# ENPM 662 Introduction to Robot Modeling

## PROJECT REPORT


# ROBOT DRESSING SUPPORT SYSTEM(RDSS)

**PRASANNA THIRUKUDANTHAI RAGHAVAN (118287546)**

**SHELVIN PAULY (118426556)**

**Acknowledgement**

We express our heartfelt gratitude to respected Dr. Reza Monafredi for providing us with the opportunity to work on the final project. We also thank him for his support, guidance and the knowledge imparted through lectures which enabled us to come with this idea and further work on it.

We extend our deep gratitude to Teaching Assistants Mr. Ajinkya Parwekar and Mr. Karan Sutradhar, who helped us through all the problems and errors we faced while working on the project. And also, their valuable suggestions to enhance the quality of the project. Without their support this project would not have taken its present shape.

**Prasanna Raghavan**

**Shelvin Pauly**

**Table of Contents**

## Introduction

This project aims to augment the caregiving industry by designing and simulating a support system which helps the dependents in dressing. As per CDC, 13.7 % of the population in the US has mobility disability and this number is only going to double by 2050, as WHO cites that the world's population of people over 60 years is anticipated to double by 2050. We also found out that dressing is the most burdensome activity and has the least implementation of assistive technology. Considering the current pandemic situation where social distancing is the norm, this has only worsened. The chances of contamination from the apparel of high-risk healthcare workers are pretty high. Hence, we designed a robotic system that will address these issues.

The robot system consists of a WAM arm (Barrett technologies) with a 282 Hand (end effector). The arm has 7 degrees of freedom to provide help to humans in dressing. We simulated the system in an empty world with a ball of unit length. The ball emulates the characteristics of a cap. Our model was able to grasp the ball properly and move it across the environment. There can be multiple dressing scenarios which can be emulated.

The report is organized in a way that it goes over every aspect of the project in a chronological order. In the first section, we discuss the applications of the support system, which is then closely followed by the description of the robot which mentions the type of robot and specifications of the robot like the degrees of freedom and dimensions. The third section mentions the key factors in the CAD modelling steps and the procedures involved in it. In forward kinematics, we have created the Denavit Hartenburg table using the Spong convention and then finally calculated the final transformation matrix. The inverse kinematics includes the calculation of the Jacobian and Jacobian inverse matrices, which is used to find joint velocities and given an initial configuration. This section is closely followed by the validation of both forward and inverse kinematics. Then, we refer to the workspace of the robot system which is augmented with screenshots from the gazebo visualization snapshots. The assumptions taken throughout the study are stated explicitly in the following section. In the tenth section, we talk about the control method used in the system. We follow through with the actual gazebo simulation and the observations made from it. We also include the problems faced by us throughout the project and the learning outcomes from it. The report ends with the conclusion of the report and the future works. The references used throughout the project work are cited at the end of the document.

## Application

RDSS is a capable system designed to help humans with dressing. As mentioned in the previous section, t certain aspects of dressing still remain aa a burden or something that needs to be solved when considering specific scenarios such as the caregiving industry or healthcare industry. This project aims to solve those problems.

The application of the robot system can be consolidated as follows:

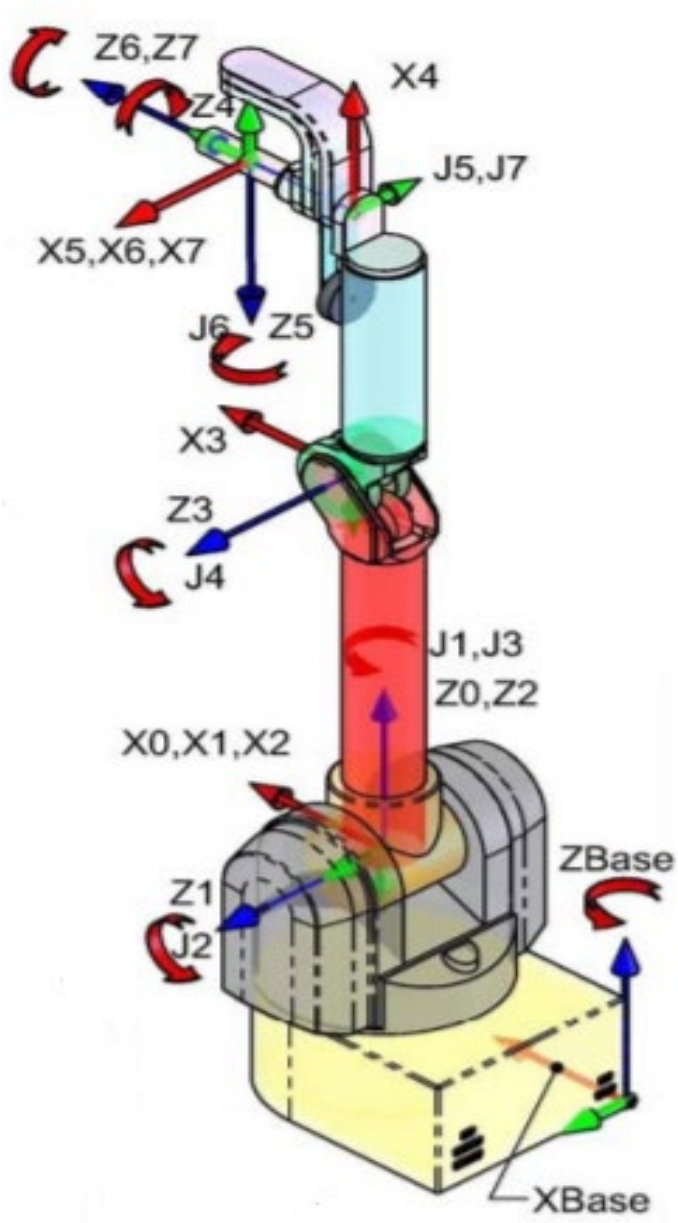
- It augments the dressing application in the caregiving industry, which is considered as the most burdensome in the industry. This, in turn, also provides the user with a sense of independence as they do not have to rely on their carer for support.
- It also can be used to reduce contamination from high-risk health care workers. It can effectively reduce the transmission of pathogens from the clothing (PPE, masks, etc.) of these health care workers.

The field of robotics is at its peak and there are numerous possibilities in every application, especially when we all are going facing these tough times(pandemic), we believe we should try and augment our health sector with the advent of technology.

The RDSS can be implemented autonomously as well as remotely, which also does not negate the concept of social distancing. During the pandemic, the caregiving industry had to close its doors to new patients because of the high risk of contamination. This support system can be implemented to all types of dressing applications like vests, jackets, etc.

**Robot Description**

The RDSS consists of a WAM arm (robotic manipulator) commercially sold by Barrett Technologies. It consists of a 7 DOF arm with a 282 Hand, 3 finger end effector.



The following table lists the robot specifications:

| | |
|---|---|
| Degree of freedom | 7-Degree of freedom arm with a 3-finger end effector.<br>Each finger has 3-Degrees of Freedom |
| Mass | 28.5 kg |
| Dimensions | 340 mm x 348 mm x 850 mm (fully extended) |
| Number of Links | 18 Links (including the base) |
| Number of Joints | 16 Rotary Joints and 1 Fixed Joint |
| End effector velocity | 2m/s |
| Payload | 4kgs |
| Repeatability | 2000$\mu$m |
| Maximum Joint Angular Displacement | 360 degrees |

The benefits of using a WAM arm are: -

- Lightweight
- Compact design
- High average joint mobility
- Back drivable
- Long and slender links
- Kinematic Redundancy which increases the dexterity of the system
- Human like grasping features with the 282 Hand

## CAD MODEL

The assembly consists of 14 parts which are used multiple times to create 18 links joined by 17 joints. Link 5 is a fixed joint and all the other joints starting from the base till the end effector contribute towards 7 degrees of freedom of the system. The system also has a human-like 3 finger end effector. Each of these fingers has 3 degrees of freedom. All the joints are revolute joints. Initially, as we were facing issues with revolute joints while exporting the package as urdf, we switched to continuous joints. Then, we explicitly changed the joist to revolute joist and the error was solved. The appropriate limits were specified to avoid joint locking. The assembly process took a while as the regular mates were not enough for this assembly. We had to explore advanced mates such as profile centers to achieve the correct functionality for the system. Majorly used mates in this assembly as co incident, concentric and profile center. The base link was exported with a reference coordinate system which was aligned according to the coordinate system in Gazebo.


Snapshot from Solidworks assembly file

**Forward Kinematics**

We know from the above that the axes have been assigned in a way that the z-axis is aligned along the direction of motion and the x-axis is towards the front of the robot. This would allow us to easily extract the dh parameters of the manipulator through either the Spong or the Craig convention. The naming of the axes starts from the base and has 8 axes moving upward. The base axis is called the Z base and the end effector axis is called the Z Tool axis.

