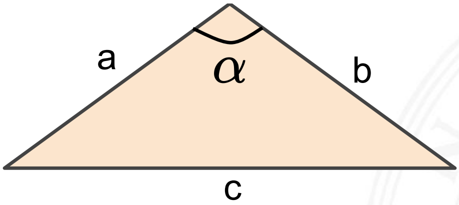
$$u = \frac{b \pm \sqrt{b^2 - a^2 - c^2}}{a + c}$$

Then:

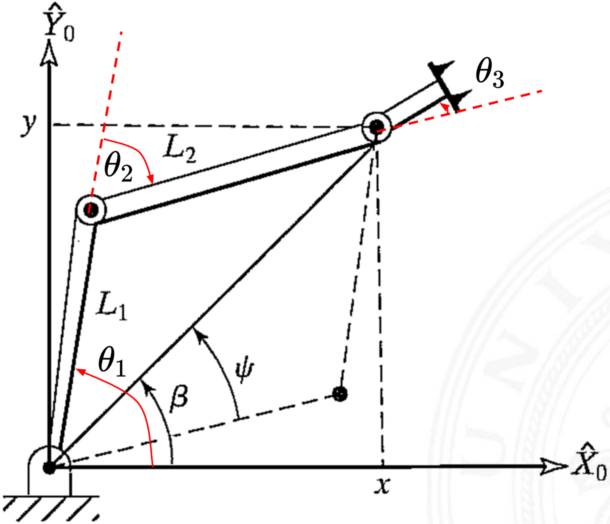
$$\theta = 2tan^{-1}(\frac{b \pm \sqrt{b^2 - a^2 - c^2}}{a + c})$$

Geometrical solution

- Decompose the spatial geometry of the arm into several plane geometry problems
- Law of cosines: $c^2 = a^2 + b^2 - 2ab\cos\alpha$



The geometrical solution for the example 1



The geometrical solution for the example 1 (cont.)

Calculate θ_2 via the `law of cosines`:

$$x^2 + y^2 = l_1^2 + l_2^2 - 2 l_1 l_2 \cos(180 + \theta_2)$$

The solution:

$$\theta_2 = \pm cos^{-1} \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$\theta_1 = \beta \pm \psi$$

where:

$$\beta = atan2_m(y, x), \quad \cos\psi = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 \sqrt{x^2 + y^2}}$$

For $\theta_1, \theta_2, \theta_3$ applies:

$$\theta_1 + \theta_2 + \theta_3 = \phi$$

Exercise

Assume we have derived the forward kinematics as:

$$
{}^{0}T_{3} = \begin{bmatrix} C_1 C_{23} & -C_1 S_{23} & S_1 & C_1(C_2 l_2 + l_1) \\ S_1 C_{23} & -S_1 S_{23} & -C_1 & S_1(C_2 l_2 + l_1) \\ S_{23} & C_{23} & 0 & S_2 l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

And we know:

$$
{}^{0}T_{3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Question: How to solve the inverse kinematics?

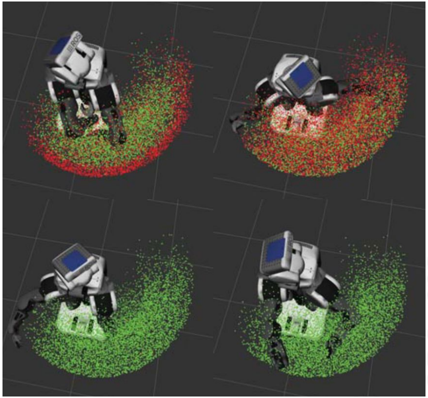
IKFast

- Closed-form solutions
- OpenRAVE
- faster (4 μs) but only work with any number of joints arranged in a chain
- Tutorial: ikfast MoveIt! kinematics_base plugin

TracIK

- TRACLabs' IK solver
- Tutorial: trac_ik MoveIt! kinematics_base plugin
- two IK implementations:
  KDL's Newton-based convergence algorithm
  SQP (Sequential Quadratic Programming) nonlinear optimization approach
- trac_ik_python (RPC)

BioIK 1



KDL

TrackIK

BioIK 2

Download link: bio_ik MoveIt! kinematics_base plugin
22

[22]Ruppel, P., Hendrich, N., Starke, S. and Zhang, J., 2018, May. Cost functions to specify full-body motion and multi-goal manipulation tasks. In 2018 ICRA (pp. 3152-3159). IEEE.

Introduction to Robotics

Lecture 4

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

IK Review

- Workspace
  - reachable workspace
  - dexterous workspace
- closed solutions:
  - algebraic solution
  - geometrical solution

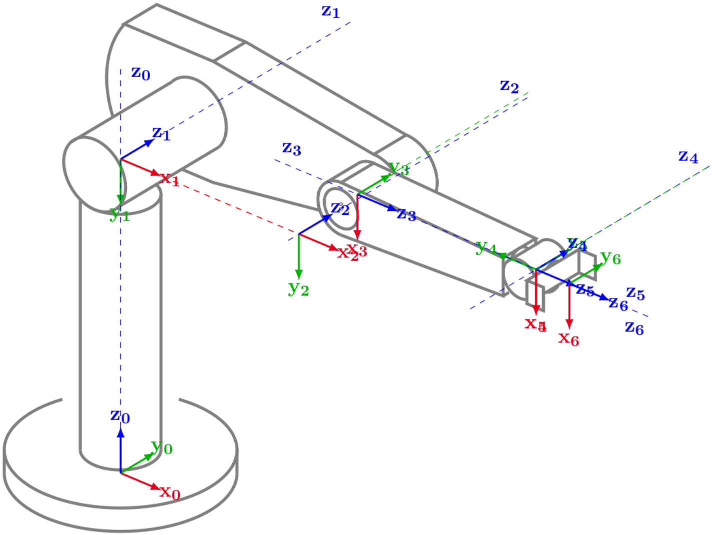
The closed solution exists if specific constraints (sufficient constraints) for the arm geometry are satisfied:

If 3 sequent axes intersect in a given point

or if 3 sequent axes are parallel to each other

- numerical solutions

Example featuring PUMA 560



Assume we have derived the forward kinematics as:

$$
{}^{0}T_{3} = \begin{bmatrix} C_1 C_{23} & -C_1 S_{23} & S_1 & C_1(C_2 l_2 + l_1) \\ S_1 C_{23} & -S_1 S_{23} & -C_1 & S_1(C_2 l_2 + l_1) \\ S_{23} & C_{23} & 0 & S_2 l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

And we know:

$$
{}^{0}T_{3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Question: How to solve the inverse kinematics?

$$
{}^0T_3 = \begin{bmatrix} C_1 C_{23} & -C_1 S_{23} & S_1 & C_1(C_2 l_2 + l_1) \\ S_1 C_{23} & -S_1 S_{23} & -C_1 & S_1(C_2 l_2 + l_1) \\ S_{23} & C_{23} & 0 & S_2 l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
S_1 = r_{13} \tag{18}
$$

$$
C_1 = -r_{23} \tag{19}
$$

Using the two-argument arctangent to solve for θ_1,

$$
\theta_1 =
$$

Exercise

$$C_1(C_2 l_2 + l_1) = p_x \tag{20}$$
$$S_1(C_2 l_2 + l_1) = p_y \tag{21}$$
$$S_2 l_2 = p_z \tag{22}$$

solve θ_2 from (20 - 22),

$$S_{23} = r_{31} \tag{23}$$

$$C_{23} = r_{32} \tag{24}$$

solve θ_3 from (20 - 22),

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

- Velocity of rigid body
- Velocity Propagation between Links
- Jacobian of a Manipulator
- Singular Configurations

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Outline (cont.)

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Differential motion

- Forward kinematics: $\theta \longrightarrow x$
- Inverse kinematics: $x \longrightarrow \theta$
- instantaneous kinematics: $\theta + \delta\theta \longrightarrow x + \delta x$
- Relationship $\delta\theta \leftrightarrow \delta x$

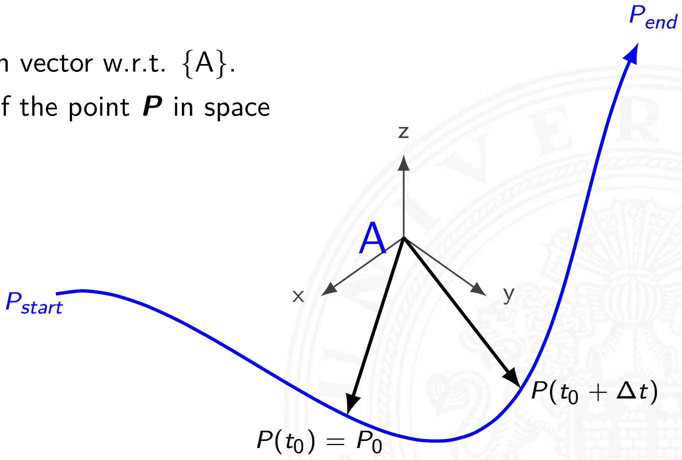
$$\dot{\theta} \leftrightarrow \dot{x}$$

Joint velocities $\leftrightarrow$ end-effector velocities

- Linear velocity
- Angular velocity

$$^{A}V_{P} = \frac{d}{dt}(^{A}P) = \lim_{\Delta t \to 0} \frac{\Delta \boldsymbol{P}(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{\boldsymbol{P}(t+\Delta t) - \boldsymbol{P}(t)}{\Delta t} \quad (25)$$

- $\boldsymbol{P}$ is a time-varying position vector w.r.t. $\{A\}$.
- $^{A}V_{P}$ is the linear velocity of the point $\boldsymbol{P}$ in space



Representing ${}^{A}V_{P}$ in another frame {B}, then we get

$$^{B}({}^{A}V_{P}) = ^{B}(\frac{d}{dt}({}^{A}P)) = \frac{d}{dt}({}^{B}R_{A}({}^{A}P)) = {}^{B}R_{A}\frac{d}{dt}({}^{A}P) = {}^{B}R_{A} \cdot {}^{A}V_{P}$$

Note, as ${}^{A}R_{B}$ remains invariant during the motion.

Notation

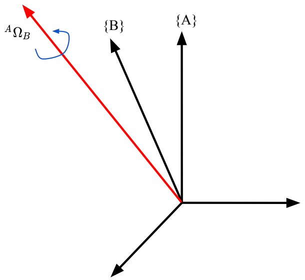
- if $\boldsymbol{P}$ is the origin of a frame {C}, which is moving, we typically use $v_c =^{U} V_C$ to denote the linear velocity of the origin of {c} w.r.t. the reference frame {U}
- ${}^{A}v_c$ means the linear velocity of the origin of {C} w.r.t. {U} expressed in {A}

Angular velocity

Angular velocity describes rotational motion of a frame.

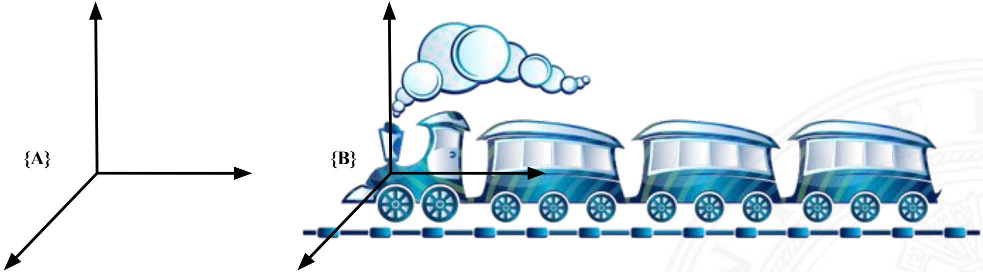
Notation

- ${}^{A}\Omega_{B}$ denotes the angular velocity of $\{B\}$ w.r.t. $\{A\}$
- $\omega_c = {}^{U}\Omega_C$ denotes the angular velocity of $\{c\}$ w.r.t. $\{U\}$



- the direction of ${}^{A}\Omega_{B}$ indicates the instantaneous axis of rotation
- the magnitude of ${}^{A}\Omega_{B}$ indicates the speed of rotation

Linear velocity of rigid body



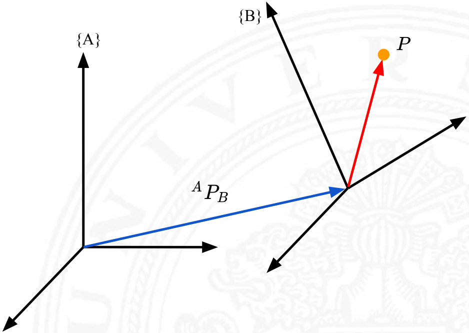
Linear velocity of rigid body

Assume that there is only a linear motion of {B} w.r.t. {A}

$$^AP = {}^AP_B + {}^AR_B \cdot {}^BP$$

Differentiating the above equation

$$\begin{aligned} ^AV_P &= {}^AV_B + \frac{d}{dt}({}^AR_B \cdot {}^BP) \\ &= {}^AV_B + {}^AR_B \frac{d}{dt}({}^BP) \\ &= {}^AV_B + {}^AR_B \cdot {}^BV_P \end{aligned}$$

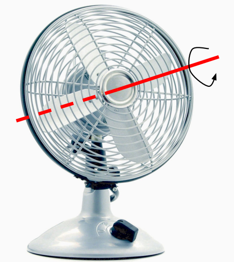


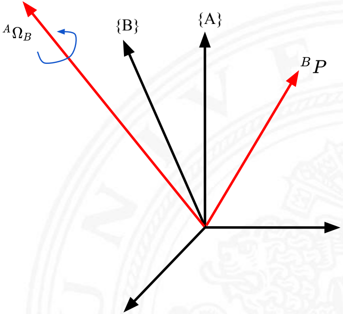
Note, as ^AR_B remains invariant during the motion.

Angular velocity of rigid body

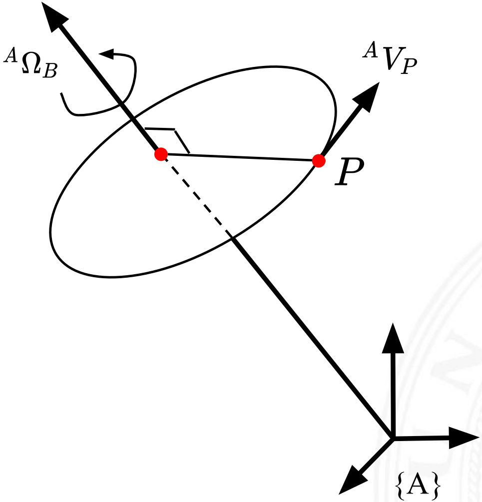
Assume that:

1. No linear velocity of {B} w.r.t. {A}
2. There is a rotational velocity of {B} w.r.t. {A}, ${}^{A}R_{B}$ is time-varying.
3. Point P is fixed in {B}

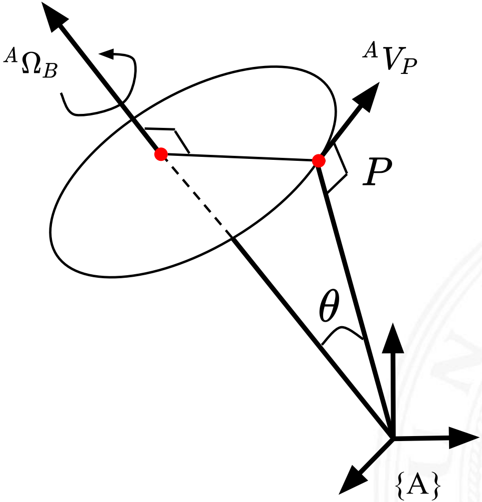




Angular velocity of rigid body (cont.)



Angular velocity of rigid body (cont.)



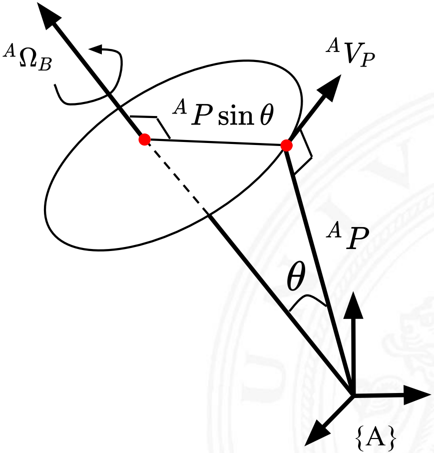
$^{A}V_P$ is proportional to:

- $\|^{A}\Omega_B\|$
- $\|^{A}P\sin\theta\|$

and

- $^{A}V_P \perp {}^{A}\Omega_B$
- $^{A}V_P \perp {}^{A}P$

$$^{A}V_P = {}^{A}\Omega_B \times {}^{A}P$$



Cross Product Operator

$$a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \longrightarrow c = a \times b \Longrightarrow c = \hat{a}b$$

$a\times \Longrightarrow \hat{a}$: a skew-symmetric matrix

vectors $\Longrightarrow$ matrices

$$c = \hat{a}b = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

Cross Product Operator

$$^{A}V_{P} = {}^{A}\Omega_{B} \times {}^{A}P = {}^{A}\hat{\Omega}_{B}{}^{A}P$$

$$^{A}\Omega_{B} = \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix}, {}^{A}P = \begin{bmatrix} {}^{A}P_x \\ {}^{A}P_y \\ {}^{A}P_z \end{bmatrix}$$

$$^{A}V_{P} = {}^{A}\hat{\Omega}_{B}{}^{A}P = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \begin{bmatrix} {}^{A}P_x \\ {}^{A}P_y \\ {}^{A}P_z \end{bmatrix}$$

Simultaneous linear and angular motion

Assume that:

1. No linear velocity of {B} w.r.t. {A}
2. There is a rotational velocity of {B} w.r.t. {A}, ${}^{B}R_{A}$ is time-varying.
3. ~~Point P is fixed in {B}~~

$$
{}^{A}V_{P} = {}^{A}\Omega_{B} \times {}^{A}P
$$

$$
\Downarrow {}^{B}V_{P}
$$

$$
\begin{aligned}
{}^{A}V_{P} &= {}^{A}R_{B}{}^{B}V_{P} + {}^{A}\Omega_{B} \times {}^{A}P \\
&= {}^{A}R_{B}{}^{B}V_{P} + {}^{A}\Omega_{B} \times {}^{A}R_{B}{}^{B}P
\end{aligned}
$$

Simultaneous linear and angular motion

Assume that:

1. ~~No linear velocity of {B} w.r.t. {A}~~
2. There is a rotational velocity of {B} w.r.t. {A}, ${}^{B}R_{A}$ is time-varying.
3. ~~Point Q is fixed in {B}~~

$$
{}^{A}V_{P} = {}^{A}R_{B}\,{}^{B}V_{P} + {}^{A}\Omega_{B} \times {}^{A}R_{B}\,{}^{B}P
$$

$$
\Downarrow {}^{A}V_{B}
$$

$$
{}^{A}V_{P} = {}^{A}V_{B} + {}^{A}R_{B}\,{}^{B}V_{P} + {}^{A}\Omega_{B} \times {}^{A}R_{B}\,{}^{B}P
$$

- Linear motion

$$^{A}V_{P} = {}^{A}V_{B} + {}^{A}R_{B}\,{}^{B}V_{P}$$

- Rotational motion

$$^{A}V_{P} = {}^{A}R_{B}\,{}^{B}V_{P} + {}^{A}\Omega_{B} \times {}^{A}R_{B}\,{}^{B}P$$

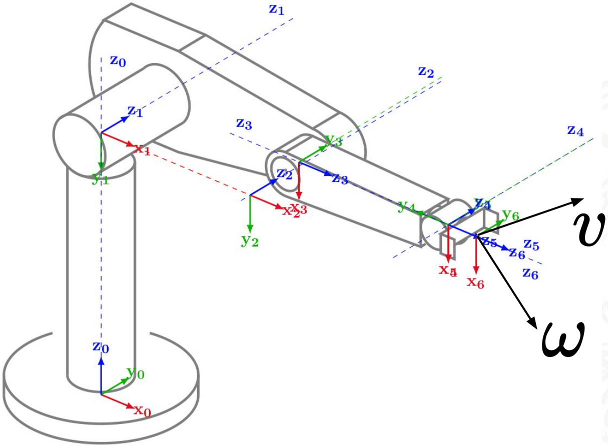
- General

$$^{A}V_{P} = {}^{A}V_{B} + {}^{A}R_{B}\,{}^{B}V_{P} + {}^{A}\Omega_{B} \times {}^{A}R_{B}\,{}^{B}P$$

Velocity propagation

Motion of the links of a manipulator.

- v: linear velocity
- ω : angular velocity

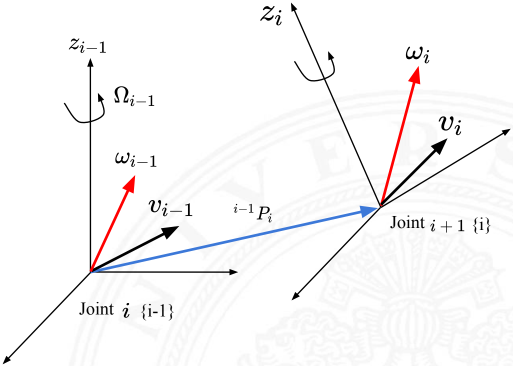


Angular velocity propagation

For a revolute joint i, the angular velocity ${}^{i-1}\omega_{i-1}$ of the link i is:

$\dot{\theta}_i {}^{i}Z_{i-1}$

- $\dot{\theta}_i$ is a scalar, the velocity of the joint i
- ${}^{i}Z_{i-1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- scalar multiplication



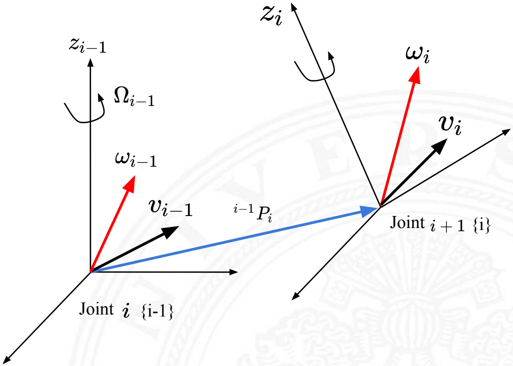
Angular velocity propagation

Angular velocity ${}^{i-1}\omega_i$ of the link $i+1$ is influenced by:

- the angular velocity ${}^{i-1}\omega_{i-1}$ of the link i
- if joint $i+1$ is a revolute joint, the joint velocity along the z-axis Z_i of the link

$${}^{i-1}\omega_i = {}^{i-1}\omega_{i-1} + {}^{i-1}R_i \dot{\theta}_{i+1} {}^{i}Z_i$$

$${}^{i}\omega_i = {}^{i}R_{i-1} {}^{i-1}\omega_{i-1} + \dot{\theta}_{i+1} {}^{i}Z_i$$

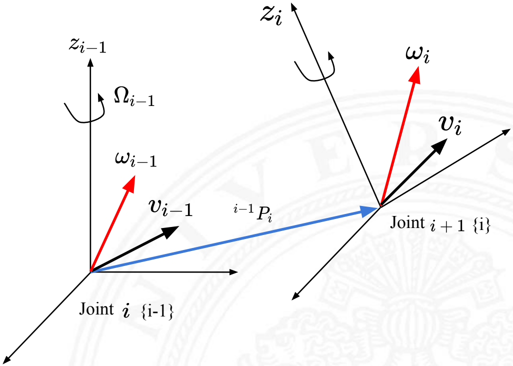


Linear velocity propagation

For a prismatic joint i, the linear velocity ${}^{i-1}v_{i-1}$ of the link i is:

$\dot{d}_i {}^{i}Z_{i-1}$

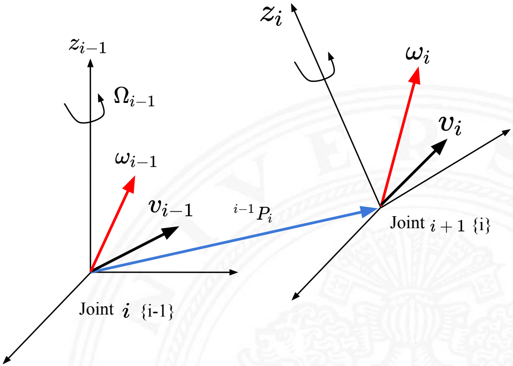
- $\dot{d}_i$ is a scalar, the velocity of the link i
- ${}^{i}Z_{i-1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
- scalar multiplication



Linear velocity propagation

Linear velocity ${}^{i-1}v_i$ of the link $i+1$ is influenced by:

- the linear velocity ${}^{i-1}v_{i-1}$ of the joint i
- if joint i is a revolute joint, the linear velocity of the origin of frame $\{i+1\}$
- if joint $i+1$ is a prismatic joint, the joint velocity along the z-axis Z_i of the joint



$$ {}^{i-1}v_i = {}^{i-1}v_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}P_i + \dot{d}_{i+1} {}^{i}Z_i $$

$$ {}^{i}v_i = {}^{i}R_{i-1}({}^{i-1}v_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}P_i) + \dot{d}_{i+1} {}^{i}Z_i $$

Velocity propagation summary

- Prismatic joint

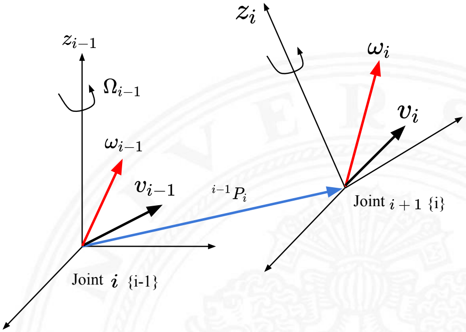
$${}^{i}v_i = {}^{i}R_{i-1}({}^{i-1}v_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}P_i) + \dot{d}_{i+1}{}^{i}Z_i$$

$${}^{i}\omega_i = {}^{i}R_{i-1}{}^{i-1}\omega_{i-1}$$

- Revolute joint

$${}^{i}v_i = {}^{i}R_{i-1}({}^{i-1}v_{i-1} + {}^{i-1}\omega_{i-1} \times {}^{i-1}P_i)$$

$${}^{i}\omega_i = {}^{i}R_{i-1}{}^{i-1}\omega_{i-1} + \dot{\theta}_{i+1}{}^{i}Z_i$$



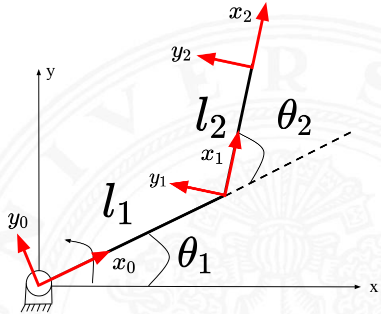
$$\begin{bmatrix} {}^{0}v_n \\ {}^{0}\omega_n \end{bmatrix} = \begin{bmatrix} {}^{0}R_n & 0 \\ 0 & {}^{0}R_n \end{bmatrix} \begin{bmatrix} {}^{n}v_n \\ {}^{n}\omega_n \end{bmatrix}$$

Given the 2dof planar robot, find the velocity of the origin of {2} w.r.t. {2} and {0}.

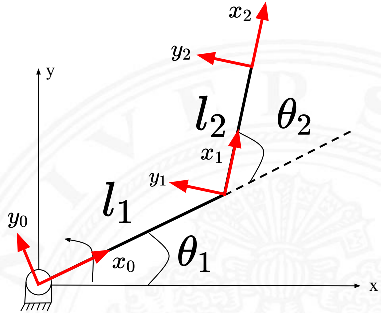
$^0\omega_0 =$ $\quad$, $^0v_0 =$

$^1\omega_1 =$

$^1v_1 =$



Example



How to simplify the calculation of end-effector velocity?

Joint velocities $\Leftrightarrow$ End-effector velocities

$$\Downarrow$$

Jacobian

Definition

In the field of robotics, we generally use Jacobians to relate joint velocities to Cartesian velocities of the end-effecter.

$$x = f(q), \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix} = \begin{bmatrix} f_1(q) \\ f_2(q) \\ \dots \\ f_n(q) \end{bmatrix} \tag{26}$$

- x is the Cartesian location of the end-effector
- m is number of degree of freedom in the Cartesian space
- Define $q = [q_1, q_2, ..q_n]^T$, $q_1, q_2, ..q_n$ are joint variables of an n-link manipulator

Jacobian of a manipulator (cont.)

By the chain rule of differentiation:

$$\delta x_1 = \frac{\partial f_1}{\partial q_1}\delta q_1 + \ldots + \frac{\partial f_1}{\partial q_n}\delta q_n$$

$$\vdots$$

$$\delta x_m = \frac{\partial f_m}{\partial q_1}\delta q_1 + \ldots + \frac{\partial f_m}{\partial q_n}\delta q_n$$

$$\delta x = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ldots & \vdots \\ \frac{\partial f_m}{\partial q_1} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix} \cdot \delta q \tag{27}$$

$$\delta x_{(m\times 1)} = J_{(m\times n)}\delta q_{(n\times 1)} \quad \textit{where} \quad J_{ij}(q) = \frac{\partial}{\partial q_j} f_i(q) \tag{28}$$

Jacobian of a manipulator (cont.)

$$\partial x_{(m \times 1)} = J_{(m \times n)} \partial q_{(n \times 1)}$$

$$\dot{x}_{(m \times 1)} = J_{(m \times n)} \dot{q}_{(n \times 1)}$$

- A Jacobian-matrix is a multidimensional representation of partial derivatives.
- If we divide both sides with the differential time element, we can think of the Jacobian as mapping velocities in q to those in x.
- Jacobians are time-varying linear transformations.

Jacobian of a manipulator (cont.)

- ${}^0\omega_n$ to be the angular velocity of the end effector
- ${}^0v_n$ is the linear velocity of the end effector
- The **Jacobian** matrix consists of two components, that solve the following equations:

$$ {}^0v_n = {}^0J_v\dot{q} \qquad \text{and} \qquad {}^0\omega_n = {}^0J_w\dot{q} $$

The manipulator Jacobian

$$ J = \begin{bmatrix} J_v \\ J_w \end{bmatrix}, \quad \begin{bmatrix} {}^0v_n \\ {}^0\omega_n \end{bmatrix} = \begin{bmatrix} J_v \\ J_w \end{bmatrix} \dot{q} \tag{29} $$

Angular velocity ${}^{i-1}\omega_i$ is:

$$ {}^{i-1}\omega_i = {}^{i-1}\omega_{i-1} + {}^{i-1}R_i \dot{\theta}_{i+1} {}^{i}Z_i $$

We get:

$$ \begin{aligned} {}^{0}\omega_n &= p_1 \dot{q}_1 {}^{0}Z_0 + p_2 \dot{q}_2 {}^{0}R_1 {}^{1}Z_1 + \ldots + p_n \dot{q}_n {}^{0}R_{n-1} {}^{n-1}Z_{n-1} \\ &= p_1 \dot{q}_1 {}^{0}Z_0 + p_2 \dot{q}_2 {}^{0}Z_1 + \ldots + p_n \dot{q}_n {}^{0}Z_{n-1} \end{aligned} $$

where:

$$ p_i = \begin{cases} 0 & \text{if joint i is prismatic} \\ 1 & \text{if joint i is revolute} \end{cases} \tag{30} $$

Angular velocity Jacobian (cont.)

The Angular Velocity Jacobian

$$J_w = [p_1 {}^0Z_0 \quad p_2 {}^0Z_1 \quad \dots \quad p_n {}^0Z_{n-1}] \tag{31}$$

(Hint: J_w is a $3xn$ matrix.)

The linear velocity of the end effector is: ${}^0v_n = {}^0\dot{x}_n = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$

By the chain rule of differentiation:

$${}^0\dot{x}_n = \frac{\partial^0 x_n}{\partial q_1}\dot{q}_1 + \frac{\partial^0 x_n}{\partial q_2}\dot{q}_2 + \ldots + \frac{\partial^0 x_n}{\partial q_n}\dot{q}_n$$

therefore the linear part of the Jacobian is:

$$J_v = \begin{bmatrix} \frac{\partial^0 x_n}{\partial q_1} & \frac{\partial^0 x_n}{\partial q_2} & \ldots & \frac{\partial^0 x_n}{\partial q_n} \end{bmatrix} \tag{32}$$

Computing the final Jacobian

Two approaches:

1. derive v, ω for each link until the end-effector
2. use the explicit form

Computing the final Jacobian

- get ${}^0J_v$

$$ {}^0T_6 = \begin{bmatrix} {}^0R_N & \boxed{{}^0P_N} \\ 0 & 1 \end{bmatrix} \quad {}^0x \quad \longrightarrow {}^0v_n \quad \longrightarrow \quad {}^0J_v $$

- get ${}^0J_\omega$

$$ J_w = [p_1 {}^0Z_0 \quad p_2 {}^0Z_1 \quad \dots \quad p_n {}^0Z_{n-1}] $$

- ${}^0x_i$ is equal to the first three elements of the 4th column of matrix ${}^0T_i$
- ${}^0Z_i$ is equal to the first three elements of the 3rd column of matrix ${}^0T_i$

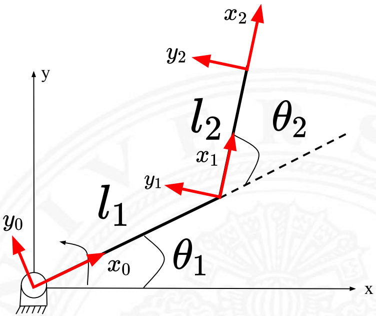
${}^0T_i$ has to be computed for every joint.

Example1

$$ {}^0\omega_2 = {}^0R_2 {}^2\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} $$

$$ {}^0v_2 = {}^0R_2 {}^2v_2 = \begin{bmatrix} -l_1 s_1 \dot{\theta}_1 - l_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ l_1 c_1 \dot{\theta}_1 + l_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix} $$

Give the ${}^0J$ Jacobian matrix.



For a 3-DOF robot, given the following transformation matrices, find the Jacobian ${}^0J$.

$$ {}^0T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^1T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^2T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & e \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^3T_4 = \begin{bmatrix} 1 & 0 & 0 & f \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where h, e, f are the length of the $1^{st}, 2^{nd} and 3^{rd}$ link, respectively.

$$ {}^0T_4 = \begin{bmatrix} c_1 c_{23} & -c_1 s_{23} & s_1 & e c_1 c_2 + f c_1 c_{23} \\ s_1 c_{23} & -s_1 c_{23} & -c_1 & e s_1 c_2 + f s_1 c_{23} \\ s_{23} & c_{23} & 0 & h + e s_2 + f s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Calculate ${}^0T_1, {}^0T_2, {}^0T_3, {}^0T_4$:

$$
{}^0T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^0T_2 = {}^0T_1 {}^1T_2 = \begin{bmatrix} c_1c_2 & -s_2c_1 & s_1 & 0 \\ s_1c_2 & -s_1s_2 & -c_1 & 0 \\ s_2 & c_2 & 0 & h \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_3 = {}^0T_2 {}^2T_3 = \begin{bmatrix} -s_2s_3c_1 + c_1c_2c_3 & -s_2c_1c_3 - s_3c_1c_2 & s_1 & ec_1c_2 \\ -s_1s_2s_3 + s_1c_2c_3 & -s_1s_2c_3 - s_1s_3c_2 & -c_1 & es_1c_2 \\ s_2c_3 + s_3c_2 & -s_2s_3 + c_2c_3 & 0 & es_2 + h \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^0T_4 = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & ec_1c_2 + fc_1c_{23} \\ s_1c_{23} & -s_1c_{23} & -c_1 & es_1c_2 + fs_1c_{23} \\ s_{23} & c_{23} & 0 & h + es_2 + fs_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Example2 (cont.)

$$
{}^{0}J = \begin{bmatrix} J_v \\ J_w \end{bmatrix} = \begin{bmatrix}
-es_1c_2 - fs_1c_{23} & -ec_1s_2 - fc_1s_{23} & -fc_1s_{23} \\
ec_1c_2 + fc_1c_{23} & -es_1s_2 - fs_1s_{23} & -fs_1s_{23} \\
0 & ec_2 + fc_{23} & fc_{23} \\
0 & s_1 & s_1 \\
0 & -c_1 & -c_1 \\
1 & 0 & 0
\end{bmatrix}
$$

Changing a Jacobian's frame of reference

Given a Jacobian written in frame $\{B\}$,

$$\begin{bmatrix} {}^B v_n \\ {}^B \omega_n \end{bmatrix} = \begin{bmatrix} {}^B J_v \\ {}^B J_w \end{bmatrix} \dot{q}$$

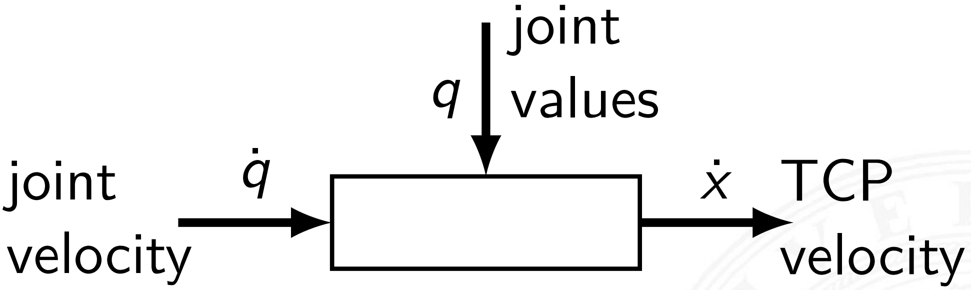
A 6 x 1 Cartesian velocity vector given in $\{B\}$ is described relative to $\{A\}$ by the transformation

$$\begin{bmatrix} {}^A v_n \\ {}^A \omega_n \end{bmatrix} = \begin{bmatrix} {}^A R_B & 0 \\ 0 & {}^A R_B \end{bmatrix} \begin{bmatrix} {}^B v_n \\ {}^B \omega_n \end{bmatrix}$$

Hence, we can get

$$\begin{bmatrix} {}^A v_n \\ {}^A \omega_n \end{bmatrix} = \begin{bmatrix} {}^A R_B & 0 \\ 0 & {}^A R_B \end{bmatrix} \begin{bmatrix} {}^B J_v \\ {}^B J_w \end{bmatrix} \dot{q} \tag{33}$$

Inverse Jacobian



Question

Is the Jacobian invertible?

If it is, then: $\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{\mathbf{x}}$

$\Longrightarrow$ to move the the end effector of the robot in Cartesian Space with a certain velocity.

Singular Configurations

For most manipulators there exist values of $\mathbf{q}$ where the Jacobian gets singular.

Singularity

$$\det J = 0 \Longrightarrow J \text{ is not invertible}$$

Such configurations are called singularities of the manipulator.

From the Task Space perspective:

- reduce the degree of freedom in velocity domain in task space

From the Joint Space perspective:

- Infinite solutions to the inverse kinematics problem may exist
- Near the singularity, small velocities in operational space may cause large velocities in the joint space.

Singularities (cont.)

