Omnidirectional Mobile Platform with a 6-DOF Robot Arm Manipulator in a Pick and Place task

Dimitrios Kavelidis Frantzis

Aristotle University of Thessaloniki / ECE
Robotics Course / Summer Semester Project 2021

Contact: kavelids@ece.auth.gr
23/6/2021

Abstract

Mobile manipulators are widely used robot systems built from a robotic manipulator arm mounted on a mobile platform. Mobile manipulation describes the coordination of the motion of the base and the robot joints to achieve a desired motion at the end effector. These systems have been studied for a long time now and there are already some known motion control and trajectory planning techniques. In this paper, the trajectory planning of an omnidirectional mobile manipulator in a pick and place task is presented, as well as the results of the problem's simulation.

I. Introduction – Problem Description

Initially, our mobile manipulator is on the reference/world frame {0}, meaning that frame {M} of the platform initially has the same position and orientation with {0}. Frame {M} is placed in the center of the lower surface of the platform. The position and orientation of the frame {B} of the base of the arm manipulator is given by p_{MB} and Q_{MB}, where ${p_{MB}}^{(0)} \equiv p_{0B}$ and ${Q_{MB}}^{(0)} \equiv Q_{0B}$. The task is to plan a trajectory to move, pick a cylinder object with {A} frame from a table and place it on the top of the platform, described by frame {F}. The goal is to get to a picking position at 10 seconds, and any collision must be avoided. The manipulator takes joints velocity as input and the platform takes body velocity as input.

Table of numerical data of the problem:

Initial position and orientation of the base (Quaternion)	$p_{MB} = [0.0\ 0.35\ 0.5]^T$ $Q_{MB} = [1\ 0\ 0\ 0]^T$
Initial position and orientation of the object (Quaternion)	$p_{0A} = [1.5\ 1.5\ 0.6]^T$ $Q_{0A} = [1\ 0\ 0\ 0]^T$
Dimensions of the object	$h = 10cm, r = 2.5cm$
Dimensions of the table	$L_a = [0.35\ \ 0.35\ \ 0.6]$
Dimensions of the platform	$L_b = [0.75\ \ 1\ \ 0.5]$
Initial joint positions	
$q_0 = [2.618\ \ -0.6695\ \ 1.2719\ \ 3.1416\ \ 1.2002\ \ -0.9821]^T$	

The Denavit – Hartenberg parameters, the Jacobian matrix and the forward kinematics of the arm are considered fully known, as the model of the 6-DOF arm is produced by the file *lwr_create.p* using Peter Corke's Robotics Toolbox for MATLAB [1].

II. Jacobian of the System

The omnidirectional platform can be modelled as an holonomic 3-DOF robot with two prismatic and one rotational joint, yielding to $\boldsymbol{q_m} = [\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{\varphi_z}]^{\boldsymbol{T}}$ as the configuration of the mobile base. Thus, the homogeneous transformation describing the position and the orientation of the platform in the {0} frame is:

$$T_{0M}(q_m) = \begin{bmatrix} R_z(\varphi_z) & \begin{matrix} x \\ y \\ z \end{matrix} \\ 0\ \ 0\ \ 0 & 1 \end{bmatrix} = \begin{bmatrix} cos\varphi_z & -sin\varphi_z & 0 & x \\ sin\varphi_z & cos\varphi_z & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (1)$$

To derive an equation for controlling the mobile manipulator, it is necessary to find the Jacobian of the whole system. Of course in this case that the base of the arm is moving, the end-effector position and orientation (described in the world frame) also changes. However, the end-effector's Jacobian only depends on the joint configuration $\boldsymbol{q} = [\boldsymbol{q_1}\ \boldsymbol{q_2}\ \boldsymbol{q_3}\ \boldsymbol{q_4}\ \boldsymbol{q_5}\ \boldsymbol{q_6}]^{\boldsymbol{T}}$, not the chassis configuration, since the end effector twist expressed in the end-effector frame is independent of the configuration of the mobile base. Thus, for the end-effector frame {e}, the configuration of {e} in {0} is:

$$\boldsymbol{X(q_m, q) = T_{0e}(q_m, q) = T_{0M}(q_m) \cdot T_{MB} \cdot T_{Be}(q) \in SE(3)}$$

where T_{MB} is a fixed offset of {B} from {M} and $T_{Be}(q)$ is the forward kinematics of the arm, as described in [2].

$$\boldsymbol{T_{MB}} = \begin{bmatrix} 1 & 0 & 0 & p_{MBx} \\ 0 & 1 & 0 & p_{MBy} \\ 0 & 0 & 1 & p_{MBz} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.35 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} (2)$$

To apply kinematic control of the end-effector frame using the wheel and joint velocities, we need the Jacobian $J_e(q) \in \mathbb{R}^{6x(m+n)}$ satisfying:

$$V_e = J_e(q)\begin{bmatrix} \dot{q_m} \\ \dot{q} \end{bmatrix} = [J_P(q) \;\; J_{Arm}(q)]\begin{bmatrix} \dot{q_m} \\ \dot{q} \end{bmatrix}, \qquad (3)$$

where m = 3 and n = 6 (number of joints of the platform and of the arm).

As it can be seen above, the Jacobian of the system $J_e(q)$ can be partitioned into $J_P(q) \in \mathbb{R}^{6xm}$ and $J_{Arm}(q) \in \mathbb{R}^{6xn}$. The contribution of the platform velocities $\dot{q_m}$ to the end-effector's velocity is expressed through the term $J_P(q) \cdot \dot{q_m}$, while the contribution of the joint velocities to the end-effector's velocity is expressed through the term $J_{Arm}(q)\dot{q}$.
As mentioned before, $J_{Arm}(q)$ is the body Jacobian of the manipulator and thus, it is considered known. Therefore, $J_P(q)$ must be found. That means that for the Jacobian of the whole system:

$$V_e = J(q, q_m)\begin{bmatrix} \dot{q_m} \\ \dot{q} \end{bmatrix} \text{ with } J(q, q_m) = [J_p(q_m) \;\; J_{Arm}(q)]$$

The 3D kinematics of the platform can be obtained from (1) and (2):

$$\boldsymbol{x = f_p(q_m)} = \begin{bmatrix} x + p_{MBx}cos\varphi_z - p_{MBy}sin\varphi_z \\ y + p_{MBx}sin\varphi_z + p_{MBy}cos\varphi_z \\ p_{MBz} \\ 0 \\ 0 \\ \varphi_z \end{bmatrix}$$

Therefore, the Jacobian $J_p(q_m)$:

$$J_p(q_m) = \frac{\partial f_p(q_m)}{\partial q_m} = \begin{bmatrix} 1 & 0 & -p_{MBx}cos\varphi_z - p_{MBy}sin\varphi_z \\ 0 & 1 & p_{MBx}sin\varphi_z - p_{MBy}cos\varphi_z \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

III. Inverse Kinematics

To solve the inverse kinematics we need an estimation of the solution of $\boldsymbol{q = f^{-1}(x)}$. Starting with a rough estimation $\boldsymbol{q_k}$, joint coordinates nearer to the final solution can be obtained by $\boldsymbol{q_{k+1} = q_k + J^{-1}(q_k) \cdot (x - x_k)}$ where $\boldsymbol{x_k = f(q_k)}$.
The iteration is aborted when the error falls below a predefined limit:

$$\boldsymbol{|x - x_k| < \varepsilon}$$

IV. Trajectory Planning - Theory

Having the complete Jacobian of the system, the desired end-effector trajectory $\boldsymbol{X(t)}$ can be tracked using numerical inverse kinematics. Therefore, the desired velocities for such trajectory can be found using:

$$\begin{bmatrix} \dot{q_m} \\ \dot{q} \end{bmatrix} = J^{\dagger}(q, q_m)V, \qquad (4)$$

where $J^{\dagger}(q, q_m)$ is the pseudoinverse of $J(q, q_m)$.

