Introduction to Robotics

Lecture 1

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

General Information

Lecture:	Friday 10:00 c. t. - 12:00 c. t.
Room:	F-334
Web:	`http://tams.inf...burg.de/lectures/`
Name:	Prof. Dr. Jianwei Zhang
Office:	F-330a
E-mail:	`zhang@informatik.uni-hamburg.de`
Consultation:	by arrangement
Secretary:	Tatjana Tetsis
Office:	F-311
Tel.:	+49 40 42883-2430
E-mail:	`tetsis@informatik.uni-hamburg.de`

Exercises /RPC:	Friday 8:00 c. t. - 10:00 c. t. / Friday 10:00 c. t. - 12:00 c. t. (alternating) see website for dates
Room:	F-334/F-304
Web:	`http://tams.inf...burg.de/lectures/`

Name:	Lasse Einig
Office:	F-324
Tel.:	+49 40 42883-2504
E-mail:	`einig@informatik.uni-hamburg.de`
Consultation:	by arrangement (eMail)

General Information (cont.)

- See website for more information

`http://tams.informatik.uni-hamburg.de/lectures/2019ss/vorlesung/itr`

Exercises/RPC

Criteria for Course Certificate:

- min. 50 % of points in the exercises
  - min. 33 % in each exercise
- regular presence in exercises and RPC
- presentation of two (sub-)tasks
- solutions in groups of 2–3
  - no solo submission
  - each member of a group must be able to present the tasks
  - failure to present results in 0 points

Previous Knowledge

- Basics in physics
  - basics of electrical engineering
- Linear algebra
- Elementary algebra of matrices
- Related computer skills
  - git (RPC)
  - Linux (RPC)
  - access to `mafiasi.de` and pool computers
  - Python (RPC and Excercises)
  - Matlab (Excercises, recommended)

Own Hardware

You may use your own laptop for the RPC (but not recommended). If you do, you require a `Ubuntu 16.04` (Live or Virtual Machine) and fully installed `ros-kinetic-desktop-full`

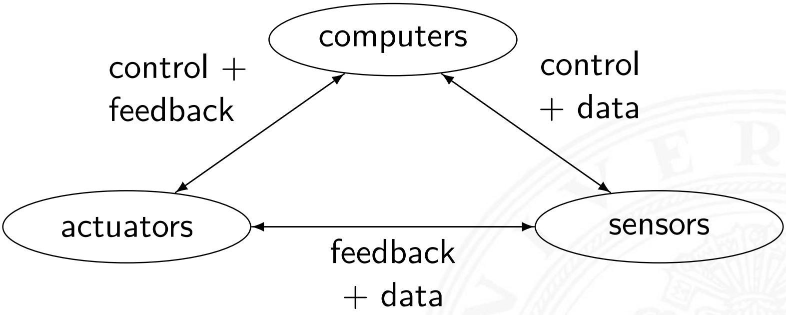
Content

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

Outline

Outline (cont.)

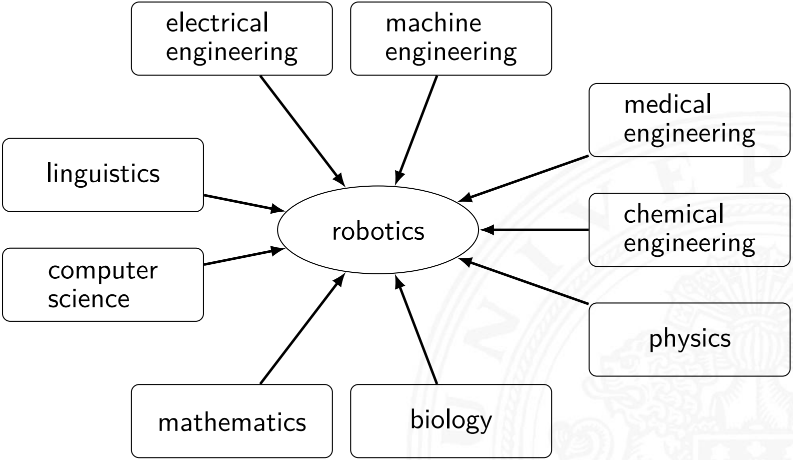
Components of a robot



Robotics

Intelligent combination of computers, sensors and actuators.

An Interdisciplinary Field



Definition of Industry Robots

According to RIA[1], a robot is:

> *...a reprogrammable and multifunctional manipulator, devised for the transport of materials, parts, tools or specialized systems, with varied and programmed movements, with the aim of carrying out varied tasks.*

Intelligent System

Is such a robot also an intelligent system?

[1] Robot Institute of America

Background of Some Terms

Robot became popular through a stage play by Karel Čapek in 1923, being a capable servant.

Robotics was invented by Isaac Asimov in 1942.

Autonomous (literally) (gr.) "living by one's own laws" (*Auto:* Self; *nomos*: Law)

Personal Robot a small, mobile robot system with simple skills regarding vision system, speech, movement, etc. (from 1980).

Service Robot a mobile handling system featuring sensors for sophisticated operations in service areas (from 1989).

Intelligent Robot, Cognitive Robot, Intelligent System ...

Degree of Freedom (DOF)

The number of independent coordinate planes or orientations on which a joint or end-point of a robot can move.
The DOF are determined by the number of independent variables of the control system.

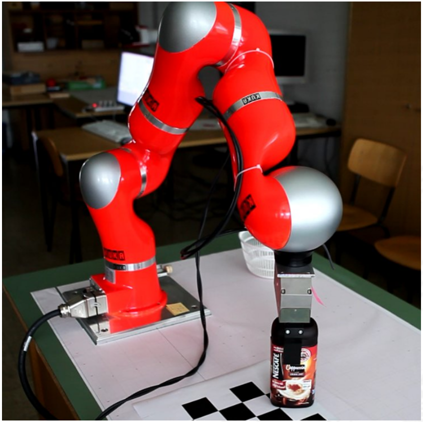
- Point on a line
- Point on a plane
- Point in space
- Rigid body
  - one a surface
  - on a plane
  - in space
- Non-rigid body
- Manipulator
  - number of independently controllable joints
  - a robot should have at least two

DOF Examples

80’s toy robot (Quickshot)

4-DOF + 1-DOF gripper

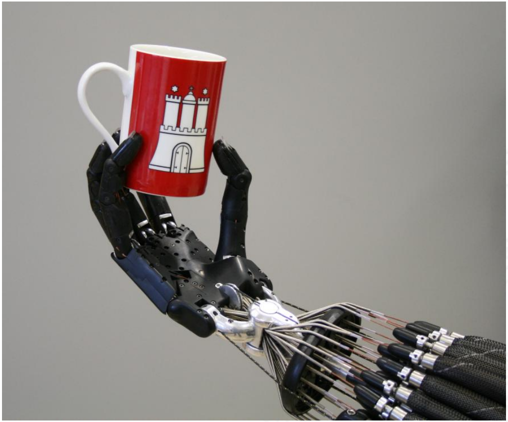
DOF Examples (cont.)



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.)

Boston Dynamics Atlas (2013)

28-DOF

DOF Examples (cont.)

PR2 service robot with Shadow C6 electrical hand

19-DOF + 20-DOF gripper

Robot classification

by input power source

- electrical
- hydraulic
- pneumatic

Robot Classification (cont.)

by field of work

- stationary
  - arms with 2 DOF
  - arms with 3 DOF
  - ...
  - arms with 6 DOF
  - redundant arms ($>$ 6 DOF)
  - multi-finger hand
- mobile
  - automated guided vehicles
  - portal robot
  - mobile platform
  - running machines and flying robots
  - anthropomorphic robots (humanoids)

Robot Classification (cont.)

by type of joint

- translatory
  - linear
  - prismatic
- rotatory
  - revolute
- combinations
  - ball
  - cylindrical
  - polar
  - cartesian

by robot coordinate system

- cartesian
- cylindrical
- spherical / polar
- SCARA
- joint-arm

Robot Classification (cont.)

by usage

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

Robot Classification (cont.)

by intelligence

- manual control
- programmable for repeated movements
- featuring cognitive ability and responsiveness
- adaptive on task level

Robotics is Fun!

- robots move — computers don't
- interdisciplinarity
  - soft- and hardware technology
  - sensor technology
  - mechatronics
  - control engineering
  - multimedia, ...
- A dream of mankind:

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

computers $\Rightarrow$ robots

Literature

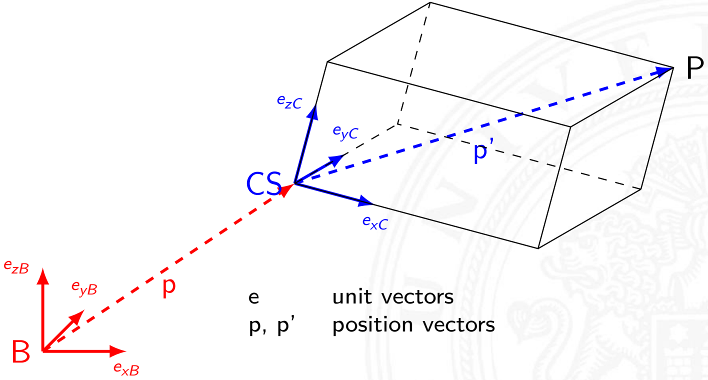
Slides and literature references @
`http://tams.informatik.uni-hamburg.de/lectures/`

- 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

Outline

Outline (cont.)

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

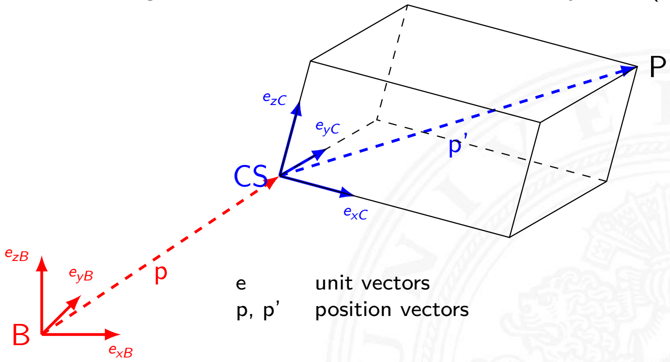


e	unit vectors
p, p'	position vectors

Specification of location and orientation

Position (object coordinates):

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

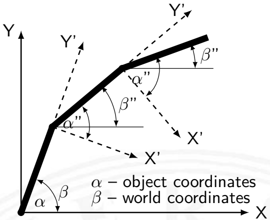


- given by $\mathbf{p} = [p_x, p_y, p_z]^T \in \mathcal{R}^3$

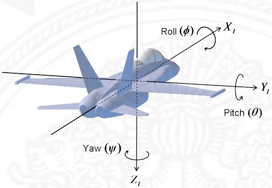
Specification of location and orientation (cont.)

`Orientation` (in space):

- Euler-angles φ, θ, ψ
  - rotations are performed successively around the axes, e. g. $ZY'X''$ or $ZX'Z''$ (12 possibilities!)
  - order depends on reference coordinates
  - object (right), world (left)
  - Gimbal lock!



- Roll-Pitch-Yaw
  - specific case of Euler-angles (used in aviation and maritime)
  - rotation with respect to **object** axes (x – Roll, y – Pitch, z – Yaw)



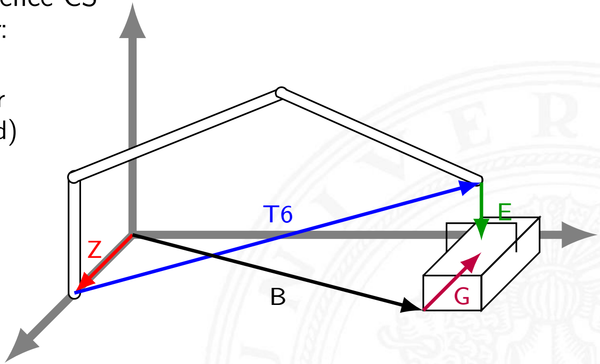
- given by Rotation-matrix $R \in \mathcal{R}^{3\times3}$
  - redundant; 9 parameters for 3 DOF

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

Specification of location and orientation (cont.)

- Position:
  - given through $\vec{p} \in \mathcal{R}^3$
- Orientation:
  - given through projection $\vec{n}, \vec{o}, \vec{a} \in \mathcal{R}^3$ of the axes of the CS to the origin system
  - summarized to rotation matrix $R = \left[\begin{array}{ccc} \vec{n} & \vec{o} & \vec{a} \end{array} \right] \in \mathcal{R}^{3\times3}$
  - redundant, since there are 9 parameters for 3 degrees of freedom
  - other kinds of representation possible, e.g. *roll*, *pitch*, *yaw* angle, quaternions etc.

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



Frame-transformations transform one frame into another.
$^{A}T_{B}$ transforms frame A to frame B (Latex: `$^{A}T_{B}$`)

Homogenous transformation

- Combination of $\vec{p}$ and R to $T = \begin{bmatrix} R & \vec{p} \\ \vec{0} & 1 \end{bmatrix} \in \mathcal{R}^{4\times 4}$
- 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.)

- `Homogeneous transformation matrices:`

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

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.

Translatory transformation

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

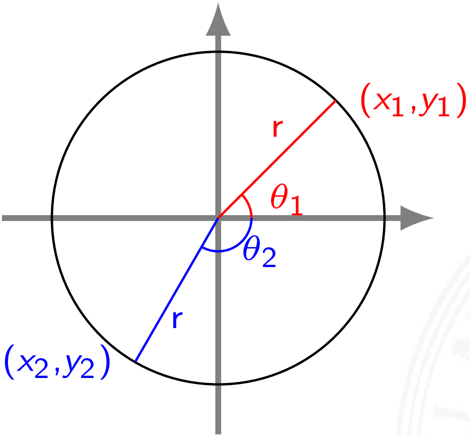
$$H = T_{(p_x, p_y, p_z)} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotatory transformation

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

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

Rotatory transformation (cont.)



Coordinates of a circle

$x = r \sin\theta, \; y = r\cos\theta$

Rotatory transformation (cont.)

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

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

Rotatory transformation (cont.)

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

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

Rotatory transformation (cont.)

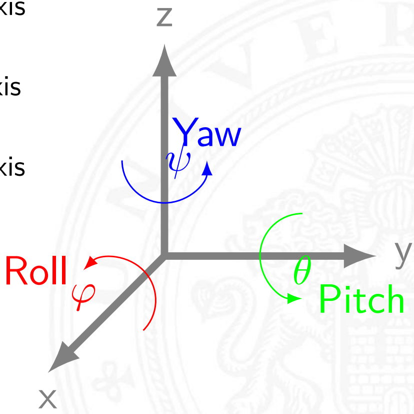
Signs of transformations:

$$R = \begin{bmatrix} + & - & + & 0 \\ + & + & - & 0 \\ - & + & + & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiple rotations

Sequential multiplication of the transformation matrices by order of rotation.

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



Concatenation of rotation matrices

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

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

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

Remark: Matrix multiplication is not commutative:

$$AB \neq BA$$

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

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

Inverse transformation

The inverse of a rotation matrix is simply its transpose:

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

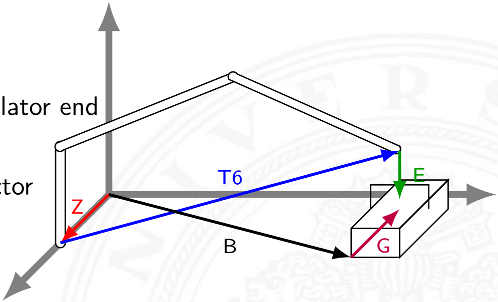
whereas I is the identity matrix.
The inverse of (1) is:

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

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

One has the following transformations:

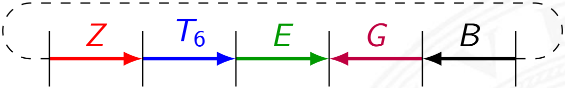
- Z:
  World $\rightarrow$ Manipulator base
- T_6:
  Manipulator base $\rightarrow$ Manipulator end
- E:
  Manipulator end $\rightarrow$ Endeffector
- B:
  World $\rightarrow$ Object
- G:
  Object $\rightarrow$ Endeffector



Transformation equation

There are two descriptions for the desired endeffector position, 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:

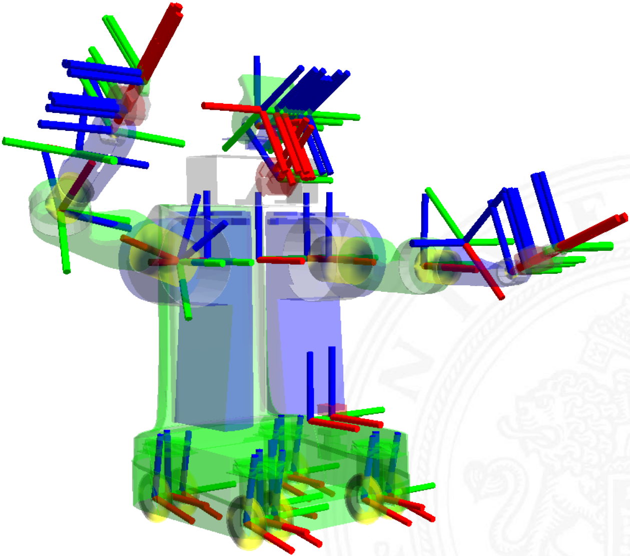
$$T_6 = Z^{-1}\,B\,G\,E^{-1}$$

In order to determine the position of the object:

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

This is also called kinematic chain.

Example: coordinate transformation



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.
- `The depiction` of an object A can be derived from a homogeneous transformation relating to object A'. This is also possible the other way around using inverse transformation.

Summary of homogeneous transformations (cont.)

- `Several translations` and rotations can be multiplied. The following applies:
  - If the rotations / translations 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.
- `A homogeneous transformation` can be segmented into a rotational and a translational part.

Robot kinematics

Reduction to area of interest

For grasping, position and orientation of the robot **gripper** are of interest.

The robot itself is reduced to a single transformation and treated as a solid object.

Coordinates of a manipulator

- `Joint coordinates:`
  A vector $\mathbf{q}(t) = (q_1(t), q_2(t), ..., q_n(t))^T$
  (a robot configuration)
- `Endeffector coordinates`
  (Object coordinates):
  A Vector $\mathbf{p} = [p_x, p_y, p_z]^T$
- `Description of orientations:`
  - Euler angle φ, θ, ψ
  - 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: Denavit-Hartenberg Convention

- `Definition` of one coordinate system per segment $i = 1..n$
- `Definition` of 4 parameters per segment $i = 1..n$
- `Definition` of one transformation A_i per segment $i = 1..n$
- $T_6 = \prod_{i=1}^{n} A_i$

Outlook

Later Denavit Hartenberg Convention will be presented in more detail!

Outlook: Kinematics

- `The direct kinematic problem`:
  Given the joint values and geometrical parameters of all joints of a manipulator, how is it possible to determine the position and orientation of the manipulator's endeffector?
- `The inverse kinematic problem`:
  Given a desired position and orientation of the manipulator's endeffector and the geometrical parameters of all joints, is it possible for the manipulator to reach this position / orientation? If it is, how many manipulator configurations are capable of matching these conditions?

Example

A two-joint-manipulator moving on a plane

Outlook: Kinematics solving

T_6 defines, how the n joint angles are supposed to be consolidated to 12 non-linear formulas in order to describe 6 cartesian degrees of freedom.

- `Forward kinematics` K defined as:
  - $K : \vec{\theta} \in \mathcal{R}^n \rightarrow \vec{x} \in \mathcal{R}^6$
  - Joint angle $\rightarrow$ Position $+$ Orientation
- `Inverse kinematics` K^{-1} defined as:
  - $K^{-1} : \vec{x} \in \mathcal{R}^6 \rightarrow \vec{\theta} \in \mathcal{R}^n$
  - Position $+$ Orientation $\rightarrow$ Joint angle
  - non-trivial, since K is usually not unambiguously invertible

Outlook: Differential movement

Non-linear kinematics K can be linearized through the *Taylor series* $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$.

- `The Jacobian matrix` J as factor for $n = 1$ of the multi-dimensional Taylor series is defined as:
  - $J(\vec{\theta}) : \dot{\vec{\theta}} \in \mathcal{R}^n \rightarrow \dot{\vec{x}} \in \mathcal{R}^6$
  - Joint speed $\rightarrow$ cartesian speed
- `Inverse Jacobian matrix` J^{-1} defined as:
  - $J^{-1}(\vec{\theta}) : \dot{\vec{x}} \in \mathcal{R}^6 \rightarrow \dot{\vec{\theta}} \in \mathcal{R}^n$
  - cartesian speed $\rightarrow$ Joint speed
  - non-trivial, since J not necessarily invertible (e.g. not quadratic)

Outlook: Motion planning

Since T_6 describes only the target **position**, explicit generation of a trajectory is necessary.

Depending on *constraints* different for:

- `joint angle space`
- `cartesian space`

Interpolation through:

- `piecewise straight lines`
- `piecewise polynoms`
- `B-Splines`
- `...`

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

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

Forward kinematics

- Movement depiction of mechanical systems
- Here, only position is addressed
- Translate a series of joint parameter to cartesian position
- Depiction of the mechanical system as fixed body chain
  - Serial robots
- Types of joints
  - rotational joints
  - prismatic joints

Mitsubishi PA10-6C

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

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

Purpose

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

Kinematic equations

- Manipulator is considered as set of links connected by joints
- In each link, a coordinate frame is defined
- A homogeneous matrix ${}^{i-1}A_i$ depicts the relative translation and rotation between two consecutive joints
  - Joint transition

For a manipulator consisting of six joints:

- ${}^{0}A_1$: depicts position and orientation of the first link
- ${}^{1}A_2$: position/orientation of the 2nd link with respect to link 1

$$\vdots$$

- ${}^{5}A_6$: depicts position and orientation of the 6th link in regard to link 5

The resulting product is defined as:

$$T_6 = {}^{0}A_1 \, {}^{1}A_2 \, {}^{2}A_3 \, {}^{3}A_4 \, {}^{4}A_5 \, {}^{5}A_6$$

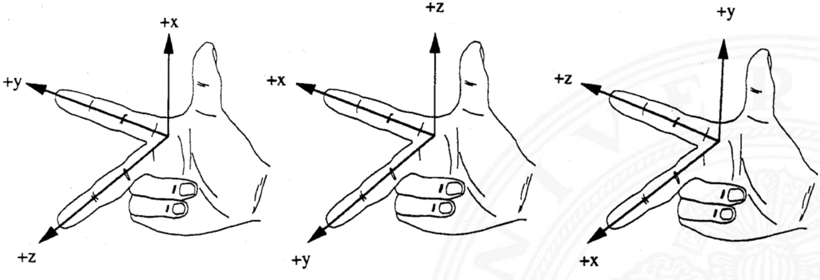
Kinematic description

- Calculation of $T_6 = \prod_{i=1}^{n} A_i$ A_i short for ${}^{i-1}A_i$
  - 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 A_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

Right-Handed Coordinate System

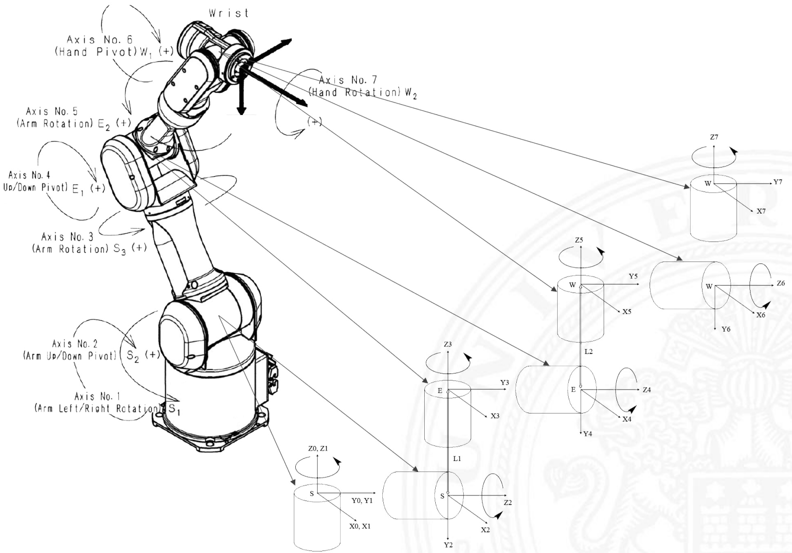


Configuration 1

Configuration 2

Configuration 3

Mitsubishi PA10-7C



Tool Center Point (TCP) description

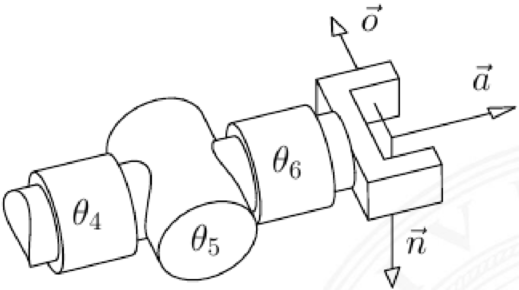
Using a vector $\vec{p}$, the TCP position is depicted.

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:

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

Denavit-Hartenberg convention

- first published by Denavit and Hartenberg in 1955
- established principle
- determination of a transformation matrix A_i using **four** parameter
  - joint length, joint twist, joint offset and joint angle (a_i, α_i, d_i, θ_i)
- complex transformation matrix A_i results from four atomic transformations

Transformation order

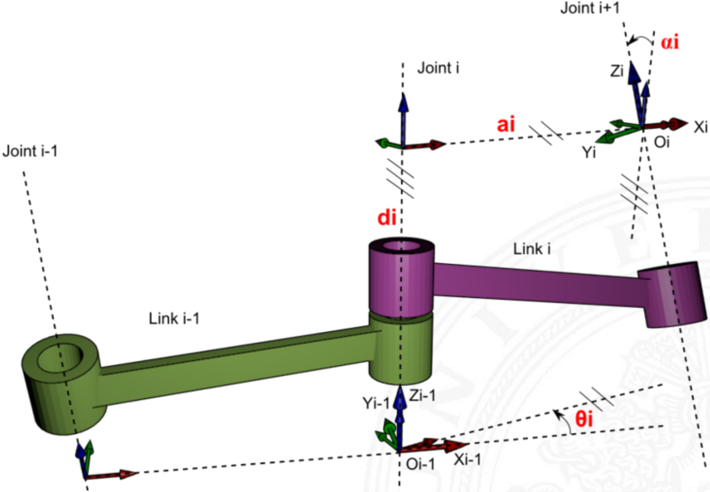
Classic:

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

Modified:

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

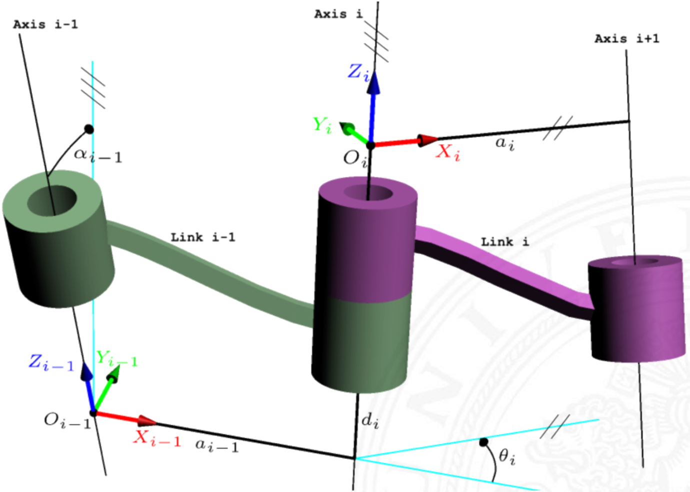
Classic Parameters



Transformation order

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

Modified Parameters



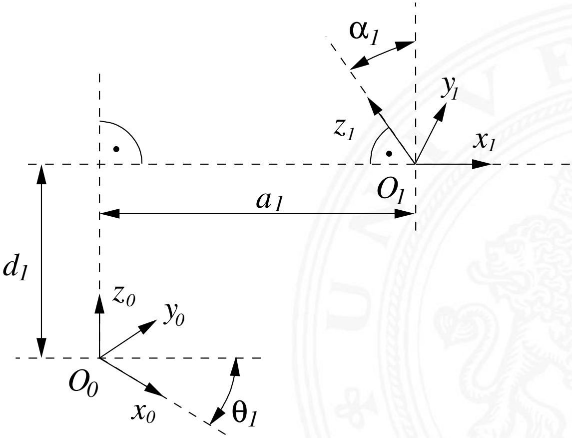
Transformation order

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

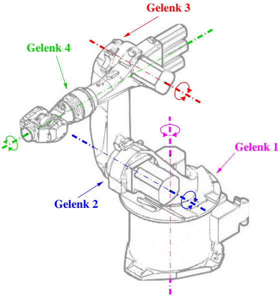
DH-Parameters and -Preconditions (classic)

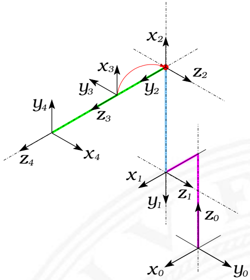
Idea: Determination of the transformation matrix A_i using four joint parameters (a_i, α_i, d_i, θ_i) and two preconditions

- DH$_1$ x_i is perpendicular to z_{i-1}
- DH$_2$ x_i intersects z_{i-1}



Definition of joint coordinate systems (classic)





- 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 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

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:

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

Frame transformation for two links (classic) (cont.)

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

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

Background of DH-convention

- using DH$_1$ $x_1 \cdot z_0 = 0$

$$0 = {}^0x_1 \cdot {}^0z_0 \tag{2}$$

$$0 = \left({}^0A_1 x_1\right)^T \cdot z_0 \tag{3}$$

$$= \left(\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right)^T \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4}$$

$$= \begin{bmatrix} r_{11} & r_{21} & r_{31} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5}$$

$$= r_{31} \tag{6}$$

$$\Longrightarrow \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ 0 & r_{32} & r_{33} \end{bmatrix}$$

- with ${}^{i-1}R_i$ being orthogonal and orthonormal

$$r_{11}^2 + r_{21}^2 = 1 \tag{7}$$
$$r_{32}^2 + r_{33}^2 = 1 \tag{8}$$

$(r_{11}, r_{21}) = (\cos\theta, \sin\theta)$ and
$(r_{32}, r_{33}) = (\sin\alpha, \cos\alpha)$
fulfill the constraint;

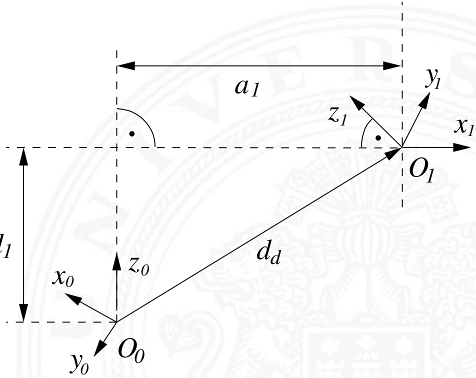
$$\Rightarrow \begin{bmatrix} \cos\theta & r_{12} & r_{13} \\ \sin\theta & r_{22} & r_{23} \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

- r_{12}, r_{13}, r_{22} and r_{23} can complete the rotational matrix

$$\Rightarrow \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix}$$

- with DH$_2$ and DH$_1$:
  the positional vector d_d from O_0 to O_1 may be represented as a linear combination of vectors z_0 and x_1

$$
\begin{aligned}
{}^0d_d &= d\,z_0 + a\ {}^0A_1 x_1 \\
&= d\begin{bmatrix}0\\0\\1\end{bmatrix} + a\ {}^0R_1\begin{bmatrix}1\\0\\0\end{bmatrix} \\
&= d\begin{bmatrix}0\\0\\1\end{bmatrix} + a\begin{bmatrix}cos\theta\\sin\theta\\0\end{bmatrix} \\
&= \begin{bmatrix}a\,cos\theta\\a\,sin\theta\\d\end{bmatrix}
\end{aligned}
$$



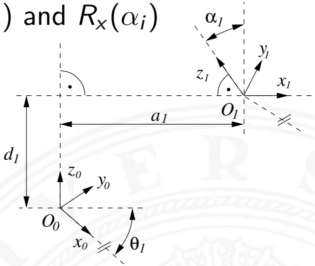
- homogeneous transformation A_i fulfills DH$_2$ and DH$_1$

$$
\begin{aligned}
A_i &= R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i) \\
&= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Partial order of transformation

Calculation of homogeneous transformation matrix A_1 from the partial transformations $R_z(\theta_i)$, $T_z(d_i)$, $T_x(a_i)$ and $R_x(\alpha_i)$

- transition CS_0 to CS_1 using local axes
- invariances
  - T_x invariant to R_x $(T_x\, R_x = R_x\, T_x)$
  - T_z invariant to R_z $(T_z\, R_z = R_z\, T_z)$



- order of transformations
  - rotation around z_1 after rotation around x_0 violates DH_2
  - thus, possible rotation orders which do not violate DH_2 and DH_1:

$$A_i = R_{x_1'''}(\alpha_1) \cdot T_{x_1''}(a_1) \cdot T_{z_0'}(d_1) \cdot R_{z_0}(\theta_1) \tag{9}$$

$$= R_{z_0}(\theta_i) \cdot T_{z_0}(d_i) \cdot T_{x_1}(a_i) \cdot R_{x_1}(\alpha_i) \tag{10}$$

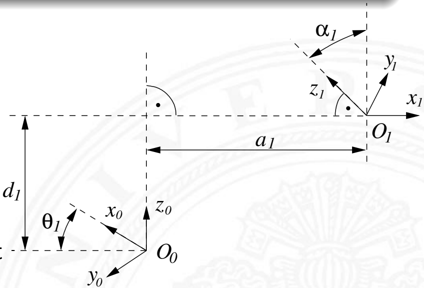
(9) is a possible valid transformation order
(10) is the standard transformation order

Definition of joint coordinate systems: Exceptions

Beware

The Denavit-Hartenberg convention is not unambiguous!

- 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



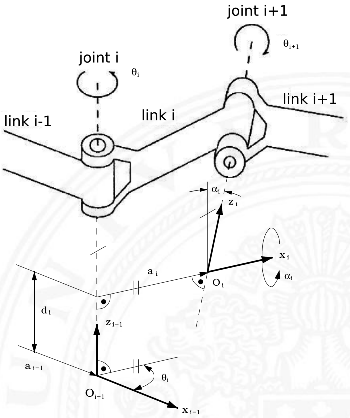
- orientation of CS_n ambigous, as no joint $n+1$ exists
  - x_n must be a normal to z_{n-1}
  - usually z_n chosen to point in the direction of the approach vector $\vec{a}$ of the tcp

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

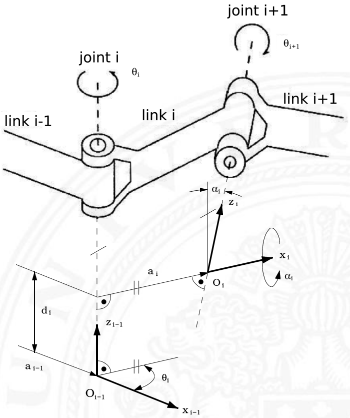
a_i and α_i are constant values due to construction



Parameters for describing two arbitrary links (cont.)

