Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

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

May 10, 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

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.

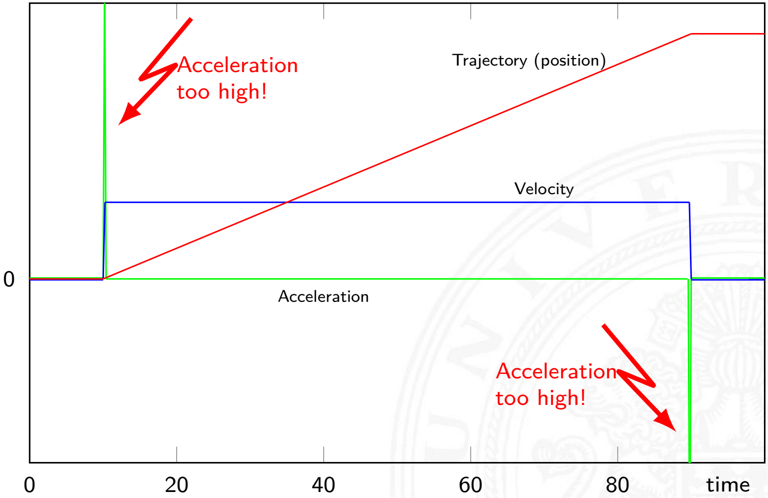
- possible only in simulation
- the moving distance for a manipulator at the next time step may be too large (velocity approaches ∞)

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

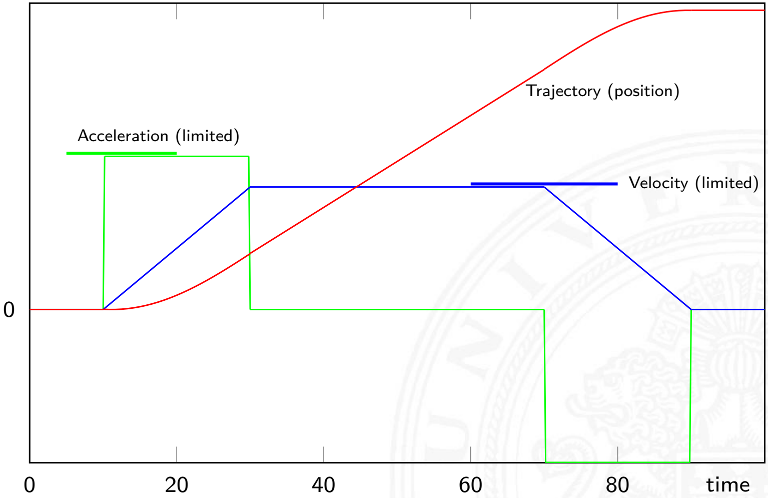
- dynamical trajectory planning
- advanced optimization methods $\rightarrow$ current topic of research

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

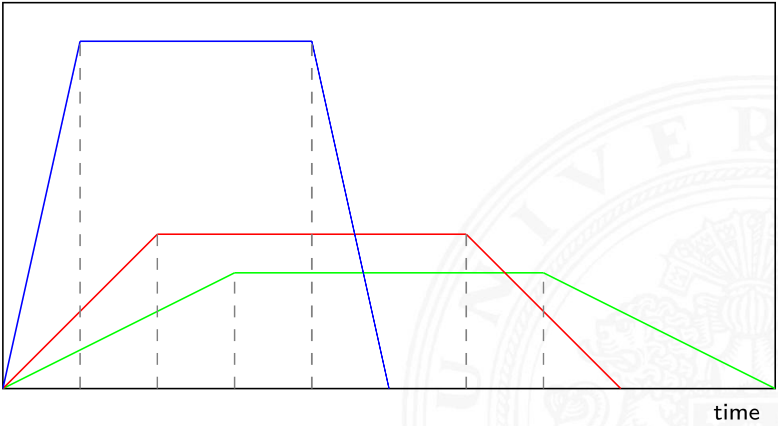
- different run time for joints (or cartesian dimensions)
- multiple velocity and acceleration contraints
- results in various `time switch points`
  - from acceleration to continuous velocity
  - from continuous velocity to deceleration
  - moving along a line in joint/cartesian space is impossible.

Solution

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

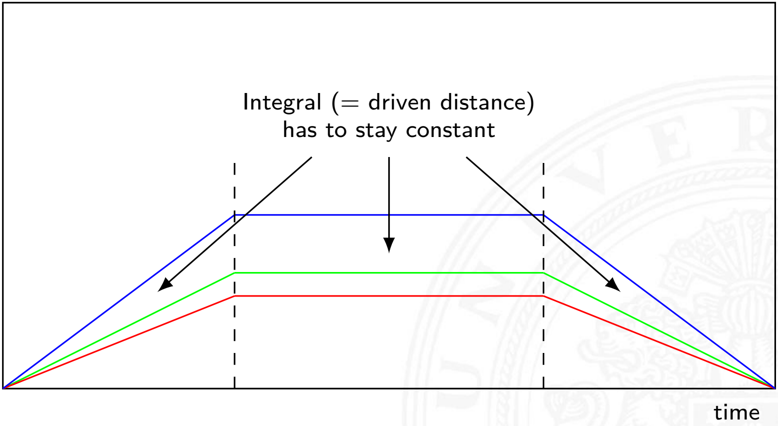
Trapezoidal interpolation – normalization

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

Polynomial interpolation (cont.)

