POLYTECHNIC OF BARI

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

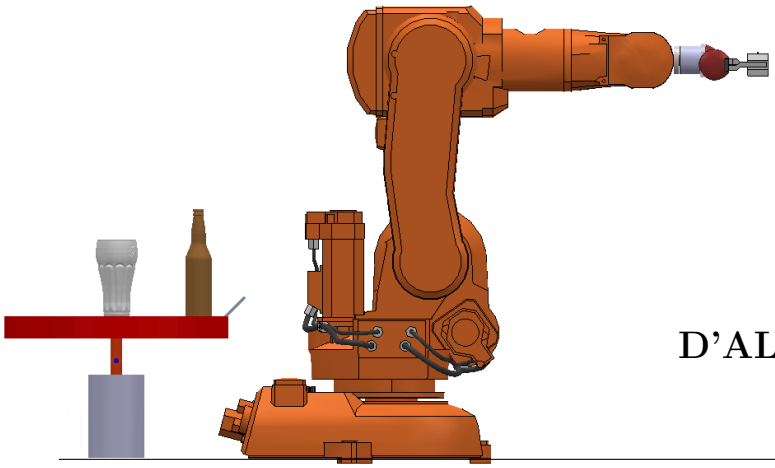
AUTOMATION ENGINEERING MASTER'S DEGREE

Course on ROBOTICS - INDUSTRIAL HANDLING

Prof. Engr. Paolo **LINO**

Project work:

Trajectory Planning for Bartender Robot using MATLAB and CoppeliaSim



Students:
D'ALESSANDRO Vito Ivano
VENEZIA Antonio

ACADEMIC YEAR 2019-2020

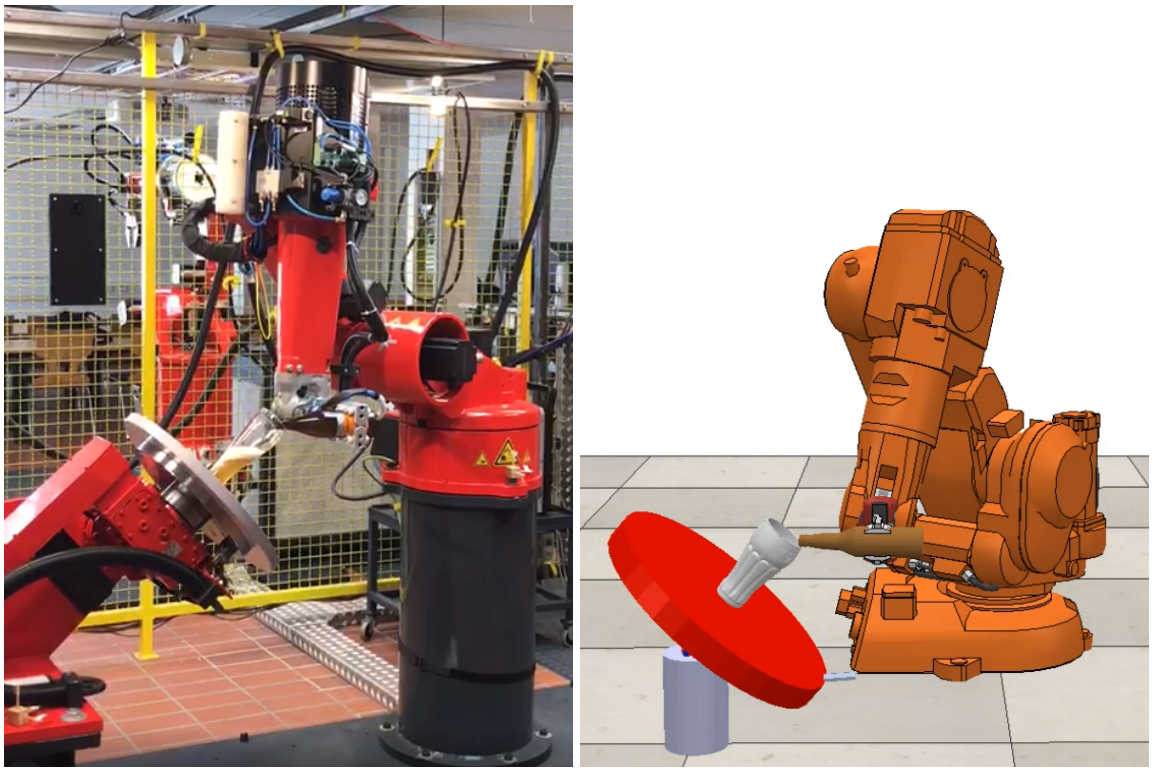
Abstract

This paper presents the trajectory planning in the operational and joint space for ABB IRB-140 robot manipulator. The task assigned to the manipulator is spilling beer from a bottle. Trajectory planning is based on a quintic polynomial that allows to obtain continuous velocity and acceleration of each joint variables of manipulator, used in the joint space and in the operational space to generate the required time laws for the primitive paths chosen. The identification of trajectory in the operational space and inverse kinematics algorithm is mandatory in order to obtain the joints position at each time instant.
Video of the simulation is available on the following link:
`https://www.youtube.com/watch?v=FVfmU47fygc&ab_channel=TeamThor`

Contents

1 Introduction

The development of researches in manipulator robots has been increasing with the objective of optimizing their capacities, carrying out iterating activities with the velocity, precision and efficiency required for whichever automatic process. The companies that build robots know that its market on these devices depends on their designs and innovations, so that they can be able to work in any environment, carrying out complex tasks with greater energetic efficiency. The use of new technologies, allow manipulator robots to work in unusual applications: in this report an industrial manipulator, the ABB IRB-140, has been used to simulate a bartender robot that can spill a beer. By using simultaneously an integrated development environment like CoppeliaSim and MATLAB by means of a remote API, an existing trajectory gathered from a video has been mostly reproduced (Fig.1.1).



(a) Real case (b) Simulated case

Figure 1.1: Study case

Due to the trajectory planning complexity for the study case, it has been decoupled into different parts, where each part has been implemented with different approaches as described in the following sections. As mentioned above, the trajectory can be considered as combination of the following trajectories/approaches, in order to accomplish the assigned task:

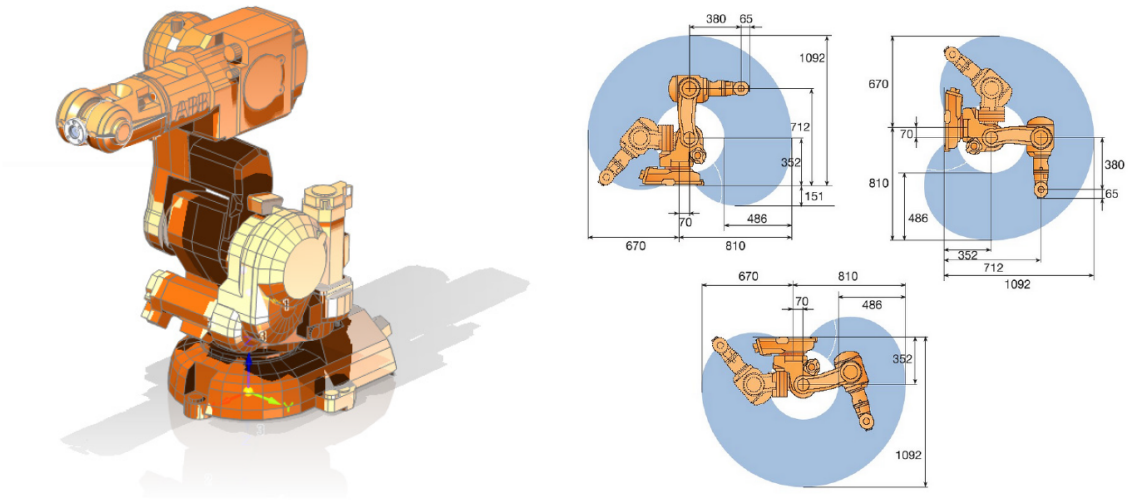
1. approaching the bottle - joint space path
2. grasp the bottle - operational space linear path
3. take off the bottle from the table - operational space linear path
4. uncork the bottle - operational space linear path

5. spill the beer for the first time, approaching the glass correctly - operational space linear path
6. shake the bottle with circular and tilted movements to reduce the beer froth - operational space circular path
7. spill the beer for the second time - operational space linear path
8. drip the bottle spill the residuals - operational space linear path
9. rotate the bottle to a vertical position and put it back on the table - operational space linear path
10. set the manipulator to the initial posture - joint space path

The report is structured as follow: in Sec.2 ABB IRB-140 robot manipulator is presented, specifying all technical and design parameters, forward and inverse kinematics solutions are derived; in Sec.3 an overview of the trajectory planning techniques are presented both in the joint and operational space; in Sect.4 a quick description of the integrated development environment used to simulate the behaviour of robot, named *CoppeliaSim*, is shown while in Sec.5 simulations and results are presented and discussed.

