Introduction to Robotics

Lecture 5

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

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

May 22, 2020

Joint velocities $\Leftrightarrow$ End-effector velocities

$\Downarrow$

Jacobian

- Jacobian

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

- Angular/Linear velocity Jacobian

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

- Computation of the final Jacobian

Singularities

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

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

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

Outline

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

Trajectories in multidimensional space

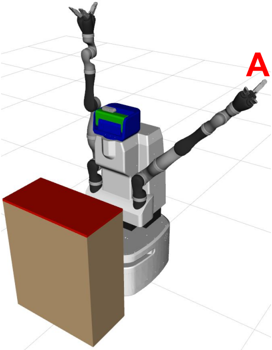
- Changes in position, velocity and acceleration of all joints are analyzed over a period of time
- Trajectory with n DOF is a parameterized function $q(t)$ with values in its motion region.
- Trajectory $q(t)$ of a robot with n DOF is then a vector of n parameterized functions $q_i(t), i \in \{1 \dots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

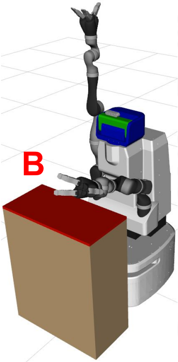
$$q(t) = [q_1(t), q_2(t), \dots, q_n(t)]^T$$

Problem

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

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





Trajectory generation

Problem

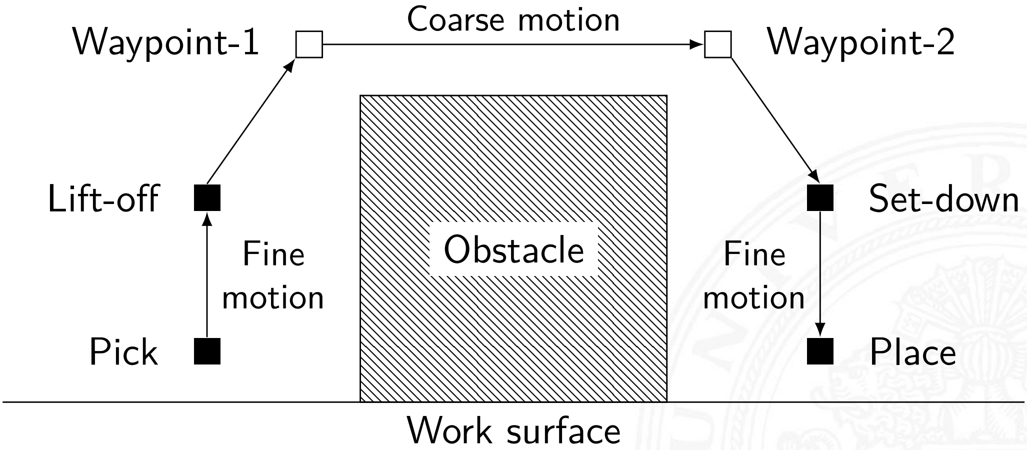
The robot is at point A and wants move to point B.

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

Solution

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

Trajectory planning



Trajectory planning (cont.)

