530.646 Final Project Report

Azwad Sabik

December 18, 2015

1 Overview

The goal of this final project was to utilize a UR5 to draw a picture on a 15cm x 15cm region of a whiteboard using both a method consisting solely of inverse-kinematics-based pose commands as well as a method additionally consisting of differential-kinematics-based pose commands. A point on the whiteboard surface and a unit (1cm length) normal pointing out of the plane of the whiteboard were provided in the UR5's world coordinate system. Only commands to the robot's joint-space position and queries of the robot's end-effector pose (both position and orientation) were permitted.

2 Methods

2.1 Path Generation

It was necessary to generate a sequence of target end-effector poses concordant with a two-dimensional "drawing" on the board for both the methods of inverse kinematics and differential kinematics. It was assumed that all target poses should maintain the same orientation relative to the surface of the board. In particular, it was decided that an optimal drawing pose would align the z-axis of the end-effector perpendicular to the board's surface, anti-parallel to the provided unit normal, and that the rotation of the pose about the z-axis was irrelevant. Thus, a simple two-rotation method was derived for generating an target orientation R_t with its z-axis aligned anti-parallel to an arbitrary vector n:

$$n = \frac{-n}{||n||}$$

$$\alpha = atan2(n_y, n_x)$$

$$\beta = atan2(\sqrt{n_x^2 + n_y^2}, n_z)$$

$$R_t = R_x(-\beta)R_z(-\alpha)$$

R_x and R_y here are simple rotations about the x-axis and y-axis respectively. Once a target-orientation R_t is calculated, the drawing's x-axis and y-axis are

chosen to be the first and second rows of R_t, referred to as R_{tx} and R_{ty} respectively from here onward..

Given a target orientation, a drawing frame (R_{tx} and R_{ty}), and a drawing center t_0 (chosen as the given point on the board, for convenience), a set of three dimensional target positions t_i for the end-effector can be generated from an input sequence of two-dimensional drawing target points (p_i):

$$t_i = p_{ix} * R_{tx} + p_{iy} * R_{ty} + t_0$$

Thus, a set of target poses are generated:

$$g_i = \begin{pmatrix} R_t & t_i \\ 0 & 1 \end{pmatrix}$$

2.2 Inverse Kinematics Method

The inverse-kinematics based method simply commanded the UR5 to each pose in the sequence of target poses by calculating an optimal (here, chosen as the set of joint angles which minimized the distance from the previous set of joint angles - this can be achieved in a simple manner by choosing the first set of angles from the set of all possible solutions for each pose) set of joint angles for each pose in the sequence. The results of this implementation in the V-rep environment can be seen in 1 and 2 in the results section. The results of the implementation using the lab's UR5 can be seen here `https://drive.google.com/file/d/0B09kwO89lJA4VmF1b0V3dEozbFE/view?usp=sharing`. (Please copy and paste url!)

2.3 Differential Kinematics Method

The differential-kinematics based method initialized the UR5 by commanding it to the inverse-kinematics based solution of the first target pose, and then proceeded to move the robot along the target trajectory using closed-loop control to move the robot from its current position, t_i, to each target position, t_{i+1}, along the path. The joint angle command q_{j+1} at each iteration of the control loop was calculated as follows:

$$\Delta t = t_{i+1} - t_i$$
$$\Delta q_j = K * J^{-1}(q_j) * \Delta t$$
$$q_{j+1} = q_j + \Delta q_j$$

until either $||\Delta t||$ fell below some tolerable error e_{max} or the final position in the path was achieved, $t_i = t_f$. The results of this implementation in the V-rep environment can be seen in 3 and 4 in the results section. The results of the implementation using the lab's UR5 can be seen here `https://drive.google.com/file/d/0B09kwO89lJA4ZEhBZTh5WENmVDQ/view?usp=sharing`. (Please copy and paste url!)

2.4 Extra: Line-Art Path Generation

A MATLAB class was create for the generation of UR5 trajectories based upon user-provided line-art. The user can call a PathGenerator object's add_points method to call up a MATLAB graphical input, draw a target trajectory by clicking (left click generates points on the board, while right click generates points at a distance obj.lift m above the board), and then generate a 3D UR5 trajectory (given an input center-point and board normal) at a scale of obj.scale (where 1 unit on the graphical user input is set to obj.scale in the output trajectory) by calling the object's generate_path method. The path is interpolated such that the UR5's target poses are at most obj.delta m apart to reduce likelihood of exceeding velocity limits. The object's draw method can be used to preview the user's drawing.