2 Robot manipulator

In the study case, ABB IRB-140 robot is considered: it is a compact industrial manipulator with 6 degrees of freedom (DOF); due to robust design, it can be mounted in the floor, in a roof of inverted form or in a wall at any angle. Furthermore, it has an integrated cabling, making it fairly flexible and allowing it to be adequate and easy to integrate into whichever robotic process.



(a) 3D CAD model (b) Workspace

Figure 2.1: ABB IRB-140 manipulator

2.1 Design parameters

All technical specifications, documents, CAD models and videos of ABB IRB-140 robot, are shown well detailed on [1]. Only the parameters from below that are useful for the kinematics analysis were taken into account.

SPECIFICATION	
Number of Axes:	
Robot Manipulator	6
Supplementary Load:	
Upper Arm	1 Kg
Wrist	0.5 Kg
Peso:	
Weight of Manipulator	98 Kg
DEVELOPMENT	
Axis No.	Work Range
1	360°
2	200°
3	280°
4	Unlimited (400° Default)
5	240°
6	Unlimited (800° Default)
ANGULAR SPEED	
Axis No.	Max Speed
1	200°/s
2	200°/s
3	260°/s
4	360°/s
5	360°/s
6	450°/s

Table 2.1: Technical data of ABB IRB-140 robot

Moreover, in Tab.2.2, length of each link of the manipulator shown in Fig.2.1 are defined.

LINK	LENGHT	UNIT
L0	0.07	m
L1	0.325	m
L2	0.36	m
L3	0.38	m
L4	0.046	m
Le	0.16	m

Table 2.2: Lenght of links ABB IRB-140 robot

2.2 Forward kinematics

In order to analyze joints' position, a set of transformation matrices can be defined, where T_i^{i-1} is a transformation matrix that expresses the coordinate transformation from frame i to $i-1$.

If the Denavit-Hartenberg convention is considered, this transformation matrix depens on four different parameters and is defined as follows:

$$T_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & L_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & L_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where

- a_i is the distance between O_i and $O_i^{'}$;
- d_i is the coordinate of $O_i^{'}$ along z_{i-1};
- α_i is the angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise;
- θ_i is the angle between axes x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.

Using DH convention the following table can be written:

Link	θ_i	d_i	a_i	α_i
1	θ_1	L1	L0	$\frac{\pi}{2}$
2	θ_2	0	L2	0
3	θ_3	0	0	$\frac{\pi}{2}$
4	θ_4	L3	0	-$\frac{\pi}{2}$
5	θ_5	0	0	$\frac{\pi}{2}$
6	θ_6	L4+Le	0	0

Table 2.1: Denavit-Hartenberg table for ABB IRB-140 manipulator

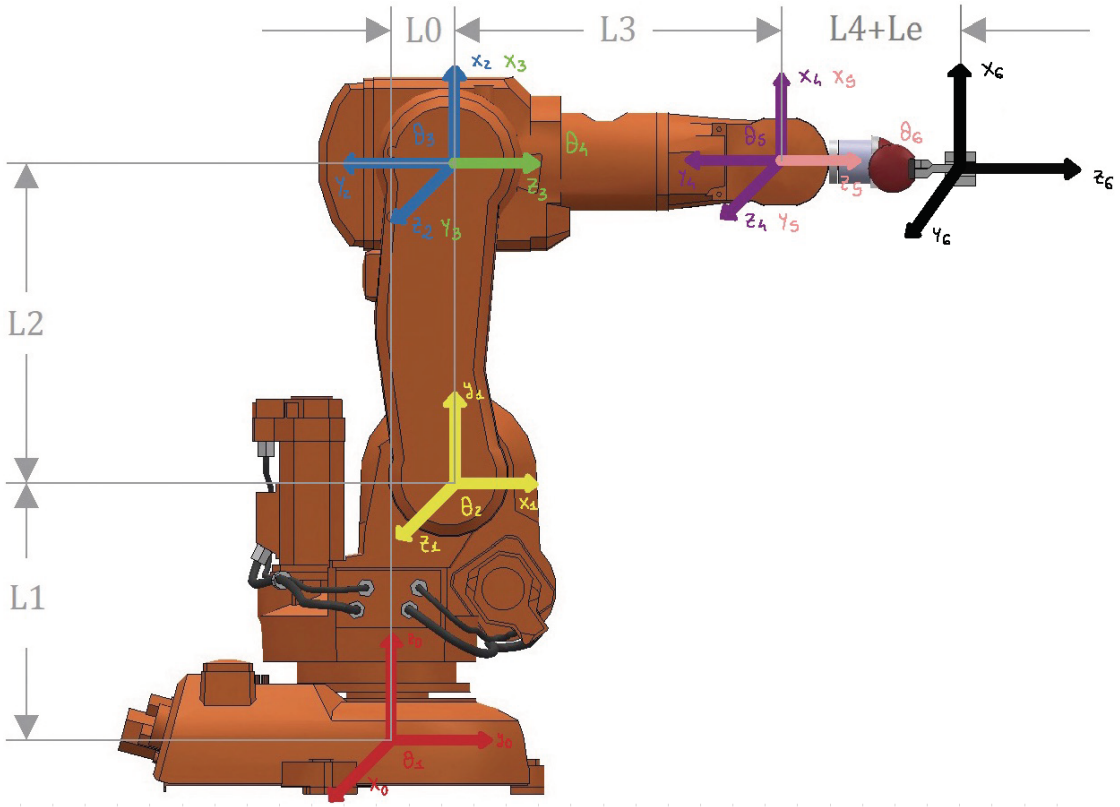


Figure 2.1: DH convention for ABB IRB-140 robot manipulator

2.3 Inverse kinematics

The inverse kinematics problem consists of the determination of the joint variables corresponding to a given end-effector position and orientation. The solution to this problem is of fundamental importance in order to transform the motion specifications, assigned to the end-effector in the operational space, into the corresponding joint space motions that allow execution of the desired motion. In the study case,a geometric approch will be used to compute the inverse kinematics solutions, so it is required geometric intuition to find those significant points on the structure with respect to which it is convenient to express position and/or orientation as a function of a reduced number of unknowns. Because of the ABB IRB-140 manipulator is a formed by an arm, in particular anthropomorphic arm, and a spherical wrist the inverse kinematics problem can be articulate into two subproblems, since the solution for the position is decoupled from that for the orientation: a suitable point along the structure can be found whose position can be expressed both as a function of the given end-effector position and orientation and as a function of a reduced number of joint variables, so for manipulator with spherical wrist, the natural choice is to locate such point W at the intersection of the three terminal revolute axes. In fact, once the end-effector position and orientation are specified in terms of $\boldsymbol{p}_e$ and $\boldsymbol{R}_e = [\boldsymbol{n}_e, \boldsymbol{s}_e, \boldsymbol{a}_e]$, the wrist position can be found as:

$$\boldsymbol{p}_W = \boldsymbol{p}_e - d\boldsymbol{a}_e \tag{2}$$

where d is the distance between the end-effector and the wrist position W along $\boldsymbol{a}_e$ axis.
As can be seen in Eq.2, the wrist position is function of the sole joint variables that determine the arm position. Hence, in the case of a nonredundant three-DOF arm, the inverse kinematics can be solved according to the following steps:

1. Compute the wrist position $\boldsymbol{p}_W(\theta_1, \theta_2, \theta_3)$;
2. Solve inverse kinematics for position (θ_1,θ_2,θ_3);
3. Compute $\boldsymbol{R}_3^0(\theta_1, \theta_2, \theta_3)$.
4. Compute $\boldsymbol{R}_6^3(\theta_4, \theta_5, \theta_6) = \boldsymbol{R}_3^{0^T} \boldsymbol{R}_e$
5. Solve inverse kinematics for orientation (θ_4,θ_5,θ_6).