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

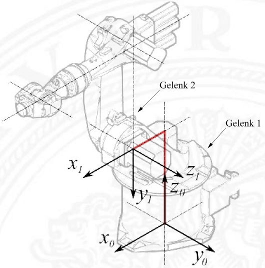
- rotational: θ_i variable, d_i fixed
- translational: d_i variable, θ_i fixed

Example DH-Parameter of a single joint

Determination of DH-Parameter (θ, d, a, α) for calculation of joint transformation: $A_1 = R_z(\theta_1) T_z(d_1) T_x(a_1) R_x(\alpha_1)$

joint angle rotate by θ_1 around z_0, such that x_0 is parallel to x_1

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

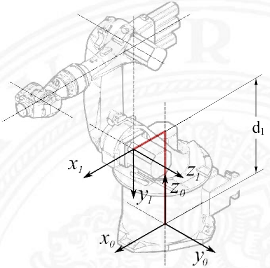


for the shown joint configuration $\theta_1 = 0°$

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

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

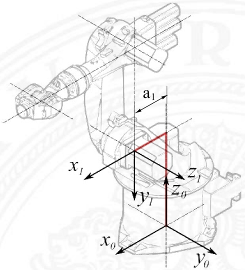
$$T_z(d_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



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

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

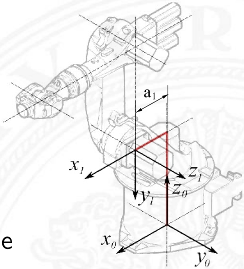
$$T_x(a_1) = \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



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

joint 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, $\alpha_1 = -90°$ due to construction

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

- total transformation of CS_0 to CS_1 (general case)

$$
\begin{aligned}
{}^0A_1 &= R_z(\theta_1) \cdot T_z(d_1) \cdot T_x(a_1) \cdot R_x(\alpha_1) \\
&= \begin{bmatrix} cos\theta_1 & -sin\theta_1 cos\alpha_1 & sin\theta_1 sin\alpha_1 & a_1 cos\theta_1 \\ sin\theta_1 & cos\theta_1 cos\alpha_1 & -cos\theta_1 sin\alpha_1 & a_1 sin\theta_1 \\ 0 & sin\alpha_1 & cos\alpha_1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

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

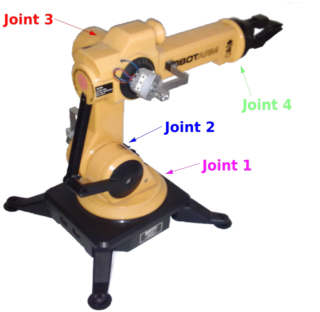
$$
\begin{aligned}
{}^0A_1 &= R_z(\theta_1) \cdot T_z(d_1) \cdot T_x(a_1) \cdot R_x(-90°) \\
&= \begin{bmatrix} cos\theta_1 & 0 & -sin\theta_1 & a_1 cos\theta_1 \\ sin\theta_1 & 0 & cos\theta_1 & a_1 sin\theta_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

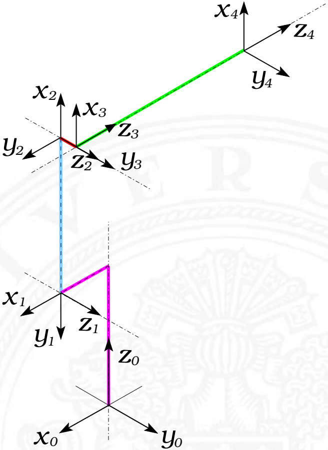
Procedure for predefined structure

- 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 parameter
- 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

$$T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4$$

$$= \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 20\cos\theta_1 \\ \sin\theta_1 & 0 & \cos\theta_1 & 20\sin\theta_1 \\ 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 160\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 160\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ 0 & 1 & 0 & 28 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 250 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1\cos\theta_4(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) - \sin\theta_1\sin\theta_4 & \dots & \dots & \dots \\ \sin\theta_1\cos\theta_4(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) + cos\theta_1\sin\theta_4 & \dots & \dots & \dots \\ -\cos\theta_4(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) & \dots & \dots & \dots \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example featuring PUMA 560

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

$$T_6 = A_1 A_2 A_3 A_4 A_5 A_6$$

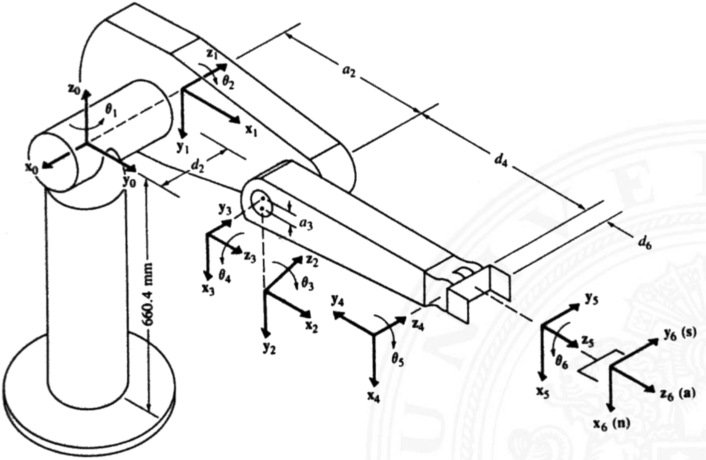
Z: Transformation manipulator base $\rightarrow$ reference coordinate system
E: Manipulator endpoint $\rightarrow$ TCP ("tool center point")
X: The position and orientation of the TCP in relation of the reference coordinate system

$$X = Z T_6 E$$

The following applies as well:

$$T_6 = Z^{-1} X E^{-1}$$

Example featuring PUMA 560 (cont.)



Link Transformations

$$T_6^0 =^0 T_1^1 T_2^2 T_3^3 T_4^4 T_5^5 T_6$$

$$^0T_1 = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & 0 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1T_2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S\theta_2 & -C\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}T_3 = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & a_2 \\ S\theta_3 & C\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}T_4 = \begin{bmatrix} C\theta_4 & -S\theta_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -S\theta_4 & -C\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link Transformations (cont.)

$$
{}^4T_5 = \begin{bmatrix} C\theta_5 & -S\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -S\theta_5 & -C\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^4T_5 = \begin{bmatrix} C\theta_6 & -S\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S\theta_6 & -C\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The solution using the example of PUMA 560

Sum-of-Angle formula

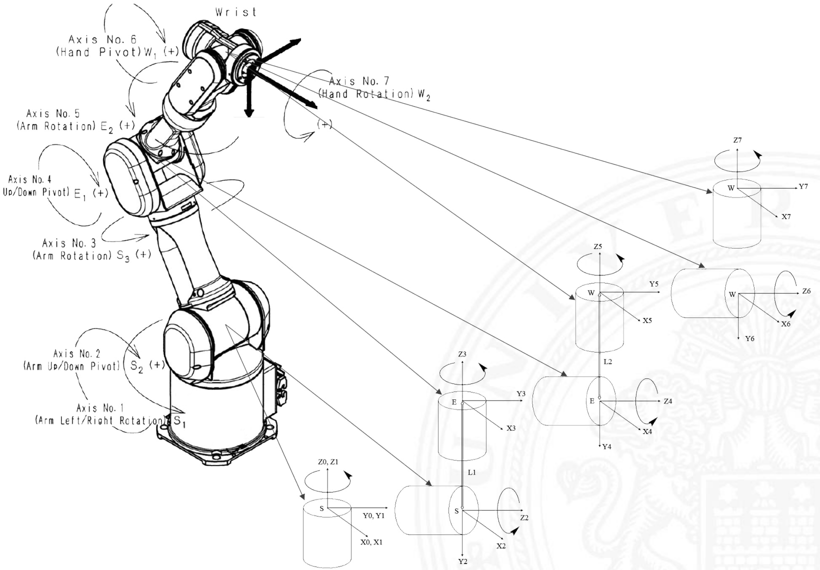
$C_{23} = C_2 C_3 - S_2 S_3,$
$S_{23} = C_2 S_3 + S_2 C_3$

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$
\begin{aligned}
n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_5) - S_{23}S_5C_5] - S_1(S_4C_5C_6 + C_4S_6) \\
n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_1(S_4C_5C_6 + C_4S_6) \\
n_z &= -S_{23}[C_4C_5C_6 - S_4S_6] - C_{23}S_5C_6 \\
&o_x, o_y, o_z = \ldots \\
&a_x, a_y, a_z = \ldots \\
p_x &= C_1[a_2C_2 + a_3C_{23} - d_4S_{23}] - d_3S_1 \\
p_y &= S_1[a_2C_2 + a_3C_{23} - d_4S_{23}] + d_3C_1 \\
p_z &= -a_3S_{23} - a_2S_2 - d_4C_{23}
\end{aligned}
$$

Mitsubishi PA10-7C



Introduction to Robotics

Lecture 3

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

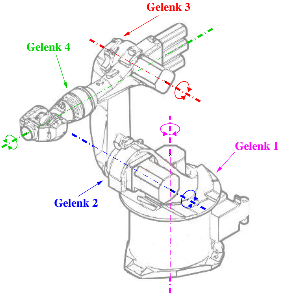
Recapitulation of DH-Parameter

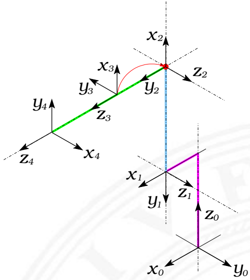
- universal minimal robot description
- based on frame transformations
- four parameters per frame transformation
- serial chain of transformations
- unique description of T_6

Drawbacks

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

Definition of joint coordinate systems





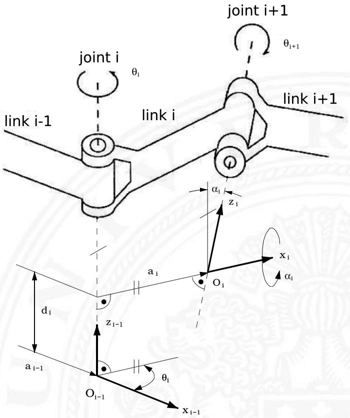
- CS_0 is the stationary origin at the base of the manipulator
- axis z_{i-1} is set along the axis of motion of the i^th joint
- axis x_i is the common normal of $z_{i-1} \times z_i$
- axis y_i concludes a right-handed coordinate system

Parameters for description of two arbitrary links

Two parameters for the description of the link structure i

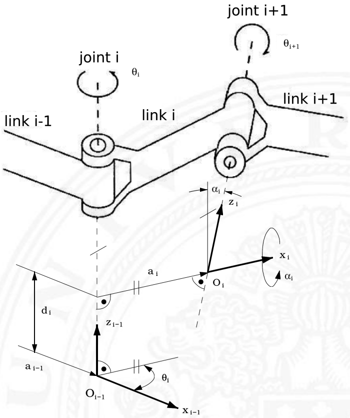
- a_i: shortest distance between the z_{i-1}-axis and the z_i-axis
- α_i: rotation angle around the x_i-axis, which aligns the z_{i-1}-axis to the z_i-axis

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

Universal Robot Description Format

Documentation

http://wiki.ros.org/urdf

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

- robot description format used in ROS[2]
- hierarchical description of components
- XML format representing robot model
  - kinematics and dynamics
  - visual
  - collision
  - configuration

[2] http://ros.org

URDF: Structure

links geometrical properties
- visual
- inertial
- collision

joints geometrical connections
- geometry
- structure
- config

sensors attached sensors

transmissions transmission properties

gazebo simulation properties

model_state robot state

URDF: XML Tree Structure

- 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: XML Tree Structure (cont.)

- 1st-level structure

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

- 2nd-level structure

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

- 3rd-level structure

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

- 4th-level structure

  ⋮

URDF: Link


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

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

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

URDF: Link – visual – primitives

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>
</link>
```

- the `<collision>` element is described identically to the `<visual>` element
- an additional `<collision_checking>` primitive can be used to approximate

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

Inertial tensor describes the dynamic physical properties of the link

- orientation and position of the inertia CS described by `<origin>` tag
- tensor is a symmetric 3×3 matrix
- diagonal values describe main inertial axes `ixx`, `iyy`, `izz`
- `ixy`, `ixz`, `iyz` are 0 for geometric primitives
- rotations around largest and smallest inertial axis are most stable

URDF: Joint


```
<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>
```

URDF: Joint (cont.)

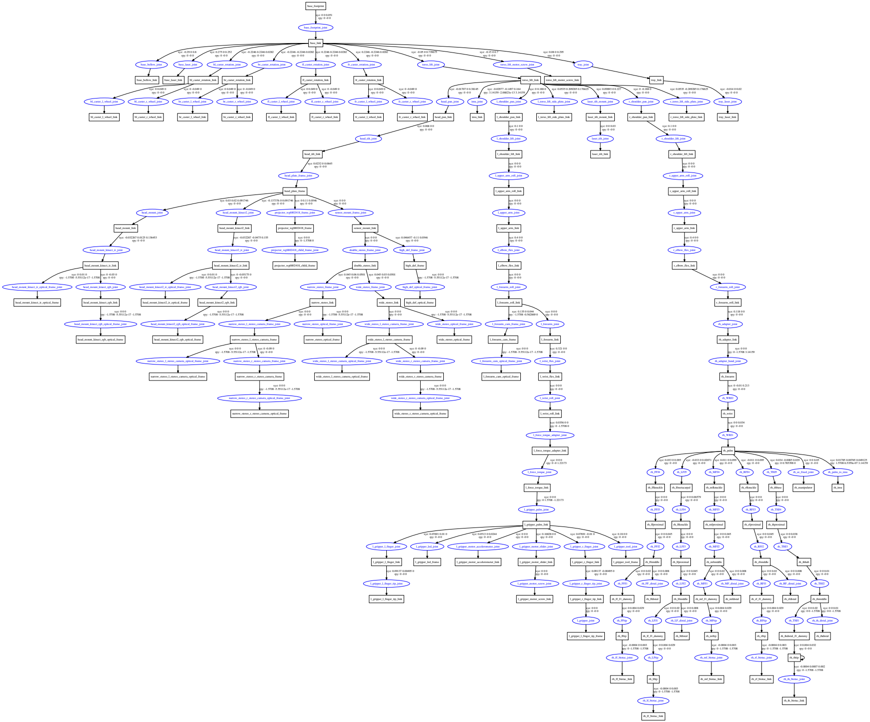
- **type** revolute, continuous, prismatic, fixed, floating, planar
- **parent_link** link which the joint is connected to
- **child_link** link which is connected to the joint
- **axis** joint axis relative to the joint CS. Represented using a normalized vector
- **limit** joint limits for motion (lower, 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.



Outline

Outline (cont.)

Inverse kinematics for manipulators

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*.

For getting a specific tool frame T related to the world, joint values $\theta(t) = (\theta_1(t), \theta_2(t), ..., \theta_n(t))^T$ should be calculated in two steps:

1. Calculation of $T_6 = Z^{-1}BGE^{-1}$;
2. Calculation of $\theta_1, \theta_2, ..., \theta_n$ via T_6.

$\Longrightarrow$ In this case the inverse kinematics is more important than the forward kinematics.

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

$$n_x = C_1[C_{23}(C_4 C_5 C_6 - S_4 S_6) - S_{23} S_5 C_6] - S_1(S_4 C_5 C_6 + C_4 S_6) \tag{11}$$

$$n_y = S_1[C_{23}(C_4 C_5 C_6 - S_4 S_6 - S_{23} S_5 S_6] + C_1(S_4 C_5 C_6 + C_4 S_6) \tag{12}$$

$$n_z = -S_{23}[C_4 C_5 C_6 - S_4 S_6] - C_{23} S_5 C_6 \tag{13}$$

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

$$o_x = \ldots \tag{14}$$

$$o_y = \ldots \tag{15}$$

$$o_z = \ldots \tag{16}$$

$$a_x = \ldots \tag{17}$$

$$a_y = \ldots \tag{18}$$

$$a_z = \ldots \tag{19}$$

$$p_x = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2) \tag{20}$$

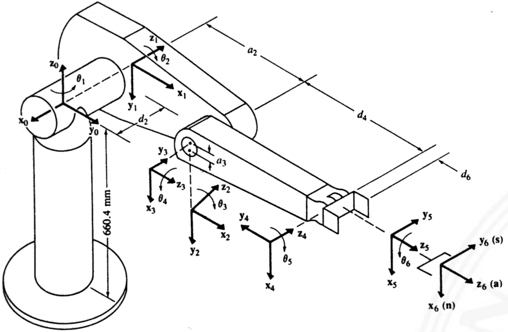
$$p_y = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + s_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2) \tag{21}$$

$$p_z = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2 \tag{22}$$

Remark

- Non-linear equations
- Existence of solutions
  `Workspace`: the volume of space that is reachable for the tool of the manipulator.
  - `dexterous` workspace
  - `reachable` workspace
- Many joint positions produce a similar TCP position using the example of PUMA 560
  - Ambiguity of solutions for $\theta_1, \theta_2, \theta_3$ related to given $\mathbf{p}$.
  - For each solution of $\theta_4, \theta_5, \theta_6$ the alternative solution exists

The PUMA560



$$\theta_4' = \theta_4 + 180^\circ$$
$$\theta_5' = -\theta_5$$
$$\theta_6' = \theta_6 + 180^\circ$$

- `Different solution strategy:` closed solutions vs. numerical solutions

Different methods for solution finding

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

Methods for solution finding

Solvability

"The inverse kinematics for all systems with 6 DOF (translational or rotational joints) in a simple serial chain is always numerical solvable."

Analytical solvability of manipulator

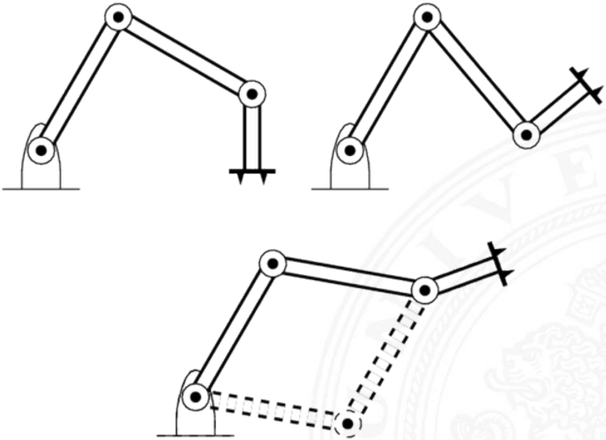
The closed solution exists if specific constraints (sufficient constraints) for the arm geometry are satisfied:

`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)

Example: a planar 3 DOF manipulator



Example: a planar 3 DOF manipulator (cont.)

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

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

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

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

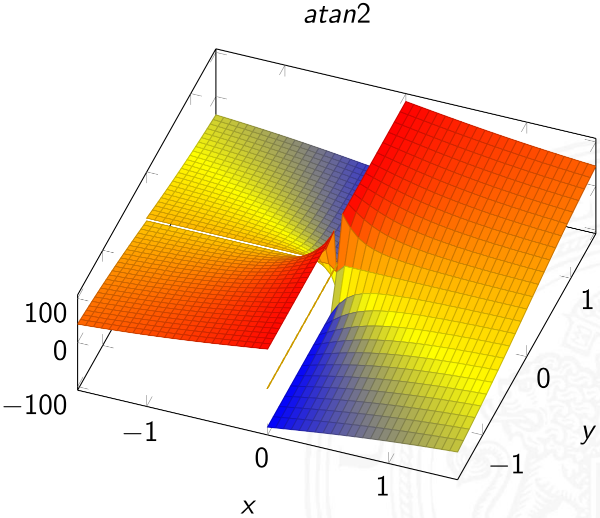
Resultant, four equations can be derived:

$$
\begin{aligned}
C_\phi &= C_{123} && (23) \\
S_\phi &= S_{123} && (24) \\
x &= l_1 C_1 + l_2 C_{12} && (25) \\
y &= l_1 S_1 + l_2 S_{12} && (26)
\end{aligned}
$$

We define the function $atan2_m$ as:

$$\theta = atan2(y,x) = \begin{cases} atan(\frac{y}{x}) & \text{for } +x \\ atan(\frac{y}{x}) + \pi & \text{for } -x, +y_0 \\ atan(\frac{y}{x}) - \pi & \text{for } -x, -y \\ \frac{\pi}{2} & \text{for } x = 0, +y \\ \frac{-\pi}{2} & \text{for } x = 0, -y \\ NaN & \text{for } x = 0, y = 0 \end{cases}$$

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



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

Square and add (25) $(x = l_1 C_1 + l_2 C_{12})$ and (26) $(y = l_1 S_1 + l_2 S_{12})$

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

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 (25) $(x = l_1 C_1 + l_2 C_{12})$ and (26) $(y = l_1 S_1 + l_2 S_{12})$ for θ_1

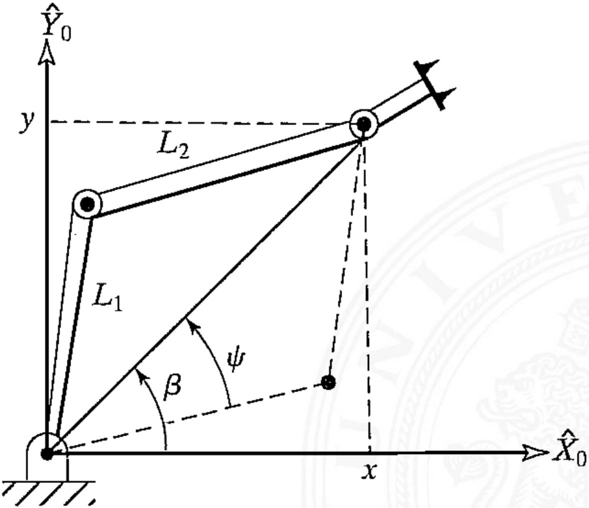
$$\theta_1 = atan2(y, x) - atan2(k_2, k_1)$$

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

solve θ_3 from (23) $(C_\phi = C_{123})$ and (24) $(S_\phi = S_{123})$

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

The geometrical solution for the example 1



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:

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

Algebraical solution (polynomial conversion)

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

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

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

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

Example:
The following transcendental equation is given:

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

After the polynomial conversion:

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

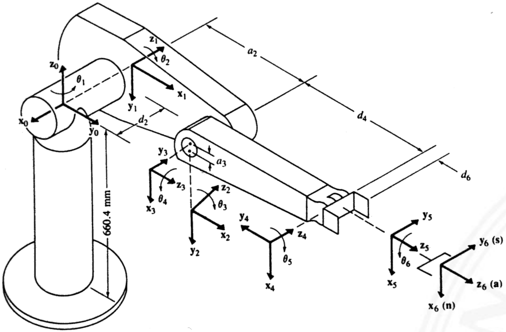
The solution for u:

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

Then:

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

The PUMA560



$$\theta_4' = \theta_4 + 180^\circ$$
$$\theta_5' = -\theta_5$$
$$\theta_6' = \theta_6 + 180^\circ$$

- `Different solution strategy:` closed solutions vs. numerical solutions

Calculation of $\theta_1, \theta_2, \theta_3$:
The first three joint angles $\theta_1, \theta_2, \theta_3$ affect the position of the TCP $(p_x, p_y, p_z)^T$ (in case $d_6 = 0$).

$$p_x = C_1[S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1d_2 \tag{27}$$

$$p_y = S_1[S_{23}d_4 + a_3C_{23} + a_2C_2] + C_1d_2 \tag{28}$$

$$p_z = C_{23}d_4 - a_3S_{23} - a_2S_2 \tag{29}$$

The outcome of this is:

$$\theta_1 = tan^{-1}\left(\frac{\mp p_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_xd_2}{\mp p_x\sqrt{p_x^2 + p_y^2 - d_2^2} + p_yd_2}\right)$$

Algebraic solution using the PUMA 560 (cont.)

$$\theta_3 = tan^{-1}\left(\frac{\mp A_3\sqrt{A_3^2 + B_3^2 - D_3^2} + B_3 D_3}{\mp B_3\sqrt{A_3^2 + B_3^2 - D_3^2} + A_3 D_3}\right)$$

where

$$A_3 = 2a_2 a_3$$
$$B_3 = 2a_2 d_4$$
$$D_3 = p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2 - d_2^2 - d_4^2$$

and

$$\theta_2 = tan^{-1}(\frac{\mp B_2\sqrt{p_x^2 + p_y^2 - d_2^2} + A_2 p_z}{\mp A_2\sqrt{p_x^2 + p_y^2 - d_2^2} + B_2 p_z})$$

where

$$A_2 = d_4 C_3 - a_3 S_3$$
$$B_2 = -a_3 C_3 - d_4 S_3 - a_2$$

J. Craig, *Introduction to Robotics: Pearson New International Edition: Mechanics and Control*. Always learning, Pearson Education, Limited, 2013

The solution for Orientation of PUMA560

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

The solution for following equation is sought:

$$R_{z,\phi}^{-1} T = R_{y,\theta} R_{x,\psi}$$

$$\begin{bmatrix} f_{11}(\mathbf{n}) & f_{21}(\mathbf{o}) & f_{31}(\mathbf{a}) & 0 \\ f_{12}(\mathbf{n}) & f_{22}(\mathbf{o}) & f_{32}(\mathbf{a}) & 0 \\ f_{13}(\mathbf{n}) & f_{23}(\mathbf{o}) & f_{33}(\mathbf{a}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C\theta & S\theta S\psi & S\theta C\psi & 0 \\ 0 & C\psi & -S\psi & 0 \\ -S\theta & C\theta S\psi & C\theta C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$f_{11} = C\phi x + S\phi y$$
$$f_{12} = -S\psi x + C\phi y$$
$$f_{13} = z$$

The solution for Orientation of PUMA560 (cont.)

The equation for $f_{12}(\mathbf{n})$ leads to:

$$-S\phi n_x + C\phi n_y = 0$$

$\Longrightarrow$

$$\phi = atan2(n_y, n_x)$$

and

$$\phi = \phi + 180^\circ$$

The solution with the elements f_{13} and f_{11} are as appropriate:

$$-S\theta = n_z$$

The solution for Orientation of PUMA560 (cont.)

and

$$C\theta = C\phi n_x + S\phi n_y$$

$\Longrightarrow$

$$\theta = atan2(-n_z, C\phi n_x + S\phi a_y)$$

The solution with the elements f_{23} and f_{22} are as appropriate:

$$-S\psi = -S\phi a_x + C\phi a_y$$

$$C\psi = -S\phi o_x + C\phi o_y$$

$\Longrightarrow$

$$\psi = atan2(S\phi a_x - C\phi a_y, -S\phi o_x + C\phi o_y)$$

Solution for arm configurations

Definition of different arm configurations

shoulder	RIGHT-arm, LEFT-arm
elbow	ABOVE-arm, BELOW-arm
wrist	WRIST-down, WRIST-up

Solution for arm configurations (cont.)

Adapted from this following variable can be defined:

$$ARM = \begin{cases} +1 & \text{RIGHT-arm} \\ -1 & \text{LEFT-arm} \end{cases}$$

$$ELBOW = \begin{cases} +1 & \text{ABOVE-arm} \\ 1 & \text{BELOW-arm} \end{cases}$$

$$WRIST = \begin{cases} +1 & \text{WRIST-down} \\ -1 & \text{WRIST-up} \end{cases}$$

The complete solution for the inverse kinematics can be achieved by analysis of such arm configurations.

Technical difficulties for control software

Problem

- Software was hard-coded for a certain robot model / type.
- Software specialized on the robot skills and geometry
- Consequently, the extending and porting software to new hardware was difficult and time consuming

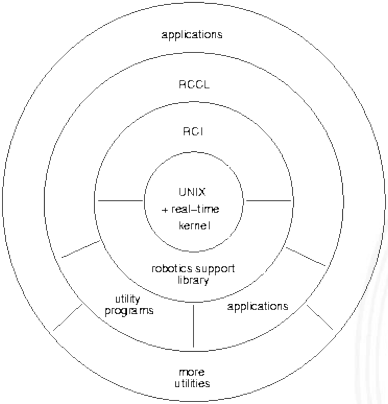
Solution

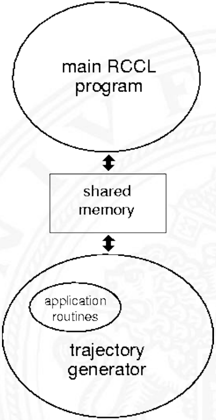
Develop a control software with the following capabilities

- Possibility to control low-level hardware properties
- Maximum portability to different platforms
- Maximum flexibility for fast programming of applications

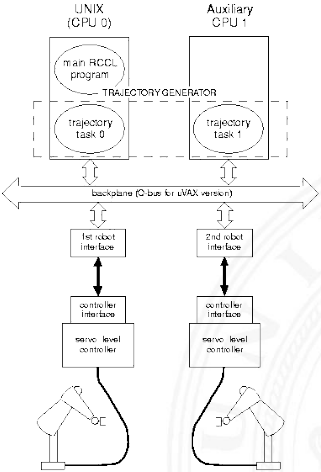
RCCL

Robot Control C Library

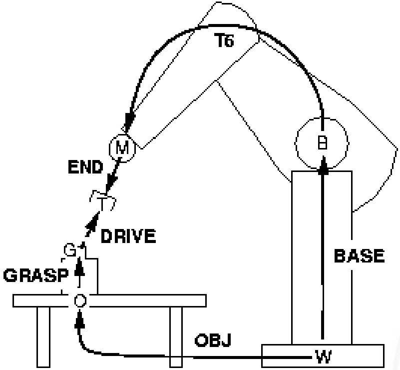


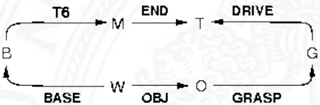


Ability to control multiple robots



Motion description with position equations





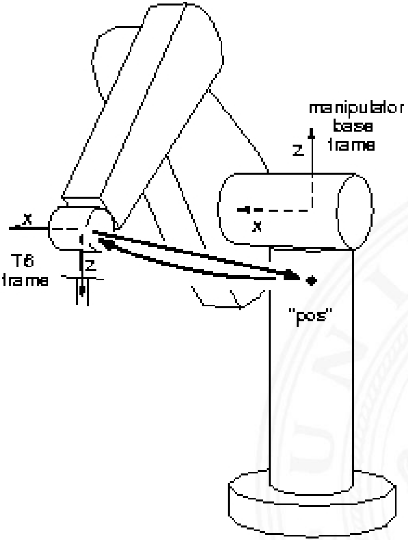
Code sample for robot control in RCCL


```
#include <rccl.h>
#include "manex.560.h"

main()
{
        TRSF_PTR p, t;                                                /*#1*/
        POS_PTR pos;                                                  /*#2*/
        MANIP *mnp;                                                   /*#3*/
        JNTS rcclpark;                                                /*#4*/
        char *robotName;                                              /*#5*/

        rcclSetOptions (RCCL_ERROR_EXIT);                             /*#6*/
        robotName = getDefaultRobot();                                /*#7*/
        if (!getRobotPosition (rcclpark.v, "rcclpark", robotName))
         { printf (''position 'rcclpark' not defined for robot\n'');
           exit(-1);
         }
                                                                      /*#8*/
        t = allocTransXyz ("T", UNDEF, -300.0, 0.0, 75.0);
        p = allocTransRot ("P", UNDEF, P_X, P_Y, P_Z, xunit, 180.0);
        pos = makePosition ("pos", T6, EQ, p, t, NULL);               /*#9*/
```

Code sample for robot control in RCCL (cont.)


```
        mnp = rcclCreate (robotName, 0);                                    /*#10*/
        rcclStart();

        movej (mnp, &rcclpark);                                             /*#11*/

        setMod (mnp, 'c');                                                  /*#12*/
        move (mnp, pos);                                                    /*#13*/
        stop (mnp, 1000.0);

        movej (mnp, &rcclpark);                                             /*#14*/
        stop (mnp, 1000.0);