3 Software

The following files were central to this project's success:

- asabik2_inv_main.m - main script to perform inverse-kinematics demo
- asabik2_diff_main.m - main script to perform differential-kinematics demo
- asabik2_extra_main.m - main script to perform line-art demo
- rotation_aligned_to_vector.m - function to return rotation that aligns a frame's z-axis to an input vector n
- points2D_on_plane3D.m - function to convert a set of 2D points into a set of 3D points centered at an input point and on a plane perpendicular to an input norm
- optimal_ur5_inv.m - function to return optimal inverse kinematics solution given an input target pose
- get_jacobian.m - function that returns the spatial Jacobian for the robot given input of its current joint angles - NOTE, it's fine that this is spatial, the robot's provided path is never going to involve a change in pose-orientation. Therefore, only positional velocities need be considered.
- generate_continuous_pokeball.m - main path generation function
- PathGenerator.m - line-art generation class
- asabik2_final_project_ur5_scene.ttt - V-rep scene used for demos
- resources - folder containing everything else, mostly generated through earlier labs, some from TAs and Professor Cowan, and some from a kinematics package online (for twist exponentiation)

4 Results

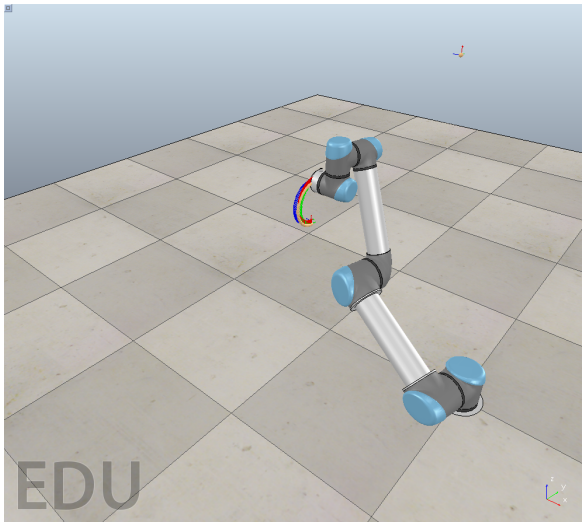

Figure 1: UR5 Drawing using Inverse Kinematics

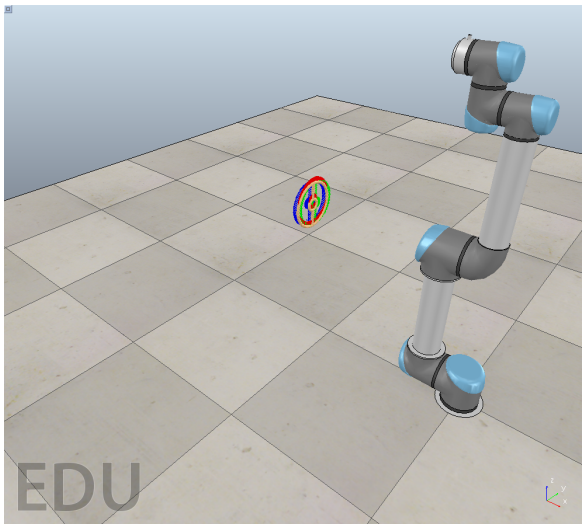

Figure 2: UR5 Inverse Kinematics Result

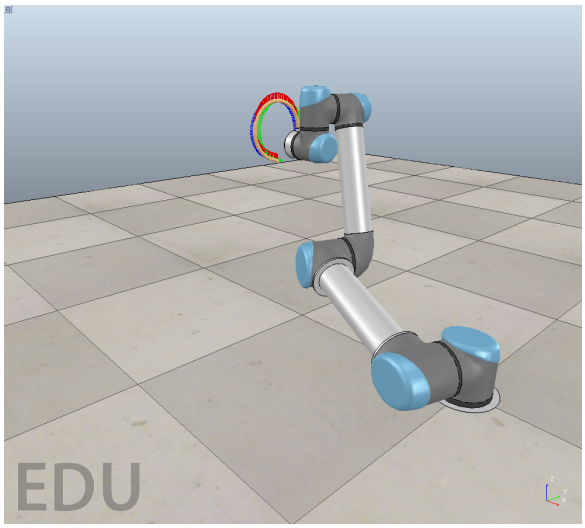

Figure 3: UR5 Drawing using Differential Kinematics

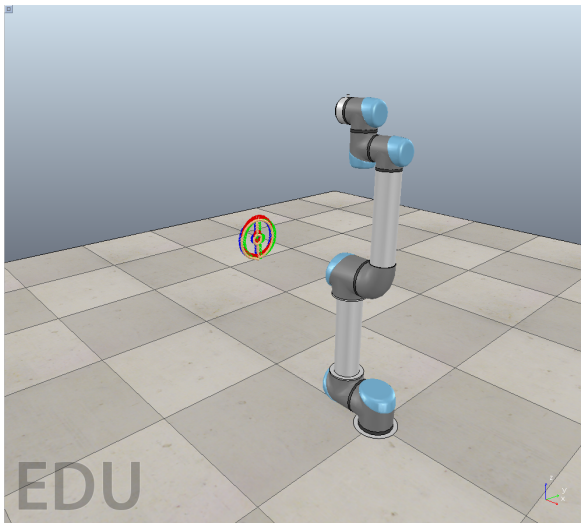

Figure 4: UR5 Differential Kinematics Result

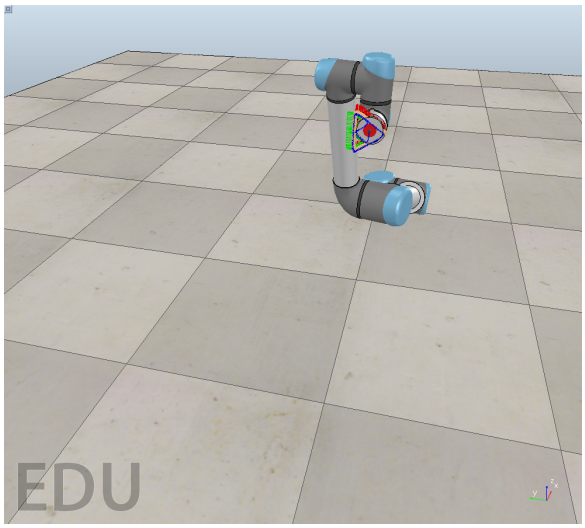

Figure 5: UR5 Drawing using Differential Kinematics with Line-art Path Generation

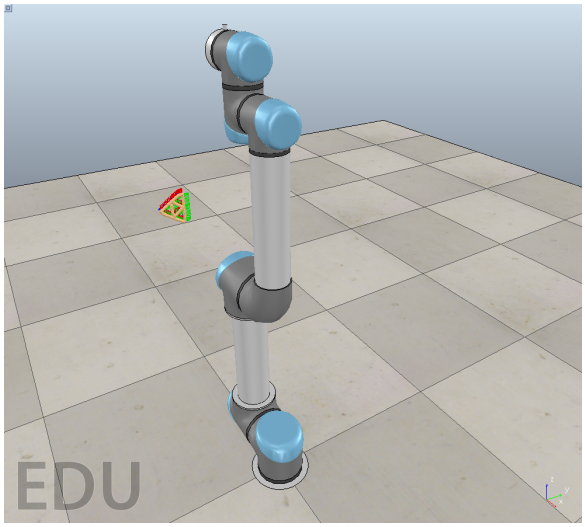

Figure 6: UR5 Differential Kinematics Result for Line-art Path Generation

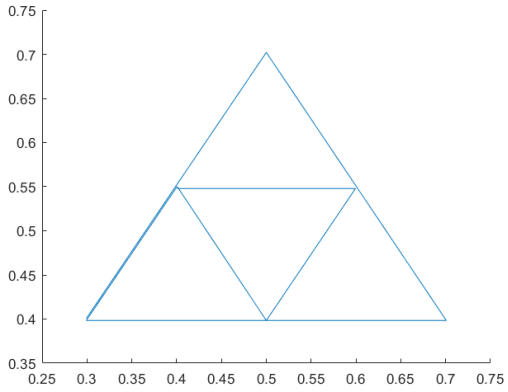

Figure 7: Goal for Line-art Path Generation