Two Main types of Singularities:

- **Workspace boundary singularities** occur when the manipulator is fully stretched out or folded back on itself in such a way that the end-effector is at or very near the boundary of the workspace.
- **Workspace internal singularities** occur away from the workspace boundary; they generally are caused by a lining up of two or more joint axes

Singular Configurations – Workarounds

$N = 6$ For fully actuated robots, the Jacobian (6×6) is invertible

$$\delta x_{(m\times 1)} = J_{(m\times n)}\delta q_{(n\times 1)} \quad where \quad J_{ij}(q) = \frac{\partial}{\partial q_j} f_i(q)$$

- m is number of degree of freedom of the manipulator in the Cartesian space
- n is the number of joint variables of the manipulator

Singular Configurations – Workarounds

$N = 6$ For fully actuated robots, the Jacobian (6×6) is invertible

$N < 6$ Under actuated robots ($6 \times N$)

$\Longrightarrow$ remove some spatial degrees of freedom, get a square Jacobian matrix.

Example:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \end{bmatrix} = J_{2\times 2} \begin{bmatrix} dq_1 \\ dq_2 \end{bmatrix}$$

for a 2-joint planar manipulator

Singular Configurations – Workarounds

$N = 6$ For fully actuated robots, the Jacobians (6×6) are invertible

$N < 6$ Under actuated robots ($6 \times N$)

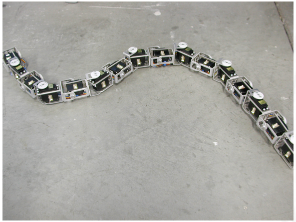
$\Longrightarrow$ remove some spatial degrees of freedom

$N > 6$ Over actuated robots ($6 \times N$)

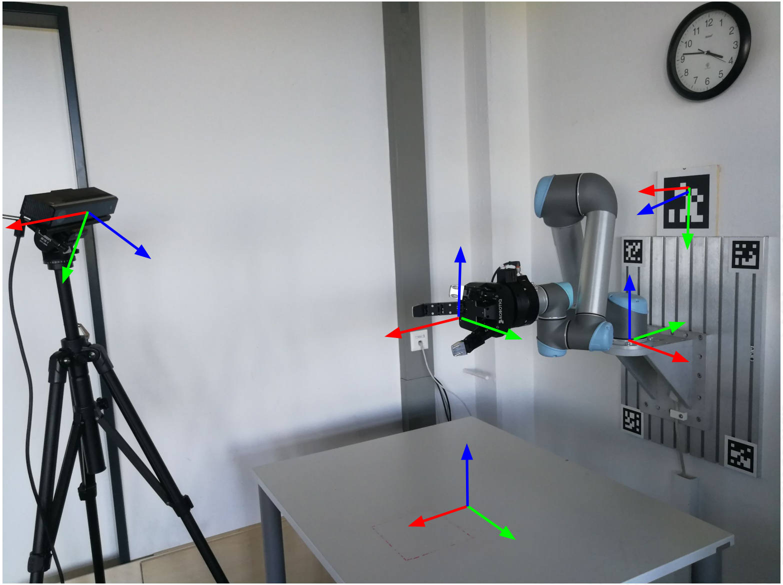
- have spare joints
- use the `pseudoinverse of J`

$$\dot{q} = J(q)^+ v \tag{34}$$

$$J^+ = (J^T J)^{-1} J^T \tag{35}$$



UR5 example



23

[23] https://www.youtube.com/watch?v=6Wmw4lUHIX8

Introduction to Robotics

Lecture 5

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Joint velocities $\Leftrightarrow$ End-effector velocities

$\Downarrow$

Jacobian

- Jacobian

$$\delta x_{(m\times 1)} = J_{(m\times n)}\delta q_{(n\times 1)} \quad where \quad J_{ij}(q) = \frac{\partial}{\partial q_j} f_i(q)$$

- Angular/Linear velocity Jacobian

$$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix}, \quad \begin{bmatrix} {}^0v_n \\ {}^0\omega_n \end{bmatrix} = \begin{bmatrix} J_v \\ J_w \end{bmatrix} \dot{q}$$

- Computation of the final Jacobian

Singularities

- Geometric singularities:
  - for any two revolute joints, the joint axes are collinear
  - any three parallel rotation axes lie in a plane
  - any four rotational axes intersect at a point
  - any three coplanar revolute axes intersect at a point
- Mathematical singularities:

$$\det J = 0 \Longrightarrow J \text{ is not invertible}$$

Where the determinant is equal to zero, the Jacobian has lost full rank and is singular.

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

- Trajectory and related concepts
- Trajectory generation
- Solutions of trajectory generation
- Optimizing motion
- Application

Trajectory Generation 2

Dynamics

Robot Control

Outline (cont.)

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Trajectory

Definition

A trajectory is a time history of

- position,
- velocity and
- acceleration

for each DOF

Describes motion of TCP frame relative to base frame

- abstract from joint configuration

Trajectories in multidimensional space

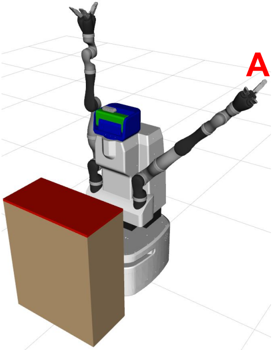
- Changes in position, velocity and acceleration of all joints are analyzed over a period of time
- Trajectory with n DOF is a parameterized function $q(t)$ with values in its motion region.
- Trajectory $q(t)$ of a robot with n DOF is then a vector of n parameterized functions $q_i(t), i \in \{1 \dots n\}$ with one common parameter t:

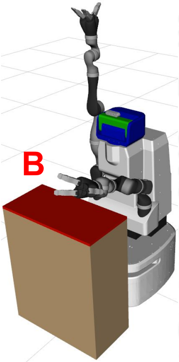
$$q(t) = [q_1(t), q_2(t), \dots, q_n(t)]^T$$

Problem

The robot is at point A and wants move to point B.

- How does the robot get to point B?
- How long does it take the left arm to get to point B?
- Which possible constraints exist for moving from A to B?





Trajectory generation

Problem

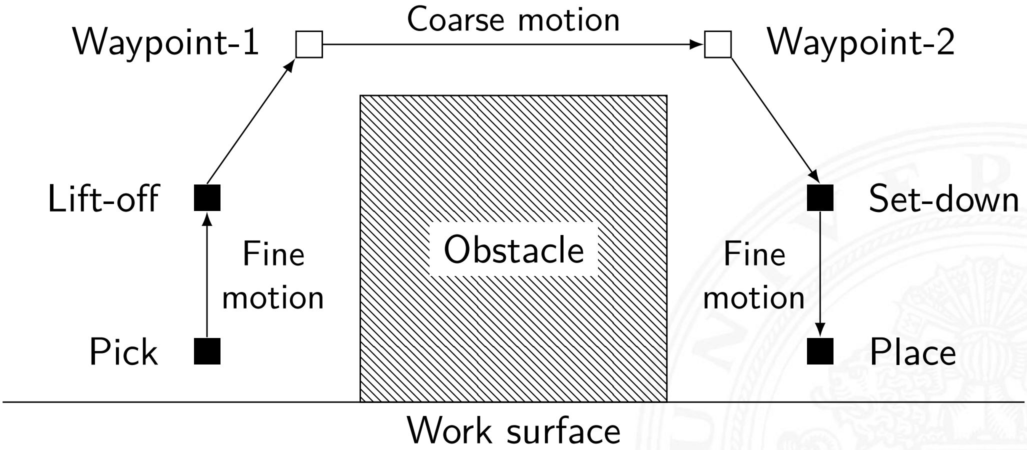
The robot is at point A and wants move to point B.

- How does the robot get to point B?
- How long does it take the left arm to get to point B?
- Which possible constraints exist for moving from A to B?

Solution

- generate a possible and smooth trajectory
- describe intermediate poses (waypoints)
  - usually fixed temporal intervals
- obey the physical boundaries of the mechanics of the robot

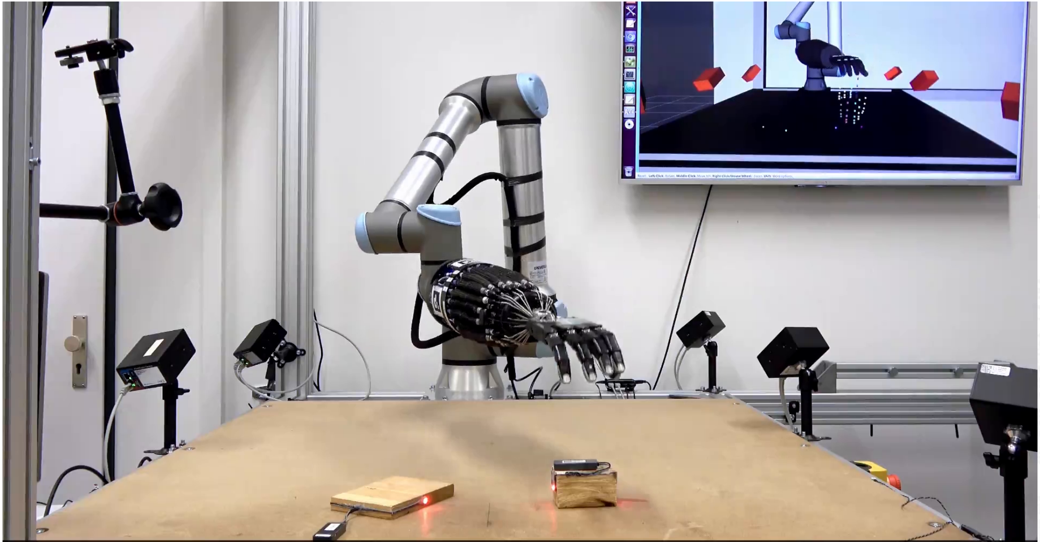
Trajectory planning



Trajectory planning (cont.)

Pick	$pos_{Start} = object$, $vel_{Start} = 0$, $acc_{Start} = 0$
Lift-off	limited velocity and acceleration
Motion	continuous via waypoints, full velocity and acceleration
Set-down	similar to Lift-off
Place	similar to Pick

Trajectory planning (cont.)



UR10e arm, Shadow C5 hand, feed-forward policy, 10 demonstrations, video speed: 2.4


fresh
CHEESE
fresh
BUTTER
soy
milk
milk
orange
juice

Generation of trajectories

Task

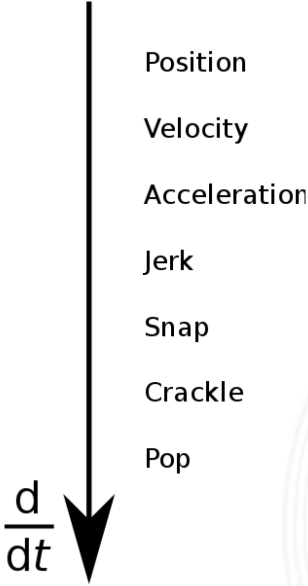
- find a smooth trajectory for moving the robot from start to goal pose
- use continuous functions of time

Continuity of Trajectories

- A trajectory is C^k-continuous, if all derivatives up to the k-th (including) exist and are continuous.
- A trajectory is called *smooth*, if it is at least C^2-continuous

- $q(t)$ is the trajectory,
- $\dot{q}(t)$ is the velocity,
- $\ddot{q}(t)$ is the acceleration,
- $\dddot{q}(t)$ is the jerk

Time-derivatives of position



Generation of trajectories

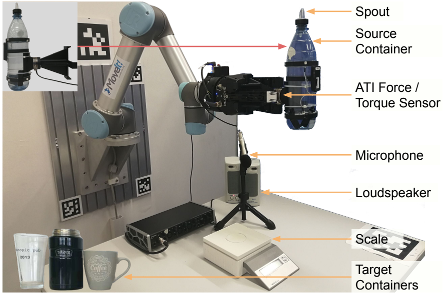
Task

- find trajectory for moving the robot from start to goal pose
- use continuous functions of time

Representation solution:

- calculation of Cartesian trajectories for the TCP
- calculation for trajectories in joint space

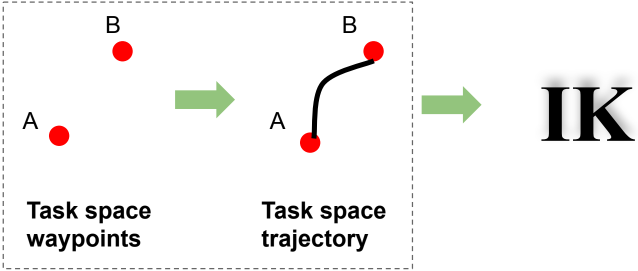
Generation of trajectories (cont.)

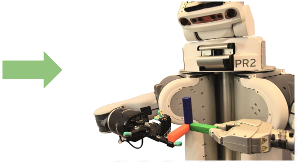


Pouring setup

Pushing setup

Trajectories in Cartesian space





Joint position commands

Advantages:

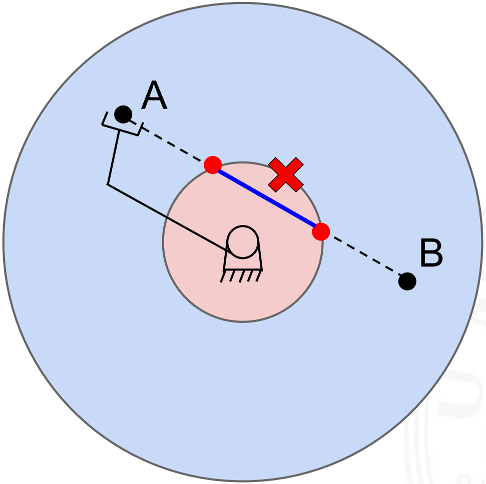
- near to the task specification
- advantageous for collision avoidance
- can specify the spatial shape of the path

Disadvantages:

- more expensive at run time
  - after the path is calculated need joint angles in a lot of points by IK
- Discontinuity problems

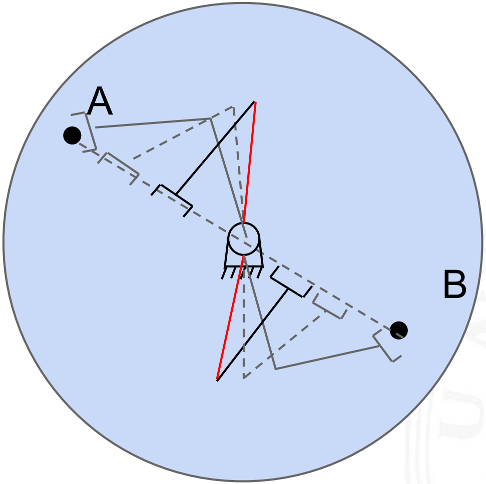
Difficulties of trajectories in Cartesian space

1. Waypoints cannot be realized
   - workspace boundaries, object collision, self-collision



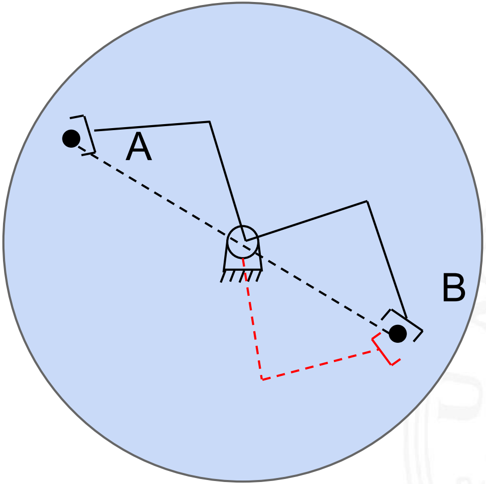
Difficulties of trajectories in Cartesian space (cont.)

2. Velocities in the vicinity of singular configurations are too high

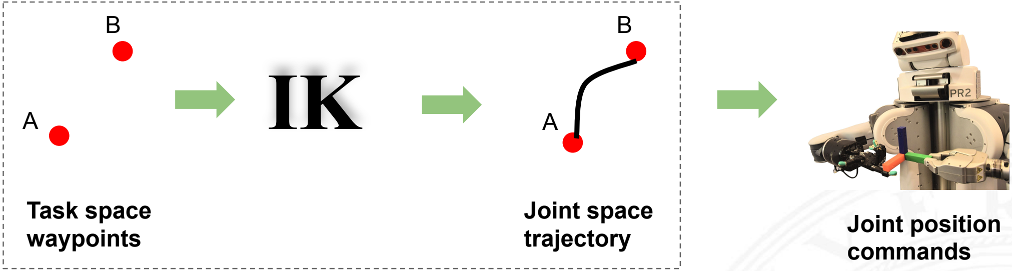


Difficulties of trajectories in Cartesian space (cont.)

3. Start and end configurations can be achieved, but there are different solutions
   - ambiguous solutions



Trajectories in joint space



Joint space:

- no inverse kinematics in joint space required
- the planned trajectory can be immediately applied
- no problem with singularities
- physical joint constraints can be considered

Primitive solution

Naive approach

Set the pose for the next time step (e.g. 10 ms later) to B.

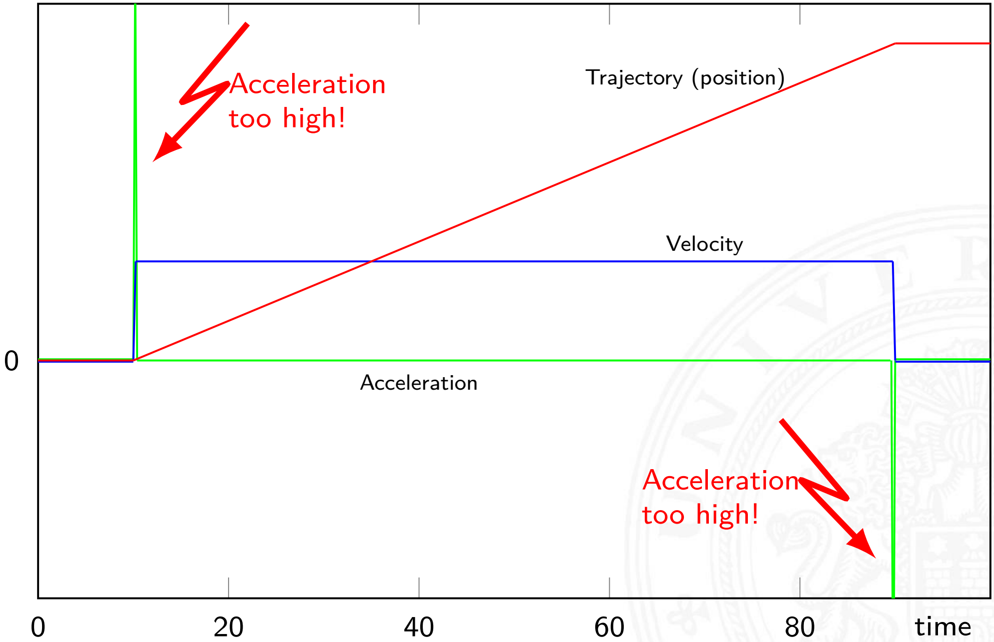
- possible only in simulation
- the moving distance for a manipulator at the next time step may be too large (velocity approaches ∞)

Linear interpolation

Next best approach

- divide distance between A and B to shorter (sub-)distances
- use linear interpolation for these (sub-)distances
- respect the maximum velocity constraint

Linear interpolation – visualization



Linear interpolation – constraints

Problem

The physical constraints are violated

- joint velocity is limited by maximum motor rotation speed
- joint acceleration is limited by maximum motor torque

Implicitly these contraints are valid for motion in cartesian space.

- robot dynamics (joint moments resulting from the robot motion) affect the boundary condition

Solution

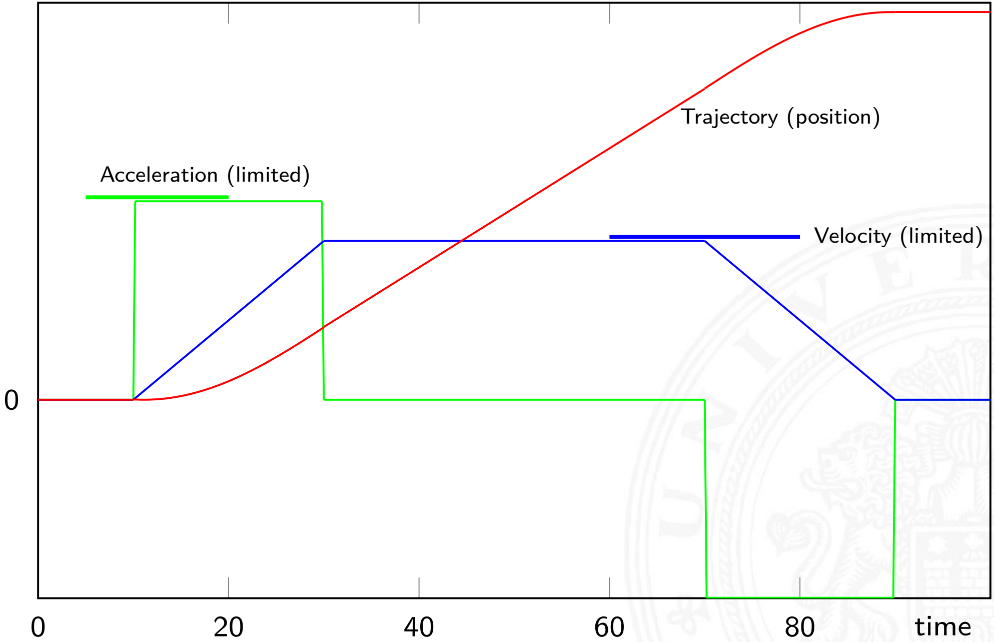
- dynamical trajectory generation
- advanced optimization methods $\rightarrow$ current topic of research

Linear interpolation – improvement

Next best approach

- Limitation of joint velocity and acceleration
- Two different methods
  - trapezoidal interpolation
  - polynomial interpolation

Trapezoidal interpolation – visualization



Trapezoidal interpolation – summary

- `Position` is quadratic during acceleration and deceleration, and linear elsewhere
  - Linear segment with Parabolic Blends
- `Velocity` linearly ramps up/down to maximum velocity
- `Acceleration` and deceleration is constant for each trajectory segment.

Trapezoidal interpolation – summary (cont.)

- consider joint velocity and acceleration contraints
- optimal time usage (move with maximum acceleration and velocity)
- acceleration is not differentiable (the jerk is not continuous)
- start and end velocity equals 0
  - not sensible for concatenating trajectories
  - improved by polynomial interpolation

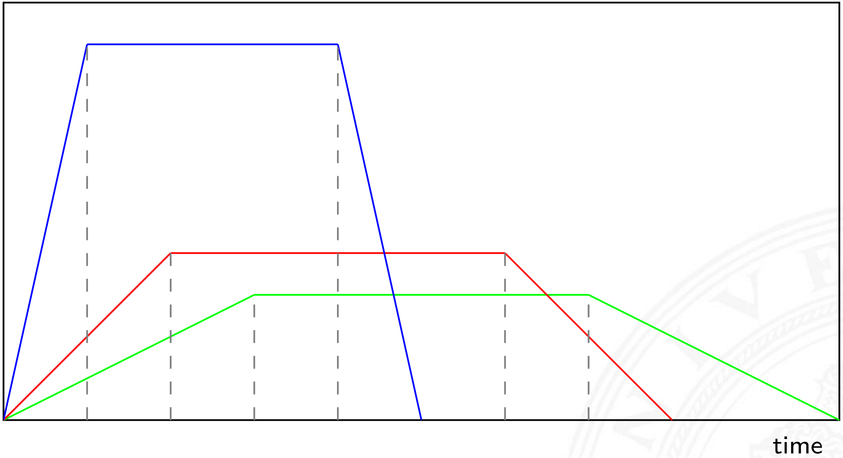
Trapezoidal interpolation – constraints

Problem

Multidimensional trapezoidal interpolations

- different run time for joints (or cartesian dimensions)
- multiple velocity and acceleration contraints
- results in various `time switch points`
  - from acceleration to continuous velocity
  - from continuous velocity to deceleration
  - moving along a line in joint/cartesian space is impossible.

Trapezoidal interpolation – constraints

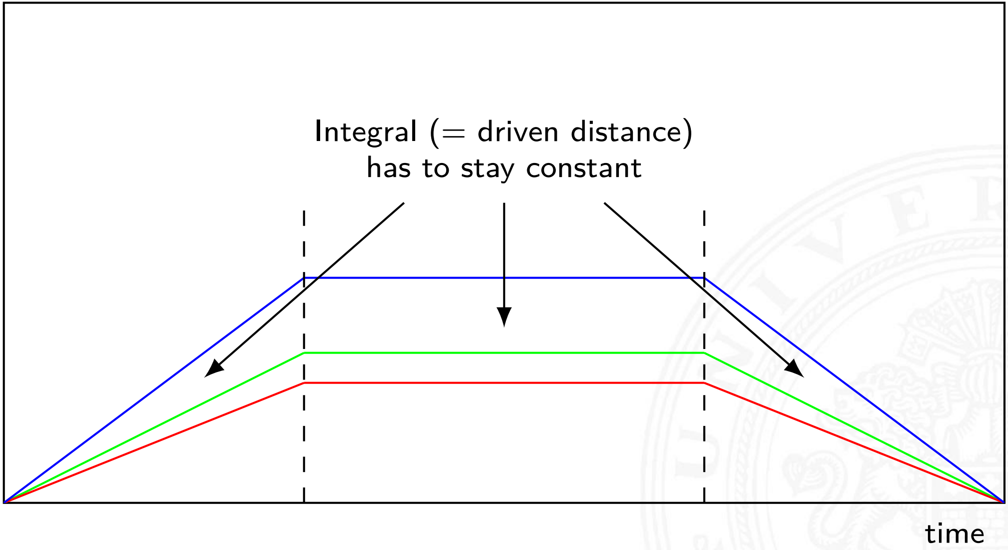


Solution

- Normalization to the joint that takes longest to reach its goal
- Synchronize phase switching points and overall execution time

Trapezoidal interpolation – normalization

Normalize to the slowest joint



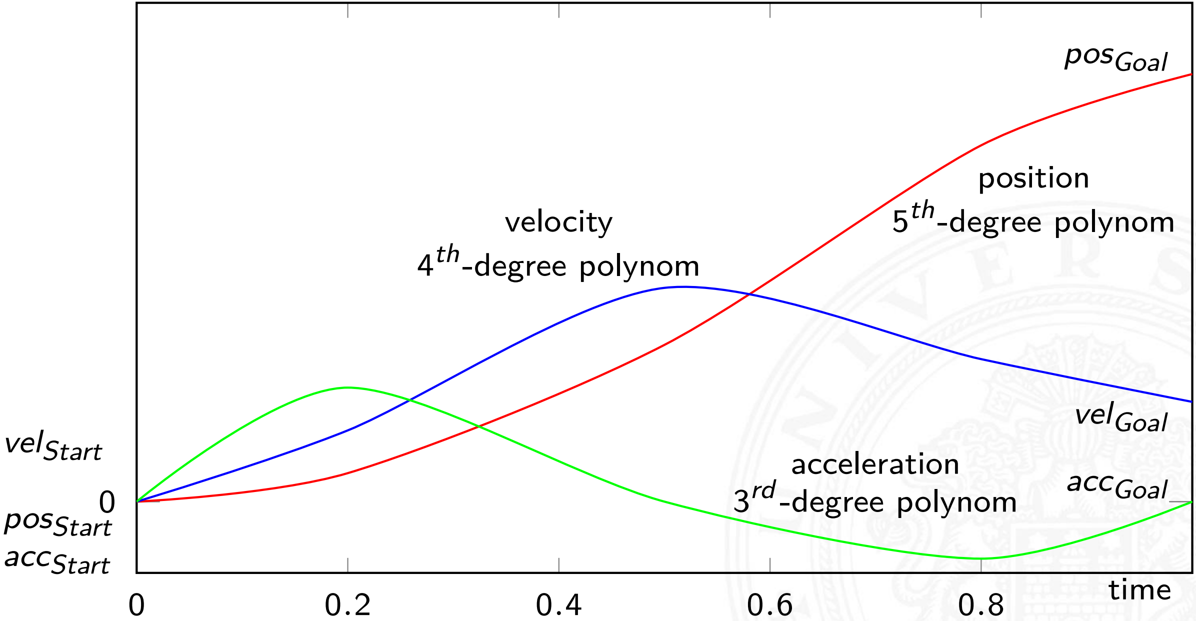
Polynomial interpolation

- Consider velocity and acceleration boundary conditions
  - calculation of extremum and duration of trajectory
- Acceleration differentiable
  - continuous jerk
  - `smooth trajectory`
  - interesting only in the theory – for momentum control
- Start and end velocity may be $\neq 0$
  - sensible for concatenating trajectories

Polynomial interpolation (cont.)

- Usually a polynomial with degree of 3 (cubic spline) or 5
- Calculation of coefficient with respect to boundary constraints
  - 3^{rd}-degree polynomial: consider 4 boundary constraints
    - position and velocity; start and goal
  - 5^{th}-degree polynomial: consider 6 boundary constraints
    - position, velocity and acceleration; start and goal

Polynomial interpolation (cont.)



Cubic polynomials between two configurations

- third-degree polynomial $\Rightarrow$ four constraints(position and velocity; start and goal):

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{\theta(t)} = a_1 + 2a_2 t + 3a_3 t^2$$

$$\ddot{\theta(t)} = 2a_2 + 6a_3 t$$

- if the start and end velocity is 0 then

$$\theta(0) = \theta_0 \tag{36}$$

$$\theta(t_f) = \theta_f \tag{37}$$

$$\dot{\theta}(0) = 0 \tag{38}$$

$$\dot{\theta}(t_f) = 0 \tag{39}$$

Cubic polynomials between two configurations (cont.)

- The solution

$$
\begin{aligned}
\text{eq. (36)} \quad a_0 &= \theta_0 \\
\text{eq. (38)} \quad a_1 &= 0 \\
a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) \\
a_3 &= -\frac{2}{t_f^3}(\theta_f - \theta_0)
\end{aligned}
$$

- Similar to the previous example:
  - positions of waypoints are given (same)
  - velocities of waypoints are different from 0 (different)

$$\theta(0) = \theta_0 \tag{40}$$

$$\theta(t_f) = \theta_f \tag{41}$$

$$\dot{\theta}(0) = \dot{\theta}_0 \tag{42}$$

$$\dot{\theta}(t_f) = \dot{\theta}_f \tag{43}$$

- The solution

$$\text{eq. (40)} \quad a_0 = \theta_0$$

$$\text{eq. (42)} \quad a_1 = \dot{\theta}_0$$

$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f$$

$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0)$$

Velocity calculation at the waypoints

- Manually specify waypoints
  - based on cartesian linear and angle velocity of the tool frame
- Automatic calculation of waypoints in cartesian or joint space
  - based on heuristics
- Automatic determination of the parameters
  - based on continous acceleration at the waypoints

Example 5^{th}-degree

$$\theta(x) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

Boundary conditions for start $(x = t_0)$ and goal $(x = t_d)$:

- $\theta(t_0) = pos_{Start}, \; \theta(t_d) = pos_{Goal}$
- $\theta(\dot{t}_0) = vel_{Start}, \; (\dot{t}_d) = vel_{Goal}$
- $\theta(\ddot{t}_0) = acc_{Start}, \; (\ddot{t}_d) = acc_{Goal}$

Remarks on generation of trajectories

- The smoothest curves are generated by infinitly often differentiable functions.
  - e^x
  - $sin(x), \; cos(x)$
  - $log(x)$ (for $x > 0$)
  - ...
- Polynomials are suitable for interpolation
  - Problem: oscillations caused by a degree which is too high
- Piecewise polynomials with specified degree are applicable
  - cubic polynomial
  - splines
  - B-Splines
  - ...

Factors for time optimal motion – Arc Length

If the curve in the n-dimensional space is given by

$$\mathbf{q}(t) = [q^1(t), q^2(t), \ldots, q^n(t)]^T$$

then the **arc length** can be defined as follows:

$$s = \int_0^t \|\dot{\mathbf{q}}(t)\|_2 \, dt$$

where $\|\dot{\mathbf{q}}(t)\|_2$ is the euclidean norm of vector $d\mathbf{q}(t)/dt$ and is labeled as a flow velocity along the curve.

$$\|\mathbf{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$$

Factors for time optimal motion – Arc Length (cont.)

With the following two points given
$\mathbf{p}_0 = \mathbf{q}(t_s)$ und $\mathbf{p}_1 = \mathbf{q}(t_f)$,
the arc length L between $\mathbf{p}_0$ and $\mathbf{p}_1$ is the integral:

$$L = \int_{\mathbf{p}_0}^{\mathbf{p}_1} ds = \int_{t_s}^{t_f} \|\dot{\mathbf{q}}(t)\|_2 \, dt$$

> *"The trajectory parameters should be calculated in the way that the arc length L under the given constraints has the shortest possible value."*

Factors for time optimal motion – Curvature

Curvature

Defines the `sharpness` of a curve. A straight line has zero curvature. Curvature of large circles is smaller than of small circles.

At first the *unit vector* of a curve $\mathbf{q}(t)$ can be defined as

$$\mathbf{U} = \frac{d\mathbf{q}(t)}{ds} = \frac{d\mathbf{q}(t)/dt}{ds/dt} = \frac{\dot{\mathbf{q}}(t)}{|\dot{\mathbf{q}}(t)|}$$

If s is the parameter of the *arc length* and $\mathbf{U}$ as the *unit vector* is given, the **curvature** of curve $\mathbf{q}(t)$ can be defined as

$$\kappa(s) = \left| \frac{d\mathbf{U}}{ds} \right|$$

Factors for time optimal motion – Bending Energy

The **bending energy** of a smooth curve $\mathbf{q}(t)$ over the interval $t \in [0, T]$ is defined as

$$E = \int_0^L \kappa(s)^2 ds = \int_0^T \kappa(t)^2 |\dot{\mathbf{q}}(t)| dt$$

where $\kappa(t)$ is the curvature of $\mathbf{q}(t)$.

> *"The bending energy E of a trajectory should be as small as possible under consideration of the arc length."*

Factors for time optimal motion – Motion Time

If a motion consists of n successive segments

$$q_j, j \in \{1 \dots n\}$$

then

$$u_j = t_{j+1} - t_j$$

is the required time for the motion in the segment $\mathbf{q}_j$. The total motion time is

$$T = \sum_{j=1}^{n-1} u_j$$

Minimum jerk trajectory

- Proposed by Flash & Hogan (1985) [7]
- Optimization Criterion minimizes the jerk in the trajectory

$$H(x(t)) = \frac{1}{2}\int_{t=t_i}^{t_f} \dddot{x}^2 dt$$

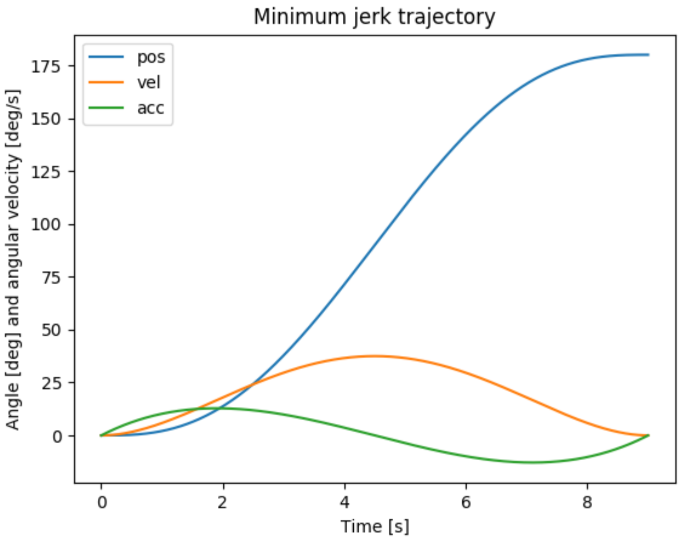
- The minimum-jerk solution can be written as:

$$x(t) = x_i + (x_i - x_f)(15(\frac{t}{d})^4 - 6(\frac{t}{d})^5 - 10(\frac{t}{d})^3)$$

- Predicts bell shaped velocity profiles

$$\dot{x}(t) = \frac{1}{d}(x_i - x_f)(60(\frac{t}{d})^3 - 30(\frac{t}{d})^4 - 30(\frac{t}{d})^2)$$

Minimum jerk trajectory (cont.)



Dynamical constraints for all joints

The borders for the minimum motion time T_{min} for the trajectory $\mathbf{q}_j^i(t)$ are defined over dynamical parameters of all joints.

For joint $i \in \{1 \dots n\}$ of trajectory part $j \in \{1 \dots m\}$ this kind of constraint can be described as follows

$$|\dot{q}_j^i(t)| \leq \dot{q}_{max}^i \tag{44}$$

$$|\ddot{q}_j^i(t)| \leq \ddot{q}_{max}^i \tag{45}$$

$$|m_j^i(t)| \leq m_{max}^i \tag{46}$$

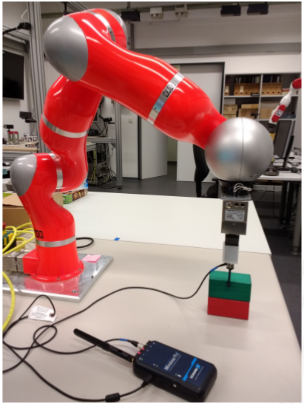
- m^i is the torque (moment of force) for the joint i and can be calculated from the dynamical equation (motion equation).
- $\dot{q}_{max}^i$, $\ddot{q}_{max}^i$ and m_{max}^i represent the important parameters of the dynamical capacity of the robot.

Reflexxes Motion Libraries

- Reflexxes Motion Libraries (Download, Overview)
- specialize on instantaneously generating smooth trajectories based on joint states and their limits
- Prof. Dr. Torsten Kroeger
  - paper: Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events [8]

Examples of using Reflexxes in TAMS

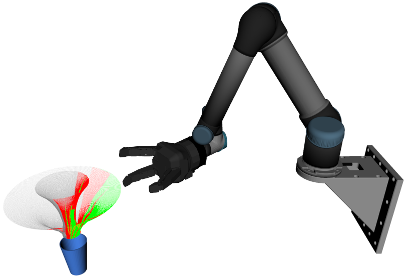
- Real-time object shape detection using ROS, the KUKA LWR4+ and a force/torque Sensor
  - to specify the target position and target velocity at the target position



24

Examples of using Reflexxes in TAMS (cont.)

- Adaptive pouring of liquids based on human motions using a Robotic Arm
  - to recalculate the speeds of a joint trajectory (returned by CCP) to match the original time-line of the



25

[24] https://tams.informatik.uni-hamburg.de/publications/2017/MSc_Stephan_Rau.pdf
[25] https://tams.informatik.uni-hamburg.de/publications/2018/MSc_Jeremias_Hartz.pdf

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

Introduction to Robotics

Lecture 6

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

- Recapitulation
- Approximation and Interpolation
- Interpolation methods
  - Bernstein-Polynomials
  - B-Splines

Dynamics

Robot Control

Outline (cont.)

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Trajectory generation – Recapitulation

The trajectory of a robot with n degrees of freedom (DoF) is a vector of n parametric functions with a common parameter:

Time

$$q(t) = [q^1(t), q^2(t), ..., q^n(t)]^T$$

Trajectory generation – Recapitulation (cont.)

- Deriving a trajectory yields
  - velocity $\dot{q}$
  - acceleration $\ddot{q}$
  - jerk $\dddot{q}$
- Jerk represents the change of acceleration over time, allowing for non-constant accelerations.
- A trajectory is C^k-continuous, if the first k derivatives of its path exist and are continuous.
- A trajectory is defined as *smooth* if it is at least C^2-continuous.

Trajectory generation – Recapitulation (cont.)

