Jacobian-based Motion Planning in Pick and Place Problem for 6 DOF and 7 DOF Manipulators

Yongxin Guo, Albert Yang

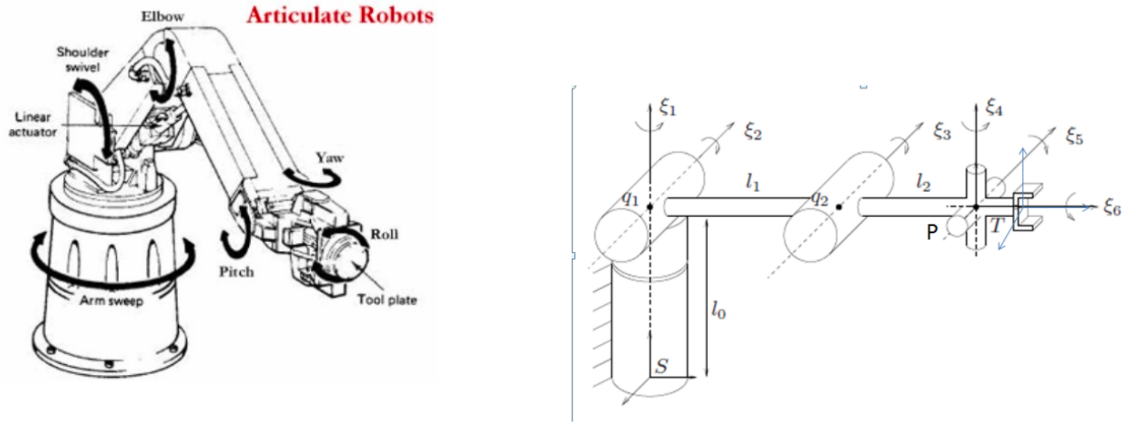
May 18, 2019

Abstract

The pick and place problem in robotics field is one of the most important topics for the future goal of transferring into industry 4.0 era. In the modern industry production line, the accuracy and precision in the pick and place problem largely determine the quality of the products being manufactured. In addition to the generation goal of the manipulator trajectory, the obstacle-avoidance task and joint limit constraints also need to be carefully considered. This report focuses on the implementation of Jacobian-based motion planning algorithm combined with Special Euclidean Group $SE(3)$ interpolation technique for 6 DOF elbow manipulator and 7 DOF Baxter robotic arm to solve a general pick and place problem coupled with the goal of obstacle-avoidance. The specific tasks were assigned to these two robotic manipulators. The quality of the selected algorithms were examined and studied following the presentation of the resulted data. The above tasks were mainly carried out in MATLAB along with Robotics Toolbox developed by Peter Corke.

1 Introduction

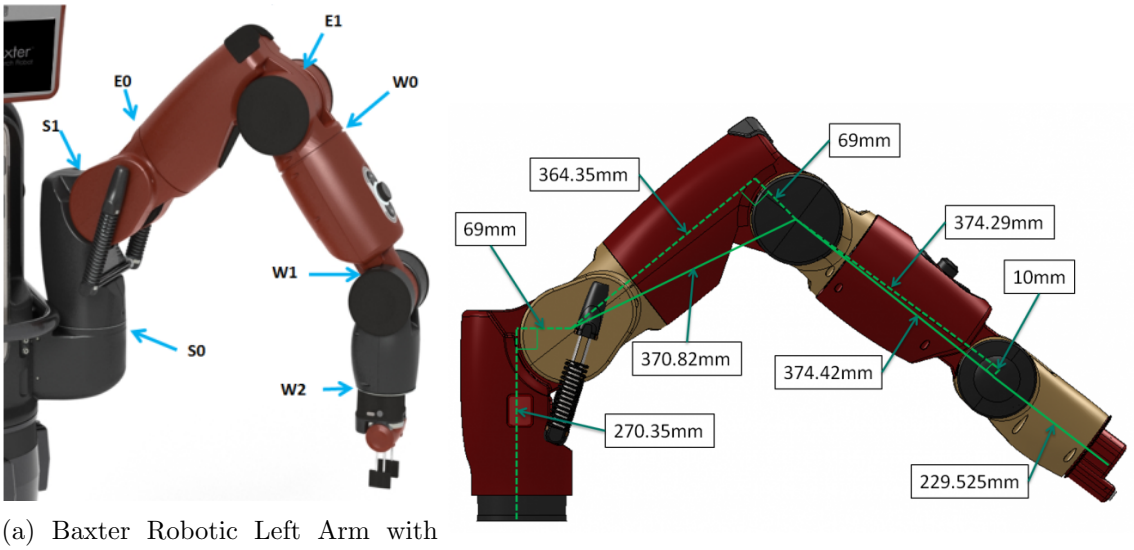
In pick and place problem, the trajectory generation in end-effector space is the core of the game. However, the trajectory can neither be purely generated from joint space(forward kinematics) nor from $SE(3)$ configuration space(inverse kinematics) since the inverse kinematics will produce multiple solutions in joint space and the process of selecting the correct one from them is usually time-consuming and a waste of computational resources. Thus, a combined algorithm using inverse velocity kinematics and interpolation in dual quaternion space was adopted for the goal of this project. There are two robotic arms as the interests of this project, which are an elbow manipulator with 6 DOF and a Baxter robotic arm with 7 DOF as shown in Figure 1.1 and 1.2