To do the required pick and place task, we can choose to plan the trajectory either in the Task Space or the Joint Space. This assignment is about a cartesian trajectory of the end-effector (Task Space) and the calculation of the reference velocities of the joints of the arm and the platform (which is the input) needed to implement this trajectory in the Joint Space. However, for a thorough analysis, trajectories will be fully planned in both Spaces, to give the freedom to take any combination of each parts' trajectories and plan a mixed trajectory along the mid-points if we want to. Therefore, before getting into practice, some theory for these spaces should be included:

i) For a trajectory in the **Task Space** (also known as *Cartesian Space*), the path planning is for the end-effector. Inverse kinematics are needed in every time step in order to calculate a point-to-point trajectory, making the planning computationally expensive, but preferable if we plan with obstacle avoidance purposes. Since the V_e has 6 elements and $q_e = [q_m \; q]^T$ (q_e for extended joint vector) is a 9-element vector, the use of pseudoinverse of Jacobian is necessary creating the need for a control law in the trajectory planning and tracking. However, in our case we have some specified constraints that allow us to use the mobile manipulator without a control law by separating the motions to platform-only-motion and arm-only-motion by giving the appropriate inputs-reference velocities $u = \begin{bmatrix} \dot{q_m} \\ \dot{q} \end{bmatrix}$. When planning cartesian trajectory in the Task Space, the Cartesian Interpolation is often used to interpolate between two poses:

$$\xi_0 \sim (\boldsymbol{R_0, t_0}) \rightarrow \xi_1 \sim (\boldsymbol{R_1, t_1})$$

Poses have a translational and a rotational component we need to interpolate separately. The state-of-the-art method to do this is by converting the rotation matrix of each pose into a Unit Quaternion $\mathring{q}$ for efficient calculations:

$$\xi_0 \sim \left(\mathring{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{t_0}\right) \rightarrow \xi_1 \sim \left(\mathring{\boldsymbol{q}}_{\boldsymbol{1}}, \boldsymbol{t_1}\right)$$

Then, the interpolation of the translation using the linear interpolation that is widely used in robotics:

$$\boldsymbol{p(s) = (1 - s)p_0 + s \cdot p_1}, \qquad \boldsymbol{p(s)} = \begin{bmatrix} x(s) \\ y(s) \\ z(s) \end{bmatrix} \boldsymbol{(5)}$$

where $\boldsymbol{s(t) \epsilon [0,1]}$ and $\boldsymbol{t \epsilon [0, T]}$ and $\boldsymbol{p_0, p_1}$ are the initial and final position respectively, while $\boldsymbol{s(t)}$ is a function of time varies smoothly from 0 to 1 over the duration (T) of the trajectory. Often, $\boldsymbol{s(t)}$ is a quintic polynomial and will be used in this assignment. To interpolate rotation, quaternion interpolation also known as **SLERP** (Spherical Linear Interpolation) is used as it is considered the shortest path between two rotations **[3]**. SLERP also uses the same function $\boldsymbol{s(t)}$:

$$\mathring{q}(s) = \frac{sin((1 - s)\boldsymbol{\theta})\mathring{q}_0 + sin(s\boldsymbol{\theta})\mathring{q}_1}{sin\theta}$$

Thus, we can convert the interpolated quaternion back to a rotation matrix, combine it with the interpolated translation and rebuild a homogeneous transformation matrix. Then we

use the inverse kinematics with an estimation as described in III. using this homogeneous transformation matrix to get the joint positions and move on to the next step using Euler's integration. Note that for the estimation of the inverse kinematics the ***Robot.teach*** method of the toolbox has been used to get the approximation of the angle positions (to avoid collisions e.g. elbow-down position at picking) Next it is easy to derive V from the desired trajectory and thus, obtain the reference velocities needed for the trajectory from (4). As mentioned above, the reference velocities are inputs for the system and therefore we can choose them along an appropriate control law to track a desired trajectory.

ii) For a trajectory in the **Joint Space**, the path planning is for the joints of the system. In our case, since q_e is a vector with length = 9, then 9 trajectories must be planned. In this case, the inverse kinematics are only needed for the mid-points of the trajectory, and thus is by far faster. Also, the motion is smoother than in a simple trajectory for the end effector. However, a trajectory in the Joint Space only let us have control of the end-effector position in specific points and not in a point-to-point trajectory, and the motion is not the optimal trajectory (fastest, shortest path, etc.). In this assignment, quintic polynomials have been used for the trajectory planning.

V. Trajectory Planning – Practice / Case Study

First, we need to set and analyze the working environment in terms of distances and orientations of objects on this specific problem. Our mobile manipulator and the table with the object on are shown on Figure 1 and 2 in top and side view respectively:

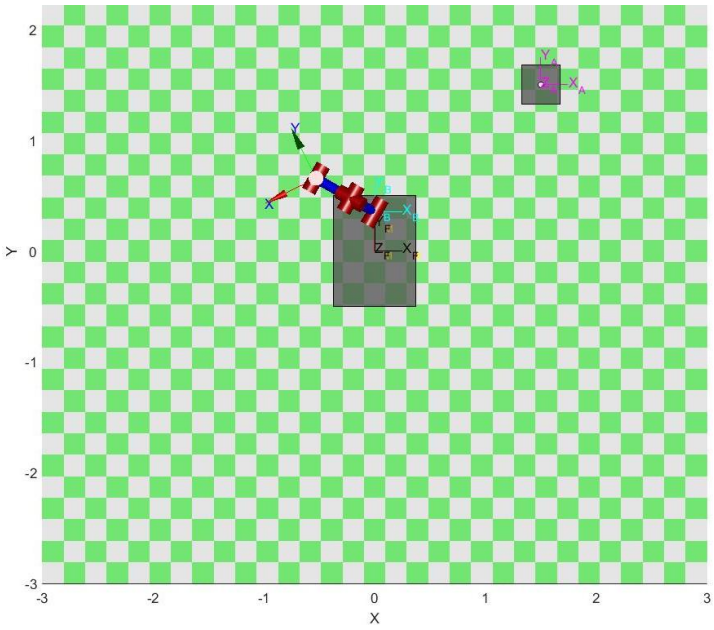

Figure 1 - Working Environment Initially / Top View

For this specific pick and place task it is required to go through 2 different motions:

1) First motion description: Moving the platform to get close to the target, without any motion of the arm and then stopping the platform at this point. Choosing to stop the manipulator right in front of the object (same displacement from the {0} frame on the x axis) and a safe distance $\boldsymbol{d_{safe} = 10cm}$ between the table and the platform, we ensure that no collision is going to happen between the robot system and the target/table body, as it can be seen in Figures 3,4.