In this way, it is possible to solve the inverse kinematics for the arm separately from the inverse kinematics for the spherical wrist.
Fig.2.1 shows a schematic representation of ABB IRB-140 manipulator, useful to solve the inverse kinematics problem of the manipulator.

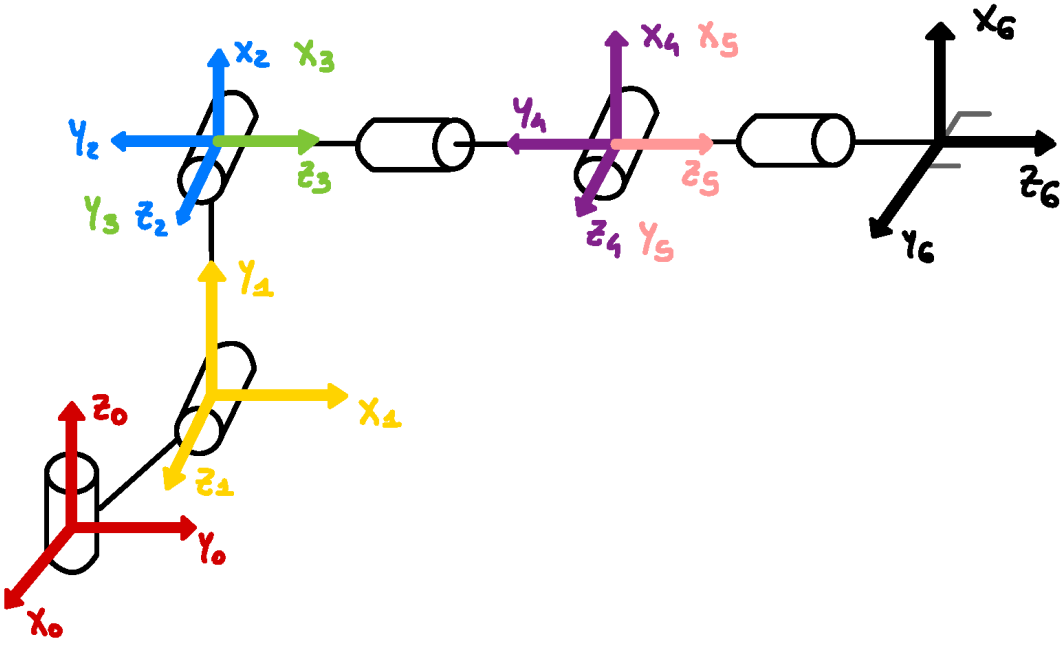


Figure 2.1: Schematized ABB IRB-140 robot manipulator

2.3.1 Solution of Anthropomorphic Arm

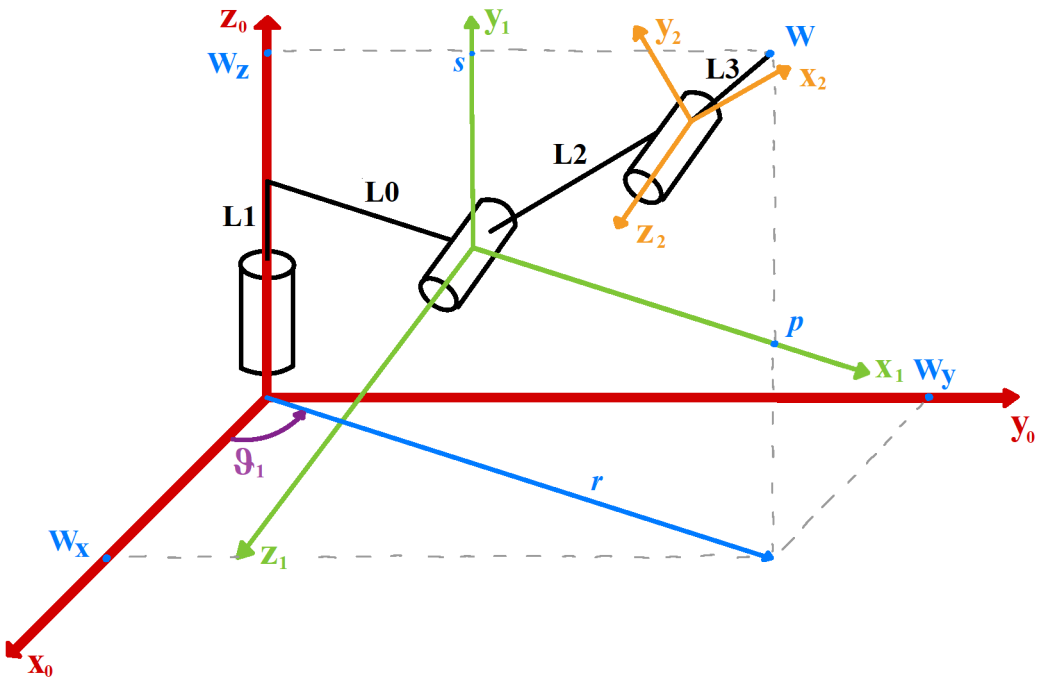


Figure 2.2: Anthropomorphic Arm

Fig.2.2 shows a schematic representation of anthropomorphic arm and each frame attached to the links of manipulator. It can be seen that the decoupling point for the inverse kinematics is $\boldsymbol{W} = [W_x, W_y, W_z]^T$. From Fig.2.2, it's quite straightforward to say that:

$$\theta_1 = \text{atan2}\,(W_y, W_x)$$

$$r = \pm\sqrt{W_x^2 + W_y^2}$$

$$s = W_z - L1$$

$$p = r - L0$$

In order to compute θ_2 and θ_3, it must be considered the law of cosine (Fig.2.3), which says that:

$$s^2 + p^2 = L2^2 + L3^2 - 2 \cdot L2 \cdot L3 \cdot \cos\Psi \tag{3}$$

In particular, $\Psi = \pi - \theta_3$, so $\cos(\Psi) = \cos(\pi - \theta_3) = -\cos(\theta_3)$. Hence, Eq.3 can be written as follows:

$$s^2 + p^2 = L2^2 + L3^2 + 2 \cdot L2 \cdot L3 \cdot \cos\theta_3 \tag{4}$$

From Eq.4 the cosine of θ_3 can be computed:

$$\cos(\theta_3) = \frac{s^2 + p^2 - L2^2 - L3^2}{L2 \cdot L3}$$

So

$$\theta_3 = \text{atan2}(\pm(1 - \cos{\theta_3}^2), \cos\theta_3) \tag{5}$$

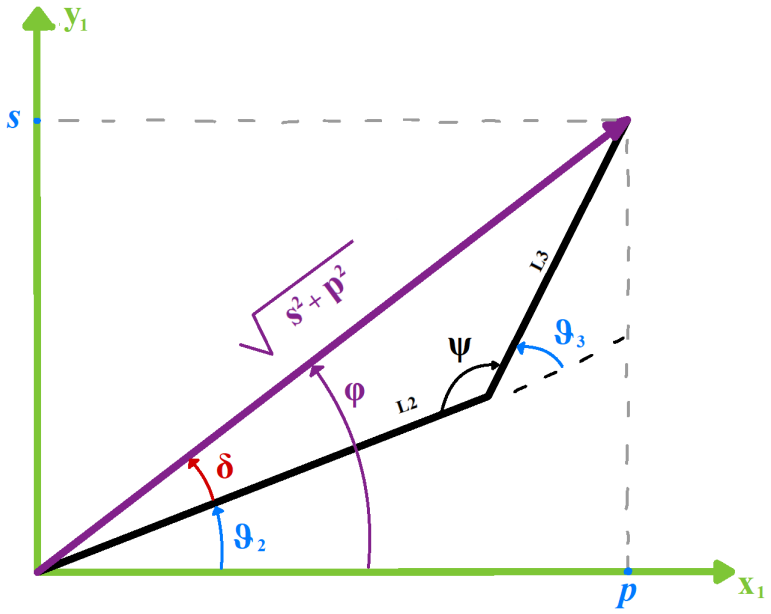


Figure 2.3: Law of cosine

At this point, the only angle must be computed is θ_2. In order to do it, it must be considered the following geometric intuitions:

$$\delta + \theta_2 = \phi$$

$$\phi = \text{atan2}(s, p)$$

$$\delta = \text{atan2}(L3 \sin\theta_3, L2 + L3 \cos\theta_3)$$

So, according to these geometric intuitions, the last angle of anthropomorphic arm can be computed as:

$$\theta_2 = \text{atan2}(s, p) - \text{atan2}(L3 \sin\theta_3, L2 + L3 \cos\theta_3) \tag{6}$$