As we can see some of the joints have motion along the same axes and hence contribute to the redundancy to the degree of freedom. The manipulator despite having multiple degree of freedom we compress multiple joints at one point. The dh of the bot is taken through the generic DH conventions(Spong or Craig) and from the above image we can infer the following:

**DH Table**:

| i | ai | alphai | di | thetai |
|---|---|---|---|---|
| 1 | 0 | -pi/2 | 0 | theta1 |
| 2 | 0 | pi/2 | 0 | theta2 |
| 3 | 0.045 | -pi/2 | 0.55 | theta3 |
| 4 | −0.045 | pi/2 | 0 | theta4 |
| 5 | 0 | -pi/2 | 0.3 | theta5 |
| 6 | 0 | pi/2 | 0 | theta6 |
| 7 | 0 | 0 | 0 | theta7 |
| 8 | 0 | 0 | 0.06 | theta8 |
| E | 0 | 0 | 0.15 | 0 |

Now the above table shows that the 8 joints have the above base configuration as shown on the above table. Then the values are substituted into transformation matrices and multiplied to produce the following matrix which is the transformation matrix from the base to the end effector.

The DH table tells us that most of the joints have a different axes than the ones either preceding or the next joints and yet still have their axes in the direction similar to other joints.

**Final Transformation Matrix:**

$$\begin{bmatrix} -0.433 & -0.75 & -0.5 & -0.213 \\ -0.866 & 0.5 & 0 & 0.286 \\ 0.25 & 0.433 & -0.866 & 0.204 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

The Transformation Matrix gives us the pose of the robot in home position.

**Inverse Kinematics:**

Using the 2nd method by differentiating each column separately similar to the assignments done in class we are able to derive a Jacobian for our manipulator. Here there are 8 columns for each of the parameters from the dh we obtained earlier.

**Jacobian Matrix:**

$$\begin{bmatrix} -0.286 & 0.204 & -0.248 & 0.0689 & 0 & -0.0779 & 0 & 0 \\ -0.213 & 0 & -0.286 & -0.339 & 0 & -0.156 & 0 & 0 \\ 0 & 0.213 & 0.143 & -0.559 & 0 & 0.045 & 0 & 0 \\ 0 & 0.5 & -0.612 & -0.5 & -0.75 & -0.5 & -0.5 & -0.5 \\ 1.0 & 0 & -0.707 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.866 & 0.354 & -0.866 & 0.433 & -0.866 & -0.866 & -0.866 \end{bmatrix}$$

The Jacobian is a 8x6 matrix as we have 8 individual joints and we get a column for each of those joints.

**Jacobian Inverse Matrix:**

As for the inverse, the Jacobian is not a square matrix, so we proceed to take a pseudo-inverse of our Jacobian matrix to get the inverse for our matrix. Then we obtain the following inverse matrix.

$$\begin{bmatrix} -0.9 & -0.885 & 0.3 & 0.272 & 0.516 & -0.0615 \\ 2.67 & -2.27 & 1.58 & 0.136 & 0.243 & 0.0185 \\ -0.807 & -0.794 & 0.269 & -0.205 & -0.434 & 0.203 \\ 0.874 & -1.17 & -1.01 & -0.000813 & -0.00946 & 0.0293 \\ 0.659 & 0.648 & -0.219 & -0.834 & 0.355 & 0.411 \\ 0.799 & -1.21 & 1.3 & 0.0061 & 0.111 & -0.352 \\ 0.496 & 0.0508 & 0.649 & -0.186 & 0.0705 & -0.262 \\ 0.496 & 0.0508 & 0.649 & -0.186 & 0.0705 & -0.262 \end{bmatrix}$$

The Jacobian inverse due to not being a square matrix needs to be accommodated to get a proper matrix.

