CS545–Introduction to Robotics

Homework Assignment 1 (Due Feb. 26)

In the following problems, you will need the NAO simulator and Matlab for CLMCPLOT for other visualizations. You find relevant files and information under:

http://www-clmc.usc.edu/Teaching/TeachingIntroductionToRoboticsHomework

IMPORTANT: In your solutions of the homework, also provide intermediate steps how you derived the solution to a problem.

1. (130 Points) In this homework, you are supposed to create a complete point-to-point reaching behavior for the NAO robot. First, you will need to create a planning system for point-to-point movements for the endeffector (**right** hand) of the robot. Second, you will need to implement the plan on the robot simulator and visualize the outcome to confirm that it works.

a) For planning, you are to create a simple dynamical planning system:

$$\dot{x} = \frac{\alpha}{\tau}\left(x_f - x\right)$$

Assume the movement duration is τ, and the movement starts at t=0. Assume that at start the movement is at x_0 and it is supposed to reach x_f after τ. Use Simulink to simulate this system, by using a "Step Input" from the "Sources" library, a transfer function from the "Continuous" blocks library, and visualize the step input and the output of the transfer function in one Scope block. Add another Scope output that visualizes $\dot{x}$: connect the output of the transfer function to a "Derivative" block ("Continuous" Library), which feeds into another Scope. Set the simulation parameters to "Fixed-Step" with 0.001s step size. Provide a print-out of i) the mathematical analytical form of the transfer function, ii) your Simulink model (i.e., a picture of the Simulink block diagram), iii) the position and velocity plots from the Scopes for $\alpha = 1, \tau = 1, x_f = 1, x_0 = 0$. Note that x_f is the "final value" parameter in the Step block.

b) Adjust α such for $\tau = 1$, the movement has reached x_f to more than 99% after one second. Provide a print-out of your choice of α and the position and velocity plot (see above) for this choice. For your choice of α, also provide the same plots for $\tau = 2, \tau = 5, \tau = 10$.

c) Provide two pros and two cons about this trajectory planning system.

d) Write a Matlab program that implements this planning system with Euler integration, again using an integration step size of 0.001s — essentially, you implement now what Simulink did for you before. This program should have a subroutine $[\dot{x}_{n+1}, x_{n+1}] = f\left(x_f, x_n\right)$ which computes the next desired state $\dot{x}_{n+1}, x_{n+1}$ based on the current state and goal. This subroutine needs to integrate the current state x_n with the computed next velocity to the next state. Store all states $\dot{x}, x$ along the trajectory in vectors or a matrix, and plot the variables at the end of your program. Provide a print-out of your program and these plots. Compare against 1a) and comment on potential differences of your results.

e) A nicer version of dynamical system for planning would be the following:

$$\ddot{x} = \frac{\alpha}{\tau}\left(\beta\left(x_f - x\right) - \dot{x}\right)$$

with $\alpha = 25, \beta = 6$. Adjust your matlab program from d) to use this dynamical system as basis of planning. Note that you have a 2nd order system now which needs to generate position, velocity, and acceleration at the next time step. Create the same plots as in d) for a movement with $\tau = 1$. Comment on the differences, pros and cons of this new planning system.

f) Another favorable form of creating moving plans is a 5th order spline (minimum jerk spline):

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$

Assume the movement duration is τ, and the movement starts at t=0. Assume that at start the movement is at x_0, and it is supposed to reach x_f at time t=τ. Determine all the constants c_0 to c_5 as a function of x_0, x_f, and their derivatives, and τ. At start and end velocities and accelerations are $\dot{x}(0) = \dot{x}_0, \dot{x}(\tau) = \dot{x}_f, \ddot{x}(0) = \ddot{x}_0, \ddot{x}(\tau) = \ddot{x}_f$.

g) A useful way to implement the min jerk spline planning system is by creating a function that takes as input variables the current state $x(t), \dot{x}(t), \ddot{x}(t)$, the remaining time to go τ_{togo}, and the target state $x_f, \dot{x}_f, \ddot{x}_f$. The output of the function would be $x(t+\Delta t), \dot{x}(t+\Delta t), \ddot{x}(t+\Delta t)$, i.e., the planned state one time increment ahead. This function can be used to plan the next desired state given the current desired state, which is useful in a control loop of a robot. Note that the time-to-go τ_{togo} needs to be decreased at every iteration of the control loop by Δt. Implement this function in matlab, and create a movement with the same start, goal, and duration as in e), and provide the same plots of the trajectory. Compare the results of the min. jerk movement with the dynamic system from e) and comment on the differences and similarities.

h) The webpage has a file min_jerk_spline_task.cpp. Modify this file to implement your min jerk spline function from matlab. Run the task using the "setTask" command in the blue task_servo window. You will automatically collect a data file, and you should save it to disk with "saveData". Visualize the trajectories of all 5 right arm DOFs in CLMCPLOT in matlab, i.e., print the data traces R_SFE_th and R_SFE_des_th in the top most chart of clmcplot, and then the data traces for R_SAA, R_HR, R_EB, and R_WR in the next 4 charts. Create a print-out of your visualization and include it in your homework. Comment on the quality of tracking for the 5 DOFs. Repeat the same for the plot, this time using joint velocities, and then another plot using joint accelerations.

i) Add three additional targets to you C-program such that the left arm creates approximately a square in Cartesian space. Note that due to the joint-space movements, the hand movement in Cartesian space is curved. Your square should end at the same point where it started. Collect data of your movement and save it such that it can be visualized with CLMCPLOT. In clmcplot, click the "PhasePlot" button, and then click LEFT_HAND_x and then LEFT_HAND_y. This creates a window that plot LEFT_HAND_x in the horizontal axis, and LEFT _HAND_y in the vertical axis. Provide this plot in your homework and comment on what is good or bad about your realization of the square. Repeat this plot for LEFT_HAND_x and LEFT_HAND_z and provide this print-out as well. Comment on the quality of your square from this view.

j) Modify your C-code from h) and i) to work with the dynamic system planner from e) and provide the same results as in h) and i) for this planning method (make your C-code such that

it can switch between the two planning methods). Comment on the differences in performance from a technical point of view, and also from the point of view how movements look like in the simulator.