ECE Final Project Report (Fall 2021)

Bruce Wang(boyongw2), Hongbo Yuan(hongboy2), John Kim (jaehank2)

GitHub: https://github.com/BOYONGw/470pick-and-plce.git

YouTube link: https://youtu.be/caWtbhZaHSs

Pick and place challenge is most important task in robotic application. In nowadays, Logistics classification and quality inspection are widely use in industry. We are going to use V-REP simulator to show how our robotic arm, UR5, will suck a cube from the belt. The robotic arm will be placing the cube into another conveyor. A camera three ray sensor will be used in the pick and place challenge. 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, proximity sensor, vison sensor Suction Cup, Robotic Arm, UR5 and V-REP.

1. Introduction

The minimum requirement for this project was given in this following statement:

“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.”

Going beyond that, we decided to attach a suction gripper to our robot and pick the object - a cube - from their initial conveyer belt and then release it in a conveyer belt. A vison sensor will help to identify the color of the cubic and then a ray sensor will stop when the vison sensor detects the preset color range. The robot arm will suck the cubic to another conveyor belt and place it into same direction. After that, the second conveyor belt will move up to make sure the there’s enough space for the next cubic.

The final goal of our project is to show the capability of the robot to perform tasks both in industrial and household tasks, such as reallocation and change the color of objects and automated search of a path without collision with itself and the environment.

For this project, we have used the following items:

- Robot: UR-5
- Simulator: V-REP
- Programming Language: Lua

In the figure 1, our UR-5 robot is shown. It is possible to notice the Baxter suction cup attached as its end effector. the scene with two conveyor belt, different color of block, a cone style proximity sensor located at the end of the first conveyor, two ray sensor locate at the front and end of the second conveyor.