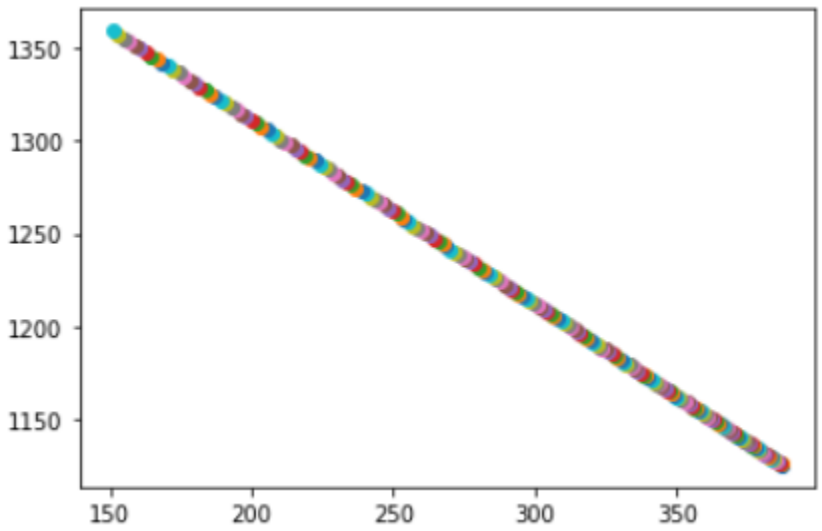
**Forward Kinematics Validation:**

Since we had used velocity controllers due to some problem with effort controllers in the package and then to reach these positions the limits to the revolute joints were given what were the same as the theta inputs to the Transformation matrix. This would mean that the joints would essentially be in the same position as they would be had we used effort controllers and with this we had tried multiple different configurations for the manipulator and then geometrically checked the end effector positions to see if they matched.

The geometric location of the end effector aligned with the coordinates obtained in the Transformation matrix for different orientations given by respective values of theta angles. We were then able to confirm that the Transformation Matrix obtained by the dh table gives us the correct coordinates to the assembled model.

**Inverse kinematics validation:**

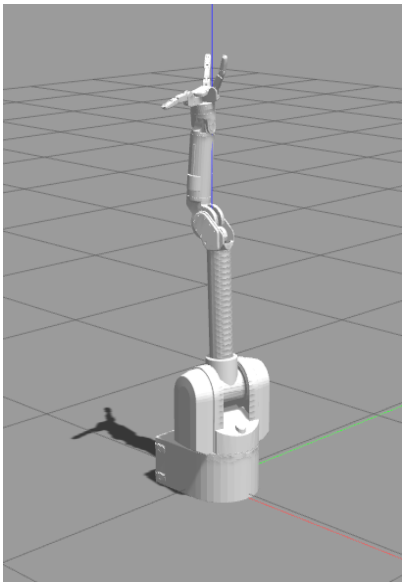
Now for making the calculation easier we take 2 redundant joint movements to be fixed and make the Jacobian invertible during calculation.

