ENPM662 Project 2 Report

Markose Jacob
Maryland Applied Graduate Engineering
University of Maryland, College Park
College Park, Maryland, USA
markj11@umd.edu

Trevian Jenkins
Maryland Applied Graduate Engineering
University of Maryland, College Park
College Park, Maryland, USA
trj0011@umd.edu

***Abstract*—Robot modeling is one of the core aspects of the study and development of robotic systems. Creating a sound model of the kinematics and dynamics of a robot is important in understanding its capabilities and constraints. This understanding is important for other robotics-related applications such as controls and motion planning. One major application of robotics is the automation of retrieving and transporting objects. We have referenced a real-world robot—the Husky unmanned ground vehicle UGV—to create a similar model that accomplishes a similar task. Specifically, our robot model—dubbed *NavBot*—transports objects using four-wheel drive, an attached manipulator, and a carrying tray. This document provides a description of the kinematic model of NavBot, its capabilities, and its implementation.**

I. Introduction

As the advancement of technology has demanded an increasing amount of automation, the opportunities to apply robotic tools have grown to meet this demand. Some of the challenges people will face, such as hazardous conditions and high levels of production, will increasingly fall under the domain of robots. We have decided to design a mobile robot named NavBot that can navigate multiple environments, with the additional ability to pick and place objects using a gripper and an on-board tray. Our theoretical robot model would also contain various sensors, a must-have for both exploring new environments and localization within known environments. In addition to sensors, a realistic version of NavBot would also have a certain degree of autonomy, navigating through both known and unknown territory using a variety of navigation algorithms. We implemented the simulation of NavBot using the Robot Operating System, or ROS. We will cover the capabilities that we were able to implement in this simulation, such as the forward kinematics, inverse kinematics, structure, and motion planning capabilities of NavBot.

II. Motivation

The primary purpose of this project is to develop a structural and kinematic model of NavBot. We chose the design for NavBot because of a particular interest in modeling mobile robots. The implementation of this project will also allow the opportunity for us to create a semi-autonomous robot with a realistic design that can effectively compute its motion and use the result for navigation.

Our reference will be an unmanned ground vehicle (UGV) known as Husky, which was created by Clearpath Robotics. The Husky, pictured in Fig. 1, contains various sensors including LIDAR, GPS, and stereo cameras. We intend to create a model of a similar robot for our project. The primary differences will be that the rotary joint at the base of the arm will rotate around an axis normal to the plane of the platform. The base of the arm will be at the exact center of the platform as well, although this detail may change later if necessary.

III. About the Robot

A. Description

NavBot is a mobile robot that can be used to automate the process of moving objects around an environment using a manipulator. The chassis of the robot is approximately the shape of a rectangular prism, at $0.83 \times 0.415 \times 0.22$ meters. A tray is suspended above

Figure 1: Husky UGV

the chassis of NavBot up to 0.13 meters high. Extending from the chassis through the tray is a robotic arm with a base link length of 0.23 meters, and three subsequent link lengths of 0.4, 0.4, and 0.345 meters, respectively, when measured from the center of the connection between each link. The end-effector extends from the last link by 0.118 meters (the last link plus end effector), providing the entire arm a fully-stretched length of 1.263 meters. All the joints are revolute joints and by using 6 six joints the arm of NavBot is able to reach objects in can pose and orientation.

NavBot uses a front-wheel-drive, with a common axle between the rear two wheels. The front two wheels can turn individually in order for the robot to steer. The wheels, with a radius of 0.8 meters, will be located below the robot, suspending the chassis above the ground by 0.6 meters. The tray will be used to place objects for temporary storage as it navigates the environment picking up objects. The dimensions of the tray is $0.36 \times 0.595 \times 0.13$ meters and objects of maximum size 0.36×0.245 meters can be placed on the tray. Forward/reverse drive, along with steering, will allow NavBot two degrees of freedom (DoF) for locomotion. The first link of the manipulator will extend vertically upward from the center of the chassis/tray, and it will be able to adjust its yaw. In total, the manipulator has six degrees of freedom:

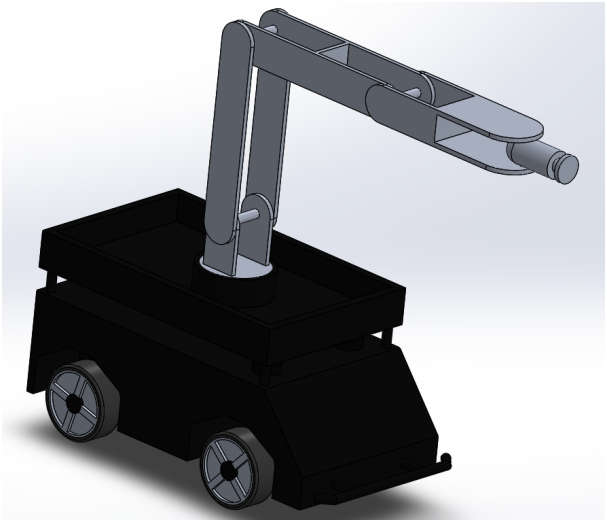
1) 2 rotary joint to adjust the yaw of the 1st and 5th link
2) 3 rotary joint to adjust the pitch of the 2nd 3rd and 4 4th link
3) 1 rotary joint to adjust the roll of the end effector which is the final link

The six rotary joints within the arm, and the two degrees of freedom provided by the four wheels offer a total of eight degrees of freedom. A picture of the model is included in Fig. 2a, with frames of reference labeled in Fig. 2b.

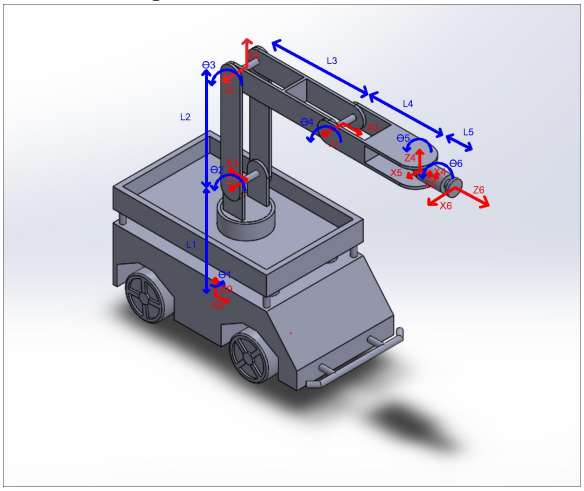
For a real, physical version of this robot, NavBot would utilize a proximity sensor to ensure that it does not collide with obstacles. A camera would also be equipped for the purpose of visualizing the environment. It would also utilize an odometer for location, or possibly a global positioning system (GPS).

In this project, NavBot uses an odometer to retrieve positional data without the need for more advanced perceptional techniques beyond the scope of this project. We wanted to also use a scale to determine if NavBot is carrying any objects.

Table I shows the Denavit-Hartenberg (DH) parameters for the frames of reference of NavBotin



(a) Proposed model of NavBot, side view



(b) Labeled frames of reference of NavBot

Fig. 2b, where the link lengths are in meters,

$$\begin{aligned} L_1 &= 0.48 \\ L_2 &= 0.40 \\ L_3 &= 0.40 \\ L_4 &= 0.345 \\ L_5 &= 0.118, \end{aligned} \tag{1}$$

and θ_n^* corresponds to rotation of each joint along the Z_{n-1} axis.

B. Materials

Physically, the chassis and axles of NavBot would be made of carbon fiber, as carbon fiber has high stiffness, high tensile strength, high chemical resistance and high temperature tolerance—all of which are suitable for a

	α	θ	a	d
1	-90°	$\theta_1^* + 180^\circ$	0	L_1
2	0°	$\theta_2^* - 90^\circ$	L_2	0
3	0°	$\theta_3^* - 90^\circ$	L_3	0
4	-90°	θ_4^*	L_4	0
5	-90°	$\theta_5^* - 90^\circ$	0	0
6	0°	θ_6^*	0	L_5

Table I: DH parameters

robot working in harsh environments. The arm, end-effector, and wheels would be made from aluminum, while the tires would be made of rubber. For the simulated version of NavBot, we were able to maintain all of these properties via SolidWorks, which automatically assigns the mass and inertial properties of the links based on their materials and geometries.

