ARM-AGGEDON

Team Members:
Shanley Mullen
Jesse Rosart-Brodnitz

MXET 400: Final Project

05/05/2020

Table of Contents

1. Roles & Team Collaboration

Jesse Rosart-Brodnitz was in charge of the CAD section as well as working on the GUI system he worked on the forward kinematics section. Meanwhile, Shanley Mullen was in charge of the matlab section, but also helped to design the initial system that was translated into CAD.

We met weekly for team update meetings with Dr. Lee throughout the semester as well as had weekly team meetings to work on the project. The work breakdown can be seen in Table 1 below.

TASK	Primary	Secondary
Initial Design	Shanley/Jesse	
CAD Gripper Design	Jesse	Shanley
Importing UR5	Shanley	Jesse
Forward Kinematics	Jesse	Shanley
Inverse Kinematics	Shanley	Jesse
Path Following	Shanley	Jesse
Forward Dynamics	Shanley	Jesse
GUI	Jesse	Shanley

Table 1. Task Breakdown

2. Introduction

We are team ARM-aggedon here to introduce our product the Hemmingway. Originally, the Hemmingway was created to act as a polydactyl prosthetic cat paw that will be an improvement from the typical prosthetic nubs seen on most cats. Also, we wanted to make a retractable cat-claw system based on a syringe based air system. This idea was conceived by looking into soft robotics and wanting to deviate from a standard approach. The research led to a pneumatic system powered by a linear actuator to control claws. In addition, a single motor was used to drive the finger joints apart. In Figure 1 below we can see our initial design of our system.

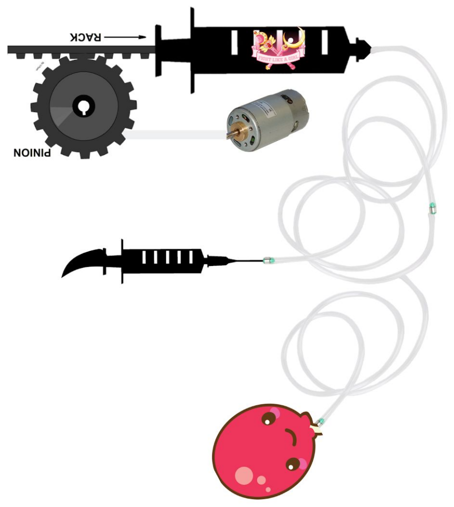


Figure 1

In our first concept we were going to use a rack and pinion system that was powered by a DC motor. This would then open and close our mega syringe that would then go to the smaller syringe to push the 3D printed claws in and out. A balloon would be used to open and close the thumb joint to grab the object while the other part of the paw would remain stationary. We made a lot of changes to this design that can be seen in the CAD portion of this document. Other things covered in this document include the matlab portion of this assignment that has been divided into three separate sections which includes simulink, forward dynamics with trajectory, and inverse kinematics.

3. CAD

The system is divided up into three sections, the paw, UR5, and pneumatic system.

The first is the creation of the actual paw system and the base mounting. This started with building finger joints and placing a motor mount inside. The next was making an adaptor the axle of the motor could connect to. This axle then connects to another finger joint. Each finger joint also contains a syringe sub assembly with a small claw mounted on the top of the plunger. The nozzle of each plunger is connected to a cable (not seen in CAD). The entire assembly is built on a base plate made to match and screw into the UR5 end effector plate. The paw assembly can be seen in Figure 2.

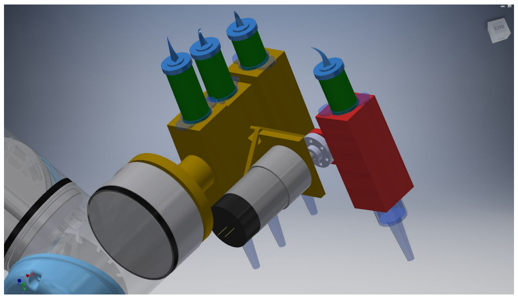


Figure 2

The second portion of the CAD assembly is the UR5. The model for the UR5 was found online as a solidworks file and the individual components were converted to Inventor files. The individual components were reassembled and constrained to create the UR5. This serves as the base for the mounting of the paw assembly and for the pneumatic system. This can be seen in Figure 3.