(a) Articulate Robot

(b) Elbow Manipulator Simplified Model

Figure 1.1: 6 DOF Elbow Manipulator



(a) Baxter Robotic Left Arm with Joints Specified

(b) Baxter Robotic Arm Dimension

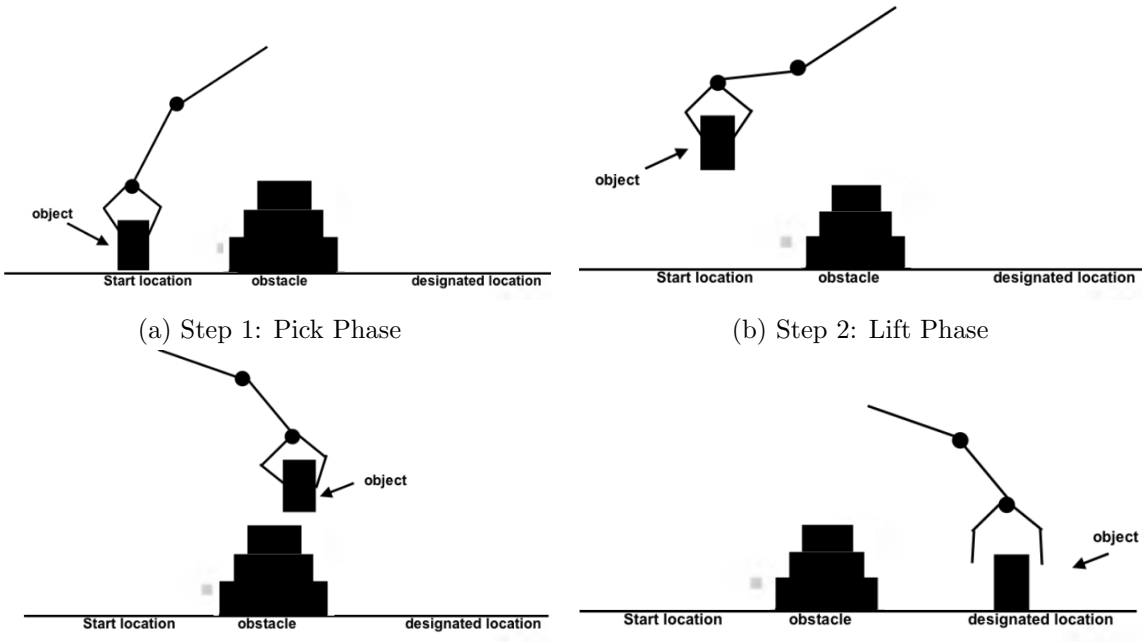
Figure 1.2: 7 DOF Baxter Robotic Arm

The dimension of the Baxter robotic arm has been specified in Figure 1.2b by the manufacturer, and the dimension for the elbow manipulator has been selected to be as shown in Table 1.1

	l_0(m)	l_1(m)	l_2(m)	T(m)
Dimension	0.8	0.8	0.5	0.2

Table 1.1: Link Dimensions of Elbow Manipulator

The task assigned to the elbow manipulator and Baxter robotic arm is a classic pick and place problem that the robotic arms has been asked to pick up a product from one side of the production line to the other side of the product line by bypassing a workbench with different types of tools placed on top of it. This is generally considered as a classic and frequent mission occurred in the automated assembly line of modern industries. A series of schematic figures were shown in Figure 1.3 as a better illustration to the task.



(a) Step 1: Pick Phase

(b) Step 2: Lift Phase

(c) Step 3: Lay Phase

(d) Step 4: Place Phase

Figure 1.3: Pick and Place Task Breakdown

As shown in Figure 1.3, the workbench has been modified into three stacking rectangular blocks with all the dimensions documented in Table 1.2. The entire pick and place process has been divided into four main steps: (1)Pick Phase, (2)Lift Phase, (3)Lay Phase and (4)Place Phase. The four steps together made up the basic guidance for generating the complete manipulator trajectory.

	Width(m)	Length(m)	Height(m)
Bottom Block	0.35	0.35	0.3
Middle Block	0.3	0.3	0.1
Top Block	0.2	0.2	0.1

Table 1.2: Dimensions of Three Stacking Blocks as Obstacles

For elbow manipulator, the task is only to bypass the obstacle in order to delivery the products from one location to the other with the orientation of the end-effector fixed all the time. However, for Baxter robotic arm, the task becomes slightly more complicated that it has to pick up the product from a horizontal product line to the other special product line with **45 degrees** of inclined angle. Thus, in addition to the constraints of obstacles, the Baxter robotic arm has to reach a specific end-effector orientation at the designated inclined production line, as illustrated in Figure 1.4.

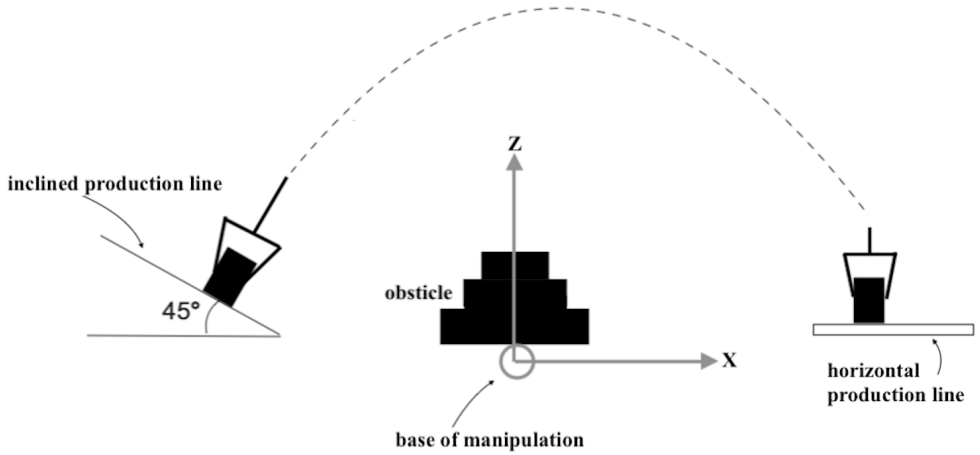


Figure 1.4: Inclined Production Line Task for Baxter Robotic Arm

2 Problem Statement

2.1 6-DOF Elbow Manipulator Problem Statement

Assuming when robotic arm picks or places the object the production line is instantaneously at rest at the moment. Then, the task can be stated as follows:

(1) Given an initial configuration of the target product at the starting production line, $\boldsymbol{g_0(\Theta_0)}$ expressed in configuration space that takes the form of transformation matrix. Also given the final configuration of the destination at the target production line, $\boldsymbol{g_m(\Theta_m)}$. Develop an algorithm that is capable of accomplishing:

$$g_0(\Theta_0),\ g_m(\Theta_d) \xrightarrow{\text{Input}} \textbf{Algorithm}\ \ \mathbf{M} \xrightarrow{\text{Output}} g_i(\boldsymbol{\Theta_i}),\ \ for\ i = 0...m \xrightarrow{\text{Input}} Physical\ Model$$

Where m+1 is the total number of configurations automatically generated by the algorithm M including g_0 and g_m. Each $\boldsymbol{\Theta_i}$ is defined as a column vector of $[\theta_{1i}, \theta_{2i}...\theta_{ni}]^T$, where n denotes the degree of freedom for the manipulator. e.g, the column vectors θ_0 and θ_m are both subsets of $\boldsymbol{\Theta_i}$. Overall speaking, $\boldsymbol{\Theta}$ is a matrix with size of n $\times$ (m+1).

