Foundations of Robotics, Fall 2017

Coding Homework 2: Kinematics
(CS 4750: 100 points, CS 5750: 120 points)

Due at the start of class, Fri, 6 Oct

Objective:

Demonstrate understanding of the mathematical formalisms underlying rigid body transforms, rotation representations, serial chain manipulator kinematics and mobile robot kinematics.

Instructions:

This assignment can be discussed as a group of arbitrary size. You may work with up to one partner on this assignment, but each student or pair of students is responsible for writing and submitting their own solutions to the problems below. Include in a comment at the top of each file your name and the names of all classmates you discussed any part of the homework with.

Use the simulator tool to run the provided code for the assignment. If you get errors about the launch file for a certain problem not existing, it may be that the framework code does not provide any programs for that problem. Please look in the `launch` subdirectory of the framework code (e.g. `src/foundations_hw2/launch`) to see which problems have provided code.

If you encounter difficulty using the simulator tool, please contact the course staff. Note that you can also run `./simulator-tool --help` to get a listing of commands and `./simulator-tool <command> --help` to get more information on the usage of a specific command.

Assignment:

Implement code to solve the following problems, using the provided simulator framework. Start by making a `catkin` workspace for your code. Create a new ROS package for this project in the workspace (call it "hw2"); write your code for each problem in a corresponding file in the package (named according to the convention "pX.py", where X is the problem number). Submit the complete ROS package on CMS in a single .tar.gz archive, named "solutions.tar.gz".

1 Forward Kinematics (20 points)

This problem is designed to assess your ability to compute the forward kinematics of a serial chain manipulator. Specifically, you will be computing the forward kinematics (FK) of a KUKA youBot [1] robot arm and developing a simple service that implements the computation of an FK query. Appendix A provides the design specifications for the youBot robot arm. You should be publishing to the target topic `'/vrep/youbot/target/position'` to make the target ball in the simulation move and test your results.

(a) (*10 points*) Derive the kinematic parameters of the KUKA youBot robot arm, following the

Denavit-Hartenberg convention.

(b) (*10 points*) Design a ROS service that computes the forward kinematics for a given arm configuration. You can get arm joint configurations for testing from `'foundations_hw2/arm_config'`.

2 Velocity Kinematics (30 points)

This problem is designed to assess your understanding of velocity kinematics and your ability to implement simple velocity kinematics computations.

(a) (*15 points*) Consider again the KUKA youBot robot arm. Compute its Jacobian matrix.

(b) (*15 points*) Design a ROS service that numerically computes the Jacobian matrix of the youBot arm at a given configuration. You can get valid arm joint configurations from the ROS topic `'foundations_hw2/arm_config'`.

3 Inverse Kinematics (30 points)

This problem is designed to assess your understanding of inverse kinematics and your ability to implement simple inverse kinematics computations. You should be publishing to the joint angles `'/vrep/youbot/arm/jointX/angle'` where X is the joint number (from one to five) with type `'std_msgs/Float64'` to move the simulated youBot arm. You will also use your implementation of the Jacobian from the previous problem.

(a) (*10 points*) Design a ROS service that computes the Inverse Kinematics of the youBot arm for a given end effector position by employing the method of the iterative method of the Jacobian transpose. You can get the effector position from `'foundations_hw2/effector_position'`. Note that your implementation should not move the robot.

(b) (*20 points*) Assume that the end-effector of the youBot arm is a pen. Consider the task of using that pen to write a word/phrase on a whiteboard. Making use of the IK service of the previous question, design a ROS service that writes a word or phrase of your preference on a virtual whiteboard in the air. You may assume that the whiteboard is a selected perpendicular plane in the proximity of the robot.

4 Rotation Representations (25 points)

This problem is designed to assess your understanding of different rotation representations. You will be interpolating between a pair of orientations of an object in 3D. Interpolation means that you define a function $f(t)$ that smoothly maps onto orientations of the object, where $f(0)$ gives the initial orientation and $f(1)$ gives the final orientation. As t varies in the range $[0, 1]$, you will see the object smoothly change in orientation on the screen from the initial to the final orientation.

Each time you call the `foundations_hw2/interpolate_problem` service, it returns an initial and final orientation expressed as Euler angles in the current frame ZYX convention. It also returns a time in seconds. You will write two nodes using two different approaches to interpolate from the initial to

the final orientation over the given number of seconds. To perform the interpolation smoothly, you should publish messages of type `foundations_hw2/EulerAngles` to the `vrep/shape_pose` topic at a rate of 10 Hz. Since you are setting the orientation of the shape at a high rate, it will appear visually to move on the screen in the simulator.

