Robotics: Path and trajectory generation

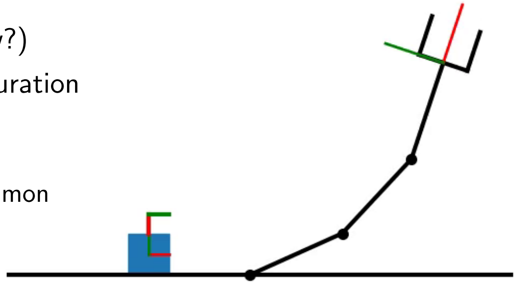
Vladimír Petrík

vladimir.petrik@cvut.cz

30.10.2023

Motivation: pick a cube

- Detect where the cube is in $SE(2)$, $SE(3)$
- Define handle(s) w.r.t. cube
- Compute gripper pose
- Solve IK (select one of the solutions, how?)
- Send robot to selected joint-space configuration
- What motion will robot follow?
  - depends on the robot
  - linear interpolation in joint space is common
  - what is motion?

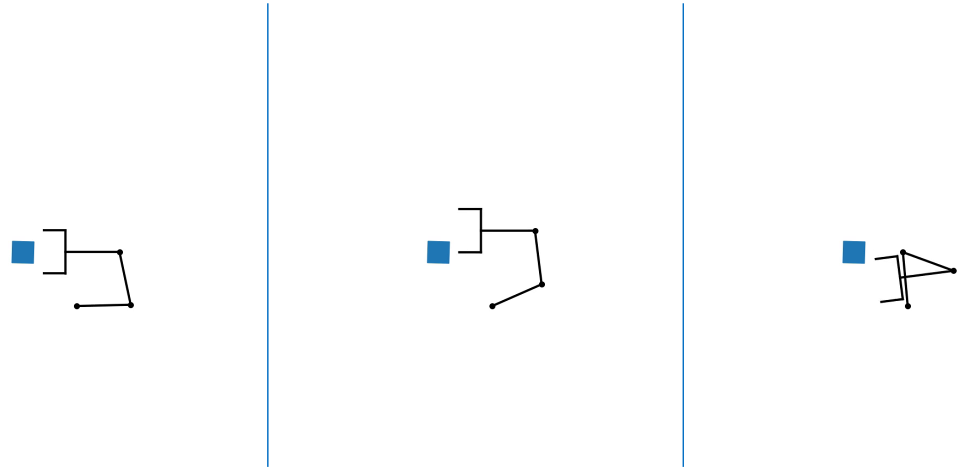


Motion

- Path
  - Geometrical description (sequence of configurations)
  - No timestamps, dynamics, or control restrictions
  - $\boldsymbol{q}(s) \in \mathcal{C}_{\text{free}}, s \in [0, 1]$
  - Main assumption is that trajectory can be computed by postprocessing
- Trajectory
  - Robot configuration in time
  - $\boldsymbol{q}(t) \in \mathcal{C}_{\text{free}},\ t \in [0, T]$

Grasping path

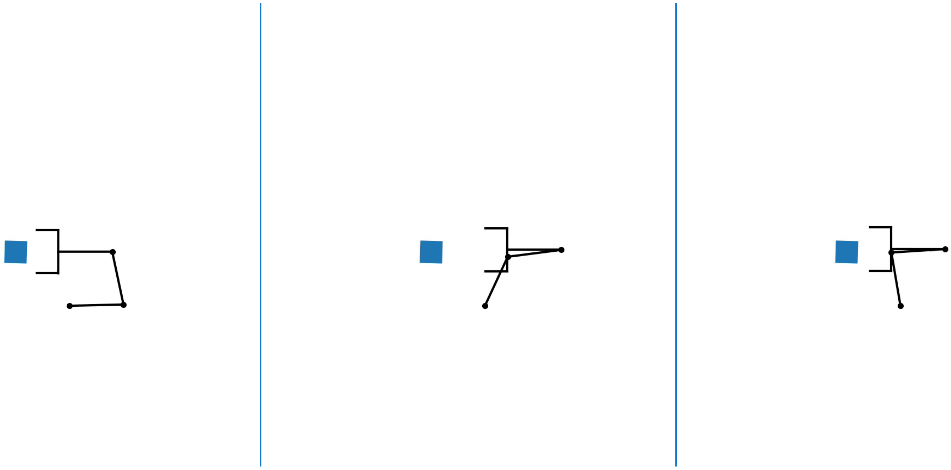
- Let us focus on **path** first
- Is grasping path safe? Depends on the start configuration.