        waitForCompleted (mnp);                                             /*#15*/
        rcclRelease (YES);                                                  /*#16*/
}
```

Code sample for robot control in RCCL (cont.)



Evolution of Robot Frameworks

Robot Operating System (ROS)

Player/Stage Framework

RCCL

Hard-coded robot software

Introduction to Robotics

Lecture 4

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

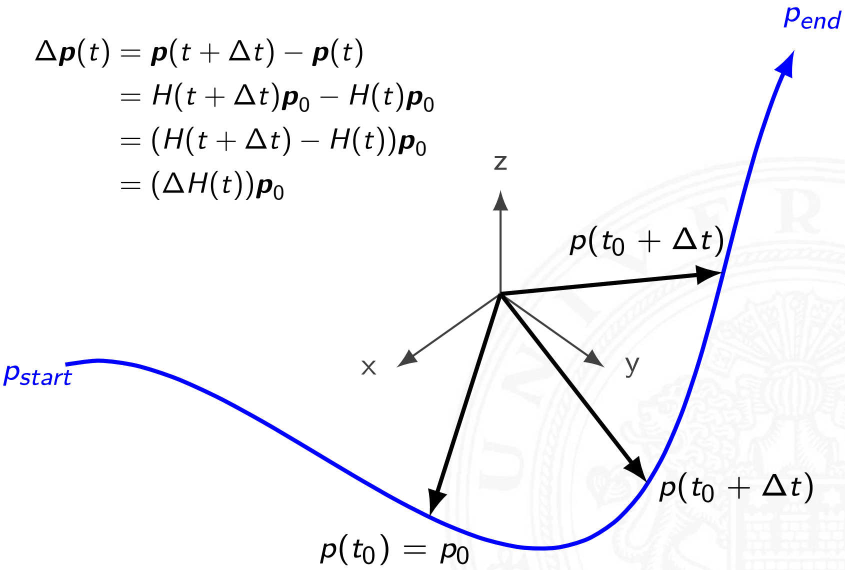
July 12, 2018

Outline

Outline (cont.)

Differential motion

$$
\begin{aligned}
\Delta \boldsymbol{p}(t) &= \boldsymbol{p}(t+\Delta t) - \boldsymbol{p}(t) \\
&= H(t+\Delta t)\boldsymbol{p}_0 - H(t)\boldsymbol{p}_0 \\
&= (H(t+\Delta t) - H(t))\boldsymbol{p}_0 \\
&= (\Delta H(t))\boldsymbol{p}_0
\end{aligned}
$$



Differential motion (cont.)

H is a 4×4 homogeneous transformation from world-frame to object-frame and $\boldsymbol{p}_0$ is given with reference to the world-frame.

Hence it is:

$$\dot{\boldsymbol{p}}(t) = \lim_{\Delta t \to 0} \frac{\Delta \boldsymbol{p}(t)}{\Delta t} \tag{30}$$

$$= \frac{dH(t)}{dt} \mathbf{p_0} \tag{31}$$

$$= \left(\frac{dH(t)}{dt} H^{-1}(t) \right) H(t) \mathbf{p_0} \tag{32}$$

$$= \left(\frac{dH(t)}{dt} H^{-1}(t) \right) \mathbf{p}(t) \tag{33}$$

Derivative of a homogeneous transformation

Consider the homogeneous transformation H

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where each element is a function of a variable t:

$$dH = \begin{bmatrix} \frac{\partial h_{11}}{\partial t} & \frac{\partial h_{12}}{\partial t} & \frac{\partial h_{13}}{\partial t} & \frac{\partial h_{14}}{\partial t} \\ \frac{\partial h_{21}}{\partial t} & \frac{\partial h_{22}}{\partial t} & \frac{\partial h_{23}}{\partial t} & \frac{\partial h_{24}}{\partial t} \\ \frac{\partial h_{31}}{\partial t} & \frac{\partial h_{32}}{\partial t} & \frac{\partial h_{33}}{\partial t} & \frac{\partial h_{34}}{\partial t} \\ 0 & 0 & 0 & 1 \end{bmatrix} dt$$

Differential translation and rotation – World-frame

Case 1 The differential translation and rotation are executed with reference to a fixed coordinate frame.

$$H + dH = Trans_{dx,dy,dz} Rot_{k,d\theta}\ H \tag{34}$$

$Trans_{dx,dy,dz}$: is a differential translation dz, dy, dz with reference to the fixed coordinate frame.

$Rot_{k,d\theta}$: is a differential rotation $d\theta$ around an arbitrary vector $\mathbf{k}$ with reference to the fixed coordinate frame.

dH is calculated as follows:

$$dH = (Trans_{dx,dy,dz} Rot_{k,d\theta} - I)\ H \tag{35}$$

Differential translation and rotation – Object-frame

Case 2 The differential translation and rotation are executed with reference to a current object coordinate frame:

$$H + dH = H \; Trans_{dx,dy,dz} Rot_{k,d\theta} \tag{36}$$

$Trans_{dx,dy,dz}$: is a differential translation dz, dy, dz with reference to the current object coordinate frame.

$Rot_{k,d\theta}$: is a differential rotation $d\theta$ around an arbitrary vector $\mathbf{k}$ with reference to the current object coordinate frame.

dH is calculated as follows:

$$dH = H \left(Trans_{dx,dy,dz} Rot_{k,d\theta} - I \right) \tag{37}$$

Differential homogeneous transformation

Definition

$$\boldsymbol{\Delta} = Trans_{dx,dy,dz} Rot_{k,d\theta} - I$$

Thus (35) can be written as

$$dH = \boldsymbol{\Delta} \cdot H$$

and (37) can be written as:

$$dH = H \cdot \boldsymbol{\Delta}$$

Differential homogeneous transformation (cont.)

The translation by **d** is defined as:

$$Trans_{\boldsymbol{d}} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\boldsymbol{d}$ is a differential vector that represents the differential change

$$d_x \overrightarrow{i} + d_y \overrightarrow{j} + d_z \overrightarrow{k}$$

($\overrightarrow{i}$, $\overrightarrow{j}$, $\overrightarrow{k}$ are three unit vectors coinciding with x, y, z).

The transformation of the rotation with θ around an arbitrary vector $\boldsymbol{k} = k_x \overrightarrow{i} + k_y \overrightarrow{j} + k_z \overrightarrow{k}$ is defined as:

$$Rot_{\boldsymbol{k},\theta} = \begin{bmatrix} k_x k_x V\theta + C\theta & k_y k_x V\theta - k_z S\theta & k_z k_x V\theta + k_y S\theta & 0 \\ k_x k_y V\theta + k_z S\theta & k_y k_y V\theta + C\theta & k_z k_y V\theta - k_x S\theta & 0 \\ k_x k_z V\theta - k_y S\theta & k_y k_z V\theta + k_x S\theta & k_z k_z V\theta + C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{38}$$

where $C\theta = \cos\theta$, $S\theta = \sin\theta$
and $V\theta = \text{versine}\,\theta = 2\sin^2(\frac{\theta}{2}) = 1 - \cos\theta$.

see R. Paul, *Robot Manipulators: Mathematics, Programming, and Control: the Computer Control of Robot Manipulators*. Artificial Intelligence Series, MIT Press, 1981, section 1.12 "General Rotation Transformation"

Differential homogeneous transformation (cont.)

With:

$$\lim_{\theta \to 0} \sin\theta \to d\theta$$

$$\lim_{\theta \to 0} \cos\theta \to 1$$

$$\lim_{\theta \to 0} vers\theta \to 0$$

(38) can be written as:

$$Rot_{\boldsymbol{k},\theta} = \begin{bmatrix} 1 & -k_z d\theta & k_y d\theta & 0 \\ k_z d\theta & 1 & -k_x d\theta & 0 \\ -k_y d\theta & k_x d\theta & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{39}$$

Differential homogeneous transformation (cont.)

$$\boldsymbol{\Delta} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -k_z d\theta & k_y d\theta & 0 \\ k_z d\theta & 1 & -k_x d\theta & 0 \\ -k_y d\theta & k_x d\theta & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

$$= \begin{bmatrix} 0 & -k_z d\theta & k_y d\theta & d_x \\ k_z d\theta & 0 & -k_x d\theta & d_y \\ -k_y d\theta & k_x d\theta & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{41}$$

Differential rotation around the x,y,z axes

Rotation matrices for rotations around x, y and z axis

$$R_{x,\psi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\psi & -S\psi & 0 \\ 0 & S\psi & C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{42}$$

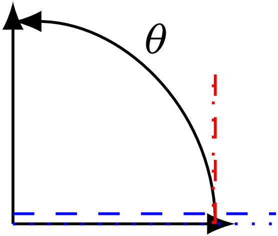
$$R_{y,\theta} = \begin{bmatrix} C\theta & 0 & S\theta & 0 \\ 0 & 1 & 0 & 0 \\ -S\theta & 0 & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{43}$$

$$R_{z,\phi} = \begin{bmatrix} C\phi & -S\phi & 0 & 0 \\ S\phi & C\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{44}$$

Differential rotation around the x,y,z axes (cont.)

Considering the differential change:

$sin\theta \rightarrow \delta\theta$ and
$cos\theta \rightarrow 1$.



$$R_{x,\delta_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\delta_x & 0 \\ 0 & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (45)$$

$$R_{y,\delta_y} = \begin{bmatrix} 1 & 0 & \delta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\delta_y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (46)$$

$$R_{z,\phi} = \begin{bmatrix} 1 & -\delta_z & 0 & 0 \\ \delta_z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (47)$$

Omitting terms of the 2nd order, one gets:

$$R_{z,\delta_z} R_{y,\delta_y} R_{x,\delta_x} = \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{48}$$

Through comparison of (39) with (48) one determines:

$$k_x d\theta = \delta_x \tag{49}$$

$$k_y d\theta = \delta_y \tag{50}$$

$$k_z d\theta = \delta_z \tag{51}$$

Equation (41) can be rewritten as:

$$\mathbf{\Delta} = \begin{bmatrix} 0 & -\delta_z & \delta_y & d_x \\ \delta_z & 0 & -\delta_x & d_y \\ -\delta_y & \delta_x & 0 & d_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Definition of differential transformation

$\mathbf{\Delta}$ is therefore fully defined by the vectors $\boldsymbol{d}$ and $\boldsymbol{\delta}$.

MIN Faculty
Department of Informatics

Introduction to Robotics

Lecture 5

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Technical Aspects of Multimodal Systems

July 12, 2018

Outline

Outline (cont.)

Jacobian of a Manipulator

Definition

- A Jacobian-matrix is a multidimensional representation of partial derivatives.
- The Jacobian of a manipulator links the joint velocities with the cartesian velocity of the TCP.
- The Jacobian matrix depends on the current state of the robot joints.

Jacobian of a Manipulator (cont.)

- Consider an n-link manipulator with joint variables $q_1, q_2, ..q_n$.
- Define $q = [q_1, q_2, ..q_n]^T$
- Let the transformation from base to end-effector frame be:

$$T = \begin{bmatrix} R_n^0(q) & o(q) \\ 0 & 1 \end{bmatrix} \tag{52}$$

- We define ω_n^0 to be the angular velocity of the end-effector
- The linear velocity of the end-effector is v_n^0
- The **Jacobian** matrix consists of two components, that solve the following equations:

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

The manipulator Jacobian

$$J := \begin{bmatrix} J_v \\ J_w \end{bmatrix}$$

We define the body velocity of the endeffector:

$$\xi := \begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} := \begin{bmatrix} {}^T d_x \\ {}^T d_y \\ {}^T d_z \\ {}^T \delta_x \\ {}^T \delta_y \\ {}^T \delta_z \end{bmatrix} \qquad \xi = J\dot{q}$$

Revolute joints

If the i^{th} joint is revolute, the axis of rotation is given by z_{i-1}.

Let $\omega_{i-1,i}^{i-1}$ represent the angular velocity of the link i w.r.t. the frame $i-1$.

Then, we have: $$\omega_{i-1,i}^{i-1} = \dot{q}_i z_{i-1}^{i-1}$$

Prismatic joints

If the i^{th} joint is prismatic, the motion of frame i relative to frame $i-1$ is a translation.

Then, we have: $$\omega_{i-1,i}^{i-1} = 0$$

Overall angular velocity:

$$\omega^0_{0,n} = \omega^0_{0,1} + R^0_1\omega^1_{1,2} + \ldots + R^0_{n-1}\omega^{n-1}_{n-1,n} \tag{53}$$

We get:

$$\omega^0_{0,n} = p_1\dot{q}_1 z^0_0 + p_2\dot{q}_2 R^0_1 z^1_1 + \ldots + p_n\dot{q}_n R^0_{n-1} z^{n-1}_{n-1} \tag{54}$$

$$= p_1\dot{q}_1 z^0_0 + p_2\dot{q}_2 z^0_1 + \ldots + p_n\dot{q}_n z^0_{n-1} \tag{55}$$

where:

$$p_i = \begin{cases} 0 & \text{if i is prismatic} \\ 1 & \text{if i is revolute} \end{cases} \tag{56}$$

Angular Velocity Jacobian (cont.)

The complete Jacobian

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_w \end{bmatrix} \dot{q} \tag{57}$$

The Angular Velocity Jacobian

$$J_w = [p_1 z_0^0 \quad p_2 z_1^0 \quad \dots \quad p_n z_{n-1}^0] \tag{58}$$

(Hint: J_w is a $3xn$ matrix; due to matrix multiplication rules the representation is equal to those on the last slide.)

Linear Velocity Jacobian

The linear velocity of the end effector is: $\dot{o}_n^0$

By the chain rule of differentiation:

$$\dot{o}_n^0 = \frac{\delta o_n^0}{\delta q_1}\dot{q}_1 + \frac{\delta o_n^0}{\delta q_2}\dot{q}_2 + \ldots + \frac{\delta o_n^0}{\delta q_n}\dot{q}_n \tag{59}$$

therefore the linear part of the Jacobian is:

$$J_v = \begin{matrix} \frac{\delta o_n^0}{\delta q_1} & \frac{\delta o_n^0}{\delta q_2} & \ldots & \frac{\delta o_n^0}{\delta q_n} \end{matrix} \tag{60}$$

Linear Velocity Jacobian – Prismatic

Every prismatic joint influences the velocity of the endeffector depending on:

- the current linear velocity of the joint ($\dot{d}_i$)
- the current orientation of the z-axis of the joint (z_{i-1})
  - depending on q

$$\dot{o}_n^0 = \dot{d}_i z_{i-1} \tag{61}$$

Therefore:

$$J_{v_i} = \frac{\delta o_n^0}{\delta q_n} = z_{i-1} \tag{62}$$

Linear Velocity Jacobian – Revolute

Every revolute joint influences the velocity of the end-effector depending on:

- the current angular velocity of the joint ($\dot{q}_i$)
- the current orientation of the z-axis of the joint (z_{i-1})
- the current vector from the joint origin o_{i-1} to the end-effector
  - the two latter depending on q

The linear velocity of the end-effector is of form:

$$\omega \times r$$

with $\omega = \dot{q}_i z_{i-1}$ and $r = o_n^0 - o_{i-1}^0$

Therefore:

$$J_{v_i} = \frac{\delta o_n^0}{\delta q_n} = z_{i-1} \times (o_n^0 - o_{i-1}^0) \tag{63}$$

Computing the final Jacobian

$$J := \begin{bmatrix} J_v \\ J_w \end{bmatrix}$$

$$J_v = \begin{bmatrix} J_{v_1} & J_{v_2} & J_{v_n} \end{bmatrix} \quad \text{with} \tag{64}$$

$$J_{v_i} = \begin{cases} z_{i-1} & \text{if i is prismatic} \\ z_{i-1} \times (o_n^0 - o_{i-1}^0) & \text{if i is revolute} \end{cases} \tag{65}$$

and

$$J_w = \begin{bmatrix} J_{w_1} & J_{w_2} & J_{w_n} \end{bmatrix} \quad \text{with} \tag{66}$$

$$J_{w_i} = \begin{cases} 0 & \text{if i is prismatic} \\ z_{i-1} & \text{if i is revolute} \end{cases} \tag{67}$$

Computing the final Jacobian (cont.)

Target

Compute z_i and o_i.

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

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

Jacobian of a Manipulator – DOF

Consider a Manipulator with 6 DOFs:

$$T_6 = A_1 A_2 A_3 A_4 A_5 A_6$$

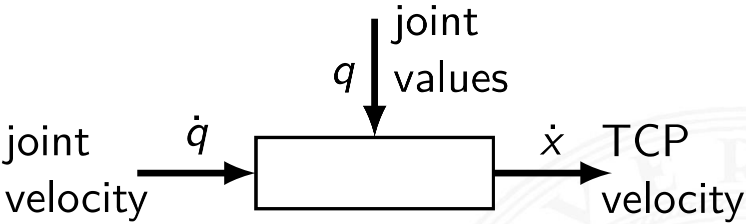
the Jacobian is:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = J_{6\times 6} \begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix}$$

$$\dot{\mathbf{x}} = J(\mathbf{q}) \quad \dot{\mathbf{q}}$$

In case of a 6-DOF manipulator, we get a 6×6 matrix.

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.
Two Main types of Singularities:

- Workspace boundary singularities
- Workspace internal singularities

Singular Configurations – Workarounds

- generally only for 6-DOF manipulators the Jacobian is invertible
- there are workarounds for other types of manipulators

$n < 6$ manually restrict the DOF of the end-effector
$\Longrightarrow$ 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

$n > 6$ use the `pseudoinverse of J`

$$A^+ = (A^T \cdot A)^{-1} \cdot A^T, \text{ linear independent colums} \tag{68}$$

$$A^+ = A^T \cdot (A^T \cdot A)^{-1}, \text{ linear independent rows} \tag{69}$$

Introduction to Robotics

Lecture 6

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

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

Series of discrete poses (TCP or joint configuration)