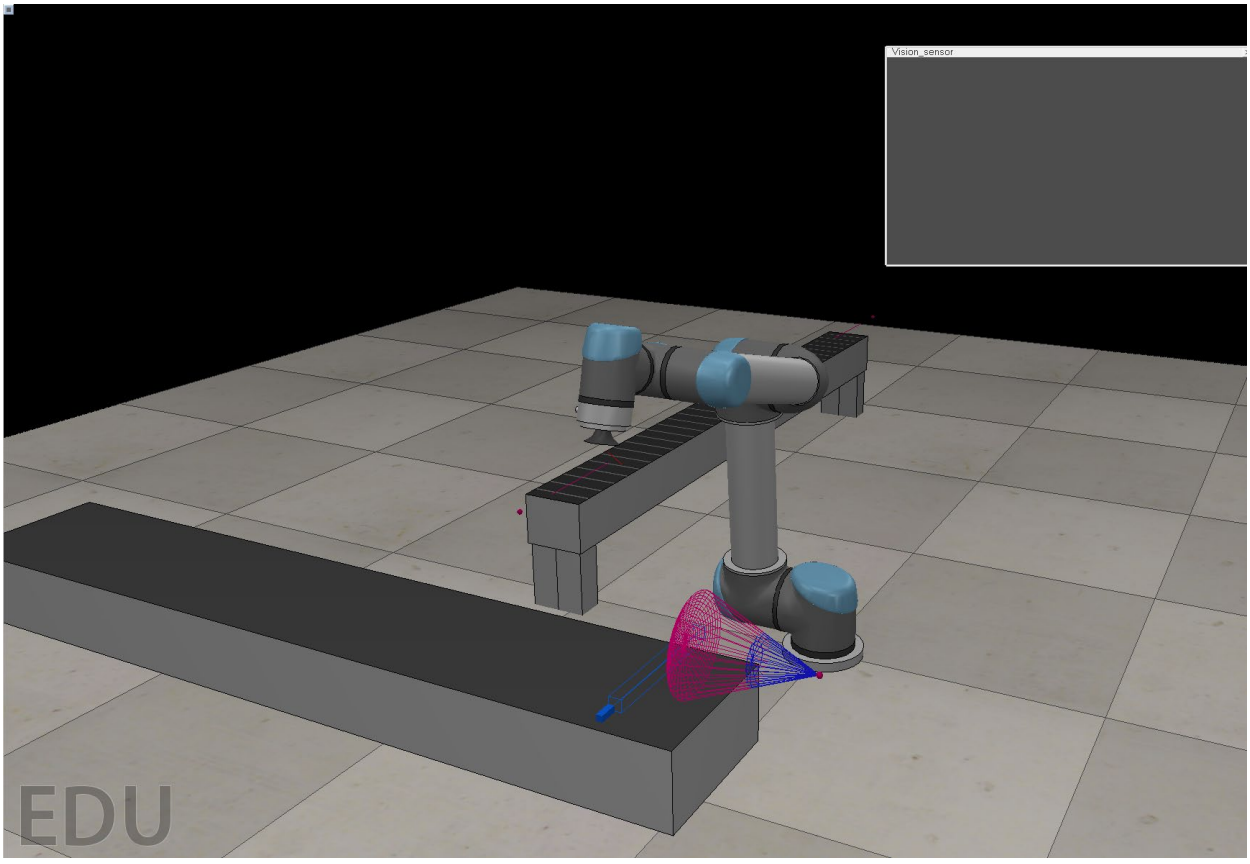

Figure 1 V-REP environment set

Solution approach:

The first goal is to enable the vision sensor to detect the color of the block, and then send a signal to the ray sensor, when the ray sensor detects the block, stop the conveyor. And then is to enable the suction cup on the robot arm, making it able to pick up some object in the scene. The most important approach we use is inverse kinematics, we calculate the joint angles with the given pose each time, i.e. where to release the cube on the conveyor belt. The inverse kinematics give us the joint angles so that we can move the robot to the given pose.

Result:

1. Once the cubic have the correct color, the conveyor belt will stop a given location.

2. The robot arm picks up the block the robot arm and put the block to the second conveyor.

3. Ones the block dropped on the second conveyor, the belt will move forward to make sure there will be enough space for the next block.

2. Forward Kinematics

2.1Schematic of the robot in the zero configuration

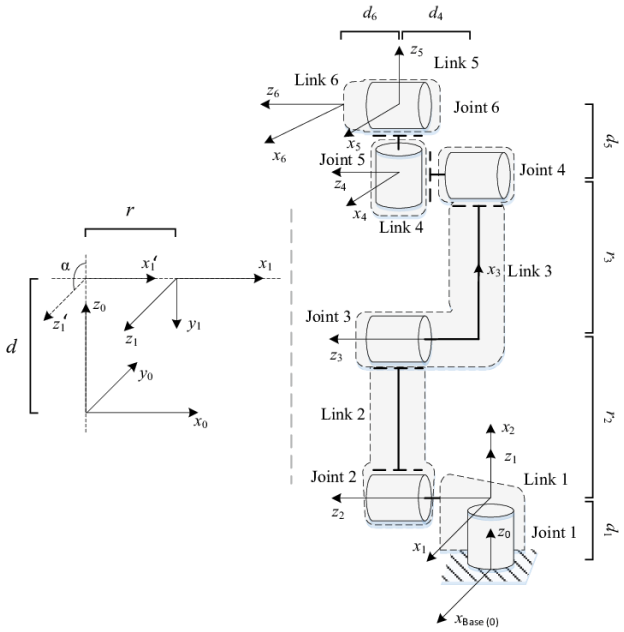

Figure 2 UR5 robot parameters by Denavit-Hartenberg method.

2.2 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:

1) Home position of the end-effector configuration M ∈ SE (3)
2) the screw axes expressed in the fixed base frame, corresponding to the joint motions when the robot is at its home position.

2.3 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 UR5 robot. Robotics and the MATLAB "Denavit-Hartenberg 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 we only need to write out the equations for the position of the tool frame with respect to the base frame

$$d_6^0(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = [Vector\ expression]$$

3. Inverse kinematics

3.1 Computation of joint variables (Algorithms)

It is an iterative code that starts with an initial set of theta and finally converges to a set of joint angles that lead to the desired pose. Once the first pose is calculated, we can calculate the Jacobian matrix. With this and the required distortion between the actual pose and the final pose, the angular velocity can be calculated. Multiply these velocities by a small constant to update the thetas set. When the norm of body distortion falls below 0.01 (no more speed is needed), the algorithm stops.

3.2 Constraint

Tool pose without a solution Some tool poses cannot be achieved because the calculated pose exceeds the joint range of each joint. For example, we use the UR-3 robotic arm to simulate our project; during the simulation, we found that the robotic arm cannot achieve the point set of z<0.

Moreover, if the distance between the point and the initial center is greater than the sum of all arm lengths, the point set cannot be achieved.

There are multiple solution tool poses. In most cases, if the tool pose can be achieved, there will always be multiple solutions. For example, for most of the poses within the maximum range of the robot arm, since the six joints of the robot arm are all rotating, a given configuration can be achieved in at least two ways.