C. Appropriateness for the task

Overall, the configuration of NavBot is well-suited to the tasks at hand. NavBot will be easily controllable because of the ability to steer the front wheels individually. The requirement for six links arises from the need to be able to reach anywhere on the tray for placing an object, so NavBot will have plenty of dexterity. The use of the vacuum gripper will be sufficient for picking and placing objects around the environment.

D. Model Assumptions

- The robot in this model is a rigid body.
- We assume that there is negligible air resistance, and that the ground has sufficient friction against which to roll the wheels.
- The wheels only rotate when propelled by actuators.
- NavBot always moves along a flat surface.
- We assume that the chassis is sufficiently heavy to balance the weight of the arm. If we find that this assumption makes the robot unstable in Gazebo, we will manually adjust the mass of the chassis within the URDF in Gazebo.
- One important design consideration we had in this phase of the project was that the end-effector could swivel around its x-axis and rotate about its z-axis. This dual rotation ability will allow the robot to grip objects that do not necessarily have a top surface that is parallel with the ground, but introduces another degree of freedom. Without the ability to rotate about the x-axis, we simplify our model, but we would have to further assume that objects must have a flat top surface parallel to the ground in order for NavBot to be able to grip them.
- The gripper is a vacuum gripper that can be used to automatically attach itself to an object.
- There is no coupling between any of the joints.
- The motor is powerful enough to propel the robot.
- There is enough friction to prevent the joints from rotating when they are not being actuated.

IV. Project Scope

We have primarily focused on modeling the mechanical structure and kinematics of the robot, which is also the focus of this report. Below is the list of tasks we managed to complete within the time frame of this project:

- Conceived an efficient design for a robot which can work in shop-floors to pick and place objects which has any orientation and pose.
- Modeled NavBot on SolidWorks. With the help of SolidWorks we were able to generate the 3D model which was later used for simulation to visually see the forward and inverse kinematics we calculated.
- Defined the joints and axes in SolidWorks and then generated the URDF file which was spawn NavBot in Gazebo.
- Modified the ROS launch file to incorporate our script for forward and inverse kinematics.
- Created a model of a shop floor on Gazebo to spawn NavBot in and do useful work.
- Computed forward kinematics on MATLAB using the DH table, and verified our result with the output we got on Gazebo for different joint values that was passed into our script for forward kinematics as arguments.
- Numerically computed inverse kinematics. We were ultimately able to parameterize the end effector pose and calculate the corresponding angle of each revolute joint needed to achieve the desired end effector location accordingly.
- Demonstrated that we were able to do useful work in the shop floor, like placing the object on the tray of NavBot and directing the robot to the goal location.

Getting the vacuum gripper to work was one more thing we wanted to do but wasn't able to achieve.

A. Forward Kinematics (FK)

Forward Kinematics is calculated only for the manipulator. The wheels are not considered for FK. Calculation can be split into 3 steps.

- Frame assignment
- Creating DH table

- Calculating transformation matrix

1) Frame Assignment: NavBot's manipulator has six revolute joints and hence we have seven frames. The base frame is on the chassis of NavBot and the second frame is located at the start of the second link. Our frame assignment can be seen in Fig. 2b. By following a certain set of the rules for frame assignment, we were able to correctly assign frames to all the joints.

- Each X_n axis is always perpendicular to, and intersects with Z_{n-1} axis
- Revolute joints always rotate around the Z_n axis, while prismatic joints move along the Z_n axis.

2) DH Table: The DH table for NavBot can be seen in table I. The table was constructed by following certain rules:

- θ_n is the rotation alone Z_{n-1} to make X_{n-1} point in the same direction as X_n
- α is the rotation alone X_n to make Z_{n-1} point in the same direction as Z_n
- a is the displacement alone X_n
- d is the displacement alone Z_{n-1}

3) Transformation Matrix: The final step is the create the final transformation matrix which will give us the pose and orientation of the end effector. We first calculated the individual transformation matrix for each of the joints by plugging in the each row of the DH table in the transformation matrix below.

$$\begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By plugging in the first row of the DH table in the above matrix we get the following transformation from base frame to link 1.

$$T_1^0 = \begin{bmatrix} -\cos\theta_1 & 0 & \sin\theta_1 & 0 \\ -\sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & -1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transform from link 1 to link 2 is calculated by plugging the second row of the DH table in the transformation matrix.

