Optimization, Robotics & Biomechanics (ORB)
Institute of Computer Engineering (ziti)
Heidelberg University

Prof. Dr. Katja Mombaur,
Benjamin Reh,
Silvan Lindner

Robotics 1 (WS 2018/2019)

Exercise Sheet 10

Presentation during exercises in calendar week 4

Exercise 10.1 – Time Scaling

Consider a straight line path $\theta(s) = \theta_{\text{start}} + s\,(\theta_{\text{end}} - \theta_{\text{start}}),\ s \in [0,1]$ from $\theta_{\text{start}} = (0,0)$ to $\theta_{\text{end}} = (\pi, \pi/3)$. The motion starts and ends at rest. The feasible joint velocities are $|\dot{\theta}_1|, |\dot{\theta}_2| \leq 2$ rad/s and the feasible joint accelerations are $|\ddot{\theta}_1|, |\ddot{\theta}_2| \leq 0.5$ rad/s^2.

- Find the fastest motion time T using a cubic time scaling that satisfies the joint velocity and acceleration limits.
  Hint: Cubic time scaling is achieved with $s(t) = a_3 t^3 + a_2 t^2 + a_1 t^1 + a_0$.
- Repeat this process for trapezoidal time scaling. Under which condition is the maximum joint-speed never reached?

Exercise 10.2 – Spline Interpolation

Splines are often used for trajectory planning, as they result in continuously differentiable paths. Given n control-points $p_i \in \mathbb{R}^N$, $i = 0, \dots, n-1$, and time-points $t_0 < t_1 < \dots < t_n$ the cubic spline function $S(t)$ continuously interpolating all n points is defined piecewise by

$$S(t) = \begin{cases} f_0(t), & t_0 \le t < t_1 \\ \qquad \cdots & \\ f_{n-1}(t), & t_{n-1} \le t < t_n \end{cases}, \tag{1}$$

where a cubic function is defined on each interval as

$$f_i(t) = a_i + b_i t + c_i t^2 + d_i t^3.$$

In order for the spline to be continuously differentiable, the following continuity conditions must also hold:

$$f_i(t_i) = p_i, \text{ and } f_i(t_{i+1}) = p_{i+1}, \quad i = 0, \dots, n-1 \tag{2}$$
$$\dot{f}_i(t_{i+1}) = \dot{f}_{i+1}(t_{i+1}), \quad i = 0, \dots, n-2 \tag{3}$$
$$\ddot{f}_i(t_{i+1}) = \ddot{f}_{i+1}(t_{i+1}), \quad i = 0, \dots, n-2 \tag{4}$$

Those equations define $4n-2$ conditions for $4n$ variables, but we need $4n$ conditions to uniquely determine all spline coefficients a_i, b_i, c_i and d_i. Thus, we introduce two additional constraints

$$\ddot{f}_0(t_0) = \ddot{f}_{n-1}(t_n) = 0 \tag{5}$$

called *natural boundary conditions.*

- Show that the third order polynomial

$$f_i(t) = M_{i+1}\frac{(t-t_i)^3}{6h_i} + M_i\frac{(t_{i+1}-t)^3}{6h_i} + \left[\frac{p_{i+1}}{h_i} - M_{i+1}\frac{h_i}{6}\right](t-t_i) + \left[\frac{p_i}{h_i} - M_i\frac{h_i}{6}\right](t_{i+1}-t) \tag{6}$$

  with $M_i = \ddot{S}(t_i)$ and $h_i = t_{i+1} - t_i$ satisfies equations (2) and (4).

- Formulate a linear system of equations that needs to be solved to determine all M_i in Matrix notation, i.e. write conditions (3) and (5) as

$$A \begin{bmatrix} M_0 \\ \vdots \\ M_n \end{bmatrix} = b,$$

  where $A \in \mathbb{R}^{(n+1)\cdot N \times (n+1)\cdot N}$ and $b \in \mathbb{R}^{(n+1)\cdot N}$.

Exercise 10.3 – Linear endeffector motion

This assignment is a continuation of the PTP motion assignment on sheet 7. However, if you did not find a (correct) solution to the PTP motion, the code provided with this assignment contains everything needed.

The linear motion will be used to make a simple drawing using the Kuka robot. The goal of this assignment is to produce an output file that can be played back on the real Kuka robot in the lab.

a) **LIN motion**
The PTP motion is a linear interpolation of the joint angles $\vec{q}$ using a trapezoid ramp function. For details please have a look at sheet no. 7. In this assignemt not the joint angles but the endeffector position is to be interpolated in such a way that the endeffector's

i path is a straight line from point P_1 to point P_2,

ii velocity follows a ramp function,

iii orientation is constant throughout the entire motion.

In general, the last point is not absolutely necessary as also the orientation can be interpolated as seen in sheet no. 5. For simplicity however we assume it to be constant for this assignment.

For each interpolated endeffector point in time, the inverse kinematic function is called to detemine the joint angles $\vec{q}$ for this time step using the last solution as a initial value q_{start}.

Download and run the code provided for this exercise. Visualize the result with meshup. If you enable the *curves* feature, you can see the result of the drawing.

1. The example provided shows a linear interpolation of the end effector with constant velocity. Implement the velocity ramps: The TCP shall accelerate up to a maximum velocity (if reached) and then decelerate again before reaching the end point.
2. Add points to the `endPoints` Vector in your code. These are the points the robot will be plotting in local coordinates on the paper. The local x and y axes are on the paper while positive z values can be used to lift the pen off the paper. Let the robot draw "Das Haus vom Nikolaus" ("the House of Santa Claus").

   Add a point at the beginning and at the end to lift the pen from the paper.

b) **Transformation from Local Frame**
It is common to define the task of the robot such as this drawing in a local coordinate frame. This has the advantage that if the object for the task (the paper) will be at a different position, the definition of the task itself stays the same while only the tranformation between the local and the global frame has to be adjusted.

A common way to obtain this transformation in the real world is by defining three points on a plane by teach-in: The robot's endeffector is moved to a point on the plane. Its position is then computed using the robot's kinematics. This is repeated for at least two more points.

A transformation class is provided in the code. It scales an input vector down by a factor, rotates it with a rotation matrix M and translates it to a different origin. Some example values are predefined in the constructor. The parameters can be defined using the `calibrate` function that takes 3 points as arguments. The calibration takes place as follows:

i Point A is the origin of the local coodinate system

ii Point B is chosen such that $\overline{AB}$ is the x axis of the local frame. $|\overline{AB}|$ has the length of one unit.
In other words: the local vector $(1,0,0)^T$ maps directly on point B.

iii Point C is not along $\overline{AB}$ and defines the plane.

Implement the transformation parameters for scaling, translation and the rotation matrix M using these three points.
Hints:

- M can be seen a the three new axes $M = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$
- Use the fact that $x \times y = z$ (and permutations) and $a \times b = -b \times a$.
- Make sure the resulting coodinate frame is still right-handed.
- Be aware: Although three points define a plane they do not define in which direction the normal of the plane is pointed to, i.e. which side of the plane is the front side and which the back side.

For convinience, some example calibrations are provided in the code and can be tested.

c) **Last Steps**
The robot shall start and end the motion from its parking position at $\vec{q} = (0, -\frac{\pi}{2}, \frac{\pi}{2}, 0, 0, 0)$. Place a PTP motion at the beginning and one at the end.