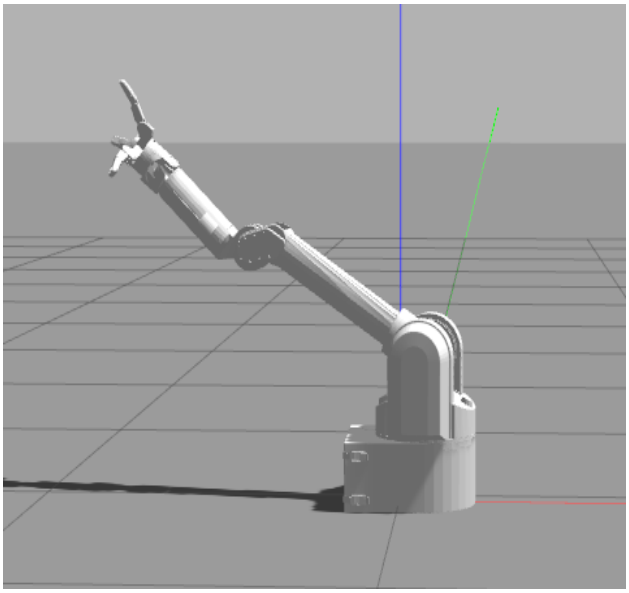


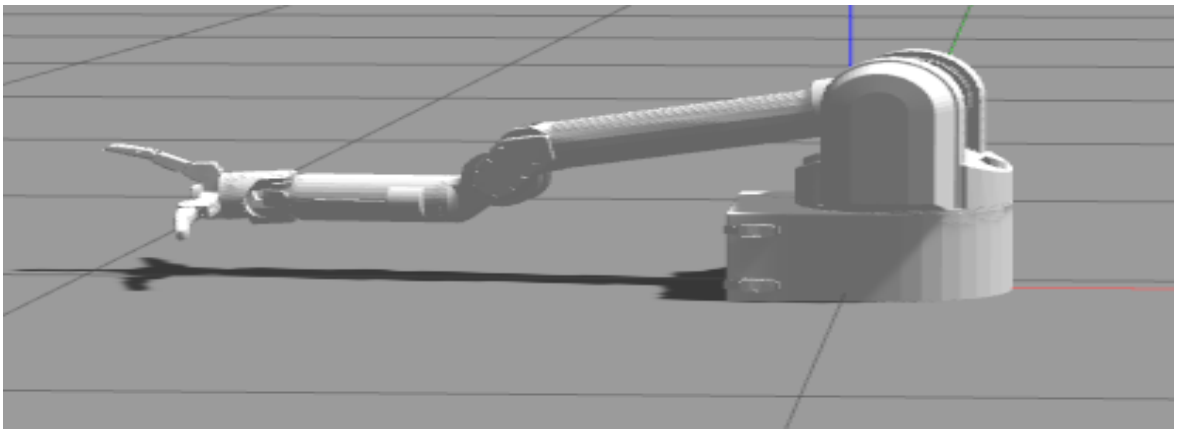

Now similar to the way we did the previous assignment we input the Jacobian parameters and then use it to scatter plot a line. Here we take the 2nd and the 7th joint to be fixed as their motion is similar to other 2 joints and the bot can reach the end position using the movement of other 2 joints alone. Having redundant joint movement does allow us to get multiple orientations for the same goal. So for the sake of simplicity, we can assume that we will be able to operate the manipulator without those joints to access the entirety of our workspace.

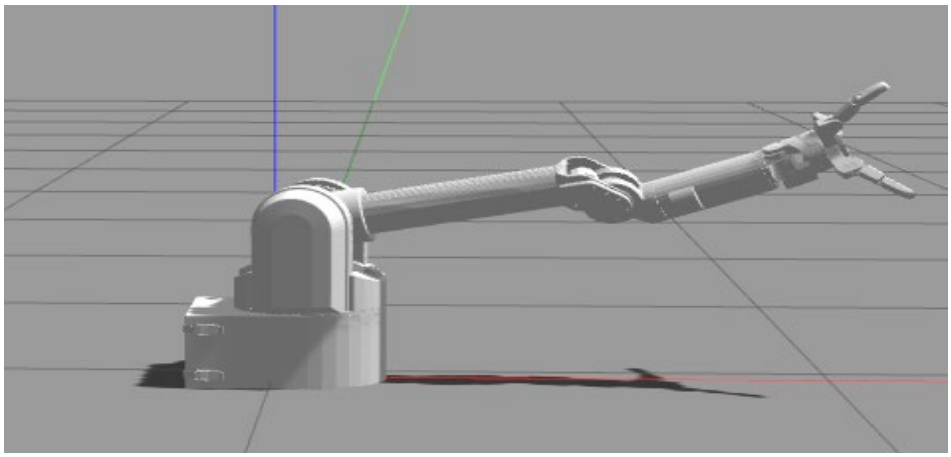
**Workspace Study:**

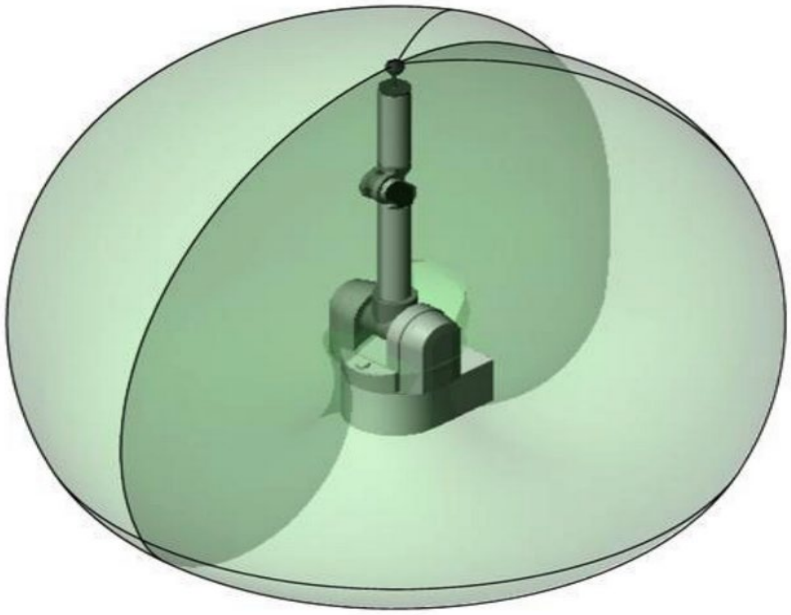
The workspace of the system is spherical in shape with an approximate diameter of 2 meters which is proved by the below images. The below images were produced by fully extending the robot arm in 1 direction in the xy plane and then rotating the first joint for full 360 degrees and then bending the elbow joints (total 3 elbow joints) to produce a spherical workspace.











The combined view of the entire workspace in isometric is shown above.