Pick $pos_{Start} = object$, $vel_{Start} = 0$, $acc_{Start} = 0$

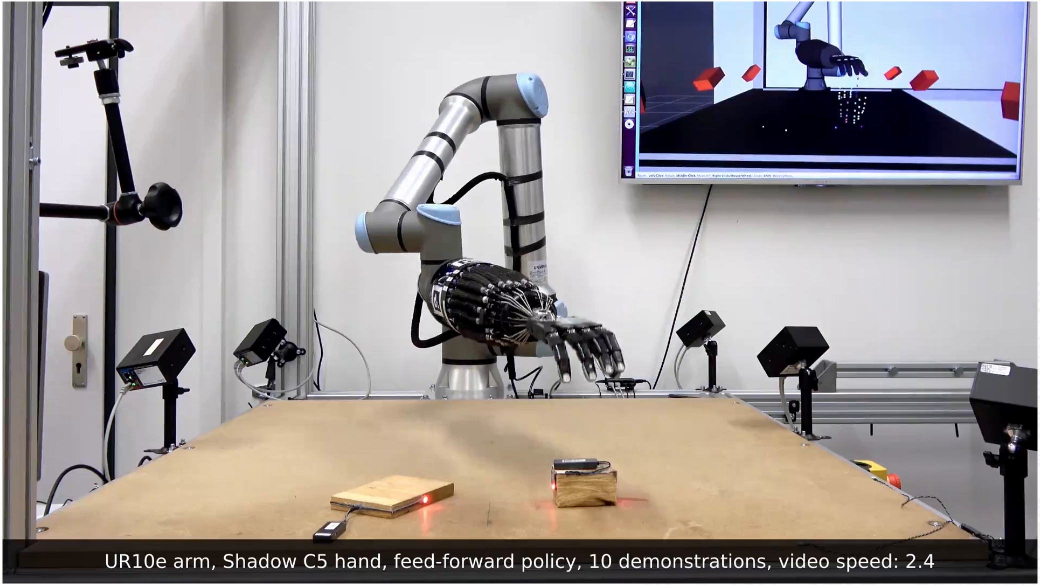
Lift-off limited velocity and acceleration

Motion continuous via waypoints, full velocity and acceleration

Set-down similar to Lift-off

Place similar to Pick

Trajectory planning (cont.)



Task level planning

Generation of trajectories

Task

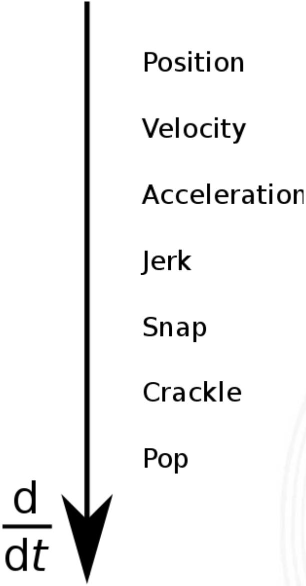
- find a smooth trajectory for moving the robot from start to goal pose
- use continuous functions of time

Continuity of Trajectories

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

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

Time-derivatives of position



Generation of trajectories

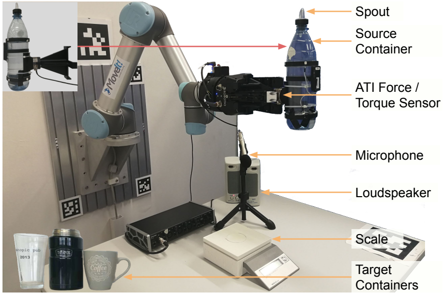
Task

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

Representation solution:

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

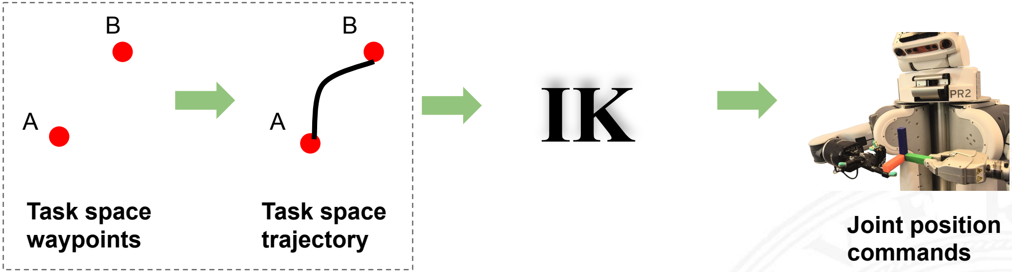
Generation of trajectories (cont.)



