UDLAP®

Fundación Universidad de las Américas Puebla

Final project Robotic Kinematics and Dynamics:
Analysis and design for a "Bin Picking" robotic arm.

Composed by:

Charbel Breydy Torres 170995
Alain León López 168508
Andoni Díaz Castellanos 169292
Pablo Emilio Peredo Vega 171960

Advisor:
Dr. César Martínez Torres

Marzo-Mayo del 2023

Contents

1 Abstract

In this document, the kinematics of a robotic arm were designed for it to be efficient at bin-picking tasks; specifically using vision systems. The direct and inverse kinematics were solved, as well as a proposal for efficient optic usage of a camera for the arm to identify and reach objects in its environment. The Denavit-Heartenberg method was successfully implemented with the implementation of the ROS software and neuronal networks.

The purpose of this project is to program, simulate and make a bin-picking robot using Robotis' Open-Manipulator X using ROS, Matlab, and Python to control and guide the robot to perform bin-picking tasks.

2 Introduction

It was a fictional idea to create someday automated systems that could help humans with their tasks, one of the earliest being "The Tale of Talos", a bronze robot conceived by Hephaestus. And now, it's a common thing to see automated computer entities in our daily life, from an Osoji vacuum cleaner to a Kuka robotics arm in a factory.

Bin Picking is a common task in the robotics industry; the ability for a robot to see, calculate the depth of an object, pick it out of a container, and maybe sort it or categorise it by using Deep Learning, cameras, and certainly a robotic arm. This makes it safer, faster, and more reliable than a human attempting the same task. This, among other reasons, is because long exposure to some materials could be harmful to humans. Bin-picking applications can be implemented in almost all industrial sectors, such as automotive, logistics, or pharmaceutics, among many others (Sandra).

3 Problem statement

3.1 Justification and general Objectives

It is known that every industrial process needs a correct analysis of the systems upon which it will be implemented. In order to achieve the correct function of the Bin Picking system the team is implementing the OpenManipulator X robot; Certain levels of knowledge and skills are necessary to solve all the problems that this mechanism will have. The industry has many examples of using this robotic model, where the user could be able to select an object that is inside of the robot's workspace and optically detect it to give instructions, but this can´t be possible without knowledge about the dimensions of the robot, the kinematics, the optic program, and programmatic, etc. The general objective of this project is to bring all these elements together in order to create a usable system. For a correct and ordered process, it would be necessary to divide this job into four principal objectives:

3.2 Objectives

- The first objective is to establish the analytical data in order to be aware of the dimensions, function, and workspace of the robot. This analysis can be achieved by calculating the direct and inverse kinematics of the robot and using Denavit Hartenberg methods to obtain this could be a good start.
- The second objective is the implementation of the analysis of the first step into a program that allows us to send this analytical information to the robot.

- The third objective is to correctly implement the optic part that could interpret the workspace of the robot and detect forms and distances inside of the workspace of the robot in order to us to know where the robot could go for the Bin Picking system.
- The fourth objective is the correct use of ROS libraries in Python in order to interconnect all these elements into a functional and usable system.

4 Theoretical framework

4.1 Denavit-Hartenberg

Denavit-Hartenberg (DH) is a convention for parameterising the kinematic equations that describe the motion of a robotic manipulator. It is a widely used method for modeling the geometry and motion of articulated robotic systems, such as industrial robots or robotic arms. The DH convention defines a standard coordinate system for each joint of the robot, and specifies the transformation matrix that relates the coordinates of adjacent joints. The convention uses four parameters for each joint: the link length, the link twist angle, the link offset, and the joint angle. These parameters are used to construct a transformation matrix for each joint, which describes the position and orientation of the joint relative to the previous joint. DH parameters provide a convenient and consistent way to represent the geometry and motion of a robotic manipulator and can be used to derive the forward and inverse kinematic equations for the robot. By using DH parameters, it is possible to simplify the kinematic equations and make them easier to compute and manipulate. DH parameters also allow for the creation of standard kinematic equation libraries for different robotic systems, which can be shared and reused across different applications.

Overall, Denavit-Hartenberg is an important tool for modeling and analysing the motion of robotic manipulators and is widely used in robotics research, education, and industrial applications.

4.2 Direct Kinematics

Direct kinematics for bin picking involves determining the position and orientation of the end-effector (such as a gripper or suction cup) relative to the robot base, given the joint angles or parameters of the robot arm. This information is needed to plan and execute the motion of the end-effector towards the object to be picked up, and to ensure that the end-effector is aligned correctly with the object.