Trajectory generation

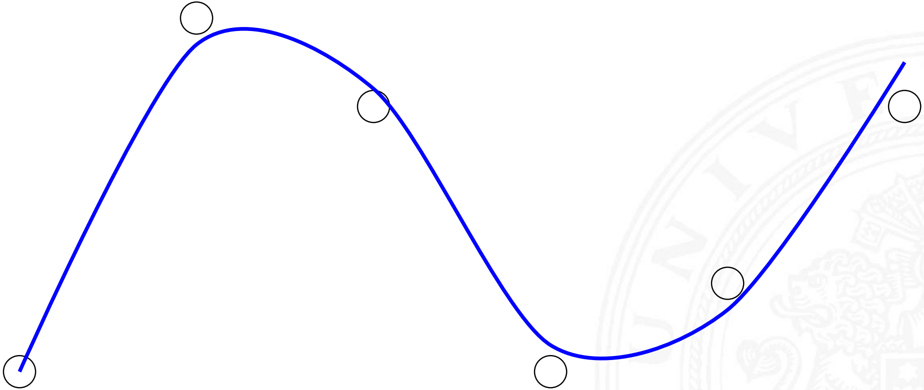
- Cartesian space
  - closer to the problem
  - better suited for collision avoidance
- Joint space
  - trajectories are immediately executable
  - limited to direct kinematics
  - allows accounting for joint angle limitations

Trajectory generation – Recapitulation (cont.)

- Linear interpolation
  - respect the minimum velocity constraint
- Trapezoidal interpolation
  - normalization
- Polynomial interpolation.
  - differentiable acceleration
  - cubic polynomials

Approximation

- Approximation of the relation between x and y (curve, plane, hyperplane) with a different function, given a limited number n of data points $D = \{\mathbf{x}_i, y_i\}$



Approximation

Definition

An approximation is a non-exact representation of something that is difficult to determine precisely (e.g. functions).

Necessary if

- equations are hard to solve
- mathematically too difficult or computationally too expensive

Advantages are

- simple to derive
- simple to integrate
- simple to compute

Approximation (cont.)

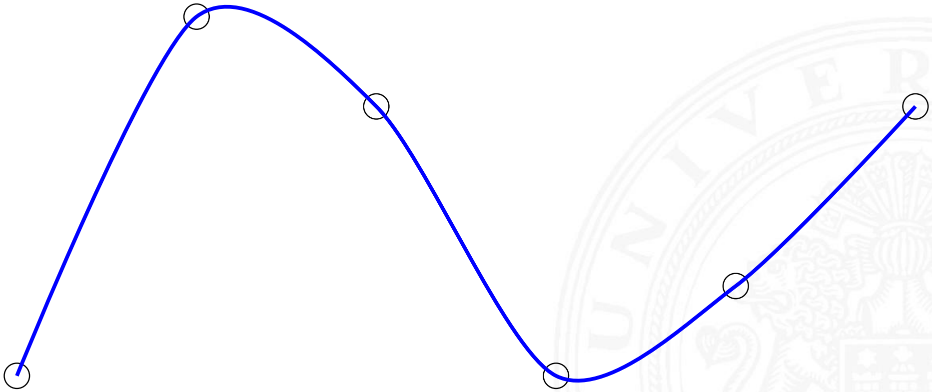
Stone-Weierstrass theorem (1937)

Theorem

- Every non-periodic continuous function on a closed interval can be approximated as closely as desired using algebraic polynomials.
- Every periodic continuous function can be approximated as closely as desired using trigonometric polynomials.

Interpolation

- A special case of approximation is interpolation, where the model exactly matches all data points.
  If many points are given or measurement data is affected by noise, approximation should preferably be used.



Interpolation

Definition

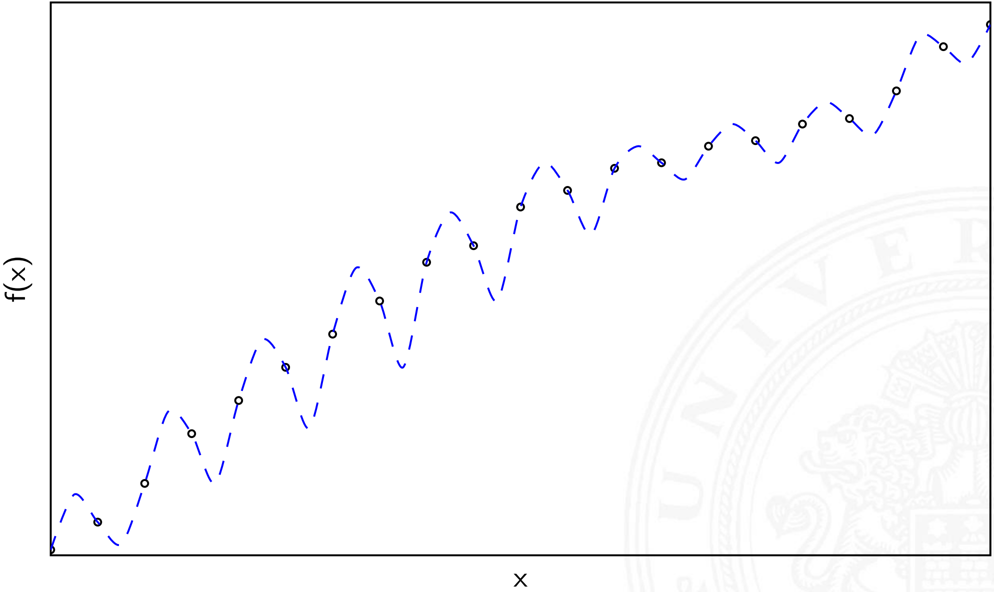
Interpolation is the process of constructing new data points within the range of a discrete set of known data points.

- Interpolation is a kind of approximation.
- A function is designed to match the known data points exactly, while estimating the unknown data in between in an useful way.
- In robotics, interpolation is common for computing trajectories and motion/-controllers.

Interpolation vs. Approximation

- Approximation: Fitting a curve to given data points.
  - Online tool: `https://mycurvefit.com/`
- Interpolation: Defining a curve exactly through all given data points
  - In the case of many, especially noisy, data points, approximation is often better suited than interpolation

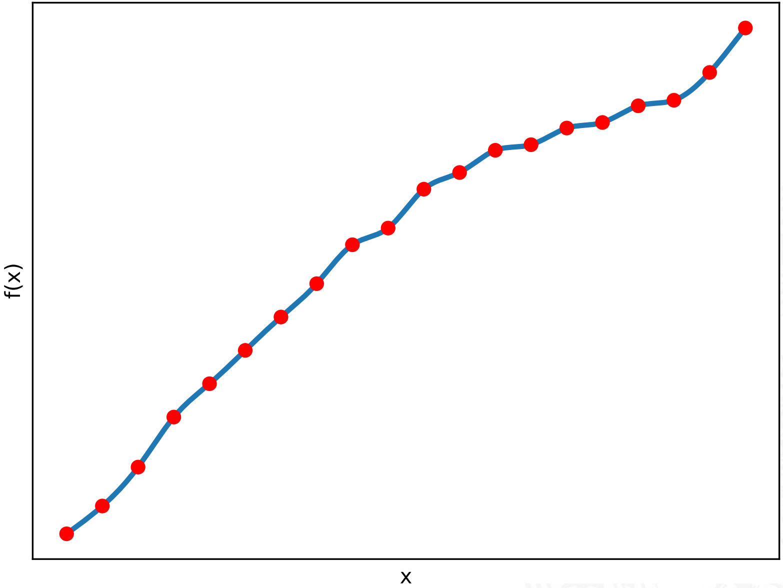
Interpolation with Overfitting



Overfitting example

Complete the sequence: $1, 3, 5, 7, \textbf{?}$

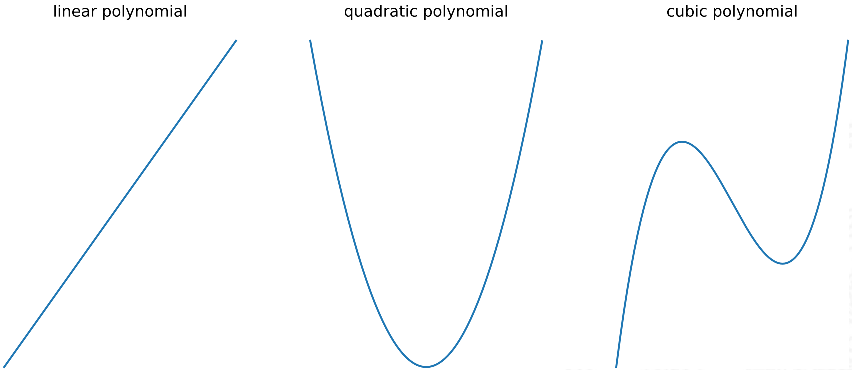
Interpolation without Overfitting



Interpolation basics

- Base
  - subset of a vector space
  - able to represent arbitrary vectors in space
    - finite linear combination
- Uniqueness
  - n^{th}-degree polynomials only have n zero-points
  - resulting system of equations is unique
- Oscillation
  - high-degree polynomials may oscillate due to many extrema
  - workaround: composition of sub-polynomials

Polynomial examples



Whatever the degree n of the polynomial is, there's $n-1$ turning points.

Interpolation methods

Generation of robot-trajectories in joint-space over multiple stopovers requires appropriate interpolation methods.

Some interpolation methods using polynomials:

- Bernstein-polynomials (Bézier curves)
- Basis-Splines (B-Splines)
- Lagrange-polynomials
- Newton-polynomials

Examples of polynomials interpolation:

- `http://polynomialregression.drque.net/online.php`
- `https://arachnoid.com/polysolve/`
- `http://www.hvks.com/Numerical/webinterpolation.html`

Bernstein-Polynomials

Bernstein-polynomials (named after Sergei Natanovich Bernstein) are real polynomials with integer coefficients.

Definition

Bernstein basis polynomials of degree k are defined as:

$$B_{i,k}(t) = \binom{k}{i}(1-t)^{k-i}t^i, \quad i = 0, 1, \ldots, k$$

where $\binom{k}{i}$ is the binomial coefficients, $\binom{k}{i} = \frac{i!}{k!(i-k)!}$ and $k \geqslant i \geqslant 0$.

Bernstein-Polynomials

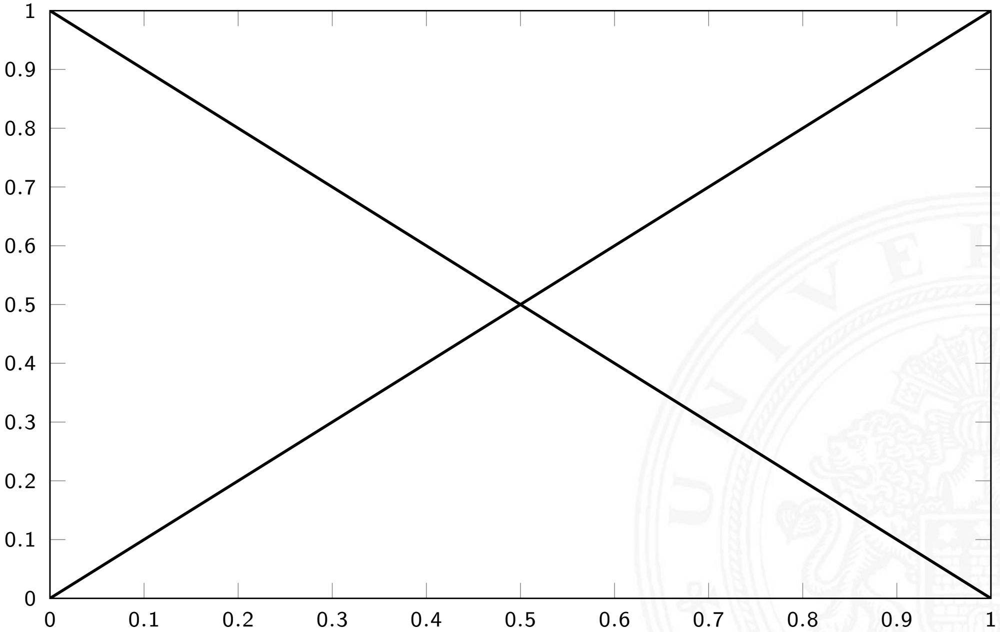
$$B_{i,k}(t) = \binom{k}{i}(1-t)^{k-i}t^i, \quad i = 0, 1, \dots, k$$

Bernstein Polynomials:

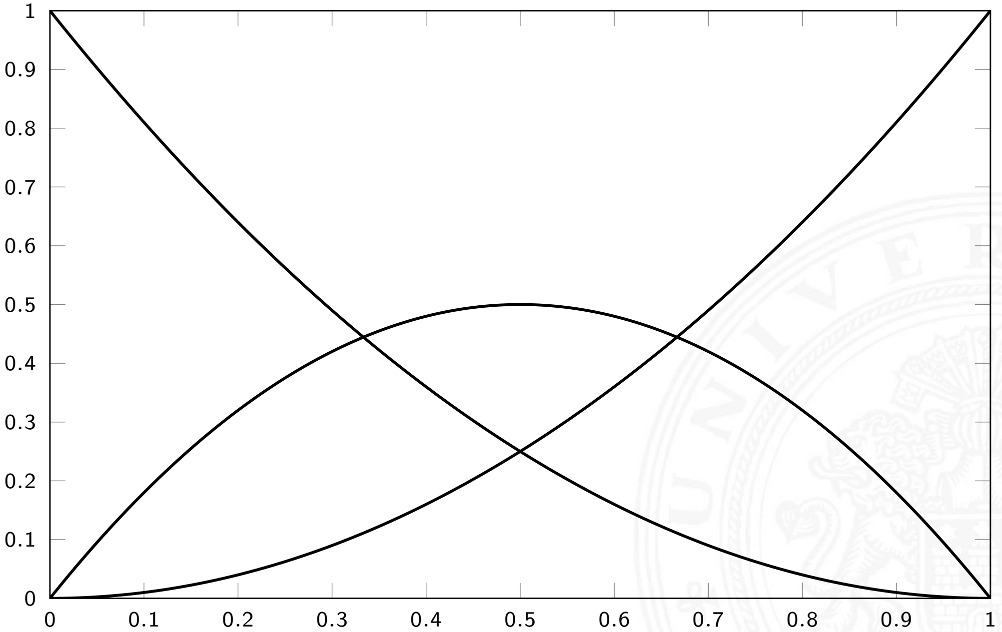
$$\mathbf{y} = \mathbf{b}_0 B_{0,k}(t) + \mathbf{b}_1 B_{1,k}(t) + \cdots + \mathbf{b}_k B_{k,k}(t)$$

where $\mathbf{b}_k$ is Bernstein coefficients.

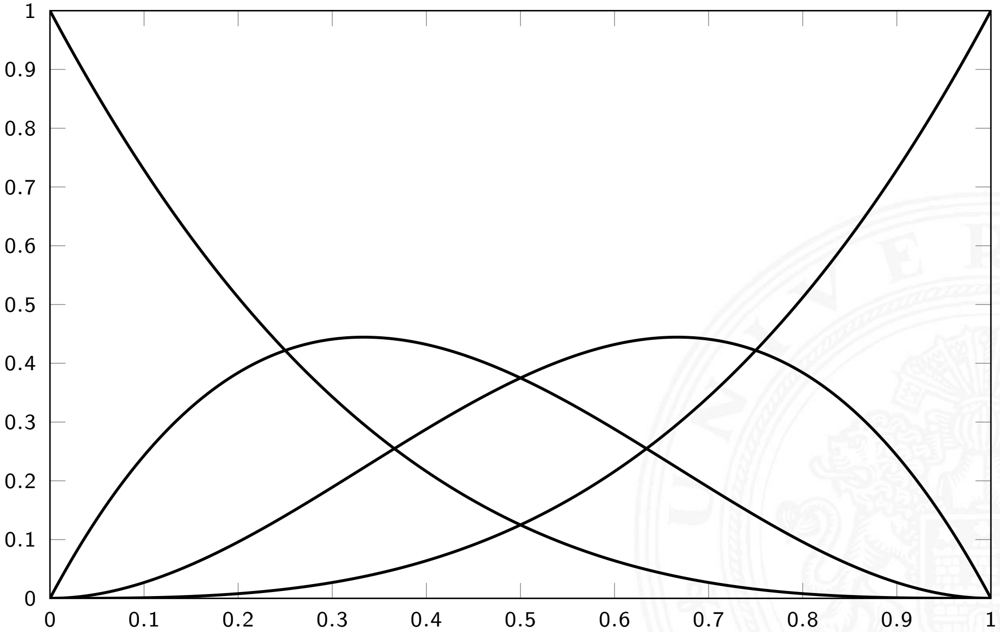
Polynomial of degree 1



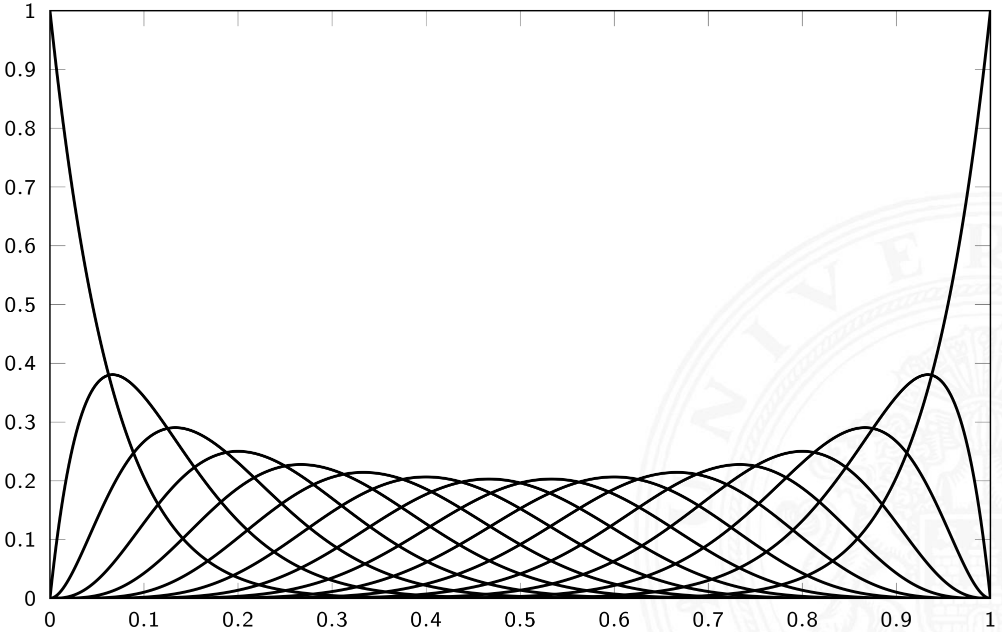
Polynomial of degree 2



Polynomial of degree 3



Polynomial of degree 15



Properties

Properties of Bernstein basis polynomials:

- `base property`: the Bernstein basis polynomials $[B_{i,k} : 0 \leq i \leq k]$ are linearly independent and form a base of the space of polynomials of degree $\leq k$,
- `positivity` $B_{i,k}(t) \geq 0$ for $t \in [0, 1]$,
- `decomposition of one`: $\sum_{i=0}^{k} B_{i,k}(t) \equiv \sum_{i=0}^{k} \binom{k}{i} t^i (1-t)^{k-i} \equiv 1$,
- `recursivity`: $B_{i,k-1}(t) = \frac{k-i}{k} B_{i,k}(t) + \frac{i+1}{k} B_{i+1,k}(t)$
- ...

Bézier curves

Bernstein Polynomials:

$$\mathbf{y} = \mathbf{b}_0 B_{0,k}(t) + \mathbf{b}_1 B_{1,k}(t) + \cdots + \mathbf{b}_k B_{k,k}(t)$$

where $\mathbf{b}_k$ is Bernstein coefficients.

If $\mathbf{b}_k$ is a set of control points $P_0, \cdots, P_n$, where n is called its order of the Bézier curve (n = 1 for linear, 2 for quadratic, etc.).

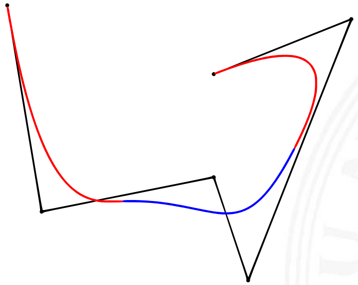
Animation of Bézier curves

Bézier curves for trajectory generation

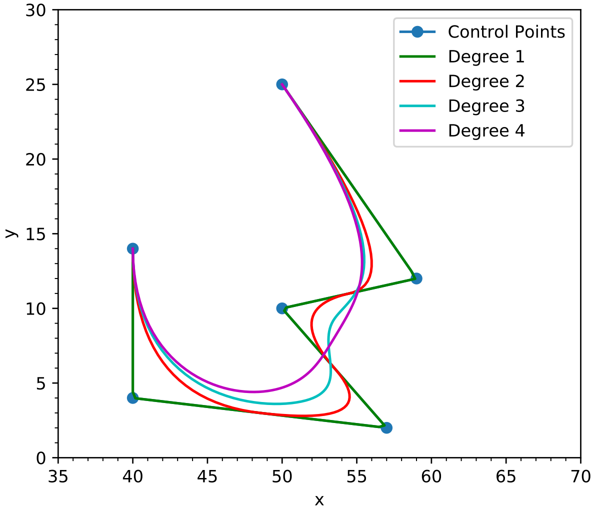
- Cubic polynomials (3^{rd}-degree) most used
- derivatives exist
  - velocity
  - acceleration
  - jerk
- provides smooth trajectory

B-spline curves and basis functions

- A `B-spline` or `basis spline` is a polynomial function that has minimal support with respect to a given degree, smoothness, and domain partition
- A B-spline curve of order k is composed of linear combinations of B-Splines (piecewise) of degree $k-1$ in a set of control points



B-spline curves and basis functions (cont.)



B-spline curves and basis functions (cont.)

Linear splines correspond to piecewise linear functions

Advantages:

- splines are more flexible than polynomials due to their piecewise definition
- still, they are relatively simple and smooth
- prevent strong oscillation
- Generally, 2^{nd} derivatives are continuous at intersections
- also applicable for representing surfaces (CAD modeling)

- the domain of B-splines are subdivided by

$$\mathbf{t} = (t_0, t_1, t_2, \ldots, t_m, t_{m+1}, \ldots, t_{m+k}),$$

where

- t : is the **knot vector**, has $m + k$ non-decreasing parameters
- m-th knot span is the half-open inteval $[t_m, t_{m+1})$
- m : is the number of **control points** to be interpolated
- k : is the **order** of the B-spline curve

Definition of B-splines

B-splines $N_{i,k}$ of order k:

- for $k = 1$, the degree is $p = k - 1 = 0$:

$$N_{i,1}(t) = \begin{cases} 1 & : \quad \text{for } t_i \leq t < t_{i+1} \\ 0 & : \quad \text{else} \end{cases}$$

- a recursive definition for $k > 1$

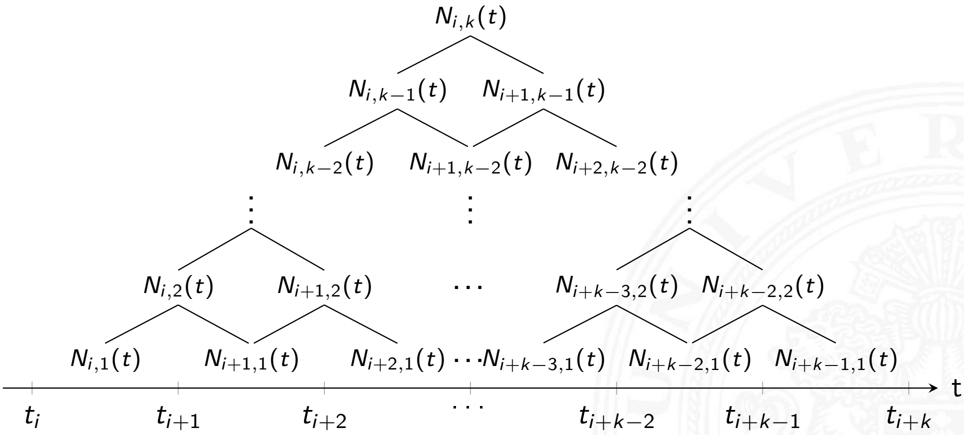
$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

with $i = 0, \ldots, m$.

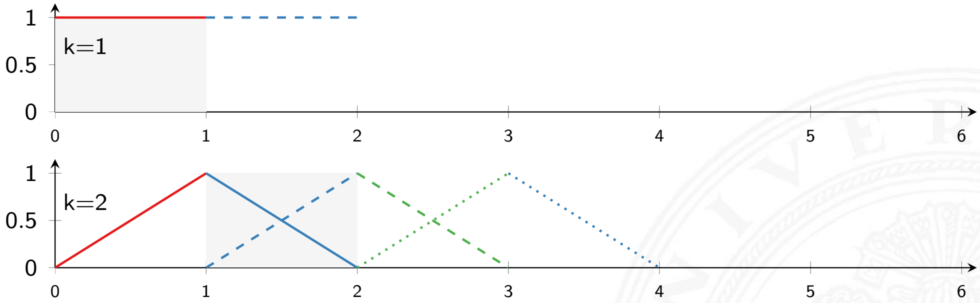
  - the above is referred to as the Cox-de Boor recursion formula

Recursive definition of a B-spline

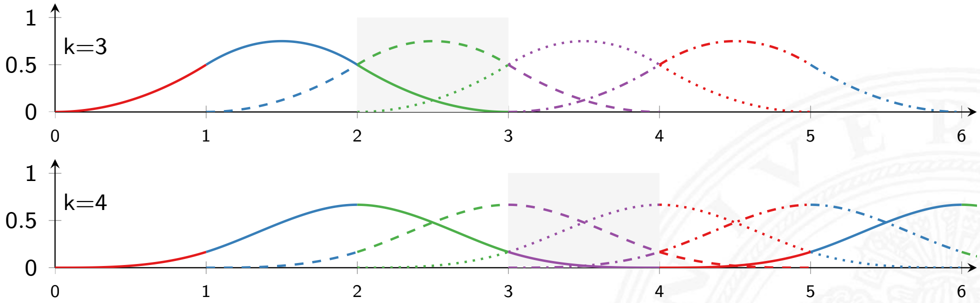
The recursive definition of a B-spline basis function $N_{i,k}(t)$:



Examples of B-splines

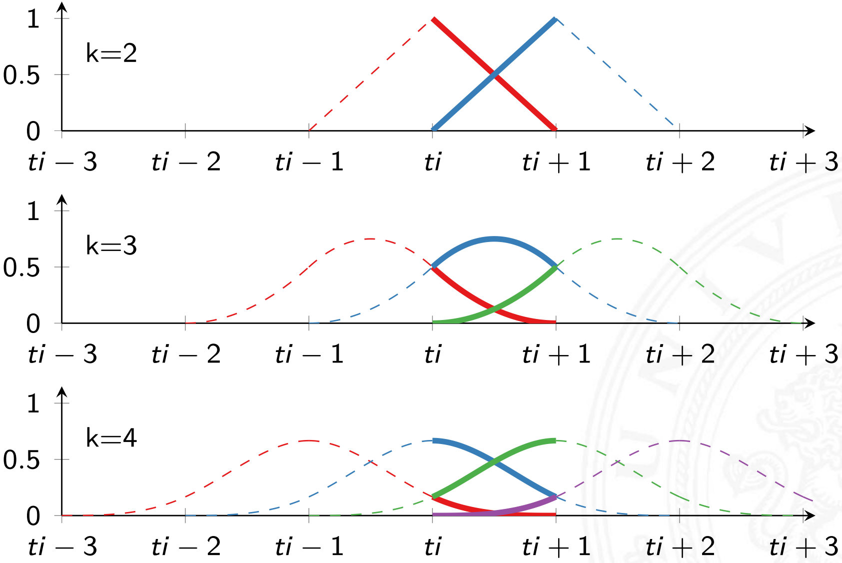


Examples of B-splines



Overlapping

There are $k = p + 1$ overlapping B-splines within an interval.

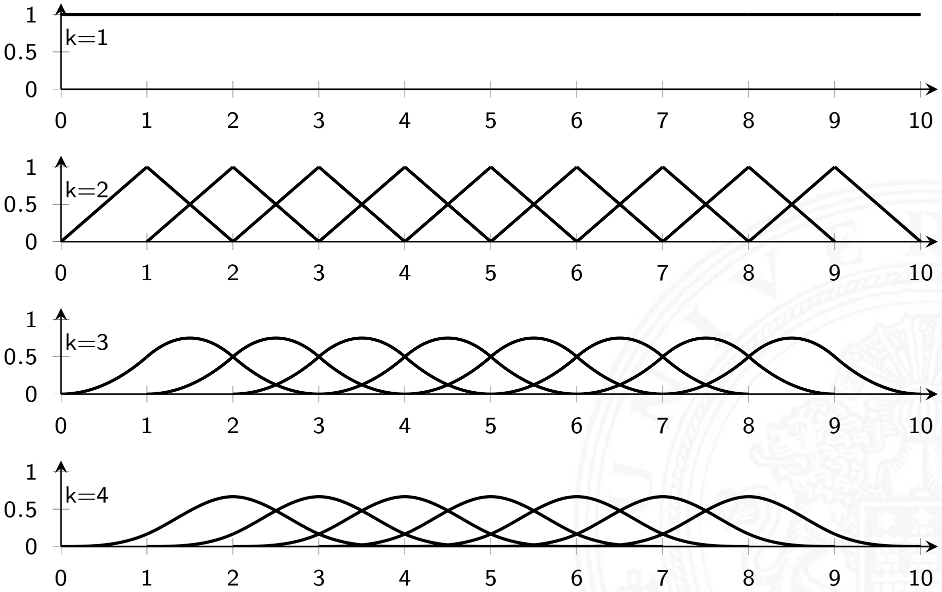


Uniform B-splines

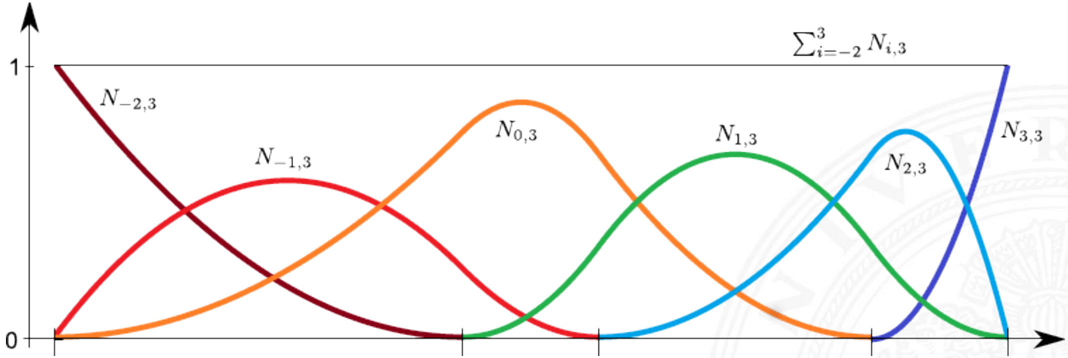
- Distance between uniform B-splines' control points is constant
- Weight-functions of uniform B-splines are periodic
- All functions have the same form
  - Easy to compute

$$B_{m,k} = B_{m+1,k} = B_{m+2,k},$$

Uniform B-splines of order 1 to 4



Non-uniform B-spline of order 3



Properties of B-splines

- Partition of unity: $\sum_{i=0}^{k} N_{i,k}(t) = 1$.
- Positivity: $N_{i,k}(t) \geq 0$.
- Local support: $N_{i,k}(t) = 0$ for $t \notin [t_i, t_{i+k}]$.
- C^{k-2} continuity:
  If the knots $\{t_i\}$ are pairwise different from each other, then

$$N_{i,k}(t) \in C^{k-2}$$

  i.e. $N_{i,k}(t)$ is $(k-2)$ times continuously differentiable.

Construction of a B-spline curve

A B-spline curve can be composed by combining pre-defined control-points with B-spline basis functions:

$$\mathbf{r}(t) = \sum_{j=0}^{m} \mathbf{v}_j \cdot N_{j,k}(t)$$

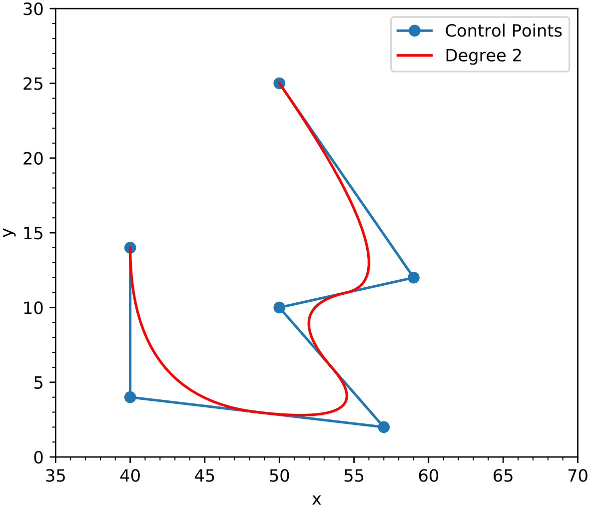
where t is the time, $\mathbf{r}(t)$ is a point on this B-spline curve and $\mathbf{v}_j$ are called its `control points` (`de-Boor points`).

$\mathbf{r}(t)$ is a C^{k-2} continuous curve if the range of t is $[t_{k-1}, t_{m+1}]$.

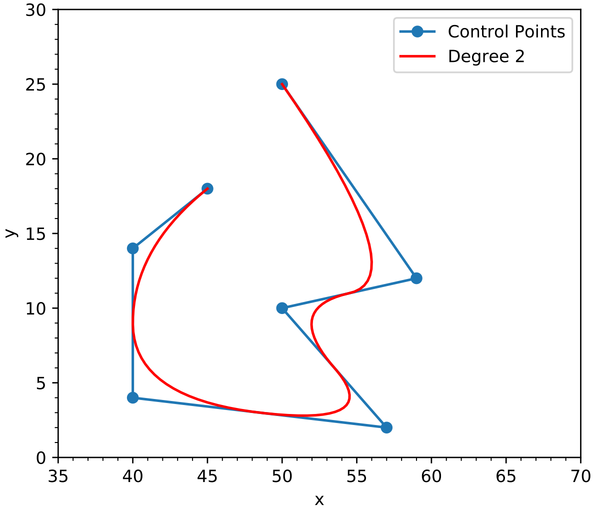
B-Spline curves

- A series of de-Boor points forms a convex hull for the interpolating curve
- Path always constrained to de-Boor point's convex hull
- De-Boor points are of same dimensionality as B-spline curve
- B-spline curves have locality properties
  - control point P_i influences the curve only within the interval $[\tau_i, \tau_{i+p}]$

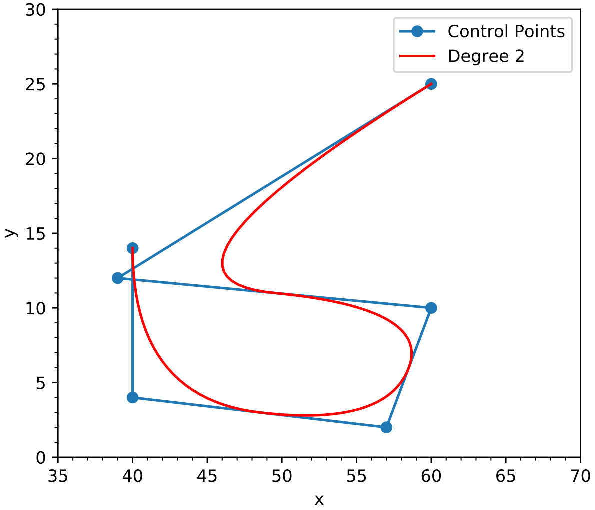
The influence of different control points



The influence of different control points (cont.)



The influence of different control points (cont.)



Generating control points from data points

Question

Given a set of m data points and a degree p, find a B-spline curve of degree p defined by m control points that passes all data points in the given order.

Two methods:

- by solving the following system of equations [9]

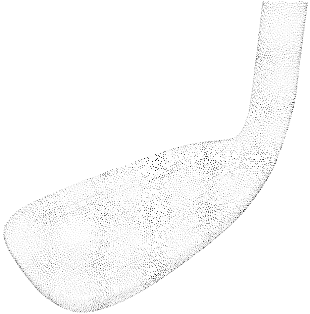
$$\mathbf{q}_j(t) = \sum_{j=0}^{m} \mathbf{v}_j \cdot N_{j,k}(t) \Longrightarrow Q = N \cdot V$$

  where $\mathbf{q}_j$ are the data points to be interpolated, $j = 0, \cdots, m$;
  N is a $m \times m$ matrix;
  V and Q is a $m \times s$ matrices, s is the space dimension.

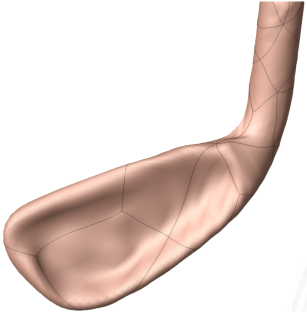
- by learning, based on gradient-descend.[10]

Surface reconstruction with B-Splines

- Surface reconstruction from laser scan data using B-splines [11]



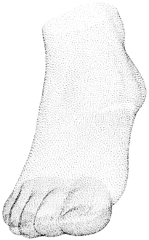
Pointcloud (16,585 points)



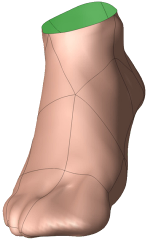
35 patches, 1.36% max. error

285 patches, 0.41% max. error

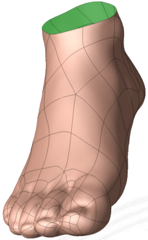
Surface reconstruction with B-Splines (cont.)



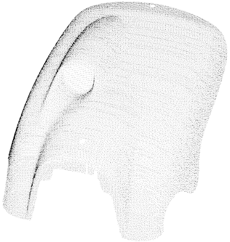
Pointcloud (20,021 points)



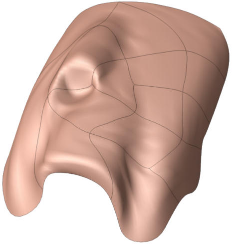
29 patches, 1.20% max. error



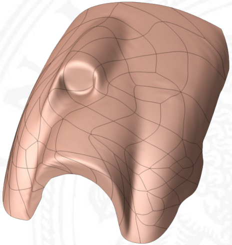
156 patches, 0.27% max. error



Pointcloud (37,974 points)



15 patches, 3.00% max. error

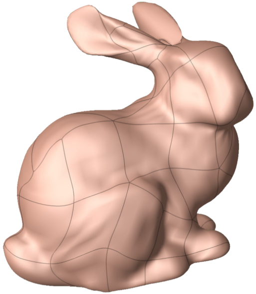


94 patches, 0.69% max. error

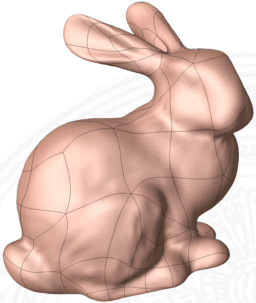
Surface reconstruction with B-Splines (cont.)