Pre-grasp pose

- We can define pre-grasp pose
  - *e.g.* 5 cm away from the object, w.r.t. handle
  - how to define 5 cm away? By design of handle.
  - fix handle orientation to have x-axis pointing towards the object
  - gripper orientation to have x-axis pointing out of gripper
  - grasp pose T_{RH}
  - if gripper T_{RG} equals T_{RH}, object is grasped
  - pre-grasp pose $T_{RP} = T_{RH} T_x(-\delta_{\text{pre\_grasp}})$
- Is path from pre-grasp to grasp safe if $\delta_{\text{pre\_grasp}}$ is small?
- Is path from pre-grasp to grasp safe if $\delta_{\text{pre\_grasp}}$ is large?

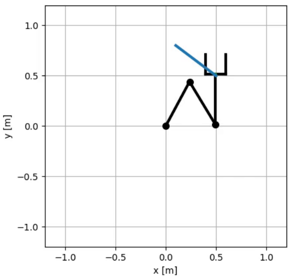
Pre-grasp pose

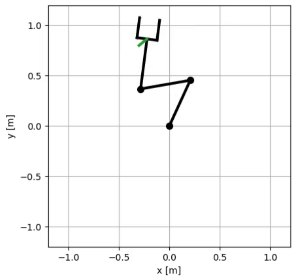


Interpolation in joint space

- Also called straight-line path, point-to-point path
- Start $\boldsymbol{q}_{\text{start}}$
- Goal $\boldsymbol{q}_{\text{goal}}$
- $\boldsymbol{q}(s) = \boldsymbol{q}_{\text{start}} + s(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}}), \quad s \in [0, 1]$
- Easy to compute, well defined
- What is the motion of the gripper?
  - likely not straight-line (for revolute joints)
  - combinations of circular paths (for revolute joints)

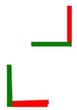
Interpolation in joint space

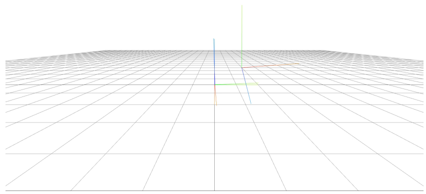




Interpolation in $SE(2)$ and $SE(3)$

- Straight-line path in task space
  - position $\boldsymbol{t}(s) = \boldsymbol{t}_{\text{start}} + s(\boldsymbol{t}_{\text{goal}} - \boldsymbol{t}_{\text{start}}), \quad s \in [0, 1]$
  - rotation $R(s) = R_{\text{start}} \exp\left(s \log(R_{\text{start}}^{-1} R_{\text{goal}})\right), \quad s \in [0, 1]$