The only solution is a single configuration tool posture. In some postures, inverse kinematics can only give solutions to singular configurations. For example, the highest pose that a robot arm can achieve is always only a single configuration. However, in our algorithm, even if there are multiple solutions, our code always returns one solution for each tool pose.

4. Motion planning

4.1 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.

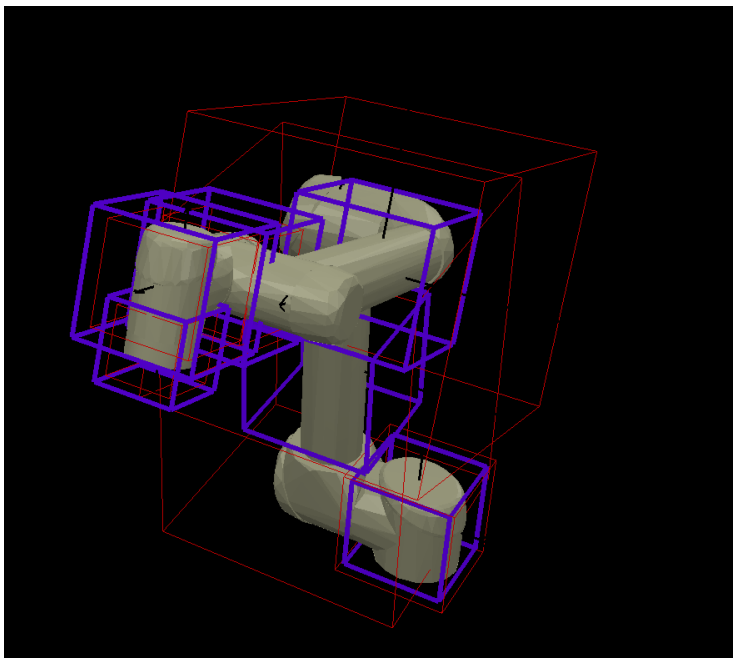

Figure 3 collision analysis

4.2 Collision-free path

First, the inverse kinematics function is used to find the theta set of the final pose. After that, from the initial set of thetas (the first node of the tree), select a new configuration, and then check whether there is a conflict between the two groups, and check whether there are 1000 sets of thetas in between. Only if there is no collision, this new configuration becomes the next node of the tree. Repeat this process until the last set of theta matches the final set. Finally, UR3 follows all set thetas until it reaches the final position.

5. Sensor selection

Sensor we have selection: cone type Proximity sensor, ray type Proximity sensor, orthograph vison sensor.

Cone type proximity sensor was located at the end of the conveyor belt. The reason of the cone type is in used is because the block will drop randomly on the belt. the block may locate at the side of the conveyor, so we need a cone type proximity sensor to detect more area.

Orthograph vison sensor will locate before the cone type proximity sensor, to ensure the color will be detect before the proximity sensor.

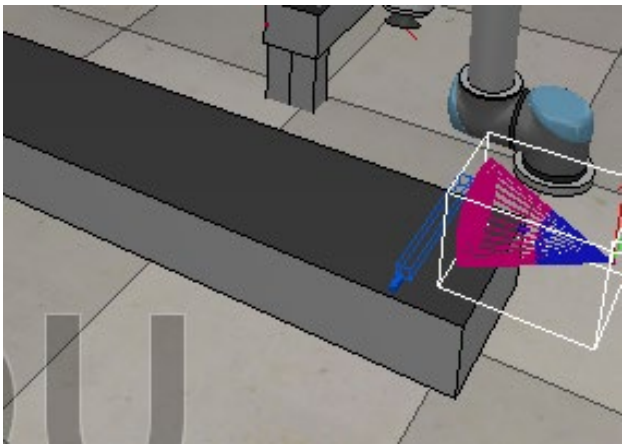

Figure 4 Sensor Position

The ray type proximity sensor is located at the front of the second conveyor. The reason to have a ray type instead of cone type is because the robot arm will put the block in line, so every time the robot arm will place the block in a same location.