- usually fixed temporal intervals
- possibly fixed distances, key frames

Trajectory planning

Problem

I am at point A and want move to point B.

- How do I get to point B?
- How long does it take me to get to point B?
- Which constraints exist for moving from A to B?

Solution

- generate a possible trajectory
- trajectory planning
- describe intermediate poses (waypoints)

Requirement

The methods for path generation should be applicable for

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

Primitive solution

Naive approach

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

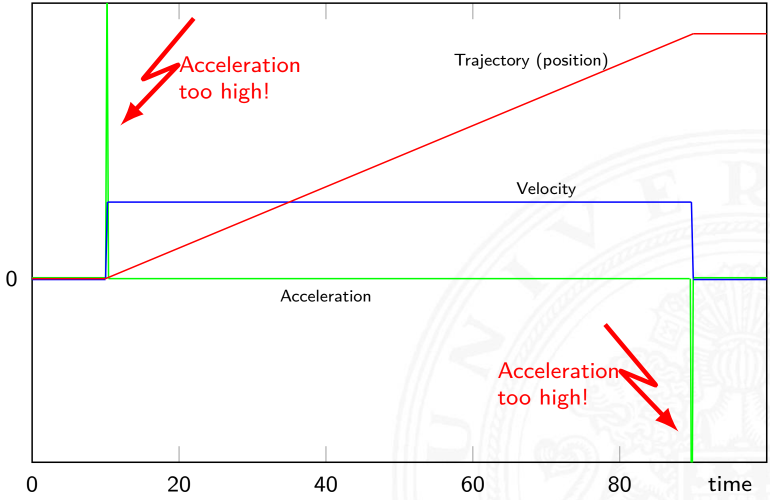
- possible only in simulation
- the moving distance for a manipulator at the next time step may be too large (velocity approaches ∞)

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

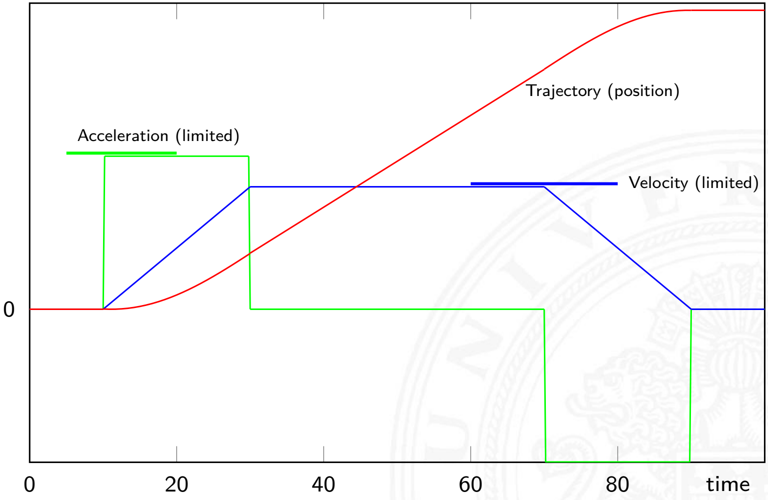
- dynamical trajectory planning
- advanced optimization methods $\rightarrow$ current topic of research

Linear interpolation – improvement

Next best approach

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

Trapezoidal interpolation – visualization



Trapezoidal interpolation

- 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

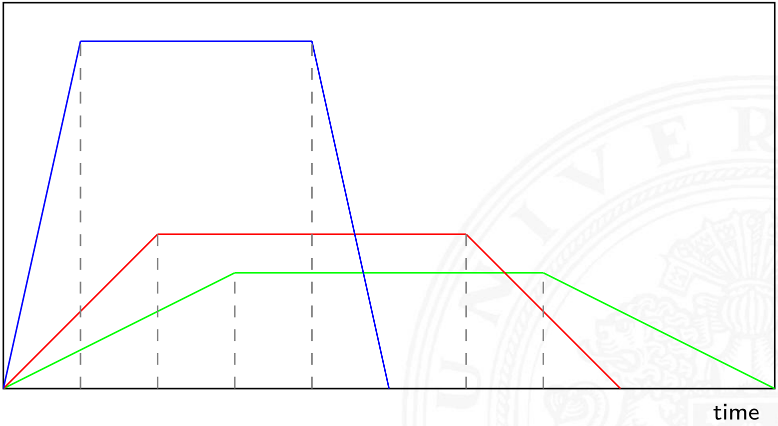
Multidimensional trapezoidal interpolations

- 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.

Solution

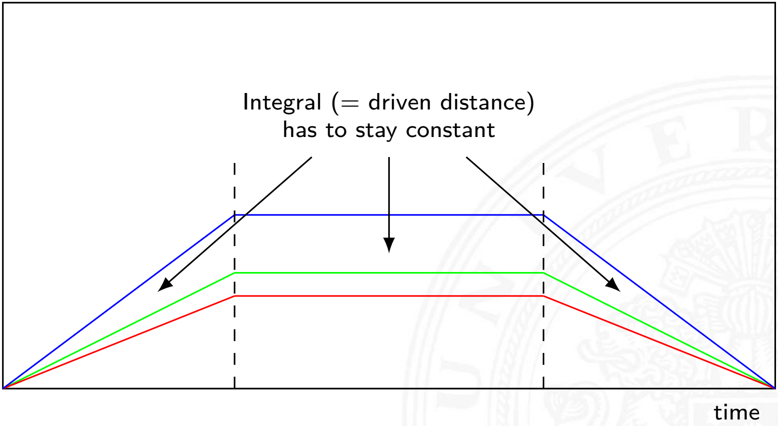
- Normalization to the slowest joint
- Use jerk and arrival time of the slowest joint instead of velocity.

Normalize to the slowest joint



Trapezoidal interpolation – normalization (cont.)

Normalize to the slowest joint



Polynomial interpolation

- Consider velocity and acceleration boundary conditions
  - calculation of extremum and duration of trajectory
- Acceleration differentiable
  - continous 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 polynom 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

Example 5^{th}-degree

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

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

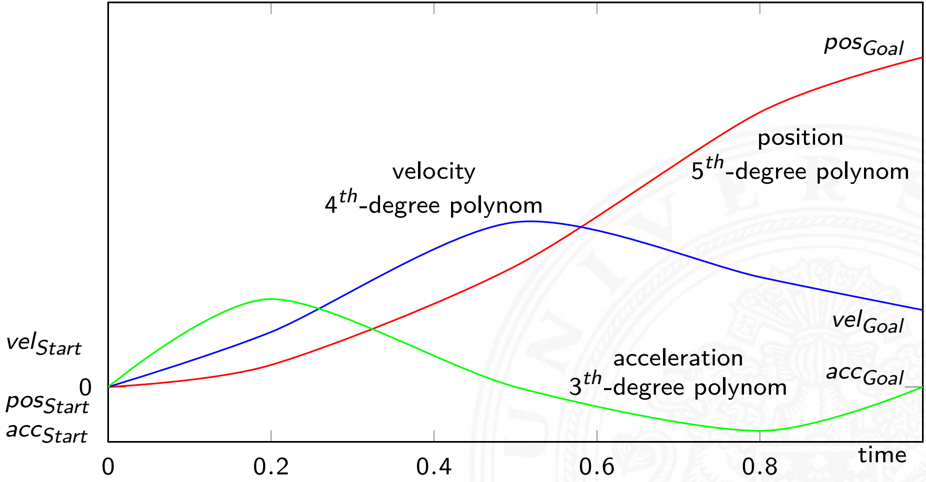
- $f(t_0) = pos_{Start}$, $f(t_d) = pos_{Goal}$
- $f'(t_0) = vel_{Start}$, $f'(t_d) = vel_{Goal}$
- $f''(t_0) = acc_{Start}$, $f''(t_d) = acc_{Goal}$

t: `formal` time from the interval [0;1]

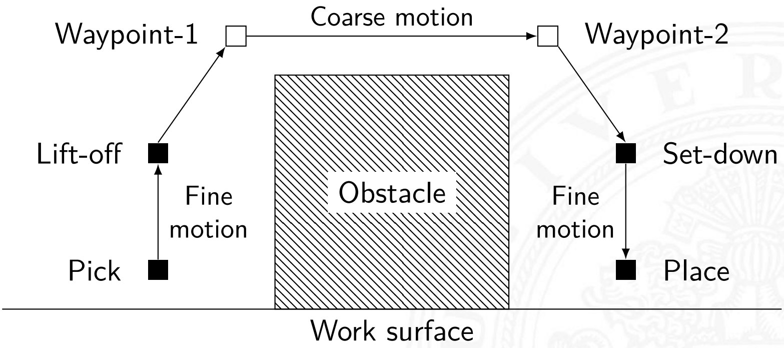
Proper position interpolation from start (A) to goal (B)

$$P(t) = Af(t) + Bf(1 - t)$$

Polynomial interpolation (cont.)



Boundary constraints
Pick-and-Place example



Boundary constraints (cont.)

Pick-and-Place example

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

Generation of trajectories

Task

- find trajectory for moving the robot from start to goal pose
  - calculate
  - interpolate
  - approximate
- use continous functions of time

Solution:

- Cartesian space
- Joint Space

Generation of trajectories (cont.)

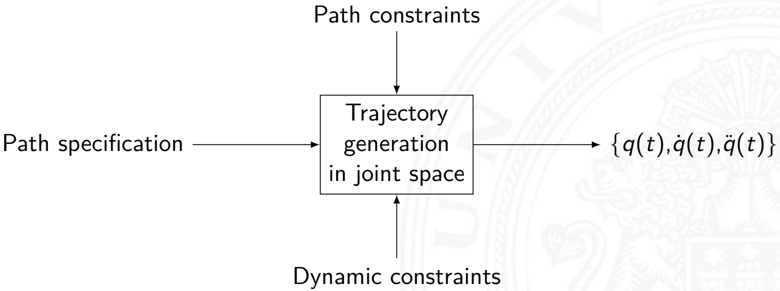
Cartesian space:

- near to the task specification
- advantageous for collision avoidance

Generation of trajectories (cont.)

Joint space:

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



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:

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

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

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

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
  - ...

Cubic polynomials between two configurations

- third-degree polynomial $\Rightarrow$ four constraints:

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

- if the start and end velocity is 0 then

$$\theta(0) = \theta_0 \tag{70}$$

$$\theta(t_f) = \theta_f \tag{71}$$

$$\dot{\theta}(0) = 0 \tag{72}$$

$$\dot{\theta}(t_f) = 0 \tag{73}$$

Cubic polynomials between two configurations (cont.)

- The solution

$$
\begin{aligned}
\text{eq. (70)} \quad & a_0 = \theta_0 \\
\text{eq. (72)} \quad & a_1 = 0 \\
& a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) \\
& a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0)
\end{aligned}
$$

Cubic polynomials with waypoints and velocities

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

$$\theta(0) = \theta_0 \tag{74}$$

$$\theta(t_f) = \theta_f \tag{75}$$

$$\dot{\theta}(0) = \dot{\theta}_0 \tag{76}$$

$$\dot{\theta}(t_f) = \dot{\theta}_f \tag{77}$$

- The solution

$$\text{eq. (74)} \quad a_0 = \theta_0$$

$$\text{eq. (76)} \quad a_1 = \dot{\theta}_0$$

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

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

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

Factors for time optimal motion – Arc Length

If the curve in the *n*-dimensional K 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:

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

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

- trajectory of circle

$$q(t) = c(t) = [r\cos(t), r\sin(t)]^T$$

- arc length L of circle (circumference)

$$L = \int_0^{2\pi} \|\dot{\mathbf{c}}(t)\|_2 \, dt \quad (78)$$

$$= \int_0^{2\pi} \left\|[-r\sin(t), r\cos(t)]^T\right\|_2 dt \quad (79)$$

$$= \int_0^{2\pi} \sqrt{r^2(\sin^2(t) + \cos^2(t))} dt \quad (80)$$

$$= \int_0^{2\pi} r dt \quad (81)$$

$$L = 2\pi\, r \quad (82)$$

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 – Curvature (cont.)

$$\text{with} \quad \kappa(s) = \left| \frac{d\mathbf{U}}{ds} \right| \quad \rightarrow \text{curvature}$$

If the parameter t, the first derivative $\dot{\mathbf{q}} = {}^{d\mathbf{q}(t)}/_{dt}$ and the second derivative $\ddot{\mathbf{q}} = {}^{d\dot{\mathbf{q}}(t)}/_{dt}$ for the curve $\mathbf{q}(t)$ are given, then the *curvature* can be calculated from the following representation

$$\kappa(t) = \frac{|\dot{\mathbf{q}} \times \ddot{\mathbf{q}}|}{|\dot{\mathbf{q}}^3|} = \frac{\left(\dot{\mathbf{q}}^2 \cdot \ddot{\mathbf{q}}^2 - (\dot{\mathbf{q}} \cdot \ddot{\mathbf{q}})^2\right)^{1/2}}{|\dot{\mathbf{q}}|^3}$$

where $\dot{\mathbf{q}} \times \ddot{\mathbf{q}}$ is the cross product and $\dot{\mathbf{q}} \cdot \ddot{\mathbf{q}}$ is the dot product

http://web.cs.iastate.edu/~cs577/handouts/curvature.pdf with $q(t) = c(t) = [r\cos(t), r\sin(t)]^T \rightarrow$ trajectory of a circle

$$\dot{c}(t) = [-r\sin(t), r\cos(t)]^T$$
$$\ddot{c}(t) = [-r\cos(t), -r\sin(t)]^T$$
$$\dot{c}^2(t) = r^2\sin^2(t) + r^2\cos^2(t) = r^2$$
$$\ddot{c}^2(t) = r^2\cos^2(t) + r^2\sin^2(t) = r^2$$
$$\dot{c}(t)\cdot\ddot{c}(t) = r^2\sin(t)\cos(t) - r^2\cos(t)\sin(t) = 0$$

Curvature of a circle

$$\kappa(t) = \frac{\left(\dot{\mathbf{c}}^2 \cdot \ddot{\mathbf{c}}^2 - (\dot{\mathbf{c}}\cdot\ddot{\mathbf{c}})^2\right)^{1/2}}{|\dot{\mathbf{c}}|^3} = \frac{\sqrt{r^4}}{r^3} = \frac{1}{r}$$

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

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

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

$$|\dot{q}_j^i(t)| \leq \dot{q}_{max}^i \tag{83}$$

$$|\ddot{q}_j^i(t)| \leq \ddot{q}_{max}^i \tag{84}$$

$$|m_j^i(t)| \leq m_{max}^i \tag{85}$$

- 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.

Difficulties for cartesian space trajectory generation

- Waypoints cannot be realized
  - workspace boundaries, object collision, self-collision
- Velocities in the vicinity of singular configurations are too high
- Start and end configurations can be achieved, but there are different solutions
  - ambiguous solutions

Motion along a line $< \mathbf{w}_0, \mathbf{w}_1 >$

- The following algorithm should create the smallest set of waypoints in the joint space under a predefined deviation $\epsilon > 0$.
- Therefore the deviation between the trajectory $\mathbf{q}(t)$ and the given line $< \mathbf{w}_0, \mathbf{w}_1 >$ must be smaller than ϵ.

Algorithm(Bounded_Deviation)

1. Calculation of possible configurations $\mathbf{q}_0, \mathbf{q}_1$ from $\mathbf{w}_0, \mathbf{w}_1$ with the help of the inverse kinematics.
2. Calculation of the center in joint space:

$$\mathbf{q}_m = \frac{\mathbf{q}_0 + \mathbf{q}_1}{2}$$

Motion along a line $< \mathbf{w}_0, \mathbf{w}_1 >$ (cont.)

3. Calculation of the corresponding point of $\mathbf{q}_m$ in the workspace with usage of direct kinematics:

$$\mathbf{w}_m = W(\mathbf{q}_m)$$

4. Calculation of the center in the workspace:

$$\mathbf{w}_M = \frac{\mathbf{w}_0 + \mathbf{w}_1}{2}$$

5. If the deviation $||\mathbf{w_m} - \mathbf{w_M}|| \geq \epsilon$, then cancel; else add the $\mathbf{w}_M$ as node point between $\mathbf{w}_0$ and $\mathbf{w}_1$.
6. Recursive application of the algorithm for two new segments $(\mathbf{w}_0, \mathbf{w}_M)$ und $(\mathbf{w}_M, \mathbf{w}_1)$.

Introduction to Robotics

Lecture 7

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Technical Aspects of Multimodal Systems

July 12, 2018

Outline

Outline (cont.)

Trajectory generation – Recapitulation

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.)

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.

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

Definition

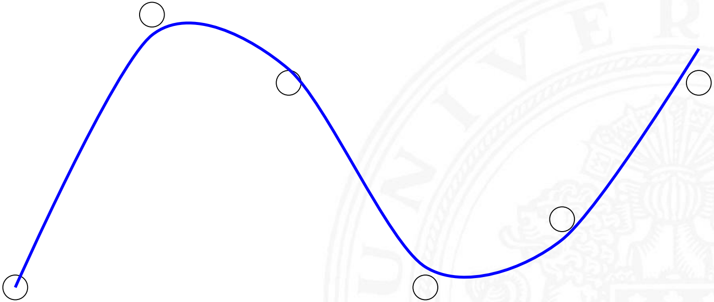
Interpolation is the process of constructing new data points within the range of a discrete set of known data points.

- 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

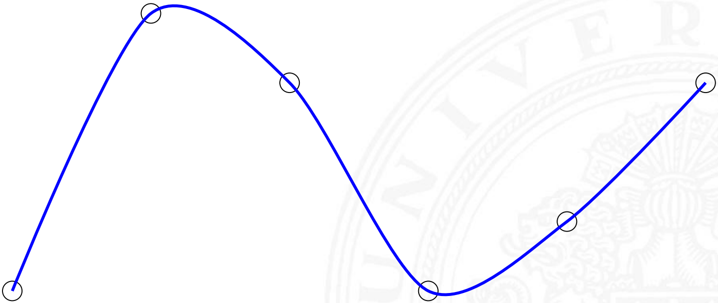
- Approximation: Fitting a curve to given data points.
  - Online tool: `https://mycurvefit.com/`
- Interpolation: Defining a curve exactly through all given data points
  - In the case of many, especially noisy, data points, approximation is often better suited than interpolation

- Approximation of the relation between x and y (curve, plane, hyperplane) with a different function, given a limited number n of data points $D = \{\mathbf{x}_i, y_i\}_{;\, i \in \{1 \ldots n\}}$.

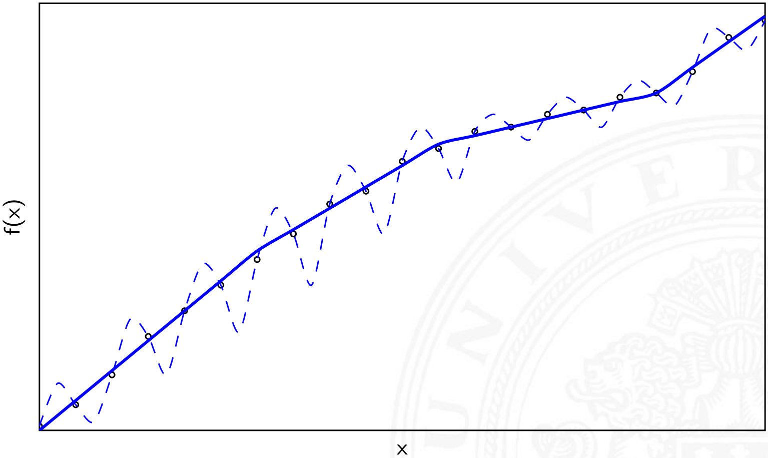


Interpolation vs. Approximation (cont.)

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



Approximation without Overfitting



Overfitting example

Complete the sequence: $1, 3, 5, 7,$ **?**

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
- Oszillation
  - high-degree polynomials may oszillate due to many extrema
  - workaround: composition of sub-polynomials

Interpolation methods

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

Some interpolation methods using polynomials:

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

Examples of polynomials interpolation can be found at

- 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-Polynomials of degree k are defined as:

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

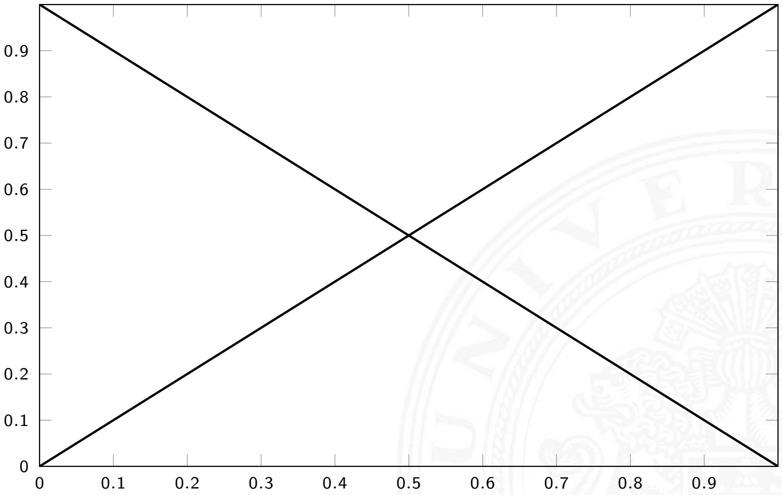
Interpolation with $B_{i,k}$:

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

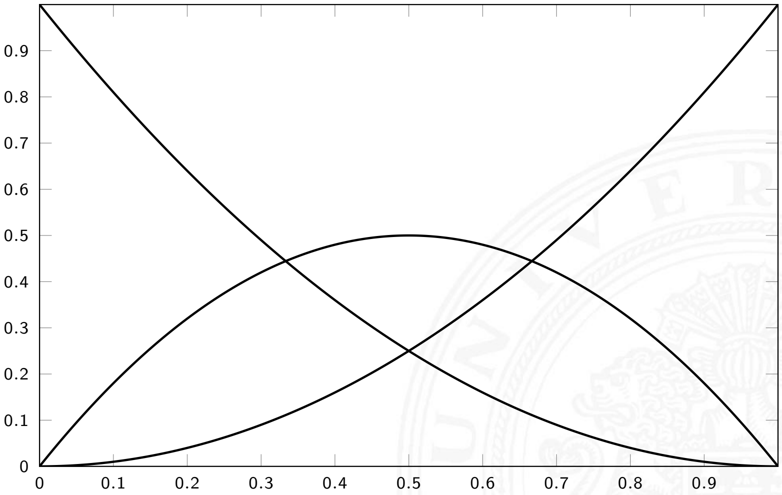
Properties of Bernstein-polynomials:

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

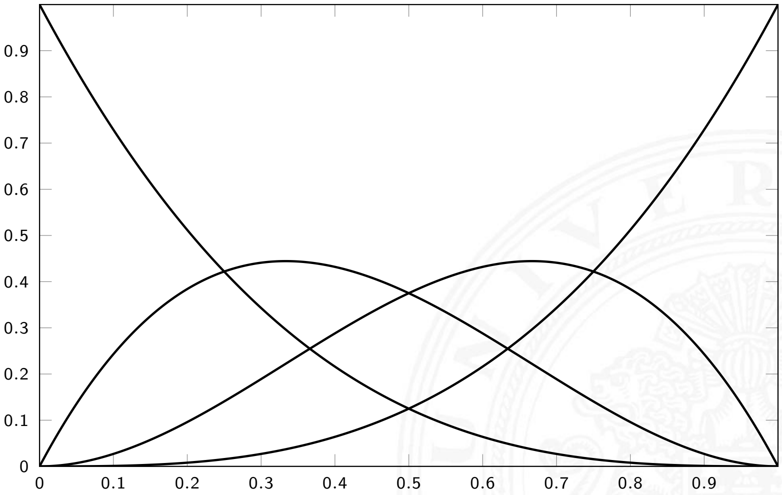
Polynomial of degree 1



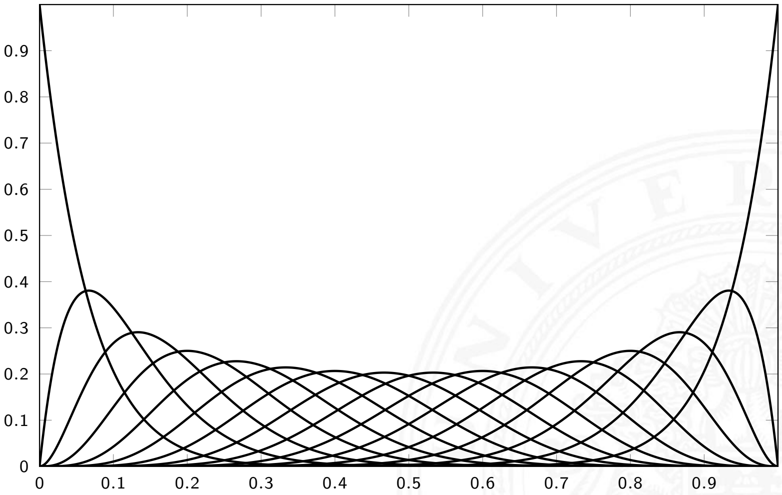
Polynomial of degree 2



Polynomial of degree 3



Polynomial of degree 15



Bernstein polynomials for trajectory generation

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

B-spline curves and basis functions

- Splines are used as `basis` function (hence Basis-Spline)
- B-spline curve is a polynomial
- B-spline curve of order k is composed of B-Splines (piecewise)
- Generally, $k-2$ derivations are continuous at intersections
- B-splines are polynomials based on the following ordered parameters

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

where

- m: is given by the number of points to be interpolated
- k: is the order of the b-spline curve

Definition of B-splines

The following functions are known as normalized B-splines $N_{i,k}$ of order k:

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

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

as well as 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 $i = 0, \ldots, m$.

B-Splines

Linear splines correspond to piecewise linear functions

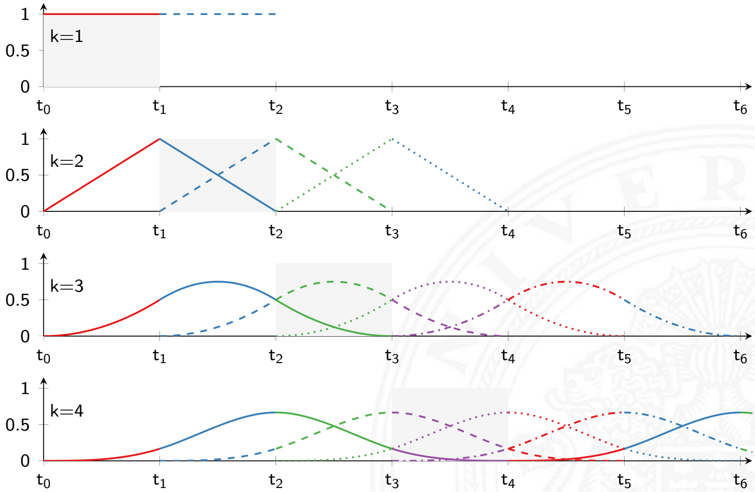
Advantages:

- splines are more flexible than polynomials due to their piecewise definition
- still, they are relatively simple and smooth
- prevent strong oscillation
- the values of the 1^{st} and 2^{nd} derivatives can be defined as constraints
- also applicable for representing surfaces (CAD modeling)

B-Spline curves

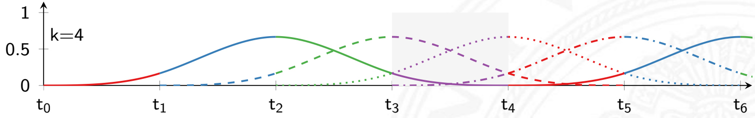
- Path controlled by de-Boor points
- Always constrained to de-Boor point's convex hull
- De-Boor points are of same dimensionality as B-spline curve
- B-spline curves have locality properties similar to Bézier-curves
  - control point P_i influences the curve only within the interval $[\tau_i, \tau_{i+p}]$

Examples of B-splines



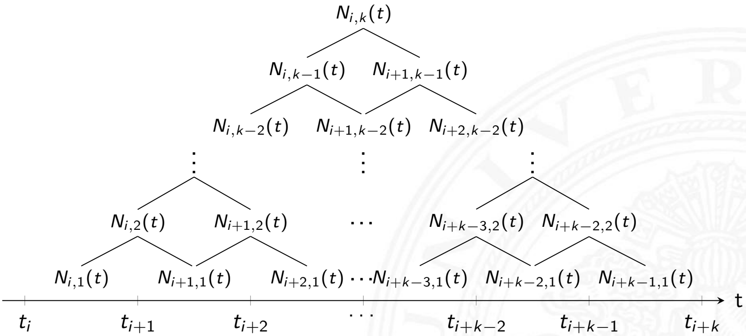
Overlapping

There are $k = p + 1$ overlapping B-splines within an interval.
An example of cubic ($p = 3$) B-splines:

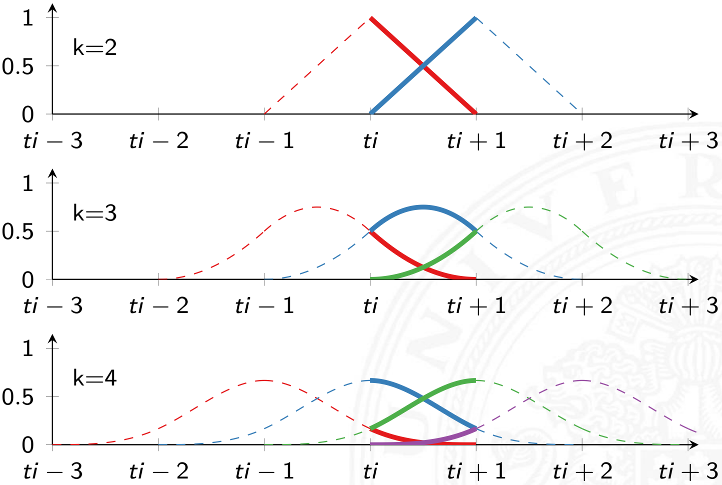


B-Splines of degree n

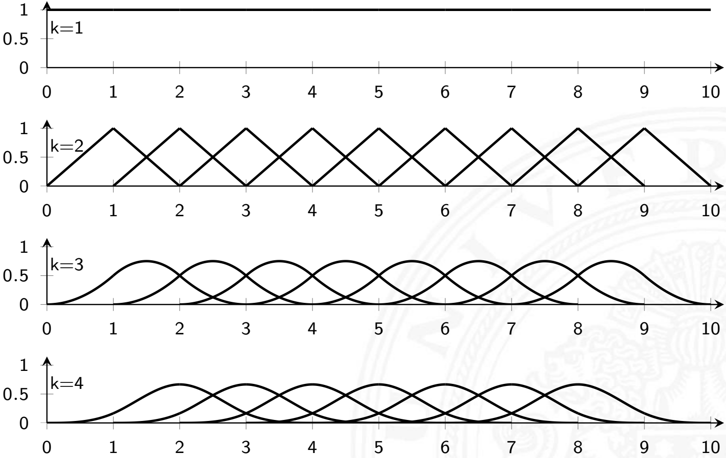
The recursive definition of a B-spline basis function $N_{i,k}(t)$:



B-Splines of degree n in interval $[t_i, t_{i+1}]$



Uniform B-splines of order 1 to 4



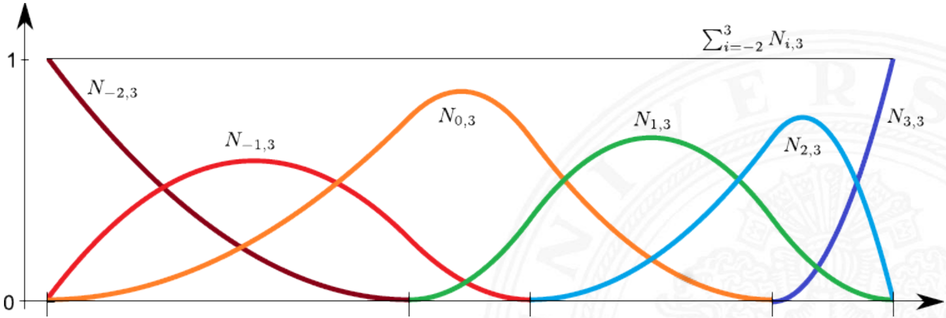
Uniform B-splines

- 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

$$B_{k,d(u)} = B_{k+1,d(u+\Delta u)} = B_{k+2,d(u+2\Delta u)},$$

u represents the control-point's values

Non-uniform B-spline of order 3



- `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 position, $\mathbf{r}(t)$ 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 $[t_{k-1}, t_{m+1}]$.

Generating control points from data points