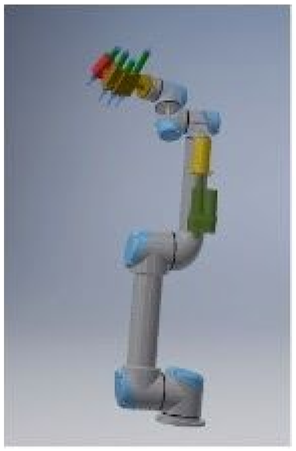


Figure 3

The third portion was the pneumatic system. The pneumatic system is composed of a linear actuator and syringe assembly. These are mounted on the side of the UR5. The tip of the syringe is connected to each of the tips of the claw syringes via a cable. When the linear actuator pulls the syringe plunger out, the claws will retract from the air pressure. When the linear actuator pushes the syringe in, the claws will shoot out because of the force of the air pressure. This can be seen in figure 4.

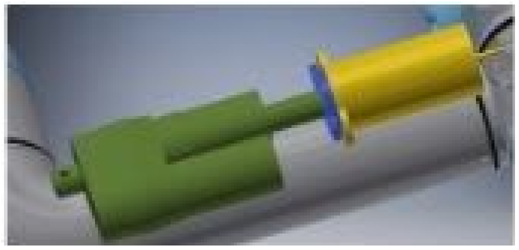
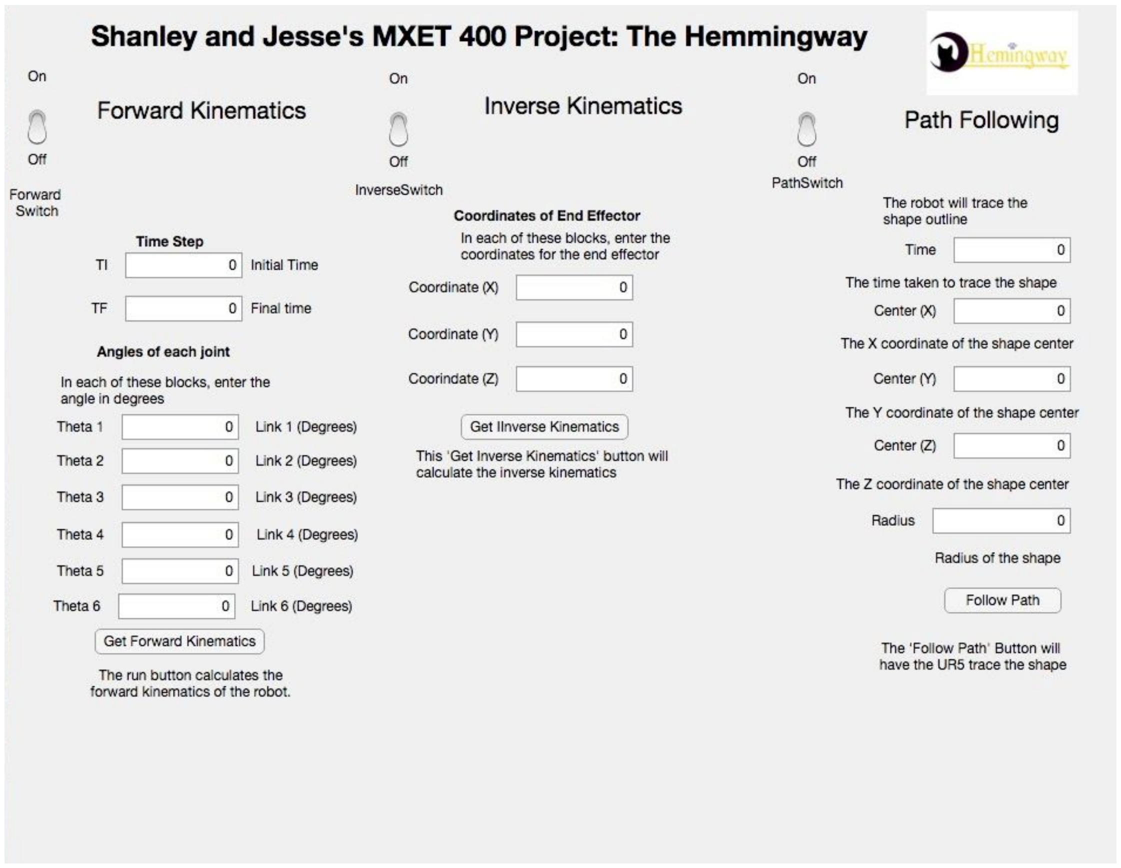


Figure 4

4. Matlab

In this portion of the document we will cover the matlab code. To get our robot model that everything is based on we used a ur5 opensource urdf file that is a RRR manipulator. For all the sections below the robot begins at the same initial robot configuration position of [0 0 0 0 0 0] and then moves to the new location. The GUI controlling all the sub programs can be seen below in Figure 5.