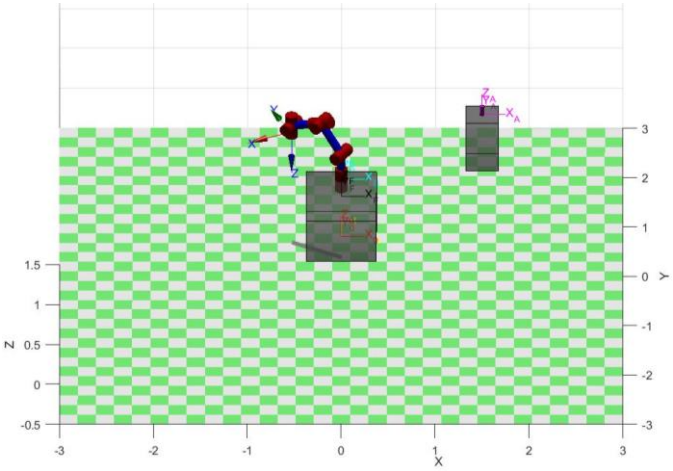

Figure 2 - Working Environment Initially / Side View

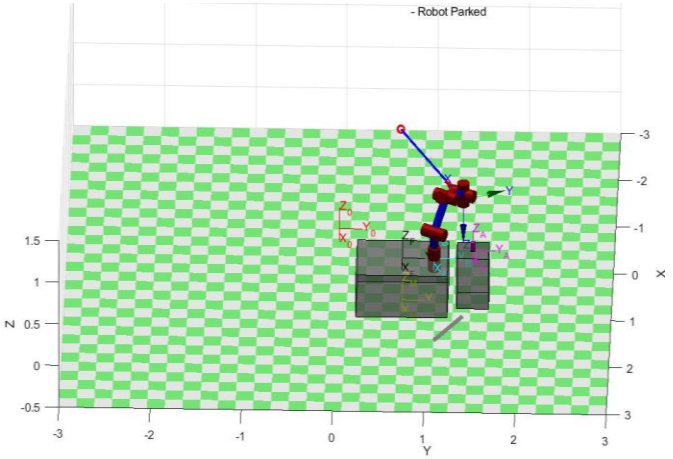

Figure 3 - Working Environment after 1st motion / Top View

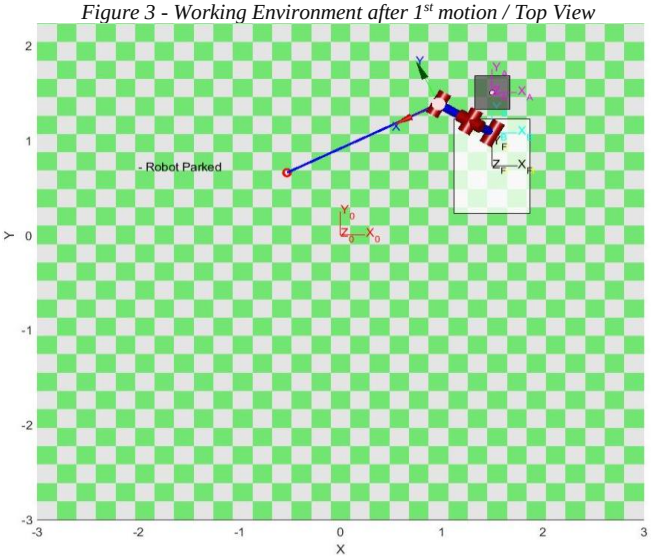

Figure 4 - Working Environment after 1st motion / Side View

2) Second motion description: Without any movement of the platform, the arm performs the precise pick and place task. This is a motion with many midpoints.

A. *Trajectory Planning in the Task Space*

Taking into consideration the restrictions, we can plan these 2 trajectories taking for granted that in the first part/motion the reference velocity of the joints $\dot{q}$ is going to be zero (since we want the end effector's orientation to stay the same for this motion) and therefore no motion of the arm joints. The same is applied for the second part/motion and the reference velocity $\dot{q_m}$ of the platform ($\dot{q_m} = 0$ in the second motion, as the platform stays still during the manipulation). Thus, for the first part we choose $\dot{q} = 0$ as input of the arm and design a cartesian trajectory for the end effector, and then use the V_e velocity to calculate the reference velocities $\dot{q_m}$ for the platform using the pseudoinverse of J_p. In every step, we update the platform's base to follow along the platform, using the homogeneous transformation:

$$\boldsymbol{T_{0B} = T_{0M}(q_m) \cdot T_{MB}}$$

Also, we can choose $\boldsymbol{\dot{\varphi}_z = 0}$ because we want the orientation of the end-effector to remain the same. However this is not completely necessary as we will obtain this $\boldsymbol{\dot{\varphi}_z = 0}$ result from the interpolation of the poses.

Accordingly, we choose $\dot{q_m} = 0$ in the second motion and obtain the reference velocities of joints $\dot{q}$ with J_a^{-1}. Note that having 6 – DOF arm manipulator, the inverse of the Jacobian can be used without any issue as long as the trajectory is not passing through singularity points.

The trajectories for the first part are pretty straightforward. We work only in the 2 dimensions x,y. We calculate the position of the platform in the desired position (Point 1) as displayed in Figure 3, and then obtain the coordinates of the end-effector using forward kinematics (after updating the base of the robot to the new position). The trajectory is the linear interpolation of the positions (as the orientation stays the same), using a quintic polynomial for $\boldsymbol{s(t)}$.
Then, having the platform in front of the target, we plan the trajectory with 6 steps:

2.1) Moving the arm at a safe distance (0.3m) above the target to a **ready-to-pick** position (Point 2.1).
Zero velocity is assumed at the start and the end point.
2.2) Moving the arm down to **pick** the object (Point 2.2)
Zero velocity is assumed at the start and the end point.
2.3) Moving the arm **up** again at the same height as step i) (Point 2.3).
Zero velocity is assumed at the start and the end point.
2.4) Moving the arm to a **mid-point** at the same height to avoid collision, with a motion to the + x direction. (Point 2.4).
Zero velocity is assumed at the start and the end point.
2.5) Moving the arm at a safe distance (0.3m) above the {F} frame to a **ready-to-place** position (Point 2.5).
Zero velocity is assumed at the start and the end point.
2.6) **Placing** the object to the {F} frame, by moving down the arm (Point 2.6).
Zero velocity is assumed at the start and the end point.

Note: The motion described from steps 2.3)-2.5) as well as (or even 2.3)-2.6)) could be also planned with linear segments with parabolic blends (LSPB), to make a smooth trajectory, however for simplicity, simple quintic polynomials are used, along with their usual solutions.

$$\boldsymbol{s(t) = k_0 + k_1 t + k_2 t^2 + k_3 t^3 + k_4 t^4 + k_5 t^5, \qquad t \epsilon [0, T_i]}$$

For each of the mid-points, the desired homogeneous transformations matrices of the end effector (with respect to the world frame {0}) used are provided below:

$\mathbf{T_{0e_0}} =$

-0.8967	-0.4426	0.0000	-0.5298
-0.4426	0.8967	-0.0000	0.6559
0.0000	-0.0000	-1.0000	0.9049
0	0	0	1.0000

$\mathbf{T_{0e_1}} =$

-0.8967	-0.4426	0.0000	0.9702
-0.4426	0.8967	-0.0000	1.3809
0.0000	-0.0000	-1.0000	0.9049
0	0	0	1.0000

$\mathbf{T_{0e_{21}}} =$

-0.8967	-0.4426	0.0000	1.5000
-0.4426	0.8967	-0.0000	1.5000
0.0000	-0.0000	-1.0000	0.9000
0	0	0	1.0000

$\mathbf{T_{0e_{22}}} =$

-0.8967	-0.4426	0.0000	1.5000
-0.4426	0.8967	-0.0000	1.5000
0.0000	-0.0000	-1.0000	0.9000
0	0	0	1.0000

These homogeneous matrices describe the via points for the required motion from point 0 to the point the robot is in picking position. For the rest of the motion:

$\mathbf{T_{0e_{23}} = T_{0e_{21}}}$

$\mathbf{T_{0e_{24}}} = \begin{bmatrix} R_x(-\pi) & p_{24} \\ \mathbf{0} & 1 \end{bmatrix} =$

1.0000	0	0	1.9250
0	-1.0000	0	1.0750
0	0	-1.0000	0.9000
0	0	0	1.0000

$\mathbf{T_{0e_{25}}} = \begin{bmatrix} R_x(-\pi)R_z(-\pi/2) & p_{0F} + 0.3\bar{z} \\ \mathbf{0} & 1 \end{bmatrix} =$

-0.0000	1.0000	0	1.5000
1.0000	0.0000	0	0.7250
0	0	-1.0000	0.8000
0	0	0	1.0000

$\mathbf{T_{0e_{26}}} = \begin{bmatrix} R_x(-\pi)R_z(-\pi/2) & p_{0F} \\ \mathbf{0} & 1 \end{bmatrix} =$

-0.0000	1.0000	0	1.5000
1.0000	0.0000	0	0.7250
0	0	-1.0000	0.5000
0	0	0	1.0000

Moreover, for each of the trajectories the desired initial and final velocities' and accelerations' restrictions are:

$$\boldsymbol{s(t_0) = s_0 = 0, \quad \dot{s}(t_0) = \dot{s}_0 = 0, \quad \ddot{s}(t_0) = \ddot{s}_0 = 0}$$
$$\boldsymbol{s(T_i) = s_f = 1, \quad \dot{s}(T_i) = \dot{s}_f = 0, \quad \ddot{s}(T_i) = \ddot{s}_f = 0}$$