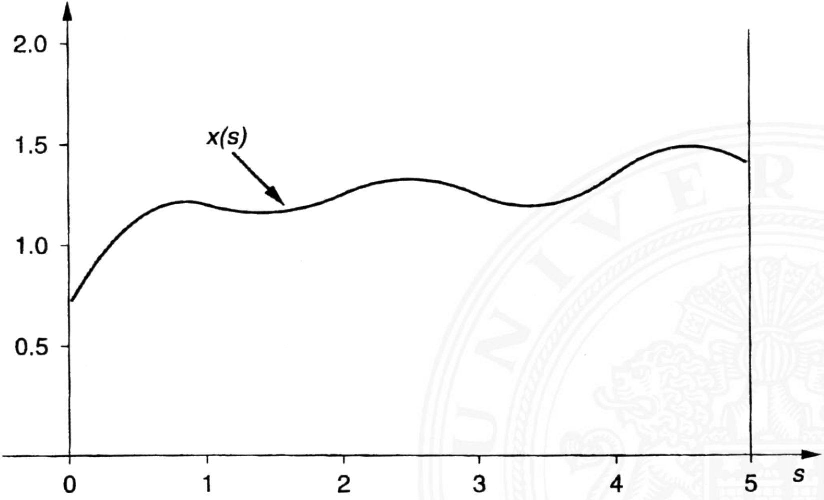
The control points $\mathbf{v}_j$ for interpolation are identical to the data points only if $k = 2$.
A series of control points forms a convex hull for the interpolating curve.
Two methods for generation of control points from data points:

- by solving the following system of equations

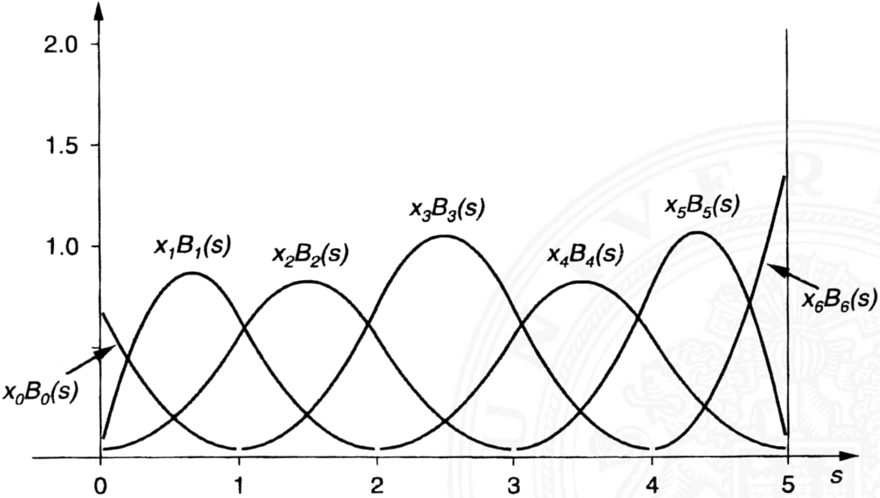
$$\mathbf{q}_j(t) = \sum_{j=0}^{m} \mathbf{v}_j \cdot N_{j,k}(t)$$

  where $\mathbf{q}_j$ are the data points to be interpolated, $j = 0, \cdots, m$.[5]:
- by learning, based on gradient-descend.[6]

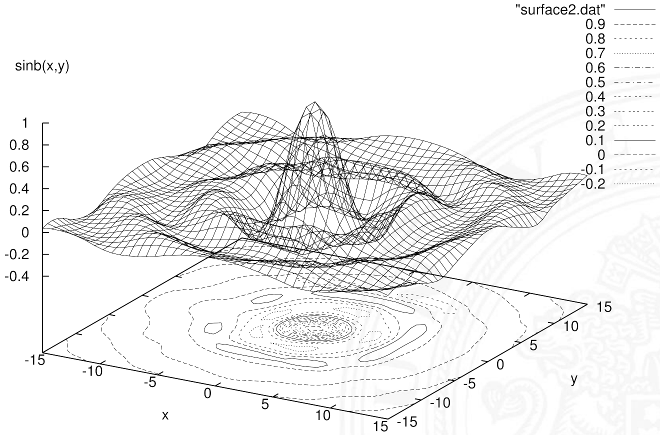
Function approximation – 1D example



Function approximation – 1D example (cont.)



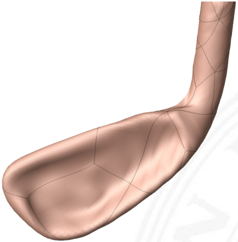
Approximation of $sinb(x,y) = sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$



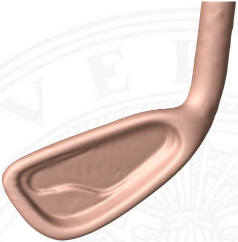
Surface reconstruction with B-Splines

- Surface reconstruction from laser scan data using B-splines [7]

Pointcloud (16,585 points)

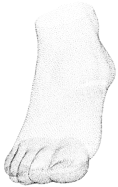


35 patches, 1.36% max. error

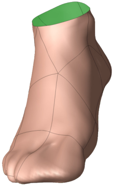


285 patches, 0.41% max. error

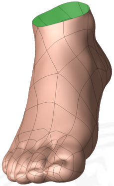
Surface reconstruction with B-Splines (cont.)



Pointcloud (20,021 points)

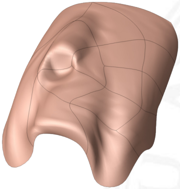


29 patches, 1.20% max. error

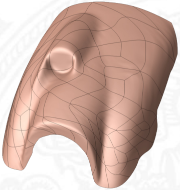


156 patches, 0.27% max. error

Pointcloud (37,974 points)



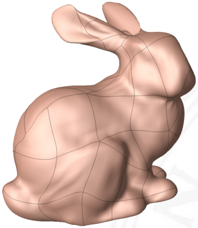
15 patches, 3.00% max. error



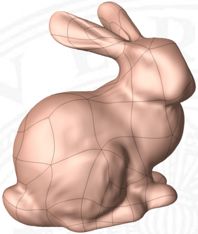
94 patches, 0.69% max. error

- Surface reconstruction 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) $(i = 0, 1, \ldots, l)$ 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 8

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

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

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 and potential energy 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

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

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

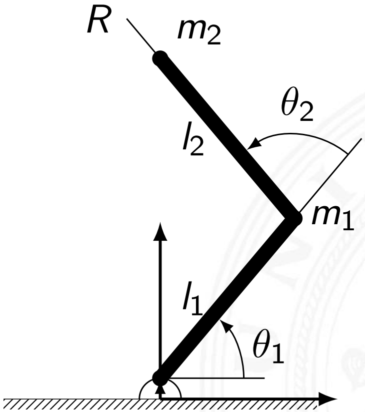
- Combining the different influence factors in the robot specific motion equation from kinematics $(\Theta, \dot{\Theta}, \ddot{\Theta})$
- Practically the Newton-, Euler- and motion-equation for each joint are combined
- Advantages: numerically efficient, applicable for complex geometry, can be modularized

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.**

Example: A 2 DOF manipulator

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



Newton-Euler-Equations for 2 DOF 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.

Lagrangian Equations

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

$$L = K - P$$

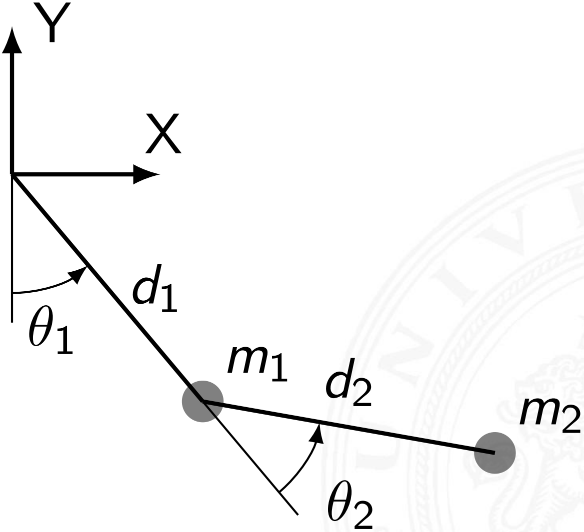
Theorem

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)

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:

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

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

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.

General dynamic equations of a manipulator

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

$M(\Theta)$: 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

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

Outline

Introduction

Coordinate systems

Kinematic Equations

Robot Description

Inverse Kinematics for Manipulators

Differential motion with homogeneous transformations

Jacobian

Trajectory planning

Trajectory generation

Dynamics

Principles of Walking

- Introduction
- ZMP
- Inverted Pendulum

Outline (cont.)

Motivation

- Enabling locomotion in difficult terrain
- Legs can be used for other things
- Necessary to integrate robots in a human environment

 [3]

 [4]

[3] http://1.bp.blogspot.com/-MhFnvPPR5V4/UmifTu4r_OI/AAAAAAAAFtI/FvJqeWu9Ahc/s1600/13-pictures-of-crazy-goats-on-cliff-transparent.png

[4] https://www.allposters.com

Problems

- Stability
- Energy consumption
- Hardware costs
- Complex control

[5]

[5] https://www.wikihow.com/Recognize-the-Signs-of-Intoxication

Types of Walking

- Static - Dynamic
- Passiv - Active
- 2,4,6,8,... legged
- Open loop - closed loop
- This lecture: active bipedal walking, no running


[6]


[7]

[6] https://3c1703fe8d.site.internapcdn.net/newman/gfx/news/hires/2017/1-sixleggedrob-transparent.png
[7] https://asl.ethz.ch/research/legged-robots.html

Types of Implementing Walking

- Control Theory
- Neural Networks
- Central Pattern Generators
- Evolutional Computing
- Expert Solution


[8]

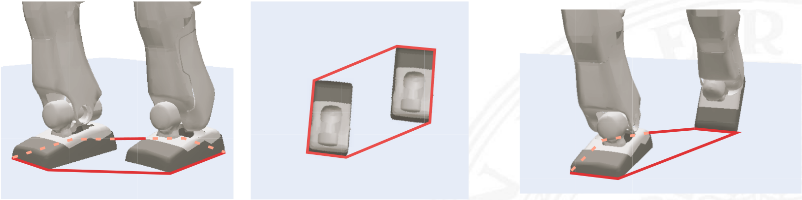
[8] https://de.wikipedia.org/wiki/Spline-Interpolation

Important Words

- Support leg/foot
- Flying leg/foot
- Torso / trunk
- Step / double step
- Sagittal / lateral

Support Polygon

- Convex hull of all ground contact points

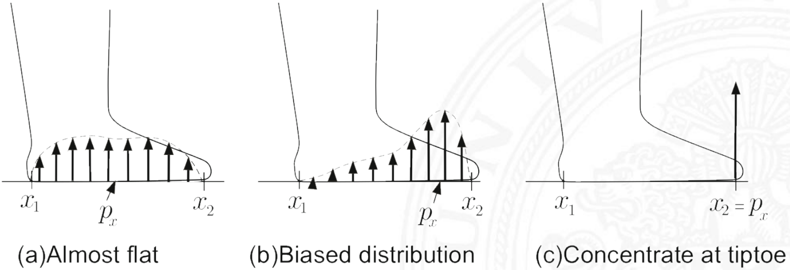


(a) Full contact of both feet (b) Partial contact [9]

[9] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Center of Pressure (CoP)

- Center of ground reaction forces
- Those can also be horizontal
- Moment becomes zero
- Equals the zero moment point (ZMP)

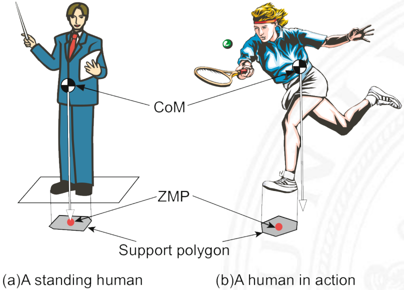


[10]

[10] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Zero Moment Point (ZMP)

- When standing, projection of CoM coincides with ZMP
- When dynamic, CoM outside of support polygon
- ZMP is always inside support polygon

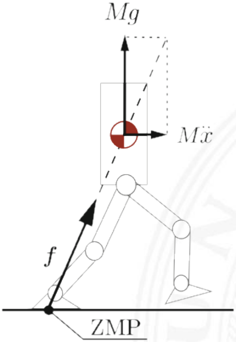


[11]

[11] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

ZMP

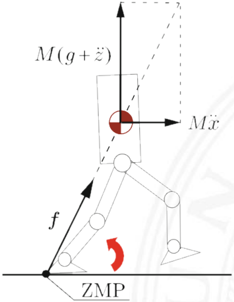
- Forces of the robot define position of ZMP
- Can it get outside of the support polygon?



12

[12] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

- No! The ZMP is always in the support polygon
- If it is on an edge, the robot rotates



[13]

[13] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

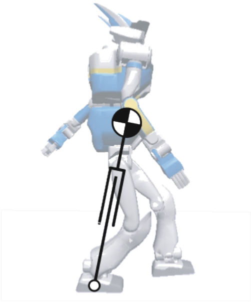
Limitations of ZMP

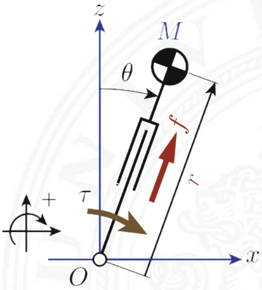
- Sole slips on ground
- Other parts of the robot are in contact with environment
- Ground is not perfectly level

[14]

[14] https://www.reddit.com/r/rickandmorty/comments/70t45i/anyone_else_wish_heshe_could_experience_true_level/

- Simplest model for walking robot or human
- Point mass at end of massless telescopic leg
- f: kick force, tau: torque

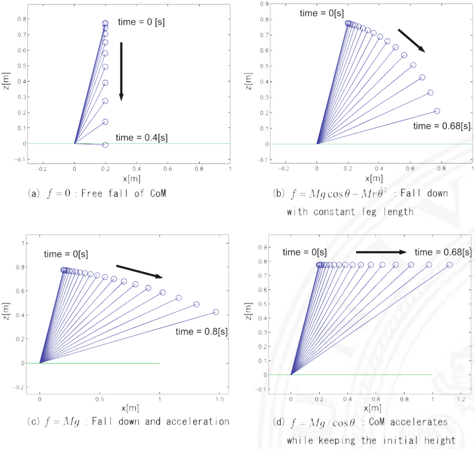




15

[15] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Inverted Pendulum

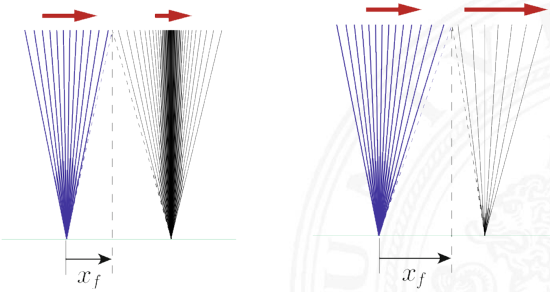


[16]

[16] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Support Leg Exchange

- Considering fixed step length
- Earlier touchdown of the next step results slow down
- Later touchdown of the next step results speed ups

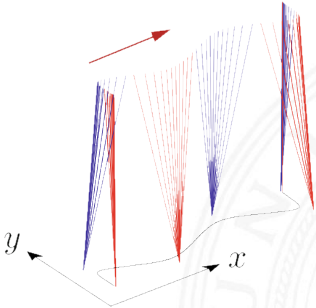


[17]

[17] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

3D Support Leg Exchange

- Transfer to 3D
- Introduction of lateral movement

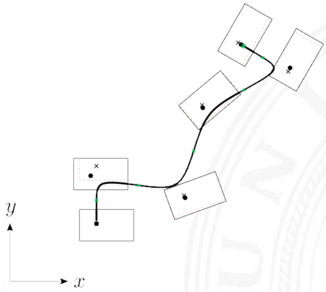


18

[18] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Omni-directional Walking

- Forward (x)
- Sideward (y)
- Turn (yaw)

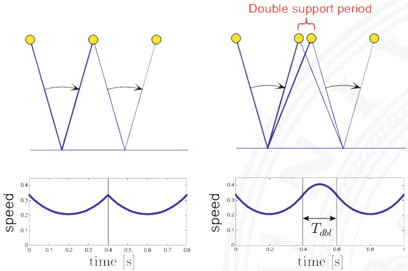


19

[19] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Double Support Phase

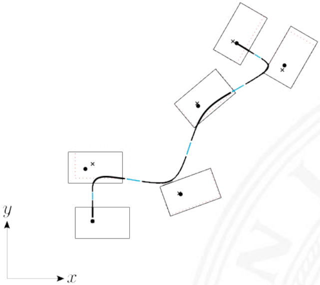
- Accelerations are extreme on support change
- Not feasible in reality
- Introduction of a double support phase



[20]

[20] Introduction to Humanoid Robotics, Shuuji Kajita, 2015

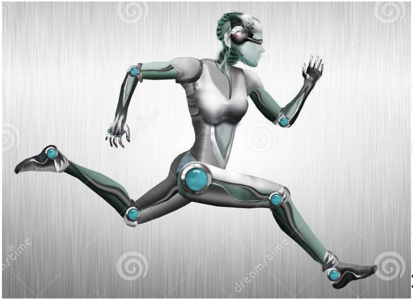
Double Support



21

21 Introduction to Humanoid Robotics, Shuuji Kajita, 2015

Zero Support



[22]

[22] https://thumbs.dreamstime.com/z/running-robot-27653003-transparent.png

?

- Why are we not finished yet?

Detecting Instability

Which senses do you think humans use for walking?

Detecting Instability

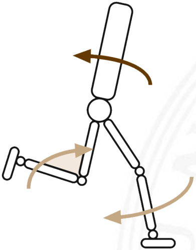
- Sensors
  - IMU(s)
  - Force sensors on foot sole
  - 6 axis force/torque sensor in ankle
  - Joint Torques
  - Camera
- Model
  - Joint positions
  - Link masses and inertia
  - Rigidity of links (especially foot soles)

Stabilizing Approaches

- Simple stopping
- Counter movements with the arms/torso
- Change of step position (capture steps)

Counter Movements with Upper Body

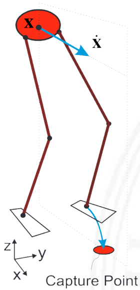
- Rotation around edge of support polygon
- Introduce counter force with arms/torso or flying leg
- Flying leg is mostly not usable


23

---
[23] Springer Handbook of Robotics, Bruno Siciliano, 2016

Capture Step

- Capture point is where the robot comes to a complete stop
- Multiple capture steps may be necessary
- You can completely base your walking on this



[24]

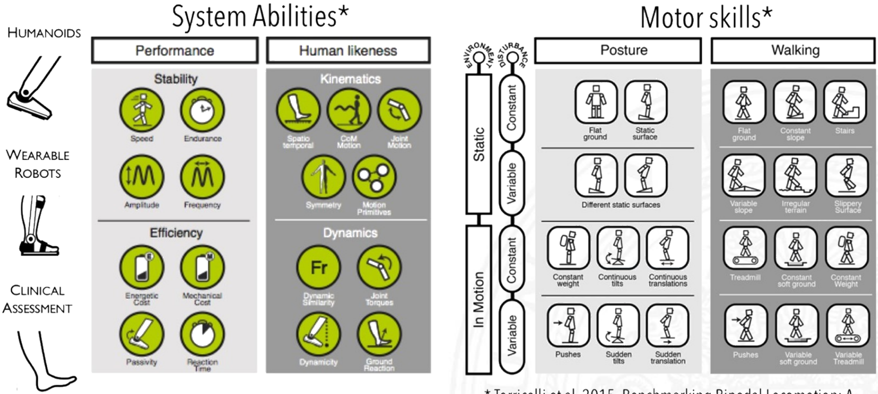
[24] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6094435

Machine Learning

- We will not cover machine learning
- If you are interested join my lecture in "Intelligent Robotics" in the winter term
- General approaches are:
  - Learning parameter of a walking pattern generator (e.g. double support length)
  - Learning neural networks from scratch
  - Learning from demonstration
  - Artificial central pattern generators

Current State of the Art

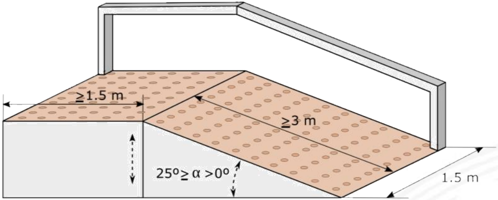
- Some very expensive robot manage to solve the problem (at least most of the time) using control theory
- Cheaper robots still struggle to achieve really stable walking
- Machine learning approaches still mostly only work in simulation (reality gap)
- Working on better comparison between approaches, e.g. EuroBench

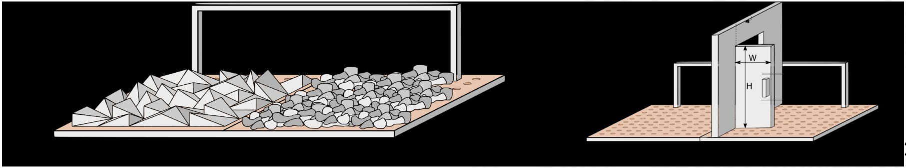


www.benchmarkinglocomotion.org

* Torricelli et al. 2015, Benchmarking Bipedal Locomotion: A Unified Scheme for Humanoids, Wearable Robots, and Humans [25]

[25] http://eurobench2020.eu/abstract/motivation-background/





[26]

[26] Eurobench Guide for Applications

Full Body Motion

- Small overview of full body motions
- Examples are: walking with hand on handrail or standing up
- Higher complexity since all limbs are involved
- Breaks assumptions that are often made for normal walking
- Motions can be periodic or non periodic

Walking with Hand Contacts

- Using handrail, pushing cart, opening door, holding hands, using walking stick, collaborative carrying
- Introduces additional forces on the robot
- Support polygon maybe totally different
- More complex models have to be used
- Currently mostly used approach: quadratic programming
  - Solve problem of optimizing a quadratic function with multiple linear constrains
  - Use rigid body dynamics together with a model
  - Problems
    - Model is not perfect
    - If caring an object, you need a model of it
    - Robot is maybe not perfectly rigid

Non Periodic Motions

- Simpler due to known start and end
- Examples
  - Standing up
  - Kicking
  - Grasping
  - Waving

Implementing Non Periodic Motions

- Keypoint teach in
  - Put robot into key positions manually
  - Save joint positions at these points
  - Interpolate
  - Useful for simple motions (e.g. waving) or static robots
- Learning from demonstration
  - Either demonstrate on the robot itself or by using motion capture
  - Normally more than one demonstration
  - Not just simply replaying
- Cartesian splines
  - Define trajectories of the limbs with Cartesian splines manually
  - Comparably easy to do for humans (much better than joint space)
  - Splines configurable with few parameters
  - Use inverse kinematics to compute joint goals
  - Optionally use additional goals in the IK solver to keep balance

Implementing Non Periodic Motions

- DeepLearning
  - Just let it learn in simulation till it works
  - Put it on the robot and hope for the best
  - Reality gap
- Control Theory
  - Have an open loop trajectory, e.g. from teach in
  - Use a stability criterion, e.g. ZMP
  - Adjust joint goals with controller, e.g. PID
- More on the learning aspect in the intelligent robotics lecture

Questions

Questions?

Introduction to Robotics

Lecture 9

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

Definitions

Controller

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

Definitions (cont.)

System

- Physical or technical construct
  - input signal – stimulus
  - output signal – response
- Transforms stimulus into response
- Symbolical illustration
  - block with marked signals
  - direction of signal effect expressed with arrows

Definitions (cont.)

Input and output variables

- Change over time
  - expressed as $u(t)$ and $v(t)$ (dynamic system)
- Infinite number of possible variables
  - for real-world dynamic technical systems (in principle)
- Description of system behaviour based on desired application
  - using the relevant variables

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

Development of Control Engineering - Timeline

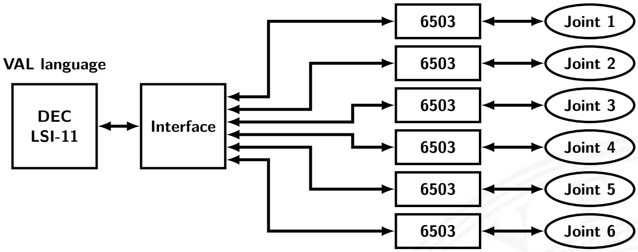
1788 J. Watt: engine speed governor
1877 J. Routh: differential equation for the description of control processes
1885 A. Hurwitz: stability studies
1932 A. Nyquist: frequency response analysis
1940 W. Oppelt: frequency response analysis, Control Engineering becomes an independent discipline
1945 H. 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

Classification of Robot Arm Controllers

As the problem of trajectory-tracking:

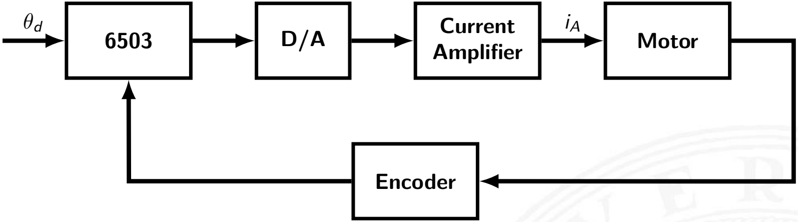
- Joint space: PID, plus model-based
- Cartesian space: joint-based
  - using kinematics or using inverse Jacobian calculation
- Adaptive: model-based adaptive control, self-tuning
  - controller (structure and parameter) adapts to the time-invariant or unknown system-behavior