2.3.2 Solution of Spherical Wrist

According to frame defined for revolute joints of spherical wrist, these angles constituite a set of Euler angles ZYZ with respect to Frame 3. So, after having computed the rotation matrix

$$\boldsymbol{R}_6^3 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

it is possible to compute the solutions directly using the solution of Euler angles ZYZ:
If $\theta_4 \in (0, \pi)$

$$\begin{aligned} \theta_4 &= \text{atan2}(r_{23}, r_{13}) \\ \theta_5 &= \text{atan2}(\sqrt{{r_{13}}^2 + {r_{23}}^2}, r_{33}) \\ \theta_6 &= \text{atan2}(r_{32}, -r_{31}) \end{aligned} \tag{7}$$

while if $\theta_4 \in (-\pi, 0)$

$$\begin{aligned} \theta_4 &= \text{atan2}(-r_{23}, -r_{13}) \\ \theta_5 &= \text{atan2}(-\sqrt{{r_{13}}^2 + {r_{23}}^2}, r_{33}) \\ \theta_6 &= \text{atan2}(-r_{32}, -r_{31}) \end{aligned} \tag{8}$$

2.4 Differential Kinematics

Differential kinematics gives the relationship between the joint velocities and the corresponding end-effector linear and angular velocity[2]. This relationship is given by a matrix, named geometric Jacobian, which depends on the mechanism configuration. Defined $\dot{\mathbf{p}}_e$ the end-effector linear velocity and $\mathbf{w}_e$ the angular velocity, equations that lead these to the joint velocities $\dot{\mathbf{q}}_i$ are:

$$\dot{\mathbf{p}}_e = \mathbf{J}_P(\mathbf{q})\dot{\mathbf{q}} \tag{9}$$

$$\mathbf{w}_e = \mathbf{J}_O(\mathbf{q})\dot{\mathbf{q}} \tag{10}$$

where $\mathbf{J}_P$ is a $3 \times n$ matrix that relates the joint velocities with the end-effector linear velocities, whereas $\mathbf{J}_O$ is also $3 \times n$ matrix that links the joint velocities to the end-effector angular velocity. The geometric Jacobian matrix is obtained matching the previous two matrices, resulting in a $6 \times n$ matrix (where n is the number of joint variables) defined as:

$$J = \begin{bmatrix} \mathbf{J}_{P1} & \mathbf{J}_{P2} & \dots & \mathbf{J}_{Pn} \\ \mathbf{J}_{O1} & \mathbf{J}_{O2} & \dots & \mathbf{J}_{On} \end{bmatrix} \tag{11}$$

If the $i-th$ joint is prismatic the geometric Jacobian matrix can be computed as:

$$\begin{bmatrix} \mathbf{J}_{Pi} \\ \mathbf{J}_{Oi} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{i-1} \\ \mathbf{0} \end{bmatrix} \tag{12}$$

while if it is a revolute joint:

$$\begin{bmatrix} \mathbf{J}_{Pi} \\ \mathbf{J}_{Oi} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} \tag{13}$$

In Eq.12 and Eq.13 all terms are given by transformation matrices, in particular:

- $\mathbf{z}_{i-1}$ is the z-axis, corresponding to the third column of rotation matrix R^0_{i-1};
- $\mathbf{p}_e$ is the position vector of the end-effector respect to the base frame 0, corresponding to the first three element of the fourth column of transformation matrix T^0_e;
- $\mathbf{p}_{i-1}$ is the position vector of the origin of frame $i-1$ respect to the base frame 0, corresponding to the first three element of the fourth column of transformation matrix T^0_{i-1}.

In the study case, the Jacobian matrix allows to compute linear velocity of the end-effector when the trajectory planning is computed in the joint space. Because of the ABB IRB-140 robot manipulator has 6 joints, the Jacobian matrix will be a 6×6 square matrix: in fact, the robot manipulator considered is a nonredundant manipulator.

The easiest way to compute the Jacobian matrix is defining a function as follows:

```
function [J_I] = Jacobian(T,T_E0)
J_I = [];
for i = 1:length(T)
    if (T(i).JT == 'R')
        JP = ...
            cross(T(i).T_Iminus1_0(1:3,3),(T_E0(1:3,end)-T(i).T_Iminus1_0(1:3,end)));
        JO = T(i).T_Iminus1_0(1:3,3);
    else
        JP = T(i).T_Iminus1_0(1:3,3);
        JO = [0 0 0]';
    end
    J_I = [J_I,[JP;JO]];
end
end
```

2.5 Inverse Differential Kinematics

Inverse differential kinematics allows to compute the joint velocities of a robot manipulator if a motion trajectory is assigned to the end-effector in terms of $\boldsymbol{v}_e$ and the initial conditions on position and orientation in the operational space.
In particular, if the manipulator is nonredundant (dimension of joint space is equal to the dimension of operational space), the joint velocities can be obtained via simple inversion of the Jacobian matrix:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(q)\boldsymbol{v}_e \tag{14}$$

It's important to notice that this technique for inverting kinematics is independent of the solvability of the kinematic structure. Nonetheless, it is necessary that the Jacobian have to be square and full rank; this demands further insight into the cases of redundant manipulators and kinematic singularity occurrence.

3 Trajectory Planning

For the execution of a specific robot task, it is worth considering the main features of motion planning algorithms. The goal of trajectory planning is to generate the reference inputs to the motion control system which ensures that the manipulator executes the planned trajectories. The user typically specifies a number of parameters to describe the desired trajectory. Planning consists of generating a time sequence of the values attained by an interpolating function (typically a polynomial) of the desired trajectory. Planning can be either *point-to-point motion* (where only the initial and final position are assigned) and *motion through a sequence of points* (where a finite sequence of point along the path are assigned). The trajectory planning can be executed in the joint space or in the operational space. In the study case, each approach are used and presented more in depth in the next sections.

3.1 Trajectory planning in the Joint Space - Point to Point Motion

When the initial and final posture, q_i and q_f, of the manipulator are given in terms of joint variables (f.e solving the inverse kinematics problem known the initial and final pose of the end-effector), one possibility is to use for the n^{th} joint variable a polynomial function which describes the position, velocity and acceleration, under certain constraints. The latter can be the initial and final position and velocity and/or acceleration. In the study case, a polynomial of $5th$ order has been chosen in order to set initial and final velocity and acceleration. So, the motion timing law for the n^{th} joint of the manipulator is then given by:

$$q_n(t) = a_{n,5}t^5 + a_{n,4}t^4 + a_{n,3}t^3 + a_{n,2}t^2 + a_{n,1}t + a_{n,0}$$

$$\dot{q}_n(t) = 5a_{n,5}t^4 + 4_an, 4t^3 + 3_{n,3}t^2 + 2_{n,2}t + a_{n,1}$$

$$\ddot{q}_n(t) = 20a_{n,5}t^3 + 12a_{n,4}t^2 + 6a_{n,3}t + 2a_{n,2}$$

whose coefficients can be computed by imposing the initial and final conditions about position, velocity and acceleration:

$$q_n(t_i) = q_{n,i}$$

$$q_n(t_f) = q_{n,f}$$

$$\dot{q}_n(ti) = 0$$

$$\dot{q}_n(t_f) = 0$$

$$\ddot{q}_n(ti) = 0$$

$$\ddot{q}_n(t_f) = 0$$

where t_i and t_f are the initial and final time instant respectively and in this particular case null initial and final velocity and acceleration have been considered.