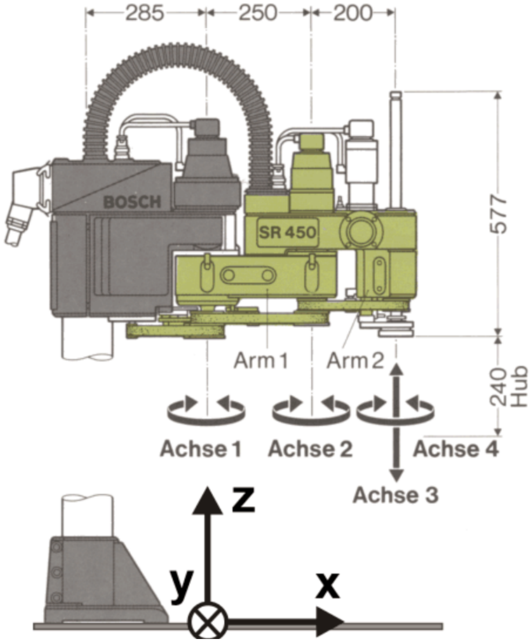


Joint-space path from task-space path

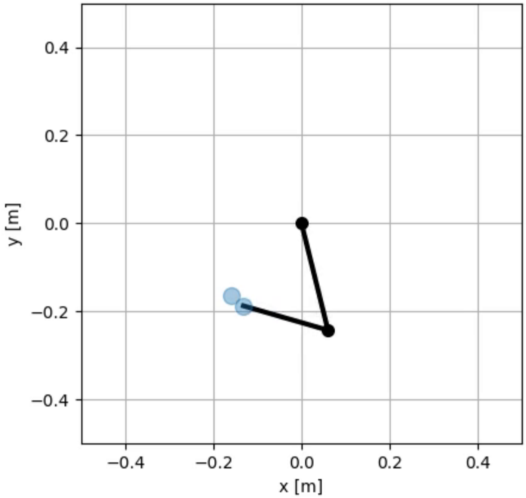
- Compute $\boldsymbol{q}(s)$ from $T_{RG}(s)$
- Solve IK for each s and pick the first solution of IK?
  - we did not define what is *first* solution of IK
  - let us use the closest solution of IK
  - can it happen that closest solution is not *close enough*? **yes**, let us see an example

SCARA robot

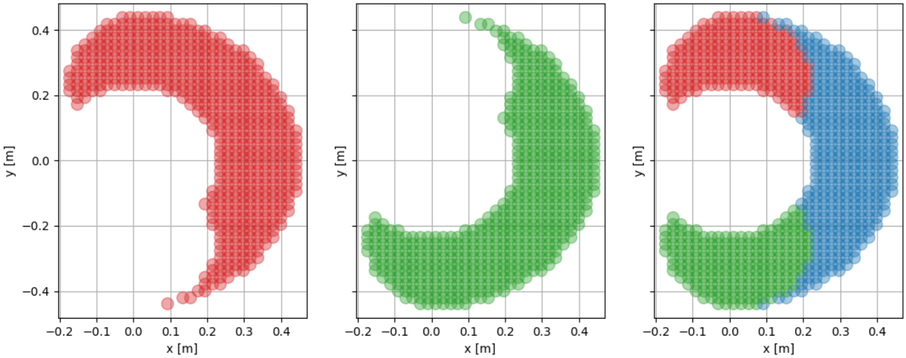
-
- Analyze kinematics of SCARA
- Structure RRPR
- Self-collisions avoided by joint limits
  - $\pm 85°$
  - $\pm 120°$
  - $(-330\text{ mm}, 5\text{ mm})$
  - $(-20°, 1080°)$
- Compute FK and IK in xy-plane