Please note that the output from Algorithm G, $g_i(\boldsymbol{\Theta_i})$, is considered as the trajectory expressed in **configuration space**, and $\boldsymbol{\Theta_i}$ is the trajectory expressed in **joint angle space**.

(2) The orientation of the end-effector remains fixed throughout the entire trajectory in order to maintain the proper and stable manipulation of the target object. The task can be stated mathematically as follows:

$$For\ i = 1...m,\ R_i = R_0,\ from\ [g_1(\Theta_1), g_2(\Theta_2)...g_m(\Theta_m)]$$

(3) The end-effector should not get very closed to the obstacle such that the delivery task can be successfully carried out. Thus, each point of the trajectory expressed in configuration space should satisfy the following rule in Euclidean space:

$$For\ i = 0...m,\ ||\boldsymbol{P_i} - \boldsymbol{P_{obstacle}}|| > d,\ from\ [g_0(\Theta_0), g_1(\Theta_1)...g_m(\Theta_m)]$$

Where $\boldsymbol{P_{obstacle}}$ is known to the operator, and d represents a distance criteria that the algorithm M must follow in order to not crash the object into the obstacle.

Please note that the statement (3) cannot solve the obstacle-avoidance problem in general but it can provide a guide within the scope of the project when selecting the configurations that make up the trajectory. Statement (3) only serves as a basic check that is computationally saving. ;

2.2 7-DOF Baxter Robotic Arm Problem Statement

The Baxter has task statements as same as statement (1), (2) and (3) as stated in Section 2.1, but with an addition constraint about the orientation at the designated location, which can be mathematically stated as follows:

$$When\ i = m,\ g_i = \begin{bmatrix} \boldsymbol{R_m} & \boldsymbol{P_m} \\ \boldsymbol{0_{1\times 3}} & 1 \end{bmatrix},\ and\ \boldsymbol{R_m} = \boldsymbol{R_{Y,\pi/4}R_{Z,\pi/4}}$$

Where P_m is known to the operator, and $\boldsymbol{R_{Y,\pi/4}R_{Z,\pi/4}}$ denotes a rotation matrix being rotated around Z-axis by 45 degrees and then rotated again around Y-axis by 45 degrees. The first rotation is to match the orientation of the end-effector with the base reference frame as shown in Figure 1.4, which is fixed to the Baxter body not the arm and the Baxter arm is designed to be inclined with the body by 45 degrees around Z-axis so that applying $\boldsymbol{R_{Z,\pi/4}}$ to the end-effector can bring the coordinate system at the end-effector back to be the same as base reference frame. The second rotation is then to rotate around Y-axis, which points into the page in Figure 1.4, by 45 degrees in order to meet the special requirement imposed by the inclined production line.

3 Solution Approach

3.1 Trajectory Generation

The trajectory generation is the core problem involved in both two robotic manipulators. The Jacobian-based motion planning algorithm combined with the $SE(3)$ interpolation has been used to generate the trajectory between two specified configurations. First of all, when two configurations has been selected as the initial and final point, the technique to obtain any points between the start and end is analogous to the interpolation in $\mathbb{R}^{\mathbf{3}}$ space. However, the points along the trajectory are expressed in $SE(3)$ space, one has to convert the transformation form into dual quaternion form. Since dual quaternion is expressed in 8-dimensional space or $\mathbb{R}^{\mathbf{8}}$, the interpolation in $\mathbb{R}^{\mathbf{8}}$ has to be done following Equation 3.1.

***SE*(3) Interpolation:**

$$C(\tau) = A \otimes (A^* \otimes B)^\tau, \quad 0 \leqslant \tau \leqslant 1 \tag{3.1}$$

Where A and B are inital and final configuration expressed in dual quaternion form, $C(\tau)$ denotes the intermediate configuration as a function of τ, and τ represents a step-size parameter. When $\tau = 1,\ C(\tau) = B$, when $\tau = 0,\ C(\tau) = A$.

Jacobian-based Motion Planning:

Now that the intermediate configurations can be determined by using Equation 3.1, one still needs to convert the configuration space into joint angle space. As discussed in the beginning of Section 1, the inverse kinematics cannot be applied in this case. Thus, the inverse velocity kinematics was used instead combined with Euler time-step discretization.

(1) Inverse Velocity Kinematics:

$$\dot{\Theta} = B\dot{\gamma} \tag{3.2}$$

(2) Euler time-step discretization applied to Equation 3.2:

$$\Theta(t+h) = \Theta(t) + \beta B(\Theta(t))(\gamma(t+h) - \gamma(t)) \tag{3.3}$$

Where,

$$B = (J^s)^T(J^s(J^s)^T)^{-1}J_2, \quad with\ J_2 = \begin{bmatrix} I & 2\hat{p}J_1 \\ 0 & 2J_1 \end{bmatrix},\ J_1 = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{3.4}$$

Note that for elbow manipulator,

$$B = (J^s)^{-1}J_2 \tag{3.5}$$

Since the spatial Jacobian J_s for elbow manipulator has size of $\mathbf{6 \times 6}$ so that there is no need to perform pseudo-inverse.

Also,

$$0 < \beta < 1 \tag{3.6}$$

β represents a step-length parameter, which serves as a correction factor when step-size parameter τ becomes coarse.

Please note that when inputting the configuration $C(\tau)$ obtained from Equation 3.1 into the Equation 3.3 to get the joint angle $\Theta(t+h)$, one has to convert the configuration from dual quaternion form to γ form, which is defined as:

$$\gamma = \begin{bmatrix} p \\ Q \end{bmatrix}$$

Where p stands for the position vector and Q stands for the rotation quaternion.

Once we are able to obtain the trajectory point expressed in joint angles, then it becomes possible to convert from joint angle space fluently to configuration space via forward kinematics algorithm $F.K(\Theta)$. Then, the new configuration space point can be treated as the initial configuration again and feed it to the starting of the loop inside the ***algorithm M***.

