Jarvis - Object Sorting

Lohitaksh Gupta
Electrical and Computer Engineering
College of Engineering
University of Illinois
Urbana - Champaign, IL 61820, USA
lg6@illinois.edu

Miguel Jimenez Aparicio
Electrical and Computer Engineering
College of Engineering
University of Illinois
Urbana - Champaign, IL 61820, USA
miguelj2@illinois.edu

Kehan Long
Electrical and Computer Engineering
College of Engineering
University of Illinois
Urbana - Champaign, IL 61820, USA
kehan2@illinois.edu

***Abstract* —Moving one object from its initial position from a desired position. We are going to use V-REP. The robotic arm, UR-3, will grasp the object from a random initial pose to a given final pose, which will be calculated beforehand. Concepts such as Forward Kinematics, Inverse Kinematics and Motion Planning will be used in order to implement the sorting of objects.**

Keywords — Forward Kinematics, Inverse Kinematics, Motion Planning, Object Sorting, Robotic Arm, UR-3 and V-REP.

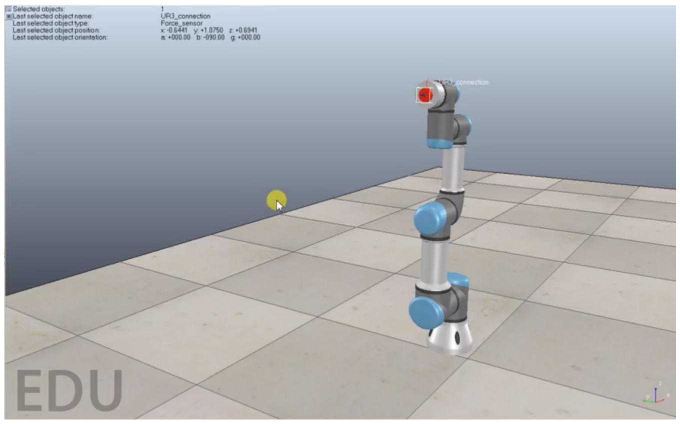
I. Introduction

Created a dynamic simulation (i.e., a simulation of real physics) in which at least one robot---with at least one arm that has at least six joints---moves at least one object (e.g., by pushing or grasping) from a random initial pose to a given final pose.

Robot: UR-3

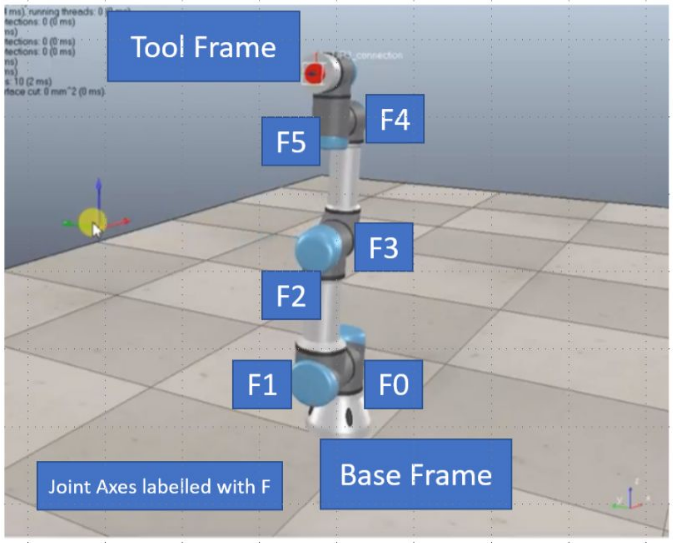
Simulator: V-REP

Picture of robot in a simulator:



II. Forward Kinematics

A. Schematic of the robot in the zero configuration



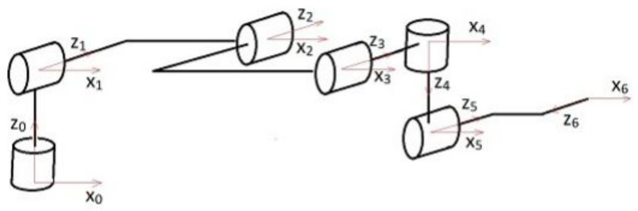


Figure 3.1: UR3 robot with DH frames assigned. Suction Cup Gripper Attached.

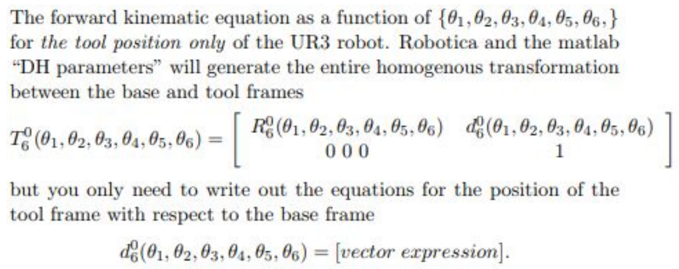
B. How to compute the pose of the tool frame

To calculate the forward kinematics of an open chain using the space form of the PoE formula, we need the following elements:

(a) the end-effector configuration M ∈ SE(3) when the robot is at its home position.
(b) the screw axes S1, . . . , Sn expressed in the fixed base frame, corresponding to the joint motions when the robot is at its home position. For this, we used the dimensions that we measured in one lab session.
(c) the joint variables θ1, . . . , θn.

$$T(\theta) = e^{([B1]\theta 1)}\, e^{([B2]\theta 2)} \cdots e^{([B6]\theta 6)} M$$

C. The derivation of the data that implement the computation



The forward kinematic equation as a function of $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6,\}$ for *the tool position only* of the UR3 robot. Robotica and the matlab "DH parameters" will generate the entire homogenous transformation between the base and tool frames

$$T_6^0(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \begin{bmatrix} R_6^0(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) & d_6^0(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

but you only need to write out the equations for the position of the tool frame with respect to the base frame

$$d_6^0(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = [\textit{vector expression}].$$

III. Inverse Kinematics

A. Computation of joint variables (Algorithms)

It is an iterative code that starts with an initial set of thetas and eventually converges in the the set of joint angles that result in the desired pose.

Once the first pose is calculated, we calculate then the jacobian matrix. With that and with the required twist between the actual pose and the final pose, it is possible to compute the angular velocities.

The set of thetas is updated adding these velocities multiplied by a small constant.

The algorithm stops when the norm of the body twist is below 0.01 (no more velocities are needed).

B. Discussion

- Tool poses for which no solution exists

Some tool poses can't be achieved because the calculated pose is out of the joint range of each joint. For example, we used UR-3 robot arm to simulate our project; during the simulation, we found that the robot arm can't achieve for the set of points which $z<0$. And, the set of points can't be achieved if the distance between the points and the initial center is greater than the sum of all arms' lengths.

- Tool poses for which more than one solution exists

In most cases, if the tool pose can be achieved, there are always more than one solution exist. For example, for most of the poses inside the maximum range of the robot arm, since all the six joints of the robot arm are revolute, a given configuration can be achieved by at least two ways.

- Tools poses for which the only solutions are singular configurations

There are a few poses that the inverse kinematics can only give solutions that are singular configurations. For example, the highest pose that the robot arm can achieve always has only one singular configurations.

But, in our algorithm, our code always return one solution for each tool pose even though there exists more than one solution.

IV. Motion Planning

A. How to decide if a given set of joint variables places a robot in collision

There are several spheres (dummies) located across the robot. When the robot moves according to a given set of joint angles, the position of each sphere is recalculated.

In order to know if two spheres are in collision, it is checked if the norm of the distance of the vectors is smaller or larger than the sum of the radius.

B. How to compute a collision-free path

First, the Inverse Kinematics function is used to find the set of thetas of the final pose.

After that, from a initial set of thetas (the first node of the tree), a new configuration is chosen and then it is checked if there is collision between both sets checking it for 1000 set of thetas in between.

Only in the case that there is no collision, this new configuration becomes the next node of the tree. This process is repeated until the last set of thetas matches the final set.

Finally, the UR3 follows all the set thetas until it reaches the final position.

C. Discussion

- Collision detector returns a false positive or false negative.

It sometimes happens.

The reason of a false positive is that the spheres that cover the joints are too big. A better way to make the collision detection would be using more and smaller spheres to cover the robot once collision is detection and check it again.

A false negative is produced when the collision happens in a zone not covered by any sphere. It can be solved covering the robot with more spheres.

- Choices that no collision-free path exists between given start and goal pose.

If the final pose is inside one of the obstacles the algorithm is not going to find a solution. Currently it would just run forever but a counter of iterations could be set and stop the algorithm if it reaches a number.

Also, as the obstacles are also dummies, it could be easy to check if the final pose is inside the volume of the obstacle.

- How much time is required to plan a collision-free path

A few seconds.

V. Future Work

A. *Final goal by the end of the semester*

Our future goal is to build a robot that takes an object from its initial position and place it in a chosen destination.

B. *Plan to achieve the goal*

We will first start our implementation with one object and place it on a pre-calculated position. We need to attach a gripper as and an effector of our robot, so we can move the object.

We have already implemented and tested our Inverse Kinematics and Motion Planning code.

Acknowledgment

The team worked with ECE 470: *Introduction to Robotics* faculty. Especially, Professor Tim Bretl and Teaching Assistants: Hang Cui, Xinke Deng, Siwei Tang and Jifei Xu, helped the team in dealing with the technicalities of python and V-REP.

In addition to that, Piazza posts helped the team in getting a better sense of the different parts of the project.

References

- Modern Robotics: Mechanics, Planning, and Control by Kevin M. Lynch and Frank C. Park

  Link: http://hades.mech.northwestern.edu/images/7/7f/MR.pdf
- ECE 470 Lecture Slides and Supplemental Notes by Professor Tim Bretl.
- ECE 470 Lab Manual

  Link: http://coecsl.ece.illinois.edu/ece470/lab.pdf