3.2 Trajectory Planning in Operational Space

A joint space trajectory planning algorithm generates a time sequence of values for the joint variables $q(t)$ so that the manipulator is taken from the initial to the final configuration, eventually by moving through a sequence of intermediate configurations. The resulting end-effector motion is not easily predictable, in view of the nonlinear effects introduced by direct kinematics. Whenever it is desired that the end-effector motion follows a geometrically specified path in the operational space, it is necessary to plan trajectory execution directly in the same space. Planning can be done either by interpolating a sequence of prescribed path points or by generating the analytical motion primitive and the relative trajectory in a punctual way. In both cases, the time sequence of the

values attained by the operational space variables is used in real time to obtain the corresponding sequence of values in the joint space variables, via an inverse kinematics algorithm. A generic trajectory in the operational space can be obtained by combining primitive of motions such as linear and circular path. The latter are presented in the next sections.

3.2.1 Linear Path

Consider the linear segment connecting point p_i to point p_f. The parametric representation of this path in terms of position, velocity and acceleration is:

$$\boldsymbol{p}(s) = \boldsymbol{p_i} + s\frac{(\boldsymbol{p_f} - \boldsymbol{p_i})}{||(\boldsymbol{p_f} - \boldsymbol{p_i})||}$$

$$\frac{d\boldsymbol{p}}{ds} = \frac{(\boldsymbol{p_f} - \boldsymbol{p_i})}{||(\boldsymbol{p_f} - \boldsymbol{p_i})||}$$

$$\frac{d^2\boldsymbol{p}}{ds^2} = 0$$

where $\boldsymbol{p}(0) = \boldsymbol{p_i}$ and $\boldsymbol{p}(||p_f - p_i||) = \boldsymbol{p_f}$. As can be seen the motion time law depends indirectly from the curvilinear abscissa s. The latter can be generated by using the same approach for trajectory planning in the joint space.

3.2.2 Circular Path

Consider a circle in the tridimensional space. Before deriving its parametric representation, it is necessary to introduce its significant parameters. Suppose that the circle is specified by assigning (Fig.3.2.1):

- the unit vector of the circle axis $\boldsymbol{r}$;
- the position vector $\boldsymbol{d}$ of a point along the circle axis;
- the position vector $\boldsymbol{p}_i$ of a point belonging to the circumference.

Using these parameters, the position vector $\boldsymbol{c}$ which identifies the center of the circle can be found. Let's consider $\boldsymbol{\delta} = \boldsymbol{p}_i - \boldsymbol{d}$; in order to describe a circumference and not degenerate it into a point, it must be verified that:

$$|\boldsymbol{\delta}^T \boldsymbol{r}| < ||\boldsymbol{\delta}||$$

If the previous condition is verified, the center vector $\boldsymbol{c}$ is defined as:

$$\boldsymbol{c} = \boldsymbol{d} + (\boldsymbol{\delta}^T \boldsymbol{r})\boldsymbol{r}.$$

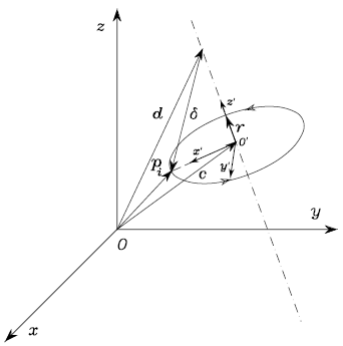


Figure 3.2.1: Parametric representation of a circle in the space

It is now desired to find a parametric representation of the circle as a function of the arc length. Notice that this representation is very simple for a suitable choice of the reference frame. To see this, consider the frame $O^{'} - x^{'}y^{'}z^{'}$, where O coincides with the center of the circle, axis x' is oriented along the direction of the vector $\boldsymbol{p}_i - \boldsymbol{c}$, axis z' is oriented along the versor $\boldsymbol{r}$ and axis y' is chosen so as to complete a right-handed frame. When expressed in the circumference frame, the parametric representation of the circle is:

$$\boldsymbol{p'}(s) = \begin{bmatrix} \rho cos(s/\rho) \\ \rho sin(s/\rho) \\ 0 \end{bmatrix}$$

where $\rho = ||\boldsymbol{p}_i - \boldsymbol{c}||$ is the radius of the circle and the point $\boldsymbol{p}_i$ has been assumed as the origin of the arc length. When expressed in a different frame (f.e $O - xyz$ frame), the path representation becomes:

$$\boldsymbol{p}(s) = c + \boldsymbol{R}\boldsymbol{p}(s)$$

where $\boldsymbol{c}$ is expressed in the frame $O - xyz$ and $\boldsymbol{R}$ is the rotation matrix of frame $O^{'} - x^{'}y^{'}z^{'}$ with respect to frame $O - xyz$. Differentiating with respect to curvilinear abscissa s gives:

$$\frac{d\boldsymbol{p}}{ds} = \boldsymbol{R} \begin{bmatrix} -sin(s/\rho) \\ cos(s/\rho) \\ 0 \end{bmatrix}$$

$$\frac{d^2\boldsymbol{p}}{ds^2} = \boldsymbol{R} \begin{bmatrix} -cos(s/\rho)/\rho \\ -sin(s/\rho)/\rho \\ 0 \end{bmatrix}$$

3.3 Orientation

End-effector orientation is typically specified in terms of rotation matrix of the (time-varying) end-effector frame with respect to the base frame. As is well known, the three columns of the rotation matrix R_E^0 represent the three unit vectors of the end-effector frame with respect to the base frame. Different approaches can be considered to describe the orientation of the end-effector over time such as using the euler angles and axis angle representation. The latter has been considered in this case: given two coordinate frames in the Cartesian space with the same origin and different orientation, it is always possible to determine a unit vector so that the second frame can be obtained from the first frame by a rotation of a proper angle θ about the axis r of such unit vector:

$$\theta = \arccos(\frac{r_{11} + r_{22} + r_{33} - 1}{2})$$

$$\boldsymbol{r} = \frac{1}{2sin\theta} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

Thus, if the rotation angle θ and the unit vector r are known, the rotation matrix R describing the change of the orientation between two frame can be determined.
In certain cases, finding the desired change in orientation of the end-effector by using the axis angle representation can be not straightforward. In fact, in the study case, defining the change of end-effector orientation with this approach, meanwhile the beer is "shaked", is not so intuitive. Thus, a different approach has been used. Let consider the end-effector frame: one of its axis will correspond to the bottle axis for the entire trajectory and, during the "shaking", it will trace the surface of a cone in the space, as can be seen in Fig.3.3.1.