The foundation of success in ***algorithm M*** relies on the correct implementation in the basic quaternion and dual quaternion operation. A tester function has been wrriten during the code development for testing these fundamental operations before moving on into the core development of $SE(3)$ Interpolation and Jacobian-based motion planning. An algorithm structure has been demonstrated in Figure 3.1

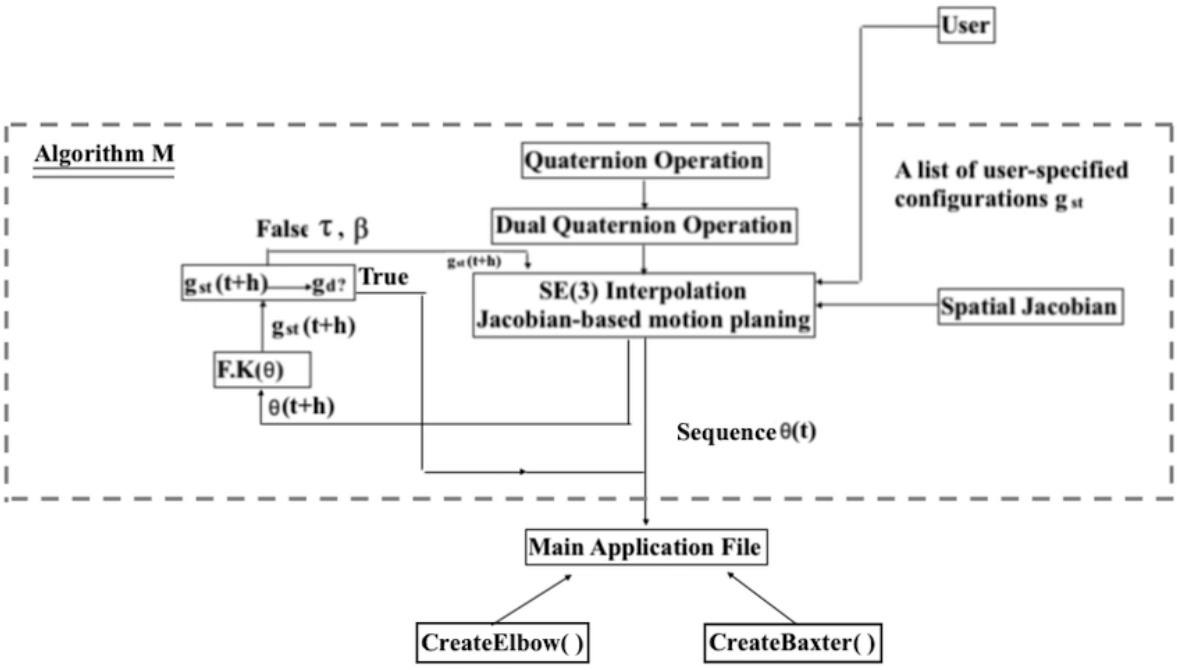


Figure 3.1: Structure of Alogrithm M

As the stopping criteria for the looping algorithm, the following criteria has been used for evaluating the proximity of position and orientation to the desired(given) configuration.

(1) Criteria for P in $\mathbb{R}^3$ space

$$d(P_i, P_m) = \sqrt{(x_i - x_m)^2 + (y_i - y_m)^2 + (z_i - z_m)^2} \tag{3.7}$$

(2) Criteria for Q in $\mathbb{R}^4$ space

$$\phi(Q_i, Q_m) = min\{\|Q_i - Q_m\|, \|Q_i + Q_m\|\} \tag{3.8}$$

The user-defined thresholds were assigned to d and ϕ.

	d	ϕ
Threshold	0.01	0.01

Table 3.1: Threshold for d and ϕ

3.2 Jacobian Singularity Problem

During the implementation of the algorithm, it is found that a initial pose of the manipulator with all joint angles equal to 0 degree can cause the Spatial Jacobian J_s fall into **singularity**. Thus, the solution to that is to assign a random small amount of angle to each of manipulator joints, 3 degrees in this case for both manipulators, and get the configuration of end-effector $g_{rand}(\Theta_{rand})$ expressed in $SE(3)$ via forward kinematics $F.K(\Theta_{rand})$. Then, the manipulator will perform the trajectory generation starting from $g_{rand}(\Theta_{rand})$. Thus, the singularity problem can be harnessed without jeopardizing the accuracy of the results.

$$\Theta_{rand} = [\mathbf{3}^\circ] \rightarrow \Theta_0 \rightarrow F.K(\theta) \rightarrow g_0(\Theta_0)$$

3.3 Dynamical step-size parameter $\boldsymbol{\tau}$

One of the features from algorithm M is that the step size gets finer and finer as it approaches to the user-specified final configuration so that the manipulator gets a large number of joint angle data at that moment. However, a large number of trajectory points will not do anything helpful once the change in joint angle is smaller than the resolution of the joint servos but it becomes a burden to the task in terms of efficiency and time. Thus, a additional change is added to the criteria (1), the Equation 3.7.

$$When\ d < 0.1, \Longrightarrow \boldsymbol{\tau = 5 \times tau, \beta = 0.9}$$

$$Otherwise \Longrightarrow \tau = 0.01,\ \beta = 1$$

3.4 Obstacle Avoidance

The solution to the obstacle avoidance is to select the intermediate points up in the air above the obstacles, which partitions the trajectory into several segments. The idea is illustrated in Figure 3.2. If directly generating the trajectory using only the inital and final configurations, it is difficult

to predict how the trajectory will evolve. This way to some extent one can have a control on the trajectory and make sure the end-effector will successfully bypass the obstacle from the above once the selected intermediate points are far enough away from the top surface of the obstacle. As shown in Figure 3.2, N represents the amount of intermediate points selected. The larger N is, the more control one can have on the trajectory. In this project, it is investigated that $\boldsymbol{N = 1}$ and $\boldsymbol{N = 2}$ would respectively satisfy the task required for elbow manipulator and Baxter robotic arm .

