

**Department of Mechanical Engineering**

**Rice University**

**MECH/ELEC/COMP 498/598: Introduction to Robotics**

**Lab #2**

**Inverse Kinematics: Robo Picasso**

*Due Thursday, March 21st at 8am*

**Assignment Description**

This assignment will most likely be very time-consuming, meaning you should start working on it well in advance of the deadline. **For this assignment, undergraduate students may work in groups of 2.** You will be required to hand in **individual** paper submissions, and a group electronic submission. The lab consists of solving the inverse kinematics problem for a 6-DOF robot, and simulation your solution in python. The goal is to get our simulated industrial robot to “draw” in 3D using a tool with different colored “brushes.” Some code has been provided for you on Canvas as well, with additional instructions in the code comments. This code deals heavily with Python classes. If you are not familiar with object-oriented programming and how classes work, please take some time to familiarize yourself with it before jumping into the lab. You will be provided with a partially complete Fanuc class, and additional complete classes. These complete classes make up parts of the Fanuc, allow for drawing, and allow easy calling and data storage. You may modify the provided classes if you’d like, but you do not need to in order to complete the assignment.

**Submission Instructions**

Your submission for this assignment will have an electronic and a paper component. The paper submission can be turned in to the box outside of the MAHI Lab, to the MECH dropbox in the Mechanical Engineering Building, or in class. For the electronic submission, just as you did in the previous lab, you should create a folder named **[your\_net\_id]\_AND\_[teammate\_net\_id]\_lab2**, where your Rice ID is your shortened email address, e.g. “cgm4.” Then, upload a compressed form of this folder named **[your\_net\_id]\_AND\_[teammate\_net\_id]\_lab2.zip** to the assignments page on Canvas, as you did before. This submission folder should contain one file **fanuc.py** that contains your unique calculations as described in the questions below. In addition, it should include all other files that were provided to you that are necessary for the class to operate including, **general\_utility.py**, **drawing\_helper.py**, and **robot\_components.py**. If you modified any of these additional files, please include your modified versions. Remember, you should *not* have to modify the additional files if you don’t want.

Be sure that your code runs properly with only the files you submit for the lab. If the code does not run, the auto-graded portions will be zero. (That’s quite a bit of the grade)

1. Fill in the **dh\_tf(alpha, a, d, theta)** helper function at the top of the **fanuc.py** file. This function accepts the 4 D-H parameters and returns the corresponding 4x4 homogeneous transformation matrix as a numpy array without any unnecessary computations or function calls. (i.e., don’t reuse **screw\_tf()** from Lab 1.) [1 point]
2. Using the attached specification sheet for dimensions of the FANUC S-500 robot, complete the unfinished areas of the class initialization function **\_\_init\_\_().** This will include setting the joint lengths and creating the overall workspace the robot will operate in. Keep all lengths in millimeters. As part of this setup, fill out the **\_setup\_joints()** function to properly create each joint and fill in its color and limits.

Next, fill in the function **calculate\_fk(joint\_angles)**. This function takes in a 6 element vector of joint angles. It should check that the joint angles are within the allowed boundaries, then set the dh transform using the **set\_dh\_transform()** function in each joint. You will then be able to grab each joint’s homogenous transformation matrix as well as the position of the end effector using the provided property **ee\_frame**. (If you’ve not seen Python properties before, give it a quick google before continuing)

Here are some notes on creating your DH parameters. As is done for the PUMA 560 robot in the textbook, frame {0} should be coincident with frame {1} for ease of constructing the forward and inverse kinematics. In this lab we will use the terminology “base” frame for the origin of our graphical representation which occurs at the actual base of the robot. Meanwhile, frame {0} will be referred to as the “station” frame. Use a station frame (frame {0}) with  pointing up and  pointing horizontally forward in front of the robot. The end effector frame, frame {6}, will have  pointing along the end effector’s axis of rotation and its x axis point in the same direction as  when in the zero configuration.

Lastly for this problem, use the function **draw\_fanuc(joint\_angles)** to draw the FANUC at different positions. Part of the robot that has been created for you is the Brush class. This creates 4 different “brushes” that come out at different angles from the end effector. You can change which brush the robot is using by setting the brush from 0 (no brush) or any of the four brushes (1-4) by **self.brush.selection = value** where value is the desired number. The best way to do this is to create a small script you can run separately that will create your Fanuc class, then you can set the brush, call the forward kinematics, and draw the robot to see different configurations and brushes. Play around with this to get to know your new robot! [19 points]

**Note:** The side-view drawing is the zero position, but if you follow D-H convention, you will be required to include an offset in one of the joint angles (probably joint 2). Also, there is a missing dimension in the vertical direction between the joint 3 and joint 4 axes, which should correspond to *a3*. Consider this distance to be 180 mm. Finally, assume the joint ranges given are to be centered about the robot’s zero position.

1. Fill in the function **calcukate\_ik(ee\_frame, prev\_joint\_angles).** This takes in the location of the end-effector (NOT the tool/brush) and a 6-element vector of the previous FANUC joint angles. It should return a Boolean of whether of not a solution exists and, if it does, the 6 joint angles that make that solution closest to the previous joint angles. (You will have multiple solutions much of the time, be sure to pick the closest option to avoid large jumps in joint angle)

You should check to see if a solution exists by checking if the **ee\_frame** is withing the workspace and that the resulting joint angles are within the boundaries of the joints you set when you setup your Fanuc.   
**Written Assignment:** Sketch the position of the robot when all joint angles are zero with the frames labeled, and fill out a D-H parameter table. This will be included in the written portion of your submission, and there should be one from **each** member of your team. [30 points]

1. Using **calculate\_fk** and **calculate\_ik,** complete the function **draw\_fanuc\_path(path\_file).** For this method, only the framework is provided. You will need to take in a string of the path to your .yaml file. The provided code converts this to a dictionary, and you must take it from there. The yaml file is a discretized path through space for the x, y, and z locations as well as the brush colors at each location. Note: Orientation is not provided, that’s up to you to pick. (I often left mine constant). These paths should move your robot continuously through space. These points are the desired location of the BRUSH tip, not of the end-effector, be sure to convert the brush location to end-effector location. Also, be sure that the transition from brush to brush is smooth. When moving through the path, the robot should move smoothly and be continuous, there should not be large jumps in joint angle at any time.

Your function should call **draw\_fanuc()** each time you want to draw the robot. Be sure to change the brush you’re using depending on the color selected as well.

Two path options have been provided to you to try out, **tetra.yaml** and **prism.yaml**. Use these to test out your robot. Note, that these shapes should be continuous, there are no large gaps that should exist, if there are, double check your work! [30 points]

1. Draw something colorful! The path files provided were simply for testing, and are not very interesting. They also trace over themselves with different colors. This part of the assignment is intentionally open ended, and you will quickly realize that you can take an increasingly more sophisticated and approach to handling the problem if you wish to put in the work. The better your IK solver and drawing code works, the more points you will be awarded. Provide both the yaml file(s) you create to draw as well as still images (screenshots) of what you drew. [20 points]
2. Each team member should include in their written submission, in addition to the FANUC forward kinematics, a brief paragraph describing the ways in which they contributed to the project, and a separate paragraph detailing all the ways in which this *simulation* of robot motion is a simplification of the real problems encountered in robot motion control.