Pouring setup

Pushing setup

Trajectories in Cartesian space



Advantages:

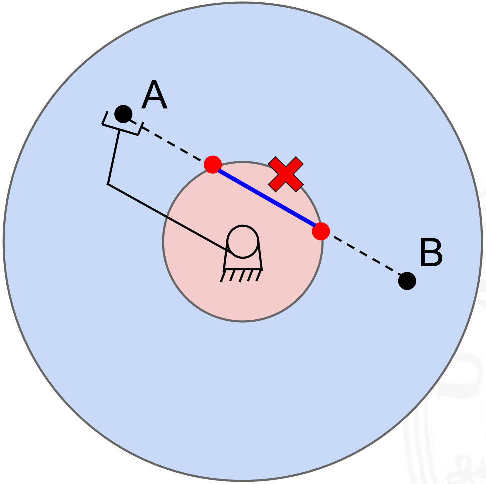
- near to the task specification
- advantageous for collision avoidance
- can specify the spatial shape of the path

Disadvantages:

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

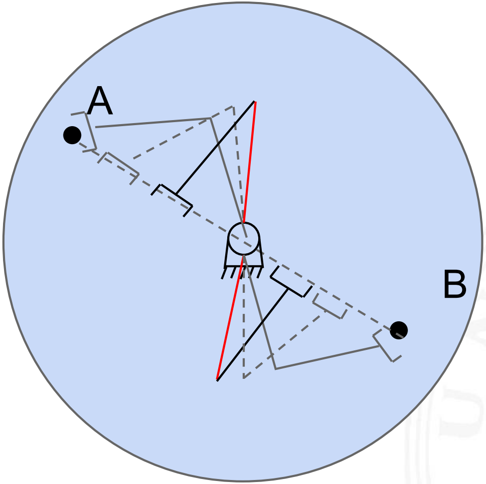
Difficulties of trajectories in Cartesian space

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



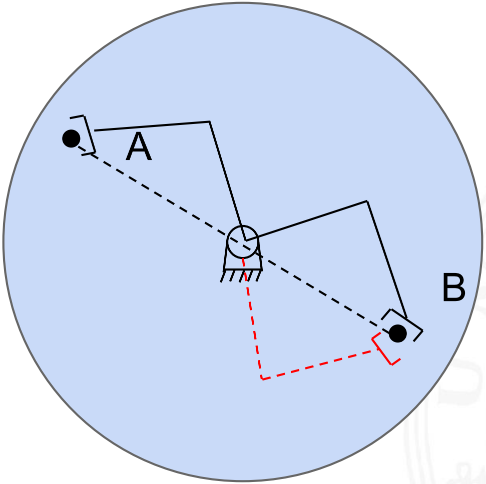
Difficulties of trajectories in Cartesian space (cont.)

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

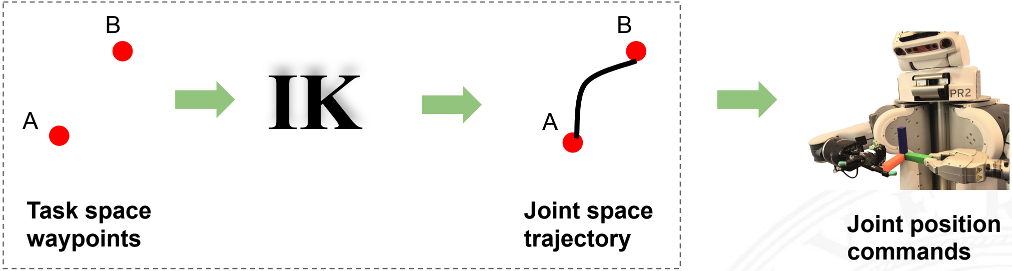


Difficulties of trajectories in Cartesian space (cont.)