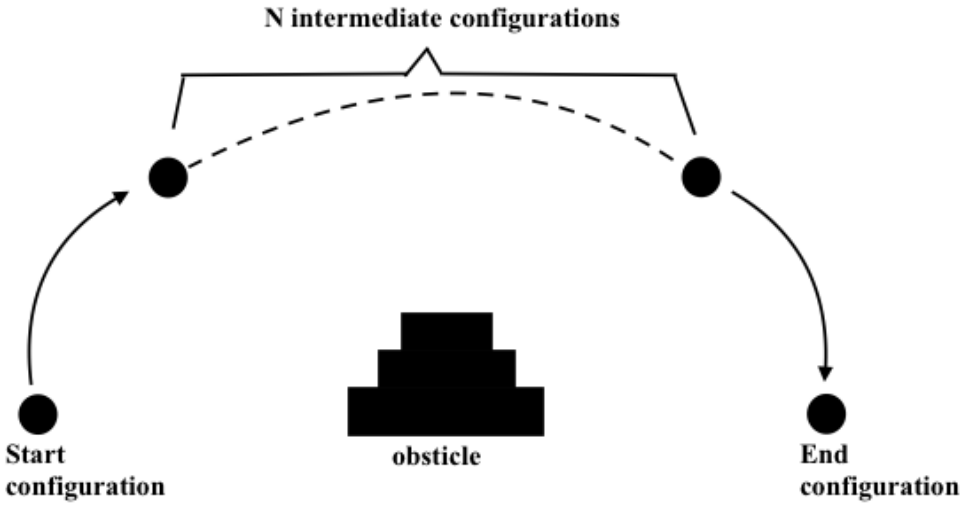


Figure 3.2: Obstacle Avoidance Solution

Let K be the number of user-specified configurations, and L be the number of trajectories automatically generated by ***algorithm M***. Then,

$$K = N + 2 \; and \; L = K - 1 \tag{3.9}$$

E.g, when $N = 1$, there are 3 user-specified configurations in total, and 2 trajectories are generated.

4 Results

4.1 Elbow Manipulator:

Input:

(1) Three stacking rectangular boxes whose sizes are illustrated in Table 1.2 locates at XY-plane with the center of obstacle's base being at $[0, 0.5, 0]^T$, which is actually placed along Y-axis.

(2) **3 degrees** of initial angles assigned to each joints of the manipulator. The motion planning starts from there. The corresponding transformation matrix is $g_0(\Theta_0) = g_{rand}(\Theta_{rand})$

(3) Three user-specified configurations.

Initial configuration:

$$g_{init} = \begin{bmatrix} R_{init} & P_{init} \\ \mathbf{0} & 1 \end{bmatrix}, \ with \ R_{init} = R_{rand}, \ and \ P_{init} = \begin{bmatrix} 0.7 \\ 0.5 \\ 0.2 \end{bmatrix}$$

Intermediate configuration:

$$g_{inter} = \begin{bmatrix} R_{inter} & P_{inter} \\ \mathbf{0} & 1 \end{bmatrix}, \ with R_{inter} \ = R_{rand}, \ and \ P_{inter} = \begin{bmatrix} 0 \\ 1 \\ 1.2 \end{bmatrix}$$

Final configuration:

$$g_{final} = \begin{bmatrix} R_{final} & P_{final} \\ \mathbf{0} & 1 \end{bmatrix}, \ with R_{final} \ = R_{rand}, \ and \ P_{final} = \begin{bmatrix} -0.7 \\ 0.5 \\ 0.2 \end{bmatrix}$$

Where R_{rand} can be directly obtained from $F.K(\Theta_{rand})$.

Result:

The computed trajectory was shown in Figure 4.1 with the coordinate frame marked, and it can be seen that the orientation of the coordinate frame throughout the entire trajectory remain fixed. It is also found that the trajectory expressed in $\mathbb{R}^3$ space is several segments of straight line. This is due to the fact that the orientation of the end-effector remains the same all the time. It can also serve as a check to the correctness of the algorithm M.