$$T_2^1 = \begin{bmatrix} \sin\theta_2 & \cos\theta_2 & 0 & l_2\sin\theta_2 \\ -\cos\theta_2 & \sin\theta_2 & 0 & -l_2\cos\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We repeat these steps until frame T_6^5. We get the final transformation matrix by multiplying all the transformation matrix together:

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5$$

Below is the final transformation matrix we got for NavBot in its home configuration. (refer 2a for this configuration)

$$T_6^0 = \begin{bmatrix} 0 & 0 & 1 & 0.8630 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.8800 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The first three elements of the last column of T_6^0 gives the end effector pose in X, Y and Z, coordinates. The first 3 rows and columns gives the rotation matrix which can be converted to roll pitch and yaw using the formulae below:

$$\begin{aligned} \text{roll} &= \arctan\left(\frac{r_{21}}{r_{11}}\right) \\ \text{pitch} &= \arctan\left(\frac{-r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right) \\ \text{yaw} &= \arctan\left(\frac{r_{32}}{r_{33}}\right) \end{aligned} \tag{2}$$

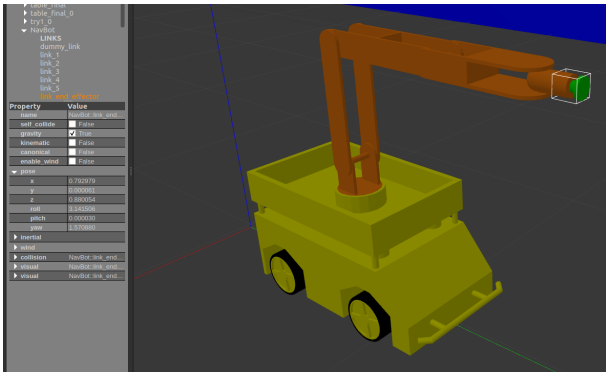
The orientation for NavBot's end effector in home configuration is

$$\begin{aligned} \text{roll} &= 90^\circ \\ \text{pitch} &= 0^\circ \\ \text{yaw} &= 90^\circ \end{aligned} \tag{3}$$

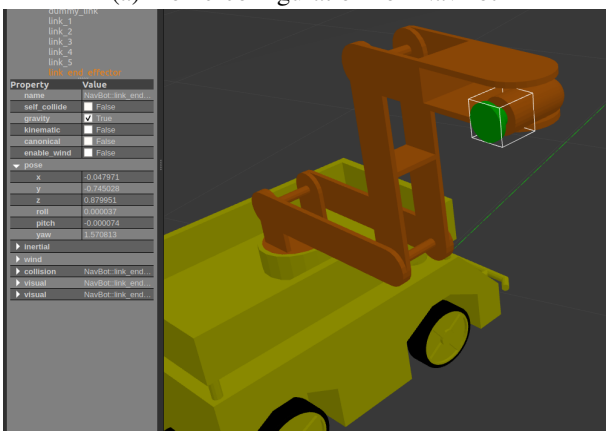
All the calculations were done in MATLAB. We then spawned NavBot in Gazebo and changed the joint values to zero degrees and observed the pose and orientation we got in Gazebo matched with what we got in MATLAB. Refer 3a to see the pose and orientation of the end effector in Gazebo.

4) Validation: To validate that our calculations are correct we gave random joint values to the different joints to come up with a random pose which can be see in 3b. The joint values what we used were,

$$\begin{aligned} \theta_1 &= 90^\circ \\ \theta_2 &= 90^\circ \\ \theta_3 &= 0^\circ \\ \theta_4 &= 90^\circ \\ \theta_5 &= -90^\circ \\ \theta_6 &= 0^\circ \end{aligned} \tag{4}$$



(a) Home configuration for NavBot



(b) random configuration for NavBot

and the pose and orientation we got on Gazebo was

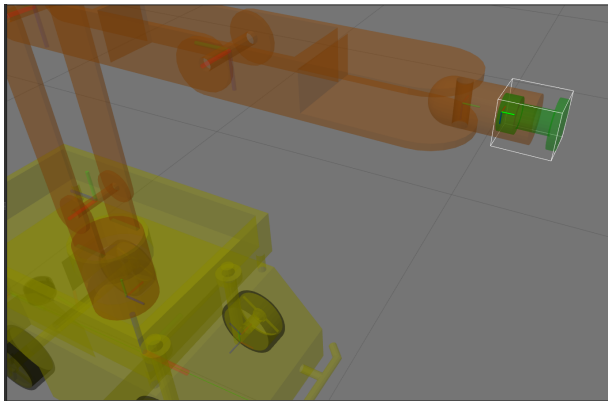
$$\begin{aligned} x &= -0.1179 \text{ m} \\ y &= -0.7450 \text{ m} \\ z &= 0.8799 \text{ m} \\ \text{roll} &= 90^\circ \\ \text{pitch} &= 0^\circ \\ \text{yaw} &= -90^\circ \end{aligned} \tag{5}$$