**Assumptions**

The robot has 7 DOF and for the entirety of the robot system simulation, we assume the following holds true:

1. The links of the robot manipulators are considered to be rigid.
2. The external factors/noise is not taken into consideration.
3. The independent joint error is considered to be zero.
4. For any given situation the path/ trajectory is a part of the set of all possible solutions and is not singular.
5. The motion of the robot is always in a way that no collision occurs.
6. Concerning the safety of the users and the application of employment proposed for the bot, the locking of joints is strictly not allowed as it may result in further complications.
7. Though any trajectory is said to be possible (without collision) any redundancy in usage is to be avoided.
8. The back driving is always done only from rest.

## Control Method

As for the method of control we have used a simple PID controller as it satisfies the scope of the project. Other accurate or adaptive controllers could be used in the future depending upon the implementation and the precision required. The PID used were tuned based on the responses that we could see from the joint movements for the different PID values and the generic oscillations of joints. The PID values were in fact manually tuned and the changes to the controllers were shown through the rqt_gui tool offered in the ros package.

As we are using manipulators, we should have been using Effort Controllers which was the initial plan. But as we reached the point on implementation, we had found that using the Effort controller was throwing multiple errors as we had previously discussed. The error seemed to be because we had used continuous arguments for each of the joints. And then after consultation we had to use revolute joints with essentially a full 360 degree of freedom that was said to rectify the error but to no avail. We then proceed to finally use Velocity controllers and even though they seemed to work we could not publish joint positions as inputs to them.

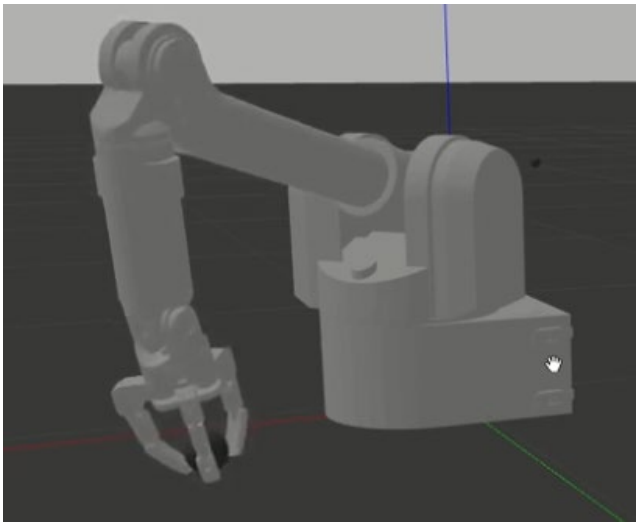
We had learnt to work around that by giving the joints limits and then using the controller input to make them reach those positions. This allowed us to orient the joints in the exact position instead of trying to manipulate the joints manually. This method of manual movement of joints might help us in reaching a certain point but we had to tune the limits in the joint to make sure that we could move them to the exact pose. This method of orienting the joints to the pose lets us easily validate the forward kinematics.