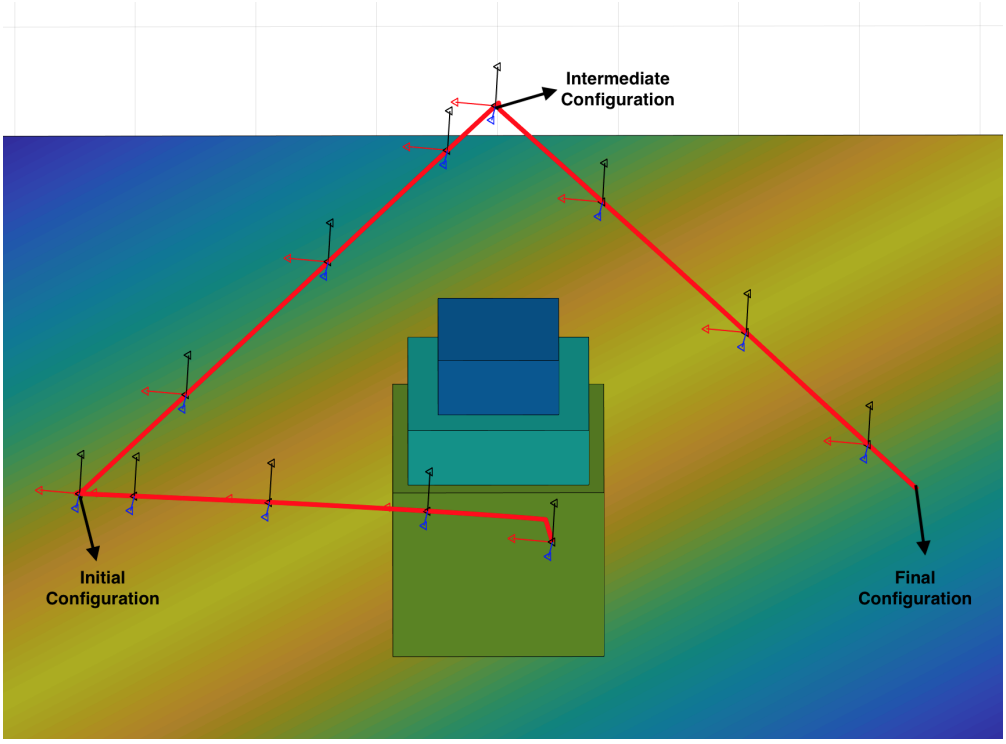
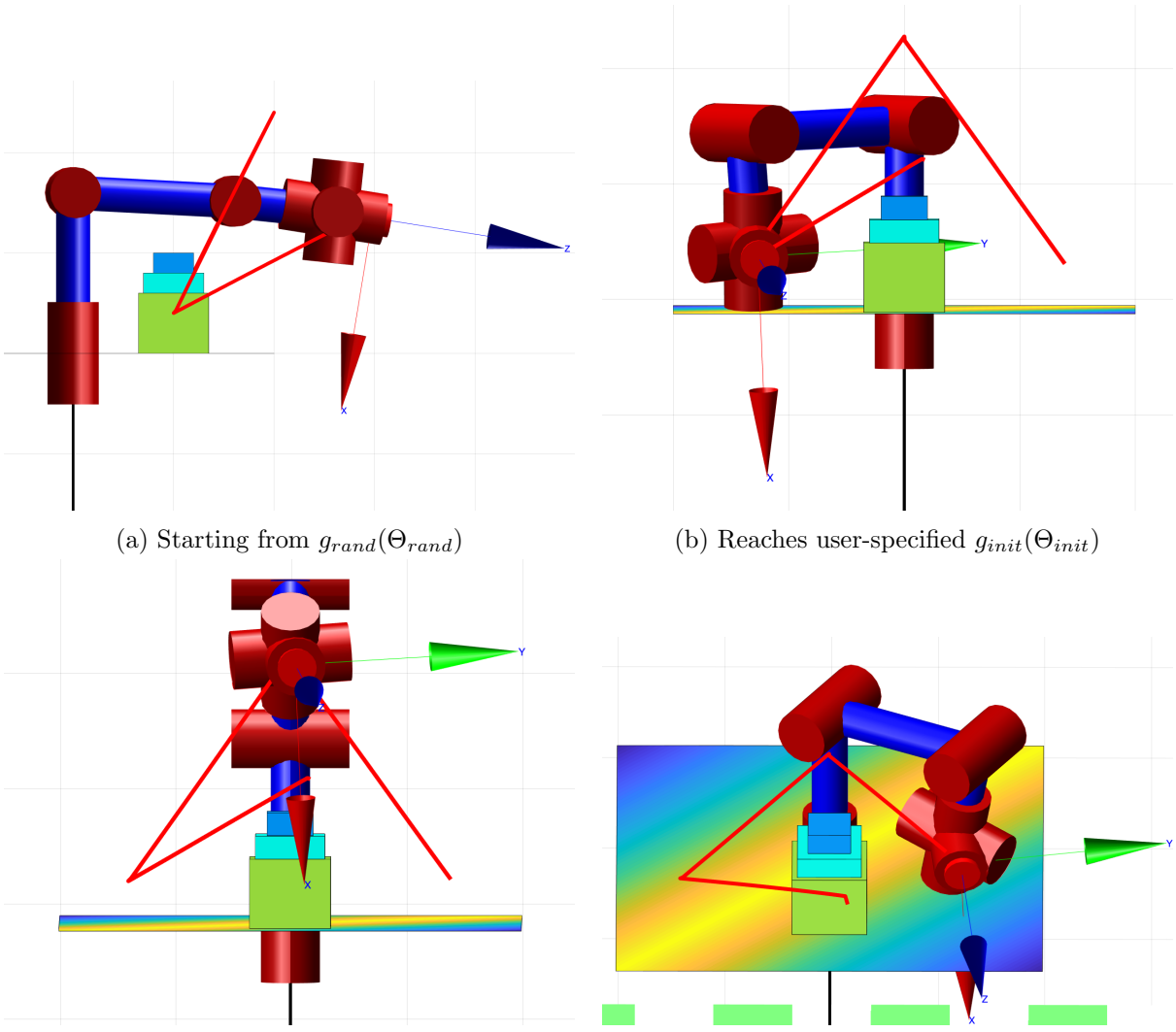


Figure 4.1: Computed Trajectory for Elbow Manipulator

By incorporating the Robotics Toolbox developed by Peter Corke, one can physically make out the elbow manipulator of SerialLink objects and create the correct robotic configuration using Denavit–Hartenberg parameters. Seeing how the manipulator changes is more intuitive and helps in terms of trajectory design. Four important moments for the elbow manipulator are shown in Figure 4.2, and along with data attached in Table 4.1

	Computational Time	Number of Steps
Reported Data	0.54346s	920

Table 4.1: Trajectory Data of Elbow Manipulator



(a) Starting from $g_{rand}(\Theta_{rand})$

(b) Reaches user-specified $g_{init}(\Theta_{init})$

(c) Reaches user-specified $g_{inter}(\Theta_{inter})$

(d) Reaches user-specified $g_{final}(\Theta_{final})$

Figure 4.2: Four Important Moments for Elbow Manipulator along the Trajectory

An animation of the elbow manipulator following the trajectory can be accessed via the link:

Elbow Manipulator Animation

4.2 Baxter Robotic Arm:

Input:

(1) Three stacking rectangular boxes whose sizes are illustrated in Table 1.2 locates at XY-plane with the center of obstacle's base being at $[0, 0.7, 0]^T$, which is actually placed along Y-axis.

(2) **5 degrees** of initial angles assigned to each joints of the manipulator. The motion planning starts from there. The corresponding transformation matrix is $g_0(\Theta_0) = g_{rand}(\Theta_{rand})$

(3) Three user-specified configurations.

Initial configuration:

$$g_{init} = \begin{bmatrix} R_{init} & P_{init} \\ \mathbf{0} & 1 \end{bmatrix}, \ with \ R_{init} = R_{rand}, \ and \ P_{init} = \begin{bmatrix} 0.6 \\ 0.7 \\ 0.2 \end{bmatrix}$$

Intermediate configuration N_1:

$$g_{N_1} = \begin{bmatrix} R_{N_1} & P_{N_1} \\ \mathbf{0} & 1 \end{bmatrix}, \ with \ R_{N_1} = R_{Z,\pi/4}, \ and \ P_{N_1} = \begin{bmatrix} 0 \\ 0.8 \\ 0.8 \end{bmatrix}$$

Intermediate configuration N_2:

$$g_{N_2} = \begin{bmatrix} R_{N_2} & P_{N_2} \\ \mathbf{0} & 1 \end{bmatrix}, \ with \ R_{N_2} = R_{Z,\pi/4}, \ and \ P_{N_2} = \begin{bmatrix} -0.6 \\ 0.7 \\ 0.5 \end{bmatrix}$$

Final configuration:

$$g_{final} = \begin{bmatrix} R_{final} & P_{final} \\ \mathbf{0} & 1 \end{bmatrix}, \ with \ R_{final} = R_{Y,\pi/4} R_{Z,\pi/4}, \ and \ P_{final} = \begin{bmatrix} -0.6 \\ 0.7 \\ 0.2 \end{bmatrix}$$

Where R_{rand} can be directly obtained from $F.K(\Theta_{rand})$.