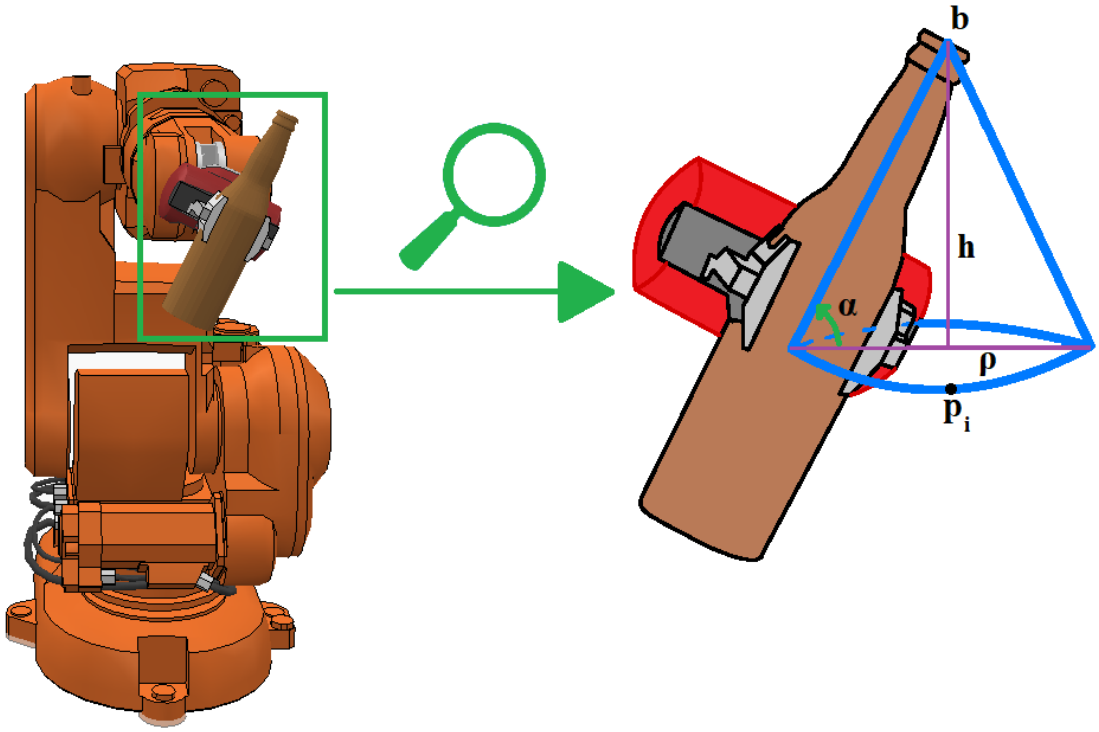


Figure 3.3.1: Cone

In this way, the unit vector k which identifies one axis of the end-effector frame coincident to the axis bottle can be determined after simple trigonometric considerations:

$$h = \rho \cdot \text{atan}(\alpha)$$
$$\boldsymbol{k} = \frac{\boldsymbol{b} - \boldsymbol{p_i}}{||\boldsymbol{b} - \boldsymbol{p_i}||}$$

where α is the tilt angle of the bottle, ρ is the radius of the circle, $\boldsymbol{b}$ is the vector which identifies the vertex of the cone, $\boldsymbol{p}_i$ the $i-th$ point of the circle in the space. Thus, for each unit vector, the direction cosine $\boldsymbol{dir}$ are determined as follow:

$$dir_x = \boldsymbol{k} \cdot \boldsymbol{x} \qquad dir_y = \boldsymbol{k} \cdot \boldsymbol{y} \qquad dir_z = \boldsymbol{k} \cdot \boldsymbol{z}$$

where:

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

are the unit versor which defines the orientation of base frame. So the unit vectors that identify the other two axis of the end-effector frame can be determined by defining the left-handed ortogonal basis as:

$$\boldsymbol{Y}_A = \begin{bmatrix} dir_x \\ dir_y \\ dir_z \end{bmatrix} \qquad \boldsymbol{X}_A = \begin{bmatrix} -dir_z \\ 0 \\ dir_x \end{bmatrix} \qquad \boldsymbol{Z}_A = \boldsymbol{X}_A \times \boldsymbol{Y}_A$$

At this point, the rotation matrix will be:

$$R = \begin{bmatrix} \boldsymbol{X}_A & \boldsymbol{Y}_A & \boldsymbol{Z}_A \end{bmatrix}$$

that gives the required orientation for each time instant in order to shake the bottle.

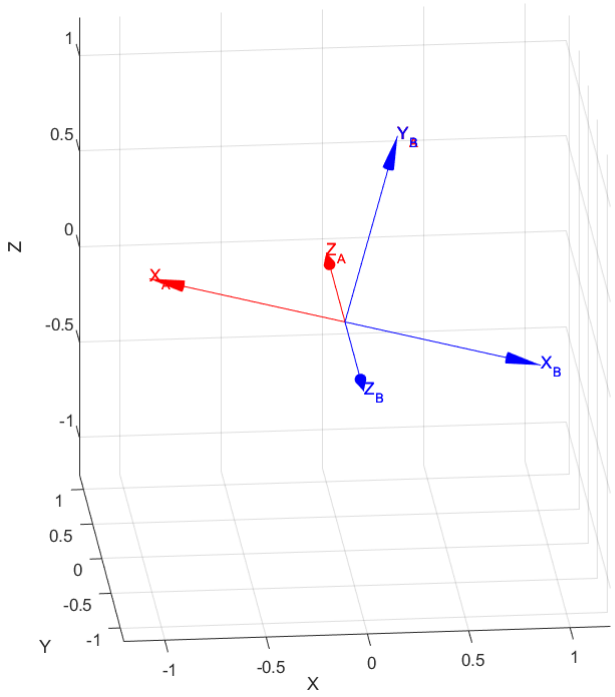


Figure 3.3.2: End-Effector frame during the shake

According to the task assigned to the manipulator, could be necessary to rotate the end-effector frame. In particular, in the study case, it has been observed that the orientation of the end-effector without any rotation matrix isn't the desired one (see red frame in Fig.3.3.2), so it has been necessary to rotate it with a rotation of π along y_B axis:

$$\boldsymbol{R}_B = \boldsymbol{R}_A \cdot \boldsymbol{R}_y(\pi)$$

4 CoppeliaSim

The robot simulator CoppeliaSim, with integrated development environment, is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS or BlueZero node, a remote API client, or a custom solution. This makes CoppeliaSim very versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab or Octave. CoppeliaSim is used for fast algorithm development, factory automation simulations, fast prototyping and verification, robotics related education, remote monitoring, safety double-checking, as digital twin, and much more.

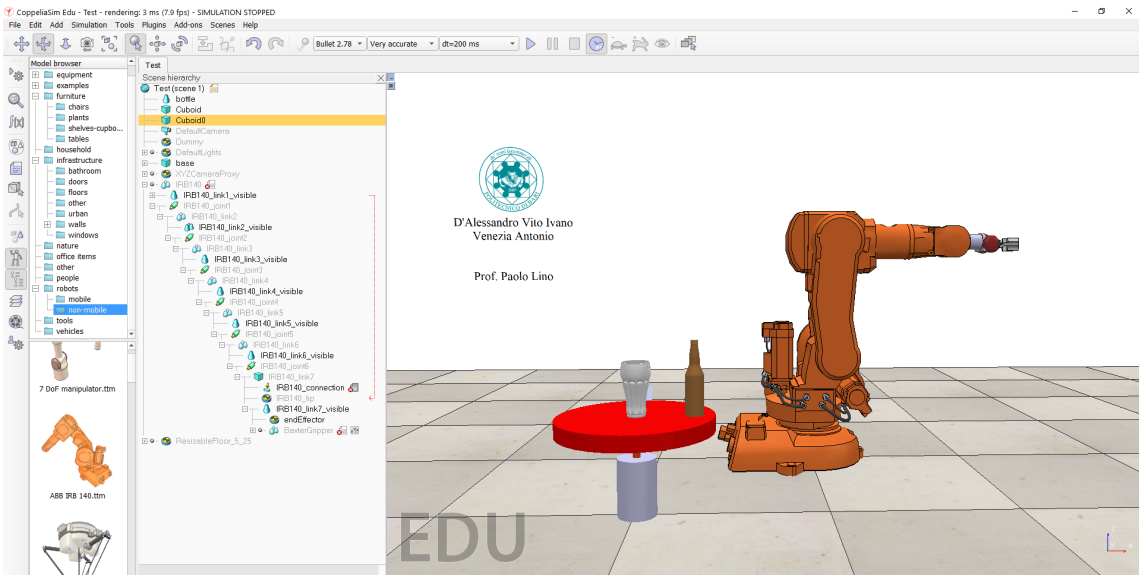


Figure 4.1: CoppeliaSim environment

```
angle = pi/2;                    % Desired angle for joint 1

vrep = remApi('remoteApi');      % Create remote api object
vrep.simxFinish(-1);             % Clear all opened connections

% Start communication to Coppelia
clientID = vrep.simxStart('127.0.0.1',19997,true,true,5000,5);

if (clientID > -1)
    disp('Connection Opened')
    vrep.simxAddStatusbarMessage(clientID,'Connection to ...
        Matlab',vrep.simx_opmode_oneshot);
    % Get Joint 1 handle
    [¬,J1_h] = ...
        vrep.simxGetObjectHandle(clientID,'IRB140_joint1',vrep.simx_opmode_blocking);
    % Enable synchronous mode
    [¬] = vrep.simxSynchronous(clientID,true);
    % Trigger CoppeliaSim
    [¬] = vrep.simxSynchronousTrigger(clientID);
    % Start simulation
    [¬] = vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot);
    % Get joint 1 position
    [¬,q1] = vrep.simxGetJointPosition(clientID,J1_h,vrep.simx_opmode_blocking);
    % Set joint 1 position
```

```
23      [¬] = ...
            vrep.simxSetJointTargetPosition(clientID,J1_h,angle,vrep.simx_opmode_oneshot);
24      % Stop simulation
25      [¬] = vrep.simxStopSimulation(clientID,vrep.simx_opmode_oneshot);
26  else
27      disp('Fail');
28  end
```

