RBE550: Motion Planning

Transmission – Report

Chaitanya Sriram Gaddipati

Introduction

For this assignment, a 3D RRT planner is implemented to find collision free path to remove the main gear shaft from a SM-465 transmission box. The pose of the shaft contains six variables the x, y and z position variables and roll, pitch and yaw orientation variables. Two variations of RRT are implemented. A normal RRT and bidirectional RRT planner is implemented to observe any differences in the path.

Approach and Planner algorithm design:

For this assignment, robotics tool box in MATLAB is used for creating the model and simulating it. The task is divided into two parts namely, a path planner implementation and the collision detection and model creation.

For the RRT planner, first a class called node is created that contains the pose 'q' and parent node information. A class called RRT is created and it contains the planner implementations and all the necessary components needed for the planner. For the planner an empty tree is created at the start with initial start pose in it. Next the planner iteratively searches for a random pose, converts it into a node and finds the nearest node to it on the tree. From this nearest node the expands to a new node in the direction of the random node using a step size. This new node is then checked for collision with the transmission case and the counter shaft. For collision checking and to create the 3D model in MATLAB, the robotics toolbox is used. Using the tool box each component is created as a rigid body tree and all the shapes are enclosed by bounding collision cylinders. For collision checking I used the built in 'checkCollision' function of the toolbox. Once the new node is checked for collision it is added to the tree with the nearest neighbour as its parent if there are no collisions. If there is a collision the new node is discarded and we move on. The planner stops once when a node within a small distance to the goal is found. To improve the speed of search or to direct the tree towards the goal for every few iterations the random node is chosen as goal node.

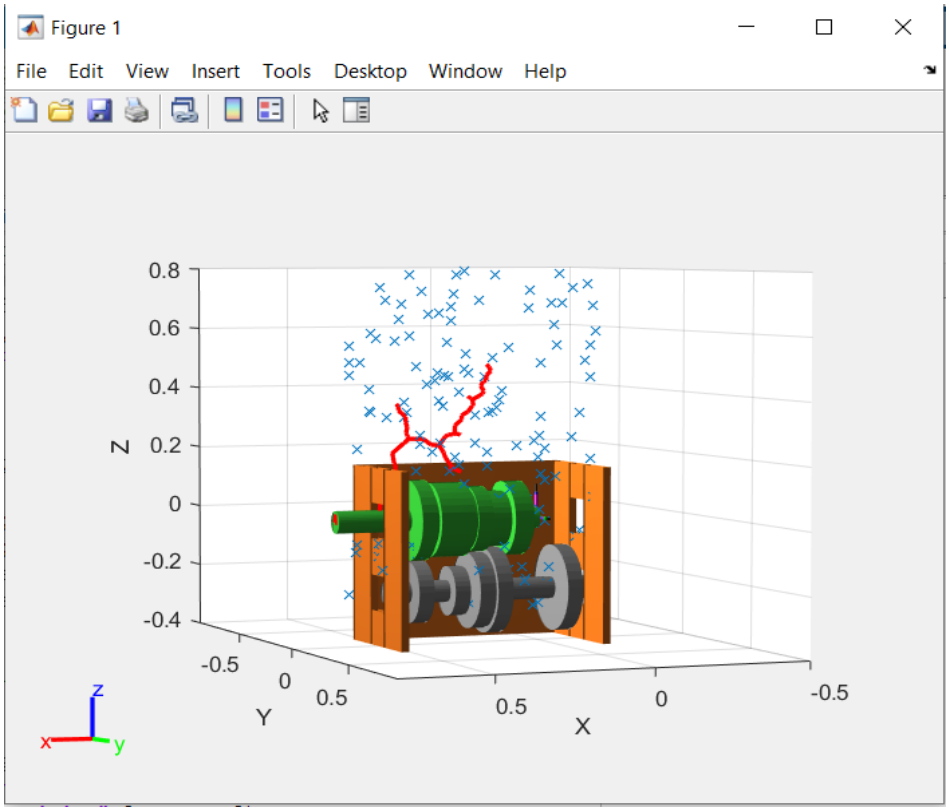
Bidirectional RRT follows the same approach, but here there are two trees, one that starts from start node and the other one from goal node. Both trees expand and once the new nodes for both the trees are within a small distance the trees are joined to find a combined path from start to the goal. Both planners return the final path containing the nodes from start to the goal.