Result:

The computed trajectory has been shown in Figure 4.3. It is noted that the trajectory is mix of straight and curvy lines, which is due to the fact that the user-specified rotation has been changed from configuration to configuration throughout the entire trajectory. Thus, the manipulator has to go in a curvy path to reflect the change in orientation at the end-effector.

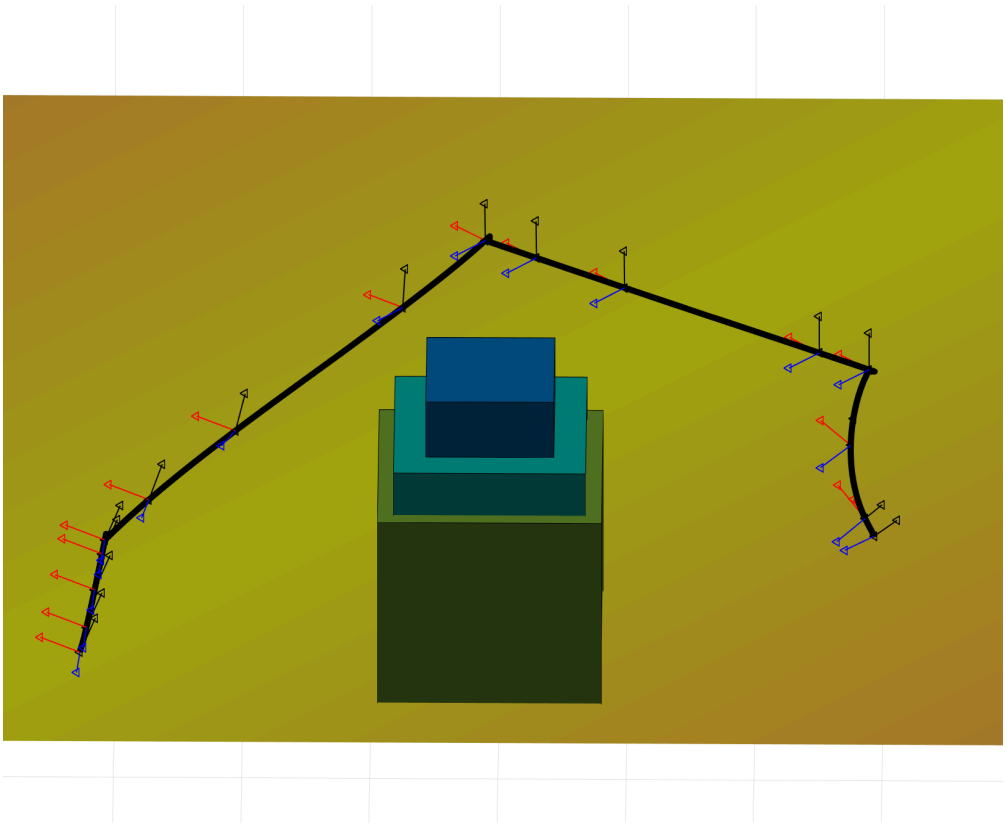
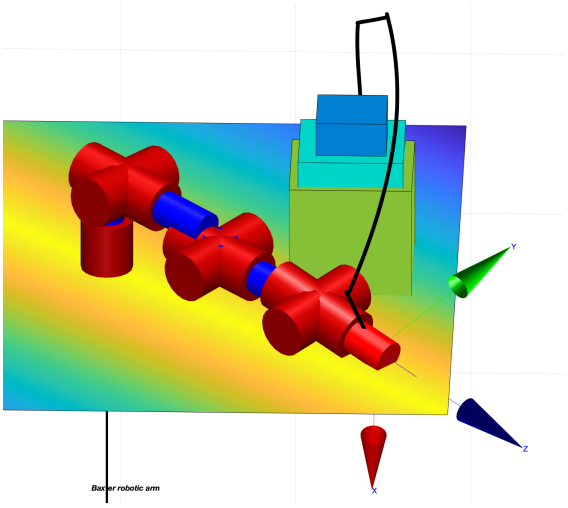


Figure 4.3: Computed Trajectory for Baxter Robotic Arm

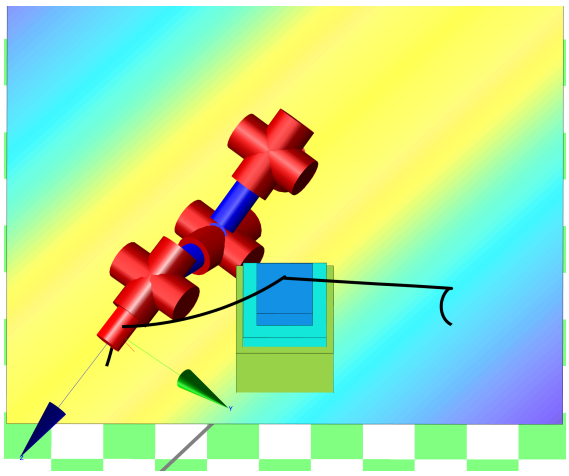
In order to present the results more intuitively, the key moments of the Baxter following the trajectory has been captured in Figure 4.4, and along with the trajectory data reported in Table