- Surface approximation from mesh data (reduced to 30,000 faces)

Mesh (69,473 faces)



72 patches, 4.64% max. error



153 patches, 1.44% max. error

Lagrange interpolation

To match $l+1$ data points (x_i, y_i) $(i = 0, 1, \dots, l)$ with a polynomial of degree l , the following approach of Lagrange can be used:

$$p_l(x) = \sum_{i=0}^{l} y_i L_i(x)$$

The interpolation polynomial in the Lagrange form is defined as follows:

$$L_i(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_l)}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_l)}$$

$$L_i(x_k) = \begin{cases} 1 \text{ if } i = k \\ 0 \text{ if } i \neq k \end{cases}$$

Introduction to Robotics

Lecture 7

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction
Spatial Description and Transformations
Forward Kinematics
Robot Description
Inverse Kinematics for Manipulators
Instantaneous Kinematics
Trajectory Generation 1
Trajectory Generation 2
Dynamics
- Forward and inverse Dynamics
- Dynamics of Manipulators
- Newton-Euler-Equation
- Langrangian Equations
- General dynamic equations

Robot Control

Outline (cont.)

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Dynamics of multibody systems

- A multibody system is a mechanical system of single bodies
  - connected by joints,
  - influenced by forces
- The term `dynamics` describes the behavior of bodies influenced by forces
  - Typical forces: weight, friction, centrifugal, magnetic, spring, ...
- `kinematics` just models the motion of bodies (without considering forces), therefore it can be seen as a subset of dynamics

Forward and inverse Dynamics

We consider a force F and its effect on a body:

$$F = m \cdot a = m \cdot \dot{v}$$

In order to solve this equation, two of the variables need to be known.

Forward Dynamics

If the force F and the mass of the body m is known:

$$a = \dot{v} = \frac{F}{m}$$

Hence the following can be determined:

- velocity (by integration)
- coordinates of single bodies
- forward dynamics
- mechanical stress of bodies

Forward Dynamics (cont.)

Input

τ_i = torque at joint i that effects a trajectory Θ.
$i = 1, \ldots, n$, where n is the number of joints.

Output

Θ_i = joint angle of i
$\dot{\Theta}_i$ = angular velocity of joint i
$\ddot{\Theta}_i$ = angular acceleration of joint i

Inverse Dynamics

If the time curves of the joint angles are known, it can be differentiated twice.

This way,

- internal forces
- and torques

can be obtained for each body and joint.

Problems of highly dynamic motions:

- models are not as complex as the real bodies
- differentiating twice (on sensor data) leads to high inaccuracy

Inverse Dynamics (cont.)

Input

Θ_i = joint angle i

$\dot{\Theta}_i$ = angular velocity of joint i

$\ddot{\Theta}_i$ = angular acceleration of joint i

$i = 1, \ldots, n$, where n is the number of joints.

Output

τ_i = required torque at joint i to produce trajectory Θ.

- **Forward dynamics:**
  - *Input*: joint forces / torques;
  - *Output*: kinematics;
  - *Application*: Simulation of a robot model.
- **Inverse Dynamics**:
  - *Input*: desired trajectory of a manipulator;
  - *Output*: required joint forces / torques;
  - *Application*: model-based control of a robot.

$$\tau(t) \rightarrow \text{direct dynamics} \rightarrow \mathbf{q}(t), (\dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$$
$$\mathbf{q}(t) \rightarrow \text{inverse dynamics} \rightarrow \tau(t)$$

Unlike kinematics, the inverse dynamics is easier to solve than forward dynamics

Dynamics of Manipulators (cont.)

Two methods for calculation:

- Analytical methods
  - based on Lagrangian equations
- Synthetic methods:
  - based on the Newton-Euler equations

Computation time

Complexity of solving the Lagrange-Euler-model is $O(n^4)$ where n is the number of joints.

$n = 6$: 66,271 multiplications and 51,548 additions.

Lagrangian equations

The description of manipulator dynamics is directly based on the relations between the kinetic K and potential energy P of the manipulator joints.

Here:

- constraining forces are not considered
- deep knowledge of mechanics is necessary
- high effort of defining equations
- can be solved by software

Lagrangian Equations

The Lagrangian function L is defined as the difference between kinetic energy K and potential energy P of the system.

$$L(q_i, \dot{q}_i) = K(q_i, \dot{q}_i) - P(q_i)$$

- K: kinetic energy due to linear velocity of the link's center of mass and angular velocity of the link
- P: potential energy stored in the manipulator that is the sum of the potential energy in the individual links

Lagrangian Equations (cont.)

The Lagrangian function L is defined as:

$$L(q_i, \dot{q}_i) = K(q_i, \dot{q}_i) - P(q_i)$$

Theorem

The motion equations of a mechanical system with coordinates $\mathbf{q} \in \Theta^n$ and the Lagrangian function L is defined by:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = F_i, \quad i = 1, \ldots, n$$

where
q_i: the coordinates, where the kinetic and potential energy is defined;
$\dot{q}_i$: the velocity;
F_i: the force or torque, depending on the type of joint (rotational or linear)

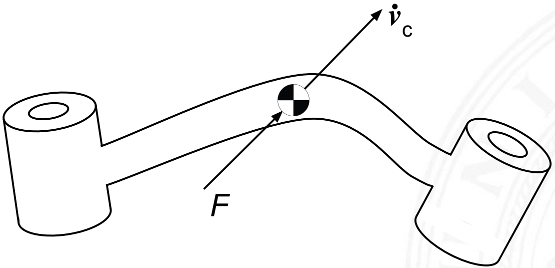
Recursive Newton-Euler Method

- Determine the kinematics from the fixed base to the TCP (relative kinematics)
- The resulting acceleration leads to forces towards rigid bodies
- The combination of constraining forces, payload forces, weight forces and working forces can be defined for every rigid body. All torques and momentums need to be in balance
- Solving this formula leads to the joint forces
- Especially suitable for serial kinematics of manipulator

1. Newton's equation

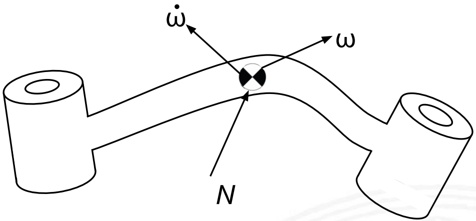
$$F = m\dot{v}_c$$

where F is the force acting at the center of mass of a body, m is the total mass of the body, v_c is the acceleration.



2. Euler's equation

$$\tau = {}^{C}\mathbf{I}\dot{\omega} + \omega \times {}^{C}\mathbf{I}\omega$$



- where ${}^{C}\mathbf{I}$ is the inertia tensor of the body written in a frame C, whose origin is located at the center of the mass.

$${}^{C}I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & Izz \end{bmatrix}$$

- τ is the torque
- $\omega, \dot{\omega}$ are the angular velocity and angular acceleration respectively

Influencing factors to robot dynamics

- Functional affordance
  - trajectory and velocity of links
  - load on a link
- Control quantity
  - velocity and acceleration of joints
  - forces and torques
- Robot-specific elements
  - geometry
  - mass distribution

Aim of determining robot dynamics

- Determining joint forces and torques for one point of a trajectory $(\Theta, \dot{\Theta}, \ddot{\Theta})$
- Determining the motion of a link or the complete manipulator for given joint-forces and -torques (τ)

To achieve this the mathematical model is applied.

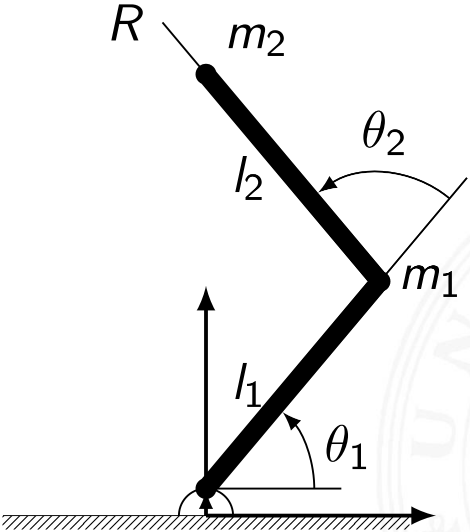
Formulation of robot dynamics

- Combining the different influence factors in the robot specific motion equation from kinematics $(\Theta, \dot{\Theta}, \ddot{\Theta})$
- Practically the Newton-, Euler- and motion-equation for each joint are combined
- Advantages: numerically efficient, applicable for complex geometry, can be modularized

Interim Conclusions

- We can determine the forces with the Newton-equation
- The Euler-equation provides the torque
- **The combination provides force and torque for each joint.**

Dynamics of a multibody system, example: a two joint manipulator.



Using Newton's second law, the forces at the center of mass at link 1 and 2 are:

$$\mathbf{F}_1 = m_1 \ddot{\mathbf{r}}_1$$

$$\mathbf{F}_2 = m_2 \ddot{\mathbf{r}}_2$$

where

$$\mathbf{r}_1 = \frac{1}{2} l_1 (\cos\theta_1 \vec{i} + \sin\theta_1 \vec{j})$$

$$\mathbf{r}_2 = 2\mathbf{r}_1 + \frac{1}{2} l_2 [\cos(\theta_1 + \theta_2)\vec{i} + \sin(\theta_1 + \theta_2)\vec{j}]$$

Newton-Euler-Equations for 2 DOF manipulator (cont.)

Euler equations:

$$\tau_1 = \mathbf{I}_1 \dot{\omega}_1 + \omega_1 \times \mathbf{I}_1 \omega_1$$

$$\tau_2 = \mathbf{I}_2 \dot{\omega}_2 + \omega_2 \times \mathbf{I}_2 \omega_2$$

where

$$\mathbf{I}_1 = \frac{m_1 {l_1}^2}{12} + \frac{m_1 R^2}{4}$$

$$\mathbf{I}_2 = \frac{m_2 {l_2}^2}{12} + \frac{m_2 R^2}{4}$$

Newton-Euler-Equations for 2 DOF manipulator (cont.)

The angular velocities and angular accelerations are:

$$\omega_1 = \dot{\theta}_1$$

$$\omega_2 = \dot{\theta}_1 + \dot{\theta}_2$$

$$\dot{\omega}_1 = \ddot{\theta}_1$$

$$\dot{\omega}_2 = \ddot{\theta}_1 + \ddot{\theta}_2$$

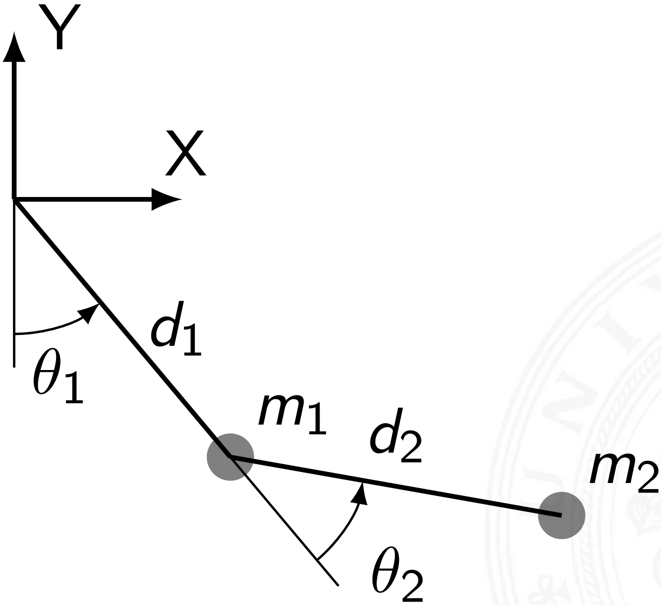
As $\omega_i \times \mathbf{I}_i \omega_i = 0$, the torques at the center of mass of links 1 and 2 are:

$$\tau_1 = \mathbf{I}_1 \ddot{\theta}_1$$

$$\tau_2 = \mathbf{I}_2(\ddot{\theta}_1 + \ddot{\theta}_2)$$

$\mathbf{F}_1, \mathbf{F}_2, \tau_1, \tau_2$ are used for force and torque balance and are solved for joint 1 and 2.

Example: A two joint manipulator



Langragian Method for two joint manipulator

The kinetic energy of mass m_1 is:

$$K_1 = \frac{1}{2} m_1 \ d_1{}^2 \ \dot{\theta_1}{}^2$$

The potential energy is:

$$P_1 = -m_1 \ g \ d_1 \ cos(\theta_1)$$

The cartesian positions are:

$$x_2 = d_1 sin(\theta_1) + d_2 sin(\theta_1 + \theta_2)$$
$$y_2 = -d_1 cos(\theta_1) - d_2 cos(\theta_1 + \theta_2)$$

The cartesian components of velocity are:

$$\dot{x}_2 = d_1 cos(\theta_1)\dot{\theta}_1 + d_2 cos(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2)$$

$$\dot{y}_2 = d_1 sin(\theta_1)\dot{\theta}_1 + d_2 sin(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2)$$

The square of velocity is:

$${v_2}^2 = {\dot{x}_2}^2 + {\dot{y}_2}^2$$

The kinetic energy of link 2 is:

$$K_2 = \frac{1}{2} m_2 {v_2}^2$$

The potential energy of link 2 is:

$$P_2 = -m_2 g d_1 cos(\theta_1) - m_2 g d_2 cos(\theta_1 + \theta_2)$$

Langragian Method for two joint manipulator (cont.)

The Lagrangian function is:

$$L = (K_1 + K_2) - (P_1 + P_2)$$

The force/torque to joint 1 and 2 are:

$$\tau_1 = \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_1} - \frac{\partial L}{\partial \theta_1}$$

$$\tau_2 = \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2}$$

Langragian Method for two joint manipulator (cont.)

τ_1 and τ_2 are expressed as follows:

$$\begin{aligned}\tau_1 =& D_{11}\ddot{\theta}_1 + D_{12}\ddot{\theta}_2 + D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 \\ &+ D_{112}\dot{\theta}_1\dot{\theta}_2 + D_{121}\dot{\theta}_2\dot{\theta}_1 + D_1\end{aligned}$$

$$\begin{aligned}\tau_2 =& D_{21}\ddot{\theta}_1 + D_{22}\ddot{\theta}_2 + D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 \\ &+ D_{212}\dot{\theta}_1\dot{\theta}_2 + D_{221}\dot{\theta}_2\dot{\theta}_1 + D_2\end{aligned}$$

where

- D_{ii}: the inertia to joint i;
- D_{ij}: the coupling of inertia between joint i and j;
- D_{ijj}: the coefficients of the centripetal force to joint i because of the velocity of joint j;
- $D_{iik}(D_{iki})$: the coefficients of the Coriolis force to joint i effected by the velocities of joint i and k;
- D_i: the gravity of joint i.

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

$M(\Theta)$: the position dependent $n \times n$-mass matrix of a manipulator
For the example given above:

$$M(\Theta) = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$$

$V(\Theta, \dot{\Theta})$: an $n \times 1$-vector of centripetal and coriolis coefficients
For the example given above:

$$V(\Theta, \dot{\Theta}) = \begin{bmatrix} D_{111}\dot{\theta}_1^2 + D_{122}\dot{\theta}_2^2 + D_{112}\dot{\theta}_1\dot{\theta}_2 + D_{121}\dot{\theta}_2\dot{\theta}_1 \\ D_{211}\dot{\theta}_1^2 + D_{222}\dot{\theta}_2^2 + D_{212}\dot{\theta}_1\dot{\theta}_2 + D_{221}\dot{\theta}_2\dot{\theta}_1 \end{bmatrix}$$

General dynamic equations of a manipulator (cont.)

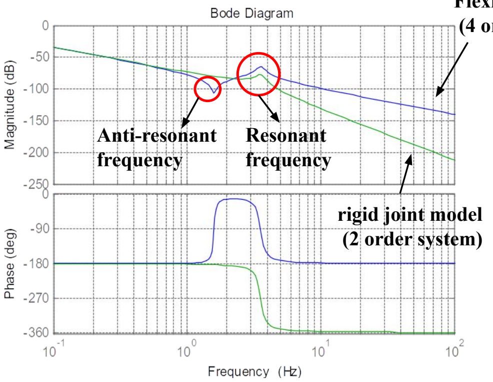
- a term such as $D_{111}\dot{\theta}_1^2$ is caused by coriolis force;
- a term such as $D_{112}\dot{\theta}_1\dot{\theta}_2$ is caused by coriolis force and depends on the (math.) product of the two velocities.
- $G(\Theta)$: a term of velocity, depends on Θ.
  - for the example given above

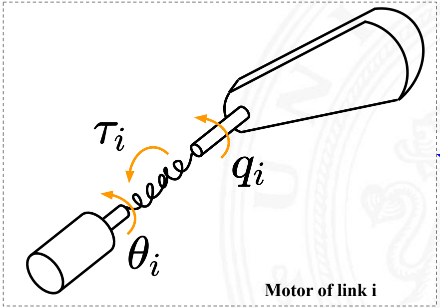
$$G(\Theta) = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}$$

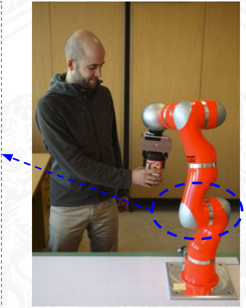
Robot dynamics with flexible joint model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + DK^{-1}\dot{\tau} + \tau_{ext}$$
$$B\ddot{\theta} + \tau + DK^{-1}\dot{\tau} = \tau_m - \tau_f$$
$$\tau = K(\theta - q)$$

- flexible joint as a two-mass model



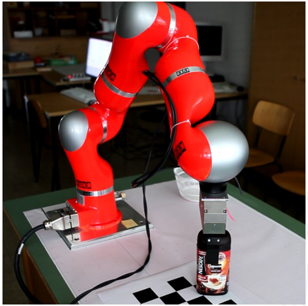




Applications of robot dynamics

KUKA LWR's model-based control

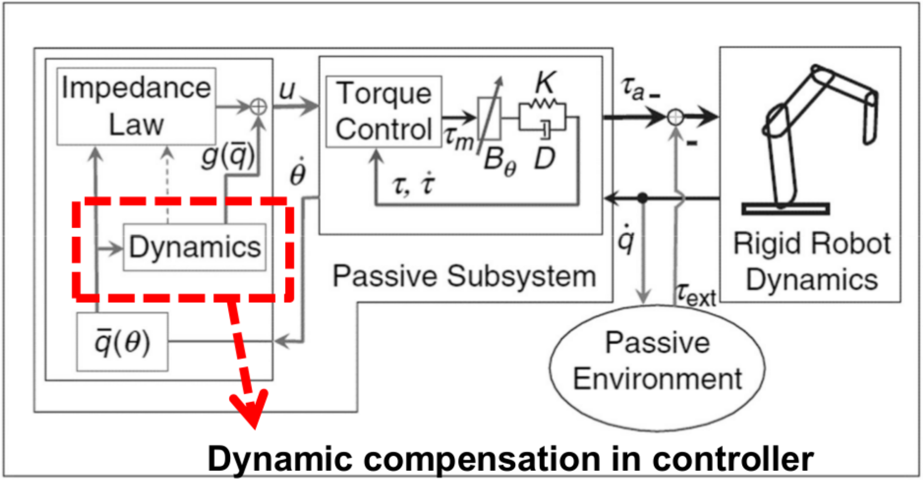
- shortening the motion time without generating overshoots
- giving large reduction of the tracking error



Applications of robot dynamics (cont.)

KUKA iiwa's hand teaching

- Free movement by hand with dynamics compensation on each joint



Introduction to Robotics

Lecture 8

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

- Introduction
- Internal Sensors of Robots
- PID controller
- Classification of Robot Arm Controllers

Outline (cont.)

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Dynamics – Recapitulation

- **Forward dynamics:**
  - *Input*: joint forces / torques;
  - *Output*: kinematics;
  - *Application*: Simulation of a robot model.
- **Inverse Dynamics**:
  - *Input*: desired trajectory of a manipulator;
  - *Output*: required joint forces / torques;
  - *Application*: model-based control of a robot.

$$\tau(t) \rightarrow \text{direct dynamics} \rightarrow \mathbf{q}(t), (\dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))$$
$$\mathbf{q}(t) \rightarrow \text{inverse dynamics} \rightarrow \tau(t)$$

Dynamics – Recapitulation

General inverse dynamic equations of a manipulator:

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

Forward dynamic equations of a manipulator:

$$\ddot{\Theta} = M^{-1}(\Theta)(\tau - V(\Theta, \dot{\Theta}) - G(\Theta))$$
$$\dot{\Theta} = \int \ddot{\Theta} dt$$
$$\Theta = \int \dot{\Theta} dt$$

Unlike kinematics, the inverse dynamics is easier to solve than forward dynamics.

Dynamics – Recapitulation

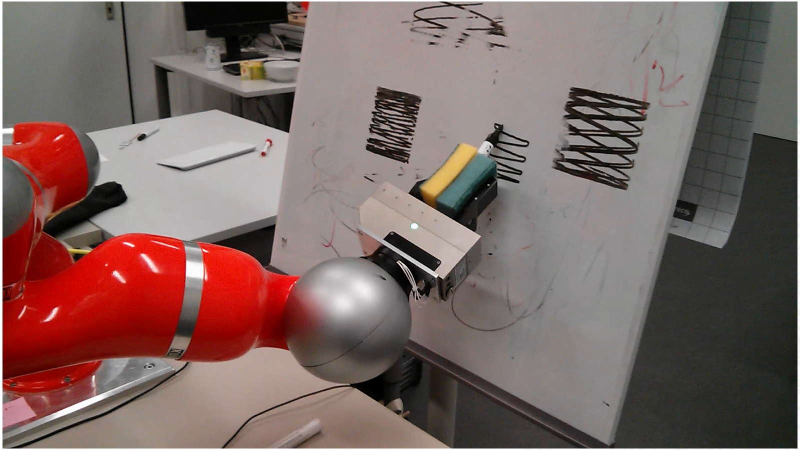
Two methods for calculation:

- Analytical methods
  - based on Lagrangian equations

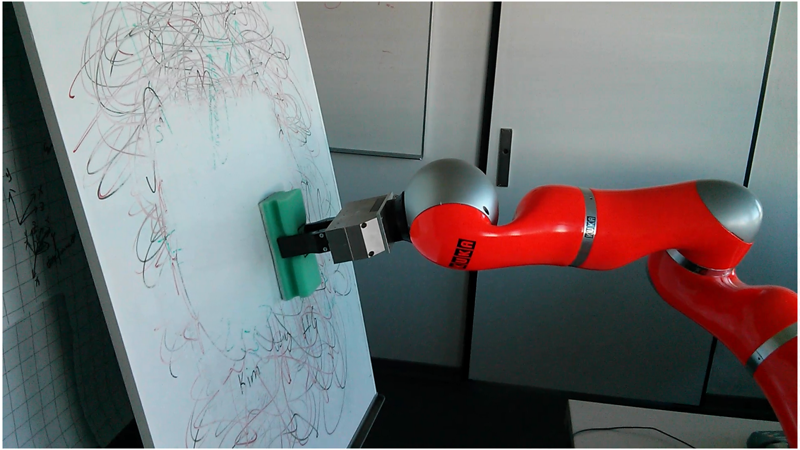
$$L(q_i, \dot{q}_i) = K(q_i, \dot{q}_i) - P(q_i)$$

- Synthetic methods:
  - based on the Newton-Euler equations

Drawing task



Wiping task



Definition

Controller

- Influences one or more physical variables
  - meet a control variable
  - reduce disturbances
- Compares actual value to reference value
  - minimize control deviation

Development of Control Engineering - Timeline

- 1788 J. Watt: engine speed governor
- 1877 J. Routh: differential equation for the description of control processes
- 1885 A. Hurwitz: stability studies
- 1932 A. Nyquist: frequency response analysis
- 1940 W. Oppelt: frequency response analysis, Control Engineering becomes an independent discipline
- 1945 H. Bode: discipline new methods for frequency response analysis
- 1950 N. Wiener: statistical methods
- 1956 L. Pontrjagin: optimal control theory, maximum principle
- 1957 R. Bellmann: dynamic programming
- 1960 direct digital control
- 1965 L. Zadeh: Fuzzy-Logic
- 1972 Microcomputer use
- 1975 Control systems for automation
- 1980 Digital device technology
- 1985 Fuzzy-controller for industrial use
- 1995 Artificial neuronal networks for industrial use

Control Problem

Given: dynamic system (to be controlled)

- Model describing dynamic system (e.g. Jacobian)
- Input variables – control variables
  - measured values (sensor data)
- Output variables – controlled variables
  - system input (force/torque data)

Problem

- Keep control variable values constant and / or
- Follow a reference value and / or
- Minimize the influence of disturbances

Control Problem (cont.)

Sought: controller (for dynamic system)

- Implement hardware or software controller
- Alter controlled-variables (output)
- Based on control variables (input)
- Solve the problem

Example: Cruise Control

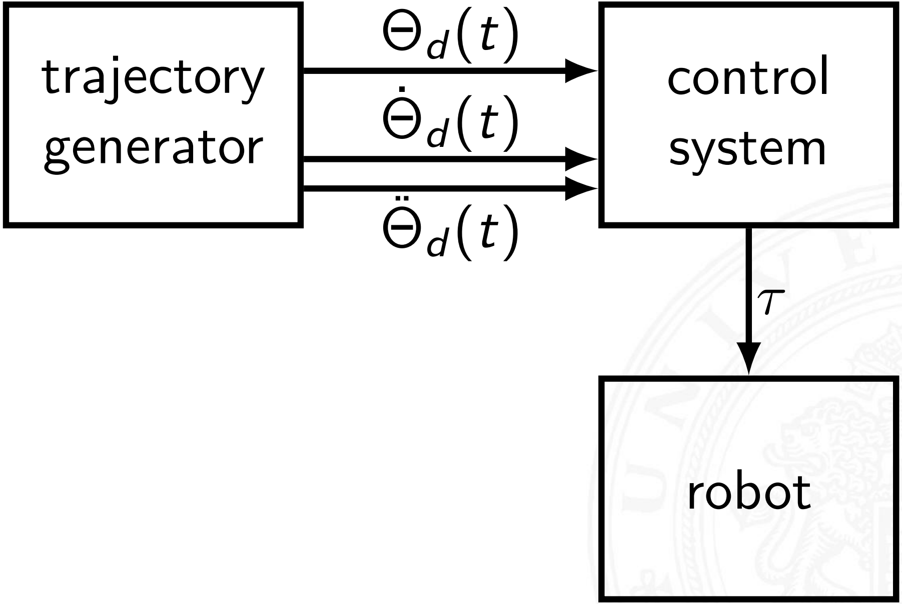
Input

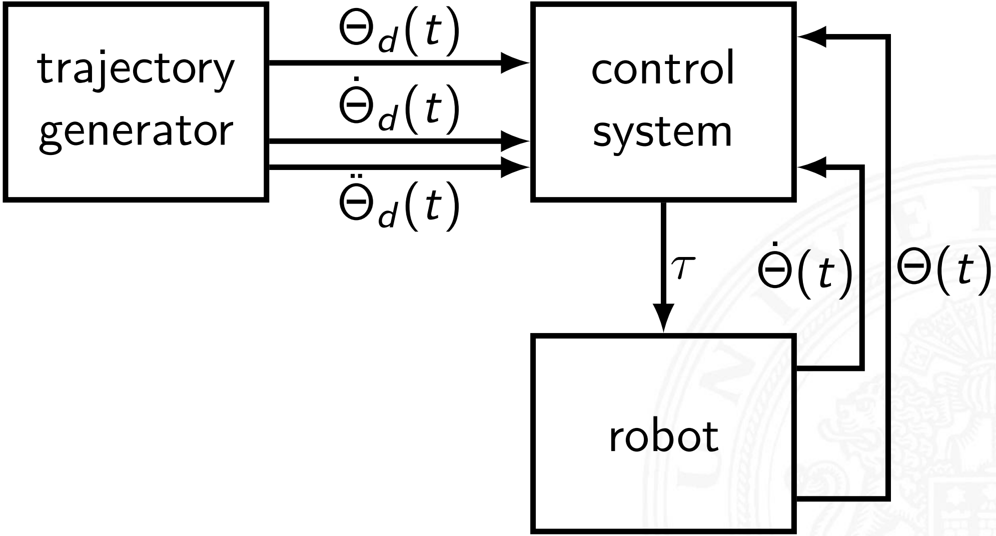
- Speed over ground
- Relative speed to traffic
- Distance to car in front
- Distance to car behind
- Weather conditions
- Relative position in road lane
- ...

Output

- Throttle
- Brakes
- Steering

Control System of a Robot

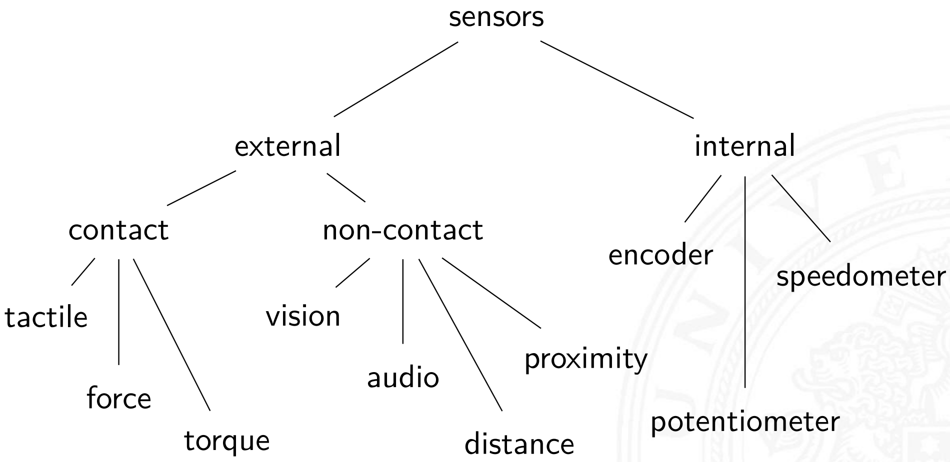



trajectory generator
$\Theta_d(t)$
$\dot{\Theta}_d(t)$
$\ddot{\Theta}_d(t)$
control system
τ
$\dot{\Theta}(t)$
$\Theta(t)$
robot

Control System of a Robot (cont.)

- Target values
  - $\Theta_d(t)$
  - $\dot{\Theta}_d(t)$
  - $\ddot{\Theta}_d(t)$
- Magnitude of error
  - $E = \Theta_d - \Theta, \dot{E} = \dot{\Theta}_d - \dot{\Theta}$
- Output (Control) value
  - $\Theta(t)$
  - $\dot{\Theta}(t)$
- Controlled value
  - τ

Sensor Classification Hierarchy



Internal Sensors of Robots

- Placed inside the robot
- Monitor the internal state of the robot
  - e.g. position and velocity of a joint

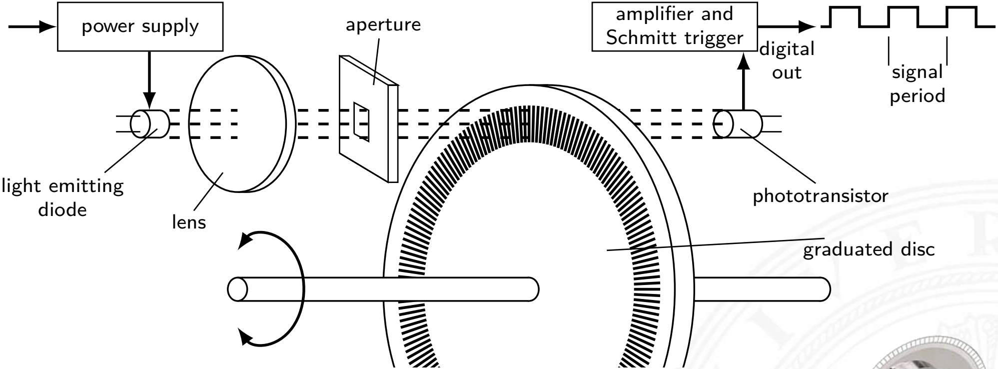
Position measurement systems

- Potentiometer
- Incremental/absolute encoder
- Resolver

Velocity measurement systems

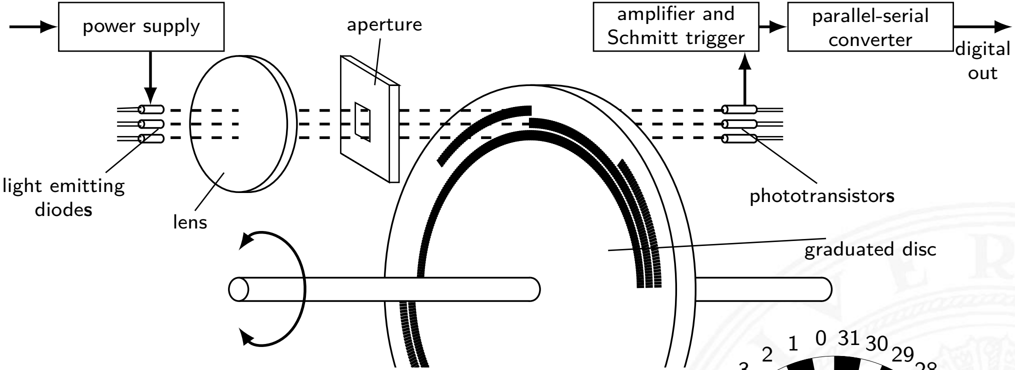
- Speedometers
- Calculate from position change over time

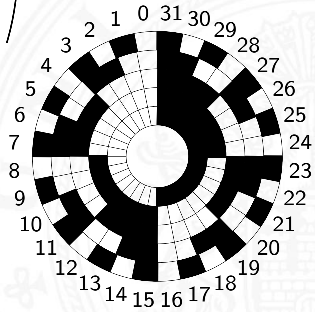
Optical Incremental Encoders



- An optical encoder reads the lines
- The disc is mounted to the shaft of the joint motor
  - PUMA-560: 1:1 ratio; .0001 rad/bit accuracy
- one special line is marked as the "zero-position"

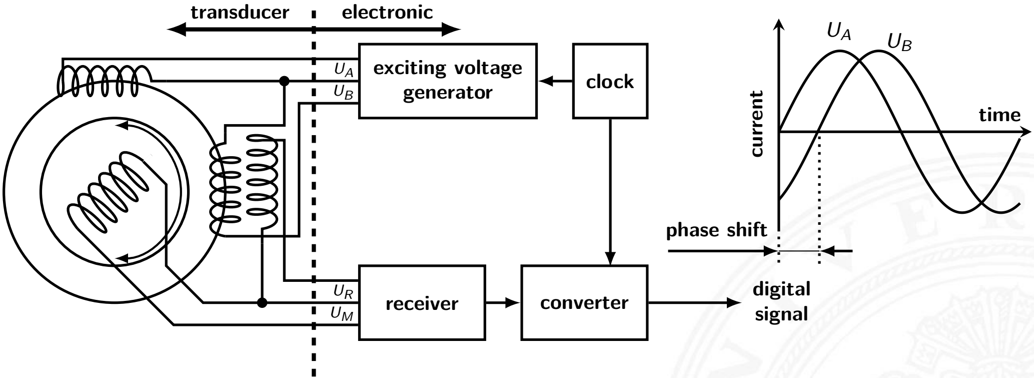
Optical Absolute Encoder





- multiple LEDs and phototransistors
- e.g. 5 bit dual code gives 32 angular positions and 11.25° resolution
- parallel-serial converter required
- absolute positioning and direction encoding

Resolver

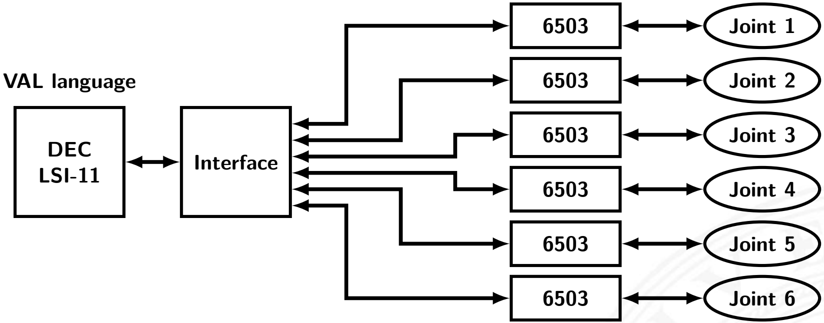


- analog rotation encoding
- phase shift between U_A and U_B determines rotation
- precision depending on digital converter

Encoder vs Resolver

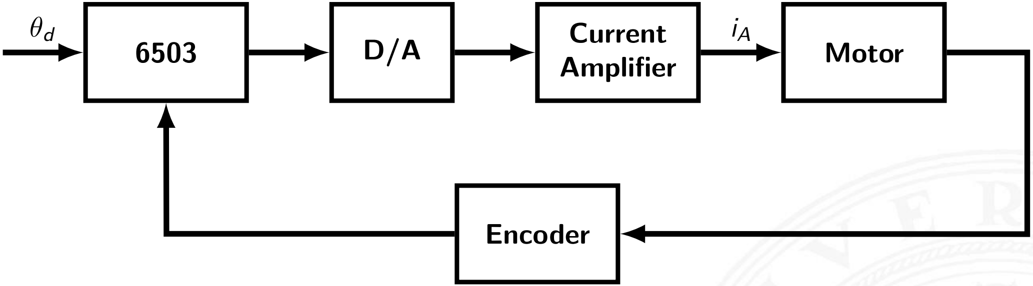
- Encoder:
  - higher accuracy
  - simplicity of integration, and update
  - suitable for applications with high acceleration and deceleration rates
- Resolver:
  - lack of sensitive optics
  - resistant to electrical disturbances
  - complexity of integrating a resolver into a system
  - suitable for extremely harsh applications, such as military and aerospace equipment

Control System Architecture of PUMA-Robot



- two-level hierachical structure of control system
- *DEC LSI-11* sends joint values at 35.7 Hz (28 ms)
  - trajectory
- Distance of actual value to goal value is interpolated
  - using 8,16,**32** or 64 increments

Control System Architecture of PUMA-Robot (cont.)



- The joint control loop operates at 1143 Hz (0.875 ms)
- Encoders are used as position sensors
- No dedicated speedometer
  - velocity is calculated as the difference of joint positions over time

Introduction—PID controller

- more than half of the industrial controllers in use today are PID controllers or modified PID controllers
- many different types of tuning rules have been proposed in the literature
- Using these tuning rules, delicate and fine tuning of PID controllers can be made on-site

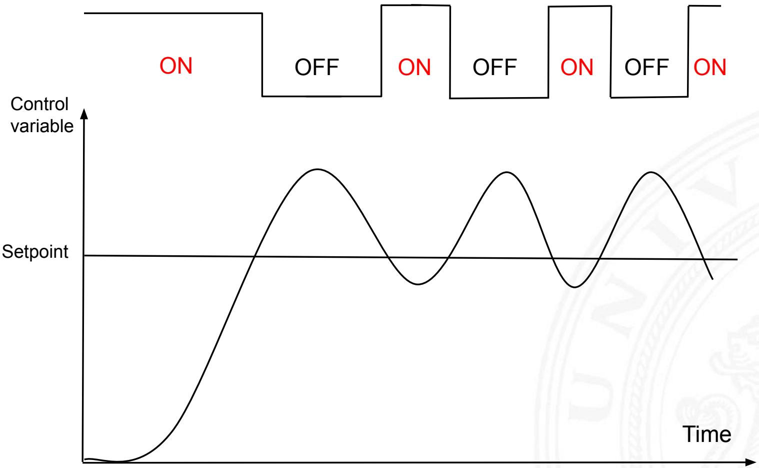
P Proportional controller

I Integral controller

D Derivative controller

Bang Bang (On-off) controller

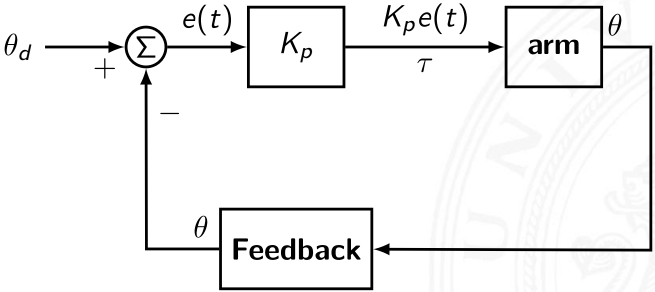
This is the simplest form of control.