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



Trajectories in joint space



Joint space:

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

Primitive solution

Naive approach

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

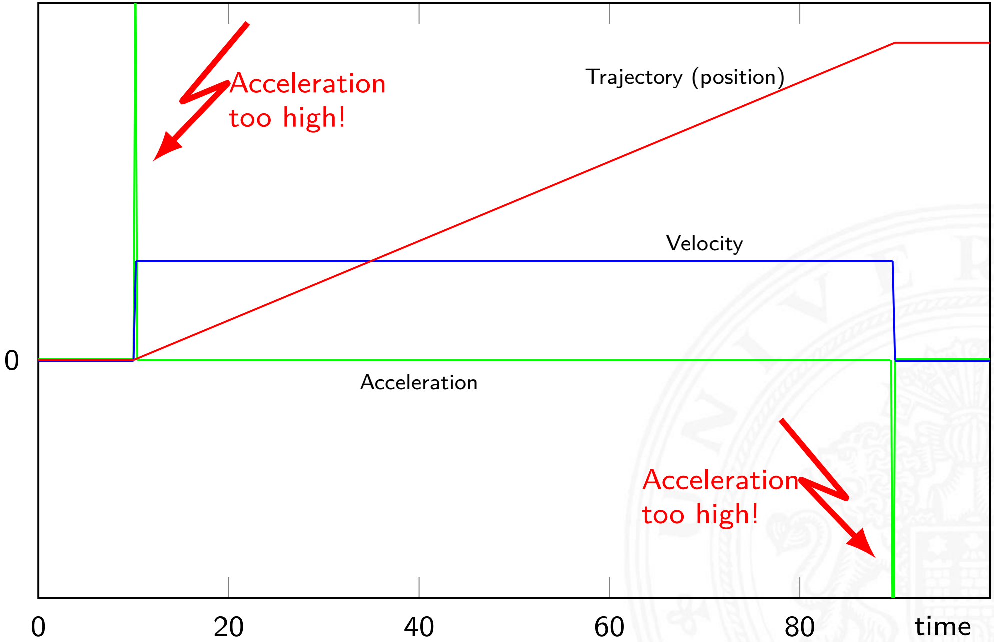
- possible only in simulation
- the moving distance for a manipulator at the next time step may be too large (velocity approaches ∞)

Linear interpolation

Next best approach

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

Linear interpolation – visualization



Linear interpolation – constraints

Problem

The physical constraints are violated

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

Implicitly these contraints are valid for motion in cartesian space.

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

Solution

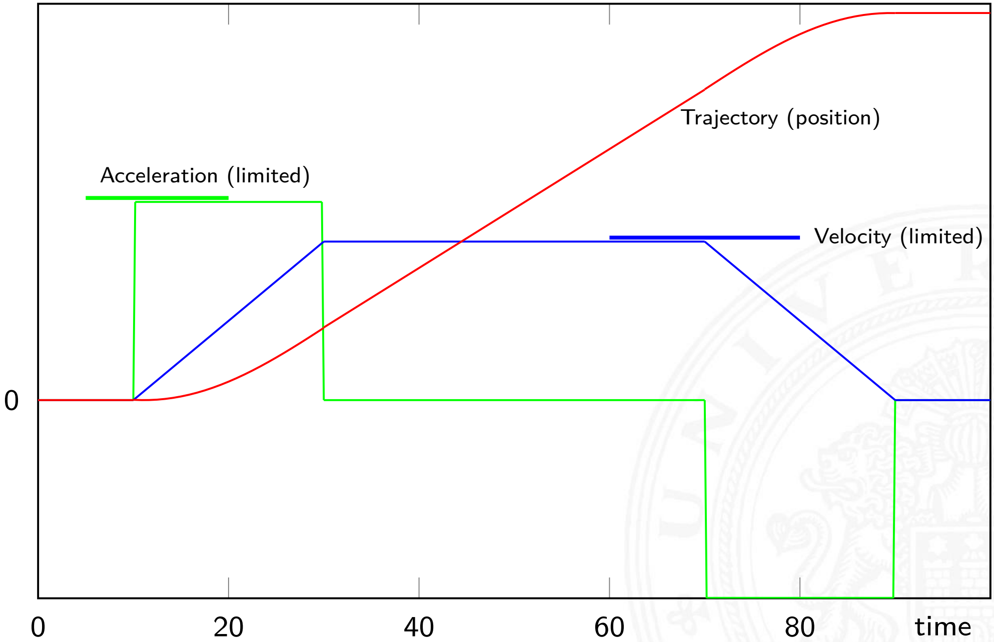
- dynamical trajectory generation
- advanced optimization methods $\rightarrow$ current topic of research

Linear interpolation – improvement

Next best approach

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

Trapezoidal interpolation – visualization



Trapezoidal interpolation – summary

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

Trapezoidal interpolation – summary (cont.)