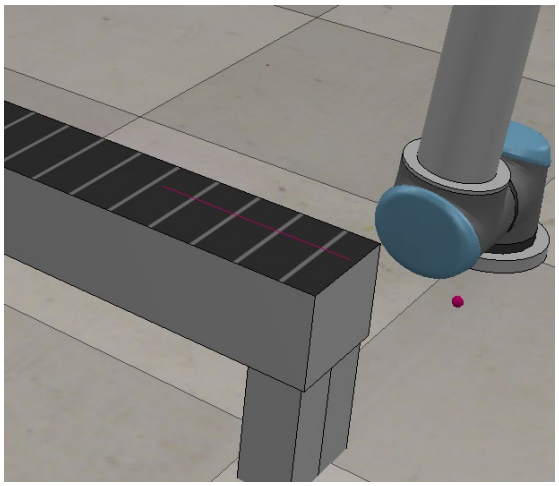

Figure 5 Sensor

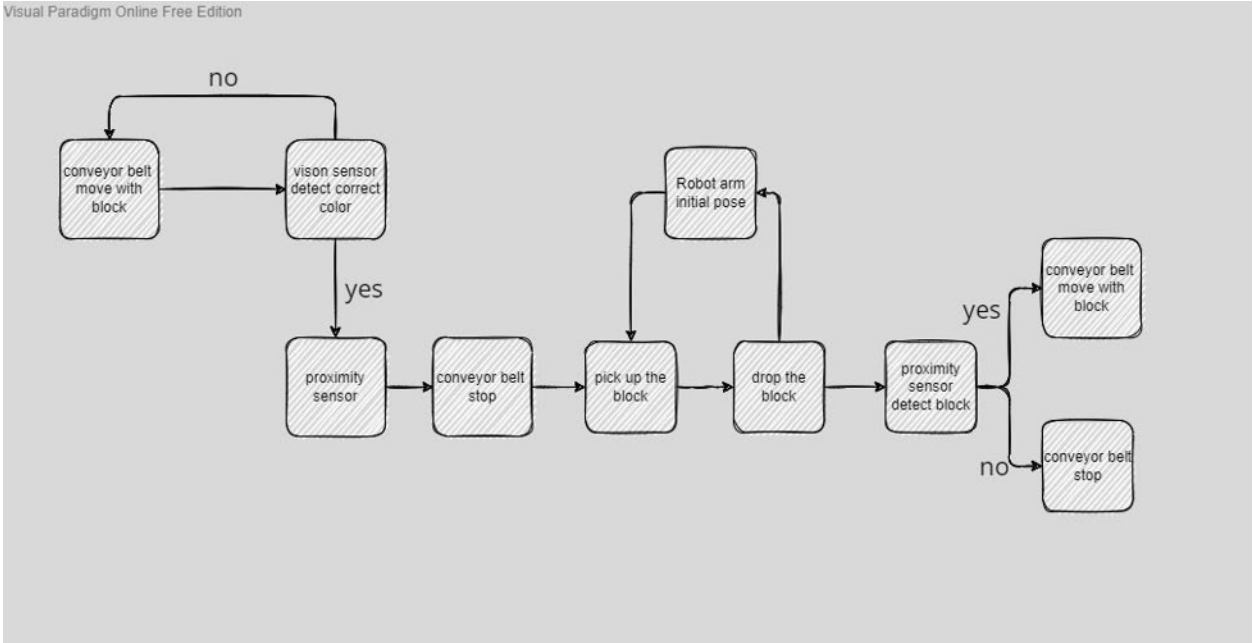

Figure 6 Flowchart of the Robot

6. Result

Results of any experiments during simulation. First We set the program to give random color of block and drop it on the conveyor belt, second the vison sensor will detect the color of the block. When the proximity sensor detect with the correct color conveyor will stop