Proportional control

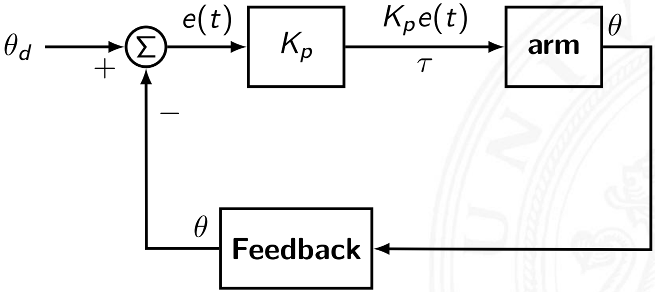
In proportional mode, there is a continuous linear relation between value of the controlled variable and position of the final control element.

- $e(t) = \theta_d - \theta$
- output of proportional controller is $\tau(t) = K_p e(t)$, K_p is proportional gain.



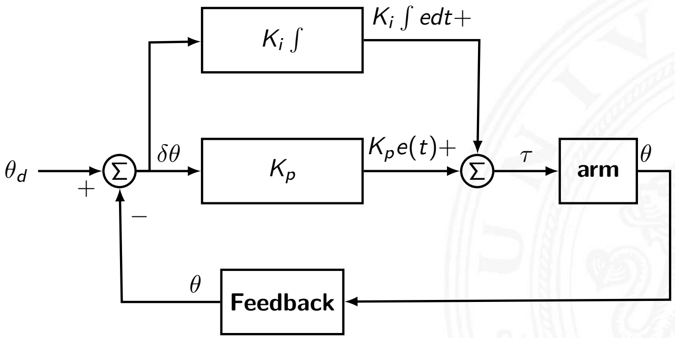
Proportional control (cont.)

- Using P control is simple, but often insufficient:
  - If K_p is small, the sensor reading will approach the setpoint slowly and never reach it
  - As the gain is increased the system responds faster to changes in set-point but becomes progressively underdamped and eventually unstable.
  - If K_p is large, the system may overshoot, oscillate (i.e. become unstable)



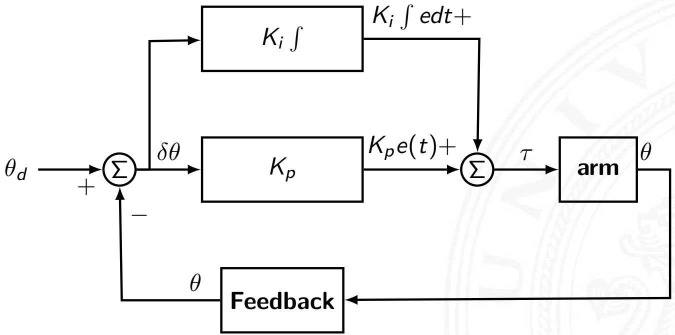
Proportional-Integral (PI) control

- $\tau(t) = K_p e(t) + K_i \int edt$
- The P term will take care of the large movement
- Integral signal is sum of all instantaneous errors
- The I term will take care of any steady-state error



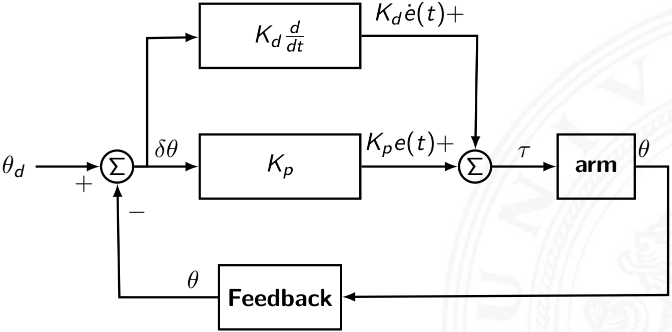
Proportional-Integral (PI) control (cont.)

- It eliminates steady-state error
- It can help with stability of the system, especially if K_p is large
- But, it responds relatively slowly to an error signal



Proportional-Derivative (PD) control

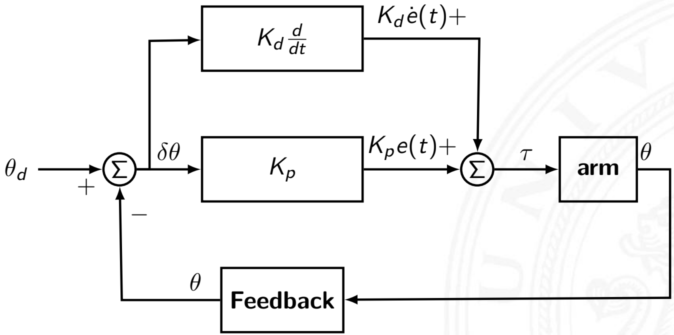
- $\tau(t) = K_p e(t) + K_d \dot{e}(t)$
- Differential term at time $n = K_d(e(n) - e(n-1)/\Delta t)$



Proportional-Derivative (PD) control (cont.)

The main advantages of the PD controllers are:

- The derivative term acts as "brake" to the system
- It can improve the system's tolerance to external disturbances

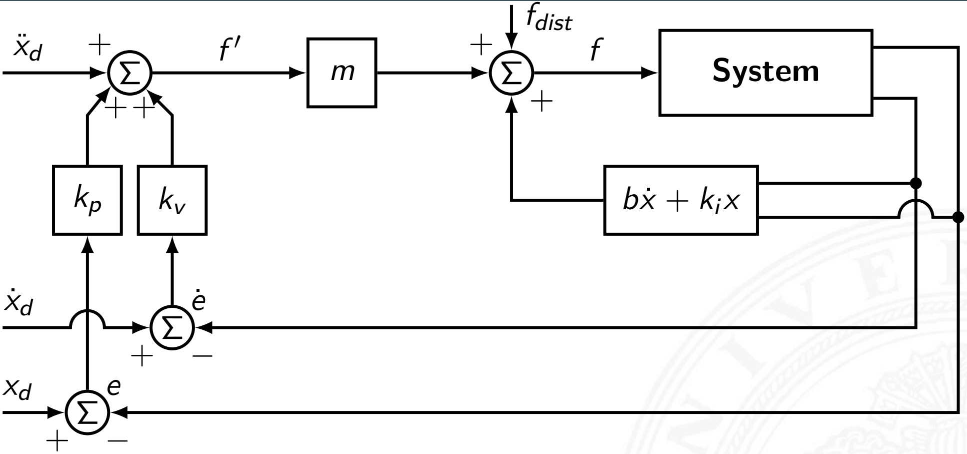


PID-Controller

- P Proportional controller: $\tau(t) = k_p \cdot e(t)$
  The amplification factor k_p defines the sensitivity.
- I Integral controller: $\tau(t) = k_i \cdot \int_{t_0}^{t} e(t')dt'$
  Long term errors will sum up.
- D Derivative controller: $\tau(t) = k_v \cdot \dot{e}(t)$
  This controller is sensitive to changes in the deviation.

Combined $\Rightarrow$ PID-controller:

$$\tau(t) = k_p \cdot e(t) + k_v \cdot \dot{e}(t) + k_i \int_{t_0}^{t} e(t')dt'$$



$$f' = \ddot{x}_d + k_v \dot{e} + k_p e + k_i \int edt \tag{47}$$

is called the principle of PID-control.

Summary: the characteristics of P, I, D controllers

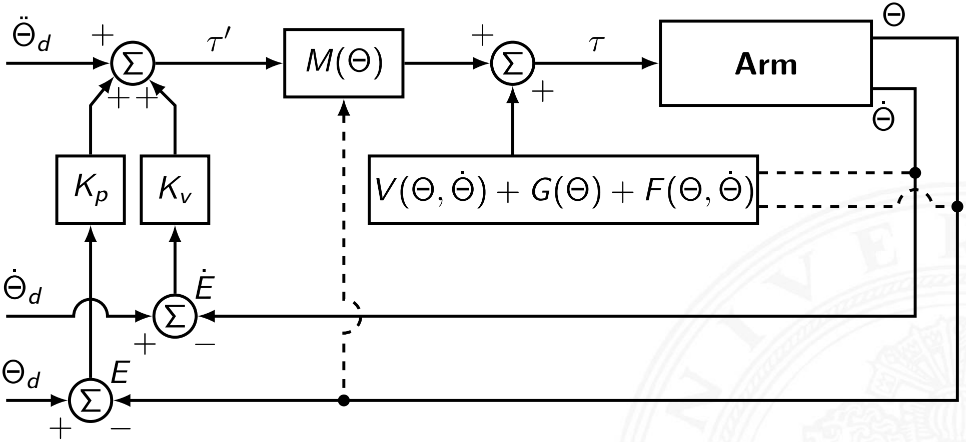
	Rise time	Overshoot	Settling time	S-S error
Kp	decrease	increase	small change	decrease
Ki	decrease	increase	increase	eliminate
Kd	small change	decrease	decrease	small change

Further Resources

- PID Control with Python (simple-pid)
- PID Control with MATLAB and Simulink

Tips for designing a PID controller

1. Obtain an open-loop response and deternine what needs to be improved
2. Add a P control to improve the rise time
3. Add a D control to improve the overshoot
4. Add a I control to eliminate the steady-state error
5. Adjust each of K_p, K_d, K_i until you obtain a desrired overall response



The dynamic equation: $$\tau = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta)$$

where $M(\Theta)$ is the position-dependent $n \times n$-mass matrix of the manipulator, $V(\Theta,\dot{\Theta})$ is a $n \times 1$-vector of centripetal and Coriolis factors, and $G(\Theta)$ is a complex function of Θ, the position of all joints of the manipulator.

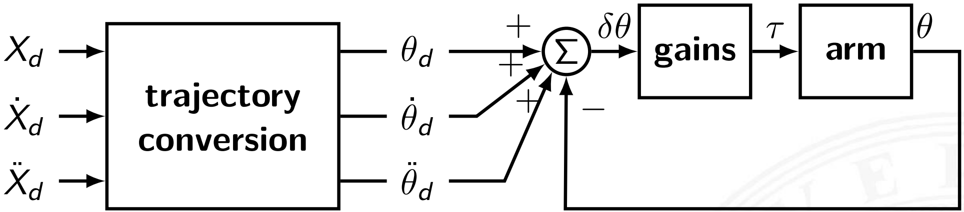
Classification of Robot Arm Controllers

As the problem of trajectory-tracking:

- Joint space: PID, plus model-based
- Cartesian space: joint-based
  - using kinematics or using inverse Jacobian calculation
- Adaptive: model-based adaptive control, self-tuning
  - controller (structure and parameter) adapts to the time-invariant or unknown system-behavior
  - basic control circle is superimposed by an adaptive system
  - process of adaption consists of three phases
    - identification
    - decision-process
    - modification
- Hybrid force and position control is also a popular research topic

Control in Cartesian Space – Method I

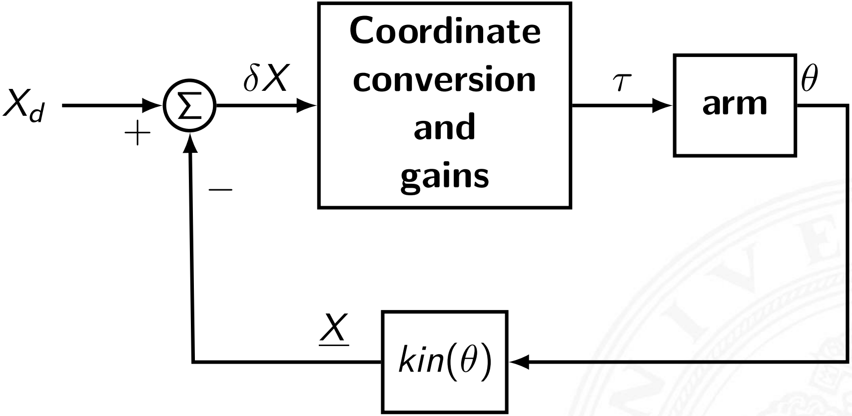
Joint-based control with Cartesian trajectory input



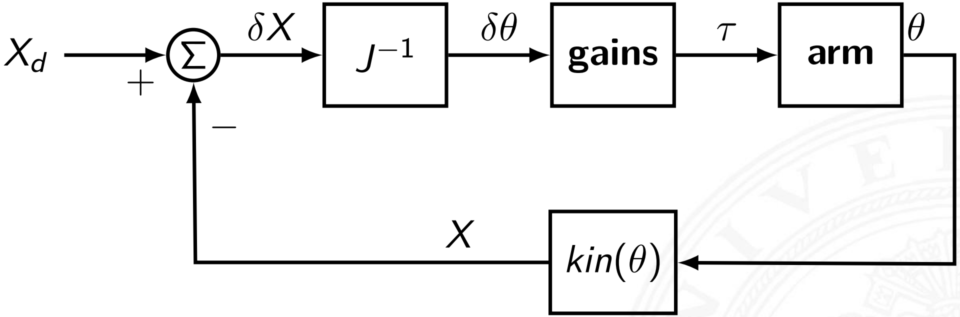
- Cartesian trajectory is converted into joint space first
- joint space trajectory is sent to the controller
- trajectory controller sends joint targets to motor controllers
- motor controller sends torque data to motor
- sensors output joint state

Control in Cartesian Space – Method II

Cartesian control via calculation of kinematics



- controller operates in cartesian space
- joint space conversion within control cycle
- error values in cartesian space using FK



- no explicit joint space conversion
- dynamic conversion using inverse Jacobian

Robot Control Improvements

Scientific Research

- model-based control
- adaptive control
- hybrid control

Industrial robotcs

- PID-control system with gravity compensation

$$\tau = \dot{\Theta}_d + K_v \dot{E} + K_p E + K_i \int E dt + \hat{G}(\Theta)$$

Hybrid Control of Force and Position

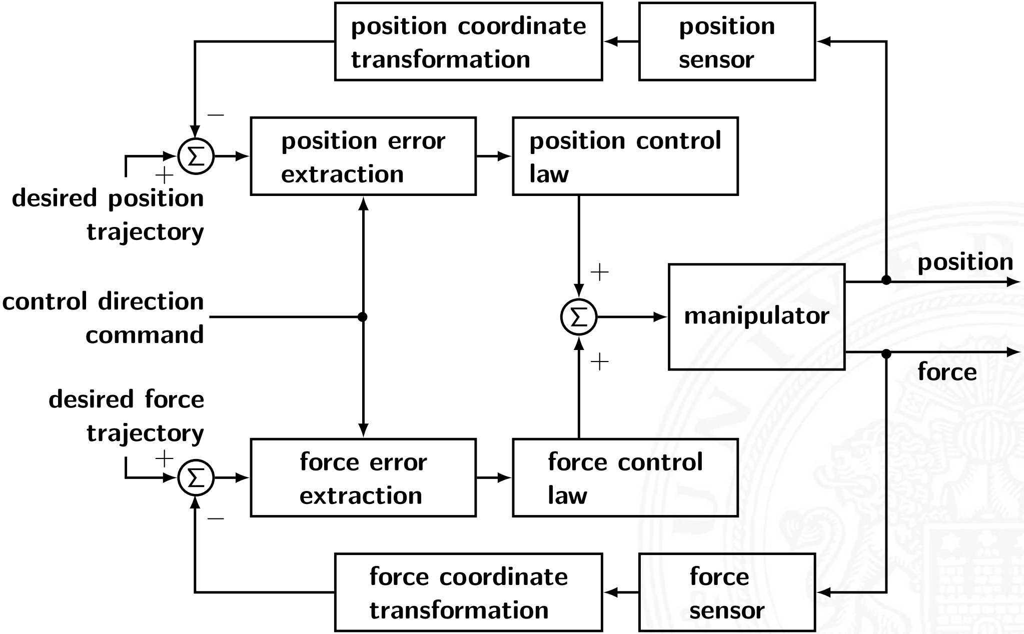
Motivation

Certain tasks require control of both: position and force of the end effector:

- assembly
- grinding
- opening/closing doors
- crank winding
- ...

An example shows two feedback loops for separate control of position and force

Hybrid Control of Force and Position (cont.)



Hybrid Force/Torque Control

Franke Emika Panda

Introduction to Robotics

Lecture 09

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

- Feasible Trajectories
- Geometry Representations
- C-Space

Outline (cont.)

- Planner Approaches
  - Discretized Space Planning
  - Potential Field Method
- Probabilistic Planners
  - Probabilistic Road Maps
  - Rapidly-exploring Random Trees
  - Expansive Space Trees
  - Auxiliary Techniques
- Optimal Planning
  - Planner*

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

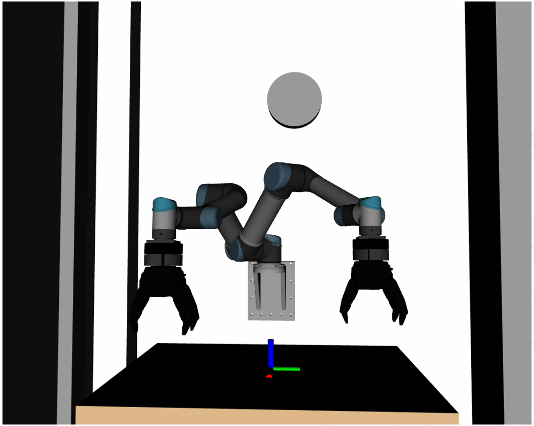
Conclusion and Outlook

From A to B

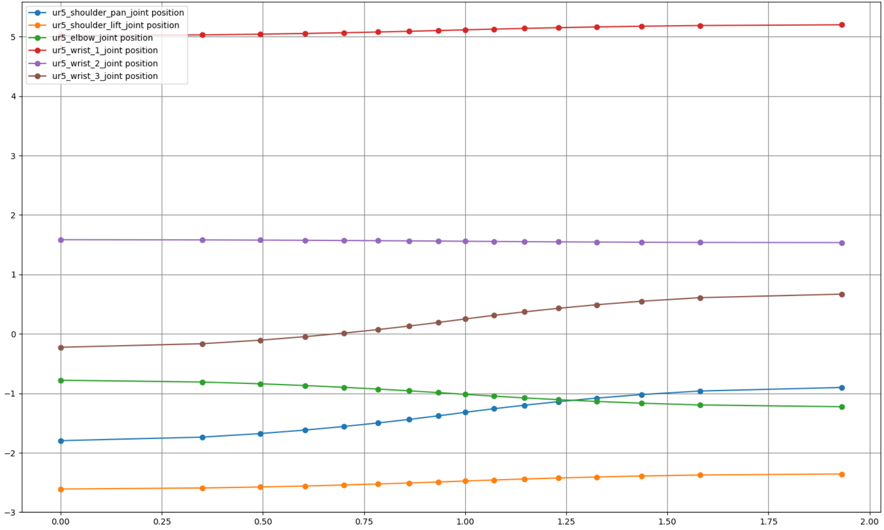
Problem: Generate a continuous trajectory from state A to state B

Approach from previous lectures:
Generate *quintic B-Splines* from A to B:

- Trapezoidal time parameterization
- Minimum jerk parameterization
- Time-optimal motion parameterization

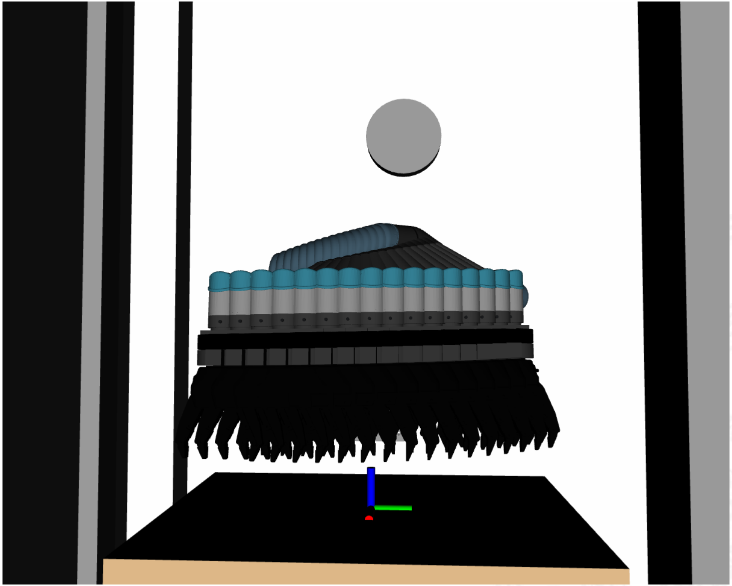


UR5 setup with exemplary start and goal states



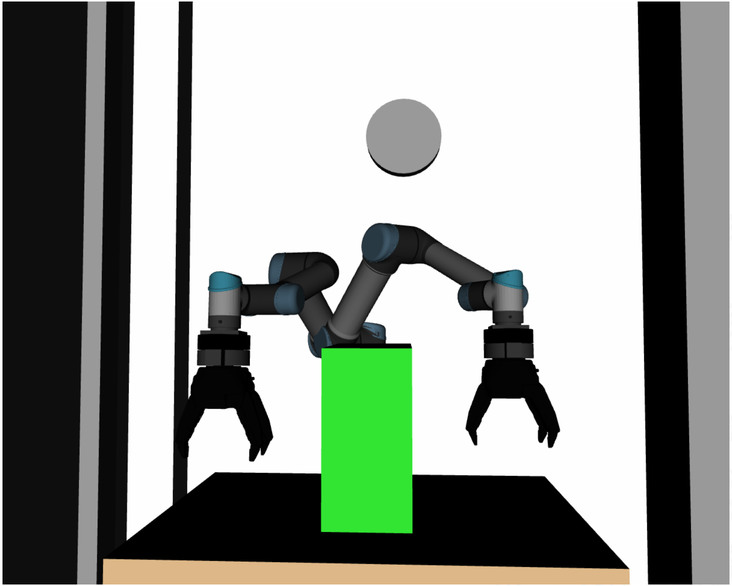
Generated splines of trapezoidal trajectory

From A to B - Trajectory Generation (2)



All waypoints of generated trapezoidal trajectory

From A to B?



Start and Goal state with box obstacle

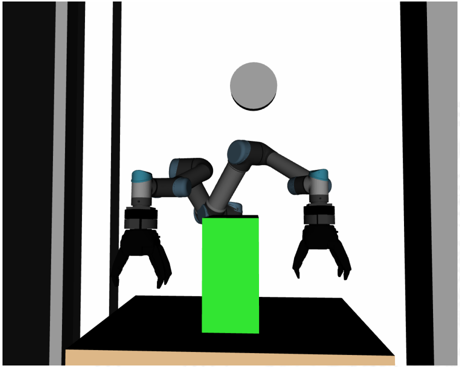
From A to B

If the path is **blocked**, the generated trajectory is **invalid/infeasible** and should not be executed!

Typical obstacles include:

- Walls / Tables
- Robot links
- Objects (to be manipulated)
- Humans

Getting this right is harder than it looks.

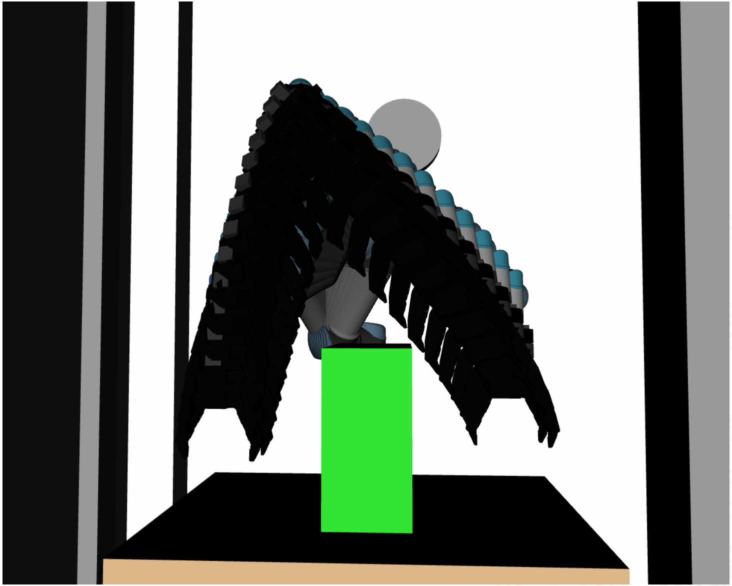


Start and Goal state with box obstacle

Infeasible Trajectories

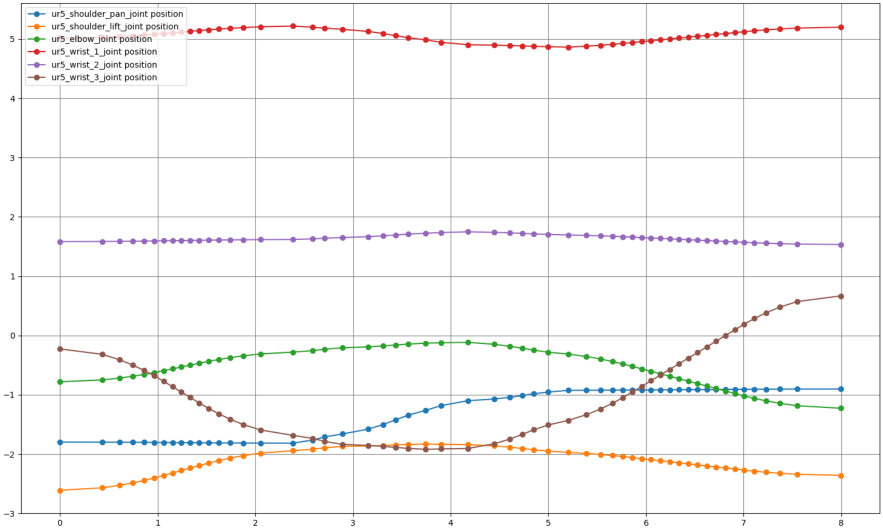
Shadow Hand rammed into styrofoam table

From A to B



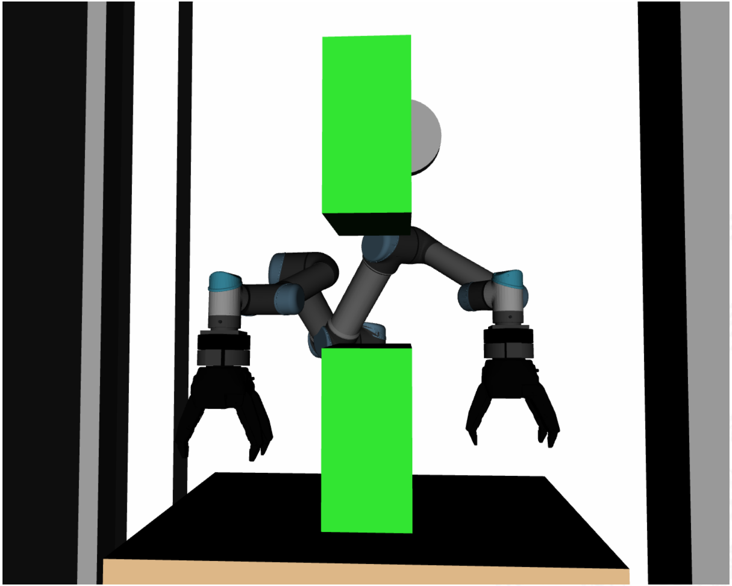
All waypoints of collision-free trajectory

From A to B



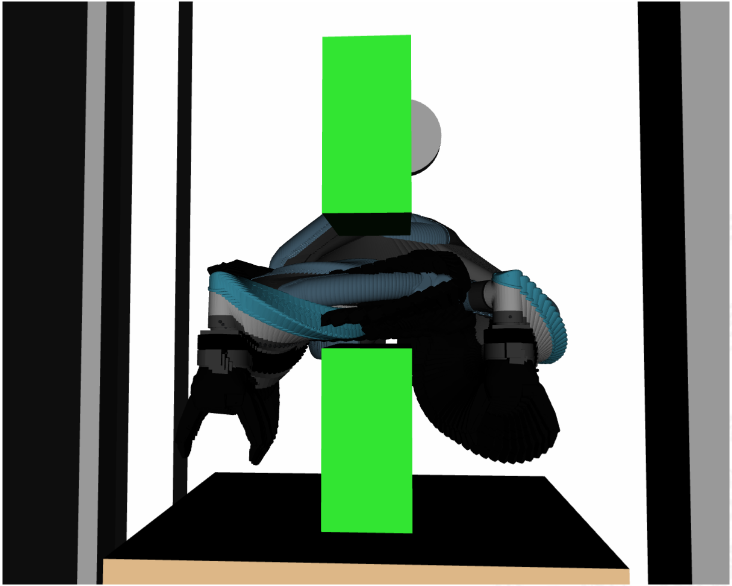
Splines of collision-free trajectory

From A to B



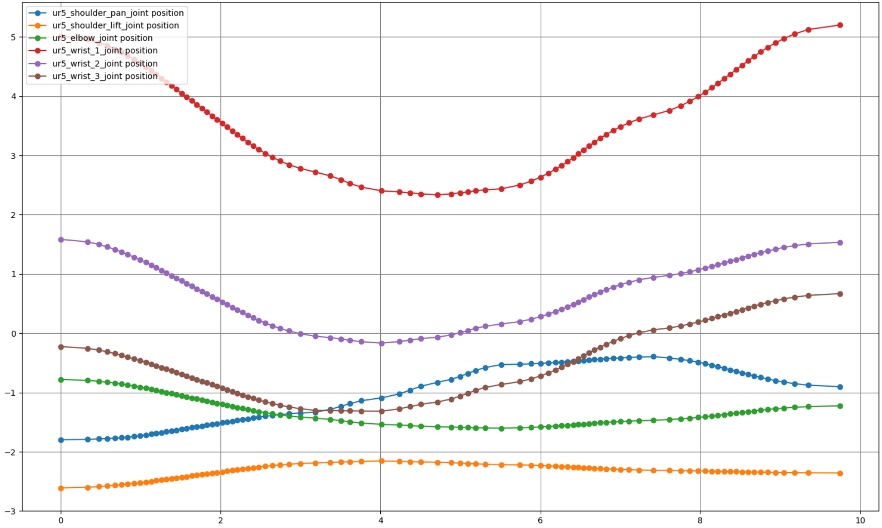
Workspace with two box obstacles

From A to B



All waypoints of collision-free trajectory

From A to B



Splines of collision-free trajectory

Some Criteria for Feasible Trajectories

Feasible trajectories have to satisfy hard geometric constraints.

The most important criterion is a **collision-free** trajectory.

- Collisions between parts of the robot (*self collisions*)
- Collisions with the environment

Countless other criteria can also be important:

- Carrying a container with liquid, no liquid must spill
- Spraying color on a workpiece, the nozzle must always point at the piece
- Getting close or moving directly towards humans

Most of these constraints define *Constraint Manifolds* in the full planning space.
This lecture focuses on collision-aware planning.

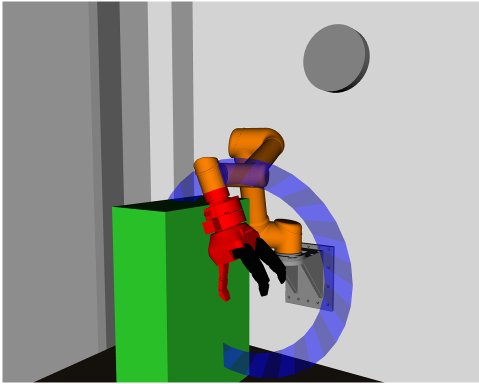
Path Planning

Feasible Trajectories
Geometry Representations
C-Space
Planner Approaches
Probabilistic Planners
Optimal Planning

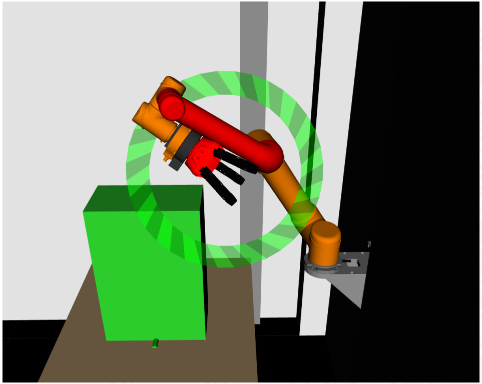
Detecting Collisions

In order to detect expected collisions, we need a geometric **Environment Model**.

- Need to represent all relevant collision shapes
- Trade-off between exact representations and computational load
- Collision tests should run as fast as possible



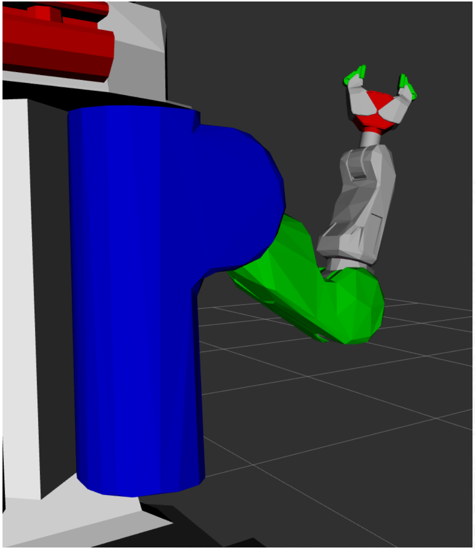
end-effector collision with box



end-effector collision with upper arm link

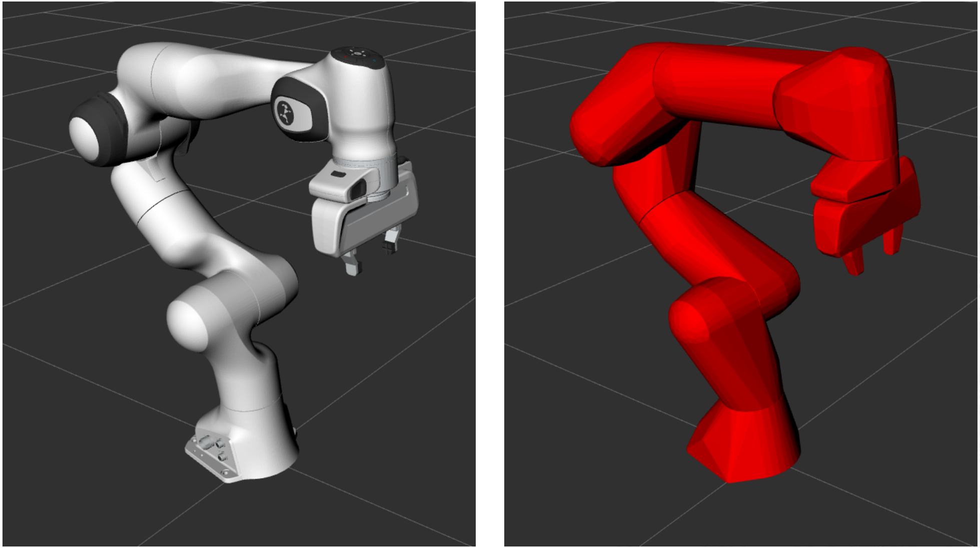
Triangle Meshes

- Standard 3D representation for arbitrary shapes
- General collision checks are costly (Triangle intersection tests)
- Modelled details should depend on required accuracy
- Usually very coarse
- **Convex Meshes** are much more efficient to test. Non-colliding objects can always be separated by a plane.



PR2 left arm mesh representation

Convex Hull Collision Shapes

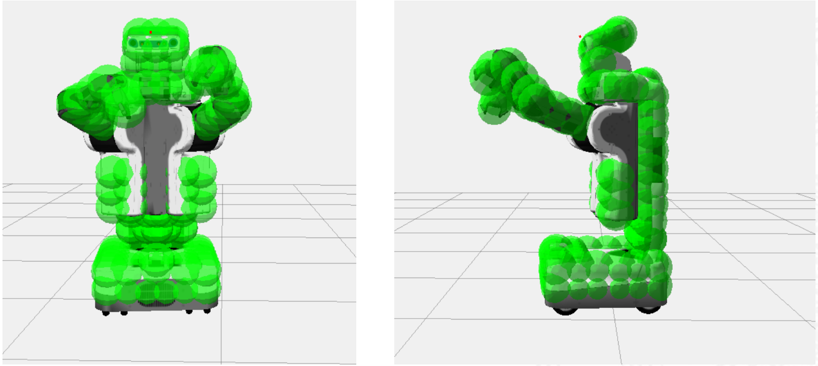


Visual model and convex collision representation of Panda robot arm

Sphere Representations

Parameters: center point c, radius r.

- Sphere/Sphere collisions afford the cheapest check: $\langle c_1, r1\rangle$ and $\langle c_2, r_2\rangle$ collided iff $|c_1 - c_2| < r_1 + r_2$
- Sufficient spheres can approximate any shape reasonably accurate:



Approximation of PR2 robot with 139 spheres with radius 10cm

Cowley 2013 [12]

Geometric Primitives

Primitive analytical shapes can be used for more accurate descriptions:

- **Cube**: pose p, scales for 3 axes
- **Cylinder**: pose p, radius r, height h
- **Cone**: pose p, radius r, height h
- **Plane**: pose p

Many analytical shapes allow for faster collision checks.

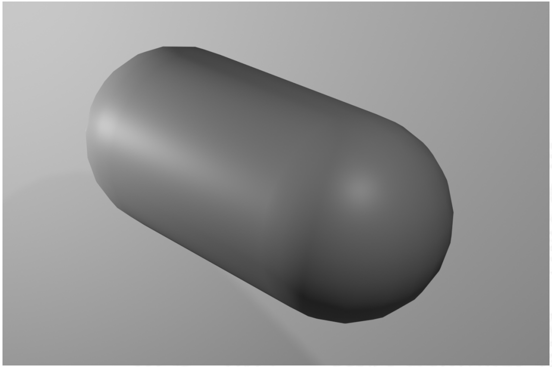
To do (**??**)

Capsules

Capsules comprise two half-spheres and a connecting cylinder.