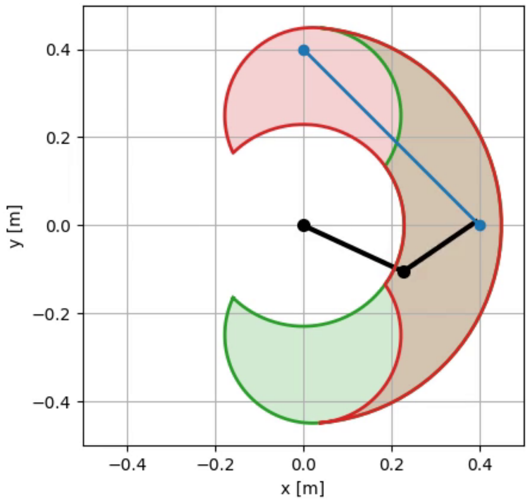
SCARA robot workspace



SCARA robot IK



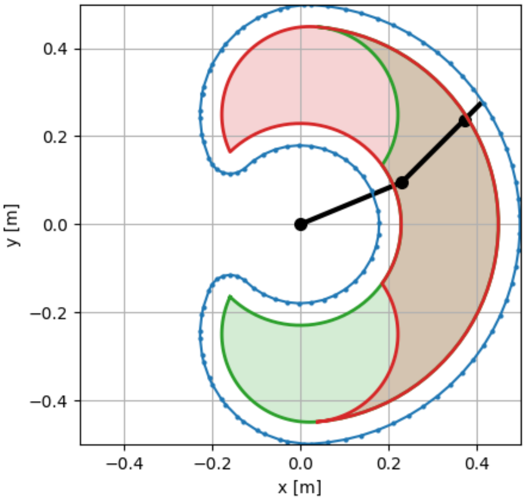
Task-space interpolation



Task space interpolation

- Not all solutions of IK are available everywhere
- We need to resolve jumps in configuration space
- To change the configuration we need to pass via singularity
- The task-space interpolation can be used for pre-grasp to grasp path

SCARA effect of the last link



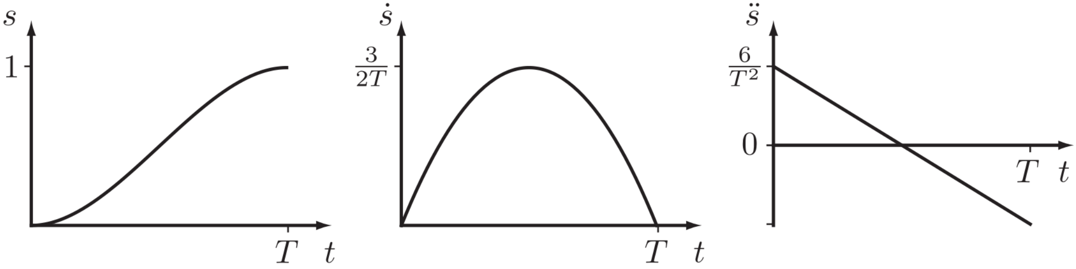
Trajectory from path

- Time scaling $s(t)$, $t \in [0, T]$, $s : [0, T] \rightarrow [0, 1]$
- A path and time scaling defines trajectory $\boldsymbol{q}(s(t))$
- Derivations:
  - velocity: $\dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}s}\dot{s}$
  - acceleration: $\ddot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}s}\ddot{s} + \frac{\mathrm{d}^2\boldsymbol{q}}{\mathrm{d}s^2}\dot{s}^2$

Straight-line path time scaling

- Path
  - position: $\boldsymbol{q}(s) = \boldsymbol{q}_{\text{start}} + s(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}}), \quad s \in [0, 1]$
  - velocity: $\dot{\boldsymbol{q}} = \dot{s}(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}})$
  - acceleration: $\ddot{\boldsymbol{q}} = \ddot{s}(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}})$
- 3rd order polynomial time scaling
  - $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
  - $\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2$
  - constraints: $s(0) = \dot{s}(0) = 0$, $s(T) = 1$, $\dot{s}(T) = 0$
  - solution that satisfies constraints: $a_0 = 0, \quad a_1 = 0, \quad a_2 = 3/T^2, \quad a_3 = -2/T^3$
- Trajectory
  - $\boldsymbol{q}(t) = \boldsymbol{q}_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3}\right)(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}})$
  - $\dot{\boldsymbol{q}} = \left(\frac{6t}{T^2} - \frac{2t^2}{T^3}\right)(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}})$
  - $\ddot{\boldsymbol{q}} = \left(\frac{6}{T^2} - \frac{12t}{T^3}\right)(\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{start}})$