  - basic control circle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

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
- Potentiometer are used for rough estimation (only PUMA-560)
- No dedicated speedometer
    - velocity is calculated as the difference of joint positions over time

Internal Sensors of Robots

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

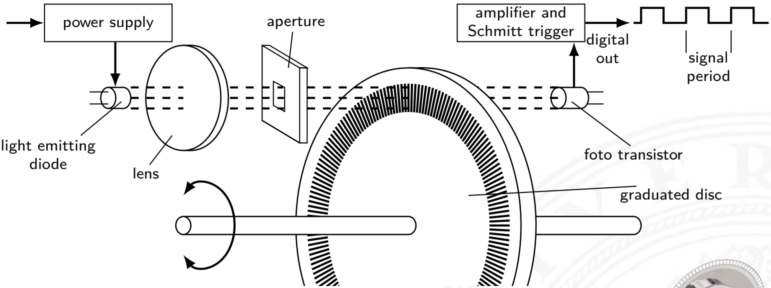
Position measurement systems

- Potentiometer
- Incremental/absolute encoder
- Resolver

Velocity measurement systems

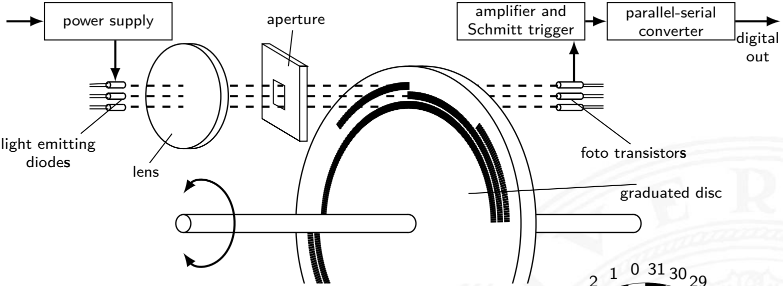
- Speedometers
- Calculate from position change over time

Optical Incremental Encoders

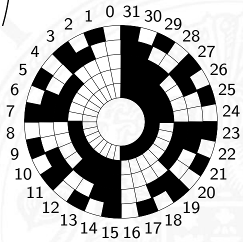


- An optical encoder reads the lines
- The disc is mounted to the shaft of the joint motor
  - PUMA-560: 1:1 ratio; .0001 rad/bit accuracy
- one special line is marked as the “zero-position”

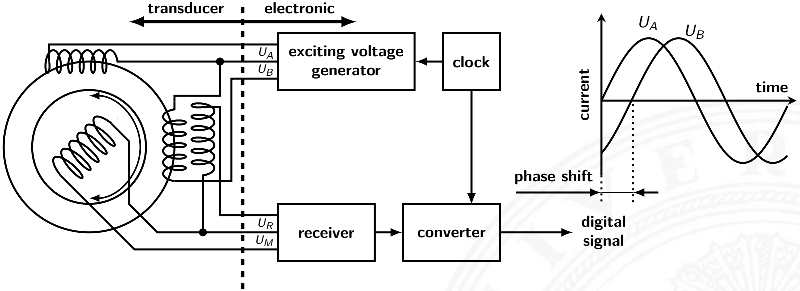
Optical Absolute Encoder



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

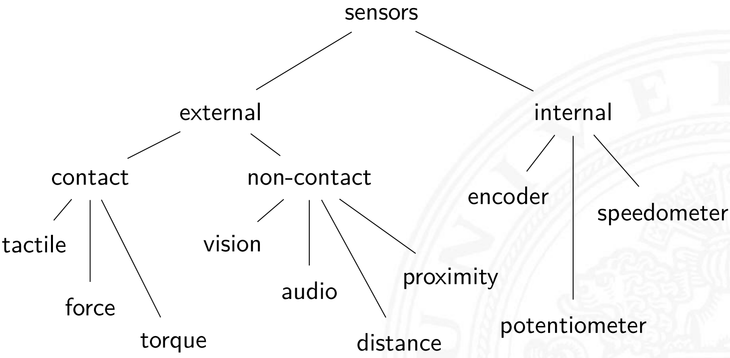


Resolver

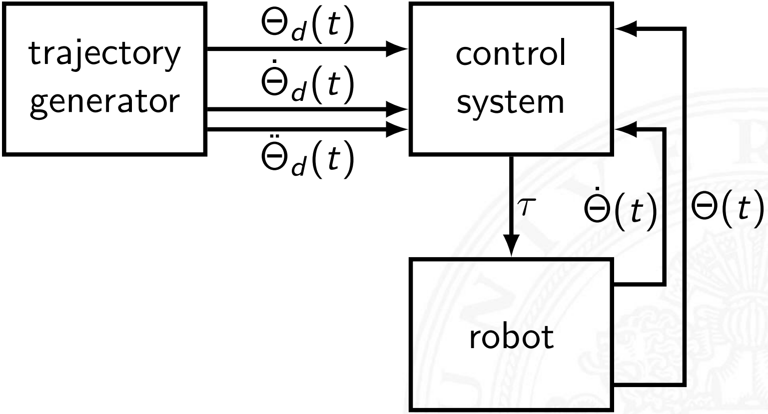


- analog rotation encoding
- phase shift between U_A and U_B determines rotation
- precision depending on digital converter

Sensor Classification Hierarchy



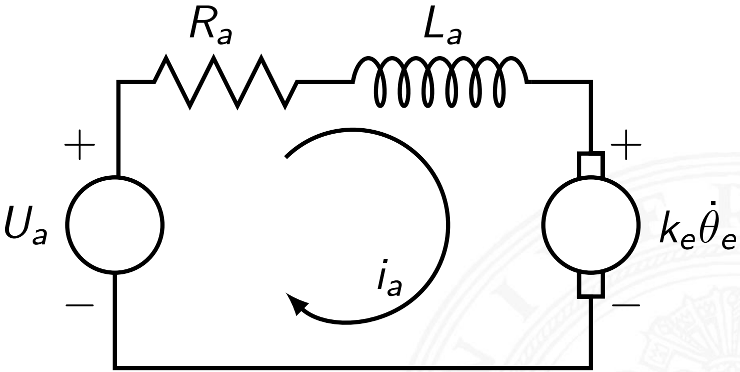
Control System of a Robot



Control System of a Robot (cont.)

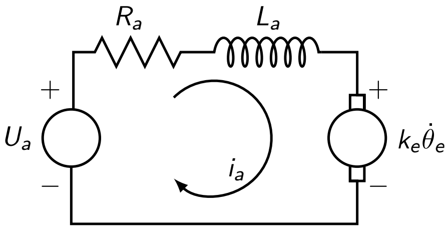
- Target values
  - $\Theta_d(t)$
  - $\dot{\Theta}_d(t)$
  - $\ddot{\Theta}_d(t)$
- Magnitude of error
  - $E = \Theta_d - \Theta, \dot{E} = \dot{\Theta}_d - \dot{\Theta}$
- Output (Control) value
  - $\Theta(t)$
  - $\dot{\Theta}(t)$
- Controlled value
  - τ

Simplified Circuit of a DC-Motor



U_a	input voltage of armature (motor) circuit
R_a	armature (motor) resistance
L_a	armature (coil) inductance
i_a	armature current (passing the motor)
k_e	exciter (motor) torque constant

Simplified Circuit of a DC-Motor (cont.)



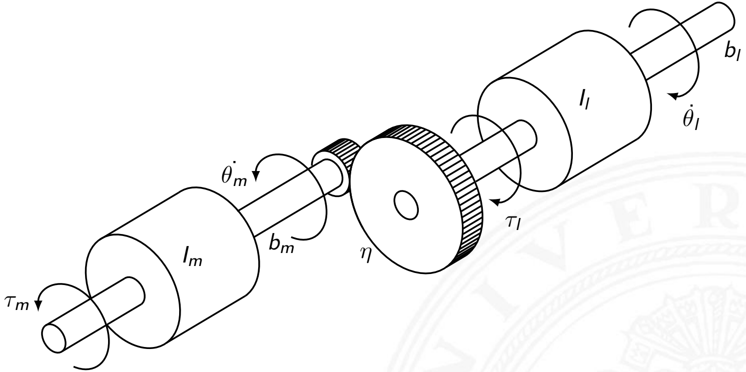
U_a	input voltage
R_a	armature resistance
L_a	armature inductance
i_a	armature current
k_e	exciter torque constant

The circuit can be described with the first order differential equation:

$$L_a \dot{i}_a + R_a i_a = U_a - k_e \dot{\theta}_e$$

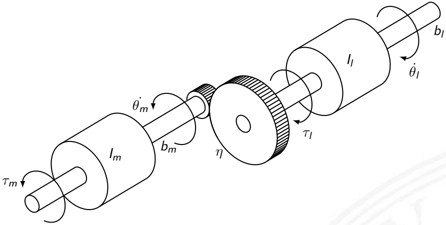
- Inductance relative to current change
- Resistance relative to absolute current
- Torque relative to rotation change

Connection Between Motor and a Joint



η	transmission ratio
$I_{m/l}$	inertia of motor/load
$\tau_{m/l}$	torque of motor/load
$\dot{\theta}_{m/l}$	rotation velocity of motor/load
$b_{m/l}$	friction factor

Connection Between Motor and a Joint (cont.)



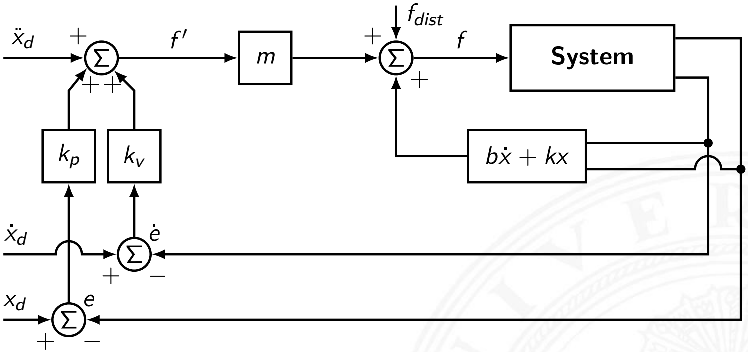
The motor torque formula is

$$\tau_m = (I_m + I_l/\eta^2)\ddot{\theta}_m + (b_m + b_l/\eta^2)\dot{\theta}_m$$

an the load torque is

$$\tau_l = (I_l + \eta^2 I_m)\ddot{\theta}_l + (b_l + \eta^2 b_m)\dot{\theta}_l$$

Linear Control for Trajectory Tracking



$$f' = \ddot{x}_d + k_v\dot{e} + k_p e + k_i \int e dt \tag{86}$$

is called the principle of PID-control.

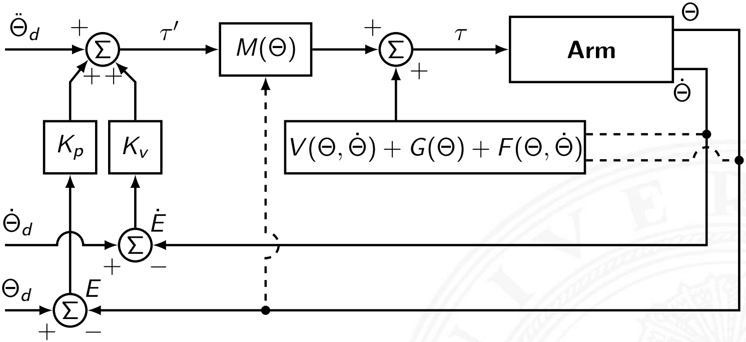
PID-Controller

- 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:

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

Model-Based Control for Trajectory Tracking



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.

Robot Control Improvements

Scientific Research

- model-based control
- adaptive control

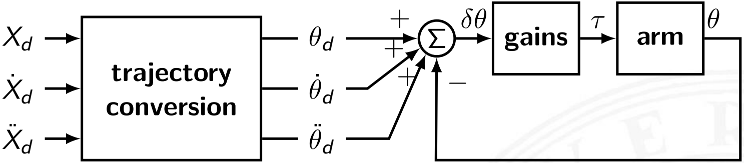
Industrial robotcs

- PID-control system with gravity compensation

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

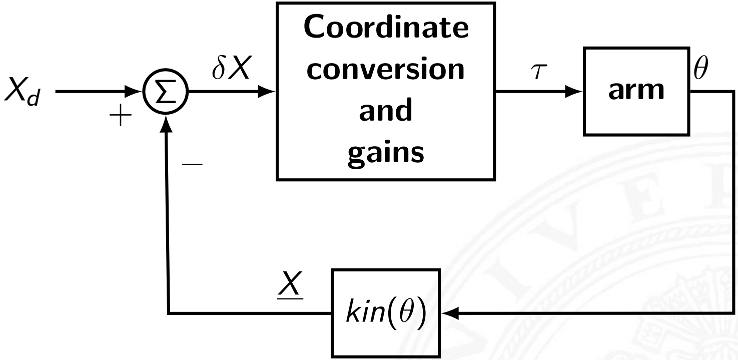
Control in Cartesian Space – Method I

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

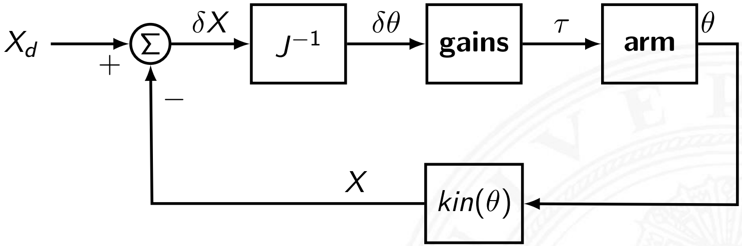
Control in Cartesian Space – Method II
Cartesian control via calculation of kinematics



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

Control in Cartesian Space – Method III

Cartesian control via calculation of inverse Jacobian



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

Hybrid Control of Force and Position

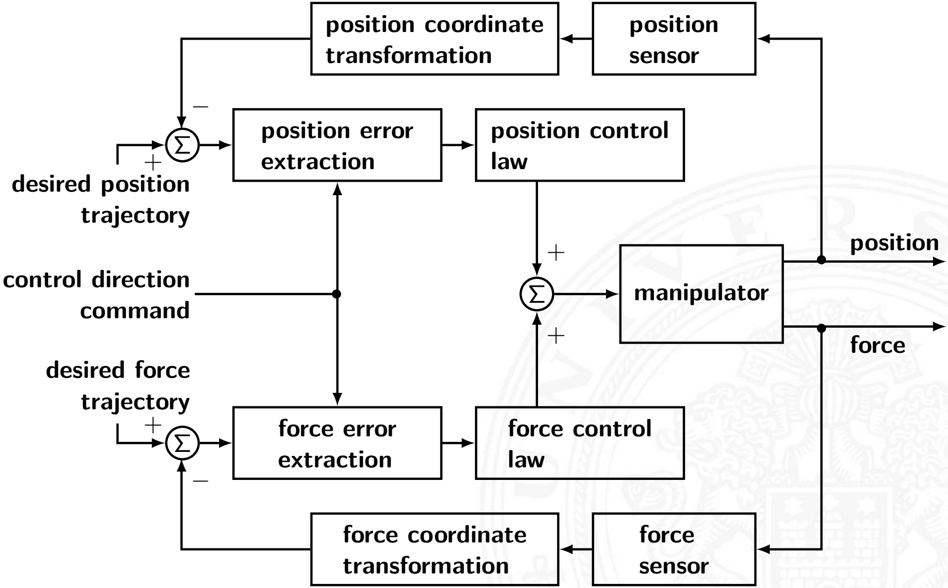
Motivation

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 seperate control of position and force

Hybrid Control of Force and Position (cont.)



Hybrid Force/Torque Control for safe HRI

Introduction to Robotics

Lecture 10

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

Basics of Task-Level Programming

Goal enable task-specification with symbolically described states
where planning of necessary movement is up to the robot system

Example driving commands should only require the target position instead of specifying how to move precisely

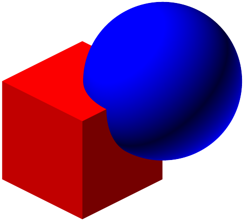
Common problem of task-level programming

Collision avoidance

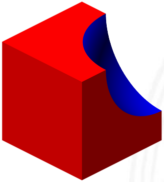
A general approach – geometric trajectory planning:
to plan collision-free motion for the known models of manipulators and obstacles in the workspace.

- Approximating methods
  - bounding box
  - convex hull
  - spherical and ellipse models
- `Constructive Solid Geometry` (CSG)
  - `Boundary Representation` (BREP)
  - `Sweep Representation`
- Spatial data structures
  - Grid-Model (`Spatial Occupancy Enumeration`)
  - Hierarchical Representation: (`quadtree`, `octree`)

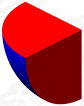
- Method to model bodies
- Direct modeling
- Design of complex surfaces
- Combination of basic shapes using the boolean operators



union



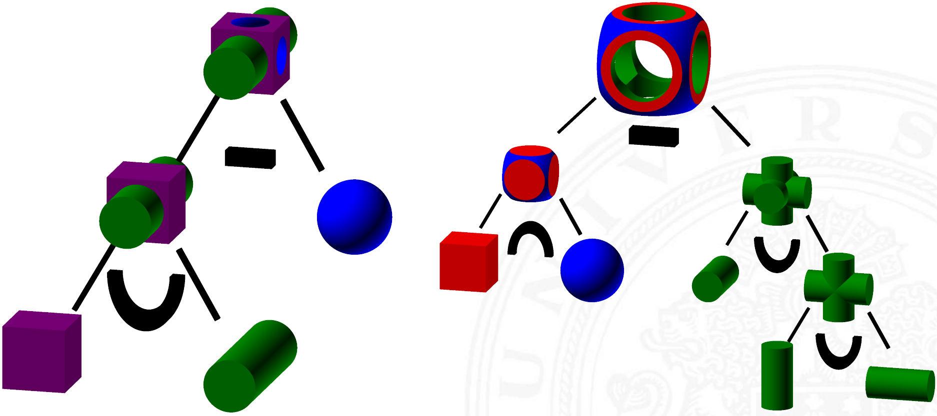
difference



intersection

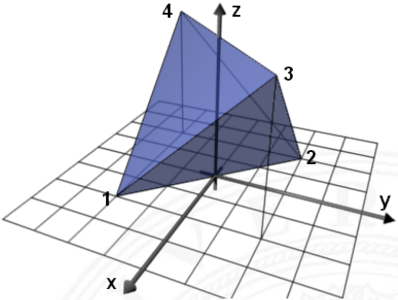
CSG Representation (cont.)

- CSG-Tree



Boundary Representation

- Method to model bodies
- Indirect modeling
- Surface / Volume model
- Vertice-Edge-Surfaces



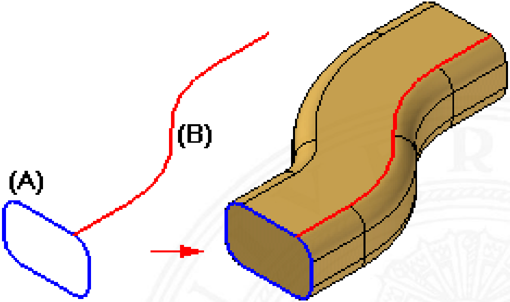
Edge-#	V-$\#_1$	V-$\#_2$
1	1	2
2	2	3
3	1	3
4	1	4
5	2	4
6	3	4

V-#	x	y	z
1	2	-2	0
2	-2	2	0
3	2	2	4
4	-2	-2	4

Surface-#	Edge order
1	1–2–3
2	3–6–4
3	2–5–6
4	1–4–5

Sweep Representation

- method to model bodies
- models in 2.5D
- intuitive
- quadratic, cubic polynomials



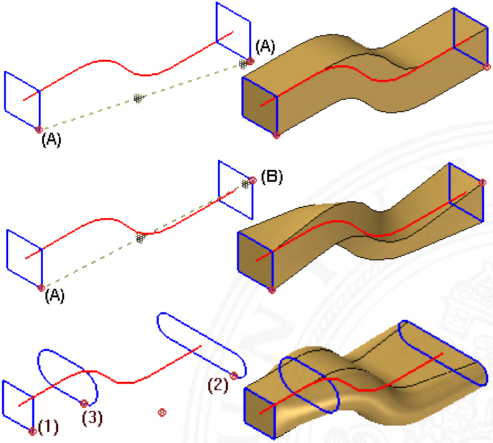
A 2D-shape
B extrusion path

Sweep Representation (cont.)

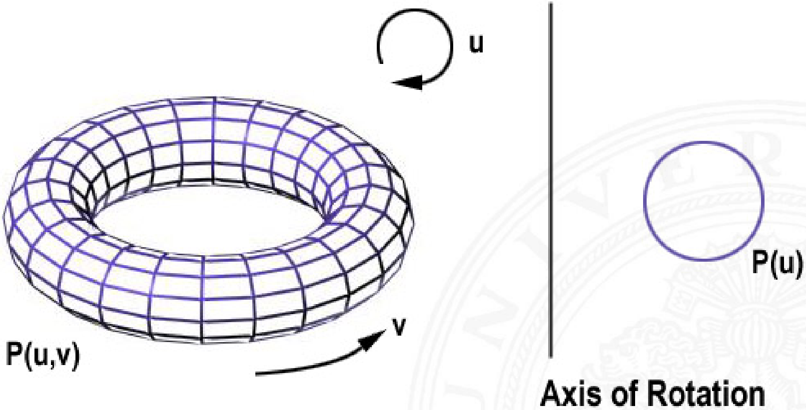
Simple path

Twisted path

Shape modification

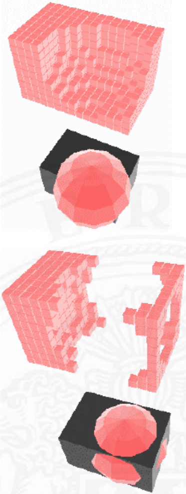


Sweep Representation (cont.)

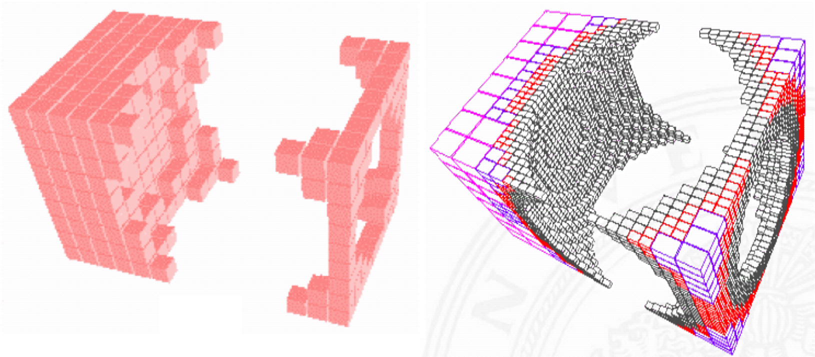


Grid-Model (Spatial Occupancy Enumeration)

- Volume model in virtual space
- Enclosed hull
- Voxel based
- Unambiguous definition from inside and outside
- Easy check for collisions between objects
- Representation using CSG or BREP



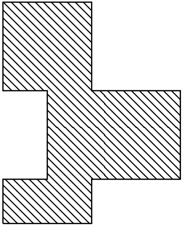
Grid-Model (Spatial Occupancy Enumeration) (cont.)



Quadtree Representation

- 2D modeling
- Taken over from DB-applications
- Surface is partitioned into 4 parts
- Indexing of created surfaces
- Level of partitioning depends on the density of the object
- `Octree` is the 3D-equivalent

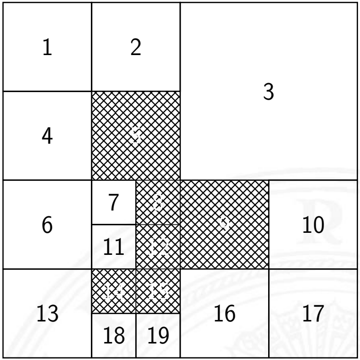
Quadtree Representation (cont.)

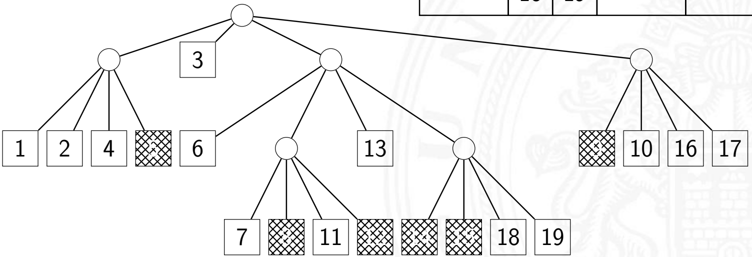


Node with descendants

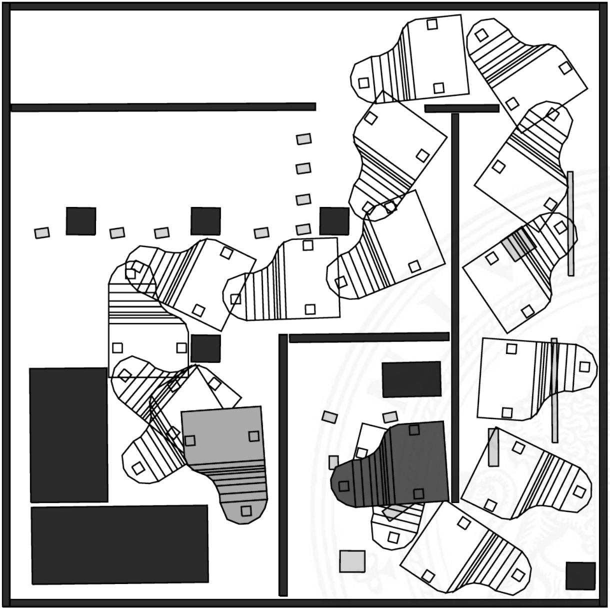
Node repesenting an empty block

Node repesenting a material block



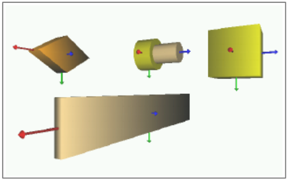


Piano Mover Problem

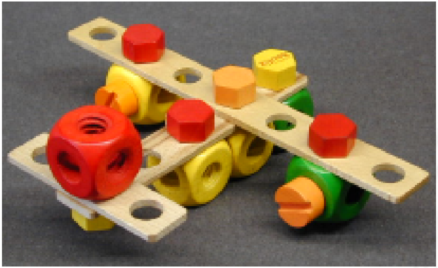


Puzzle solving

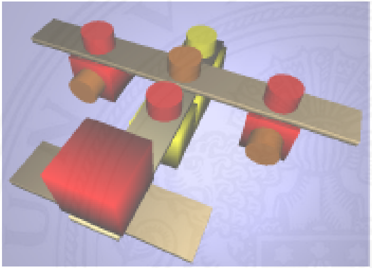
Assembly Strategies



assembly parts



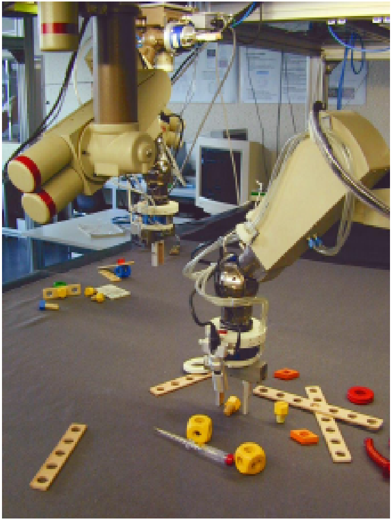
physical assembled plane



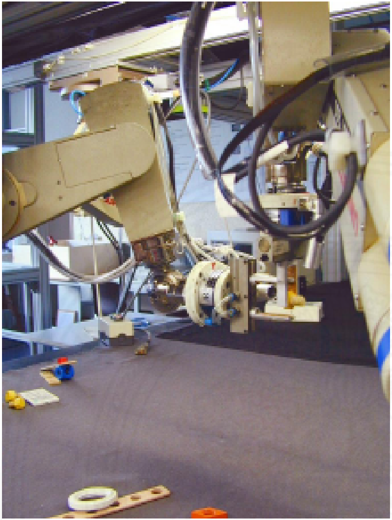
simulated assembled plane

Learning of Assembly Strategies in a distributed Multi-Robot-Environment [8]

Assembly Strategies (cont.)

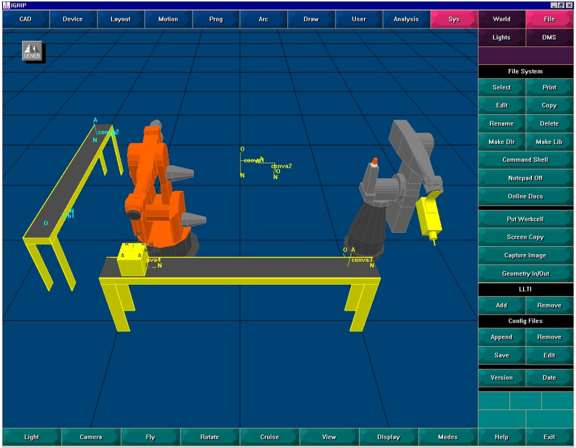


assembly start

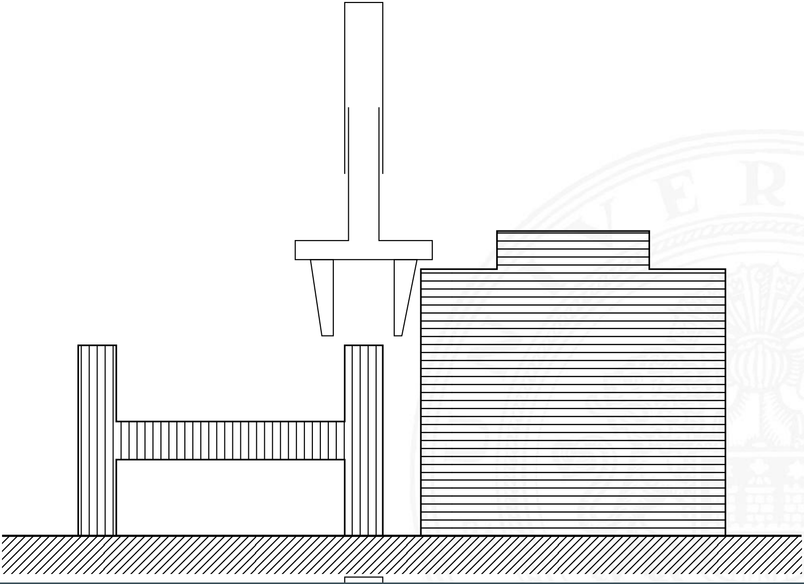


during assembly

Robot Programming



Positioning of a Gripper



Tasks comprised:

- Geometric paths
- Trajectories
  - position, velocity and acceleration functions over time
- Instruction order for sensor-based motion

Goals comprised:

- Motion to goal position without colliding
- Autonomous assembly of an aggregate
- Spatial recognition

Configuration of an Artifact

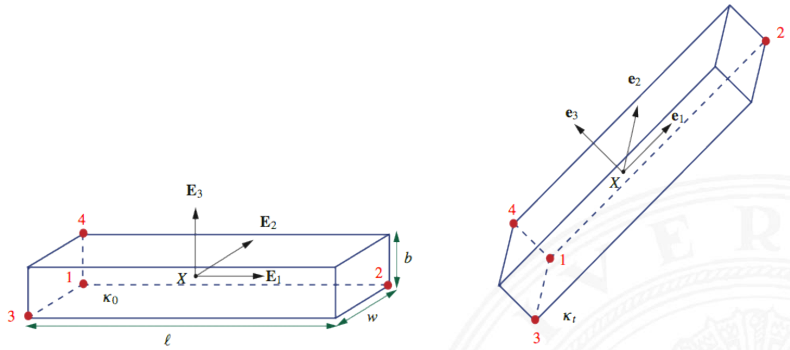
Artifact

A virtual or real body, that can change its place and form over time.

A configuration of an artifact is a set of independent parameters, which define the position of all its points in a reference frame.

- Can be expressed as a geometrical state-vector
- Number of parameter for the specification of the configuration is equal to the degrees of freedom

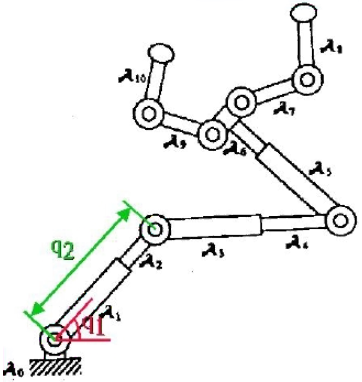
Configurations of a Rigid Body

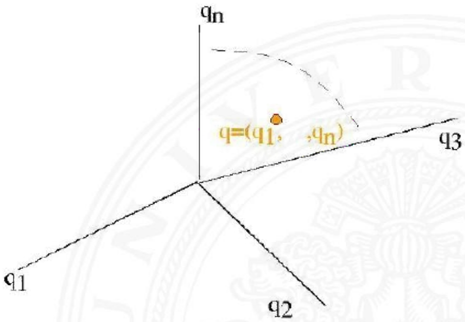


Configuration of an object

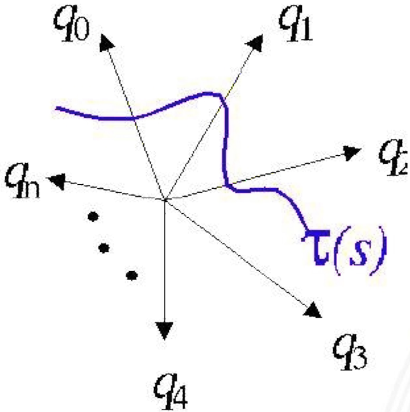
- 2D: (x, y, θ)
- 3D: $(x, y, z, \alpha, \beta, \gamma)$
- Plane: (`longitude`, `latitude`, `altitude`, `roll`, `pitch`, `yaw`)

Configurations of a Multi-joint Manipulator





Configurations and Paths of a Human Body



Path

A steady curve, connecting two configurations

$\tau : s \in [0, 1], \tau(s) \in \texttt{configuration space}$

Definition
Basic path problem

Generalized motion problem

> *"Given a number m of statical obstacles and an artifact with d degrees of freedom, the task of geometrical path planning is to determine a path between two configurations without collisions."*

A `complete` path-planner shall always deliver a valid plan if one exists, otherwise it should notify about the non-existence of a path.

Input and Output

Known are:

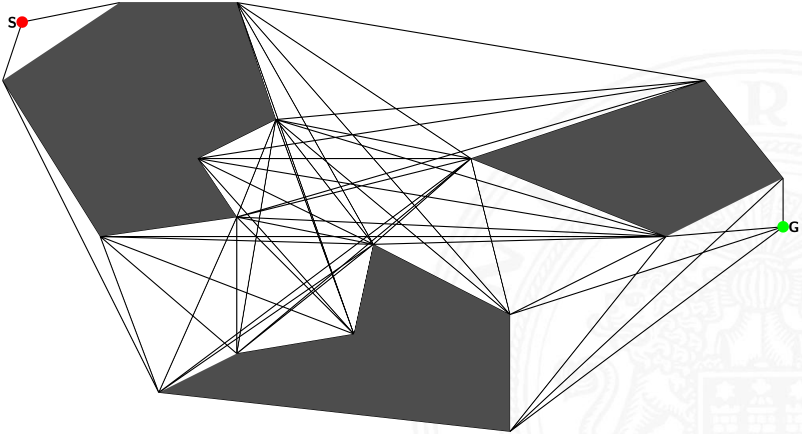
- Completely a `priori` modeled geometry of the artifact and the obstacles
- Kinematics of the artifact (a rigid body or a body with alterable shape)
- Start and goal configuration

To determine:

- Sequence of steady transformations of collision-free configurations of the artifact from the start to the goal configuration

Visibility Graph

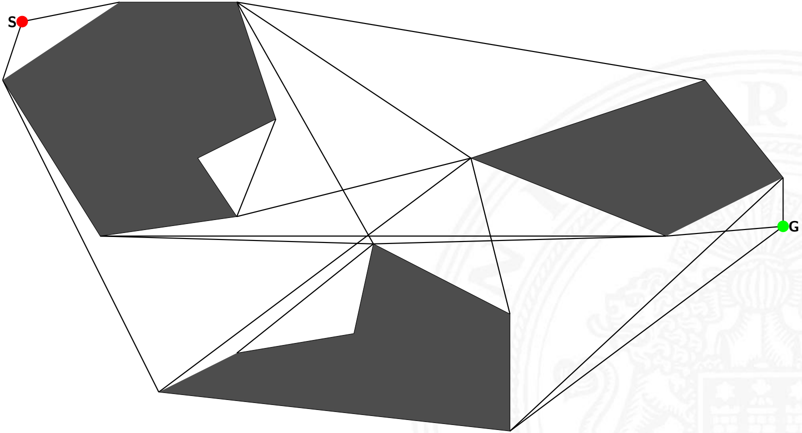
The `Visibility Graph` (V-Graph) is constructed by linking the visible corner points of the obstacles (`visible`: line does not intersect obstacle).



Complexity: $O(m^2)$, m is the no. of obstacle polygon vertices

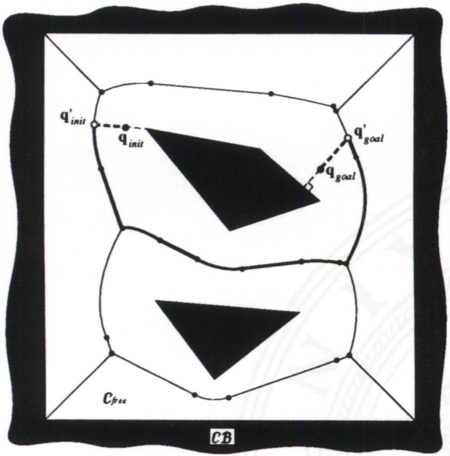
Tangent Graph

The `Tangent Graph` (T-Graph) was introduced as a subgraph of the V-Graph. It can be proven, that the shortest route between the start and goal is a subset of the T-graph.



Complexity: $O(m^2)$

Voronoi Diagram



Construction complexity: $O(m \log m)$
Search complexity: $O(m)$

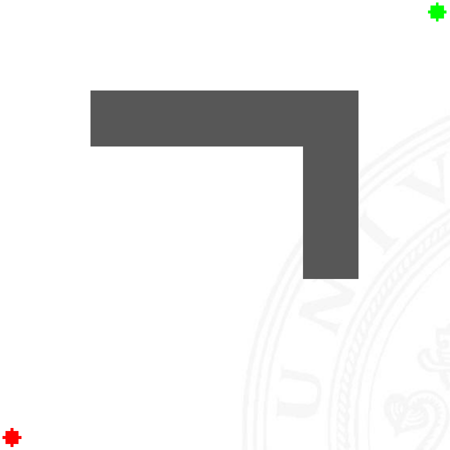
- A^*-algorithm is used to find the least-cost path
- Search a path from the initial node **s** to (one of) the goal node(s) **z**
- A heuristic cost function f is used, which assigns a value to every route from the initial to an arbitrary node **q**
- This value is used to estimate the complete costs from the initial node to the goal node (passing node **q**)
- The estimation function f can be defined as an addition of two functions g and h
  - g describes the known cost from the initial node to node **q**
  - h estimates the cost of the shortest route from **q** to the goal node **z**
- If h is chosen the way that the actual costs are not over-estimated, the search algorithm is called A^*

- It is guaranteed, that the shortest existing route can be found with the A^*-algorithm
- In order to find not only the shortest, but also the smoothest route, the costs of a route contain also a factor for direction changes. g and h are defined such that
  - $g = e(s,q) + w_f \cdot c_f(s,q)$
  - $h = e(q,z) + w_f \cdot c_f^*(q,z)$
  - $e(x,y)$ is the euclidean distance from x to y
  - w_f is a weight factor for the smoothness of the route
  - $c(x,y)$ is the measure of curvature of the route from x to y
    - * this value has to be estimated
- All possible route candidates from **s** to **q** are inserted into an open list
- The route candidate with the minimal f-value is moved from the open list to the closed list

Heuristical Search (cont.)

- This closed list route candidate is then expanded to all reachable neighbor-nodes and the new f function is evaluated.
- This is repeated until the goal-node is is expanded
  - a route has been found
  - there is no route from **s** to **z** if the open list is empty

A* path finding



Boundaries of Path Planning Algorithms

First lower boundary

`PSPACE`-hard, i.e. at least as complex as an `NP-problem`, in the worst case an exponential computing time for every algorithm to solve this problem [9]

First upper boundary

Double exponential time-complexity with the DOF d [10]

Second upper boundary

Single exponential time-complexity using silhouette-method [11]

Introduction to Robotics

Lecture 11

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

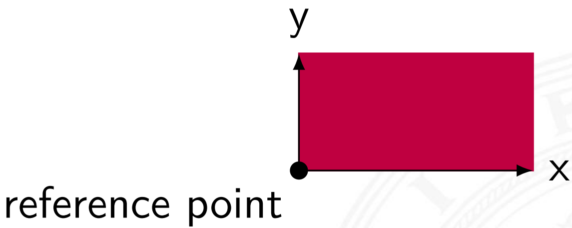
July 12, 2018

Outline

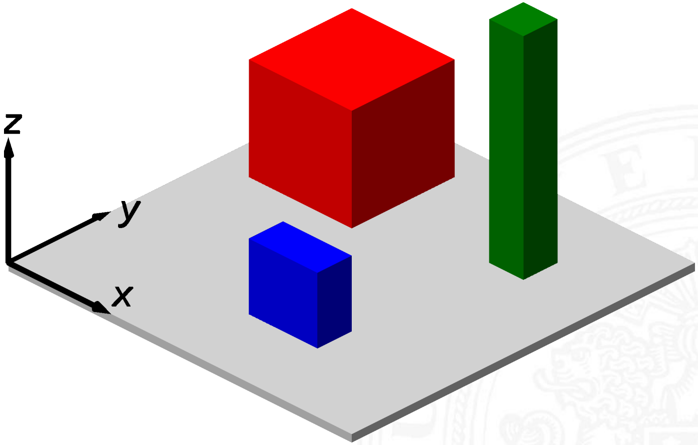
Outline (cont.)

Task-level Programming – Basics

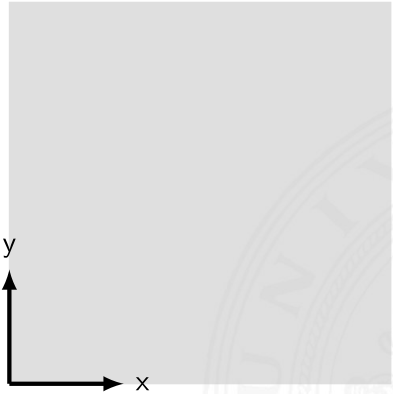
Robot Single reference point with physical attributes



Work space The cartesian space of the environment

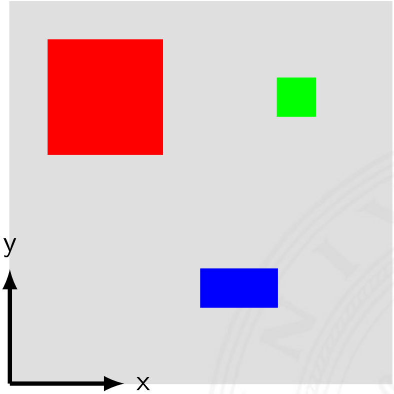


Configuration space $\mathtt{C}$ Set of all possible configurations

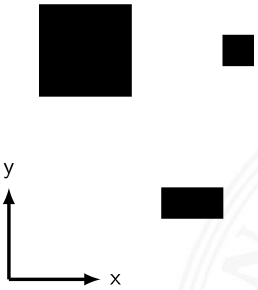


Task-level Programming – Basics

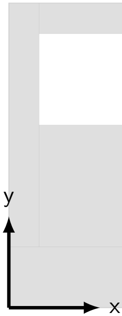
Obstacles in work space `C-Obstacles` in configuration space



Obstacle space $C_{\texttt{obstacle}}$ Union of C-Obstacles



Free space $C_{\texttt{free}}$ the complement of Obstacle space



Task-level Programming – Basics

Robot Single reference point with physical attributes

Work space The cartesian space of the environment