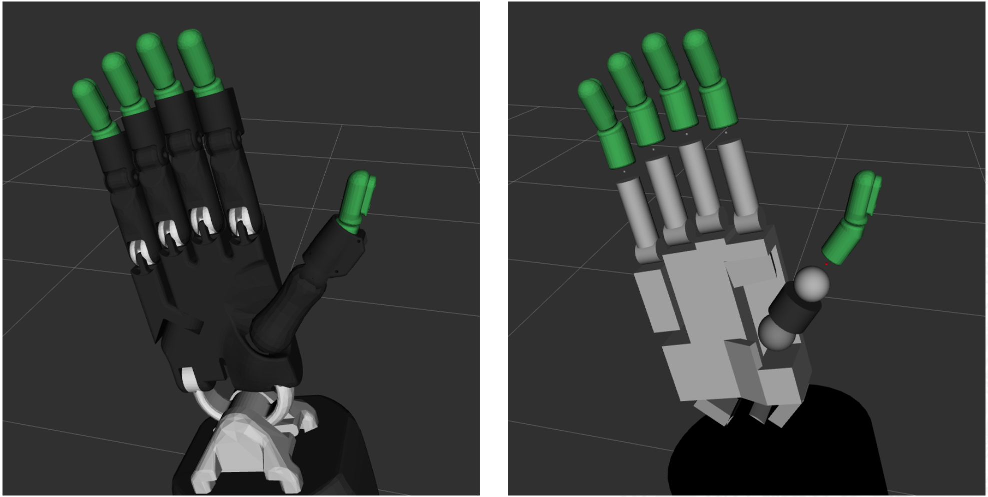
Less common analytical shape, supported in many robotics contexts.

Parameters: pose p, radius r, height h, optionally scale parameters



A primitive capsule

To do (??)



Visual and collision model of a Shadow Dexterous Hand with tactile fingertips

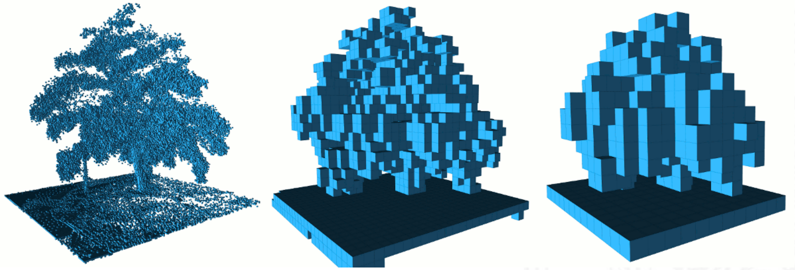
Voxelgrids / Octomaps

All analytical shapes require geometric knowledge about the scene.

Octomaps represent sensor data (depth measurements) directly

- Keeps geometric structure
- Sparse representation
- Efficient updates

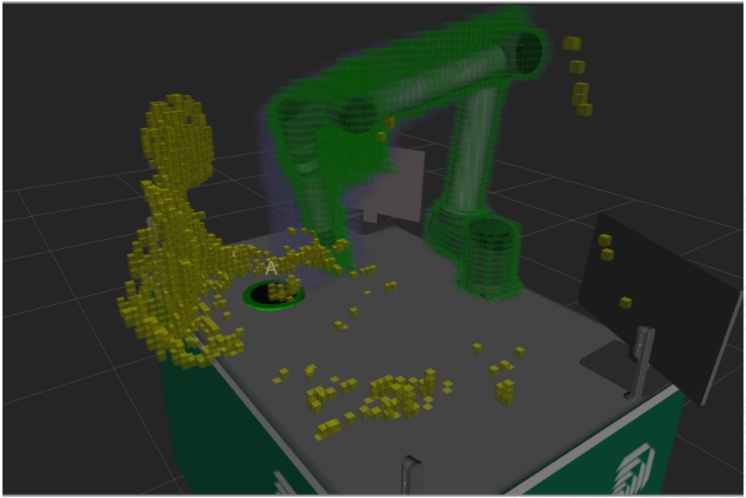
Parameters: pose p, minimal voxel resolution r, datapoints



Octomap representation of a tree at different resolutions

A. Hornung et.al. 2013 [13]

Voxelgrids / Octomaps (2)



Voxel representation of a human interacting with a UR10 robot

Collision Detection Summary

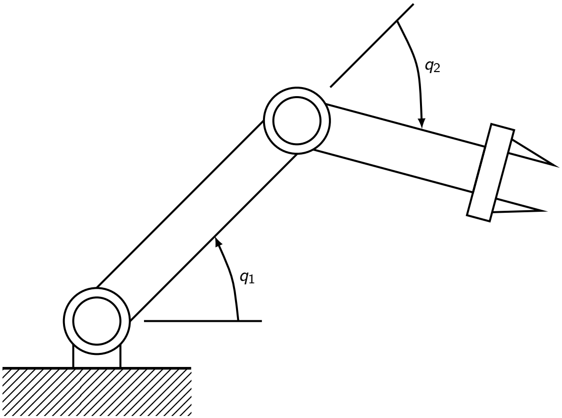
- Hybrid models allow to trade-off computation time and accuracy
- Requires collision checks between each pair of types of collision body

Huge amount of background literature and research in 3D Computer Graphics. Collision checking in full scenes can be optimized much further optimization:

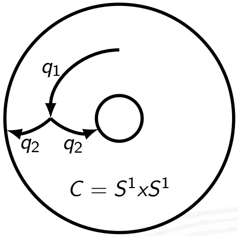
- Broadphase-collision checking
- Convex decompositions
- Hardware-accelerated checking
- ...

Path Planning
Feasible Trajectories
Geometry Representations
C-Space
Planner Approaches
Probabilistic Planners
Optimal Planning

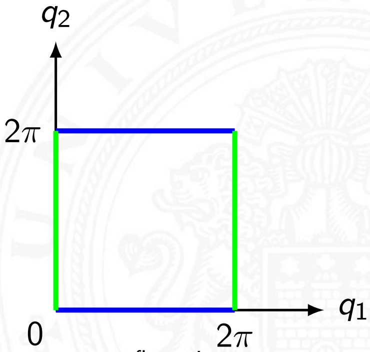
Workspace And Configuration Space – Illustration



2dof robot model



reachable workspace



configuration space

Definition
The parameters that define the configuration of the system are called **Generalized Coordinates**, and the vector space defined by these coordinates is called the **Configuration Space** $\mathcal{X}$.

In robotics, generalized coordinates include

- Joint positions for each controlled joint
- Cartesian poses for mobile robots

$\mathcal{X}_{obs} \subset \mathcal{X}$ describes the set of all configurations in collision.

$\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ describes the collision-free planning space.

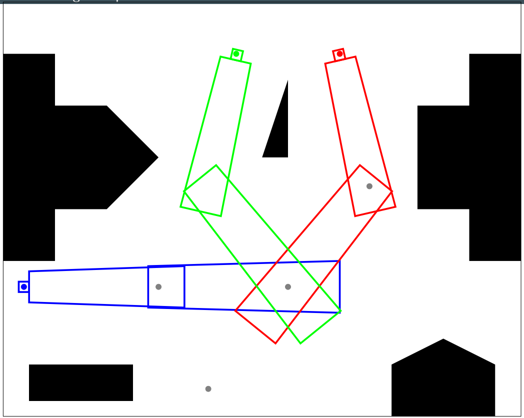
Workspace and Configuration Space

Whereas all intuitive reasoning and system description takes place in the **Workspace**, planning usually proceeds in the **C-space**.

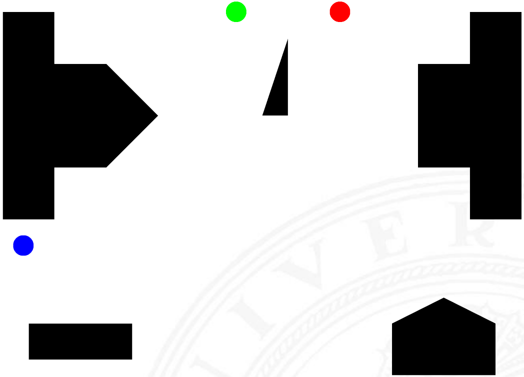
Confusing terminology:

- The workspace is often referred to as reachable **Cartesian space**.
- Configuration space is often shortened to **C-space**.
- For mobile robots, Cartesian poses can be (part of) the C-space.

Workspace to Configuration Space – Example



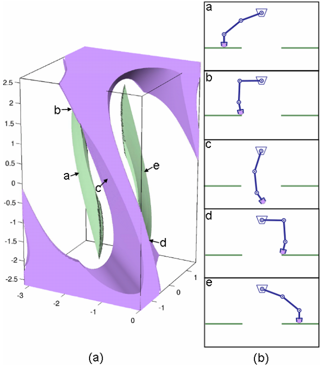
Workspace scheme with multiple states



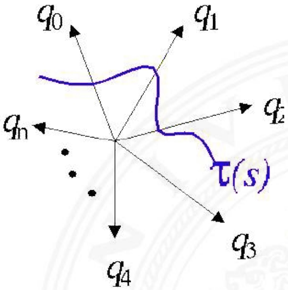
Workspace with target end-effector regions

On Dimensionality

- Workspaces (position-only) are described by 2 or 3 dimensions
- Effective C-spaces have 6 or more dimensions



C-space visualization for simulated 3dof arm



Trajectory in n-dimensional C-space

D. Berenson et.al. 2009 [14]

C-Space Summary

- The parameters of a system, i.e. **Generalized Coordinates**, span a vector space
- This space is called the **C-space** $\mathcal{X}$ of the system

- $\mathcal{X}_{free}$ describes the collision-free subspace of $\mathcal{X}$
- $x \in \mathcal{X}_{free}$ can be tested by collision-checking
- Usually the space is not parameterized (can not be easily described)

- Cartesian space and C-space can coincide in navigation tasks where only the pose of the robot is a parameter

Path Planning

- Feasible Trajectories
- Geometry Representations
- C-Space
- Planner Approaches
  - Discretized Space Planning
  - Potential Field Method
- Probabilistic Planners
- Optimal Planning

Definition

A **Path Planning Problem** is described by a triple $\langle \mathcal{X}_{free}, x_{start}, \mathcal{X}_{goal} \rangle$, where

- $x_{start} \in \mathcal{X}_{free}$ is the start state
- $\mathcal{X}_{goal} \subset \mathcal{X}$ describes a goal region

Definition

A mapping $\tau : [0, 1] \rightarrow R^n$ onto a C-space $\mathcal{R}^n$ is called a

- **Path** if it describes a finite, continous trajectory.
- **Collision-free Path** if $Range(\tau) \subseteq \mathcal{X}_{free}$
- **Feasible Path** if it is collision-free, $\tau(0) = x_{start}$, and $\tau(1) \in \mathcal{X}_{goal}$

adapted from S. Karaman et.al. 2011 [15]

Feasible Path Planning

Feasible Path Planning requires planners to find a **feasible path** for any given path planning problem. The ideal planner is

- **correct** - all reported paths are feasible
- **complete** - if a feasible path exist, it will be found
- performs with **bounded runtime** - if no path exists, it will fail

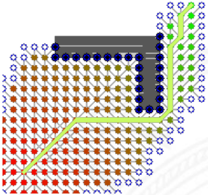
In practice,

- **correctness** is often traded for feasible runtime performance.
- *actual* correctness is defined by the real world, not by the planning model. If an object is not modelled, it will not be considered.
- most methods can not report failures and are only asymptotically complete.

Discretized Space Planning

Simple Idea: Discretize planning space & run A* on the resulting grid

- Classical path search algorithm
- Returns optimal plan in grid
- Works well for planar path planning



A* planner finding an optimal path in the grid

Not Tractable in 6+ dimensions

- Solutions limited to grid resolution
- Sufficiently high resolution required for correctness/completeness

- Discretization explicitly represents the whole space volume
- Curse-of-Dimensionality:
  - assuming 1 deg resolution and 360 deg joint range
  - 2 joints yield 129600 unique states
  - 3 joints yield 46656000 unique states
  - 6 joints yield $\sim 2.18e15$ unique states
- Explicit representation of the whole space is clearly not feasible.

Potential Field Method (PFM)

Alternative Idea: Represent space entirely through continuous function $f : R^n \rightarrow R$.

- No explicit space representation
- Can be evaluated as needed

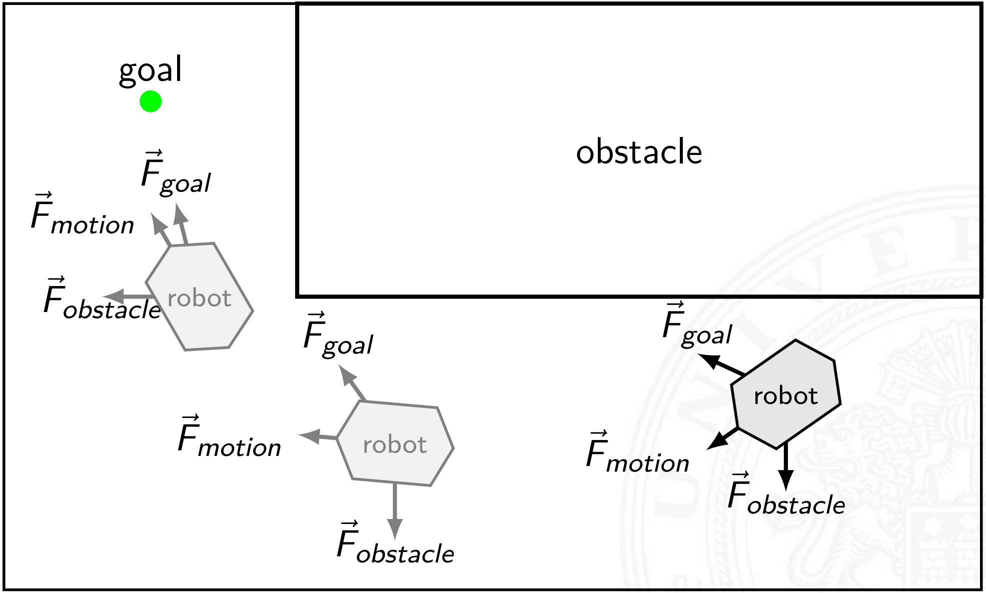
Khatib 1986:

> *The manipulator moves in a field of forces. The position to be reached is an attracting pole for the end effector and obstacles are repulsive surfaces for the manipulator parts.* [16]

Basic Principle

- Initially developed for real-time collision avoidance
- Potential field associates a scalar value $f(p)$ to every point p in space
- Robot moves along the negative gradient $-\nabla f(p)$, a "force" applied to the robot
- f's global minimum should be at the goal configuration
- An ideal field used for navigation should
  - be smooth
  - have only one global minimum
  - the values should approach ∞ near obstacles

Basic Principle (cont.)



Basic Principle (cont.)

- The attracting force (of the goal)

$$\vec{F}_{goal}(\mathbf{p}) = -\kappa_\rho(\mathbf{p} - \mathbf{p}_{goal})$$

  - where
    κ_ρ is a constant gain factor

Basic Principle (cont.)

- The potential field (of obstacles)

$$U(\mathbf{x}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\mathbf{p})} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(\mathbf{p}) \leq \rho_0 \\ 0 & \text{else} \end{cases}$$

  - where
    η is a constant gain factor
    $\rho(\mathbf{p})$ is the shortest distance to the obstacle O
    ρ_0 is a threshold defining the region of influence of an obstacle

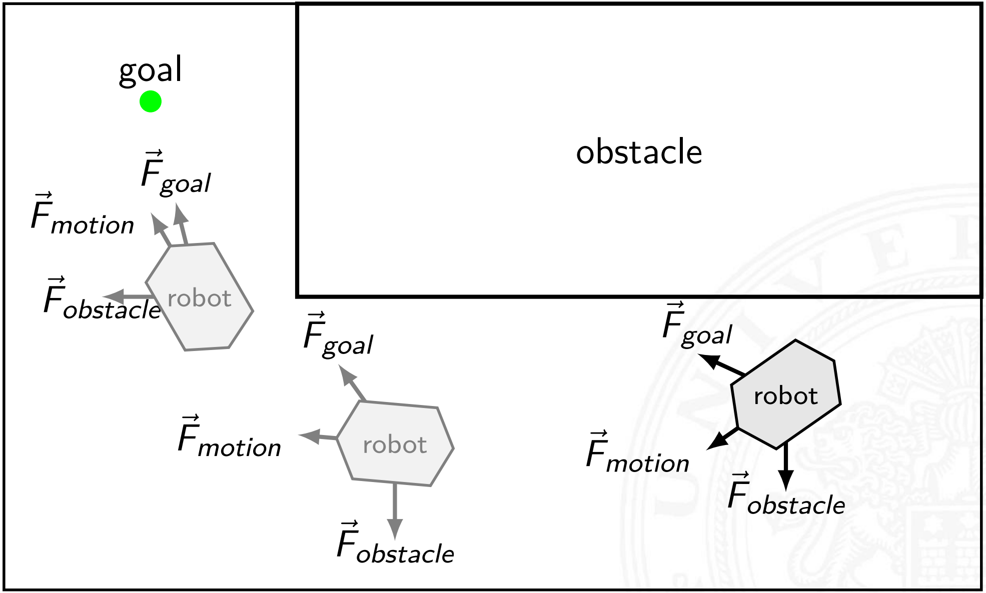
Basic Principle (cont.)

- The repulsive force of an obstacle

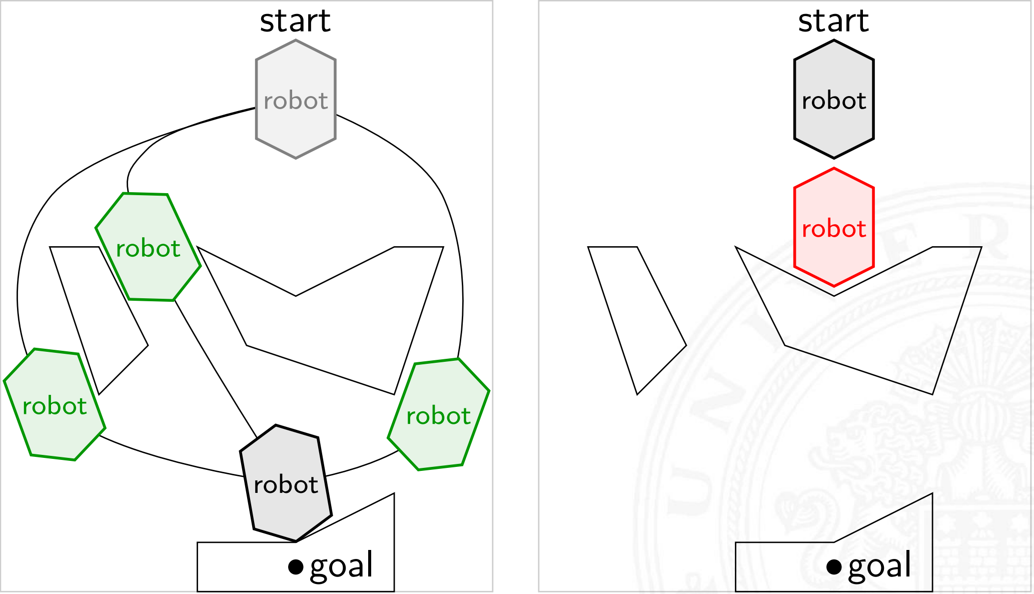
$$\vec{F}_{obstacle}(\mathbf{p}) = \begin{cases} \eta(\frac{1}{\rho(\mathbf{p})} - \frac{1}{\rho_0})\frac{1}{\rho(\mathbf{p})^2}\frac{d\rho(\mathbf{p})}{d\mathbf{p}} & \text{if } \rho(\mathbf{p}) \leq \rho_0 \\ 0 & \text{if } \rho(\mathbf{p}) > \rho_0 \end{cases}$$

  - where $\frac{d\rho(\mathbf{p})}{d\mathbf{p}}$ is the partial derivative vector of the distance from the point to the obstacle.

Basic Principle



Local Minima of PFM



Advantages and Disadvantages of PFM

Advantages:

- Implicit State Representation
- Real-time capable

Disadvantages:

- Incomplete algorithm
  - Existing solution might not be found
  - Calculation might not terminate if no solution exists
- $\rho(p)$ is only intuitive in 2D and 3D
- Obstacles in 6D C-space have complex shapes

Introduction to Robotics

Lecture 10

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

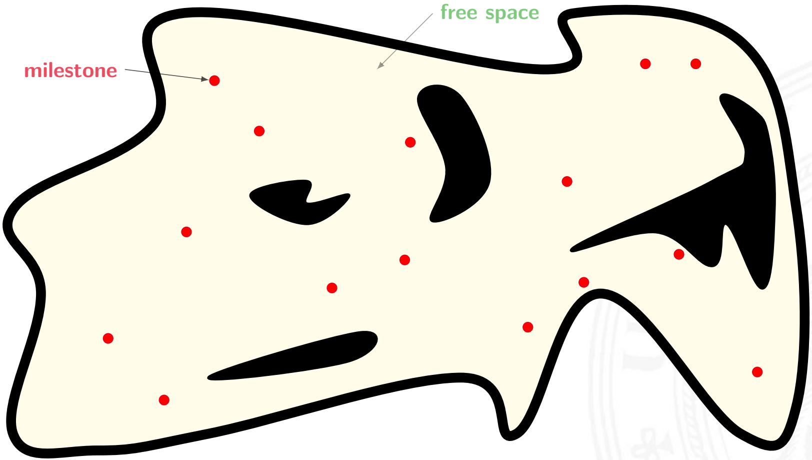
July 11, 2020

Path Planning

- Feasible Trajectories
- Geometry Representations
- C-Space
- Planner Approaches
- Probabilistic Planners
  - Probabilistic Road Maps
  - Rapidly-exploring Random Trees
  - Expansive Space Trees
  - Auxiliary Techniques
- Optimal Planning

Idea

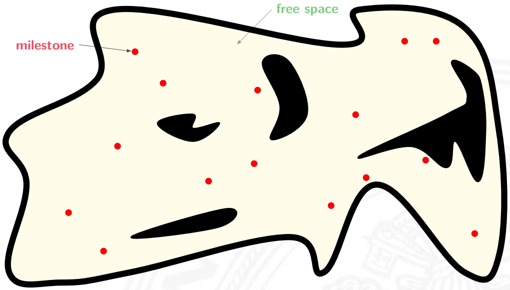
- Planning on graphs of reasonable size is simple
- Operating on grids ignores continuous spaces in $\mathcal{X}_{free}$
- Instead rely on **Probabilistic Sampling** to represent the space



Idea

Key questions:

- How to generate the samples?
- How can the samples be connected to form a planning graph?
- How many samples do you need to describe the space?



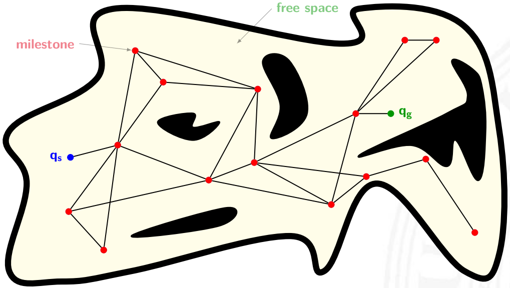
Abstract C-space with sampled valid states

Probabilistic Road Maps

Proposed by Lydia E. Kavraki et.al. 1996 [17]

Two Step algorithm:

1. Construction Phase - Build Roadmap
2. Query Phase - Connect start and goal to graph and solve graph search



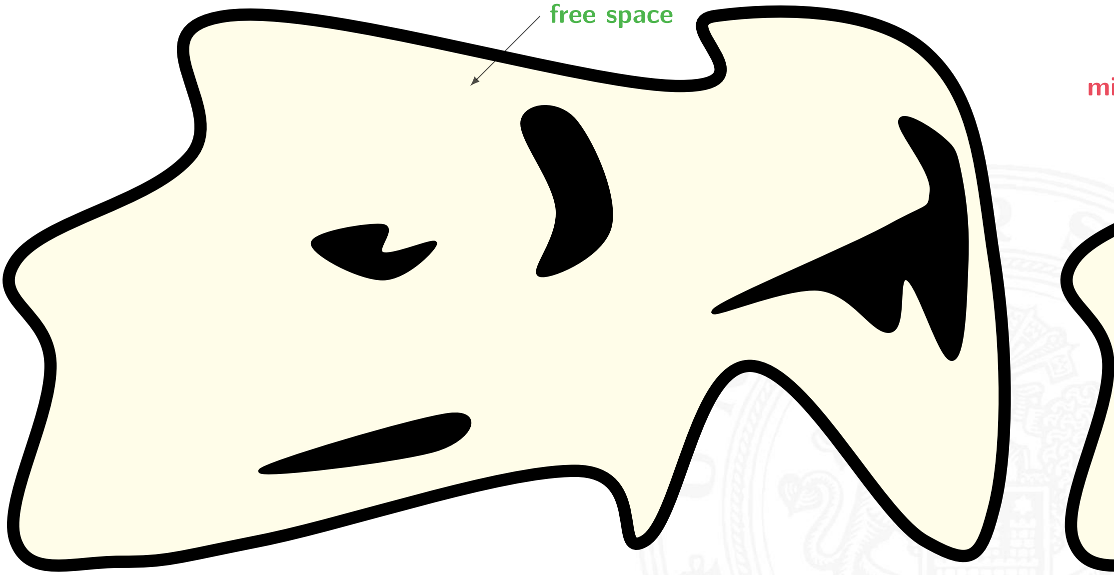
Abstract C-space with sampled valid states

Probabilistic Road Maps - Algorithm

Algorithm: sPRM

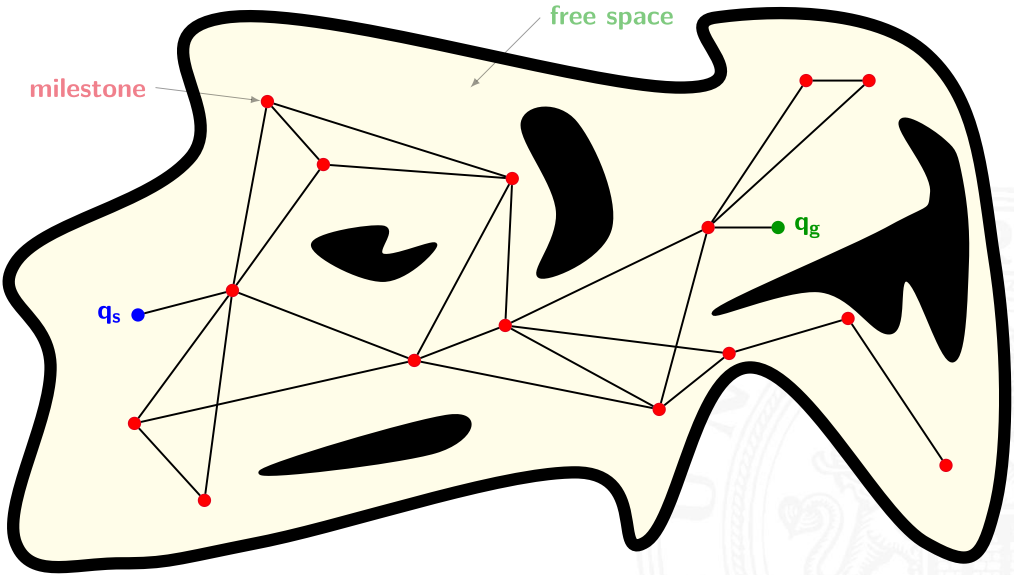
```
V ← {x_init} ∪ {SampleFree_i}_{i=1,...,n}; E ← ∅;
foreach v ∈ V do
    U ← Near(G = (V, E), v, r) \ {v};
    foreach u ∈ U do
        if CollisionFree(v, u) then E ← E ∪ {(v, u), (u, v)}
return G = (V, E);
```

Adapted from [15]



free space
mi

Milestones and Roadmap - Query



Probabilistic Road Maps - Aspects

Algorithm: sPRM

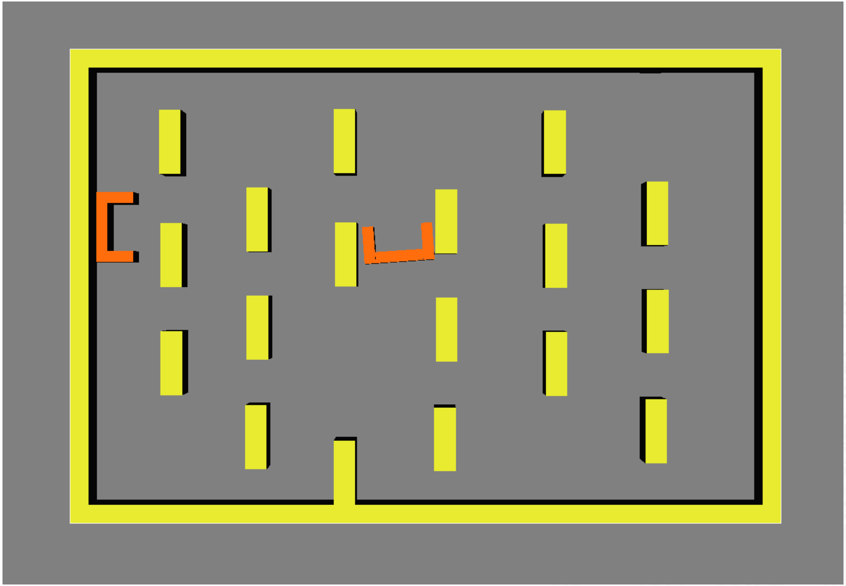
$V \leftarrow \{x_{\text{init}}\} \cup \{\texttt{SampleFree}_i\}_{i=1,\dots,n}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
  $U \leftarrow \texttt{Near}(G = (V, E), v, r) \setminus \{v\}$;
  foreach $u \in U$ **do**
    if $\texttt{CollisionFree}(v, u)$ **then** $E \leftarrow E \cup \{(v, u), (u, v)\}$
return $G = (V, E)$;

- `SampleFree` - Sample states from $\mathcal{X}_{free}$
- `Near` - Choose Distance metric and threshold
- `CollisionFree(v,u)` - Check motion **between** states for collisions

`SampleFree` – sample states from $\mathcal{X}_{free}$

- Traditionally: Rejection Sampling
  Take samples uniformally, add sample if $x \in \mathcal{X}_{free}$
- Alternatives:
  - Projective Sampling: Replace samples $x \in \mathcal{X}_{obs}$ by closest state $x' \in \mathcal{X}_{free}$
  - Generative Sampling: For a sufficient **parameterized** space $\mathcal{X}' \subset \mathcal{X}_{free}$:

Example



3dof planning problem

Single and Multi-Query Planning

Definition
If only a single path is requested in a potentially changing scene, this is called **single-query** planning. If datastructures remain valid between motion requests, this is called **multi-query** planning.

PRM solves a multi-query problem by building an undirected graph.

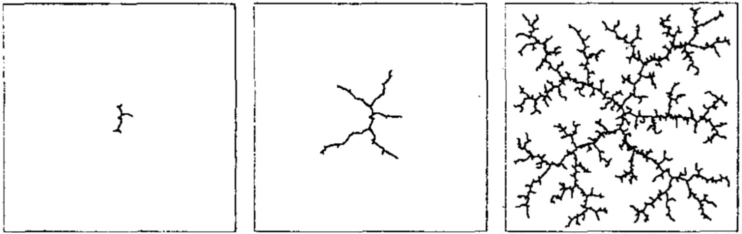
For single-shot planning, the graph search can be avoided altogether.

Rapidly-exploring Random Trees (RRT) - Basic Idea

Proposed by Kuffner and LaValle 2000 [18]

Instead of building a graph, grow a tree from the start state.

If for any leaf state $x \in \mathcal{X}_{goal}$, a solution is found.



RRT at multiple stages of extension

Rapidly-exploring Random Trees (RRT) - Algorithm

Algorithm 3: RRT

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
    $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
    $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
    $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
    if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
        $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$;
return $G = (V, E)$;

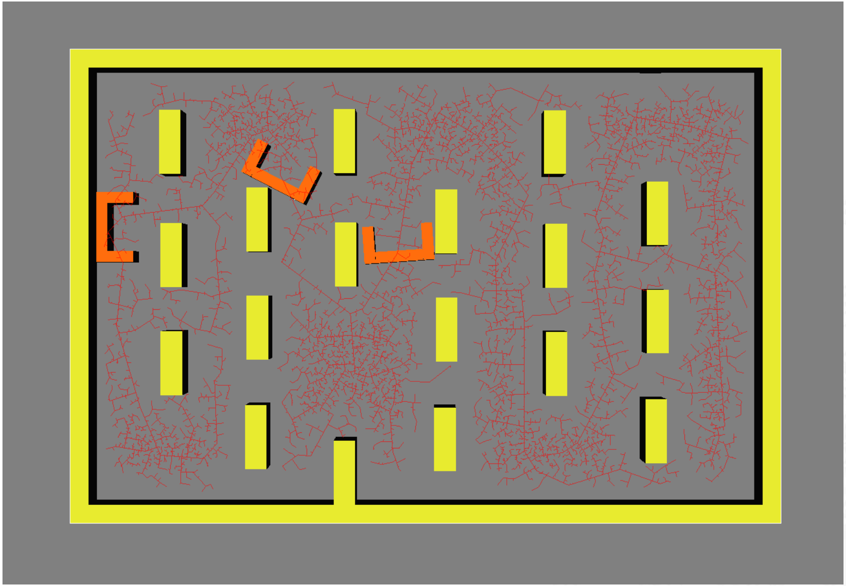
$\texttt{Steer}(x, y)$ - Compute new state x'

- Move from x towards y: $\|y - x'\| < \|y - x\|$
- $\|x - x'\| < \eta$ to limit step size
- Alternatively compute closest $x' \in \mathcal{X}_{free}$ reachable via straight motion

$\texttt{SampleFree}$ – sample states from $\mathcal{X}_{free}$

- Traditionally: uniform sampling

Example



RRT graph of an example

Bi-Directional Search

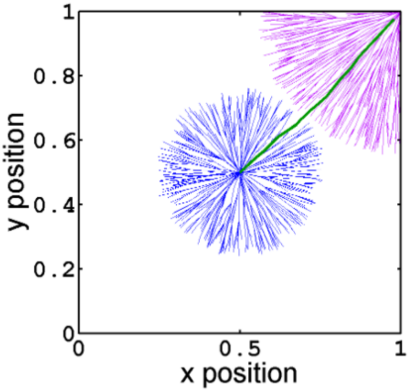
In robotics, start and goal are often in constraint areas of $\mathcal{X}_{free}$, e.g., close to obstacles.

The **transition phase** between these states is often quite flexible.

Instead of growing a single tree towards the goal

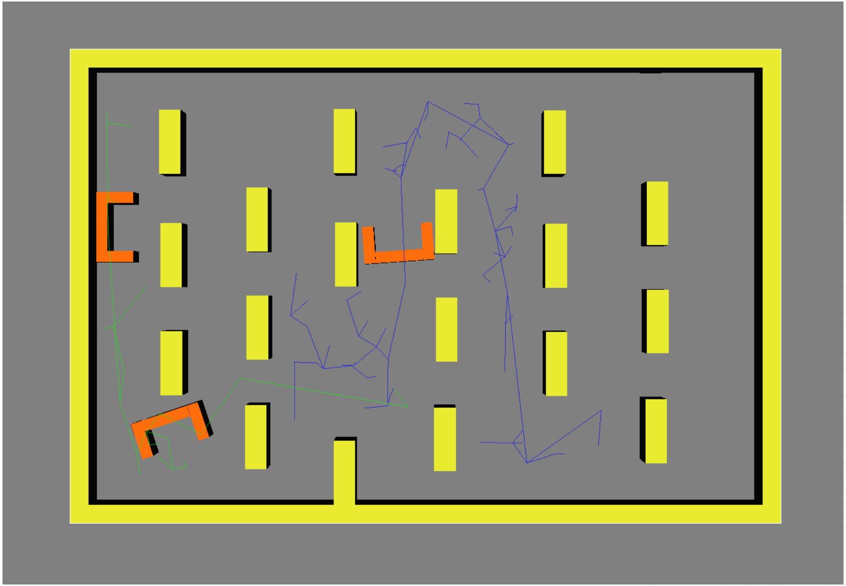
- Grow two trees from start and goal each.
- Attempt to connect them at each step.

In practice, this speeds up planning to the first solution significantly.



Bi-directional search trees [19]

RRT-Connect - Example



RRT-Connect for an example

Expansive Space Trees

PRM and RRT sample random configurations from $\mathcal{X}_{free}$.
Thus they also sample in areas which are already well-represented by milestones.

Definition

The *density* around a state x can be represented by
the cardinality of its neighborhood within a distance d: $|N_d(x)|$.

Ideas

- Sample next expansion step weighted by inverse density $w(x) = \frac{1}{|N_d(x)|}$
- Stochastically reject samples in high-density areas

Hsu et.al. 1997 [20]

Expansive Space Trees - Algorithm

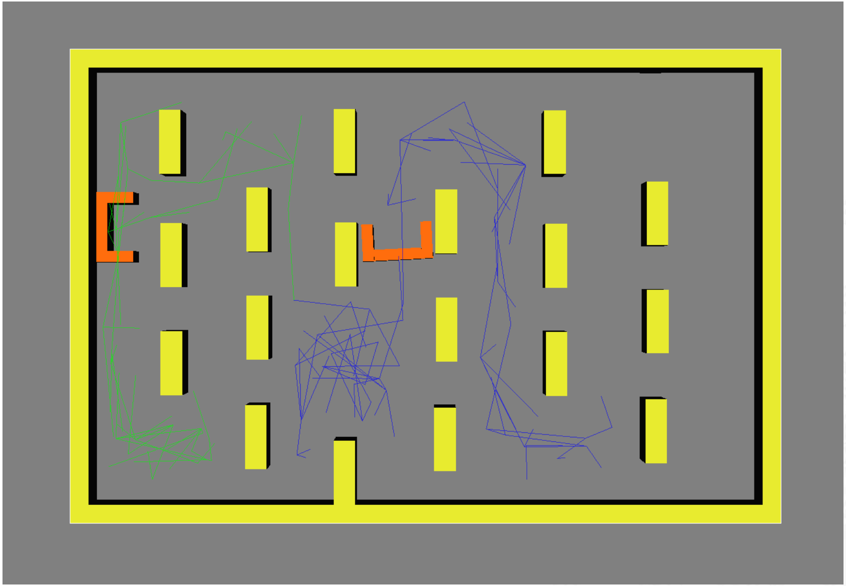
Algorithm *expansion*

1. Pick a node x from V with probability $1/w(x)$.
2. Sample K points from $N_d(x) = \{q \in \mathcal{C} \mid dist_c(q, x) < d\}$, where $dist_c$ is some distance metric of $\mathcal{C}$. (K and d are parameters.)
3. **for** each configuration y that has been picked **do**
4. calculate $w(y)$ and retain y with probability $1/w(y)$.
5. **if** y is retained, $clearance(y) > 0$ and $link(x, y)$ returns YES
6. **then** put y in V and place an edge between x and y.

- Expand from an existing node instead of global samples from $\mathcal{X}$
- Samples rejected in 4. are never collision checked!
- Original formulation is bidirectional

Hsu et.al. 1997 [20]

(Bi)EST - Example



(Bi-directional) EST for an example

Path Postprocessing

The resulting paths are not smooth and often contain unnecessary motions.

Traditional post-processing includes:

- Path Shortcutting
  - Repeatedly pick two non-consecutive waypoints and attempt to connect them
- Perturbation of individual waypoints
  - Optional
  - Can reduce solution costs
  - Computationally expensive
  - For differentiable costs: exploit gradient
- Fit smooth splines through waypoints

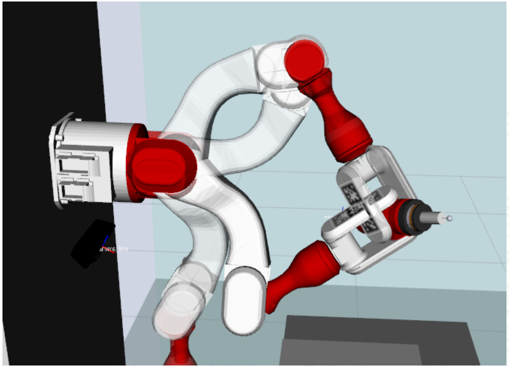
All modifications need to be collision checked.