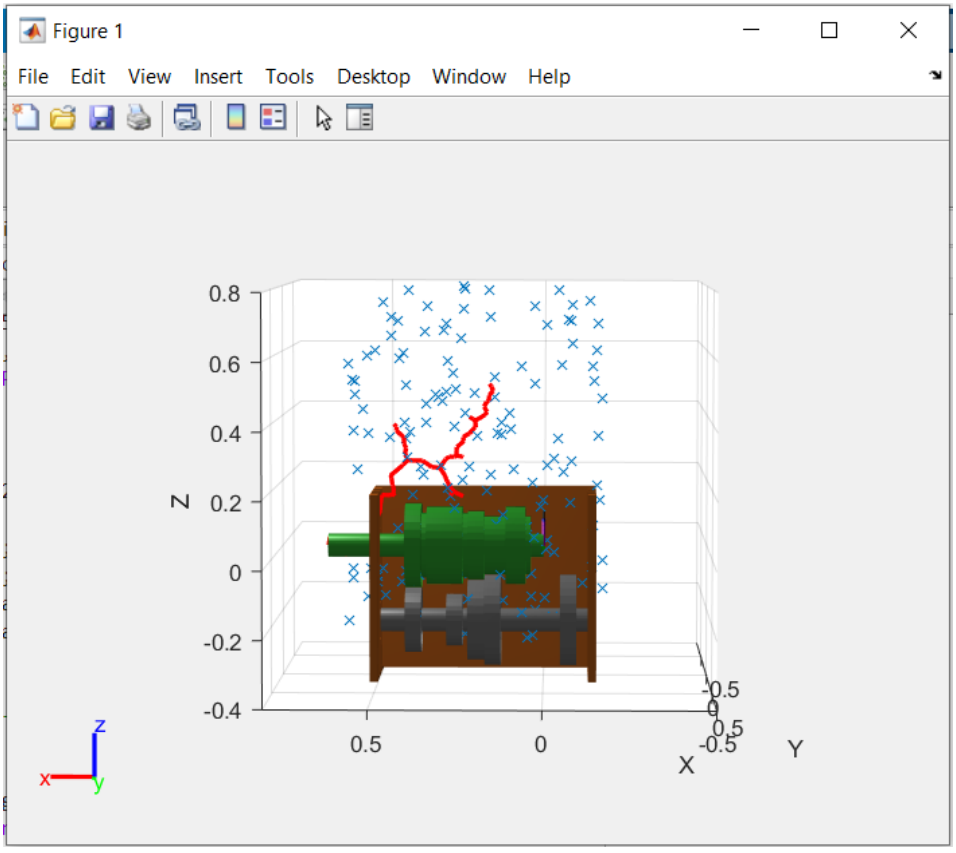
Results and Path Depiction:

The plots obtained for both the planners are attached below. For running the code go to 'main.m' file and run.