	Computational Time	Number of Steps
Reported Data	0.63648s	865

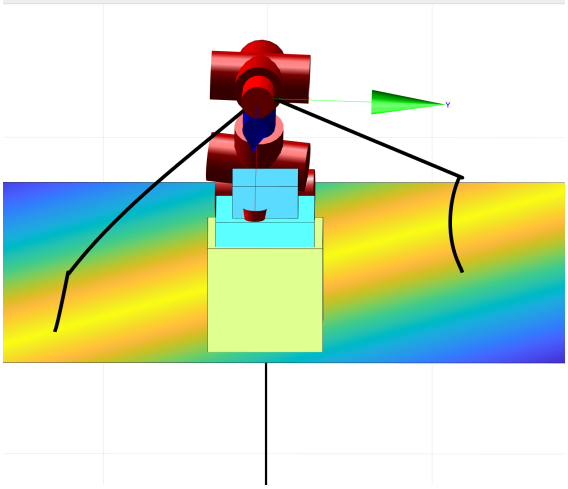
Table 4.2: Trajectory Data of Baxter



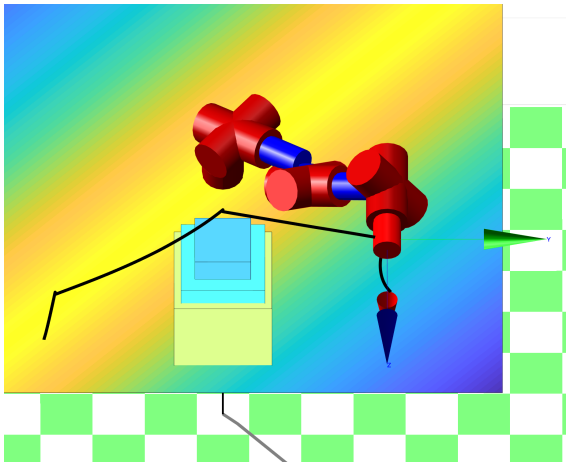
(a) Starts at g_{rand}



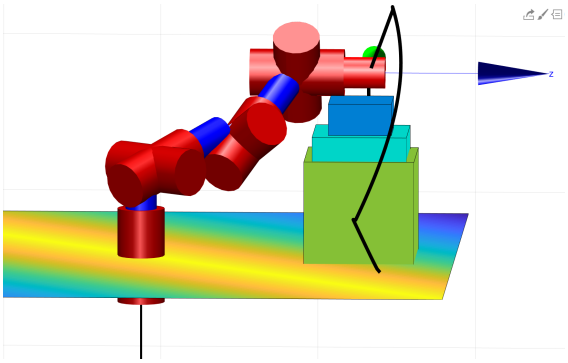
(b) Reaches at g_{init}



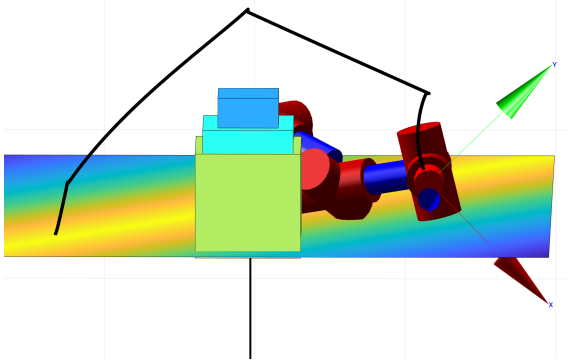
(c) Reaches at g_{N_1}



(d) Reaches at g_{N_2}



(e) g_{N_2} at another angle



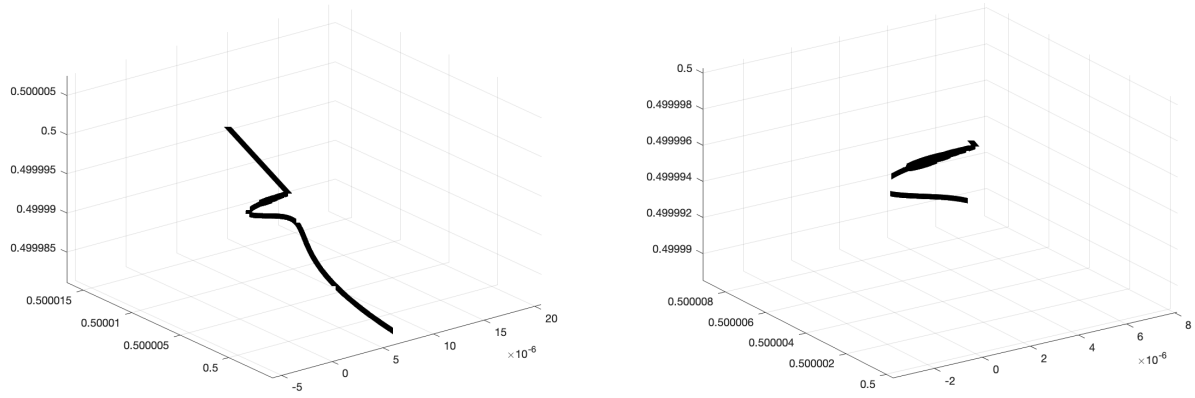
(f) Reaches at g_{final}

Figure 4.4: Key Moments of Baxter Following the Computed Trajectory

The animation can be accessed via the link: Baxter Robotic Arm

4.3 Trajectory Micro-flaw

During the trajectory generation, some micro-flaw and anomaly along the trajectory has been discovered shown in Figure 4.5. It is found that there is some certain micro discontinuity during the moment when the trajectory is in transition to the next configuration. The micro-flaw has been mitigated by adjusting the τ and β dynamically when near the transition.



(a) Broad view (b) Close view

Figure 4.5: Trajectory Micro-flaw along Trajectory

5 Conclusion

In this project, the motion planning algorithm involved in the pick and place problem along with obstacle-avoidance has been successfully implemented for both elbow manipulator and Baxter robotic arm. An obstacle-avoidance solution has been proposed using the method of selecting intermediate configurations. However, the solution evidently has the limitation and only works in a enclosed environment where the positions of the obstacles are already known and remain still. The solution can only prevent the end-effector from collision with the obstacle. The physical hardware of the manipulator still has potential to collide with the obstacle when following the computed trajectory. The future improvement to the solution would be to select a certain amount of points on the surface of the manipulator hardware, and one can obtain the location of those points in motion via forward kinematics $F.K(\Theta)$. Next, one can model a sphere with a proper radius that enclose the entire obstacle. By monitoring the distance between the selected points on the manipulator and the center of the sphere in real-time, a better and more robust obstacle solution can be developed.