To plan a trajectory from pose to pose, we interpolate positions and Quaternions (orientations).

and therefore the coefficients for the quintic polynomials used to interpolate between these positions are (in matrix form):

Coeff / Trajectory	$\mathbf{k_0}$	$\mathbf{k_1}$	$\mathbf{k_2}$	$\mathbf{k_3}$	$\mathbf{k_4}$	$\mathbf{k_5}$	T_i
Traj. 1	0	0	0	0.0800	-0.0240	0.0019	5
Traj. 2.1	0	0	0	0.6400	-0.3840	0.0614	2.5
Traj. 2.2	0	0	0	0.6400	-0.3840	0.0614	2.5
Traj. 2.3	0	0	0	0.6400	-0.3840	0.0614	2.5
Traj. 2.4	0	0	0	0.3704	-0.1852	0.0247	3
Traj. 2.5	0	0	0	0.3704	-0.1852	0.0247	3
Traj. 2.6	0	0	0	2.963	-2.963	0.7901	1.5

Table 1 – Trajectory coefficients of s(t) and time durations for Task Space Trajectory Planning

For the trajectories of velocities we take the derivative of $\boldsymbol{s(t)}$:

$$\dot{s}(t) = k_1 + 2k_2 t + 3k_3 t^2 + 4k_4 t^3 + 5k_5 t^4$$

Setting $s = \sigma/L, \sigma \in [0, L]$ where L is the arc length, meaning $L = ||(\mathbf{p_1} - \mathbf{p_0}||$ gives the current length of the path, so the unit vector $\frac{\mathbf{p_1}-\mathbf{p_0}}{||(\mathbf{p_1}-\mathbf{p_0}||}$ of directional cosines of the line appears in the derivative of the interpolated translation:

$$\dot{\boldsymbol{p}}(\mathbf{s}) = \frac{\partial \boldsymbol{p}}{\partial \boldsymbol{s}}\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{t}} = (\mathbf{p_1} - \mathbf{p_0})\dot{s} = \frac{\mathbf{p_1}-\mathbf{p_0}}{\mathbf{L}}\dot{\sigma}$$

The coefficients for $\dot{\boldsymbol{s}}(\mathbf{t})$ can be derived from Table 1 and therefore obtain the cartesian trajectory of velocities $\dot{\boldsymbol{p}}(\mathbf{s})$.

The choice of time durations is this to give some time for the platform motion (5s), then 2.5s to get to a ready-to-pick position, then 2.5s to get to picking position and achieve the 10 seconds goal for picking. Then another 2.5s to get the arm up, 6s to turn it (3+3) to a placing position and finally 1.5s for a cautious placing.

B. Trajectory Planning in the Joint Space

The trajectory planning in the Joint Space is more analytical since we specify the mid-points positions for every joint of the system. Note that the end effector of the robot now does not follow a straight line as it does in the Task Space. Now the extended vector of joint positions $q_e = [q_m\ q]^T$ is used to interpolate between positions, but instead of using $\boldsymbol{s(t)}$, we rather use the quintic polynomial for the trajectory of each joint position itself with normalized time (same as having a path of $\boldsymbol{s(t)}$ as described before, since with the correct restrictions it lead to the interpolation between the first and the last position):

$$q_{di}(\tau) = k_0 + k_1\tau + k_2\tau^2 + k_3\tau^3 + k_4\tau^4 + k_5\tau^5, t\epsilon[0, T_i]$$
$$\dot{q}_{di}(\tau) = k_1 + 2k_2\tau + 3k_3\tau^2 + 4k_4\tau^3 + 5k_5\tau^4, \qquad \tau = t/T_i$$

and applying the restrictions accordingly:

$$\mathbf{q_{di}(\tau_0) = q_{i0}}, \quad \dot{\mathbf{q}}_{\mathbf{di}}(\boldsymbol{\tau_0}) = \dot{\boldsymbol{q}}_{\mathbf{i0}} = \mathbf{0}, \quad \ddot{\mathbf{q}}_{\mathbf{di}}(\boldsymbol{\tau_0}) = \ddot{\boldsymbol{q}}_{\mathbf{i0}} = \mathbf{0}$$
$$\mathbf{q_{di}}(T_i) = \mathbf{q_{if}}, \quad \dot{\mathbf{q}}_{\mathbf{di}}(T_i) = \dot{\boldsymbol{q}}_{\mathbf{if}} = \mathbf{0}, \quad \ddot{\mathbf{q}}_{\mathbf{di}}(T_i) = \ddot{\boldsymbol{q}}_{\mathbf{if}} = \mathbf{0}$$

Thus, we since we have 9 joints for the robot system, we need 9 trajectories (one for each joint) for each part of motion. To get the via points (mid-points) for the whole trajectory, we calculate the desired poses/homogeneous transform and get the inverse kinematics to find the joint positions.

For the Trajectory 1 (first motion), the choice of the arrival position is pretty straightforward:

$$\mathbf{q_f} = [\mathbf{D_x D_y}\ \mathbf{0}\ \ \mathbf{q_0}^T]$$

Where $\mathbf{D_x}$ and $\mathbf{D_y}$ are the displacements of the platform, in the position where the platform is in front of the target.

Then we easily compute:

$$D_x = p_{0A}(1)$$
$$D_y = p_{0A}(2) - d_{safe} - L_{ay}/2 - L_{by}/2$$

And $\varphi_z = 0$ (optionally) because we assume that there is no change in the orientation of the end-effector. Nevertheless, $\dot{\boldsymbol{\varphi}}_{\boldsymbol{z}} = 0$ will occur, resulting to $\varphi_z = 0$ for the whole trajectory. Taking the forward kinematics and we get $\mathbf{T_{0e_1}}$.

For the second part of the motion where only the manipulator moves, we choose the same homogeneous transformation matrices $\boldsymbol{T_{0e_1}, T_{0e_{21}}, T_{0e_{22}}, T_{0e_{23}}, T_{0e_{25}}, T_{0e_{26}}}$.

However, since the interpolation is now been performed in rotations, the previous collision-avoidance by adding a pose described by $\boldsymbol{T_{0e_{24}}}$, is now not necessary. Nevertheless, we choose a mid-point corresponding to $\boldsymbol{T_{0e_{24}}'}$ that is now different. Specifically, we choose the position/translation of $\boldsymbol{T_{0e_{24}}'}$ to be the antipodal point of end-effector on the 2d circle with center at the position of the arm base while only the first joint moves. This is:

$$\boldsymbol{T_{0e_{24}}'} = \begin{matrix} 0.0000 & 1.0000 & 0.0000 & 1.5000 \\ 1.0000 & -0.0000 & -0.0000 & 0.6500 \\ 0.0000 & 0.0000 & -1.0000 & 0.9000 \\ 0 & 0 & 0 & 1.0000 \end{matrix}$$

On each step, we use forward kinematics to perform all the calculations needed and get the end-effector's coordinates. Note that even though interpolation leads to a path between two points, the path is different between the Joint Space and the Task Space (arc -vs- straight line respectively), and this is why reaching an antipodal point without a via point is feasible in the Joint Space but not in the Task Space. Since the trajectory planning in the Joint Space is only for thoroughness, the choice of time durations T_i : [3,1.5,1,1,1,1,1.5], to complete the whole pick and place operation in 10 seconds. Note that in this trajectory, the restriction for the same orientation of the end-effector throughout the whole operation is not fulfilled in these trajectories in the Joint Space, but it is rather simply a demonstration of such a planning in this Space.