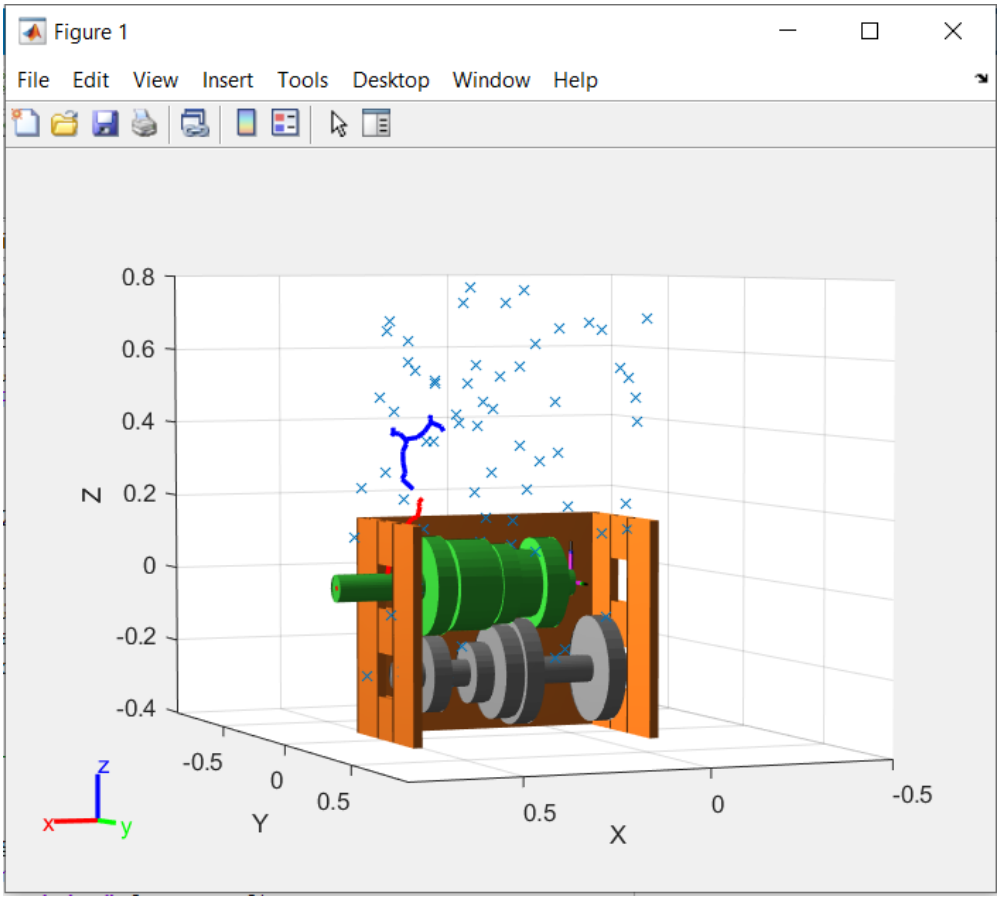
RRT graph plot:

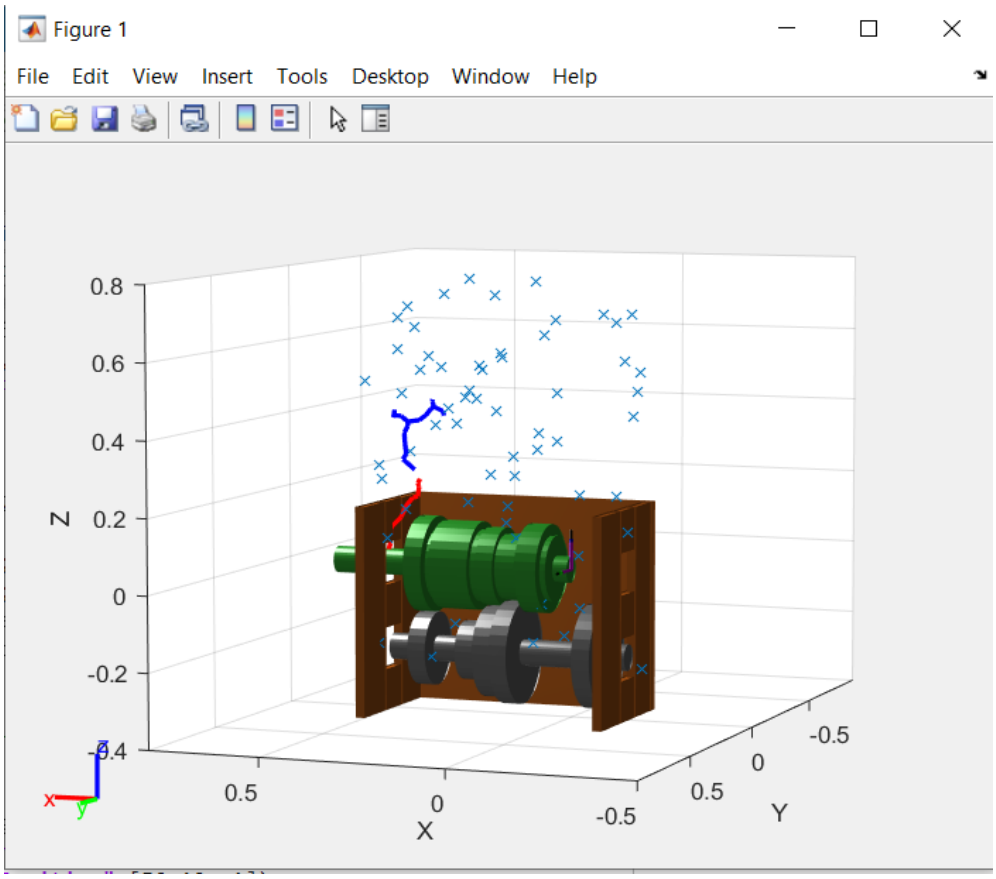
For unidirectional RRT planner:





For bidirectional RRT planner:

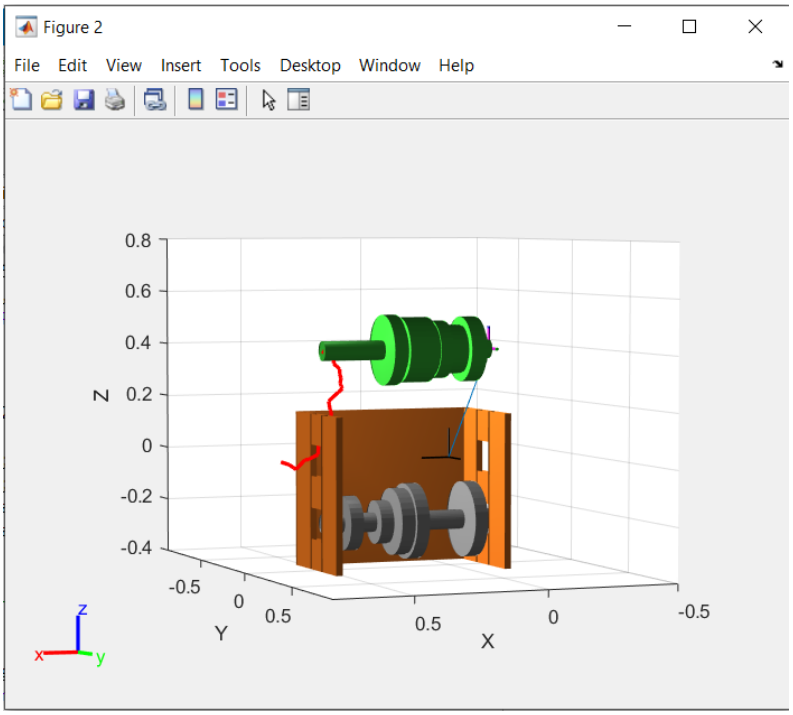


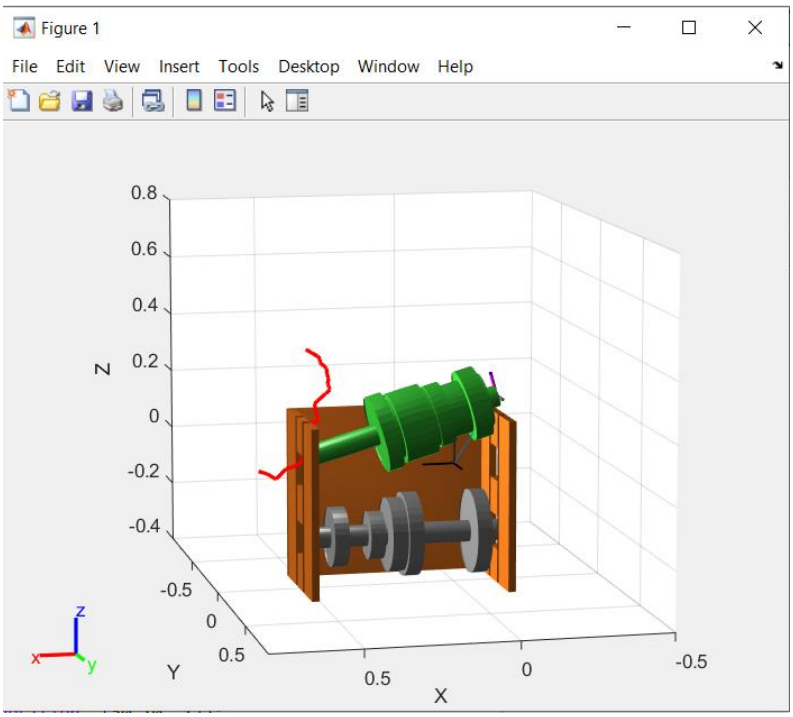


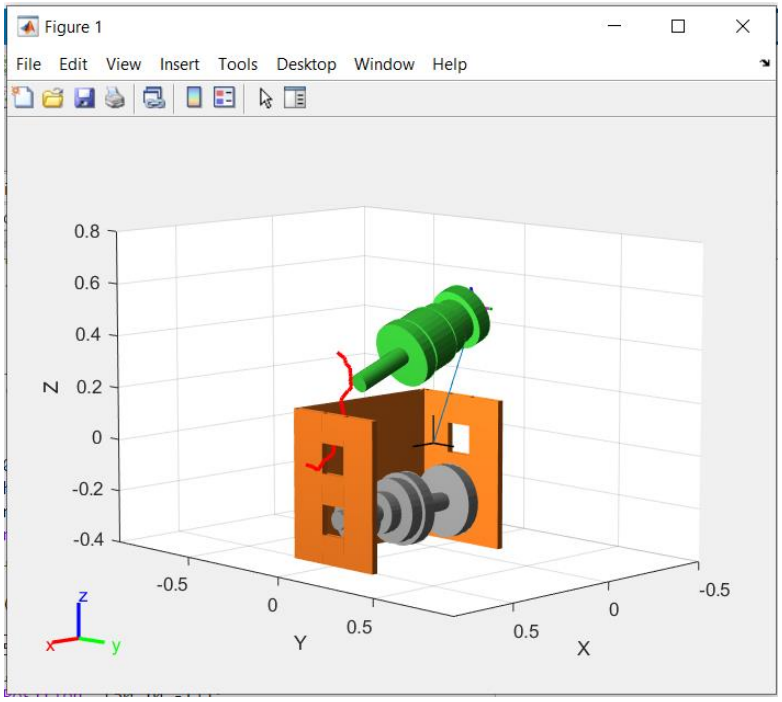
In the above plots of the planner trees, the ‘X’ mark represents the node positions considered by the planner and the line shows the tree of the planner expanding and exploring. For bidirectional planner, the red line shows the tree from start