Robotics
Exercise 2

Marc Toussaint
Lecturer: Peter Englert
TAs: Matt Bernstein, Danny Driess, Hung Ngo
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

October 25, 2016

1 Task spaces and Jacobians

In the lecture we introduced the basic kinematic maps $\phi^{\mathsf{pos}}_{i,v}(q)$ and $\phi^{\mathsf{vec}}_{i,v}(q)$, and their Jacobians, $J^{\mathsf{pos}}_{i,v}(q)$ and $J^{\mathsf{vec}}_{i,v}(q)$, where i may refer to any part of the robot and v is any point or direction on this part. In the following you may assume that we have routines to compute these for any q. The problem is to express other kinematic maps and their Jacobians in terms of these knowns. In the following you're asked to define more involved kinematics maps (a.k.a. task maps) $\phi(q)$ to solve certain motion problems. Please formulate all these maps such that the overall optimization problem is

$$f(q) = \|q - q_0\|^2_W + \|\phi(q)\|^2$$

that is, the motion aims to minimize $\phi(q)$ to zero (absorb the y^* in the definition of ϕ.)

a) Assume you would like to control the pointing direction of the robot's head (e.g., its eyes) to point to a desired external world point x^W. What task map can you define to achieve this? What is the Jacobian?

b) You would like the two hands or the robot to become parallel (e.g. for clapping). What task map can you define to achieve this? What is the Jacobian?

c) You would like to control a standard endeffector position p_{eff} to be at y^*, as usual. Can you define a 1-dimensional task map $\phi : \mathbb{R}^n \to \mathbb{R}$ to achieve this? What is its Jacobian?

2 Verify some things from the lecture

On slide 02:30 we derived the basic inverse kinematics law. Verify that (assuming linearity of ϕ, i.e., $J\delta q = \delta y$) for $C \to \infty$ the desired task is fulfilled exactly, i.e., $\phi(q^*) = y^*$. By writing $C \to \infty$ we mean that C is a matrix of the form $C = \varrho C_0$, $\varrho \in \mathbb{R}$, and we take the limit $\varrho \to \infty$.

3 IK in the simulator

Installation instructions:

1. On the course website https://ipvs.informatik.uni-stuttgart.de/mlr/teaching/robotics/, there is a section 'Software' where you can find a link to the libRobotics software and an instruction on how to install it.
2. Download the code for this exercise here: https://ipvs.informatik.uni-stuttgart.de/mlr/16-Robotics/e02-code.tbz2
3. Unpack the code and copy the folder 'kinematics' into 'robotics15/share/teaching/RoboticsCourse/'
4. Compile the code by running '*make*' in the folder 'robotics15/share/teaching/RoboticsCourse/kinematics'

5. In the main.cpp there are two functions *simpleArrayOperations* and *openingSimulator* that show some basic operations of our code. In the main function you can set the 'mode' variable that selects which function to execute. You can also set the mode on the command line while executing the code with '*./x.exe -mode 1*'.

The goal of this task is to reach the coordinates $y^* = (-0.2, -0.4, 1.1)$ with the right hand of the robot. Assume $W = \mathbf{I}$ and $\sigma = .01$.

a) The provided function *reach()* already generates a motion in one step using inverse kinematics $\delta q = J^\sharp \ \delta y$ with $J^\sharp = (J^\top J + \sigma^2 W)^{-1} J^\top$. Record the task error, that is, the deviation of hand position from y^* after each step. Why is it initially large?

b) Try to do 100 smaller steps $\delta q = \alpha J^\sharp \ \delta y$ with $\alpha = .1$ (each step starting with the outcome of the previous step). How does the task error evolve over time?

c) Generate a nice trajectory composed of $T = 100$ time steps. Interpolate the target linearly $\widehat{y} \leftarrow y_0 + (t/T)(y^* - y_0)$ in each time step.

d) Generate a trajectory that moves the right hand in a circle centered at $(-0.2, -0.4, 1.1)$, aligned with the xz-plane, with radius 0.2.