Shanley and Jesse's MXET 400 Project: The Hemmingway

Forward Kinematics

On / Off
Forward Switch

Time Step
TI 0 Initial Time
TF 0 Final time

Angles of each joint
In each of these blocks, enter the angle in degrees

Theta 1	0	Link 1 (Degrees)
Theta 2	0	Link 2 (Degrees)
Theta 3	0	Link 3 (Degrees)
Theta 4	0	Link 4 (Degrees)
Theta 5	0	Link 5 (Degrees)
Theta 6	0	Link 6 (Degrees)

Get Forward Kinematics
The run button calculates the forward kinematics of the robot.

Inverse Kinematics

On / Off
InverseSwitch

Coordinates of End Effector
In each of these blocks, enter the coordinates for the end effector

Coordinate (X) 0
Coordinate (Y) 0
Coorindate (Z) 0

Get IInverse Kinematics
This 'Get Inverse Kinematics' button will calculate the inverse kinematics

Path Following

On / Off
PathSwitch

The robot will trace the shape outline

Time 0
The time taken to trace the shape
Center (X) 0
The X coordinate of the shape center
Center (Y) 0
The Y coordinate of the shape center
Center (Z) 0
The Z coordinate of the shape center
Radius 0
Radius of the shape

Follow Path
The 'Follow Path' Button will have the UR5 trace the shape

Figure 5

a. Forward Kinematics

To begin, we imported the UR5 urdf file using sim in order to generate a usable simulink code. This code can be seen in Figure 6 below. The point of this was to make sure that we had the proper naming convention for the remainder of the matlab code to generate the joint angles for forward dynamics. Each link corresponded to a link in the GUI where the user could input a joint angle. This can be seen in figure 7.

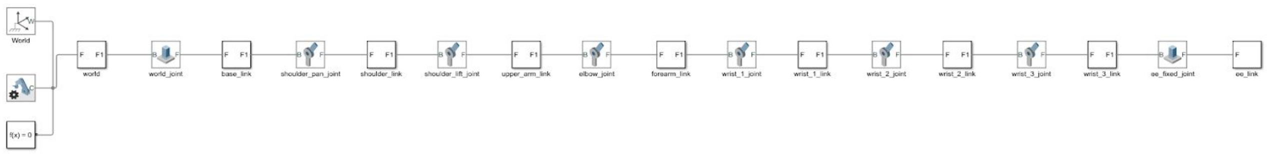
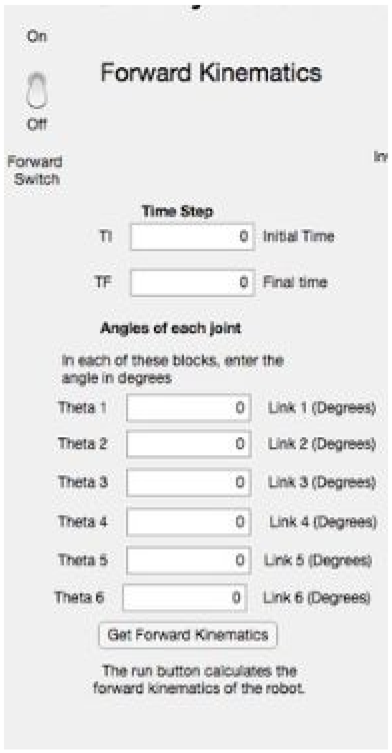


Figure 6


On

Forward Kinematics

Off

Forward Switch

Time Step

TI | 0 | Initial Time

TF | 0 | Final time

Angles of each joint

In each of these blocks, enter the angle in degrees

Theta 1 | 0 | Link 1 (Degrees)

Theta 2 | 0 | Link 2 (Degrees)

Theta 3 | 0 | Link 3 (Degrees)

Theta 4 | 0 | Link 4 (Degrees)

Theta 5 | 0 | Link 5 (Degrees)

Theta 6 | 0 | Link 6 (Degrees)

Get Forward Kinematics

The run button calculates the forward kinematics of the robot.

Figure 7

These values were then inputted into the necessary transformation functions in order to calculate how much the robotic manipulator link needed to move. This can be seen in Figure 8. Since the UR5 has fixed link lengths these were not changed.