location and blue line is the tree from the goal location. It can be seen that both trees are merged when the distance between the two nodes on each of the trees is below a small tolerance value.

Plots of the resulting Final path:

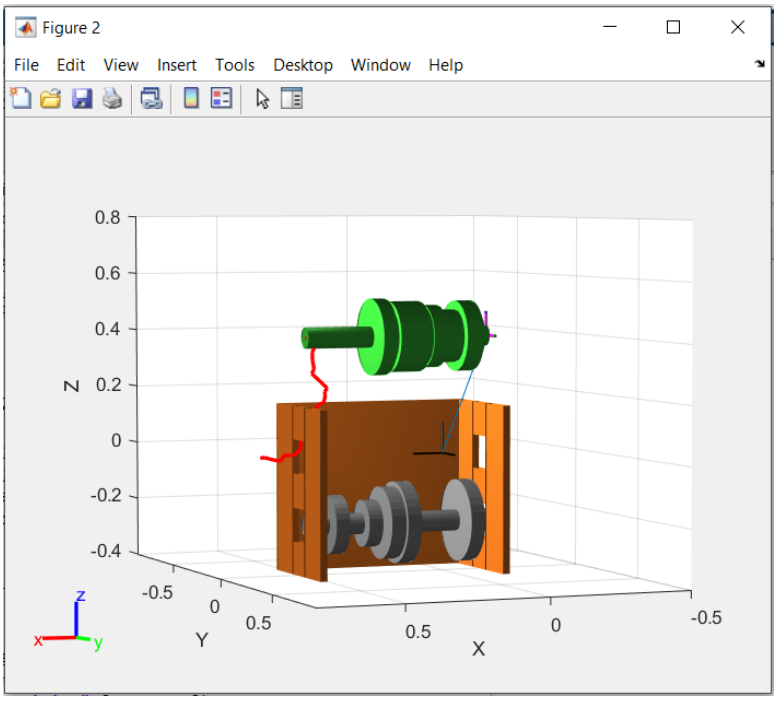
For unidirectional RRT planner the resulting final path is shown below with main shaft at different locations along the final path.

