Robotics
Exercise 3

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

October 26, 2014

1 Task spaces and Jacobians

In the lecture we introduced the basic kinematic maps $\phi^{\text{pos}}_{\text{eff},v}(q)$ and $\phi^{\text{vec}}_{\text{eff},v}(q)$, and their Jacobians, $J^{\text{pos}}_{\text{eff},v}(q)$ and $J^{\text{vec}}_{\text{eff},v}(q)$. 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) to solve certain motion problems. For this, keep in mind that inverse kinematics aims to minimize a squared error $\|\phi(q) - y^*\|^2_C$ in the task space.

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 IK in the simulator

Download the simulator code from http://ipvs.informatik.uni-stuttgart.de/mlr/marc/source-code/libRoboticsCourse.14.tgz. (See last exercise for instructions. The `libRoboticsCourse.13.msvc.tgz` might include a project file for MSVC.) The header `<src/Ors/roboticsCourse.h>` provides a very simple interface to the simulator—we will use only this header and some generic matrix functionalities. In `share/examples/Core/array` you find many examples on how to use the `arr` for working with vectors and matrices (many conventions are similar to Matlab).

Consider the example in `teaching/RoboticsCourse/01-kinematics` (rename `main.problem.cpp` to `main.cpp`). The goal 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 example solution 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.