5 Simulation and results

In this section, the main results of the simulation are presented. As can be seen in Fig.5.2, the continuity in angle position is respected, as well as for angular velocity (Fig.5.3) where no discontinuity occur and the constraint on initial and final velocity/acceleration are fulfilled. Fig.5.1 shows the trajectory of the end-effector in the tridimensional space, where linear and circular path can be clearly identified. In addition, the manipulator never goes beyond the allowed operational space, related to its design parameters, avoiding kinematics singularities that can lead to trouble in the inverse kinematics resolution.
Moreover, Fig.5.4 shows position, linear velocity and acceleration of the end-effector of the manipulator during the whole trajectory. As can be seen, there are not discontinuity on the position, velocity and acceleration as it should be in order to avoid unfeasible behaviour of the manipulator.

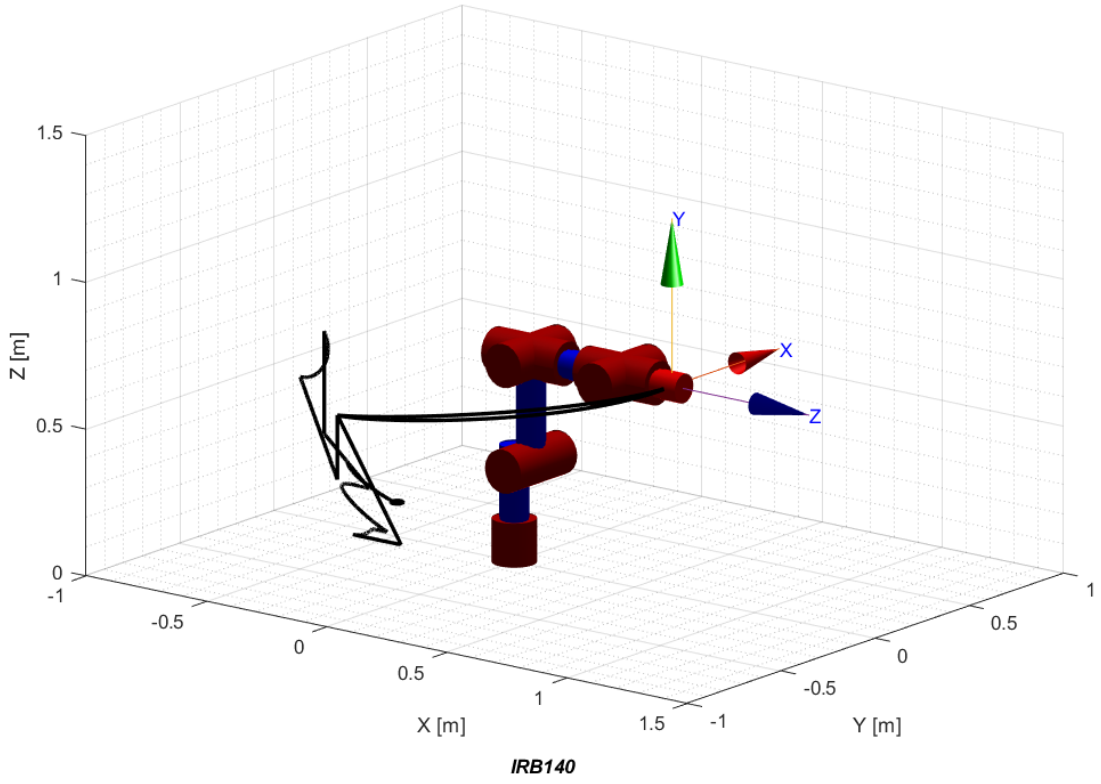


Figure 5.1: Plot of robot manipulator using Peter Corke Toolbox

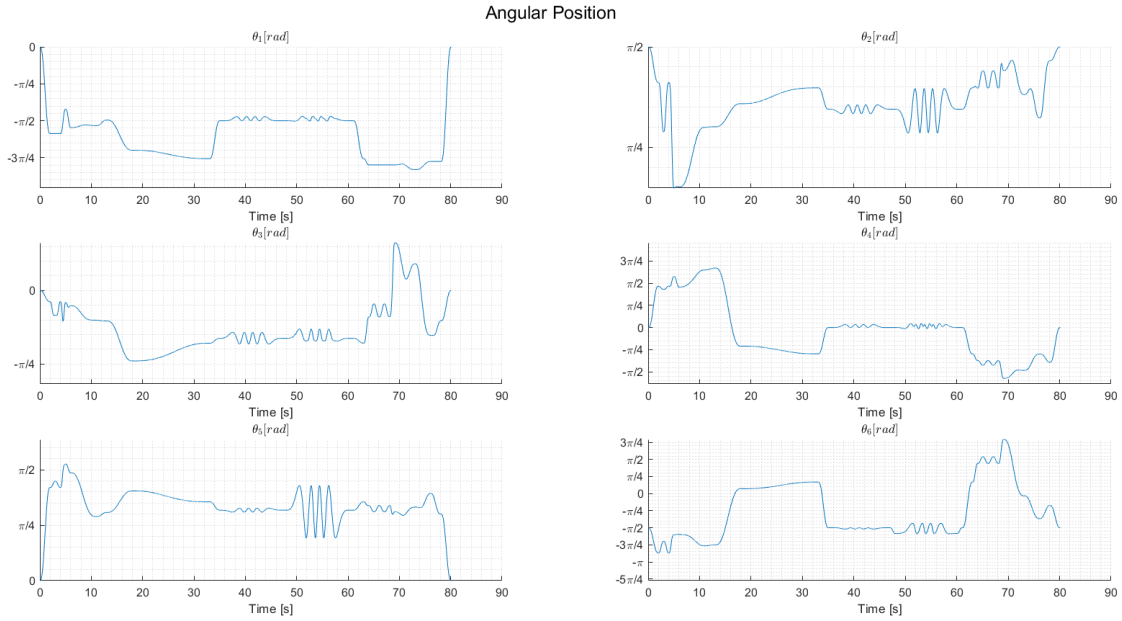


Figure 5.2: Angular position of joint variables

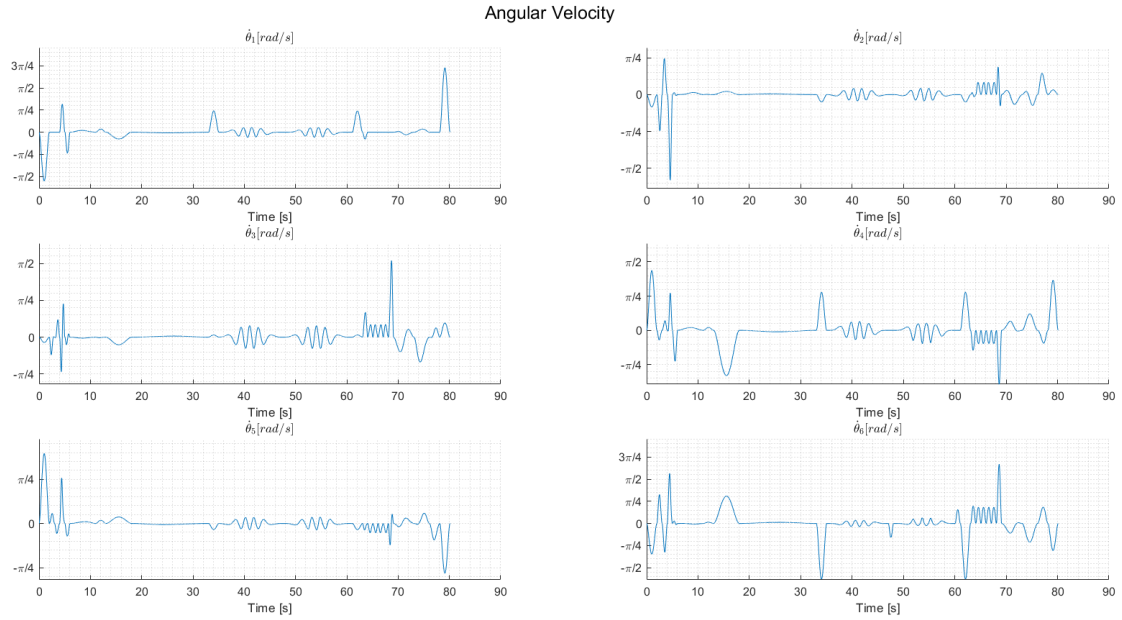


Figure 5.3: Angular velocity of joint variables

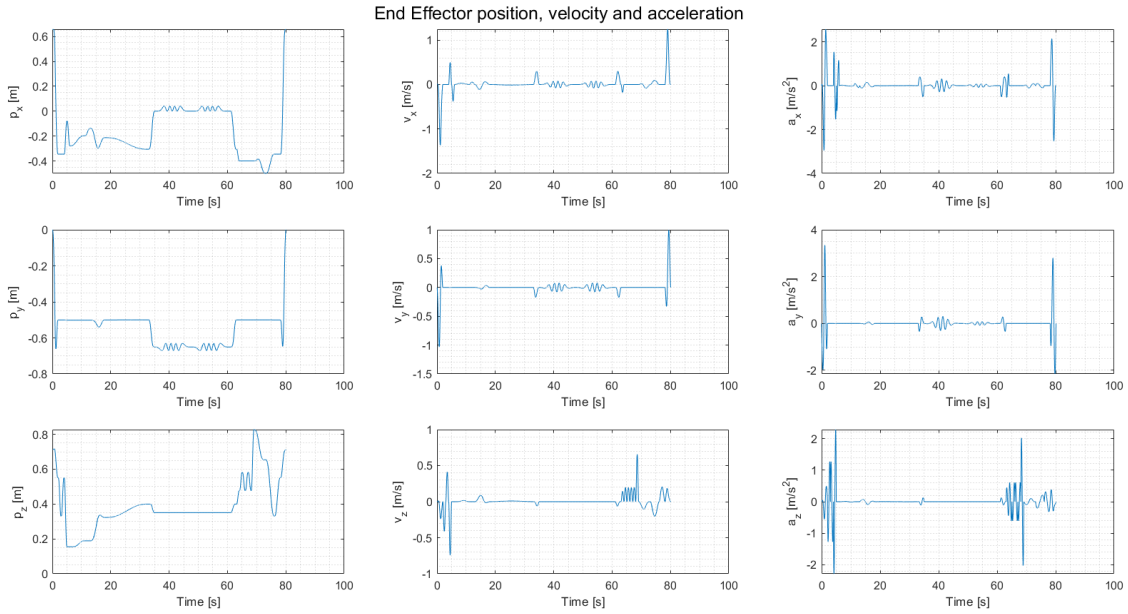


Figure 5.4: Position, velocity and acceleration of End-Effector

In order to understand if the task assigned to the manipulator can be achieved, it's mandotory to determine its workspace. In particular, in Fig.5.5 workspace of ABB IRB-140 robot manipulator within the trajectory assigned to the manipulator is shown.