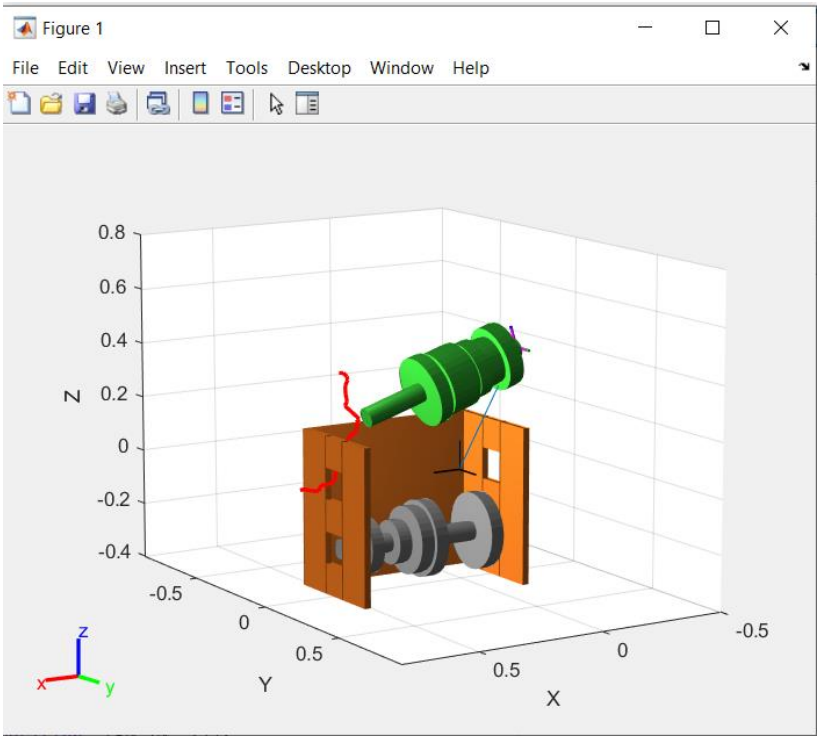


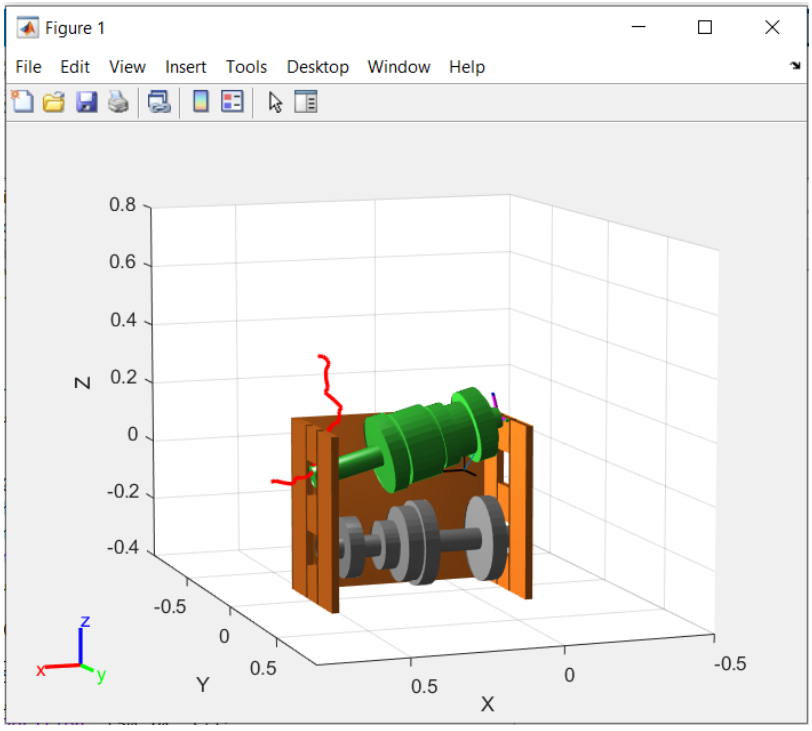




Similar plots for the bidirectional planner are given below.







Conclusion/Discussion:

Both the planners were successfully able to take the main gear shaft out of the transmission box. For both the planners it was observed that the paths generated were not very smooth and uniform and this could be improved by adding constraints on the new node that is being attached to the tree. In terms of the time taken to find the path, it has been observed that there was no significant difference between the unidirectional or bidirectional planners. This is due to the stopping constraint for the bidirectional planner where the search terminates if

the new node for the start and goal trees is less than certain distance. This stopping condition could further be improved by searching both trees for nearest node to each other for every iteration. A significant computational time in both the planners is wasted in finding the nearest neighbour iteratively by comparison. This could be improved by using a KD Tree like data structure. The collision checker utilized is accurate up to 10^-5th order so the collision detection is accurate and fast.

References:

For collision detection and collision geometry model creation:

https://www.mathworks.com/products/robotics.html

For gif or animation creation:

https://www.mathworks.com/matlabcentral/fileexchange/63239-gif