We observed similar results when we passed in these joint values to out MATLAB script refer Fig. 4a.

Note that a value of 0.07 meters (lenght of the end effector) needs to be added to the pose that is seen in the x axis in gazebo, this is because the last frame in gazebo is at the start and of the end effector where as we have taken the last frame at the end of the end effector. This can be seen in 4b.

```
roll =

pi/2


pitch =

0


yaw =

-pi/2

>> [ 0,   0, -1,  -59/500]
[ 1,   0,  0, -149/200]
[ 0, -1,   0,    22/25]
[ 0,   0,  0,        1]

ans =

         0         0   -1.0000   -0.1180


ans =

    1.0000         0         0   -0.7450


ans =

         0   -1.0000         0    0.8800


ans =

     0     0     0     1
```

(a) MATLAB output for random configuration of NavBot



(b) Frames in Gazebo

B. Inverse Kinematics

The inverse kinematics of this model were calculated using an algorithm called *FABRIK*, which stands fro Forward- And Backward-Reaching Inverse Kinematics. FABRIK is a heuristic approach to calculating inverse kinematics that has proven superior to other popular inverse kinematics solvers in several situations [2]. For the purpose of our implementation of NavBot, FAB-

RIK proved adequate in moving the end-effector to a desired location by calculating the intermediate manipulator joints. The process for doing so is explained here subsequently.

1) Implementation: The algorithm is generally performed as follows:

1) In the first iteration $n = 0$, the joint locations $(x_{i,n}, y_{i,n})$, in Cartesian coordinates, are calculated from the initial joint angles θ_i, $i \in [1, M]$, (in reference to the positive y-axis) and link lengths d_i as follows:

$$x_{i,0} = x_{i-1,0} + d_i \sin\left(\sum_{k=0}^{M} \theta_k\right)$$

$$y_{i,0} = y_{i-1,0} + d_i \cos\left(\sum_{k=0}^{M} \theta_k\right)$$

We assume the initial joint location $(x_{1,0}, y_{1,0})$ to be $(0, 0)$.

2) In each iteration n, the end-effector location $(x_{i,n}, y_{i,n})$ is translated to a target end-effector location (t_x, t_y).
3) Starting from the penultimate joint location $(x_{M-1,n}, y_{M-1,n})$, each joint is placed along a line from $(x_{i,n}, y_{i,n})$ to $(x_{i-1,n-1}, y_{i-1,n-1})$ at a distance d_i from joint i.
4) Step (3) is repeated up to the initial joint location $(x_{1,n}, y_{1,n})$.
5) A first threshold $\varepsilon_{n,a}$ is calculated for this iteration as follows:

$$\varepsilon_{n,a} = \sqrt{x_{1,n}^2 + y_{1,n}^2}$$

6) The initial joint location is moved back to $(x_{1,0}, y_{1,0})$, and step (3) is repeated, but backward to the end-effector location.
7) A second threshold $\varepsilon_{n,b}$ is calculated for the backwards loop as follows:

$$\varepsilon_{n,b} = \sqrt{x_{M,n}^2 + y_{M,n}^2}$$

8) Steps (2) through (7) are repeated until both thresholds $\varepsilon_{a,b}$ and $\varepsilon_{n,b}$ are less than a tolerance μ, or until the number of iteration n exceeds a limit N. This condition is stated in the following logical expression:

$$(\varepsilon_{n,b} < \mu \textbf{ and } \varepsilon_{n,b} < \mu) \textbf{ or } (n \geq N)$$

9) Trigonometry is used to calculate the joint angles from the joint locations of the final iteration.