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

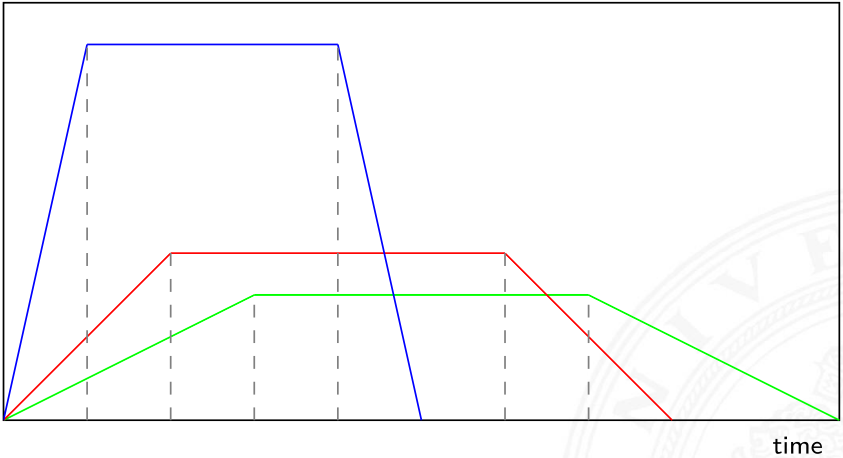
Trapezoidal interpolation – constraints

Problem

Multidimensional trapezoidal interpolations

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

Trapezoidal interpolation – constraints

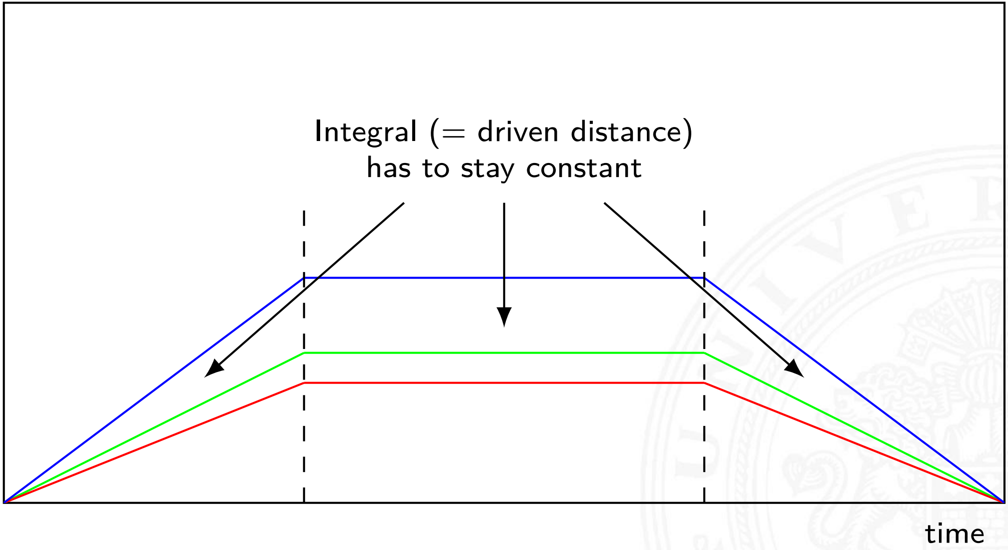


Solution

- Normalization to the slowest joint

Trapezoidal interpolation – normalization

Normalize to the slowest joint



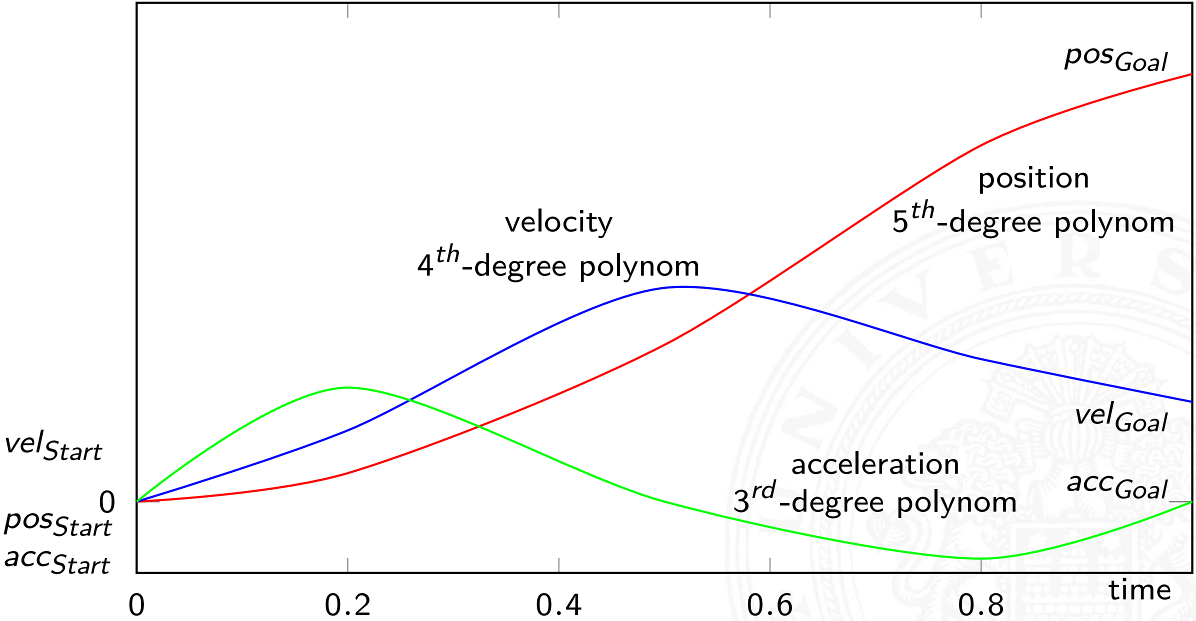
Polynomial interpolation

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

Polynomial interpolation (cont.)

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

Polynomial interpolation (cont.)



Cubic polynomials between two configurations

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

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

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

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

- if the start and end velocity is 0 then

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

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

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

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

Cubic polynomials between two configurations (cont.)

- The solution

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

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)

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

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

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

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

- The solution

$$
\begin{aligned}
\text{eq. } (40) \quad & a_0 = \theta_0 \\
\text{eq. } (42) \quad & a_1 = \dot{\theta}_0 \\
& a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f \\
& a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0)
\end{aligned}
$$