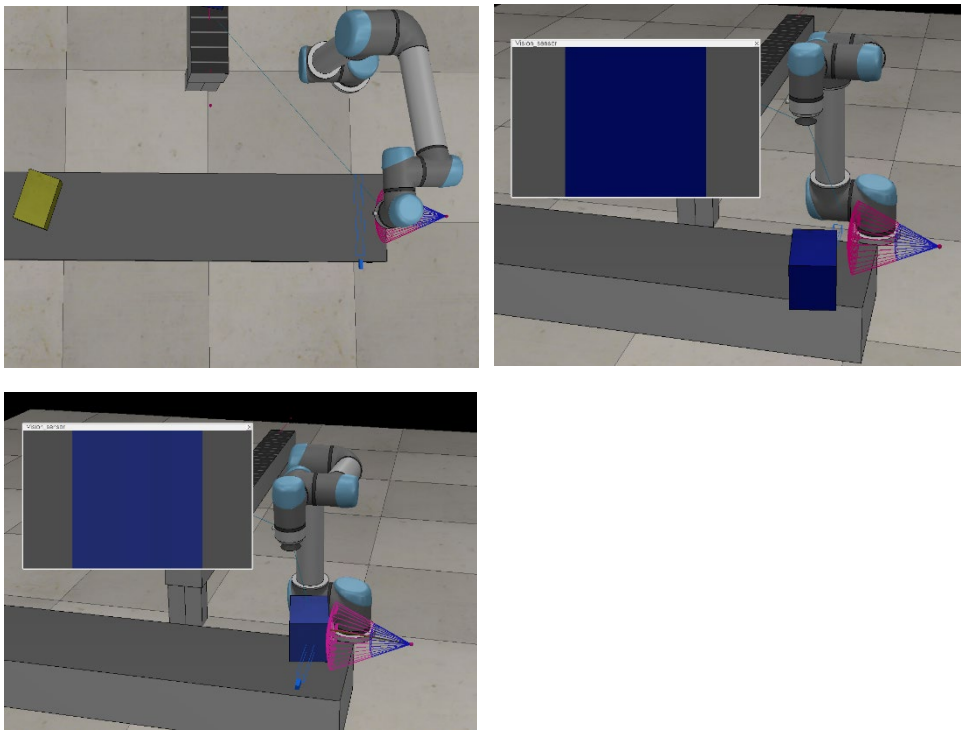

Figure 7 Simulation

Inverse kinematics function will calculate the six joint angles for the six joints. Finally, V rep to move our robot arm to the given pose and set the suction cup on to pick up the cuboid in the scene.

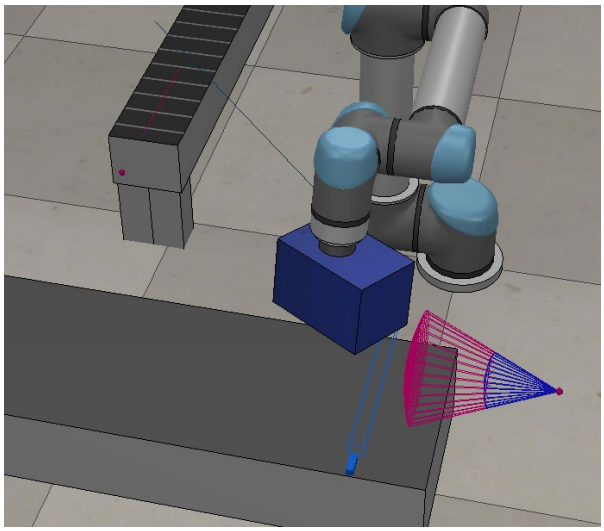

Figure 8. Releasing Cube

Similarly, we set the point we need to reach and called the inverse kinematics to find joint angles again. Besides, this time we also called collision check function to avoid collisions when finding the path to the conveyor belt. Finally, we dropped the cuboid by setting the signal to 0 and then we moved the robot arm back to its home position and get ready to pick another blue block.

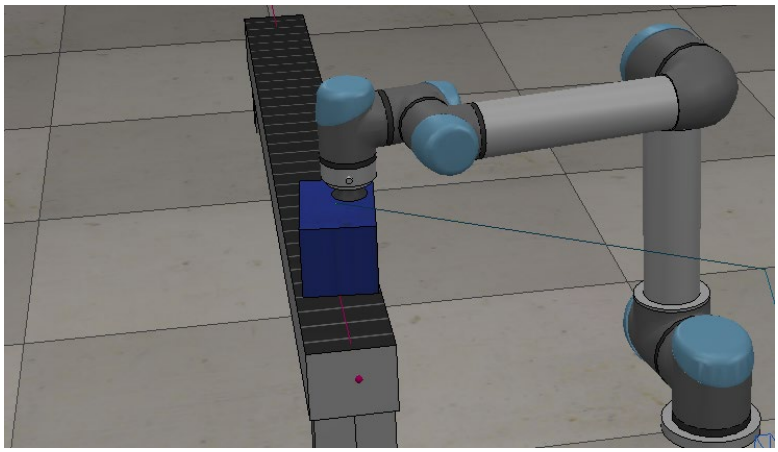

Figure 9. Placing the Cube

Our simulation has completed the pick and place challenge, and system can be detected different color of the blocking. This kind of sorting problem is one of the most common systems in the real-life project. The system work with the proximity sensor, and the vision sensor. We also have the collision check and find the collision free path. This is important when we have something more difficult environment.

Failure Result

We test 50 times in V-rep and 86% percent of simulation run correctly. The top three problem occur during the simulation is unable to pick up the black, miss the correct color block and block did not place in line.