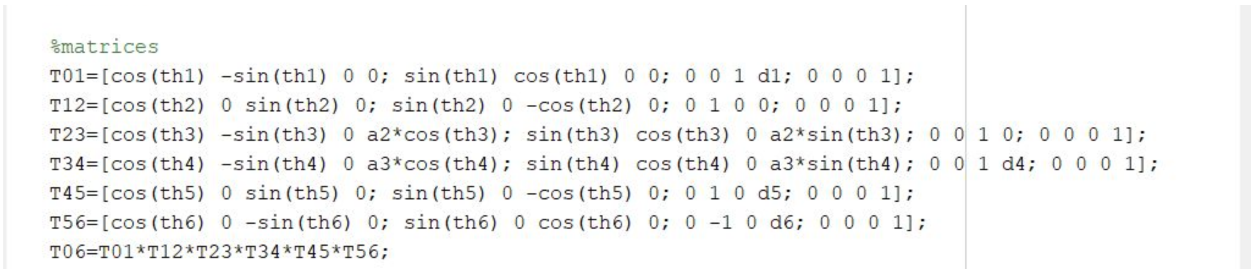
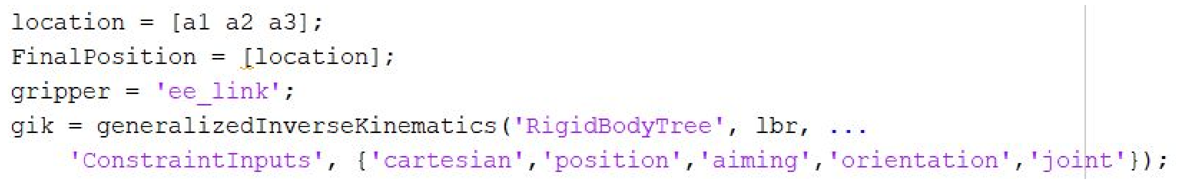

```
%matrices
T01=[cos(th1) -sin(th1) 0 0; sin(th1) cos(th1) 0 0; 0 0 1 d1; 0 0 0 1];
T12=[cos(th2) 0 sin(th2) 0; sin(th2) 0 -cos(th2) 0; 0 1 0 0; 0 0 0 1];
T23=[cos(th3) -sin(th3) 0 a2*cos(th3); sin(th3) cos(th3) 0 a2*sin(th3); 0 0 1 0; 0 0 0 1];
T34=[cos(th4) -sin(th4) 0 a3*cos(th4); sin(th4) cos(th4) 0 a3*sin(th4); 0 0 1 d4; 0 0 0 1];
T45=[cos(th5) 0 sin(th5) 0; sin(th5) 0 -cos(th5) 0; 0 1 0 d5; 0 0 0 1];
T56=[cos(th6) 0 -sin(th6) 0; sin(th6) 0 cos(th6) 0; 0 -1 0 d6; 0 0 0 1];
T06=T01*T12*T23*T34*T45*T56;
```

Figure 8

b. Inverse Kinematics

To do the inverse kinematics the user was asked to input an X position, Y position, and Z position of where the user wants to move the end effector too. We achieved this by setting the ee_link, our end effector, to move to the desired input. All of the inverse kinematics were calculated using the inverse kinematics function within matlab. This can be seen in Figure 9.



```
location = [a1 a2 a3];
FinalPosition = [location];
gripper = 'ee_link';
gik = generalizedInverseKinematics('RigidBodyTree', lbr, ...
    'ConstraintInputs', {'cartesian','position','aiming','orientation','joint'});
```

Figure 9

c. Path Following

We created a basic path following function based on a previous lab. All we did was set the ee_link to follow a trajectory generated by the user. The user can follow a simple shape over a specific time period. The user is allowed to input the x y and z coordinates for the center of the shape as well as set the radius and the time step. The shape of the system is determined by the radius and by the time step given. This configuration was chosen to demonstrate the robot following a trajectory.

d. Forward Dynamics

To do forward dynamics we utilized the forward dynamics function where the force exerted on the joints could be easily changed. There was no GUI for this section of the code. The force entered was then calculated using the external force function within matlab that was then passed into the forward dynamics function. The use would change the force values and the robot would move accordingly.

5. Conclusion

In conclusion we designed a robotic gripper using CAD and created forward dynamics, inverse dynamics, path trajectory, and forward dynamics within matlab. These components encompassed what we have learned in MXET 400 and provided us a way to incorporate the material into a simulated setting to see their value in the real-world. Overall, this was a good indicator of the course.