Multiple IK Solutions

Redundant robots generate multiple joint solutions per pose.

Each Cartesian goal region adds a number of disjoint C-space goal regions.

Most tree-based planners naturally extend to **Multi-Goal Planning**, implicitly building multiple goal trees.



Multiple IK solutions for one target pose © Hendrich

Optimal Planning

Definition

An **Optimal Path Planning Problem** is defined by a path planning problem $\mathcal{P} = \langle \mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal} \rangle$ and a cost function $c(\tau) : R \geq 0$. It requires to find a feasible path τ^* such that $\tau^* = \text{argmin}_\tau \{c(\tau) \mid \tau \text{ is feasible for } \mathcal{P}\}$

In practice:

- Two-step process:
  - Find feasible path(s)
  - Optimize path(s)
- Planners are **asymptotically** optimal
  - Convergence might take long
  - Non-trivial to detect ε-optimal solution
- What cost function should be used?
  - C-space path length
  - Accumulated clearance (distance to obstacles)
  - Cartesian end-effector path length
  - Physical work

Planner*

Method

Instead of stopping at the first trajectory, continue sampling to improve solution.

Karaman and Frazzoli 2011 [15] introduced **PRM*** and **RRT***.
Both are efficient, asymptotically optimal versions of the basic algorithms.

PRM is asymptotically optimal as-is.

- Eventually all points on the optimal path will be added to the roadmap.

Ensure minimal required graph connectivity of $O(n \cdot \log(n))$.

- Reduce the neighborhood radius r with sample size n:

$$r(n) = \gamma_{PRM} \cdot \left(\frac{log(n)}{n}\right)^{\frac{1}{d}}$$

where γ_{PRM} depends on the planning space, d is the dimensionality of $\mathcal{X}$

RRT*

Method

Update tree whenever new samples yield cheaper paths to root.

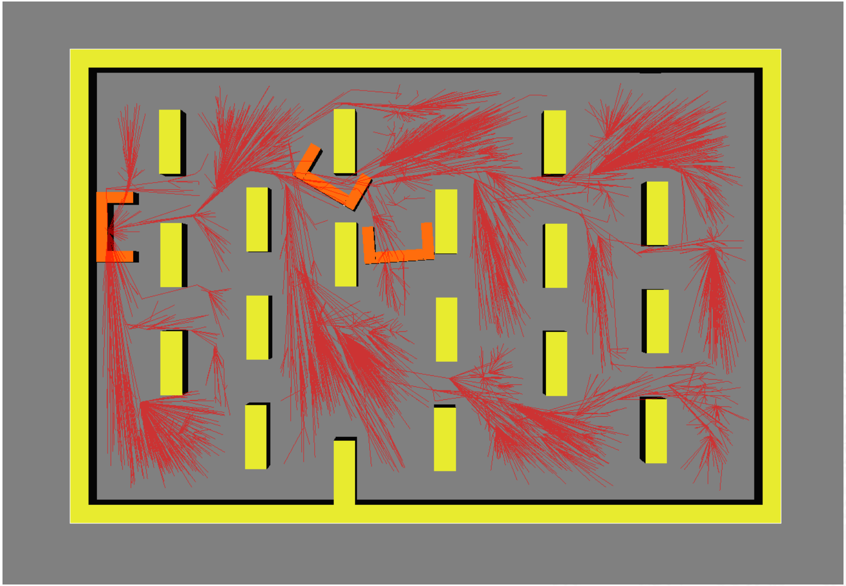
- Instead of connecting the new states to *closest node*, connect to the *cheapest node* in neighborhood
- Change parent of neighboring states to new state if new path is cheaper

RRT* - Algorithm

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
    $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
    $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
    $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
    if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
        $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
        $V \leftarrow V \cup \{x_{\text{new}}\}$;
        $x_{\text{min}} \leftarrow x_{\text{nearest}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
        foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
            if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\text{min}}$ **then**
                $x_{\text{min}} \leftarrow x_{\text{near}}$; $c_{\text{min}} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
        $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
        foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
            if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$
            then $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
            $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

RRT* - Example

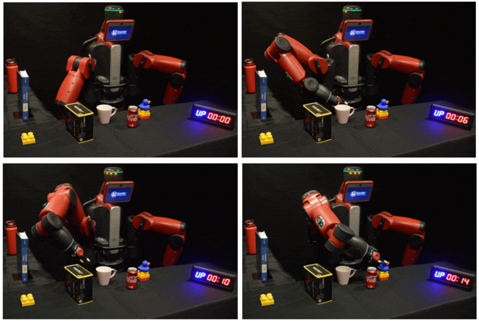


RRT* for an example

Summary - Sampling Based Planning

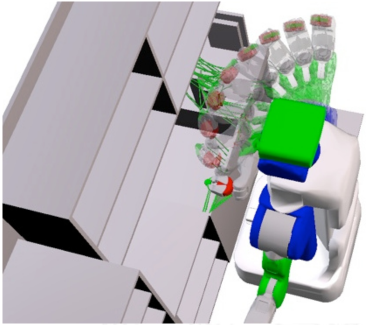
- Represent $\mathcal{X}_{free}$ probabilistically through samples
- Relies heavily on binary collision checking
- Post-processing solutions is essential
- Various (dozens) of algorithms with varying performance
- Straight-forward extensions for asymptotically optimal planning

MPNet



Fast deep-learning system learning from planners [21]

TrajOpt



Sequential convex optimizer solving trajectories [22]

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

Task/Manipulation Planning

- Grasp Detection
- Task Planning

Outline (cont.)

- Multi-Modal Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

Motivation

Robotic manipulation consists of more than waypoint-to-waypoint planning.

Even with a perfect path planner,

- **Where should you go?**
  - *Grasp Planning*
- **In what order should you go there?**
  - *Task Planning*
- Different planning steps usually operate in **different** $\mathcal{X}$ **or** $\mathcal{X}_{free}$
  - *Multi-Modal Planning*

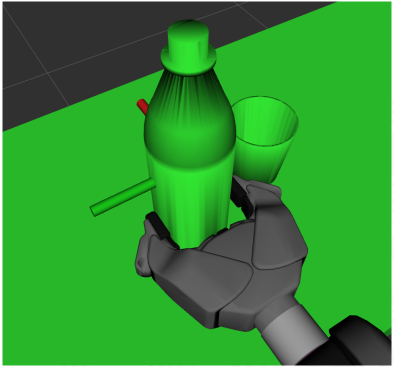
The field is extremely spread out and only a few ideas are mentioned here.

Identify Grasps

If you know where your object is,
you can annotate fixed grasps.

To pick up the object, move to
Cartesian pose relative to object.

Shortcoming
Pose must be reachable and collision-free.



Single Grasp for a bottle mesh

Identify Grasps - By Simulation

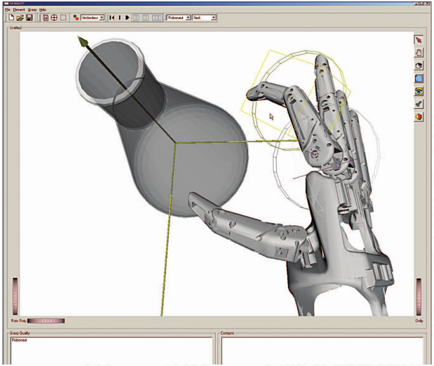
For complex manipulators, the grasp has many parameters.

Approach

Simulate force interaction to generate reachable, stable grasps.

Shortcomings

- Computationally expensive
- Grasps without natural interpretation/use intention



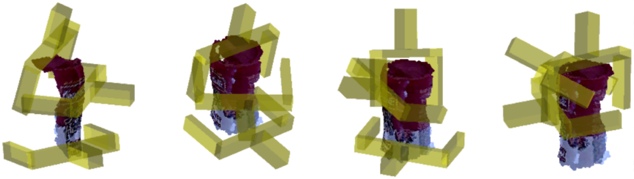
GraspIt: Grasp stability simulator [23]

For unknown or unmodelled objects, neither method is usable.

Approach

Learn to estimate good grasps from vision.

- Predict success rate for candidate grasps
- Or directly predict grasp parameters
- Often restricted to < 6 degrees of freedom (2 or 3)



Grasp candidates for a two-finger parallel gripper grasping a can

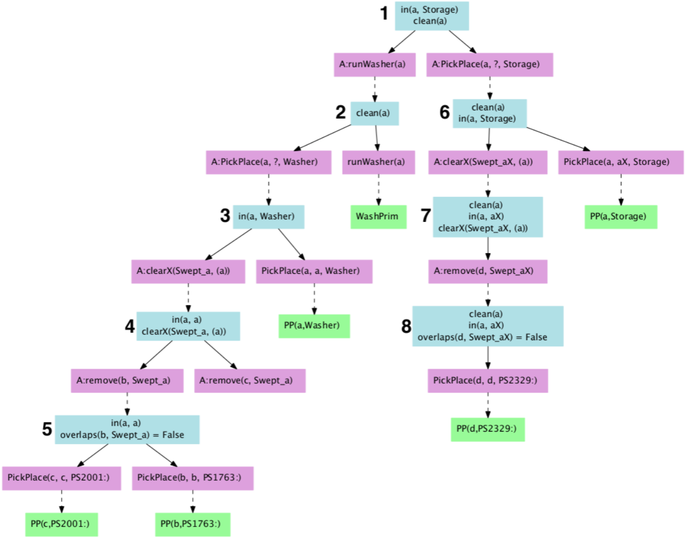
ten Pas et.al. 2017 [24]

Task Planning

Definition
Task Planning refers to the process of finding a feasible sequence of actions and their parameters to achieve a specified goal.

Requires well-defined action descriptions and goal specifications, e.g. `pickup(a)`.

Hierarchical Task Network



HTN plan for cleaning a through a washer and storing it away [25]

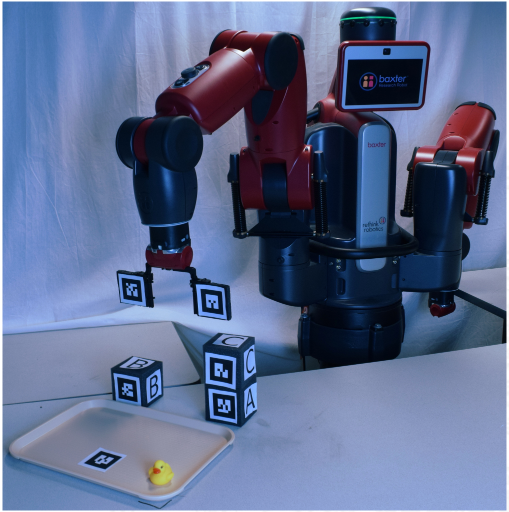
Task and Motion Planning (TMP)

In robotics, task planning and motion planning are often entwined.

To pickup `A`, `C` has to be moved away.

Action preconditions include reachability constraints solved through Path Planning.

In practice these constraints are often implicit.



TMP framework implementing a traditional blocks-world task [26]

Motion Phases

Manipulation actions can be split up in motion phases with different concerns.

- Transit phase
  - Move towards object
- Approach phase
  - Move in contact with object
- Stabilization phase
  - Acquire sufficient grasp
- Lift phase
  - Retract grasped object from surface
- ...

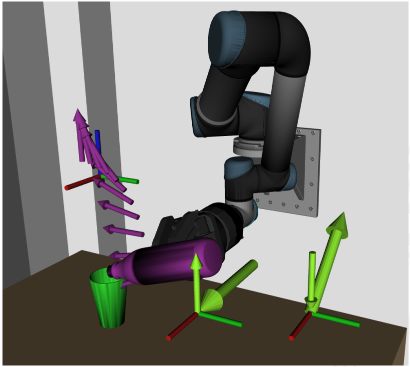
Motion Phases

These different motions . . .

- Require different controllers
  - Position control, effort control, impedance control
- Have different motion characteristics
  - Restricted approach direction or variable free-space motion
- Have different validity concerns
  - Transit must not collide, approach will collide with object
- Actuate different joint sets
  - Gripper, arm, mobile base

Approach

- Split up manipulation action along custom motion phases
- Allow custom path solvers for each phase
- Exchange interface states between the solvers



Combined manipulation plan to pick, pour from and place a bottle [28]

Multi-Modal Planning Through Sampling

Idea

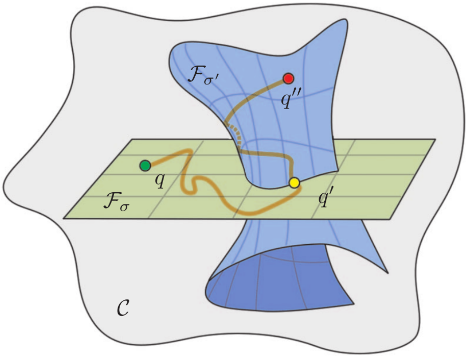
Manipulation plans can be interpreted as connected paths on multiple intersecting manifolds in $\mathcal{X}$.

Picking up an object might consist of

- Moving to a pose from which grasping is possible
- Moving grasped object to target location

Approach

Sample from each manifold *and each intersection* in turn.



Sketch of two intersecting planning manifolds [29]

Introduction to Robotics

Lecture 11

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction
Spatial Description and Transformations
Forward Kinematics
Robot Description
Inverse Kinematics for Manipulators
Instantaneous Kinematics
Trajectory Generation 1
Trajectory Generation 2
Dynamics
Robot Control
Path Planning
Task/Manipulation Planning
Telerobotics
- Introduction

Outline (cont.)

- Teleoperation classification by input devices
- Bilateral control and force feedback
- Go beyond teleoperation

Architectures of Sensor-based Intelligent Systems

Summary

Conclusion and Outlook

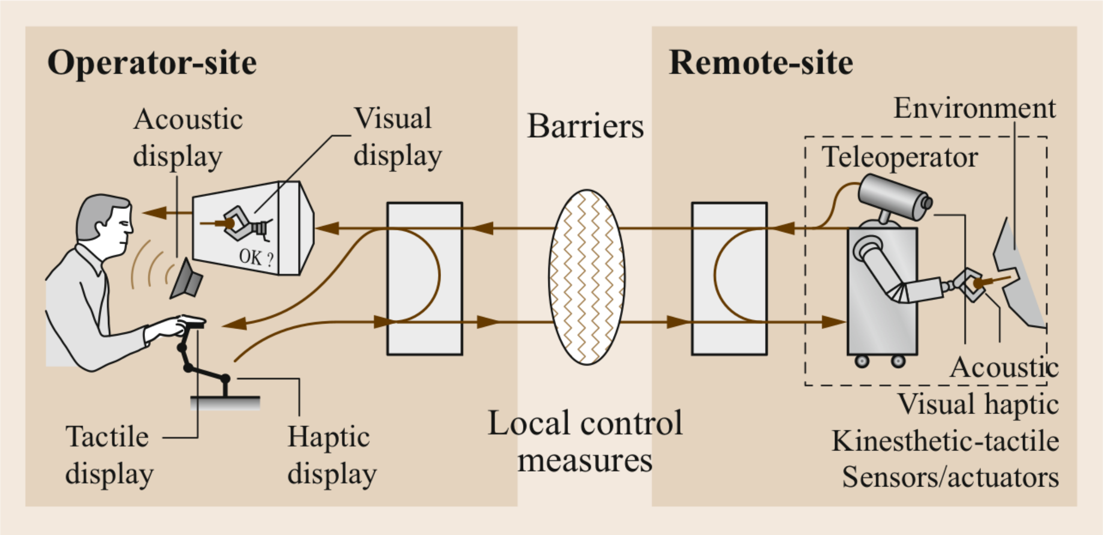
Motivation

Motivation

- Human-in-the-loop
- Handle unknown and hazardous environments
- Take fast decisions and dealing with corner cases

Telerobotics is perhaps one of the earliest aspects and manifestations of robotics.[30]

Telerobotics system



Terminologies

- *Telerobotics*
- *Teleoperation*
  - task-level operations
- *Telemanipulation*
  - object-level manipulation
- *Master-slave systems*
- *Telepresence*
  - an ultimate goal of master–slave systems and telerobotics in general
  - Bilateral telemanipulation

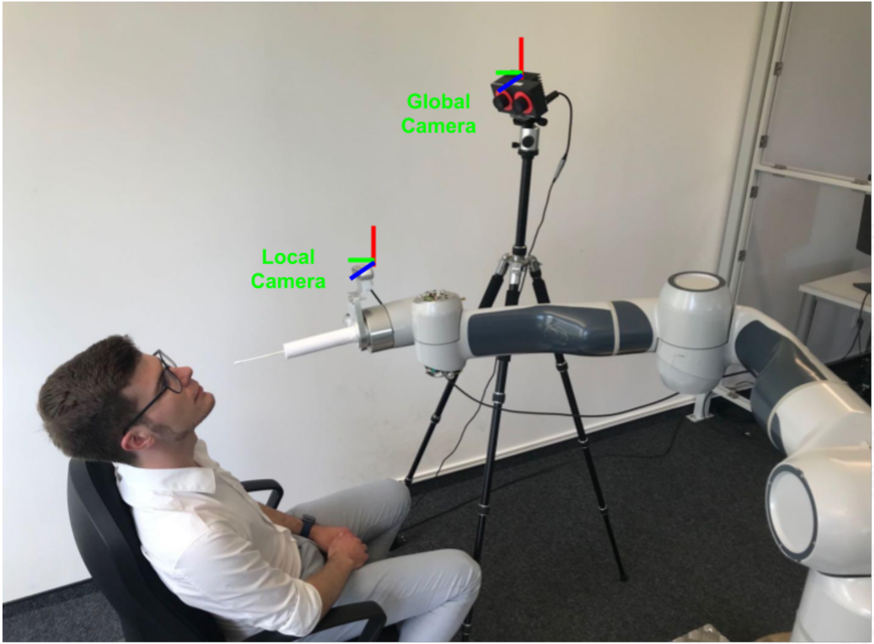
Control architectures in Telerobotics

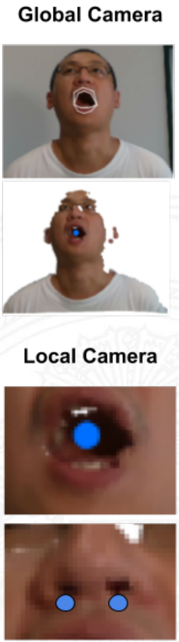
- *Direct control/manual control*
  - the user is controlling the motion of the robot directly
- *supervisory control*
  - the users only provide high-level commands
  - allow more autonomy and intelligence to shift to the robot system
  - is advantageous to the telerobotic systems with large time delays
- *shared control*
  - combine the basic reliability and sense of presence achievable by direct control with the smarts and possible safety guarantees of autonomous

Swab sampling robot – shared control

Automatic swab robot

Automatic swab robot

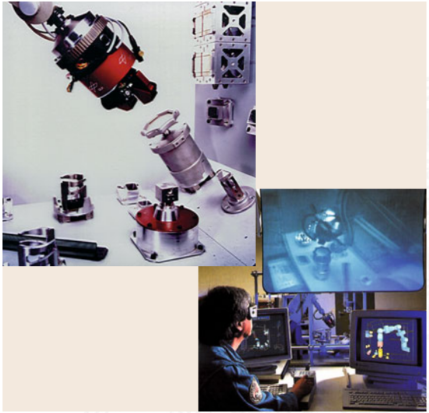




Telerobotic applications

- Robots in hazardous/unstructured workplaces
  - Nuclear robots – where telerobotics starts
  - Space robots
  - Rescue robots

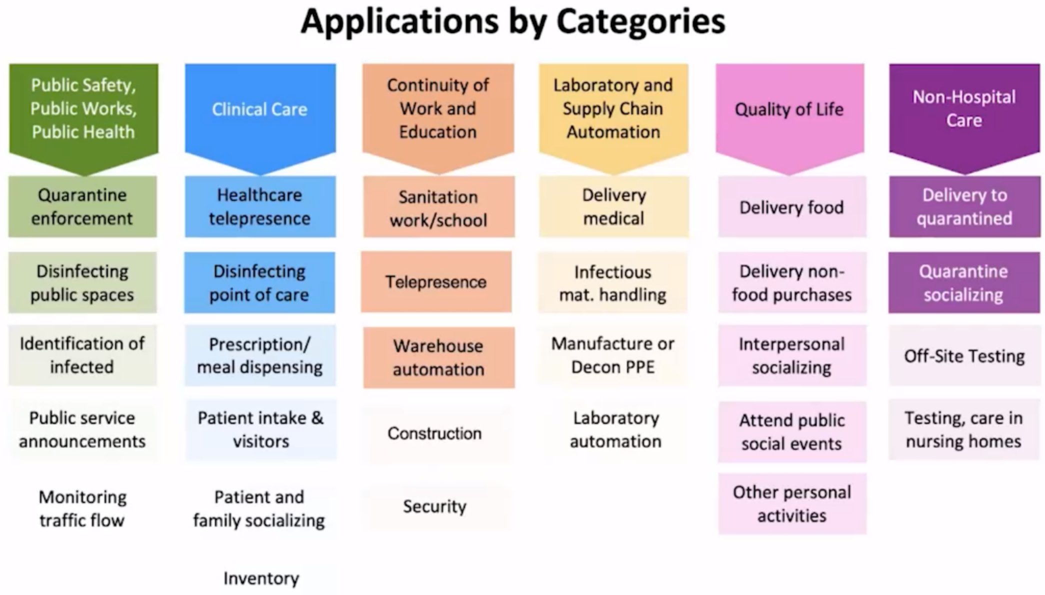
Raymond C. Goertz



ROTEX
the first teleoerated space robot

- Medical robots – Da vinci robots

- ICRA2020 Plenary Panel - COVID-19 : How Roboticist Can Help?

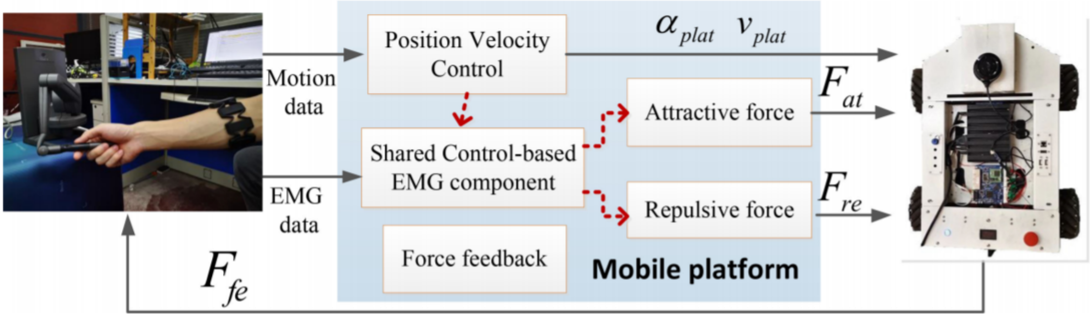


Telerobotics in medical robots

- Surgical robots
- Incorporate haptic feedback
- Multisensory (image (endoscopy), haptic, IMU) fusion
- most are shared control

Telerobotics in mobile robots

- Shared control
- Obstacle avoidance

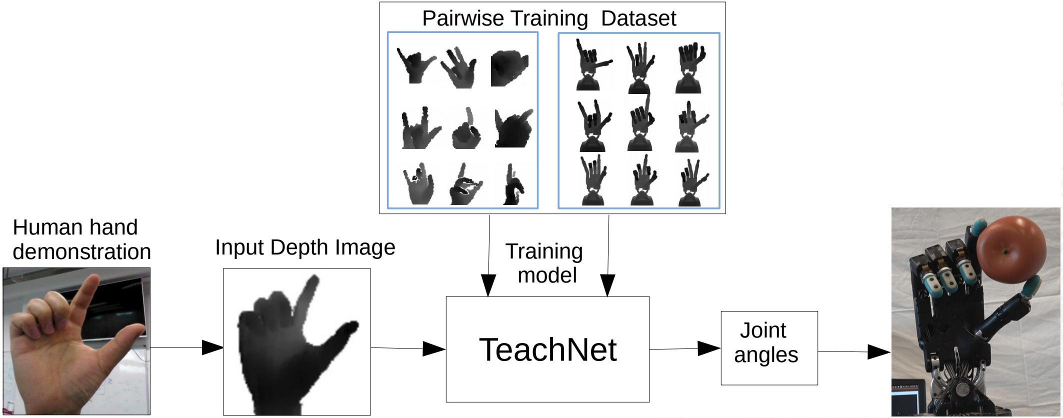


26

---

[26] Luo et al. A Teleoperation Framework for Mobile Robots Based on Shared Control. IEEE Robotics and Automation Letters. 2020

- Direct control
- An end- to-end fashion



27

[27] Li, et al. TeachNet: Vision-based Teleoperation for Shadow Hand. ICRA2019

Control variables in hand teleoperation

- joint mapping
- fingertip mapping
- pose mapping

Problems in Telerobotics

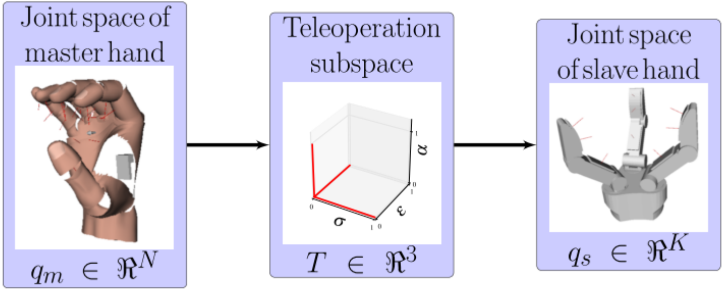
- Time delay
- Force feedback
- Teleoperation between dissimilar kinematic structures
- Multilateral Telerobotics

Robot classification by input devices

- Contact devices
  - Joystick
  - Apriltags
  - wearable gloves/suits/glass
    - Data glove
    - Optical markers
    - IMU (Inertial and magnetic measurement unit)
    - EMG (Electromyography)
    - VR/AR device
    - Haptic devices
- Contactless devices
  - Depth camera(s)
  - Ultraleap

Dataglove

- Cyberglove or wired glove
- Intuitive Hand Teleoperation
  - a low-dimensional and continuous teleoperation subspace
  - mapping between different hand pose spaces



1

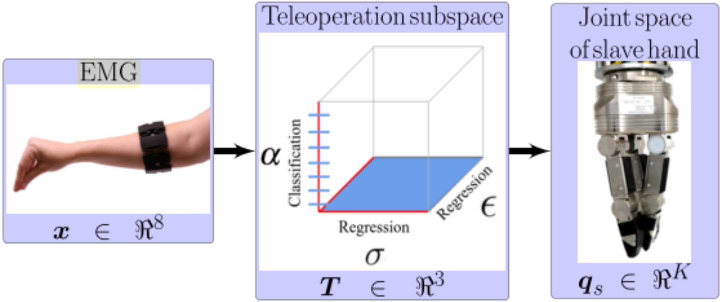
[1] Meeker, et al. Intuitive Hand Teleoperation by Novice Operators Using a Continuous Teleoperation Subspace. ICRA2018

Markers-based teleoperation

- Multi-camera motion capture systems, such as PhaseSpace, OptiTrack
  - accurate point tracking solutions
  - suits must be customized and easily obstruct natural joint motions
  - the correspondence problem between markers on the fingers and cameras

EMG-based teleoperation

- Commercial devices, such as Myo Armband
- EMG-controlled hand teleoperation
  - extracted force information from skeletal muscles through surface EMG
  - mapping forearm EMG into a subspace relevant to teleoperation



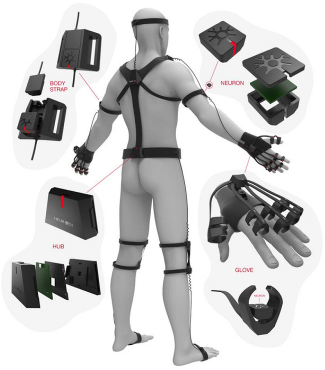
[1] [2]

[1] Meeker, et al. EMG-Controlled Hand Teleoperation Using a Continuous Teleoperation Subspace. ICRA2019

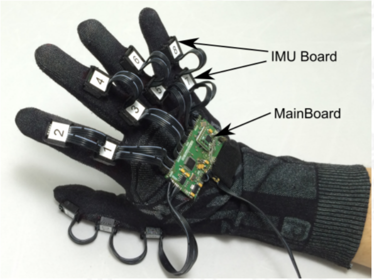
[2] Wen, et al. Force-guided High-precision Grasping Control of Fragile and Deformable Objects using sEMG-based Force Prediction. ICRA2020

IMU-based teleoperation

- Commercial devices, such as PerceptionNeuron
- Sensitive to magnetic/metal environments
- Convert the orientation, angular velocity and acceleration information of human into the control instruction flow of the robotic hand-arm



PerceptionNeuron



Cie-dataglove

Summary of EMG- and IMU-based methods

Advantages:

- Inexpensive
- Easy to setup and use

Disadvantages:

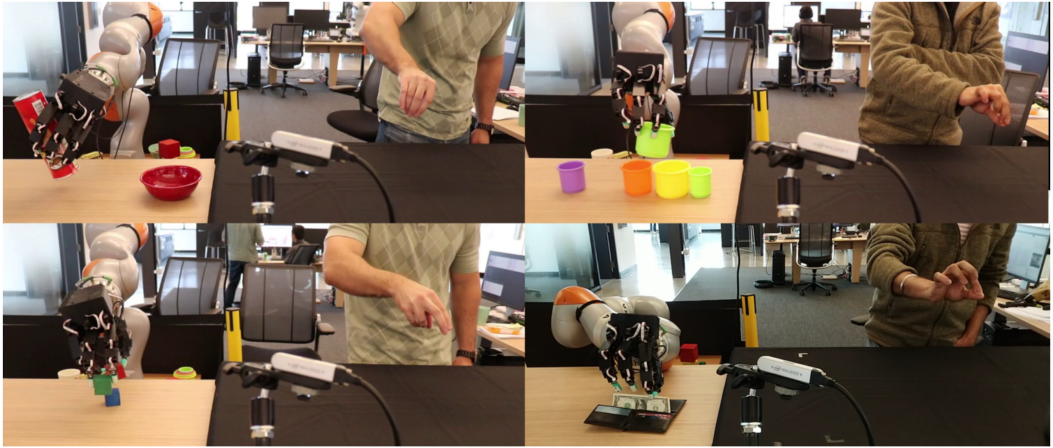
- Provide less versatility and dexterity
- Necessary calibration before start to use it
- More suitable for robotic arms

MR-based teleoperation

28

[28]Krupke, et al. Comparison of Multimodal Heading and Pointing Gestures for Co-Located Mixed Reality Human-Robot Interaction. IROS2018

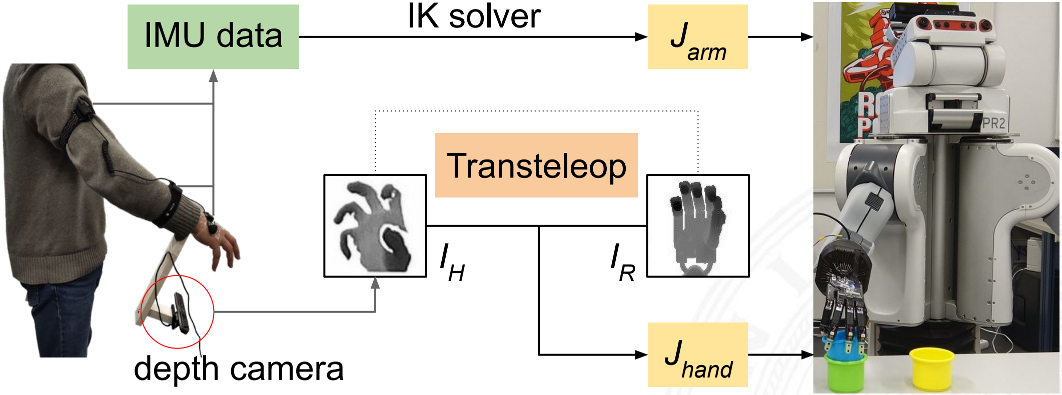
DexPilot: Vision Based Teleoperation of Dexterous Robotic Hand-Arm System



29

[29] Handa, et al. DexPilot: Vision Based Teleoperation of Dexterous Robotic Hand-Arm System. ICRA2020

A Mobile Robot Hand-Arm Teleoperation System by Vision and IMU



[30]

[30] Li, et al. A Mobile Robot Hand-Arm Teleoperation System by Vision and IMU. IROS2020

Summary of Vision-based methods

Advantages:

- Inexpensive
- Easy to setup and use
- Allow natural, unrestricted limb motions and be less invasive

Disadvantages:

- Highly based on human cognitive
- Open-loop control

Future research:

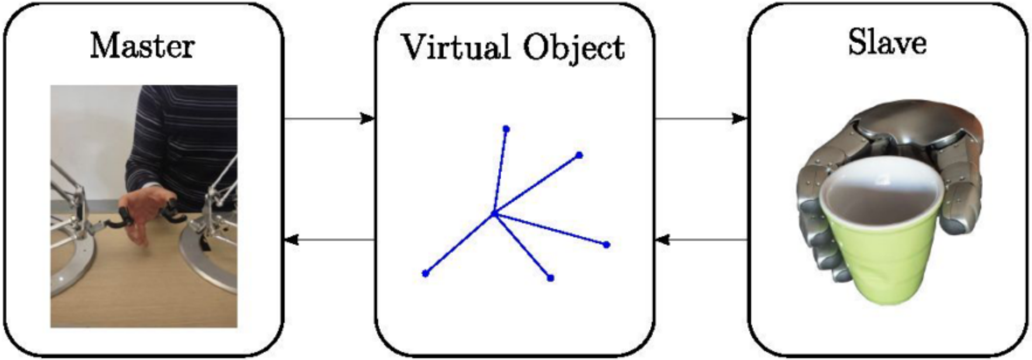
- Real-time hand tracking to achieve an unlimited workspace for the novice
- Closed-loop control (slip detection and force estimation)

Bilateral control and force feedback

- Provide both forward and feedback pathways from the user to the environment and back
- Explicit force feedback
  - the slave's controller forces, which include forces associated with the spring–damper and slave inertia
  - the external forces acting between the slave and the environment
- Also can use alternate displays, such as audio or tactile devices

Object-based Bilateral Telemanipulation

- The master sub-system setup has two Omega.3 haptic devices
- The slave robot is a DLR-HIT II Hand.



31

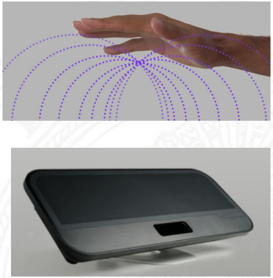
[31] Salvietti, et al. Object-based Bilateral Telemanipulation Between Dissimilar Kinematic Structures. IROS2013

Commercial devices

- CyberTouch (http://www.cyberglovesystems.com/cybertouch)
- HaptX gloves (https://haptx.com/technology)
- Ultraleap (https://www.ultraleap.com/haptics)

Cyber Touch

HaptX Gloves



Ultraleap

Imitation learning (learn from demonstration)

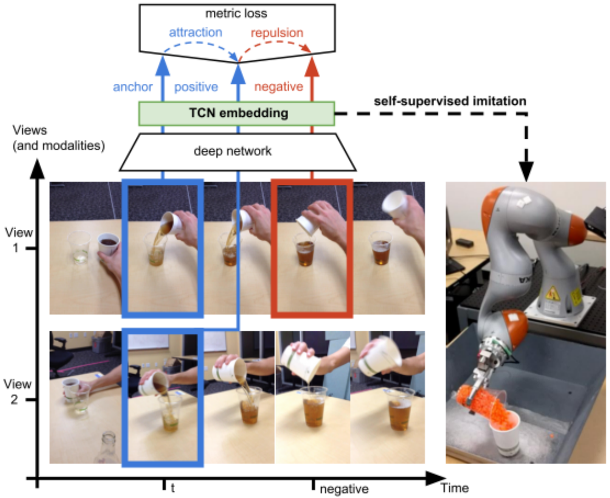
	Imitation learning	$\Longleftrightarrow$	Telerobotics
Given	demonstrations or demonstrator		demonstrations or demonstrator
Goal	train a policy to mimic demonstrations		repeat/copy demonstrations

Research goals

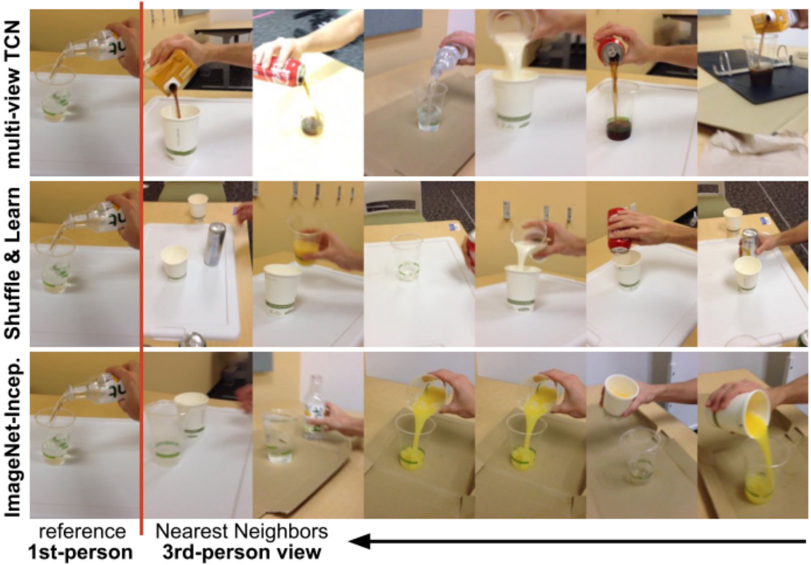
1. Learn suitable representations for understanding object interaction and enabling robotic imitation of a human
2. One-shot/few-shot learning
3. ...