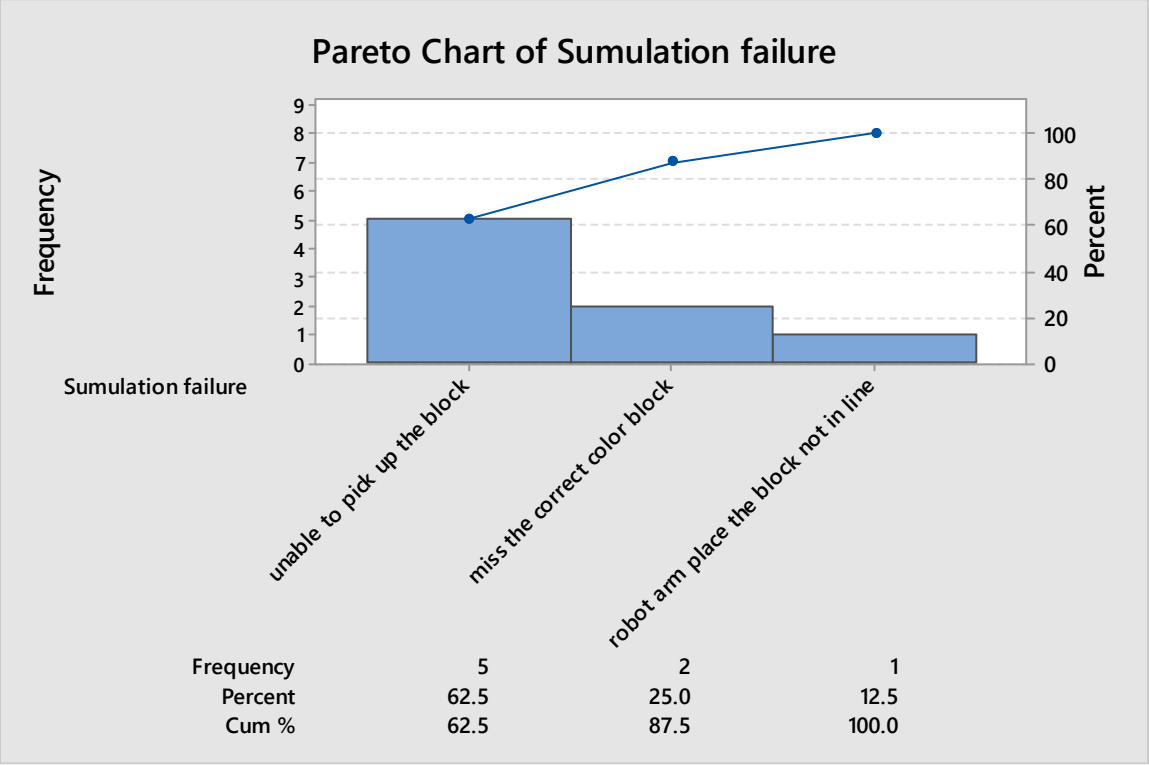

Figure 10. Simulation Failure

7. Future Improvements

The robot we built fulfills the goals of our pick and place challenge, however, there is still room for future improvements.

First, we want our robot to be able to “see” different colors. Being able to recognize colors means that we can put different colored blocks into different boxes, improving the practicability of the robot. This can be done using vision sensors, and mapping out the HSV values of different colors. Then, using forward and inverse kinematics, we can place different colored blocks into different bins.

Second, we want to be able to pick and place blocks while the conveyer belt is moving. Currently, to pick and place a correctly colored block, the conveyer belt halts, reducing the efficiency of the robot. If the robot can pick up blocks while the conveyer belt is moving, time can be saved. This requires more advanced motion planning, and a sensor, maybe the camera, that can trace the block and determine its velocity. We can also possibly change the speed of the conveyor belt, making it go slower while the picking is happening so that it’s easier for the robot to pick up the block accurately.

References

Lynch, K., & Park, F. C. (2019). *Modern Robotics: Mechanics, planning, and Control*. Cambridge University Press.

Home. Index page. (n.d.). Retrieved December 17, 2021, from http://www.forum.coppeliarobotics.com/

Ur5 robot parameters by Denavit-Hartenberg Method ... (n.d.). Retrieved December 17, 2021, from https://researchgate.net/figure/UR5-robot-parameters-by-Denavit-Hartenberg-method_fig2_347021253

Coppeliasim User Manual. (n.d.). Retrieved December 17, 2021, from https://www.coppeliarobotics.com/helpFiles/

ECE 470 Lecture Slides and Supplemental Notes

ECE 470 Lab Manual