Considering all things mentioned above, the trajectories' coefficient choices for each part are:

Trajectory 1: Initial Point → Point 1						
Coeff / Joint	$\mathbf{k_0}$	$\mathbf{k_1}$	$\mathbf{k_2}$	$\mathbf{k_3}$	$\mathbf{k_4}$	$\mathbf{k_5}$
Joint 1 P-x	0	0	0	15	-22.5	9
Joint 2 P-y	0	0	0	7.25	-10.875	4.35
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	2.618	0	0	0	0	0
Joint 5 R-q_2	-0.6695	0	0	0	0	0
Joint 6 R-q_3	1.2719	0	0	0	0	0
Joint 7 R-q_4	3.1416	0	0	0	0	0
Joint 8 R-q_5	1.2002	0	0	0	0	0
Joint 9 R-q_6	-0.9821	0	0	0	0	0

Table 2 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Initial Point → Point 1

Trajectory 2.1: Point 1 → Point 2.1						
Joint \ Coeff	k_0	k_1	k_2	k_3	k_4	k_5
Joint 1 P-x	1.5	0	0	0	0	0
Joint 2 P-y	0.725	0	0	0	0	0
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	2.618	0	0	-10.472	15.7081	-6.2832
Joint 5 R-q_2	-0.6695	0	0	4.106	-6.1589	2.4636
Joint 6 R-q_3	1.2719	0	0	6.3676	-9.5514	3.8205
Joint 7 R-q_4	3.1416	0	0	-31.416	47.124	-18.8496
Joint 8 R-q_5	1.2002	0	0	-21.7423	32.6135	-13.0454
Joint 9 R-q_6	-0.9821	0	0	17.6750	-26.5125	10.6050

Table 3 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Point 1 → Point 2.1

Trajectory 2.2: Point 2.1 → Point 2.2						
Joint \ Coeff	k_0	k_1	k_2	k_3	k_4	k_5
Joint 1 P-x	1.5	0	0	0	0	0
Joint 2 P-y	0.725	0	0	0	0	0
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	1.5708	0	0	0	0	0
Joint 5 R-q_2	-0.2589	0	0	-6.6039	9.9058	-3.9623
Joint 6 R-q_3	1.9087	0	0	0.3590	-0.5386	0.2154
Joint 7 R-q_4	0	0	0	0	0	0
Joint 8 R-q_5	-0.974	0	0	6.9629	-10.4443	4.1777
Joint 9 R-q_6	0.7854	0	0	0	0	0

Table 4 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Point 2.1 → Point 2.2

Trajectory 2.3: Point 2.2 → Point 2.3						
Joint \ Coeff	k_0	k_1	k_2	k_3	k_4	k_5
Joint 1 P-x	1.5	0	0	0	0	0
Joint 2 P-y	0.725	0	0	0	0	0
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	1.5708	0	0	0	0	0
Joint 5 R-q_2	-0.9193	0	0	6.6039	-9.9058	3.9623
Joint 6 R-q_3	1.9446	0	0	-0.3590	0.5386	-0.2154
Joint 7 R-q_4	0	0	0	0	0	0
Joint 8 R-q_5	-0.2777	0	0	-6.9629	10.4443	-4.1777
Joint 9 R-q_6	0.7854	0	0	0	0	0

Table 5 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Point 2.2 → Point 2.3

Trajectory 2.4: Point 2.3 → Point 2.4						
Joint \ Coeff	k_0	k_1	k_2	k_3	k_4	k_5
Joint 1 P-x	1.5	0	0	0	0	0
Joint 2 P-y	0.725	0	0	0	0	0
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	1.5708	0	0	-31.416	47.124	-18.8496
Joint 5 R-q_2	-0.2589	0	0	0	0	0
Joint 6 R-q_3	1.9087	0	0	0	0	0
Joint 7 R-q_4	0	0	0	0	0	0
Joint 8 R-q_5	-0.974	0	0	0	0	0
Joint 9 R-q_6	0.7854	0	0	-7.854	11.781	-4.7124

Table 6 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Point 2.3 → Point 2.4

Trajectory 2.5: Point 2.4 → Point 2.5						
Joint \ Coeff	k_0	k_1	k_2	k_3	k_4	k_5
Joint 1 P-x	1.5	0	0	0	0	0
Joint 2 P-y	0.725	0	0	0	0	0
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	-1.5708	0	0	0	0	0
Joint 5 R-q_2	-0.2589	0	0	-0.4357	0.6536	-0.2614
Joint 6 R-q_3	1.9087	0	0	2.9703	-4.4554	1.7822
Joint 7 R-q_4	0	0	0	0	0	0
Joint 8 R-q_5	-0.974	0	0	3.4060	-5.1090	2.0436
Joint 9 R-q_6	0	0	0	0	0	0

Table 7 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Point 2.4 → Point 2.5

Trajectory 2.6: Point 2.5 → Point 2.6						
Joint \ Coeff	k_0	k_1	k_2	k_3	k_4	k_5
Joint 1 P-x	1.5	0	0	0	0	0
Joint 2 P-y	0.725	0	0	0	0	0
Joint 3 R-φ_z	0	0	0	0	0	0
Joint 4 R-q_1	-1.5708	0	0	0	0	0
Joint 5 R-q_2	-0.3025	0	0	-8.6473	12.9709	-5.1884
Joint 6 R-q_3	2.2057	0	0	-1.8548	2.7822	-1.1129
Joint 7 R-q_4	0	0	0	0	0	0
Joint 8 R-q_5	-0.6334	0	0	6.7925	-10.1887	4.0755
Joint 9 R-q_6	0	0	0	0	0	0

Table 8 – Trajectory coefficients for Joint Space Trajectory Planning / Motion Point 2.5 → Point 2.6

It is obvious again that coefficients for the trajectories of velocities are derived from Tables 2-8 since we know the values of $\mathbf{k_1, k_2, k_3, k_4, k_5}$.

VI. Results / Plots – Task Space Trajectories

The simulation of the robot and the plots of the results are produced with MATLAB 2018a scripts. As we can see below, there are some discontinuities in the motion of joints in the plots indicating that in real conditions, the actual system will not be able to respond with these instant changes. In real conditions we would rather use some control law on the dynamic equations of the system to track the desired trajectory and compute the needed torques for each joint needed for the operation. In this assignment, the desired trajectories are assumed to be the real trajectories and the simulation is done without the use of a control law, justifying the existence of the discontinuities in the plots.

1) Plots of Positions:

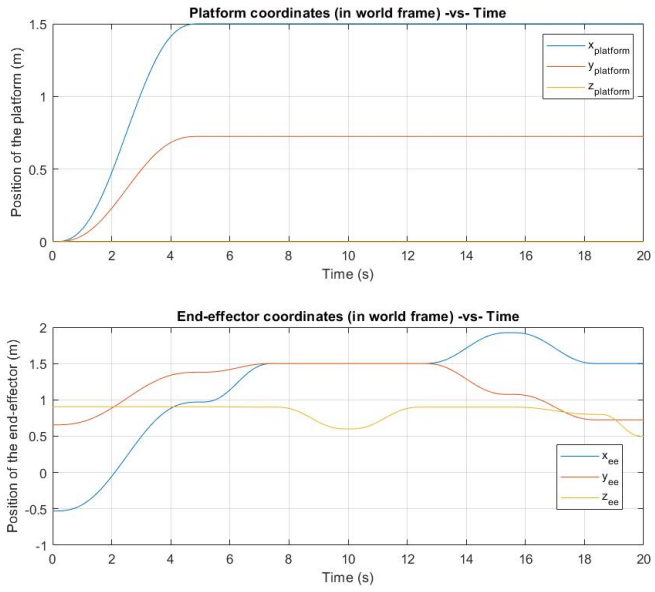

Figure 5 – End-effector's and platform position -vs- Time (TS)

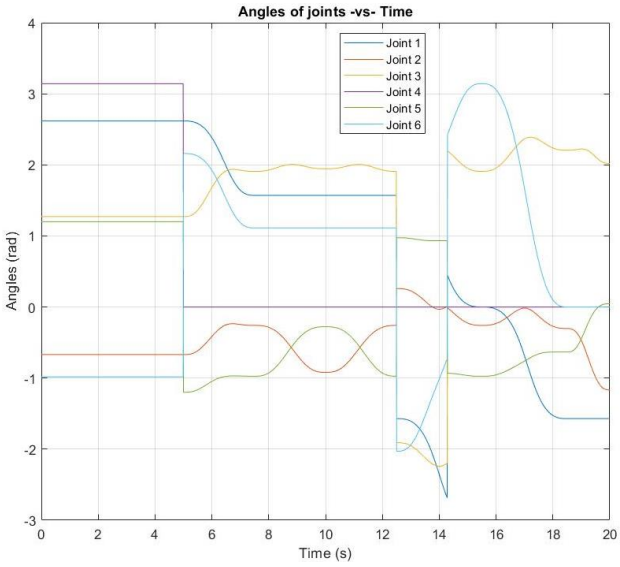

Figure 6 – Joints' positions (Angles) -vs- Time (TS)