2) Simplifications: For NavBot, the incorporation of the FABRIC algorithm was not seamless. There were minor adjustments to be made that simplified calculations. To elaborate, the base joint θ_1 and the end effector joint θ_5 were the only two joints which rotated about the z-axis, so we could find a feasible solution by calculating the base joint angle θ_1 as

$$\theta_1 = \arctan\left(\frac{t_z}{\sqrt{t_x^2 + t_y^2}}\right),$$

while keeping $\theta_5 = 0°$. Furthermore, we assume a base location of $x = 0, y = 0$, and translate the coordinates to the base location of the robot at the conclusion of the algorithm. This way, we only had to perform the FABRIK algorithm with the joints θ_2, θ_3, and θ_4. See Fig. 5a and Fig. 5b for an illustration of FABRIK we calculated for NavBot.

We plan to demonstrate our model using two different videos. In the first video, we will demonstrate the exact calculations discussed in the previous paragraph. For the second video, we will demonstrate NavBot's capability to pick up an object and place it into a designated area.

V. Goals

The goal of this project was to come up with a robot which can be used in the shop floor to pick and place objects.

A. Immediate Goals Achieved

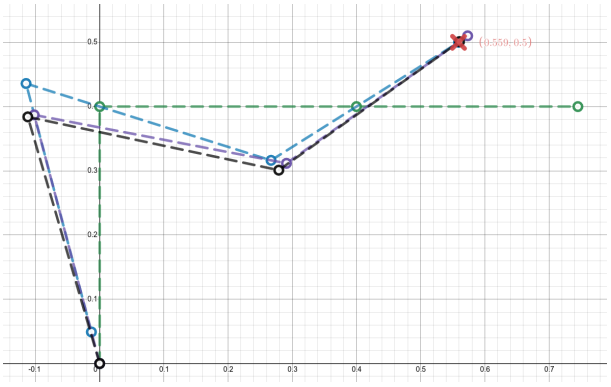
- We were able to get the NavBot to place the cube on the tray of NavBot using inverse kinematics.
- We were able to study the forward and inverse kinematics and implement the same on NavBot

B. Ambitious Goals Achieved

- We were able to successfully implement path planning using A* algorithm and hence NavBot can now move to any location on the map that has a valid path while avoiding obstacles.
- Use teleop to move the robot
- Come up with a user friendly code which can take joint values/ end effector pose to calculate FK and IK and to move the arm accordingly.

C. Goals Not Achieved

- We could not get the vaccum gripper to work and hence we had to push the object into the tray of NavBot



(a) Visual of FABRIK iterations, with the target location $t_x, t_y = 0.559, 0.5000$

Initial position		
Joint	x	y
1	0	0
2	0	0.4
3	0.4	0.4
4	0.745	0.4

First backward iteration		
Joint	x	y
1	-0.0129	0.0491
2	-0.1147	0.4359
3	0.2699	0.3163
4	0.559	0.5

First forward iteration		
Joint	x	y
1	0	0
2	-0.102	0.387
3	0.291	0.3117
4	0.573	0.5101

Second backward iteration		
Joint	x	y
1	0	0
2	-0.112	0.384
3	0.279	0.301
4	0.561	0.5011

(b) Table for iterative FABRIK calculations

VI. Challenges

We faced several challenges in this project but two of the most challenging tasks were mentioned here.

1) FABRIK implementation: We realised calculating equations for each joint of a robot with 6 DoF is very complicated using the method we know. While searching for a solution we came across the FABRIK method which we found much easier to do.

2) Gathering odometry information: Gathering the odometry information was another tricky part which took up our time.

VII. Future Work

Overall we are very pleased with what we were able to achieve in this project but there is always room for improvement. Some of the things we would like to do are

- Figure out why we couldn't get the gripper to work. By getting the gripper to work we would be able to complete the project and have virtual robot is fully capable of picking and placing multiple objects at the same time.
- Fixing our package to make sure the robot stops sliding in Gazebo. We are not sure why this happens and we like to figure this out and prevent the robot from sliding.
- Adding the capability of using multiple tools.

References

[1] Husky UGV - Outdoor Field Research Robot by Clearpath: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

[2] Aristidou, Andreas, and Joan Lasenby. "FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem." Graphical Models, 1 Sept. 2011, dl.acm.org/doi/10.1016/j.gmod.2011.05.003