Configuration space $\mathtt{C}$ Set of all possible configurations

Obstacles in work space `C-Obstacles` in configuration space

Obstacle space $\mathtt{C}_{\mathtt{obstacle}}$ Union of C-Obstacles

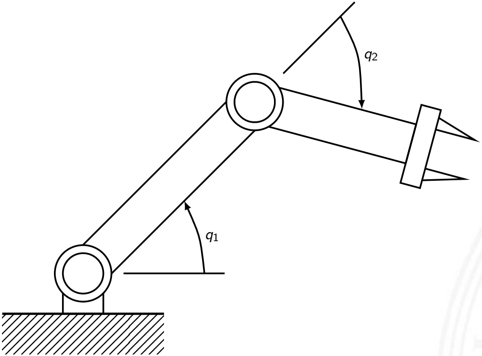
Free space $\mathtt{C}_{\mathtt{free}}$ the complement of Obstacle space

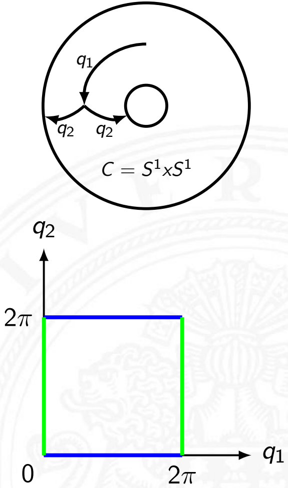
Path-planning for Work-/Configuration-Space

Search for a path for the reference point of the artifact in the free space.

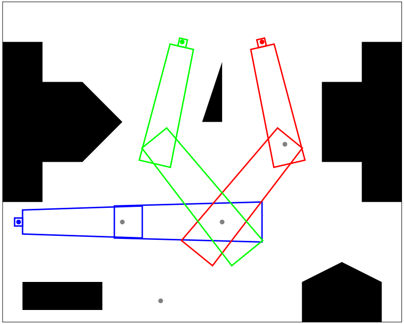
Configurations of the artifact in free space have no intersection with obstacles

Work Space to Configuration Space – Illustration

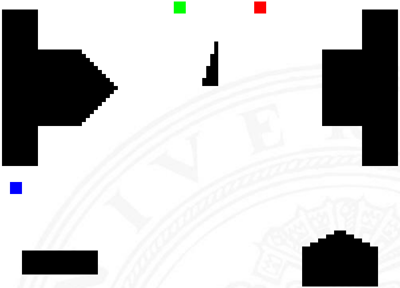




Work Space to Configuration Space – Example

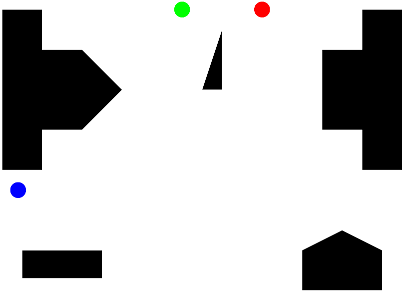


Workspace scheme with start and goal positions

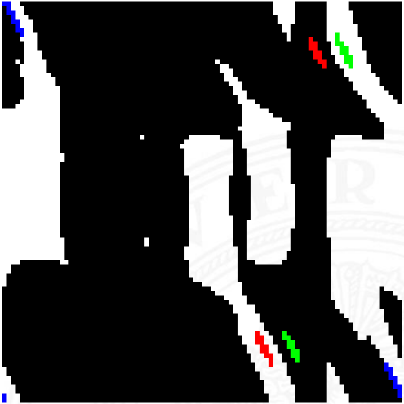


Discretized workspace
$xscale = 100,\ y^{scale} = 80$

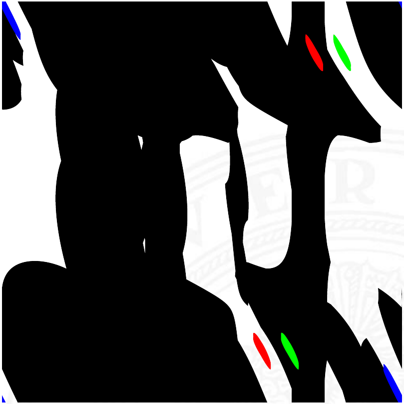
Work Space to Configuration Space – Example



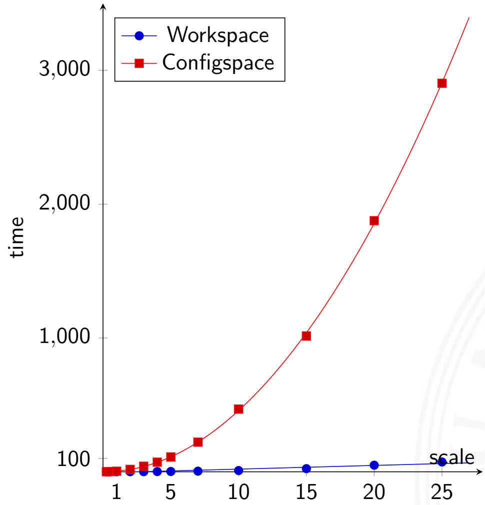
Discretized workspace $x^{scale} = 2000$, $yscale = 1600$



Discretized configuration space $q_1^{scale} = 90$, $q_2^{scale} = 90$

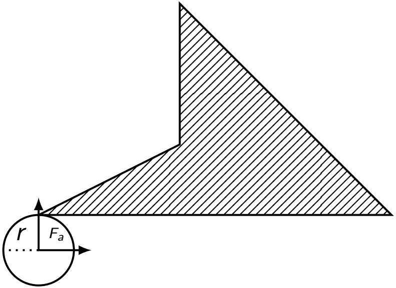


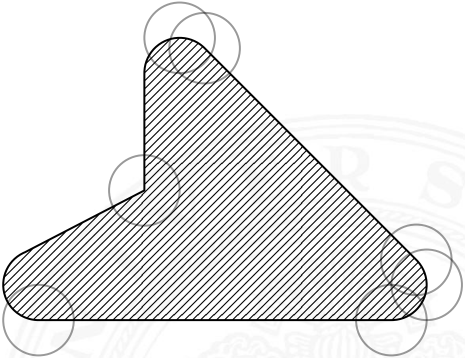
Discretized configuration space $q_1^{scale} = 3600$, $q_2^{scale} = 3600$



- Python
- Brute forward kinematics
- using polygon collisions
  - shapely library
- 56 cpus
  - Intel® Xeon® E5-2690 v4 (2.60GHz)

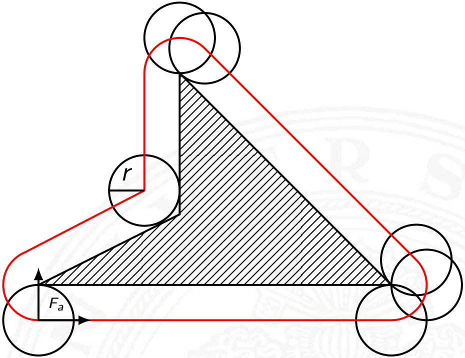
C-Obstacle for a circular artifact





Obstacle & artifact (radius r) Expanded C-Obstacle

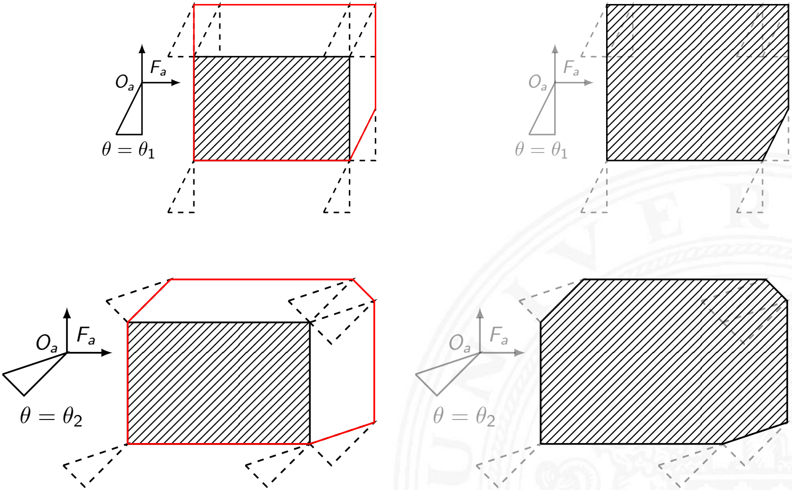
C-Obstacle for a circular artifact



Obstacle & artifact (radius r)

Path of minimal distance to obstacle

C-Obstacle for Polygons



Obstacle & polygon artifact with $\theta = \theta_1 \vee \theta_2$; minimum distance to obstacle.

A C-Obstacle of a fixed, convex obstacle with respect to a moving convex robot (part) may be theoretically represented as the `Minkowski Sum` of the corresponding objects.

$C_O(H)$ is the C-obstacle of a fixed convex polyhedra H, with respect to the (moving) convex object O.

Minkowski-Sum (Minkowski-Difference) of H and O (H and $-O$)

$$C_O(H) = H \ominus O = H \oplus (\ominus O)$$

where

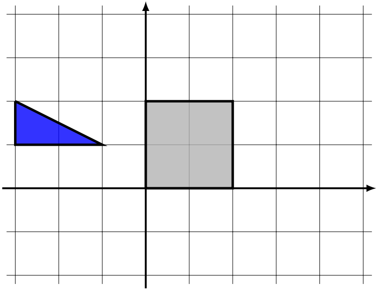
$$H \ominus O := \{h - o \mid h \in H \wedge o \in O\}$$

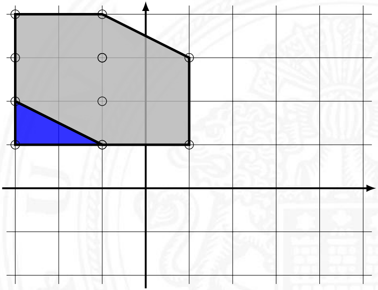
$$A = \{(0,0),(2,0),(2,2),(0,2)\} \qquad B = \{(-1,1),(-3,2),(-3,1)\}$$

$$A \oplus B = \{(-1,1),(-3,2),(-3,1),(1,1),(-1,2),(-1,1),\\ (1,3),(-1,4),(-1,3),(-1,3),(-3,4),(-3,3)\}$$

The convex hull (eliminating duplicates & inner points)

$$conv\{A \oplus B\} = \{(-3,1),(1,1),(1,3),(-1,4),(-3,4)\}$$





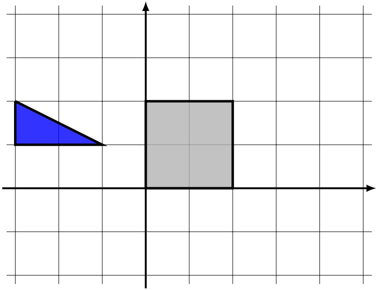
Minkowski Sum & Difference – 2D Example (cont.)

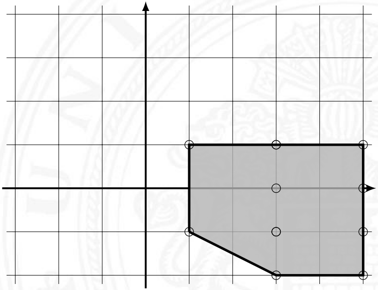
$$A = \{(0,0),(2,0),(2,2),(0,2)\} \qquad B = \{(-1,1),(-3,2),(-3,1)\}$$

$$A \ominus B = \{(1,-1),(3,-2),(3,-1),(3,-1),(5,-2),$$
$$(5,-1),(3,1),(5,0),(5,1),(1,1),(3,0),(3,1)\}$$

The convex hull (eliminating duplicates & inner points)

$$conv\{A \ominus B\} = \{(1,-1),(3,-2),(5,-2),(5,1),(1,1)\}$$

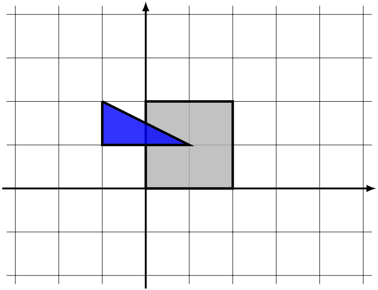


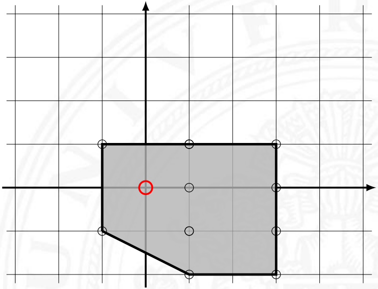


Minkowski Sum & Difference – 2D Example (cont.)

Collision detection

Two objects are colliding, if their Minkoswki difference contains the origin of the coordinate frame.

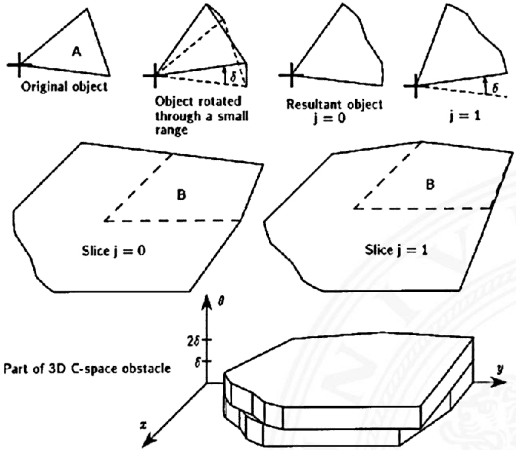




There is an interactive applet on the web:

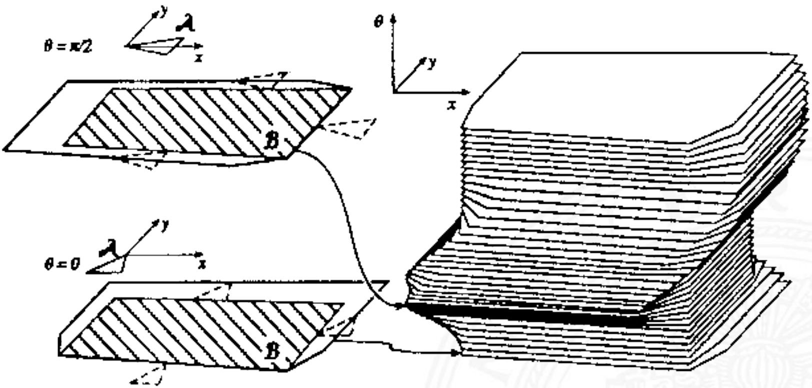
`http://www.cut-the-knot.org/Curriculum/Geometry/PolyAddition.shtml`

C-Obstacles for 2-D translation and 1-D rotation



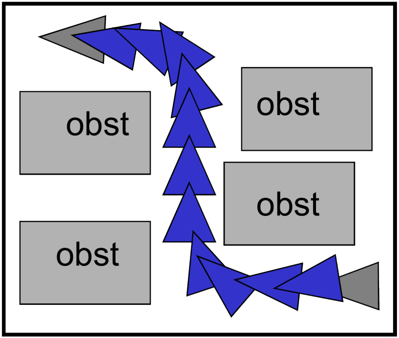
Represent rotational configuration of the C-obstacle as slice for each θ configuration of the robot.

C-Obstacles for 2-D translation and 1-D rotation (cont.)

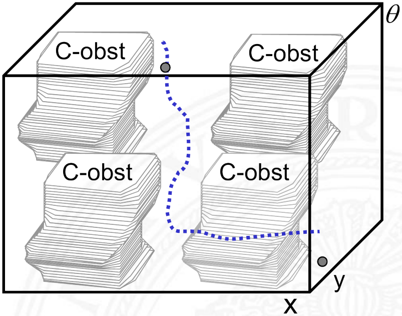


The configuration space for a *k*-DOF robot is a *k*-Dimensional coordinate system.

C-Obstacles for 2-D translation and 1-D rotation (cont.)

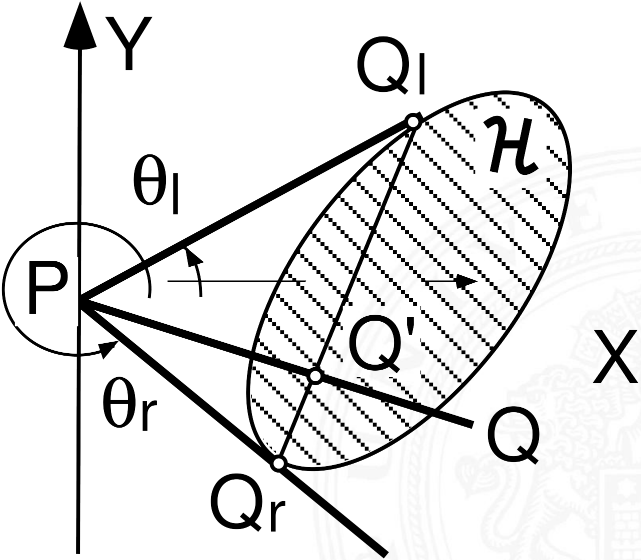


Work space (x, y)

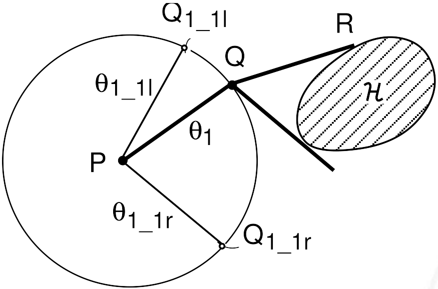


Configuration space (x, y, θ)

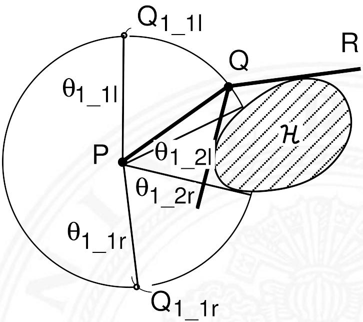
C-obstacles of a pole



C-obstacles of a 2-DOF Chain of Poles

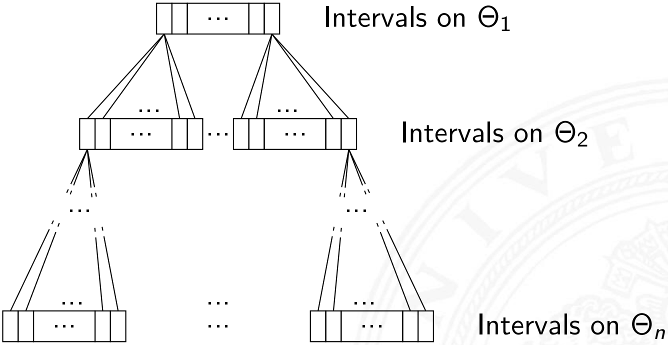


(1)



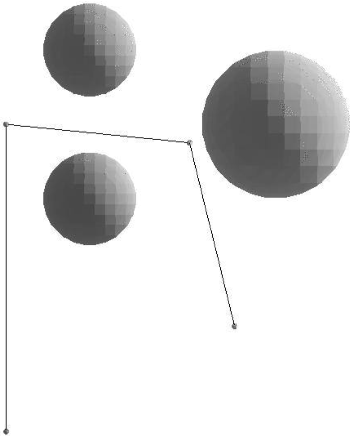
(2)

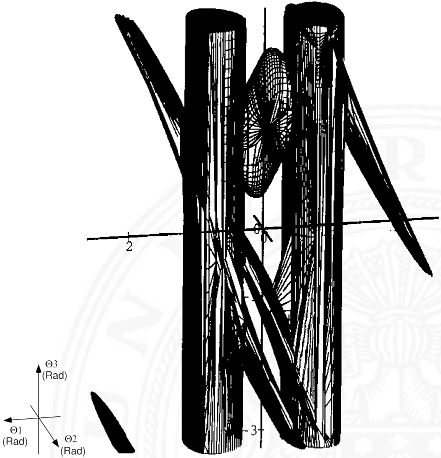
Tree-structure for Configuration Space partitioning



Curse of dimensionality

Configuration Space of a 3-DOF Chain of Poles





Partition Representation of C-Space

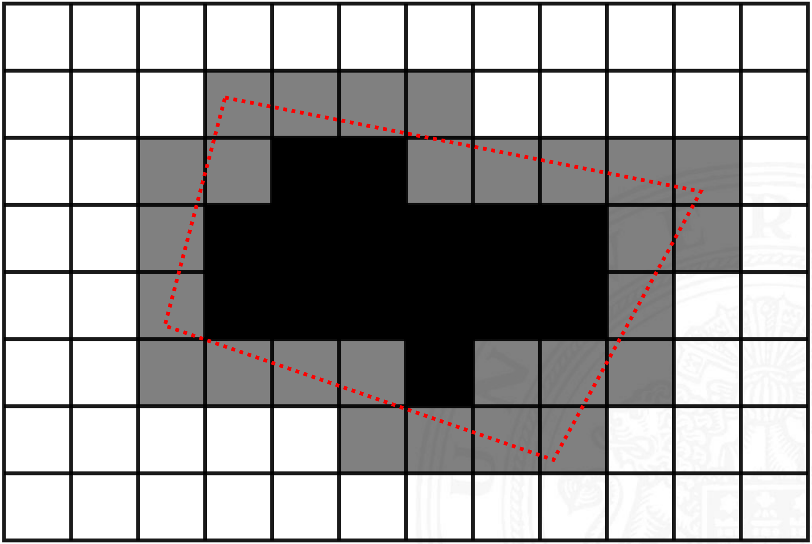
The free space is partioned into cells using

- Geometrical partition
  - uniform cubes
  - a hierarchical tree-structure (Quad-tree, Oct-tree, etc.)
  - slices and scanlines
  - `bubbles` of variable size

  The union of the non-overlapping cells is part of the free space. Neighborship graphs represent the connectivity of free space.
- Topological partition
  - overlapping generalized cones
  - critical points of the C-obstacle connection graph

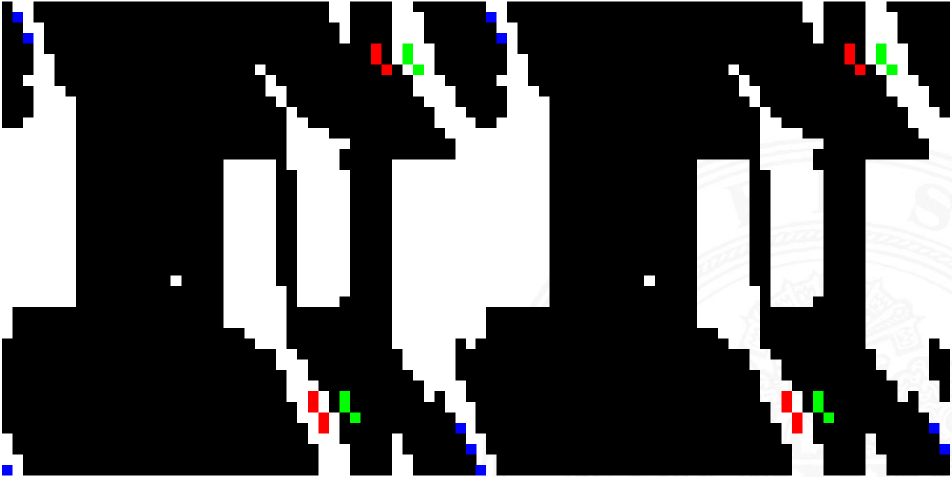
  The union of the overlapping cells is equal to the free space.

Squares-Partitioning of Configuration Space



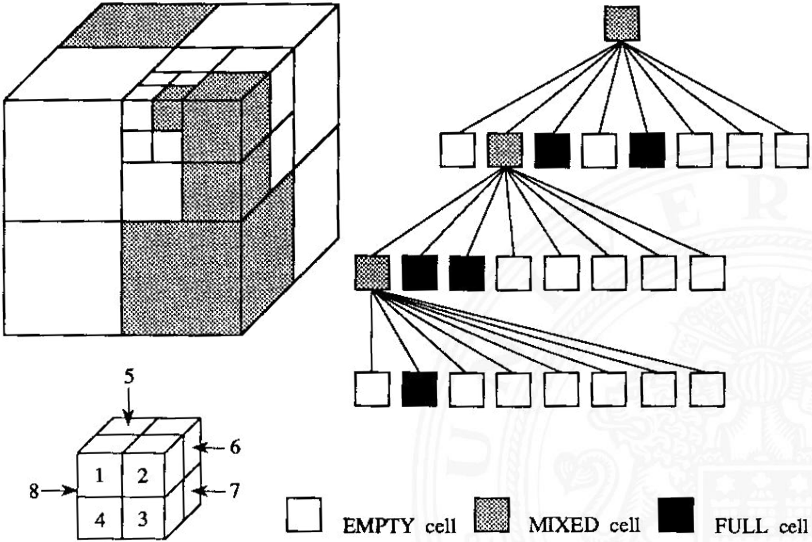
Resulting bitmap of configuration space using squares partitioning

Squares-Partitioning of Configuration Space (cont.)

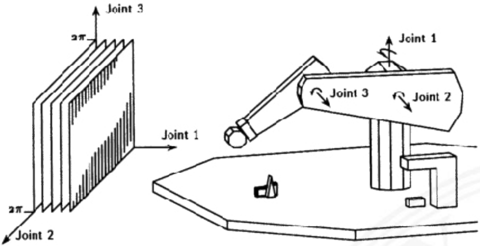


Bitmap of configuration space

Partitioning of the configuration space using Octrees



Partitioning of the configuration space using Slices

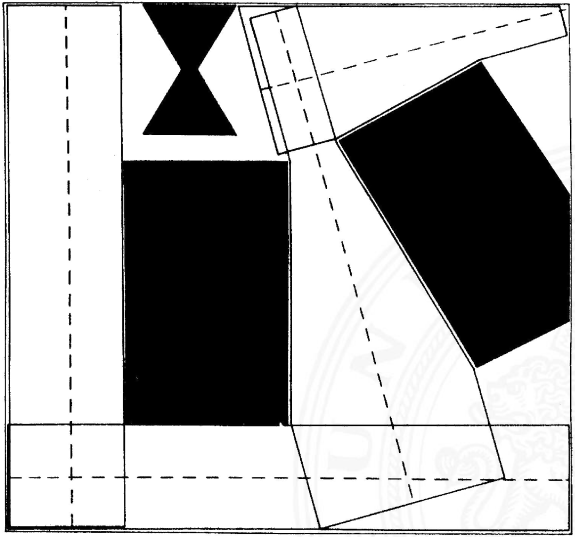


Complexity regarding the transformation of the C-obstacles

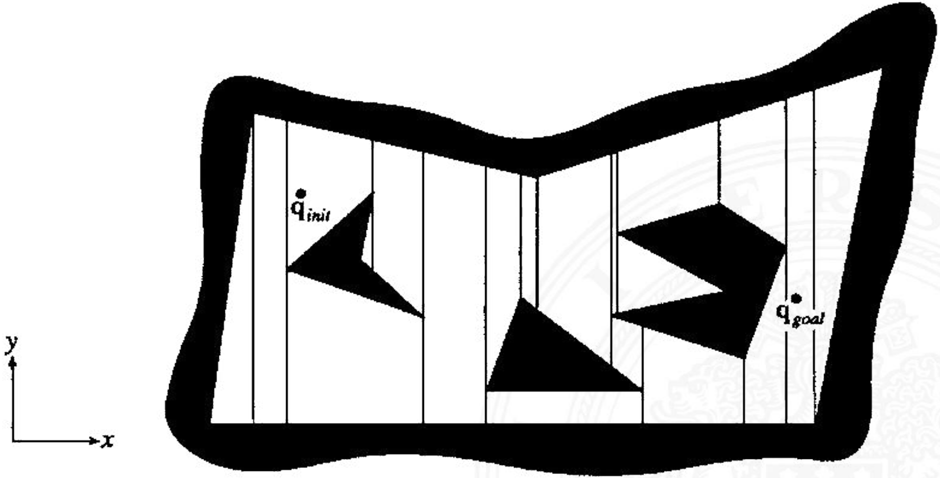
$$r^{d-1}f(m)$$

where r: the number of discretization steps for each DOF,
d: DOF of the robot arm
$f(m)$: the computing time of one slice
m: the number of edges of all obstacles

Representation of free space with generalized cones

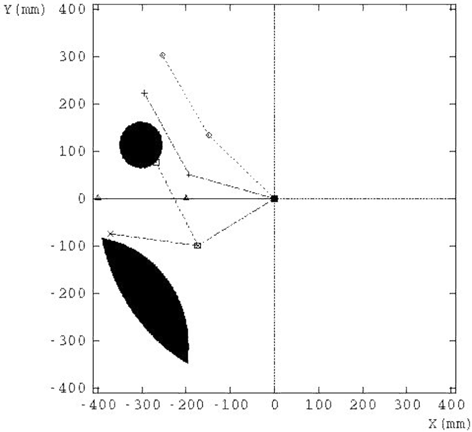


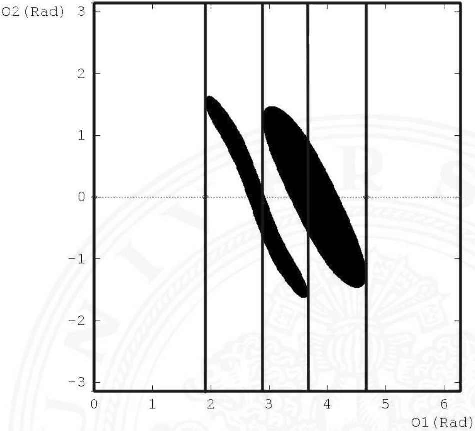
Exact Partition of Configuration Space



Trapezoidal partitioning of the configuration space

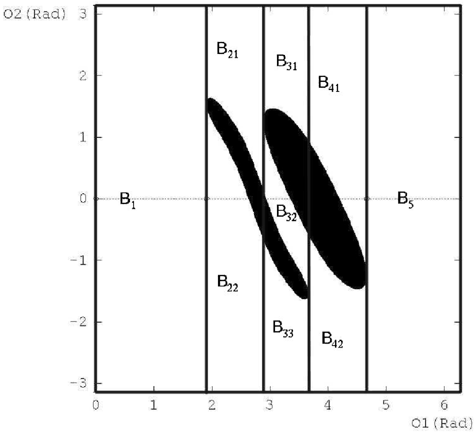
Exact Partition of Configuration Space (cont.)

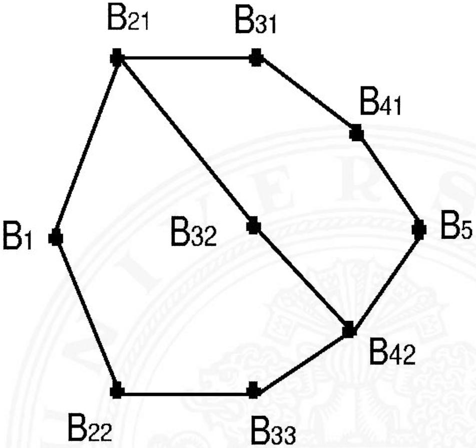




Cylindrical partitioning using critical points

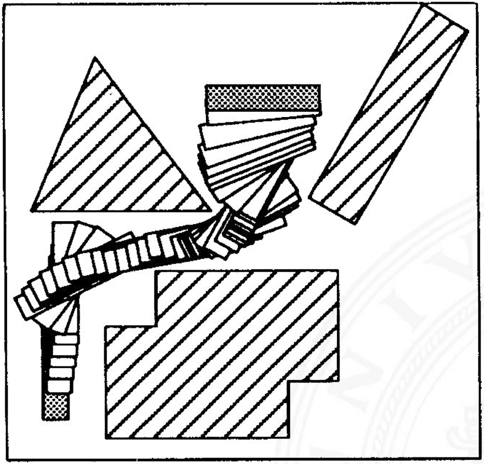
Exact Partition of Configuration Space (cont.)





Cylindrical partitioning and connectivity graph

Planning Results



[12]

Serial computing: 3-DOF C-space
Massive-parallel computing: up to 6-DOF C-Space

Partition based Path Planning

Advantages:

- Complete in case of sufficient resolution
- Global overview

Disadvantages:

- High demand for RAM
  - `Curse of Dimensionality`
- Complex to implement
- Practically implementable only for few degrees of freedom

$$\Downarrow$$

Path planning without explicit representation of free space?

$\Longrightarrow$ next Lecture!

Introduction to Robotics

Lecture 12

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

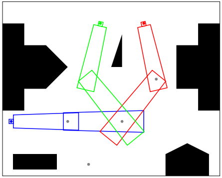
University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

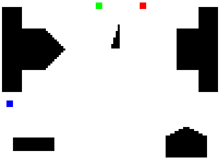
July 12, 2018

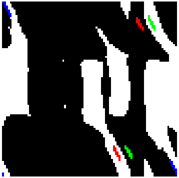
Outline

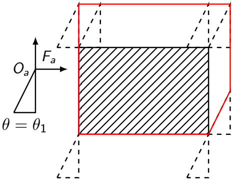
Outline (cont.)

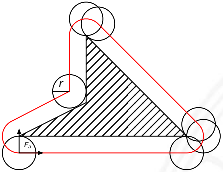
Partition based Path Planning – Methods

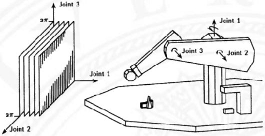


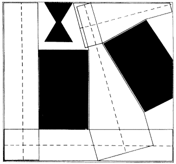


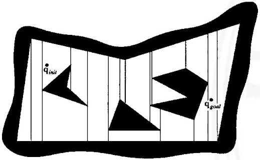


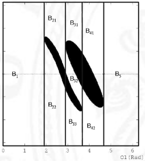


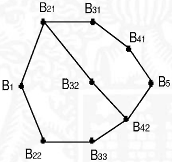












Partition based Path Planning

Advantages:

- Complete in case of sufficient resolution
- Global overview

Disadvantages:

- High demand for RAM
  - `Curse of Dimensionality`
- Complex to implement
- Practically implementable only for few degrees of freedom

$$\Downarrow$$

Path planning without explicit representation of free space?

$\Longrightarrow$ this Lecture!

Potential Field Method (PFM)

Definition

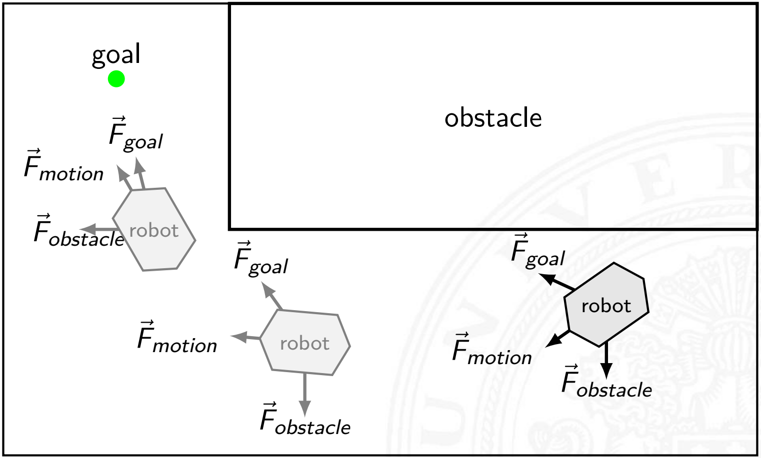
The manipulator moves in a field of forces. The position to be reached is an attracting pole for the end effector and obstacles are repulsive surfaces for the manipulator parts.

[13]

Basic Principle

- Initially developed for real-time collision avoidance
- Potential field associates a scalar value to every point in space
- An ideal field used for navigation should
  - be smooth
  - have only one global minimum
  - the values should approach ∞ near obstacles
- Force applied to the robot is the negative gradient of the potential field
- Robot moves along this force
- A function is defined in the free space, which has a global minimum at the goal configuration
- Motion follows steepest descend of the gradient

Basic Principle (cont.)



- The attracting force (of the goal)

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

  - where
    κ_ρ is a gain factor
    $(\mathbf{x} - \mathbf{x}_{Goal})$ is the distance between current and goal position

Basic Principle (cont.)

- The potential field (of obstacles)

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

  - where
    η is a constant gain factor
    $\rho(\mathbf{x})$ 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

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

  - where
    $\frac{d\rho(\mathbf{x})}{d\mathbf{x}}$ is the partial derivative vector of the distance from the point to the obstacle. This way, the direction of the force vector is expressed

[13]

Advantages and Disadvantages of PFM

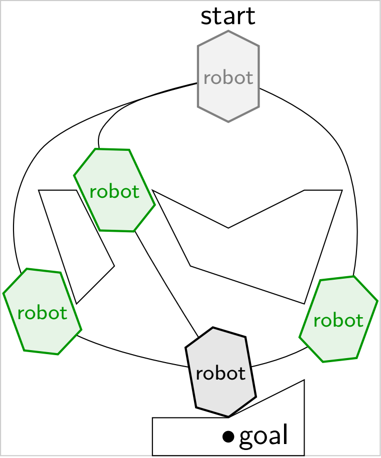
Advantages:

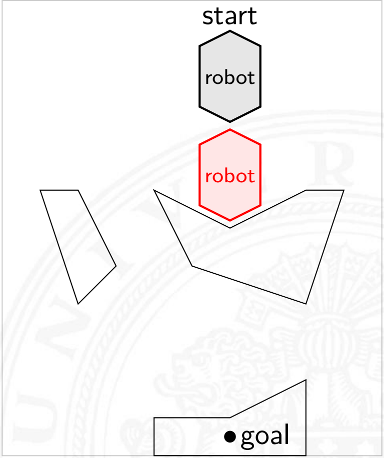
- Usage of heuristics
- Real-time capable

Disadvantages:

- Completeness
  - existing solution might not be found
  - calculation might not terminate if no solution exists
- Problem with multiple local minima may occur often
- No formal proof of capabilities
- No further constraints can be considered

Local Minima of PFM





Probabilistic Approaches

Demand for an efficient (i.e. fast, robust, easy to implement) framework to plan robot motion supporting high DOF.

Ideas:

1. Random samples in the region of interest
2. Test the samples for collisions
3. Connect samples using simple trajectories
4. Search in the resulting graph

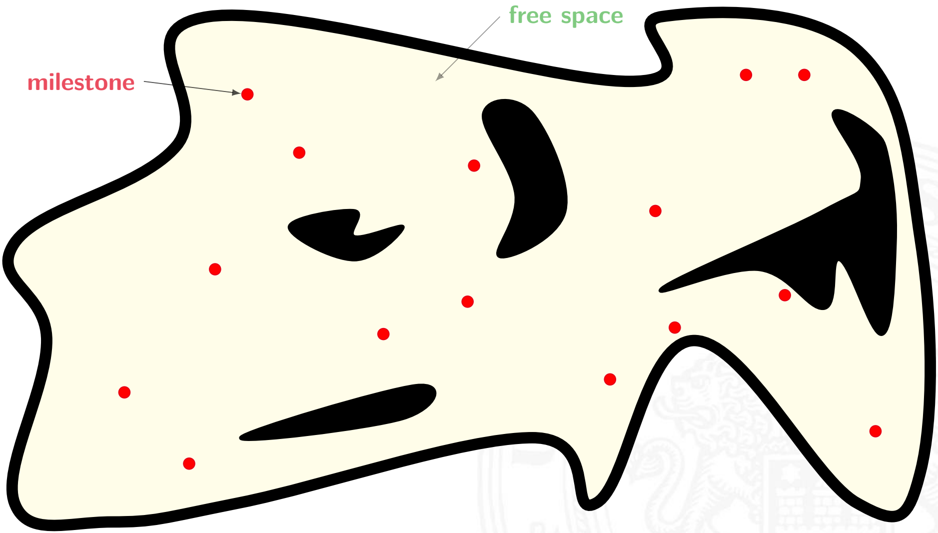
Motivation

Collision detection and distance estimation are faster than the generation of an explicit representation of free space.

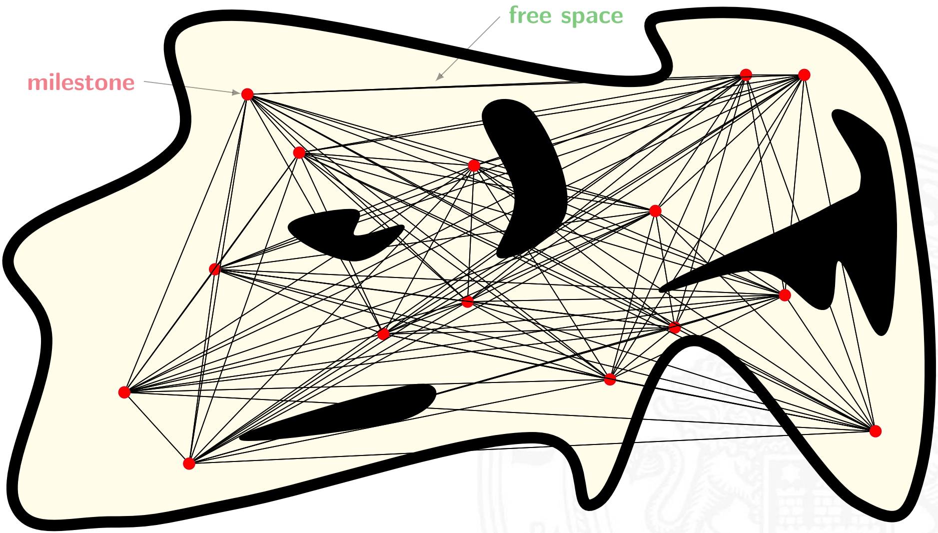
$\Longrightarrow$ Probabilistic Roadmaps [14]

Milestones and Roadmaps

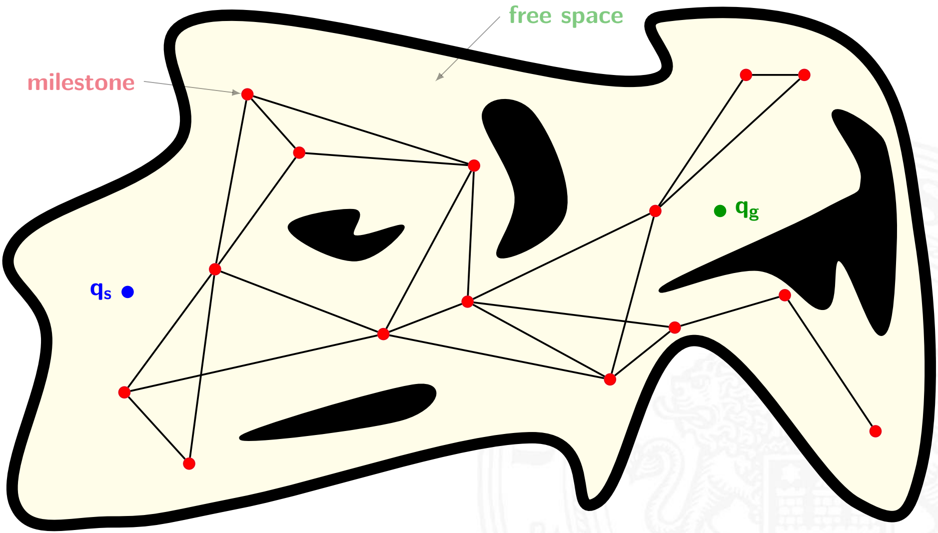
Milestones and Roadmaps



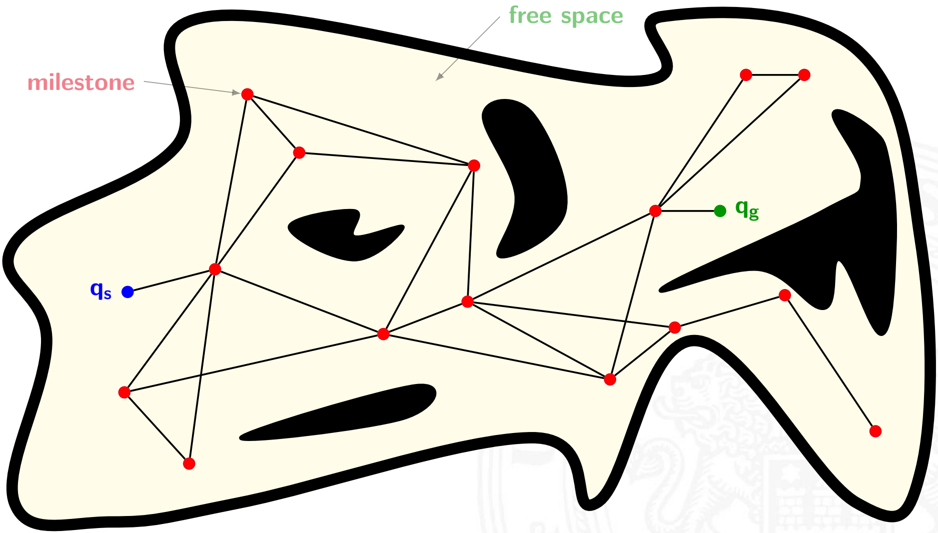
Milestones and Roadmaps



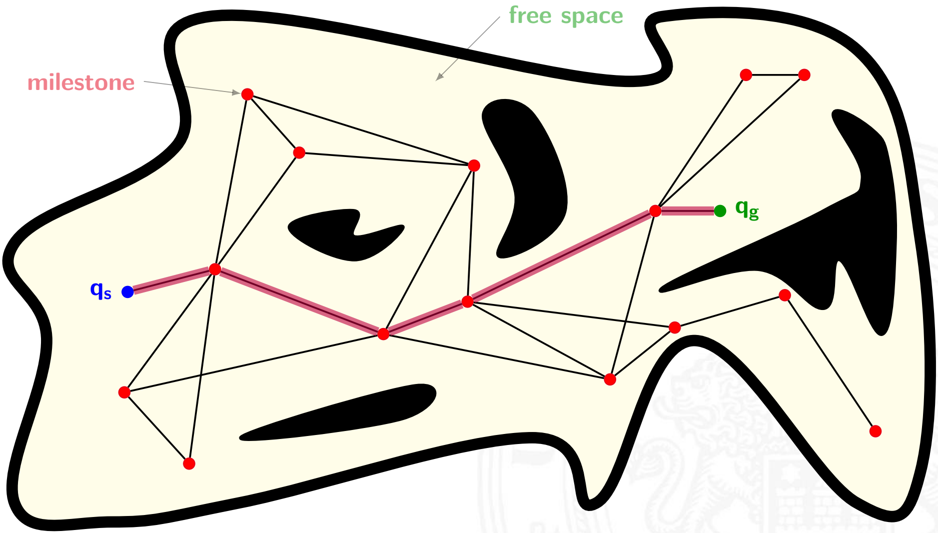
Milestones and Roadmaps



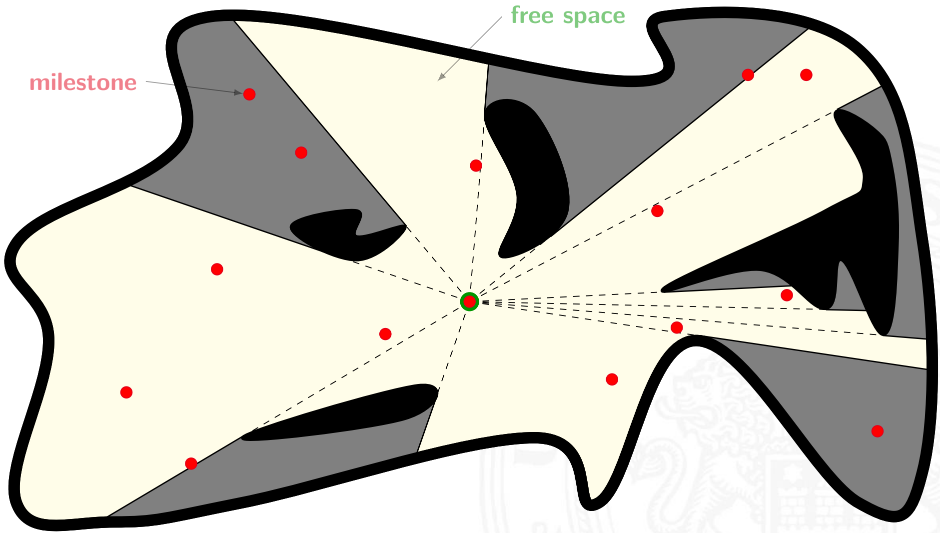
Milestones and Roadmaps



Milestones and Roadmaps



Parallels to the Art-Gallery-Problem



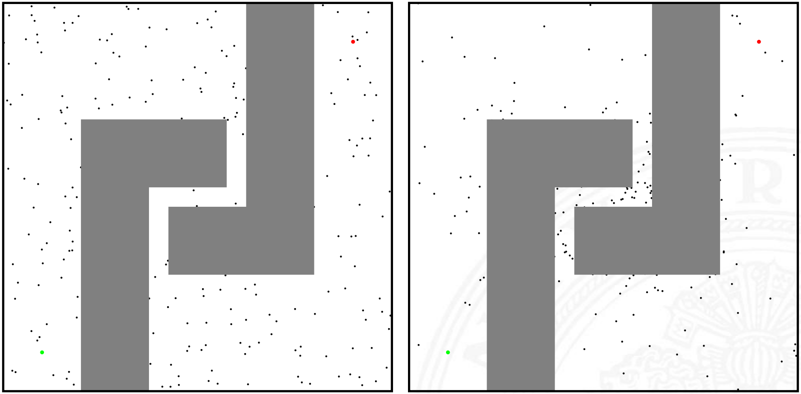
Strategies of Taking Samples

Problem 99% computation time of a probabilistic roadmap planner is used for collision checks.
Solution Intelligent strategy to reduce the size of the roadmap and thus the time for collision checks?

- Multi- vs single-exploration strategy
- Uniform
- Multi-level (coarse to fine)
- Obstacle-aware (shift colliding sample to free space)
- Lazy collision checks
- Probabilistic default values

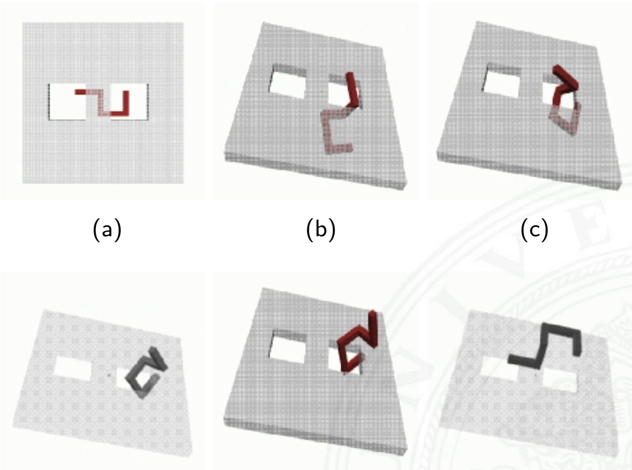
[15]

Process of taking Samples



In an expansive free space: $P_{fail} \sim e^{-N}$
where N: the number of milestones

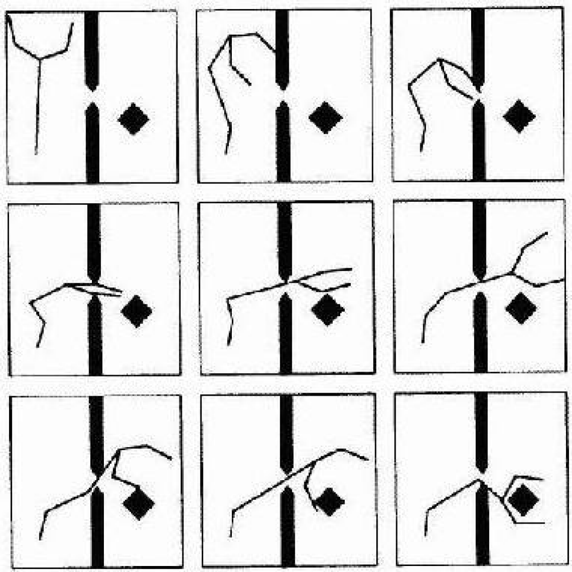
Successful 6D plan for narrow passages



(a) (b) (c)

(d) (e) (f)

Planning Results for a multi-joint artifact



Summary Probabilistic Approaches

Disadvantages

- No strict termination criteria, if no solution can be found
  - only probabilistic completeness (an existing solution will eventually be found...)
- Missing insight to planning process

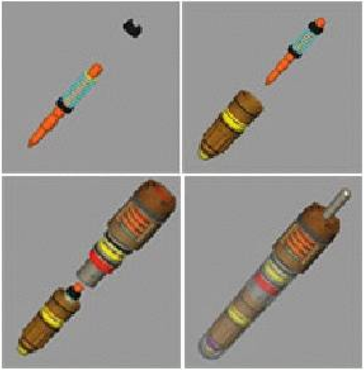
Advantages

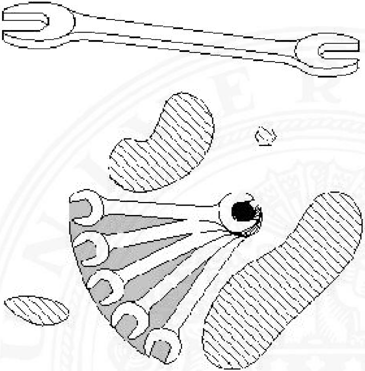
- Easy to implement
- Fast, scalable for problems with high DOF
- Rate of convergence increases with milestones

Application fields

- Production: robot programming, assembly, layout planning
- Sequence generation for maintenance tasks
- Autonomous mobile robots
- Graphical animations
- Motion planning for medical appliances
- Simulation of realistic paths of cells and molecules
- ...

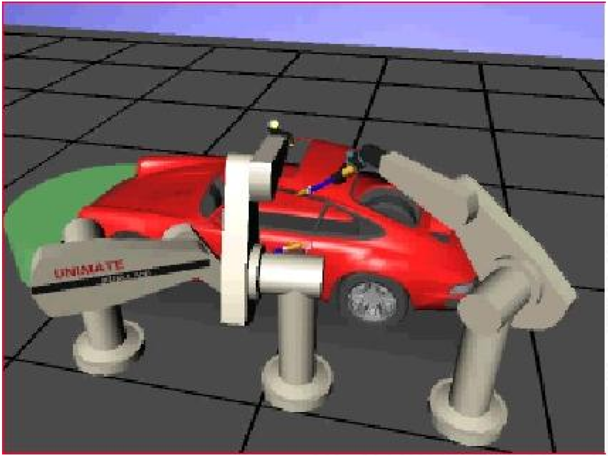
Using a path planner, the complexity of a product can be assessed. The assembly-process can be planned.



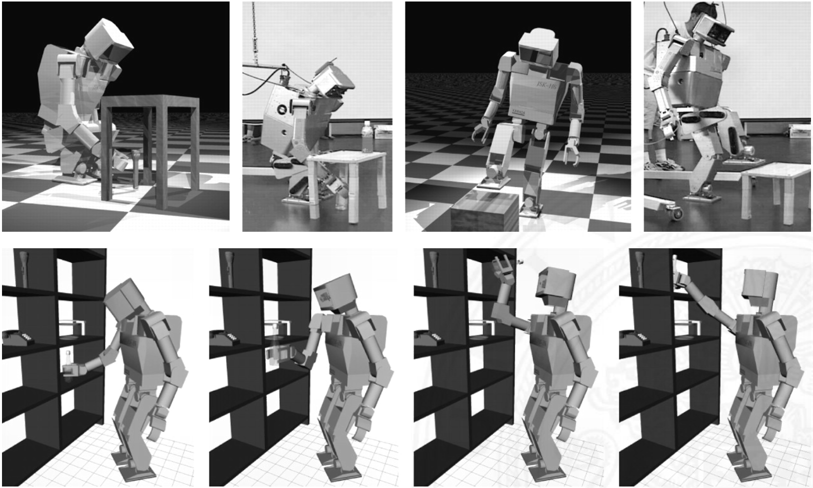


Layout Planning

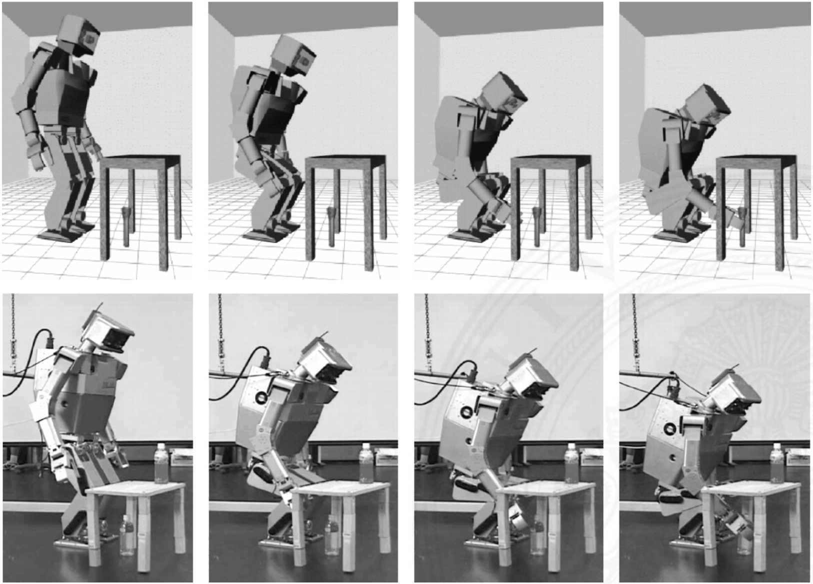
Path planning combined with optimization methods generate optimal positioning of robots and other equipment in a work cell.



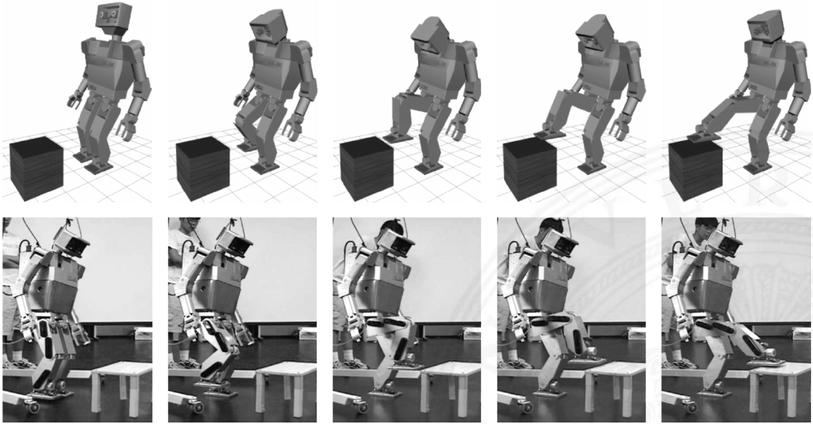
Humanoid



Humanoid (cont.)



Humanoid (cont.)

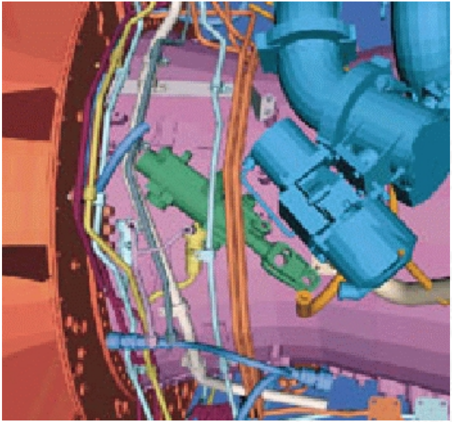


[16]

High DOF path planning is required for humanoid motion

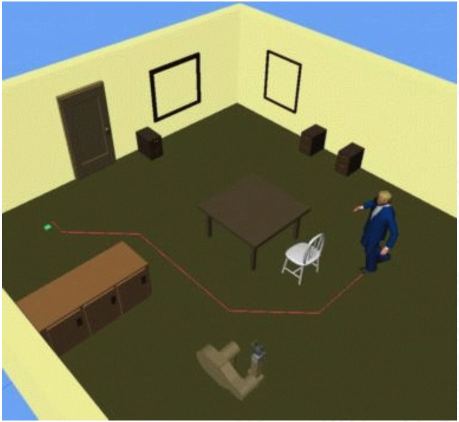
Engine Maintenance

Path planner can be used to automatically check the disassembly methods of parts.
This way the products can be easier maintained and repaired.

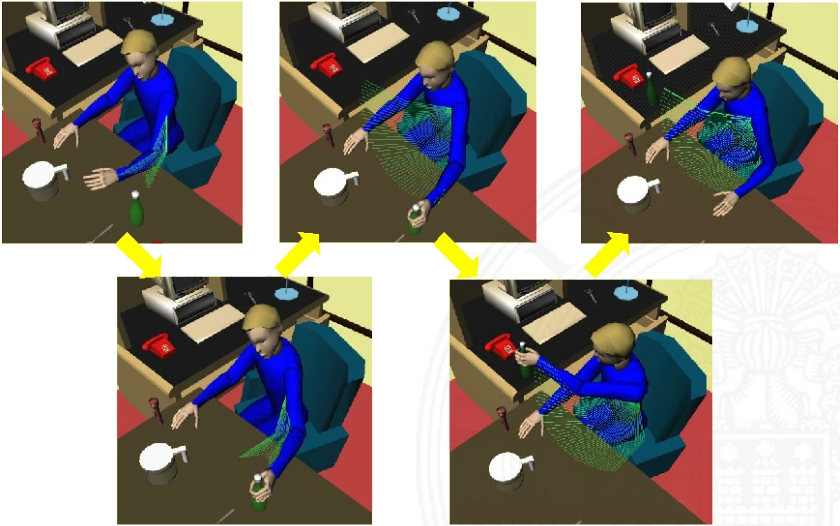


Animation of Task Oriented Programming

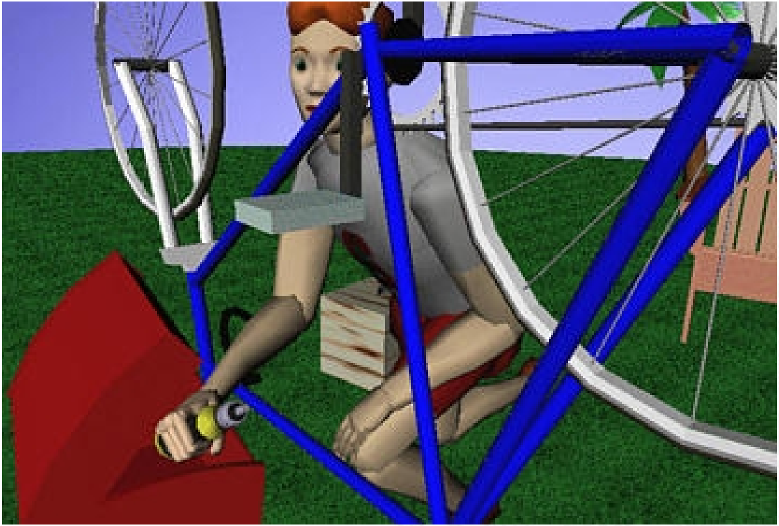
Simulation and visualization gives insight to path planning resulting from task oriented programming.



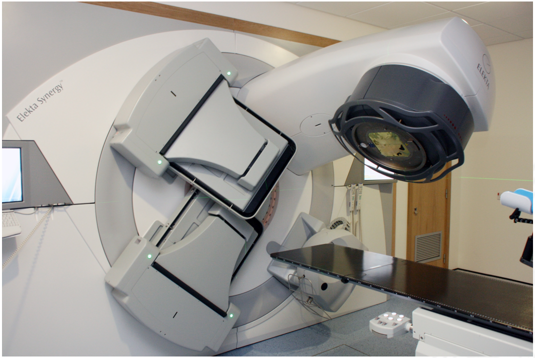
Animation of Manipulation Scripts

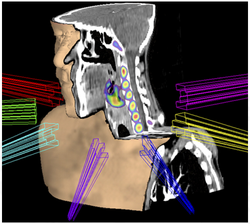