Time-Contrastive Networks (TCN)[31]

- learn robotic behaviors from unlabeled videos recorded from multiple viewpoints
- Anchor, positive, negative



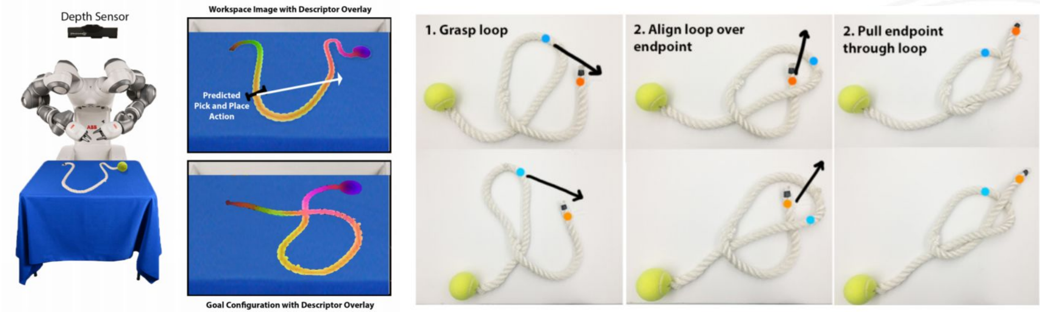
Label-free pose imitation by TCN



Label-free pose imitation by TCN

Learning Rope Manipulation Policies

- Dense depth object descriptors
- Learn from video demonstrations
- Trained on synthetic depth Data

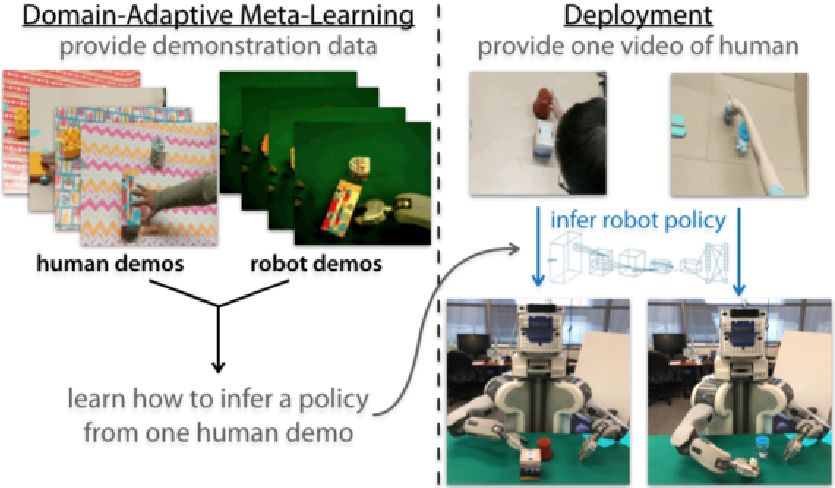


32

[32] Sundaresan, et al. Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data. ICRA2020

One-shot/few-shot imitation learning

- quickly learn a new task from a small amount of demonstrations
- Model-Agnostic Meta-Learning (MAML)[32]



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Introduction to Robotics

Lecture 12

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Outline (cont.)

- The CMAC-Model
- The Subsumption-Architecture
- Control Architecture of a Fish
- Procedural Reasoning System
- Hierarchy
- Architectures for Learning Robots

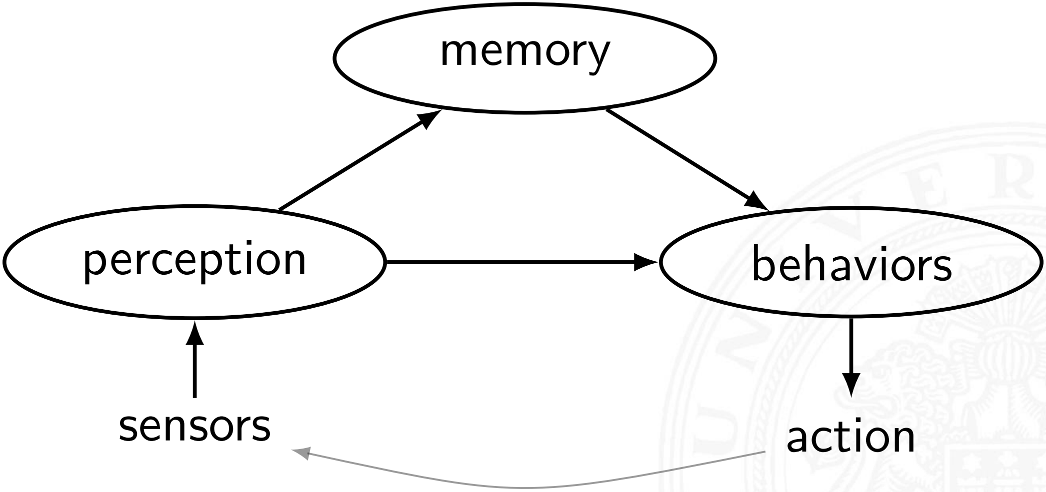
Summary

Conclusion and Outlook

Overview

- Basic behavior
- Behavior fusion
- Subsumption
- Hierarchical architectures
- Interactive architectures

The Perception-Action-Model with Memory



CMAC-Model

CMAC: `Cerebellar Model Articulation Controller`

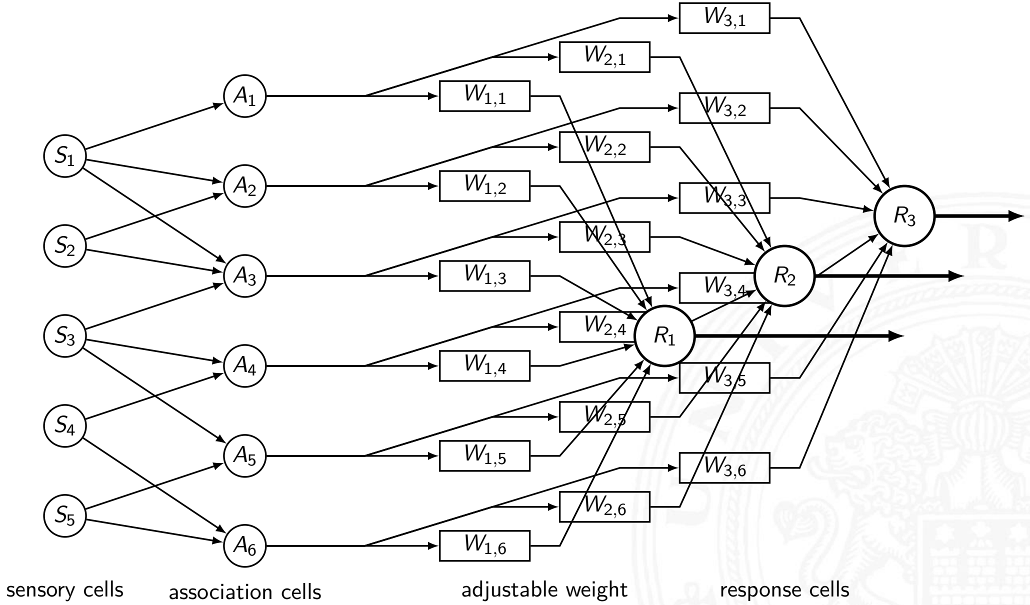
- **S** sensory input vectors (firing cell patterns)
- **A** association vector (cell pattern combination)
- **P** response output vector $(\mathbf{A} \cdot W)$
- W weight matrix

The CMAC model can be viewed as two mappings:

$$f : \mathbf{S} \longrightarrow \mathbf{A}$$

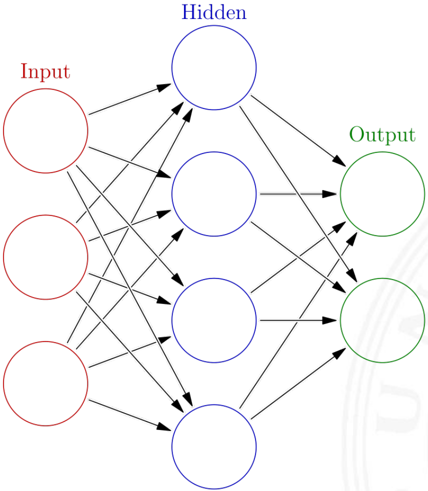
$$g : \mathbf{A} \xrightarrow{W} \mathbf{P}$$

CMAC-Model (cont.)



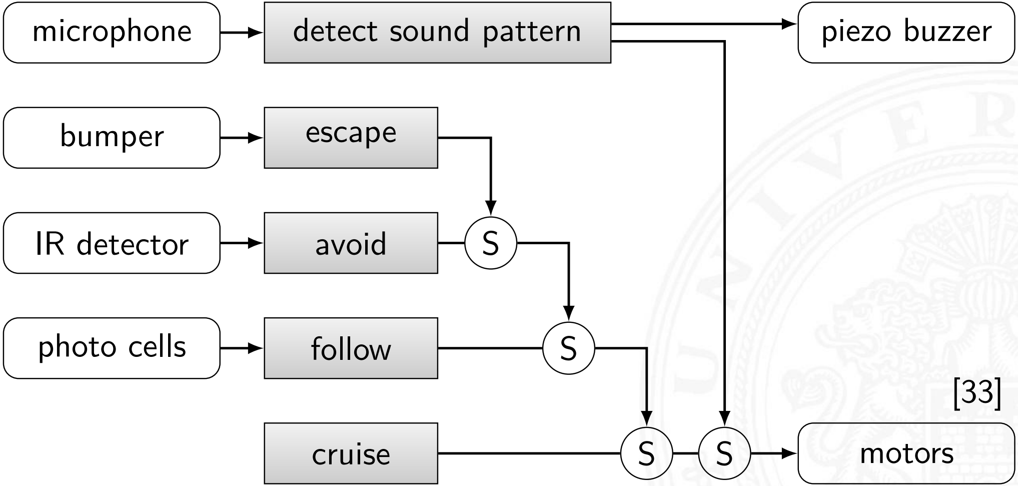
Artificial Neural Network

Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains.



The Subsumption Architecture

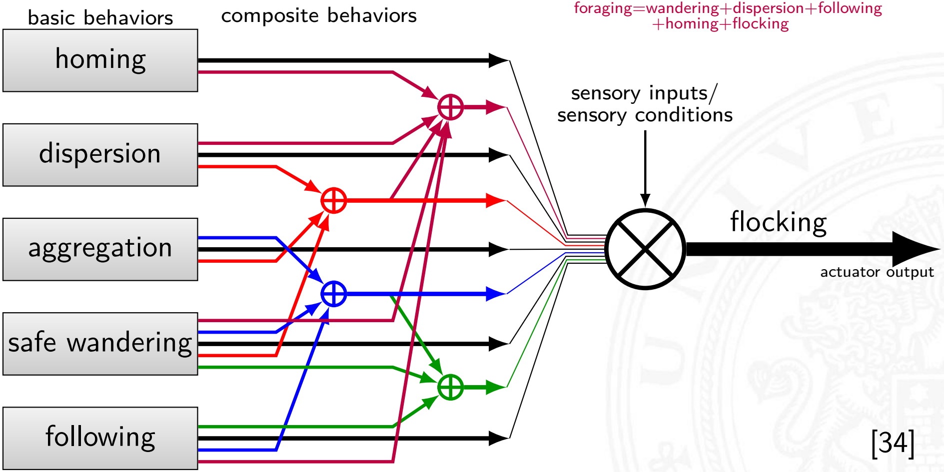
- hierarchical structure of behavior
- higher level behaviors subsumpe lower level behaviors



[33]

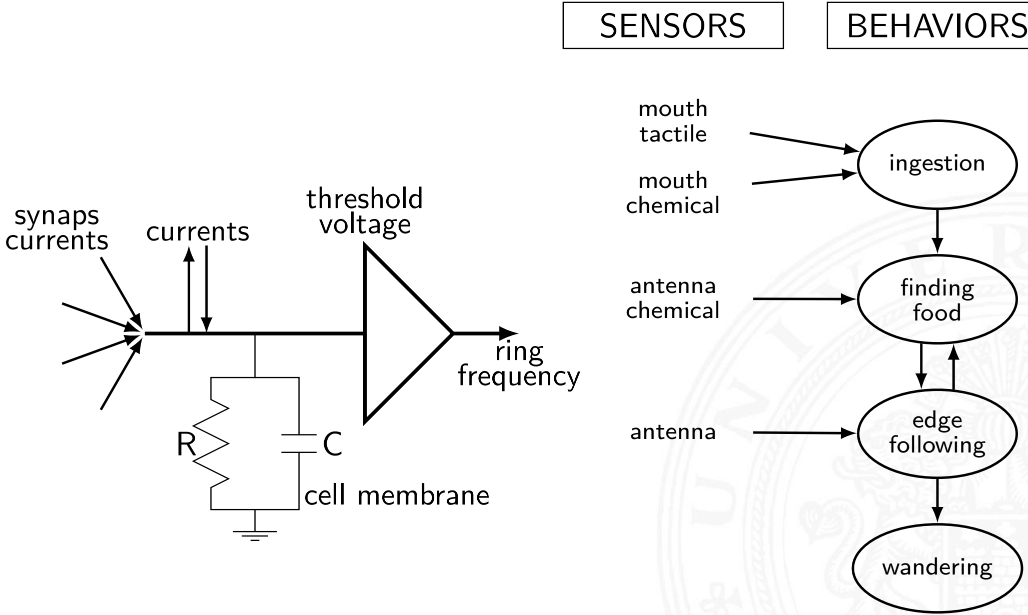
- multi-robot architecture
- basic behaviors are sequentially executed

flocking=wandering+aggregation+dispersion
surrounding=wandering+following+aggregation
herding=wandering+surrounding+flocking
foraging=wandering+dispersion+following +homing+flocking



[34]

Cockroach Neuron / Behaviors



Control and information flow in artificial fish

Perception sensors, focuser, filter

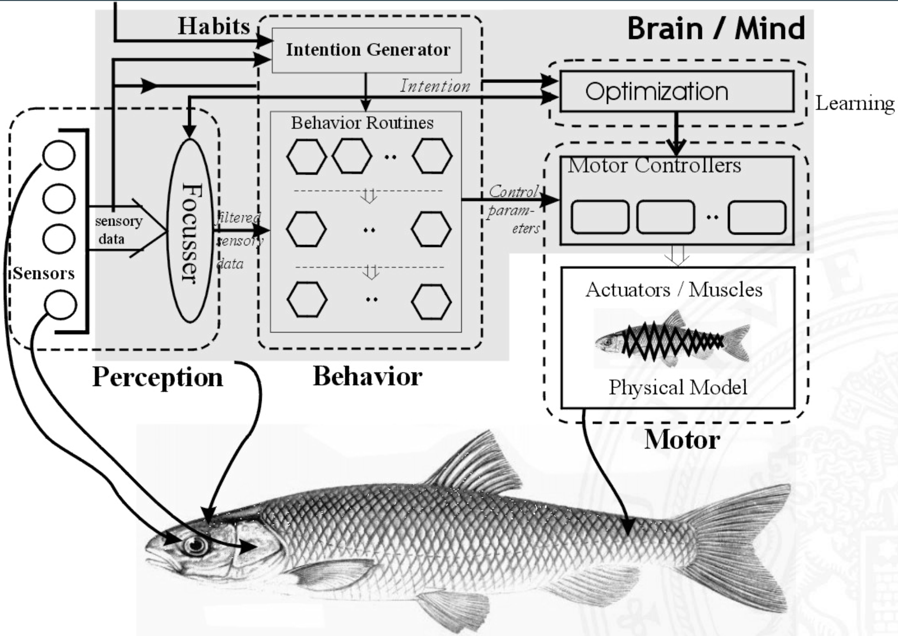
Behaviors behavior routines

Brain/mind habits, intention generator

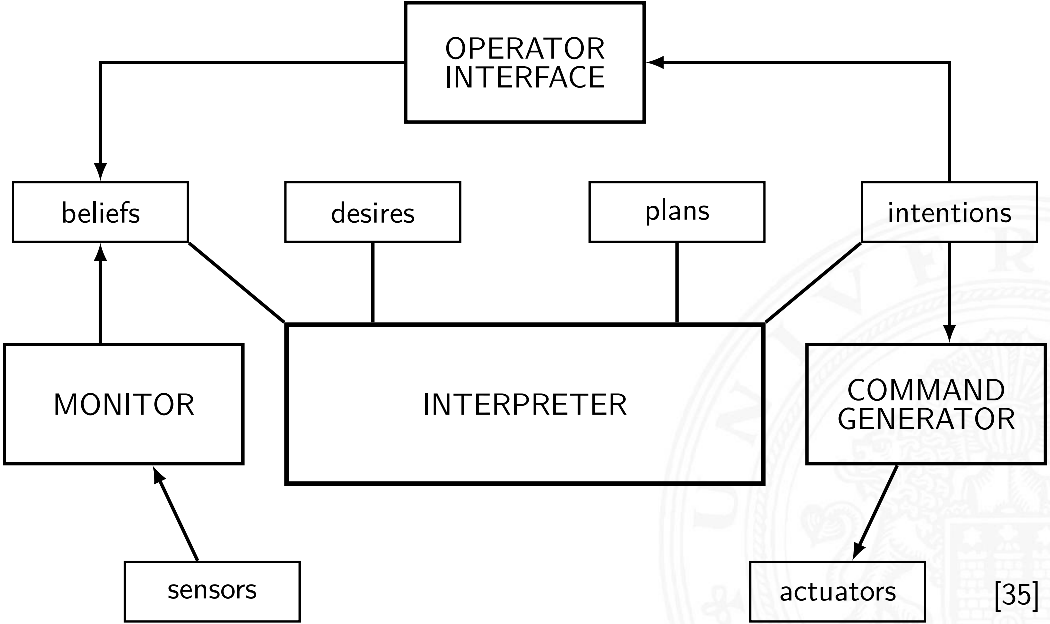
Learning optimization

Motor motor controllers, actuators/muscles

Control Architecture of a Fish (cont.)



Procedural Reasoning System



[35]

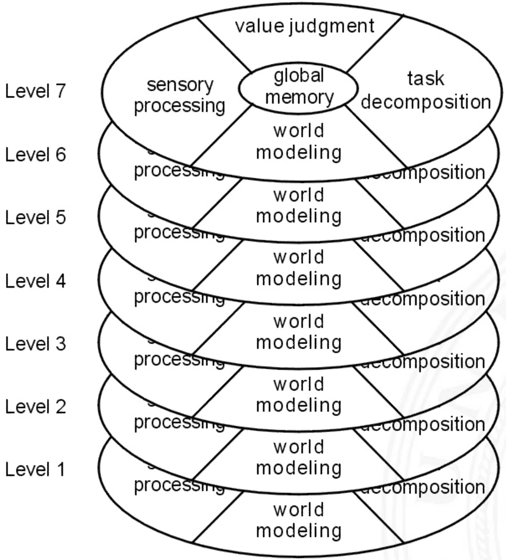
Real-Time Control System (RCS)

- RCS reference model is an architecture for intelligent systems.
- Processing modes are organized such that the BG (Behavior Generation) modules form a command tree.
- Information in the knowledge database is shared between WM (World Model) modules in nodes within the same subtree.

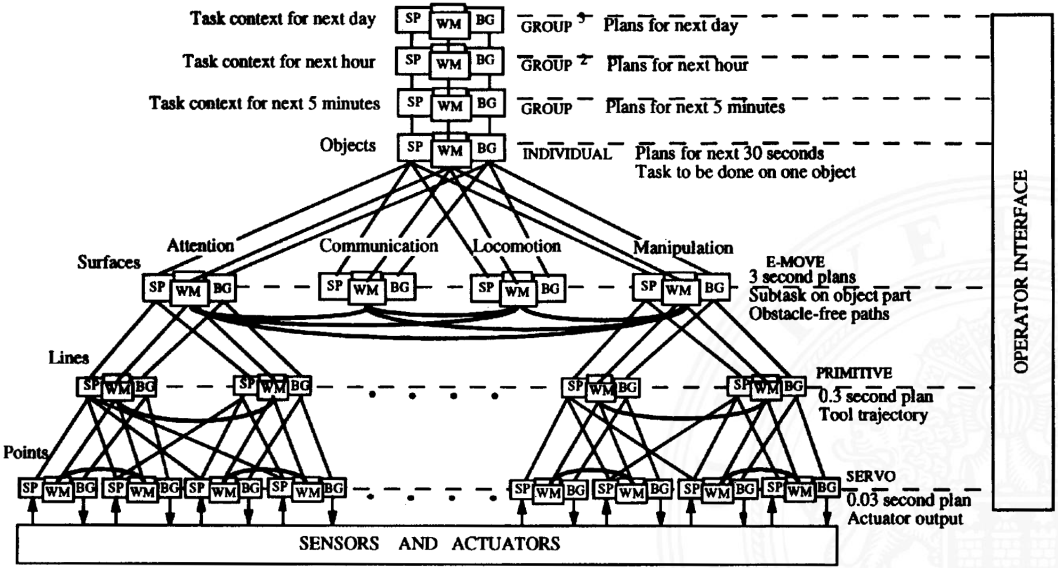
[36]

Examples of functional characteristics of the BG and WM modules:

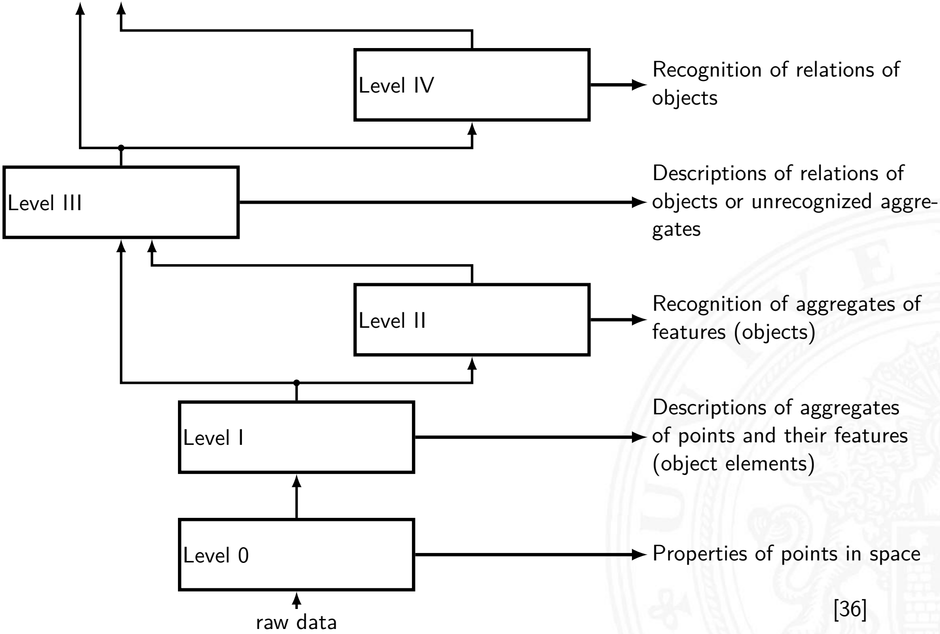
Hierarchy (cont.)



Hierarchy (cont.)

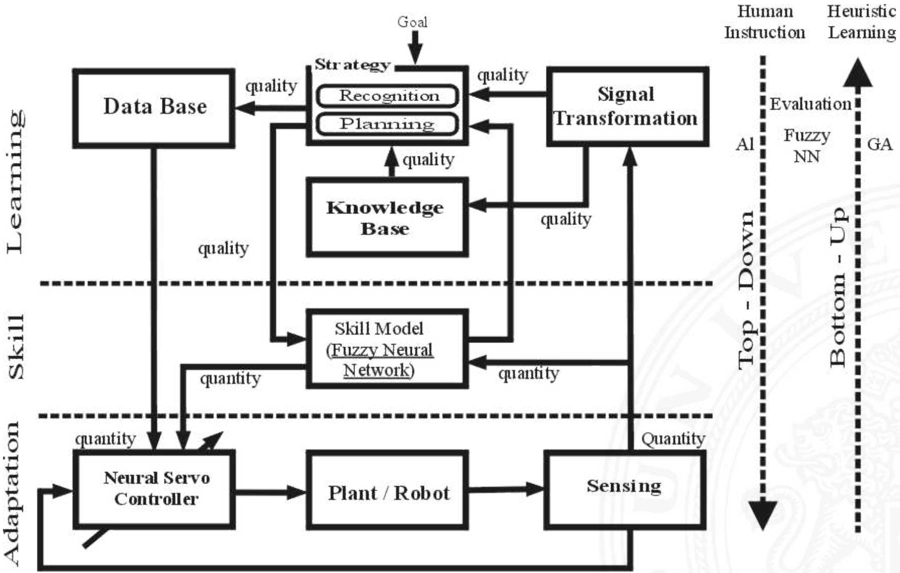


Sensor-Hierarchy



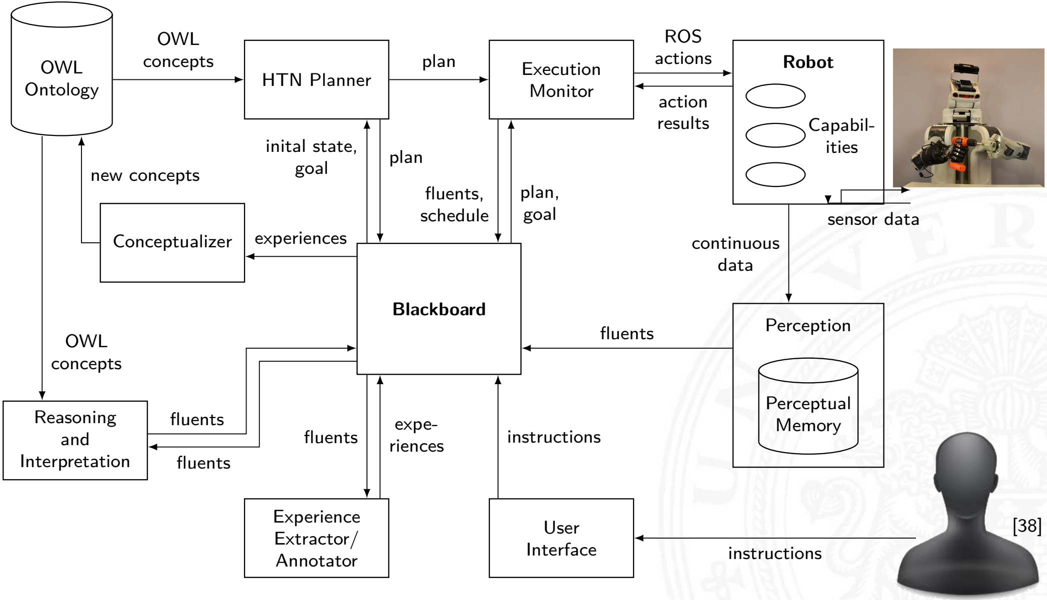
[36]

An Architecture for Learning Robots



[37]

RACE
Robustness by Autonomous Competence Enhancement



Introduction to Robotics

Summary

Shuang Li, Jianwei Zhang
[sli, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

July 11, 2020

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Outline (cont.)

Summary

Conclusion and Outlook

Overall Summary

Introduction

+ Definition;
+ Basic components;
+ DOF;
− Classification

Spatial Description and Transformations

+ Specification of position and orientation;
+ Rotation matrices, their inverse and their operations;
+ Homogeneous transformations;
+ Transformation equations [5, 39, 6, 4];
+ More on presentation of orientation

Forward Kinematics and Robot Description

+ DH-conventions and their applications (classic or modified);
+ Universal Robot Description Format (URDF)

Overall Summary (cont.)

Inverse Kinematics

+ Difference and problems of forward and inverse kinematics;
– Algebraic and geometric solution of inverse kinematics;

Jacobian

+ Differential motion and velocity;
– velocity propagation;
+ Jacobian-matrices;
+ Singularities [5, 39, 6, 4]

Trajectory Generation

+ Tasks and constraints;
+ Trajectory generation methods;
– Polynomial solutions between two and four points;
+ Linear motion in cartesian space and problems;
– Factors of an optimal motion;
+ Concepts and properties of B-Spline interpolation;
– B-Spline basis functions [39, 6, 4, B-Spline Literature]

Overall Summary (cont.)

Dynamics

- + Problems;
- + Newton-Euler equations and Lagrangian Equations;
- – Solution for arms with 1 or 2 joints, multiple joints as excercise;
- + Structure of a dynamical equation [39, 6, 4]

Control

- – Control systems of a PUMA robot;
- – Linear and model-based control;
- + PID controller;
- + Control concepts in Cartesian space [39, 6, 4]

Sensors

- ◦ Classification;
- + Intrinsic sensors, principle and application in control;
- – extrinsic sensors [39, 6, 4]

Overall Summary (cont.)

Path planning

+ Configuration space;
– Object representation;
– Discretized Space Planning;
+ Potential field method;
+ Probabilistic approaches;
+ Rapidly-exploring Random Trees;
– Task and Manipulation Planning

Control architectures

– Subsumption;
– CMAC;
– Hierarchical

Additional references: [40, 41, 42, 43]

Summary – Control

- Industrial Robots:
  - position control with PID controllers
  - featuring gravity compensation
- Research:
  - model-based control
  - hybrid force-position control
  - `under-actuated control`
  - `backwards controllable` (`direct drive`, `artificial muscle`) structure
  - external-sensor based control
    $\rightarrow$ Intelligent Robots/Applied Sensor Technology

Summary – Mechanical Structures of Robots

Things we talked about

- Open chain of rotational joints
- Hybrid joints for rotational and translational motion (SCARA)
- Mobile robots, running machines

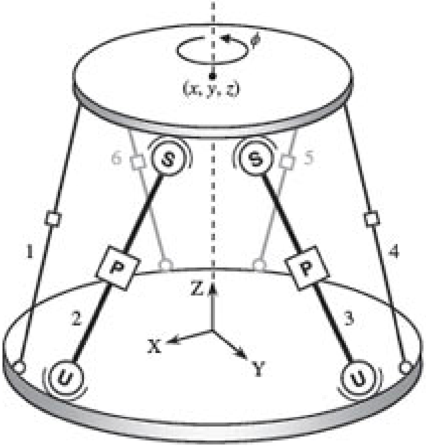
Things we did not talk about

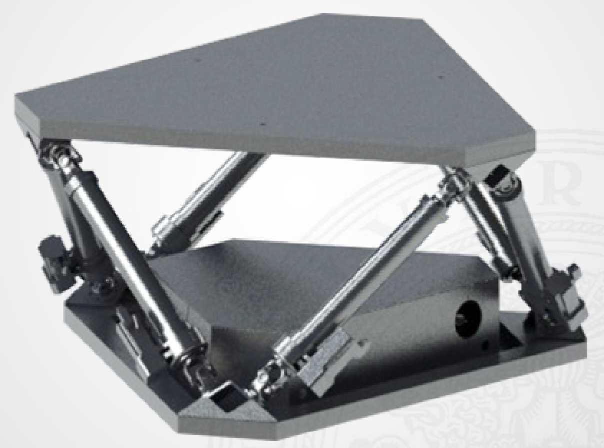
- Closed chain, including Steward Mechanism [39, p. 279]
- Drive without motors (micro- and biomimetic-robots)

The Stewart-Platform

- Tool plate mounted to base plate with six translational joints (usually hydraulic) called `leg`
- `Legs` are connected to the plates with universal joints
- Mathematically 6-DOF configuration space without singularities
- Parallel mechanism provides high payload
  - Sequential manipulator applies forces and torques unequally

The Stewart-Platform (cont.)





Summary – Algorithms

- Transformations
- Forward and inverse kinematics
- Trajectory generation (e.g. linear Cartesian trajectory)
- Approximated representation of robot joints and objects
- Search algorithms
- Further path planning algorithms
- Sensor fusion
- Vision
  - detection (static, dynamic)
  - reconstruction of position and orientation
- Action planning
- Sensor guided motion

Outline

Introduction

Spatial Description and Transformations

Forward Kinematics

Robot Description

Inverse Kinematics for Manipulators

Instantaneous Kinematics

Trajectory Generation 1

Trajectory Generation 2

Dynamics

Robot Control

Path Planning

Task/Manipulation Planning

Telerobotics

Architectures of Sensor-based Intelligent Systems

Outline (cont.)

Summary

Conclusion and Outlook

Underlying robot-technique as described, additionally:

External Recognition
- Reliable measurements of the environment;
- Scene interpretation

Knowledge base
- About environment;
- Its own state;
- Everyday knowledge comparable to a human

Autonomous planning
- Action;
- Coarse motion;
- Grasping;
- Sensor data acquisition

Intelligent Robots (cont.)

Human friendly interface

- Understanding of naturally spoken commands;
- Generation of robot actions;
- Solving of disambiguity in context-aware situations

Adaptive Control

- Evolution instead of programming;
- Ability to learn

Autonomous Planning Systems

Action Planning

- Task-Specification;
- State representation;
- Task-decomposition;
- Action-sequence generation

Motion Planning

- Representation of the robot and the environment;
- Calculation and representation of configuration space;
- Search algorithms

Planning of Sensing

- Which sensors;
- Which time intervals;
- Where to measure;
- Internal and external parameters of the sensor

Sensor driven motion

Goal

Intelligent Control including the ability to adapt to different situations and to react to uncertainties

Control Architecture

Integration of perception, planning and actions

Tasks of sensor data processing

Position detection;
Proximity detection;
Slip detection;
Success confirmation;
Error detection;
Inspection

Sensor driven motion (cont.)

Applied sensors

- Tactile sensors;
- Vision systems;
- Force-torque measurement systems;
- Distance sensors

Strategies

- `calibrated` based on absolute reference values;
- `uncalibrated` based on relative information

Types of perception

- `passive` based on a certain sensor-actor configuration;
- `active` depending on the plan for sensing

Future Commercial Robots

will be:

- dexterous
- smaller
- faster
- lightweight
- powerful
- intelligent
- easier to operate
- cheaper

Challenges in the Field of Robotics

Methods

- Symbolical understanding of the environment;
- Integrated sensor-motor-coupling;
- Self-learning

Systems

- Synergetic multi-sensor;
- Agile mobility;
- Dexterous manipulation capabilities

Technical

- Sensor complexity similar to a human;
- New drive types;
- Nano-robots;
- Multifinger hand;
- Anthropomorphic robots;
- Flying robots

Continuing Education at University of Hamburg

Intelligent Robots Project
: Build a complex robotic system from the available hardware at TAMS. Current Hardware includes PR2, TASER, 2 KUKA lightweight arms, 2 Mitsubishi PA10-6C, UR5 Arm, 4 Turtlebots, Shadow Hand C6, Shadow Hand C5, Robotiq adaptive gripper, SCHUNK gripper, 2 Barret Hands...

Intelligent Robots/Applied Sensor Technology Lecture
: Intrinsic and Extrinsic sensor technology and their application for intelligent robotic systems.

Machine Learning Lecture
: Machine learning techniques allow robots to learn from observation and experience

Neural Networks Lecture
: Neural Networks allow robots to learn and offer new approaches to planning and control

Image Processing I&II Lecture
: Image processing is required for robots to observe the environment and recognize/classify/detect objects and humans

Continuing Education at University of Hamburg (cont.)

Knowledge Processing Lecture
: The gained knowledge from observance and sensing has to be processed efficiently

Language Processing Lecture
: How to extract knowledge and information from human speech

Real-Time Systems Lecture at TUHH
: Robots have to process information and act in Real-Time environments

Fundamentals of Control Technology Lecture at TUHH
: Control Technology is required for the technical control of robotic systems. Advanced Lecture with large prerequisites.

Bibliography

[1] G.-Z. Yang, R. J. Full, N. Jacobstein, P. Fischer, J. Bellingham, H. Choset, H. Christensen, P. Dario, B. J. Nelson, and R. Taylor, "Ten robotics technologies of the year," 2019.

[2] J. K. Yim, E. K. Wang, and R. S. Fearing, "Drift-free roll and pitch estimation for high-acceleration hopping," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8986–8992, IEEE, 2019.

[3] J. F. Engelberger, *Robotics in service*.
MIT Press, 1989.

[4] K. Fu, R. González, and C. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*.
McGraw-Hill series in CAD/CAM robotics and computer vision, McGraw-Hill, 1987.

[5] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control: the Computer Control of Robot Manipulators*.
Artificial Intelligence Series, MIT Press, 1981.

[6] J. Craig, *Introduction to Robotics: Pearson New International Edition: Mechanics and Control*.
Always learning, Pearson Education, Limited, 2013.

[7] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[8] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2009.

[9] W. Böhm, G. Farin, and J. Kahmann, "A Survey of Curve and Surface Methods in CAGD," *Comput. Aided Geom. Des.*, vol. 1, pp. 1–60, July 1984.

[10] J. Zhang and A. Knoll, "Constructing Fuzzy Controllers with B-spline Models - Principles and Applications," *International Journal of Intelligent Systems*, vol. 13, no. 2-3, pp. 257–285, 1998.

[11] M. Eck and H. Hoppe, "Automatic Reconstruction of B-spline Surfaces of Arbitrary Topological Type," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), pp. 325–334, ACM, 1996.

Bibliography (cont.)

[12] A. Cowley, W. Marshall, B. Cohen, and C. J. Taylor, "Depth space collision detection for motion planning," 2013.

[13] Hornung, Armin and Wurm, Kai M. and Bennewitz, Maren and Stachniss, Cyrill and Burgard, Wolfram, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, 2013.

[14] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE International Conference on Robotics and Automation*, pp. 625–632, 2009.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] O. Khatib, "The Potential Field Approach and Operational Space Formulation in Robot Control," in *Adaptive and Learning Systems*, pp. 367–377, Springer, 1986.

[17] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[18] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning.," vol. 2, pp. 995–1001, 01 2000.

[19] J. Starek, J. Gómez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2015, 07 2015.

[20] D. Hsu, J. . Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*, vol. 3, pp. 2719–2726 vol.3, 1997.

[21] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124, IEEE, 2019.

[22] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *in Proc. Robotics: Science and Systems*, 2013.

[23] A. T. Miller and P. K. Allen, “Graspit! a versatile simulator for robotic grasping,” *IEEE Robotics Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[24] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp pose detection in point clouds,” *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[25] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical task and motion planning in the now,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, 2011.

[26] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, “Incremental task and motion planning: A constraint-based approach.,” in *Robotics: Science and Systems*, pp. 1–6, 2016.

[27] J. Ferrer-Mestres, G. Francès, and H. Geffner, “Combined task and motion planning as classical ai planning,” *arXiv preprint arXiv:1706.06927*, 2017.

[28] M. Görner, R. Haschke, H. Ritter, and J. Zhang, “MoveIt! Task Constructor for Task-Level Motion Planning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[29] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[30] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[31] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141, IEEE, 2018.

[32] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[33] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, pp. 14–23, Mar 1986.

[34] M. J. Mataric, "Interaction and intelligent behavior.," tech. rep., DTIC Document, 1994.

[35] M. P. Georgeff and A. L. Lansky, “Reactive reasoning and planning.,” in *AAAI*, vol. 87, pp. 677–682, 1987.

[36] J. S. Albus, “The nist real-time control system (rcs): an approach to intelligent systems research,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 157–174, 1997.

[37] T. Fukuda and T. Shibata, “Hierarchical intelligent control for robotic motion by using fuzzy, artificial intelligence, and neural network,” in *Neural Networks, 1992. IJCNN., International Joint Conference on*, vol. 1, pp. 269–274 vol.1, Jun 1992.

[38] L. Einig, *Hierarchical Plan Generation and Selection for Shortest Plans based on Experienced Execution Duration*.
Master thesis, Universität Hamburg, 2015.

[39] J. Craig, *Introduction to Robotics: Mechanics & Control. Solutions Manual*.
Addison-Wesley Pub. Co., 1986.

Bibliography (cont.)

[40] H. Siegert and S. Bocionek, *Robotik: Programmierung intelligenter Roboter: Programmierung intelligenter Roboter*. Springer-Lehrbuch, Springer Berlin Heidelberg, 2013.

[41] R. Schilling, *Fundamentals of robotics: analysis and control*. Prentice Hall, 1990.

[42] T. Yoshikawa, *Foundations of Robotics: Analysis and Control*. Cambridge, MA, USA: MIT Press, 1990.

[43] M. Spong, *Robot Dynamics And Control*. Wiley India Pvt. Limited, 2008.