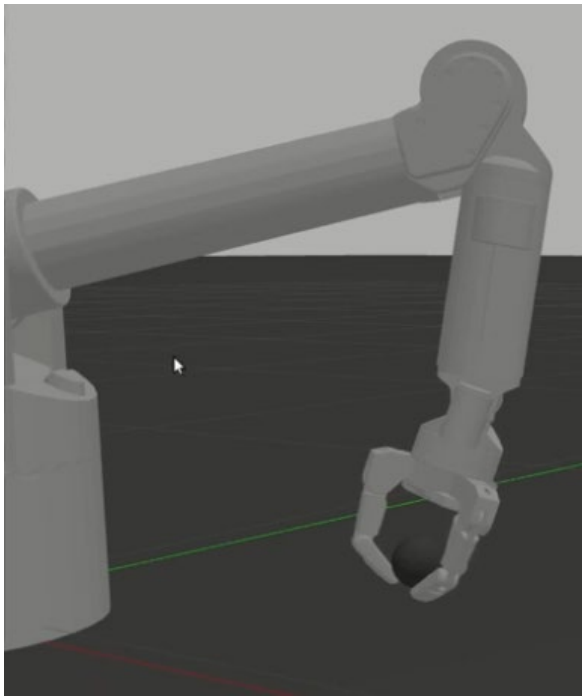
## Visualization

### Gazebo Simulation:

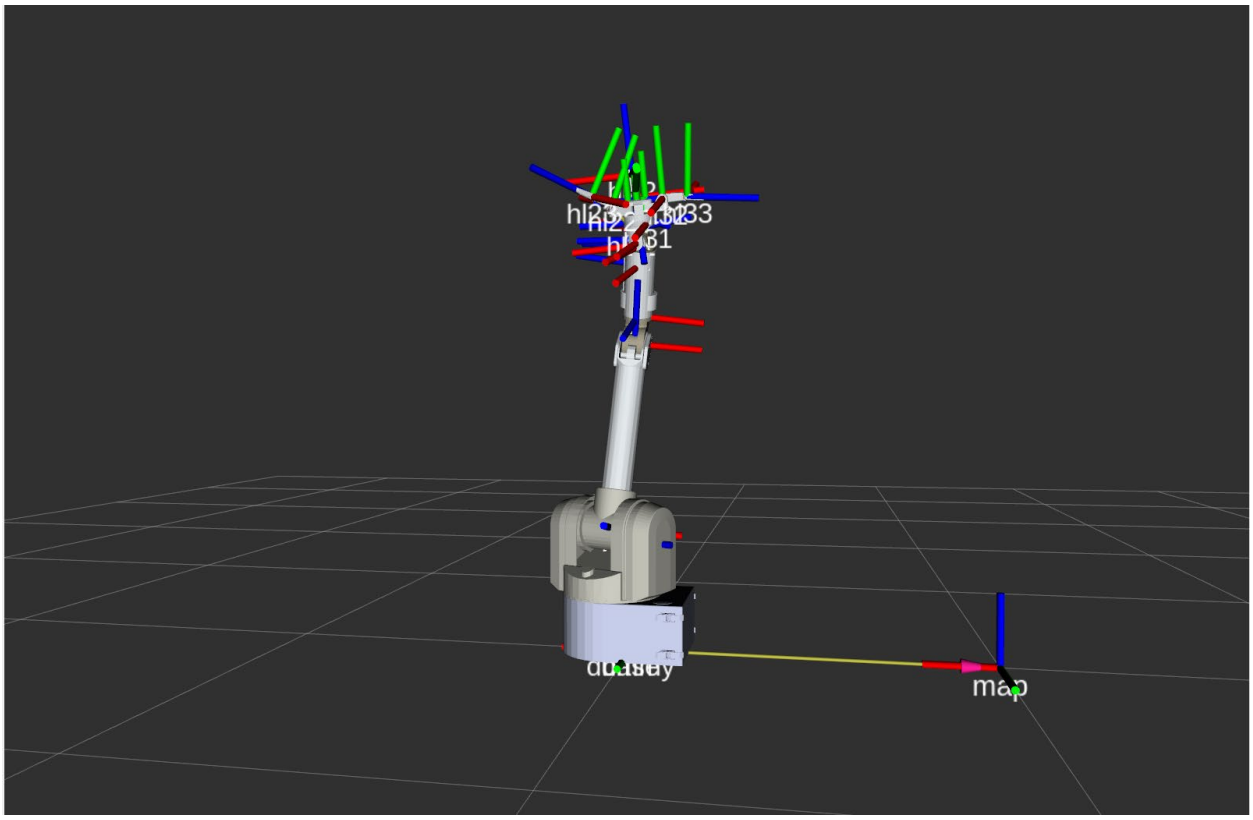
The simulation has been done on gazebo by grasping objects using tele-op to control the end effector to pick up various objects. The usual 2 finger end effector are able to pick up various objects but relies on being able to fully cover the object within the 2 finger but the one that has been implemented in this model has 3 claw type end effector with the ability to individually grip any type of object by being able to both hold the objects from outside as well as being able to fold inward into a “hook-type” of shape which allows to pick up objects such as clothes or even more intricately shaped objects.
The end effector in our simulation video is shown to fully envelop a ball and then proceed to lift it up and move it around for a while.
We observed that even though it is handy to use a 3-finger gripper in this dressing situation but controlling such a gripper can become tedious and difficult to code.

https://drive.google.com/drive/folders/1msxDopHkG6TL6kg3BVi8V42OveKyJZjA?usp=sharing





**RViz**:




We have used the tf topic to show the various frames and their hierarchy in the rviz terminal and the generic position of the robot with respect to the map frame.

**Problems Faced**

- Our initial plan was to use a CAD model directly imported from the Barrett technology website; however, the model had many issues while converting to URDF. We finally used the same parts with small tweaks done by us. This was then re-oriented and assemblies and sub-assemblies were done by us.
- We still faced issues while initially spawning the robot with masses of links and their moments which made the robot fall to the ground. We manually tuned in values using the hit and trial method as the values provided by the company in the data sheet did not make much difference.
- PID tuning took a while as we manually tuned in the values. We used a graphical tool later on for completing this step. The tool used was rqt_gui which allows us to send a function as an input to any of the controllers and then visualize the output in comparison to the desired behavior.
- Our initial plan was to use effort controllers, however, there was an issue while loading those controllers on our systems and eventually we had to switch to velocity controllers.
- Since we had 16 joints, we had a lot of key bindings in our tele-op file, we tried to reduce the number of keys by joining the input to the end effector fingers. As the more key bindings, the more difficult it will be to control the bot.
- Finally, the simulation took very long because orienting the end effector to pick the ball up was difficult. The grasping function took very long, because the ball every time would slip off of the end effector. So, we created more key bindings to orient every hand joint accordingly. We think this problem could have been overcome by the use of effort controllers as well.