Example 5^{th}-degree

$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$

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

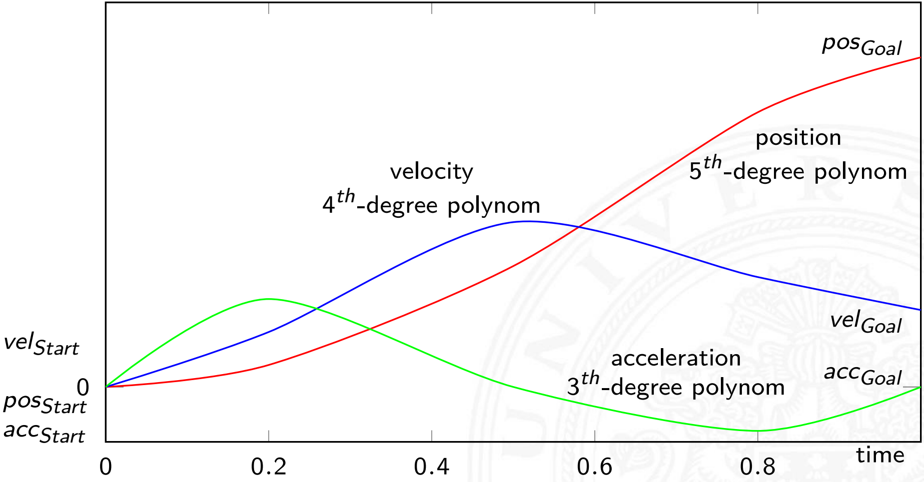
- $f(t_0) = pos_{Start}$, $f(t_d) = pos_{Goal}$
- $f'(t_0) = vel_{Start}$, $f'(t_d) = vel_{Goal}$
- $f''(t_0) = acc_{Start}$, $f''(t_d) = acc_{Goal}$

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

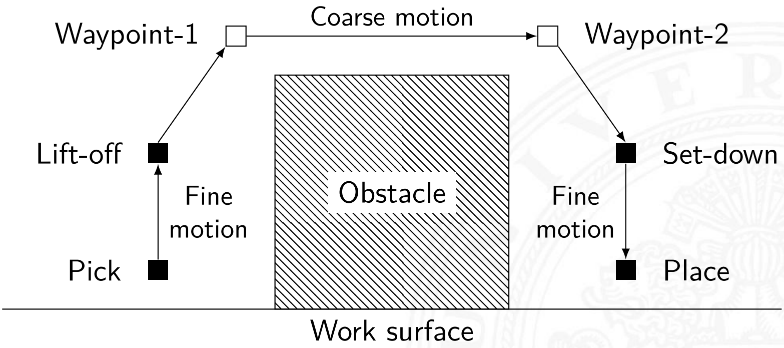
Proper position interpolation from start (A) to goal (B)

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

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.)

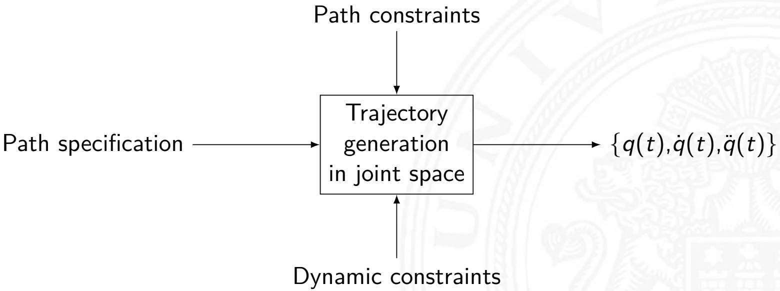
Cartesian space:

- 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)

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

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

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

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

Cubic polynomials with waypoints and velocities (cont.)

- 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."*

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

- trajectory of circle

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

- arc length L of circle (circumference)

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

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

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

$$= \int_0^{2\pi} r dt \tag{81}$$

$$L = 2\pi r \tag{82}$$

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

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

Curvature of a circle

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

Factors for time optimal motion – Bending Energy

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

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

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

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

Factors for time optimal motion – Motion Time

If a motion consists of n successive segments

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

then

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

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

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

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 $i \in \{1 \ldots n\}$ of trajectory part $j \in \{1 \ldots m\}$ 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 $(\mathbf{w}_M, \mathbf{w}_1)$.

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.

Bibliography (cont.)

[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.

[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.

Bibliography (cont.)

[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.

Bibliography (cont.)

[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.