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

Quintic polynomials

Example 5^{th}-degree

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

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

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

Remarks on generation of trajectories

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

Factors for time optimal motion – Arc Length

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

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

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

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

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

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

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

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

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

Factors for time optimal motion – Curvature

Curvature

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

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

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

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

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

Factors for time optimal motion – Bending Energy

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

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

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

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

Factors for time optimal motion – Motion Time

If a motion consists of n successive segments

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

then

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

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

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

Minimum jerk trajectory

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

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

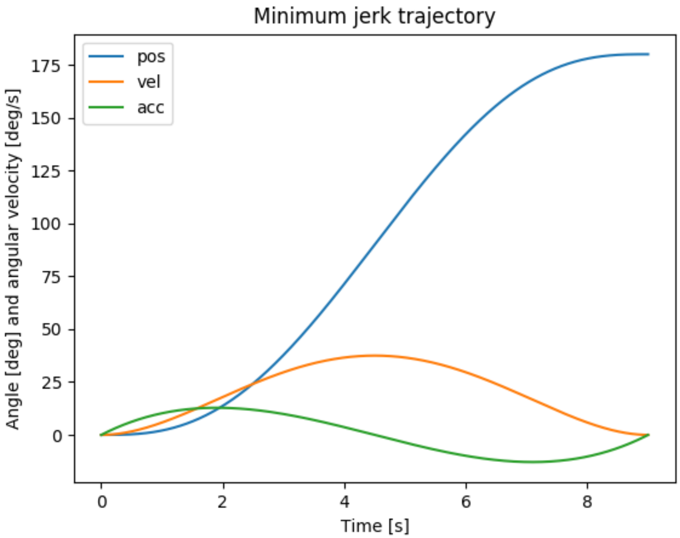
- The minimum-jerk solution can be written as:

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

- Predicts bell shaped velocity profiles

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

Minimum jerk trajectory (cont.)



Dynamical constraints for all joints

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

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

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

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

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

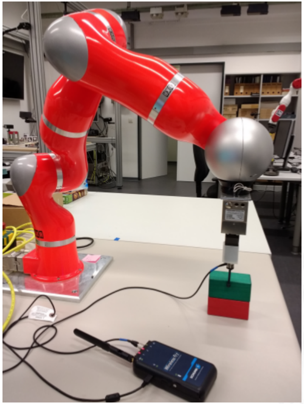
- m^i is the torque (moment of force) for the joint i and can be calculated from the dynamical equation (motion equation).
- $\dot{q}_{max}^i$, $\ddot{q}_{max}^i$ and m_{max}^i represent the important parameters of the dynamical capacity of the robot.

Reflexxes Motion Libraries

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

Examples of using Reflexxes in TAMS

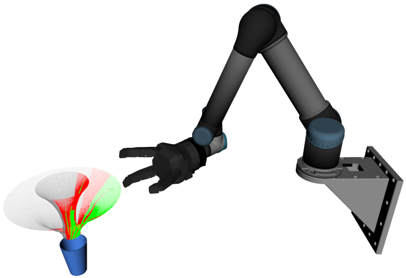
- Real-time object shape detection using ROS, the KUKA LWR4+ and a force/torque Sensor
  - to specify the target position and target velocity at the target position



24

Examples of using Reflexxes in TAMS (cont.)

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



25

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

Bibliography

[1] G.-Z. Yang, R. J. Full, N. Jacobstein, P. Fischer, J. Bellingham, H. Choset, H. Christensen, P. Dario, B. J. Nelson, and R. Taylor, "Ten robotics technologies of the year," 2019.

[2] J. K. Yim, E. K. Wang, and R. S. Fearing, "Drift-free roll and pitch estimation for high-acceleration hopping," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8986–8992, IEEE, 2019.

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

[4] K. Fu, R. González, and C. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*.
McGraw-Hill series in CAD/CAM robotics and computer vision, McGraw-Hill, 1987.

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

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

Bibliography (cont.)

[7] T. Flash and N. Hogan, “The coordination of arm movements: an experimentally confirmed mathematical model,” *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[8] T. Kröger and F. M. Wahl, “Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events,” *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2009.

[9] W. Böhm, G. Farin, and J. Kahmann, “A Survey of Curve and Surface Methods in CAGD,” *Comput. Aided Geom. Des.*, vol. 1, pp. 1–60, July 1984.

[10] J. Zhang and A. Knoll, “Constructing Fuzzy Controllers with B-spline Models - Principles and Applications,” *International Journal of Intelligent Systems*, vol. 13, no. 2-3, pp. 257–285, 1998.

[11] M. Eck and H. Hoppe, “Automatic Reconstruction of B-spline Surfaces of Arbitrary Topological Type,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), pp. 325–334, ACM, 1996.

Bibliography (cont.)

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

[13] 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.

[14] 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.

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

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

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

[18] 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.

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

[20] 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.

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

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

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

Bibliography (cont.)

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

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

Bibliography (cont.)

[30] 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.

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

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

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

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

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

Bibliography (cont.)

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