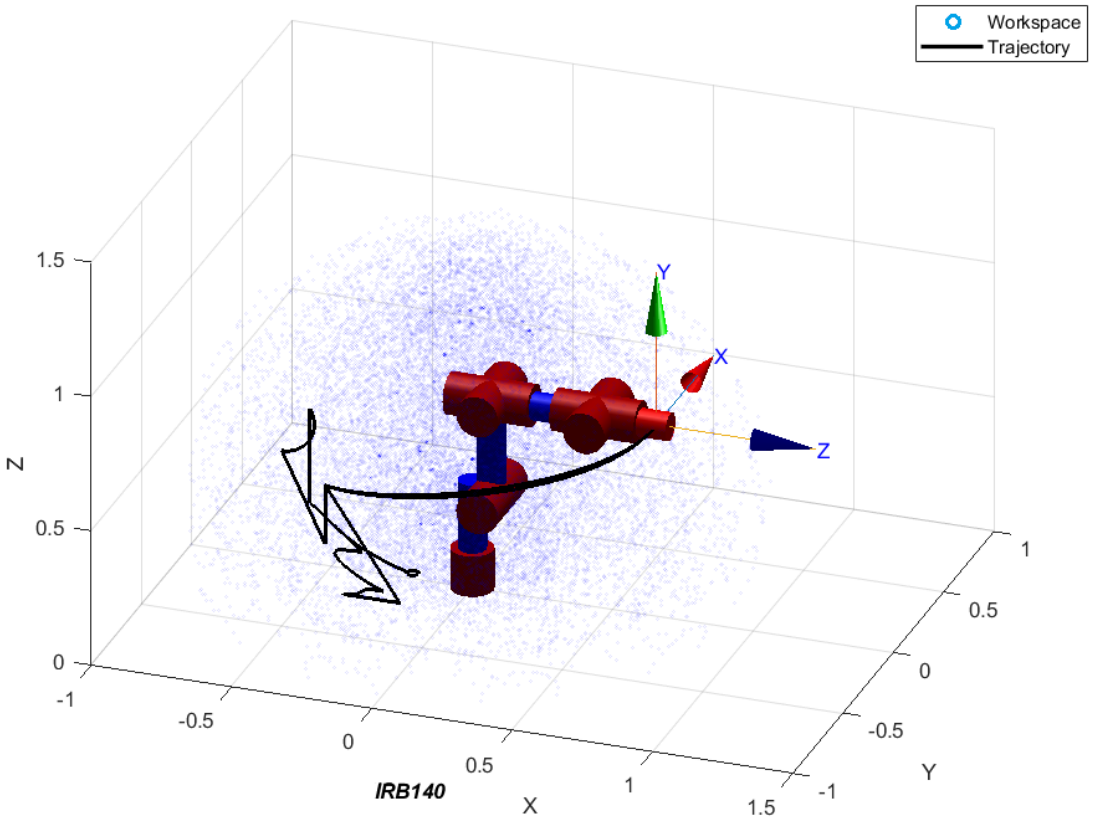


Figure 5.5: Workspace of ABB IRB-140 within trajectory planned

Obviously, the workspace is computed using a specific step for the joint variables of the manipulator, so in order to obtain a more detailed workspace a reduction of this step must be considered.

6 Conclusions & Future Works

In this report the trajectory planning of an industrial manipulator has been presented with the purpose of replicate a trajectory to accomplish the task to spill a beer. Different solutions have been taken into account by considering path in the joint and operational space. As can be seen also from the attached video, the desired trajectory has been replicated fairly faithfully.
Future works can be related to optimizing the code generated in Matlab, by increasing the sample time in order to reduce the computational effort due to the inverse kinematics algorithm. In this way, a linear microinterpolation has to be carry out in order to improve the performances. In addition, derivation of the dynamic model of the manipulator is mandatory for a dynamic analysis, applying a feasible control action in the operational space that guarantee stability of the whole system.

References

[1] ABB. *ABB IRB-140 Product specification*. ABB, 2020.

[2] L.Villani B.Siciliano L.Sciavicco. *Robotics, Modelling, Planning and Control*. Third edition. Mc-GrawHill, 2008. ISBN: 978-88-386-6322-2.

List of Figures

List of Tables