3) RPY angles and Quaternion's theta angle / Orientation of the end-effector -vs- time:

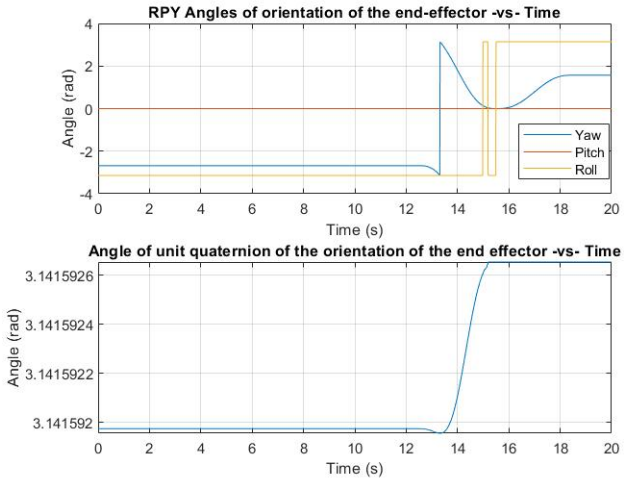

Figure 8 – End-effector's orientation -vs- Time (TS)

2) 3D Representation of End-Effector' and Platform's trajectories:

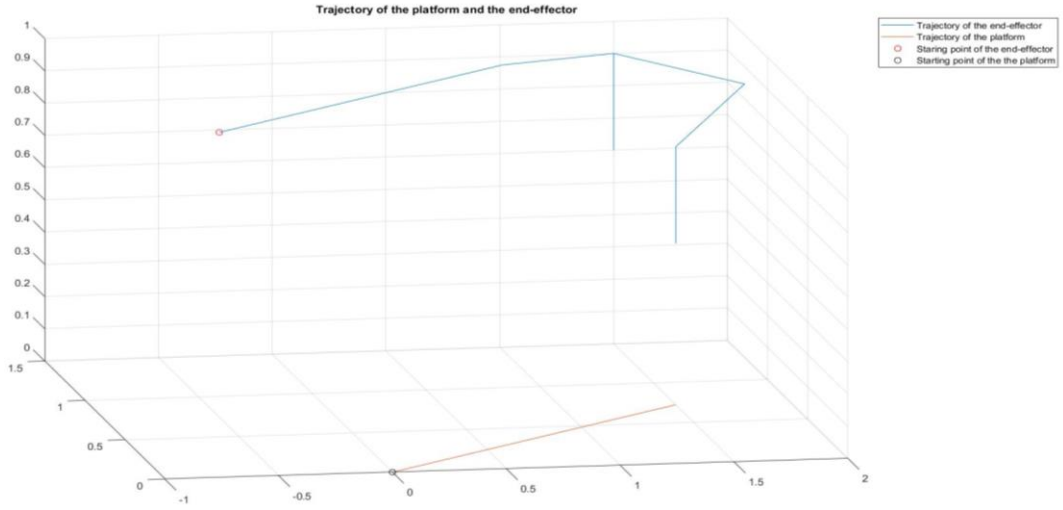

Figure 7 – 3D Representation of End-effector' and Platform's trajectories (TS)

5) Velocities:

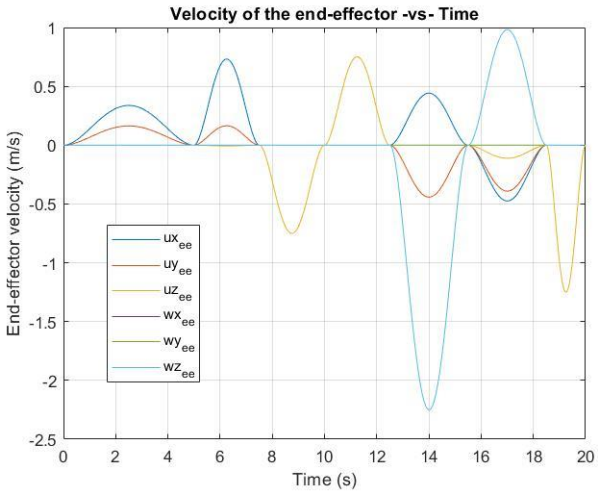

Figure 9 – Velocities of the End-effector -vs- Time (TS)

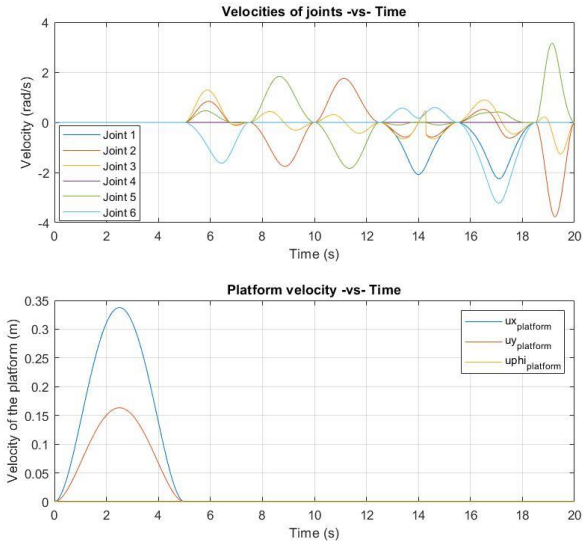

Figure 10 – Velocities of the Joints/Platform -vs- Time (TS)

6) Side view of the simulated trajectory:

Note: To clearly show the placed object, the robot arm is finally plotted in the initial joints q_0