3rd order polynomial time scaling

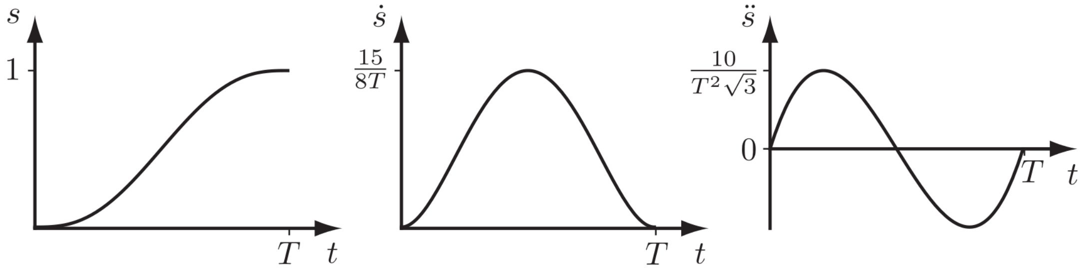


Straight-line path time scaling

- Maximum joint velocities:
  - $t = T/2$
  - $\dot{\boldsymbol{q}}_{\text{max}} = \frac{3}{2T}(\boldsymbol{q}_{\text{goal}}, \boldsymbol{q}_{\text{start}})$
- Maximum joint acceleration:
  - $t = 0$ and $t = T$
  - $\ddot{\boldsymbol{q}}_{\text{max}} = \left\| \frac{6}{T^2}(\boldsymbol{q}_{\text{goal}}, \boldsymbol{q}_{\text{start}}) \right\|$
  - $\ddot{\boldsymbol{q}}_{\text{min}} = -\left\| \frac{6}{T^2}(\boldsymbol{q}_{\text{goal}}, \boldsymbol{q}_{\text{start}}) \right\|$
- How to use this information?
  - check if requested motion T is feasible given the velocity/acceleration limits
  - find minimum T such that velocity and acceleration constraints are satisfied

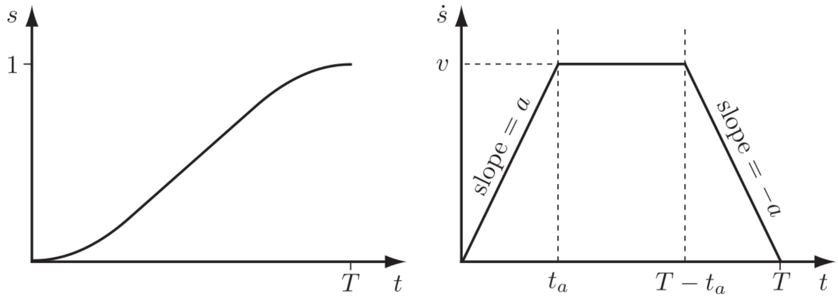
5th order polynomial

- 3rd order polynomial does not enforce zero acceleration at the beginning and end
  - infinite jerk (derivative of acceleration)
  - can cause vibrations
- We can use 5th order polynomial



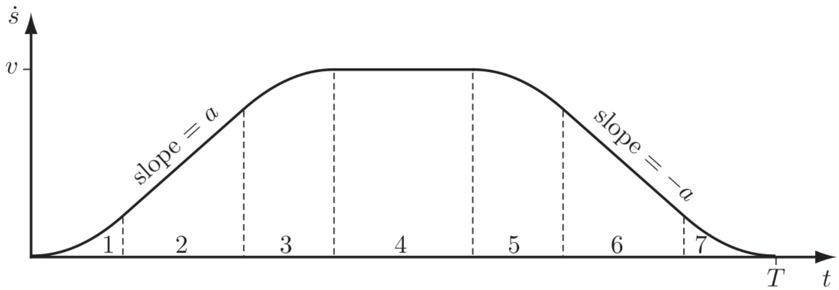
Trapezoidal time scaling

- Constant acceleration phase
- Constant velocity phase
- Constant deceleration phase
- Not smooth but it is the fastest straight-line motion possible



S-Curve time scaling

- Trapezoidal motions cause discontinuous jumps in acceleration
- S-curve smooths it to avoid vibrations
  - constant jerk, constant acceleration, constant jerk, constant velocity, constant jerk, constant deceleration, constant jerk



Summary

- Path/Trajectory
- Grasping path generation
- Interpolation in joint space and task space
- Time scaling parameterization

Laboratory

- Laboratories this week are **mandatory**
  - safety
  - robot control tutorial
- Room: JP-B-415
  - TA will pick you up in front of the room