**Lessons Learned**

- We improved our skills in CAD modelling and re orienting the parts and assembly according to the coordinate system in Gazebo.
- We got to learn about some advanced mates and also used reference shapes to mate parts.
- PID tuning can be done conveniently with the use of the ROS graphical tool(rqt_gui).
- The base of the robot needs to be strong enough to support the links above it, therefore, URDF should be checked properly for all the links.
- The URDF file can be manually tweaked to resolve minor spawning errors.
- Use of controllers based on the type of joints, like for a continuous joint, velocity controller is apt, and for a revolute joint effort controller.
- The tele-op file must be a well thought implementation especially when the number of joints to be controlled are very high. Similarly, the key bindings need to be intuitive so that it is easy for the human to maneuver the bot.
- We also learned how to create two separate packages for the manipulator arm and end effector and then integrate both in the same world using Xacros. However, we did not use this method as the process became very complex and debugging the errors became a hassle.

## Conclusion

We were able to assemble the robot from the parts that we extracted from the official website for the WAM developers and despite the differences in the origin of different parts with respect to their joint positions, we shifted them to the required axes for the final assembly.

The URDF was particularly tricky as we had to make changes to the urdf constantly to make sure the robot was able to properly spawn, and the weights and the joint types were tuned according to the spawn stability. The joint types were changed from initial continuous to revolute and consequently were given joint and velocity limits based on the kinematic configurations.

The forward kinematics of the robot was rather easy as we already had the exact dimensions and followed the conventions to get the dh table for the calculation. Then the validation for which was done through geometric observation and the tuning of joint limits as mentioned above.

The inverse kinematics of the robot had more than the number of columns needed for a perfect square matrix and its inverse was only calculated using the pseudo inverse method. As for the validation we fixed 2 redundant joints and were able to make the Jacobian a perfect square and then proceed to make the robot trace a straight line which was plotted by the values provided.

In the simulation we showed that our robot was able to pick up one of the objects by individually controlling the end effector joints forming them into an appropriate shape for which the key bindings had 32 keys. The end effector can form complex shapes by manipulating the individual joints which it can use to pick up differently shaped objects.

Although we were able to pick up objects we wanted, we were not able to model the environment in which we could simulate scenarios. We will further work on making the robot follow trajectories and pick up highly complicated objects. We want to implement a controller that lets us have a better movement of the robot and hence allows for high precision applications such as wound dressing, etc.

## Future Work

This specific application is an ongoing project by a few institutions across the world such as Institute of Robotics and Industrial Informatics, Spain, Bristol Robotics Laboratory, UK and IDIAP Institute, Switzerland. They have been trying to induce concepts such as perception, multi-modal interaction, system integration, robot learning and so on. For our project, the immediate future work should be to develop on the control's aspect of the end effector. To make the system more dexterous, we want to add another manipulator and implement it as a single system and work towards more complex dressing applications. The use of sensors will provide feedback on how to orient the robot according to the clothing material as that was one of the biggest challenges in our project. As mentioned earlier, the use of path planning and perception will also be very handy and work towards making the system autonomous. Since this system interacts with humans, several safety aspects should also be kept in mind and simulated accordingly so that no harm is done to humans during the real-life application. If all this is implemented properly, we can clearly see such a system being very helpful for humans in the coming future.

## References


- Greg Chance, IMechE, Antonella Camilleri, Benjamin Winstone, Praminda Caleb-Solly, Sanja Dogramadzi-" An Assistive Robot to Support Dressing – Strategies for Planning and Error Handling "-June 29, 2016.
- C. Torras. Service robots for citizens of the future. European Review, 17-30, 2016
- G. Canal, G. Alenyà and C. Torras. A taxonomy of preferences for physically assistive robots, 26th IEEE International Symposium on Robot and Human Interactive Communication, 2017, Lisbon, Portugal, pp. 292-29.
- cdc.gov
- brl.ac.uk/
- chistera.eu
- advanced.barrett.com
- i-dress-project.iri.upc.edu/
- www.iri.upc.edu/
- who.int/