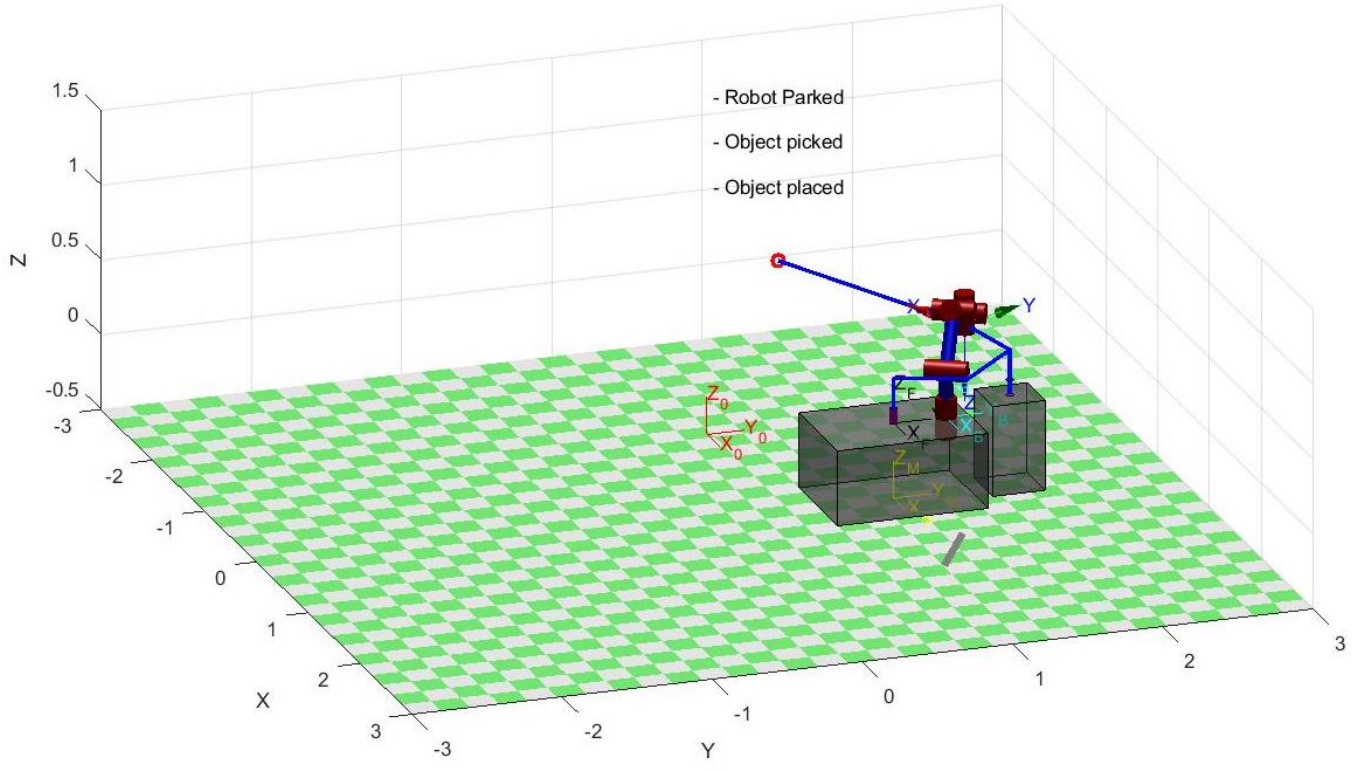

Figure 11 – Plotted Trajectory for the end-effector in the simulation environment (TS)

VII. Results / Plots – Joint Space Trajectories

The results for the Joint Space Trajectories are shown below. As we can see now, the positions of the joints are continuous.

1) *Positions:*

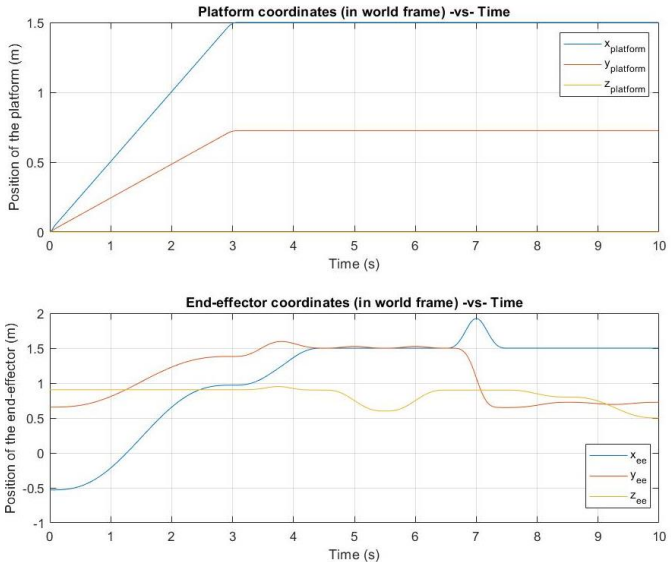

Figure 12 – End-effector's and platform position -vs- Time (JS)

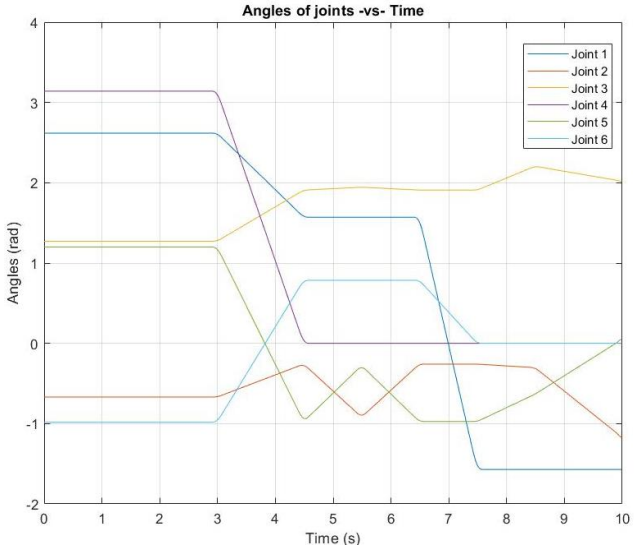

Figure 13 – Joint's Positions (Angles) -vs- Time (JS)

3) *RPY angles and Quaternion's theta angle / Orientation of the end-effector -vs- time:*

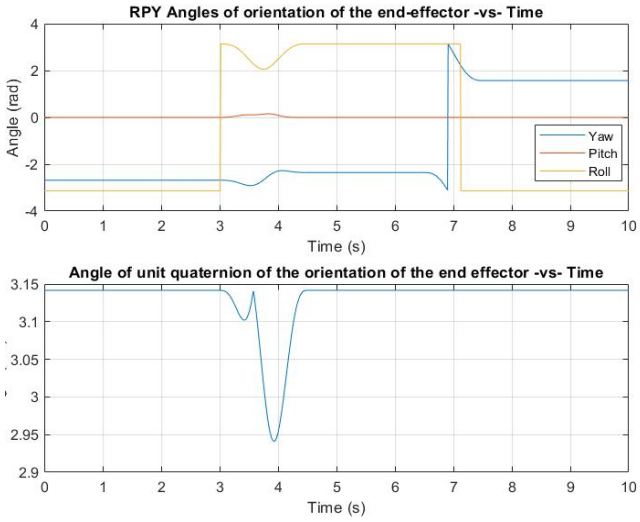

Figure 15 - End-effector's orientation -vs- Time (TS)

2) *3D Representation of End-Effector' and Platform's trajectories:*

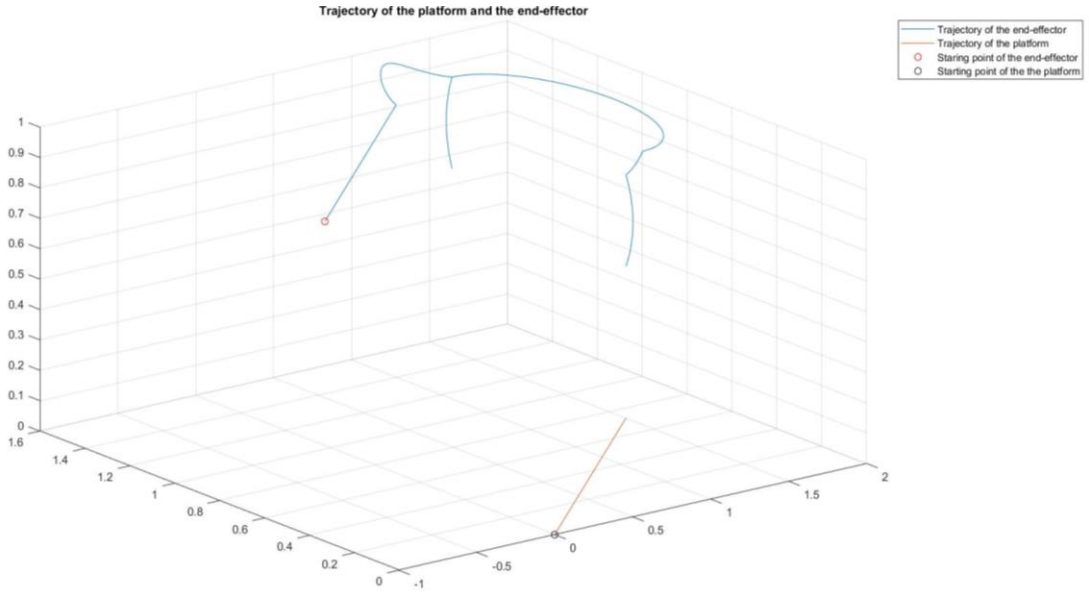

Figure 14 - 3D Representation of End-effector' and Platform's trajectories (JS)

4) Velocities:

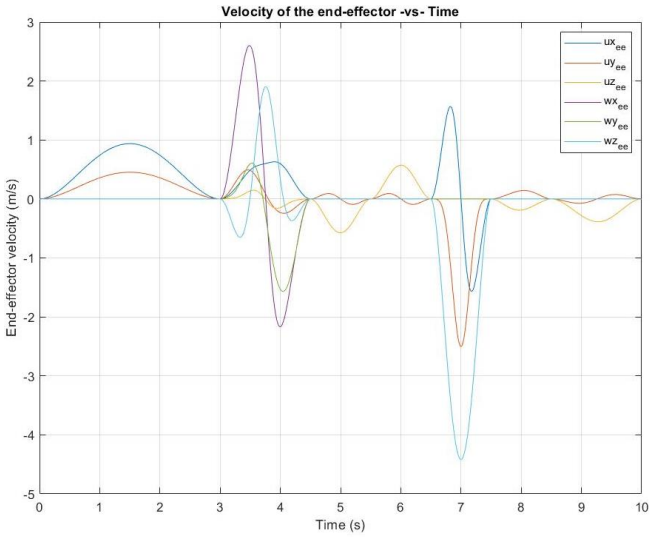

Figure 16 - Velocities of the End-effector -vs- Time (JS)

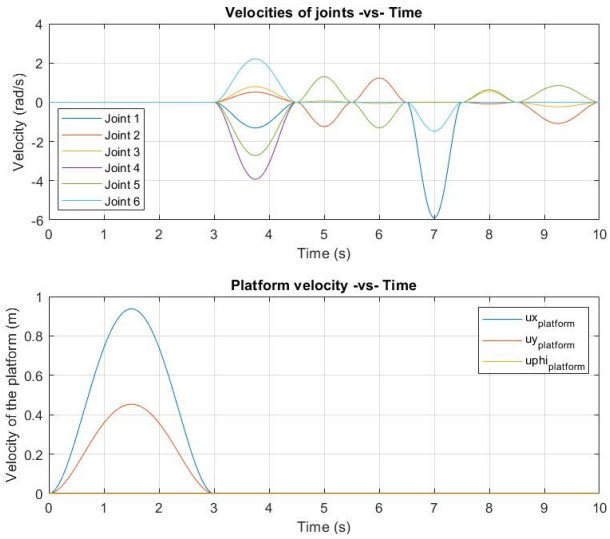

Figure 17 – Velocities of the Joints/Platform -vs- Time (JS)

5) Side view of the simulated trajectory:

Note: Again, to clearly show the placed object, the robot arm is finally plotted in the initial joints q_0

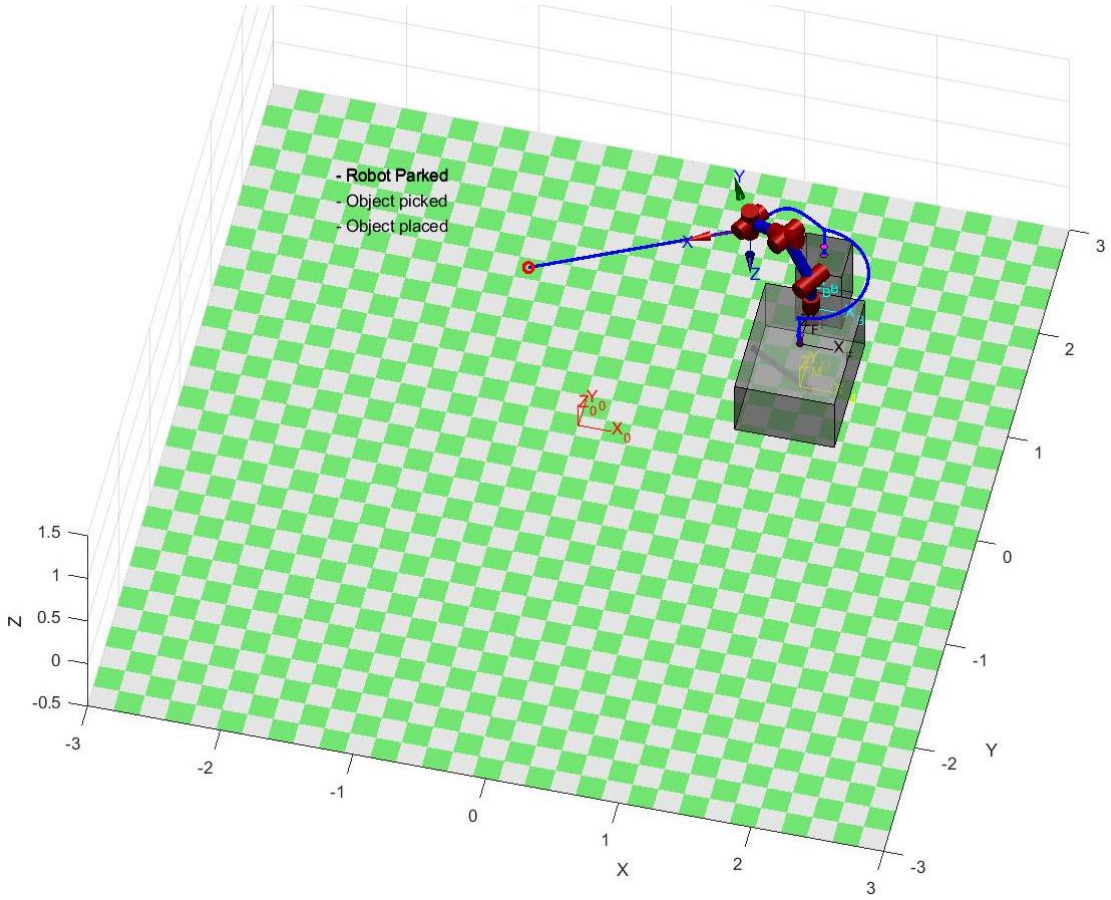

Figure 18 – Plotted Trajectory for the end-effector in the simulation environment (JS)

VIII. Conclusion

In this paper the analysis for the trajectory planning of a omnidirectional mobile manipulator with 6-DOF is presented for a simple pick and place task. The trajectories were planned in both spaces: Joint Space and Task Space. In the case of trajectory planning in Task Space, the trajectories have been planned and used the inverse kinematics on every point to get the positions of the angles and implement the trajectory in the Joint Space. In the case of trajectory planning in Joint Space, we used inverse kinematics only to get the via points for the joints and then forward kinematics on each step to follow the pose of the end-effector along the trajectory as well as calculating its velocities. Also, a mixed trajectory can be planned and could be a sum of trajectory parts 1, 2.1-2.6 as planned and presented above, in order to get a more fitting solution. For example, we see that in the Joint Space, for the downwards movement of the arm, the trajectory is not a straight line as we would like, and this could lead to collision or even damage with/to the object itself. On the contrary, in the Task Space the trajectory is a straight line, and thus more fitting for this task. The trajectory 2.4 which brings the arm closer to the placing position by passing through a midpoint, is clearly smoother in the Joint Space as the arm performs a rotational move (2.4 is completely optional in the Joint Space, we could immediately move from Point 2.3 to Point 2.5 without any problem again by performing a smooth rotational motion). We can choose which trajectories (and in which Space) to use based on the needs of every task.

IX. Future work

Many different adaptations, tests, and experiments have been left for the future due to lack of time (the assignment conducted in the university's exam period). Future work concerns deeper analysis of trajectory planning on the task with:

a) Multi segments using LSPB (linear segment with parabolic blends) parts and multi-axis trajectories to make the motion smoother and continuous (both for joints' motion and the end-effector's motion) without stopping at the via points unless it is necessary for the needs of the task.

b) Experiments with mixed trajectories

c) Bring the simulation into a real time system, work on its dynamic equations of motion and use a control law to track the planned trajectory.

References

[1] P.I. Corke, "Robotics, Vision & Control", Springer 2017, ISBN 978-3-319-54413-7

[2] Kevin M. Lynch and Frank C. Park. 2017. Modern Robotics: Mechanics, Planning, and Control (1st. ed.). Cambridge University Press, USA. - p.500

[3] DAM, E., KOCH, M., AND LILLHOLM, M., 1998. "Quaternions, interpolation and animation". Tech. rep. DIKU-TR-98/5, University of Copenhagen.