(a) (*10 points*) Write a ROS node that directly interpolates Euler angle values.

(b) (*15 points*) Write a ROS node that performs the interpolation in quaternions. You will need to convert the initial and final orientations to quaternions and then perform spherical linear interpolation or Slerp (see for example http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/slerp/, https://en.wikipedia.org/wiki/Slerp).

5 Mobile Robot Kinematics (15 points)

This problem is designed to assess your understanding of different types of mobile robot kinematics.

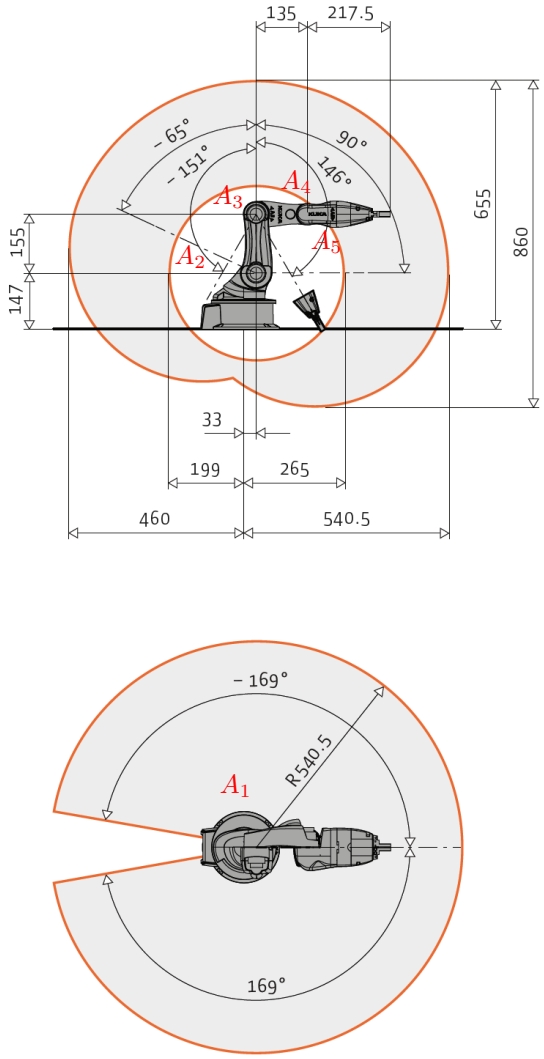
You can subscribe to '`foundations_hw2/cmd_vel`' for the velocities of type '`geometry_msgs/Twist`'. You should be publishing to '`vrep/youbot/base/cmd_vel`' and '`vrep/ackermann/cmd_vel`' to move the robots.

(a) (*15 points*) Implement a ROS node that subscribes to '`foundations_hw2/cmd_vel`' to get velocity commands. The node should then convert each command as necessary to the format of command motions needed for the KUKA youBot (with an omnidirectional base) and the car model (with Ackermann steering). Publish each velocity command the node receives to both topics to command the two vehicles.

References

[1] R. Bischoff, U. Huggenberger, and E. Prassler. Kuka youbot - a mobile manipulator for research and education. In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

A KUKA youBot Design Specifications



youBot arm

Serial kinematics	5 axes
Height	655 mm
Work envelope	0.513 m³
Weight	5.8 kg
Payload	0.5 kg
Structure	Magnesium Cast
Positioning repeatability	0.1 mm
Communication	EtherCAT: 1 ms cycle
Voltage connection	24 V DC
Drive train power limitable to	80 W

Axis data	**Range**	**Speed**
Axis 1 (A1)	+/- 169°	90°/s
Axis 2 (A2)	+ 90°/- 65°	90°/s
Axis 3 (A3)	+ 146°/- 151°	90°/s
Axis 4 (A4)	+/- 102°	90°/s
Axis 5 (A5)	+/- 167°	90°/s

Gripper	**Detachable, 2 fingers**
Gripper stroke	20 mm
Gripper range	70 mm
Motors	2 independent stepper motors

Figure 1: Design specifications for the KUKA youBot robot arm. The schematics on the left provide basic dimensions (linear dimensions are in *mm* and angular dimensions are in degrees), whereas the tables on the right provide other technical specifications, including the limits of the rotational joints and the gripper aperture.