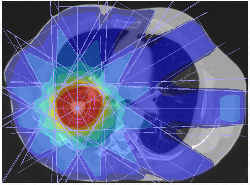
Animation as Simulation



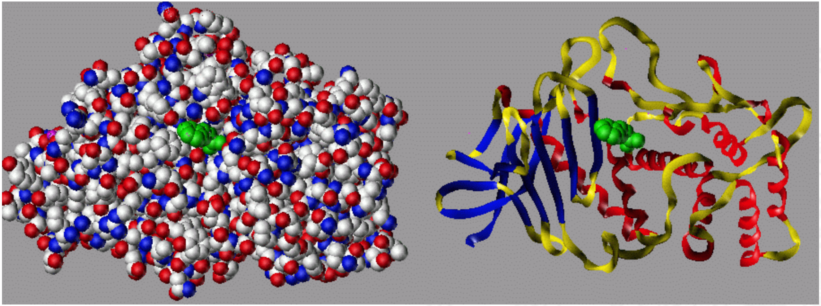
Planning of Radiotherapy







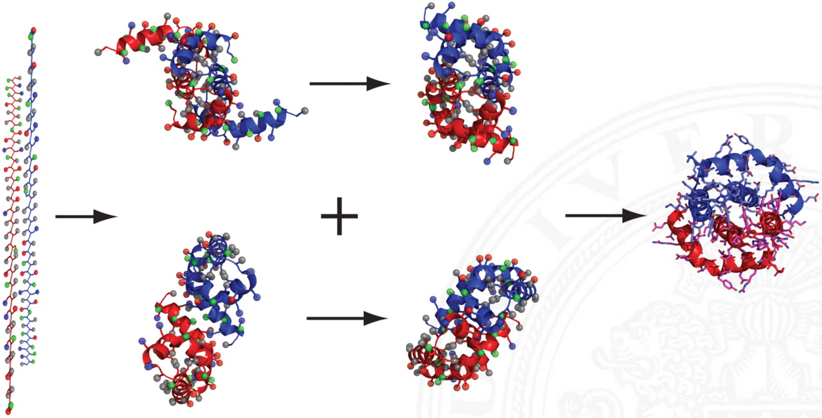
Generation of Docking Motion of Molecules



Generation of Docking Motion of Molecules (cont.)

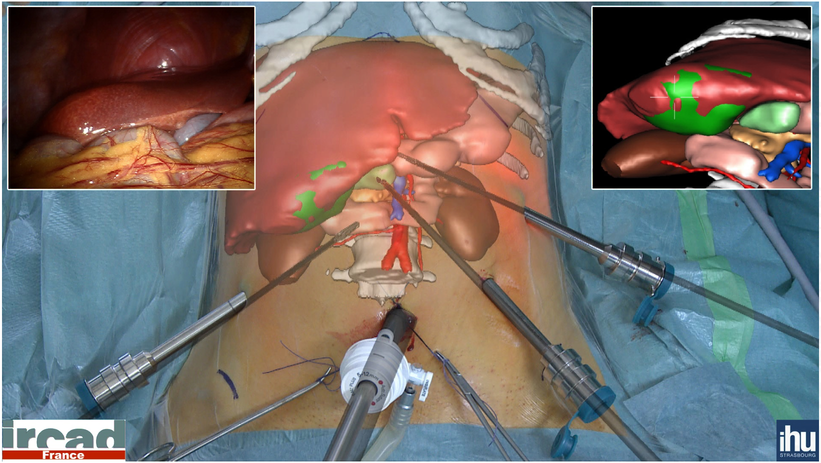
- moving obstacles
- multiple moving objects
- objects with deformable shape
- unspecified goals
- non holonomic constraints
- dynamic constraints
- planning for optimal time
- fuzzy sensing and plan execution
- highly complex artifacts

Handling of over 1000 degrees of freedom



Skip next slide if sensible to blood and organs.

Path planning for soft objects



Autonomous Virtual Actors


A Bug's Life (1998, Disney/Pixar)


Antz (1998, DreamWorks/PDI)


Toy Story 3 (2010,Disney/Pixar)


Final Fantasy VIII (1999, Square)


Tomb Raider 5 (2000, Eidos Interactive)


The Legend of Zelda: Skyward Swords (2011, Nintendo)

Summary

- Explicit representation of configuration space yields a complete solution
  - for sufficient resolution/precision
  - applicability is limited
- Distributed probabilistic approach for high DOF
- Path planning is native in the field of robotics
  - widely used in other fields
    - manufacturing, VR, animation, gaming, biology, chemistry, ...
- Simulated environments fulfill the requirements of geometrical path planning
  - known models of the environment
  - specified start and goal configurations
  - ideal execution

Summary (cont.)

- Increasing computation power allows real time application
- Real robots face various uncertainties in the environment
  - Extension of basic problem requires additional research
- Embedded (robotic) systems get more and more powerful
  - motion modeling and calculation of intelligent devices open new fields of research

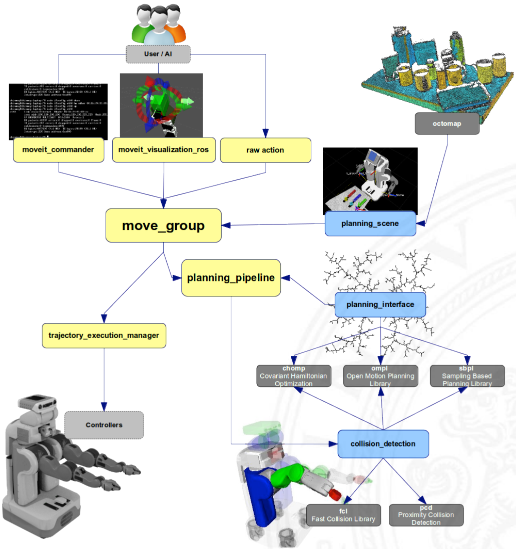
Open Motion Planning Library (OMPL)

- Library of sampling based motion planning algorithms
- Integrated in ROS arm navigation stack (used on the PR2)
- Integrated in MoveIt! project
- Includes state-of-the-art motion planning algorithms
- No collision checking
- Demo videos at
  `http://ompl.kavrakilab.org/gallery.html`
- Tutorials on how to integrate OMPL at
  `http://wiki.ros.org/ompl_ros_interface/Tutorials`

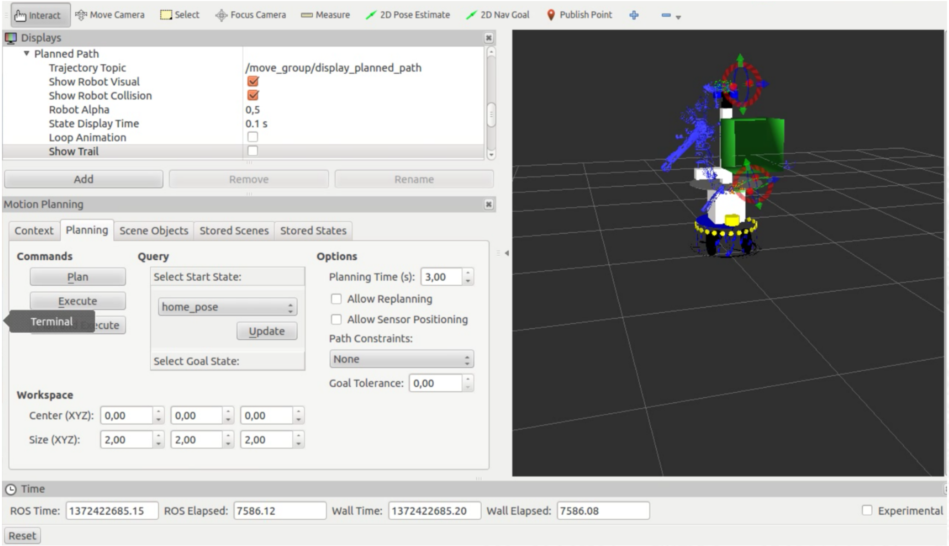
MoveIt! - A Planning Framework

- Features
  - kinematics
  - dynamics
  - collision
  - checking
  - constraint evaluation
  - visualization
  - ...
- Tools
  - motion plan specification
  - configuration
  - debugging
  - visualization
  - benchmarking
  - ...
- Planning and executing motion plans for different robots
- Overview at `http://moveit.ros.org`

MoveIt! - A Planning Framework (cont.)



MoveIt! - A Planning Framework (cont.)



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

Introduction to Robotics

Lecture 13

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

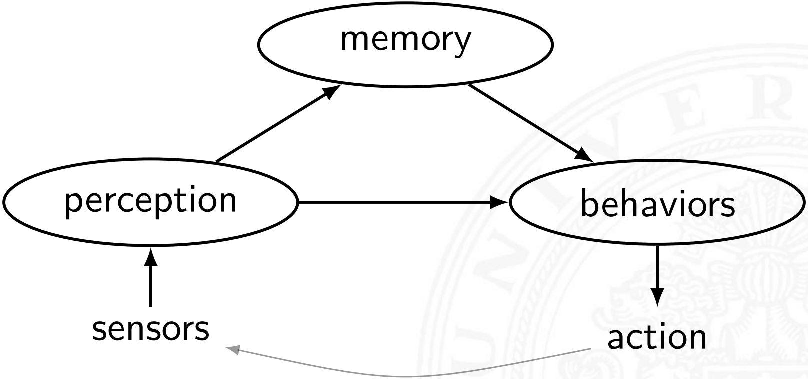
Outline

Outline (cont.)

Overview

- Basic behavior
- Behavior fusion
- Subsumption
- Hierarchical architectures
- Interactive architectures

The Perception-Action-Model with Memory



CMAC-Model

CMAC: `Cerebellar Model Articulation Controller`

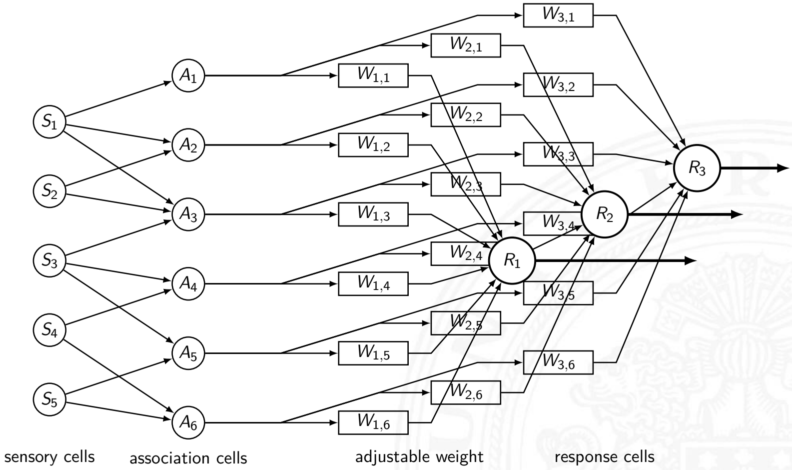
- **S** sensory input vectors (firing cell patterns)
- **A** association vector (cell pattern combination)
- **P** response output vector ($\mathbf{A} \cdot W$)
- W weight matrix

The CMAC model can be viewed as two mappings:

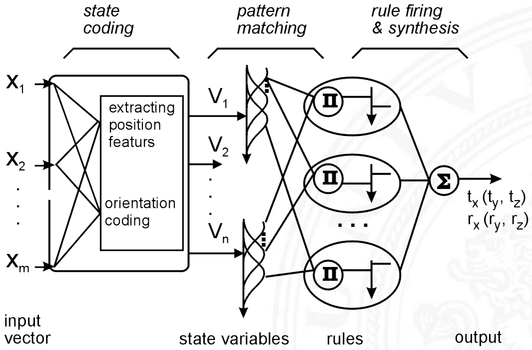
$$f : \mathbf{S} \longrightarrow \mathbf{A}$$

$$g : \mathbf{A} \xrightarrow{W} \mathbf{P}$$

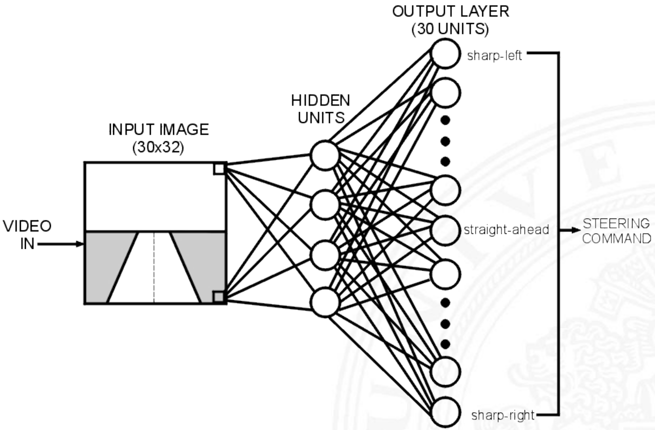
CMAC-Model (cont.)



The B-Spline model is an ideal implementation of the CMAC-Model. The CMAC model provides an neurophysiological interpretation of the B-Spline model.

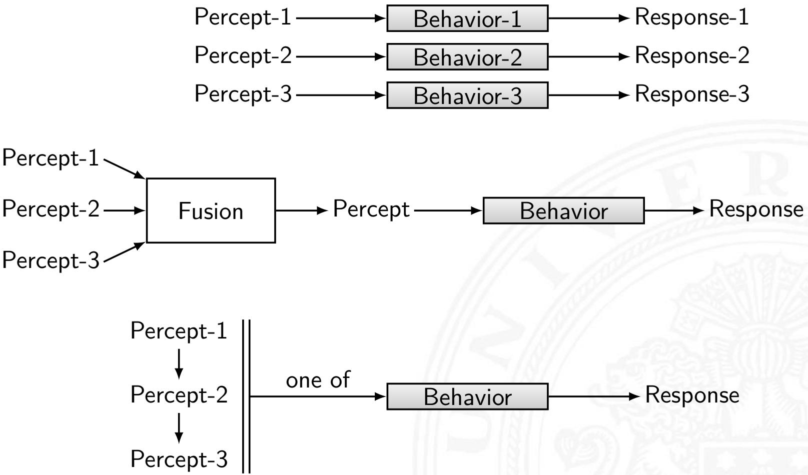


Alvinn – Visual Navigation



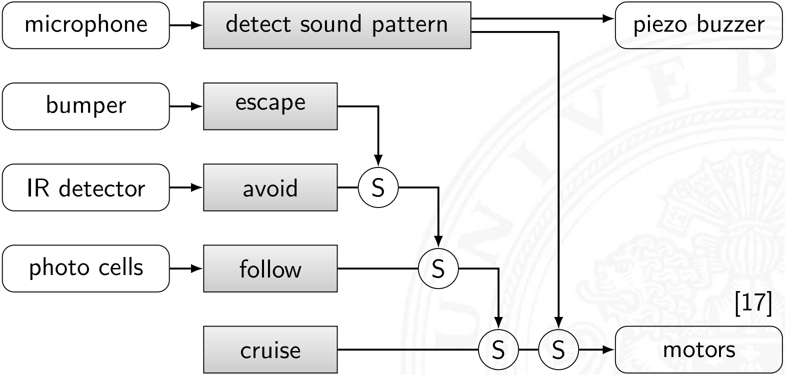
CMU – Carnegie Mellon University

Action-oriented Perception



The Subsumption Architecture

- hierarchical structure of behavior
- higher level behaviors subsumpe lower level behaviors

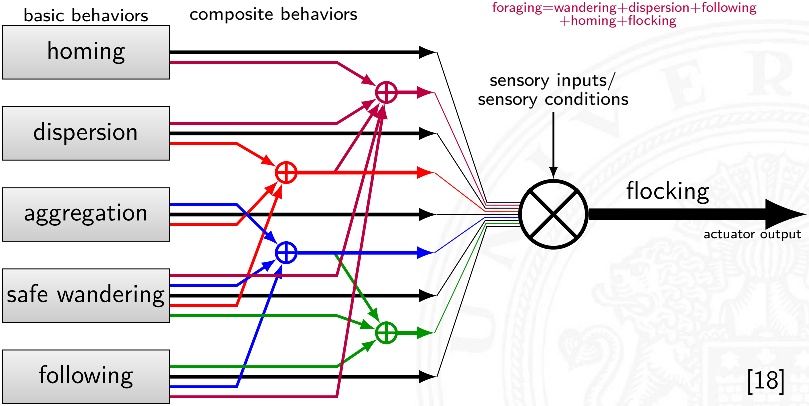


[17]

Foraging and Flocking

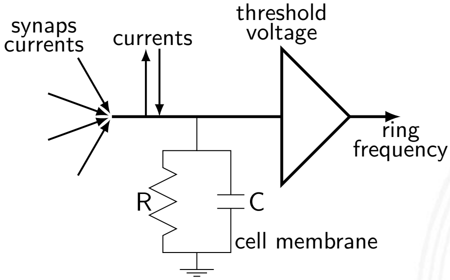
- multi-robot architecture
- basic behaviors are sequentially executed

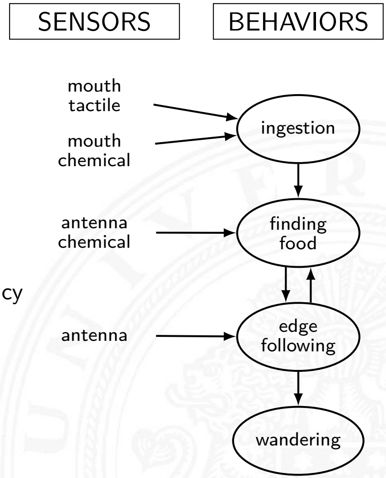
flocking=wandering+aggregation+dispersion
surrounding=wandering+following+aggregation
herding=wandering+surrounding+flocking
foraging=wandering+dispersion+following
+homing+flocking



[18]

Cockroach Neuron / Behaviors





Control Architecture of a Fish

Control and information flow in artificial fish

Perception sensors, focuser, filter

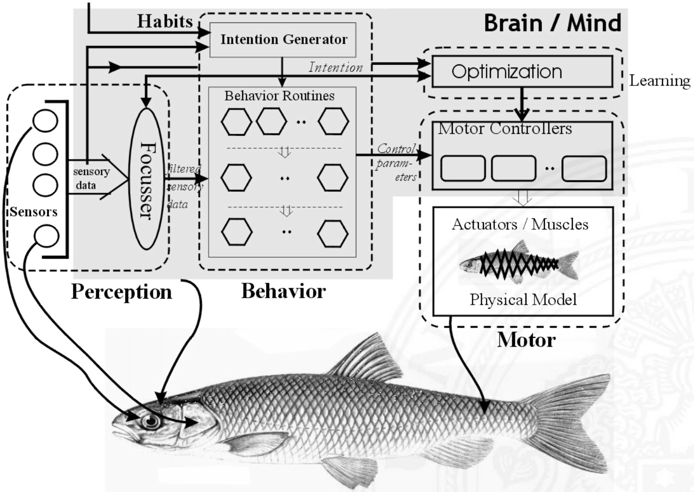
Behaviors behavior routines

Brain/mind habits, intention generator

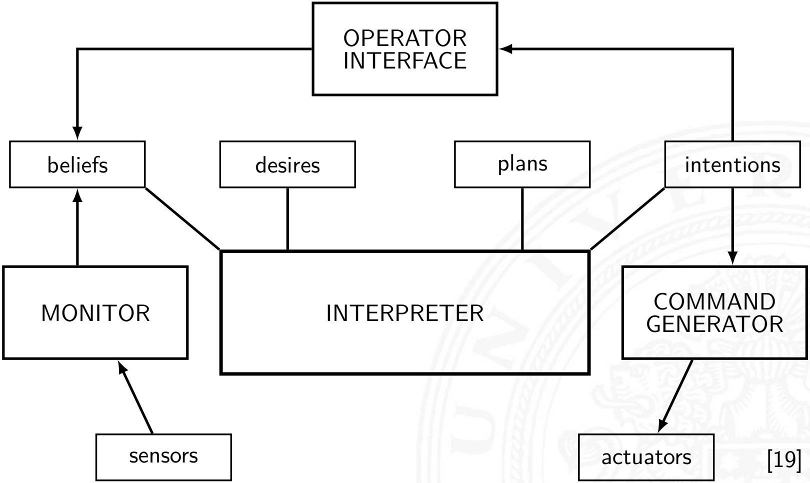
Learning optimization

Motor motor controllers, actuators/muscles

Control Architecture of a Fish (cont.)

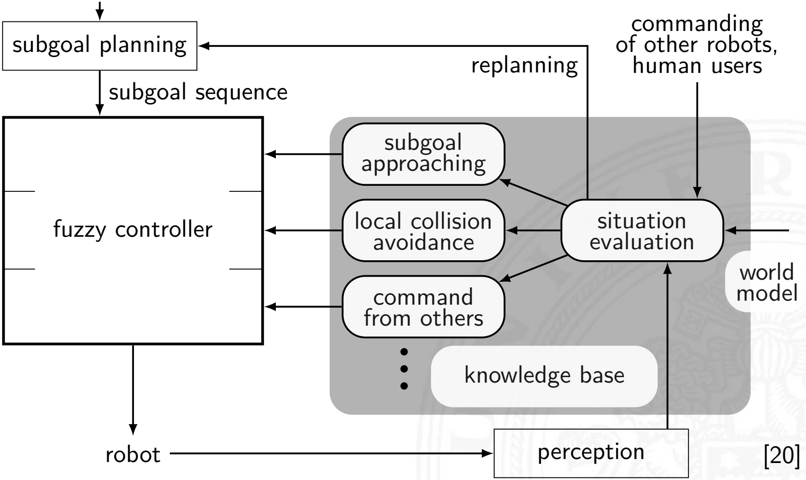


Procedural Reasoning System



[19]

Hierarchical Fuzzy-Control of a Robot



[20]

Fuzzy rules evaluate current situation.

Situation evaluation determines 3 fuzzy-parameters

- the priority K of the LCA rule base
- the replanning selector
- `NextSubgoal` (whether a subgoal has been reached)

Typical rule IF (*SL*85 IS HIGH) AND (*SL*45 IS VL) AND (*SLR*0 IS VL) AND (*SR*45 IS VL) AND (*SR*85 IS VL) THEN (*Speed* IS LOW) AND (*Steer* IS PM) K IS HIGH AND *Replan* IS LOW

Translation If the leftmost proximity sensor detects an obstacle which is near and the other sensors detect no obstacle at all, then steer halfway to the right at low speed. Mainly perform obstacle avoidance. No re-planning required.

Coordination of multiple rule bases

$$Speed = Speed_{LCA} \cdot K + Speed_{SA} \cdot (1 - K)$$

$$Steer = Steer_{LCA} \cdot K + Steer_{SA} \cdot (1 - K)$$

LCA: Local Collision Avoidance SA: Subgoal Approach

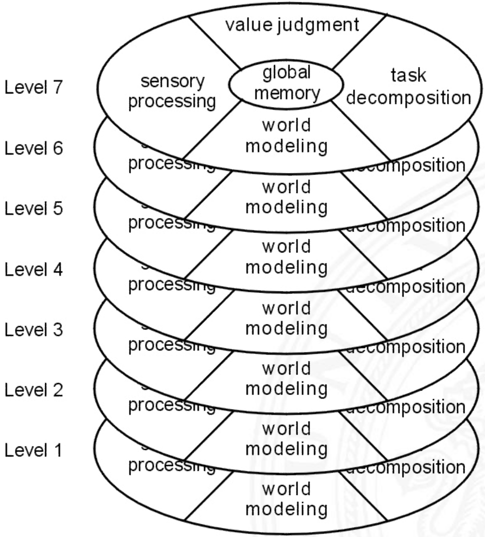
Real-Time Control System (RCS)

- 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.

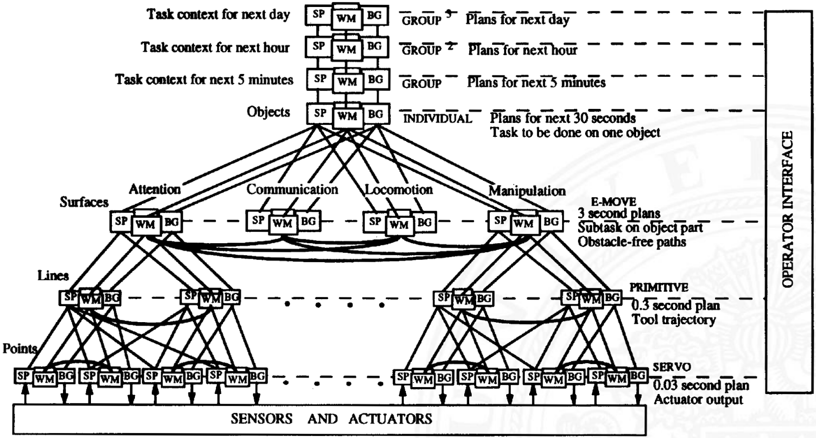
[21]

Examples of functional characteristics of the BG and WM modules:

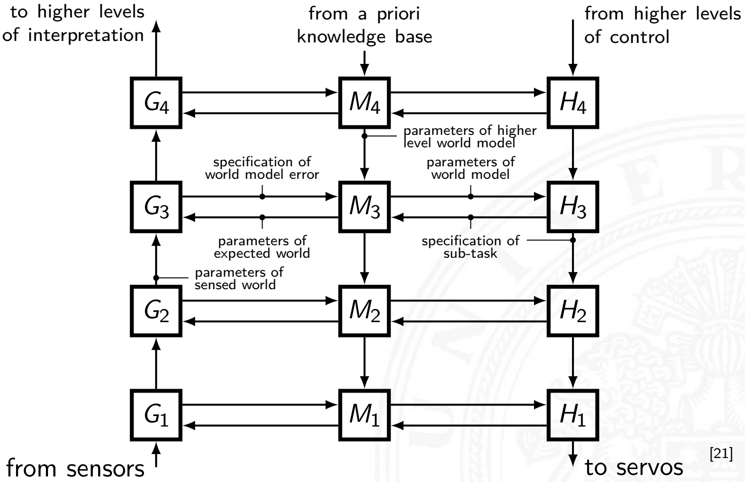
Hierarchy (cont.)



Hierarchy (cont.)

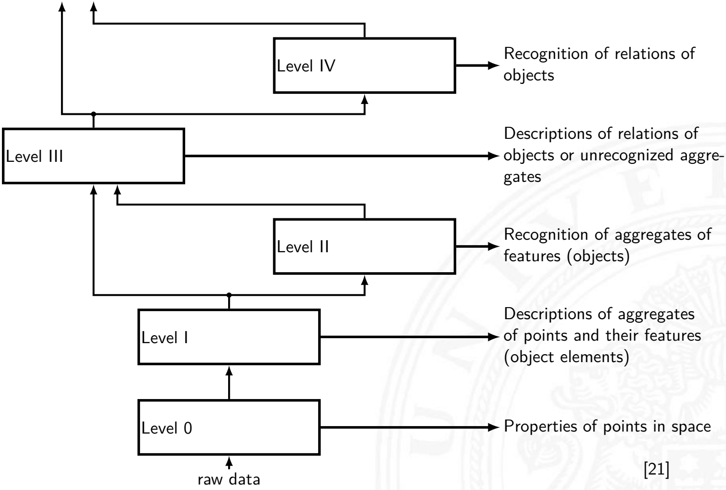


Hierarchy (cont.)



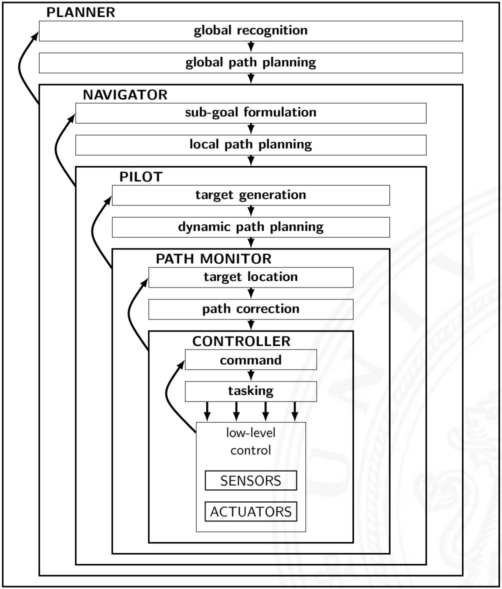
[21]

Sensor-Hierarchy



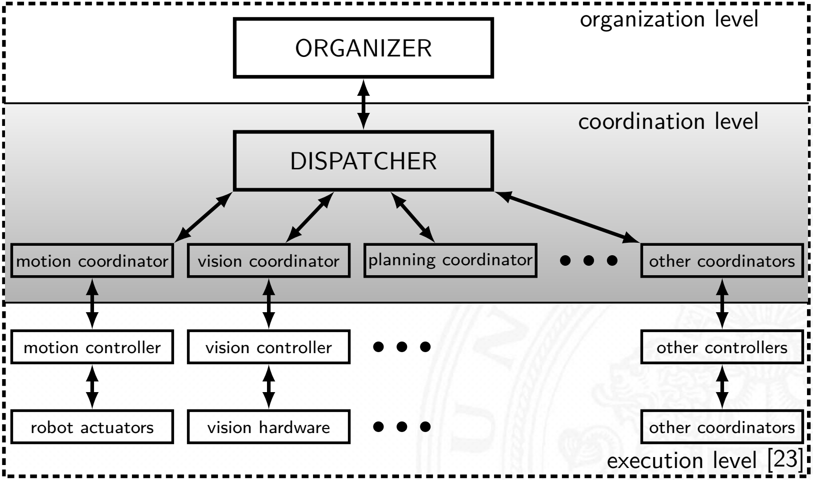
[21]

Other examples

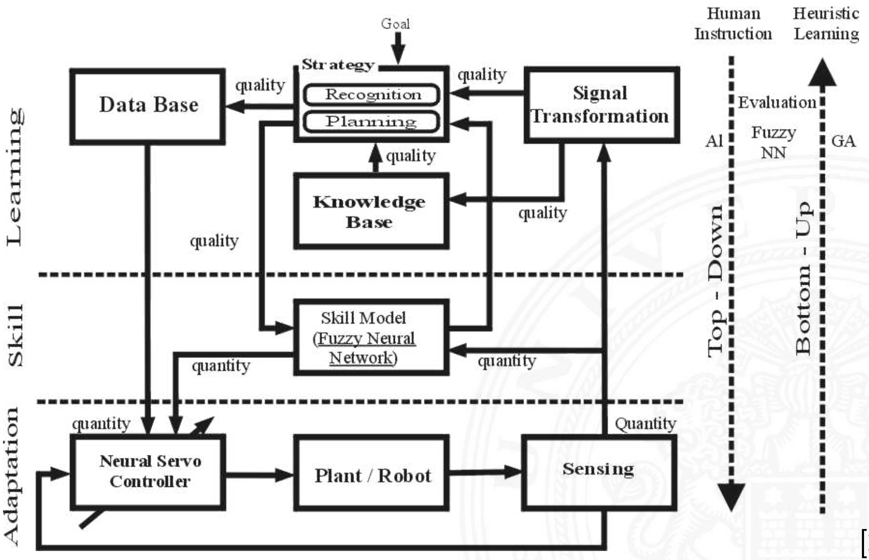


[22]

Other examples

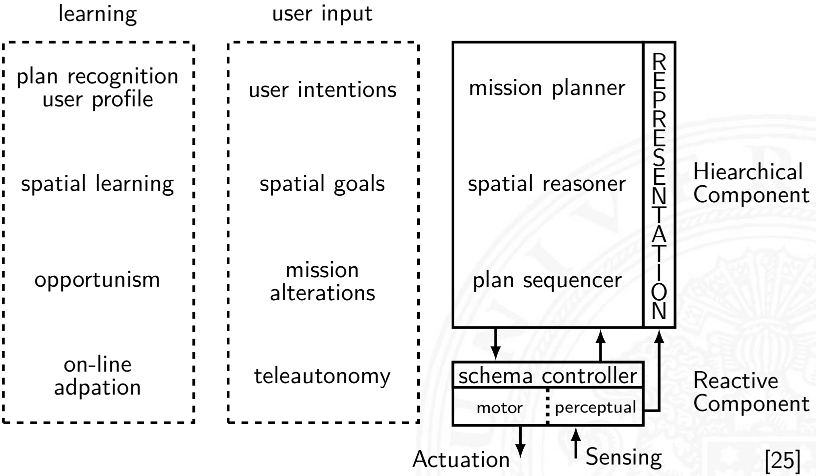


An Architecture for Learning Robots



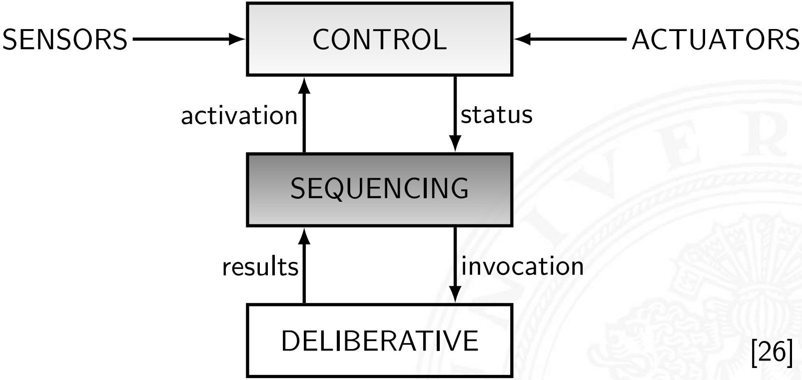
[24]

AuRA Architecture



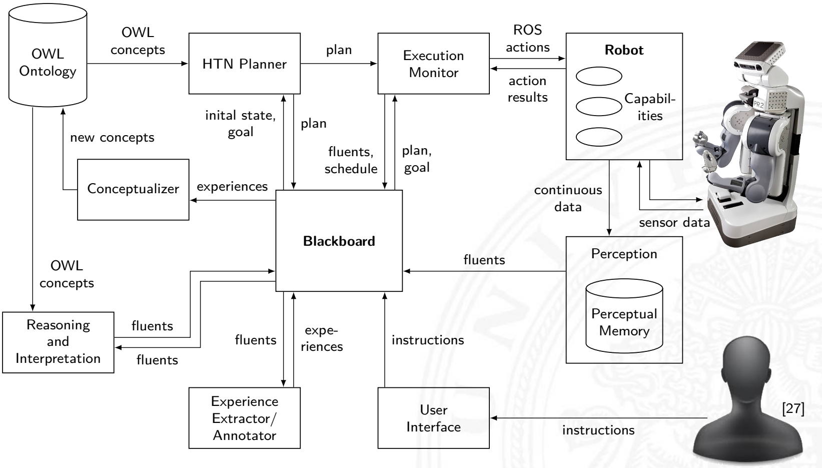
[25]

Atlantis Architecture



[26]

RACE
Robustness by Autonomous Competence Enhancement



Introduction to Robotics

Summary

Lasse Einig, Jianwei Zhang
[einig, zhang]@informatik.uni-hamburg.de

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

July 12, 2018

Outline

Outline (cont.)

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, Stanford)
- Mobile robots, running machines

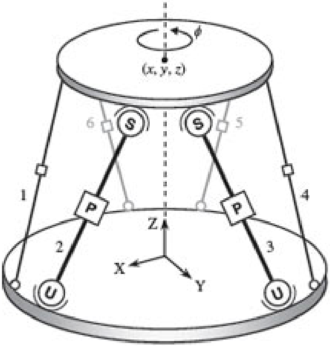
Things we did not talk about

- Closed chain, including Steward Mechanism [28, p. 279]
- Drive without motors

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
- Trajectory generation (e.g. linear Cartesian trajectory)
- Approximated representation of robot joints and objects
- Graph generation (V-Graph, T-Graph, ...)
- Search algorithms
- Further path planning algorithms
- Sensor fusion
- Vision
  - detection (static, dynamic)
  - reconstruction of position and orientation
- Action planning
- Sensor guided motion

Overall Summary

Introduction

+ Definition;
– Classification;
+ Basic components;
+ DOF

Coordinate Transformation

+ Manipulator-coordinates (Robot&Table);
+ Homogeneous transformations;
+ Rotation matrices, their inverse and their operations;
+ Transformation equations [2, 28, 3, 1]

Robot Description

+ DH-conventions and their applications (classic or modified);
○ Universal Robot Description Format (URDF)

Overall Summary (cont.)

Kinematics

+ Problems of forward and inverse kinematics;
– Algebraic and geometric solution of inverse kinematics;
+ Differential homogeneous transformations;
+ Jacobi-matrices;
+ Singularities [2, 28, 3, 1]

Trajectory Generation

+ Tasks and constraints;
– Polynomial solutions between two and four points;
◦ Factors of an optimal motion;
+ Linear motion in cartesian space, realization and problems;
+ Concepts of B-Spline interpolation;
– B-Spline basis functions [28, 3, 1, B-Spline Literature]

Overall Summary (cont.)

Programming

- − Task description, steps from the definition of frames to the implementation of programs;
- − Advantages and concepts of RCCL [2, RCCL-Guide];
- − Types of robot programming;
- − offline-programming [28, 3]

Control

- − Control systems of a PUMA robot;
- − Linear and model-based control;
- ○ PID controller;
- + Control concepts in Cartesian space [28, 3, 1]

Sensors

- ○ Classification;
- + Intrinsic sensors, principle and application in control;
- − extrinsic sensors [28, 3, 1]

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 [28, 3, 1]

Collision avoidance

+ Configuration space;
+ Concepts of transformation to configuration space;
∘ Object representation;
+ Potential field method;
+ Probabilistic approaches

Overall Summary (cont.)

Control architectures

- Subsumption;
- CMAC;
- Hierarchical

Additional references: [29, 30, 31, 32]

Outline

Outline (cont.)

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

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] 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.

[2] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control: the Computer Control of Robot Manipulators*.
Artificial Intelligence Series, MIT Press, 1981.

[3] J. Craig, *Introduction to Robotics: Pearson New International Edition: Mechanics and Control*.
Always learning, Pearson Education, Limited, 2013.

[4] J. F. Engelberger, *Robotics in service*.
MIT Press, 1989.

[5] 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.

[6] 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.

[7] 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.

[8] M. C. Ferch, *Lernen von Montagestrategien in einer verteilten Multiroboterumgebung*.
PhD thesis, Bielefeld University, 2001.

[9] J. H. Reif, "Complexity of the Mover's Problem and Generalizations - Extended Abstract," *Proceedings of the 20th Annual IEEE Conference on Foundations of Computer Science*, pp. 421–427, 1979.

[10] J. T. Schwartz and M. Sharir, "A Survey of Motion Planning and Related Geometric Algorithms," *Artificial Intelligence*, vol. 37, no. 1, pp. 157–169, 1988.

[11] J. Canny, *The Complexity of Robot Motion Planning*. MIT press, 1988.

[12] T. Lozano-Pérez, J. L. Jones, P. A. O'Donnell, and E. Mazer, *Handey: A Robot Task Planner*. Cambridge, MA, USA: MIT Press, 1992.

[13] O. Khatib, "The Potential Field Approach and Operational Space Formulation in Robot Control," in *Adaptive and Learning Systems*, pp. 367–377, Springer, 1986.

[14] J. Barraquand, L. Kavraki, R. Motwani, J.-C. Latombe, T.-Y. Li, and P. Raghavan, "A Random Sampling Scheme for Path Planning," in *Robotics Research* (G. Giralt and G. Hirzinger, eds.), pp. 249–264, Springer London, 1996.

Bibliography (cont.)

[15] R. Geraerts and M. H. Overmars, "A Comparative Study of Probabilistic Roadmap Planners," in *Algorithmic Foundations of Robotics V*, pp. 43–57, Springer, 2004.

[16] K. Nishiwaki, J. Kuffner, S. Kagami, M. Inaba, and H. Inoue, "The Experimental Humanoid Robot H7: A Research Platform for Autonomous Behaviour," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 365, no. 1850, pp. 79–107, 2007.

[17] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, pp. 14–23, Mar 1986.

[18] M. J. Mataric, "Interaction and intelligent behavior.," tech. rep., DTIC Document, 1994.

[19] M. P. Georgeff and A. L. Lansky, "Reactive reasoning and planning.," in *AAAI*, vol. 87, pp. 677–682, 1987.

[20] J. Zhang and A. Knoll, *Integrating Deliberative and Reactive Strategies via Fuzzy Modular Control*, pp. 367–385. Heidelberg: Physica-Verlag HD, 2001.

[21] 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.

[22] A. Meystel, "Nested hierarchical control," 1993.

[23] G. Saridis, "Machine-intelligent robots: A hierarchical control approach," in *Machine Intelligence and Knowledge Engineering for Robotic Applications* (A. Wong and A. Pugh, eds.), vol. 33 of *NATO ASI Series*, pp. 221–234, Springer Berlin Heidelberg, 1987.

[24] 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.

[25] R. C. Arkin and T. Balch, "Aura: principles and practice in review," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 175–189, 1997.

[26] E. Gat, "Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 70–74, 1991.

[27] L. Einig, *Hierarchical Plan Generation and Selection for Shortest Plans based on Experienced Execution Duration*.
Master thesis, Universität Hamburg, 2015.

[28] J. Craig, *Introduction to Robotics: Mechanics & Control. Solutions Manual*.
Addison-Wesley Pub. Co., 1986.

[29] H. Siegert and S. Bocionek, *Robotik: Programmierung intelligenter Roboter: Programmierung intelligenter Roboter*.
Springer-Lehrbuch, Springer Berlin Heidelberg, 2013.

Bibliography (cont.)

[30] R. Schilling, *Fundamentals of robotics: analysis and control*. Prentice Hall, 1990.

[31] T. Yoshikawa, *Foundations of Robotics: Analysis and Control*. Cambridge, MA, USA: MIT Press, 1990.

[32] M. Spong, *Robot Dynamics And Control*. Wiley India Pvt. Limited, 2008.