4.3 Inverse Kinematics

Inverse kinematics for bin picking involves determining the joint angles or parameters that will result in the end-effector (such as a gripper or suction cup) being able to pick up an object from a bin or container. This requires taking into account the position and orientation of the object within the bin and the location and configuration of the robot arm and end-effector. Overall, inverse kinematics for bin picking involves a combination of sensing, computation, and motion planning to achieve successful object manipulation with a robot. direct kinematics for bin picking involves using kinematic models and transformation matrices to compute the position and orientation of the end-effector relative to the robot base in order to plan and execute precise motion for successful object manipulation with a robot.

4.4 How does Bin Picking work?

Bin picking refers to the task of autonomously selecting and retrieving objects from a bin or container. It is an important problem in the field of robotics, as it has applications in many industries, including manufacturing,

logistics, and e-commerce. Bin Picking can help to guide the design of algorithms and systems for solving this problem.

Steps for the correct development of Bin Picking:

- Perception: The first step in bin picking is to perceive the objects that are present in the bin. This involves using sensors such as cameras, depth sensors, or LiDAR to capture a 3D representation of the bin and its contents. The perception system must be able to accurately detect and localise the objects in the bin and estimate their poses and orientations.
- Planning: Once the objects in the bin have been perceived, the next step is to plan a sequence of actions that will enable the robot to retrieve the target object(s) from the bin. The planning system must take into account the positions and orientations of the objects in the bin, as well as any constraints on the robot's movements.
- Grasping: The grasping stage involves selecting an appropriate gripper or end-effector for the target object and then executing a grasp that will enable the object to be securely held and lifted from the bin. The grasping system must take into account the shape and size of the object, as well as any obstacles or other objects in the bin.
- Execution: The final stage in bin picking is executing the planned action sequence. This involves controlling the robot's movements to approach the target object, grasp it, and then move it to the desired location. The execution system must be able to handle any unforeseen obstacles or errors that may occur during the process.

In addition to these stages, there are several other factors that must be considered when designing a bin-picking system. These include:

- Object recognition and segmentation: The perception system must be able to accurately recognise and segment the objects in the bin, even if they are partially occluded or have similar appearances.
- Gripper selection: The choice of gripper or end-effector will depend on the shape, size, and weight of the target object, as well as any constraints on the robot's movements.
- Motion planning: The planning system must generate a sequence of actions that avoids collisions with other objects in the bin and ensures that the robot can safely reach the target object.
- Object pose estimation: Accurately estimating the pose and orientation of the target object is critical for planning an effective grasp and executing it successfully.
- Error recovery: Bin picking is a complex task, and errors can occur at any stage of the process. The system must be designed to handle these errors and recover from them in a safe and efficient manner.

Overall, Bin Picking should take into account the various stages involved in the process, as well as the many factors that can influence the success of the task. By considering these factors, it is possible to design more effective algorithms and systems for solving this challenging problem.

5 Development

5.1 Denavit Hartenberg

Denavit-Hartenberg Method with steps:

1. Number the links: "0" will be called the "ground", or fixed base where the robot is anchored. "1" the first moving link, etc.
2. Number the joints: "1" will be the first degree of freedom, and "n" the last.
3. Locate the axis of each joint: For torques of revolution, it will be the axis of rotation. For binoculars, it will be the axis along which the link moves.
4. Z axes: We begin to place the XYZ systems. We place the Z_{i-1} on the axes of the joints i with $i = 1$, no. That is, Z_0 goes on the axis of the 1st joint, Z_1 goes on the axis of the second degree of freedom, etc.
5. Coordinate system 0: The origin point is located at any point along Z_0. The orientation of X_0 and Y_0 can be arbitrary, as long as it obviously respects that XYZ is a right-handed system.
6. Rest of systems: For the rest of systems i = 1, N - 1, place the origin point at the intersection of Zi and Zi + 1. If the two Z axes intersect, place it at the point of intersection. If the are parallel, place it at some point of the joint + 1.
7. X axes, each Xi goes in the direction of the normal common to Zi-1 and Zi in the direction of Zi-1 towards Zi.
8. Y axes: Once the Z and X axes are located, the Y have their directions determined by the constraint of forming a clockwise XYZ.
9. Robot end system: Then the XYZ system is placed at the robot (tool) end, with its Z axis parallel to Zn - 1 and X and Y in any valid direction.
10. Angle θ: Each θi is the angle fom Xi-1 to Xi revolving around Zi.
11. Distances from: Each day is the distance from the XY system Zi - 1 to the intersection of the common normals froms Zi - 1 to Zi, along Zi - 1.
12. Distances a: Each ai is the length of said common normal.
13. Angles α: Angle that must be rotated Z_{i-1} to reach Z_i rotating around X_i.
14. Each link defines transformation matrix: i_{1Ai}
15. Total transformation: The total transformation matrix that relates the base of the robot to its tool is the chaining (multiplication) of all those matrices: $T = 0 * A_1 * A_2 ... n - 1_{An}$.

To analyse the robot, Denavit-Hartenberg's parameters were used; the analysis and the information used can be seen here:

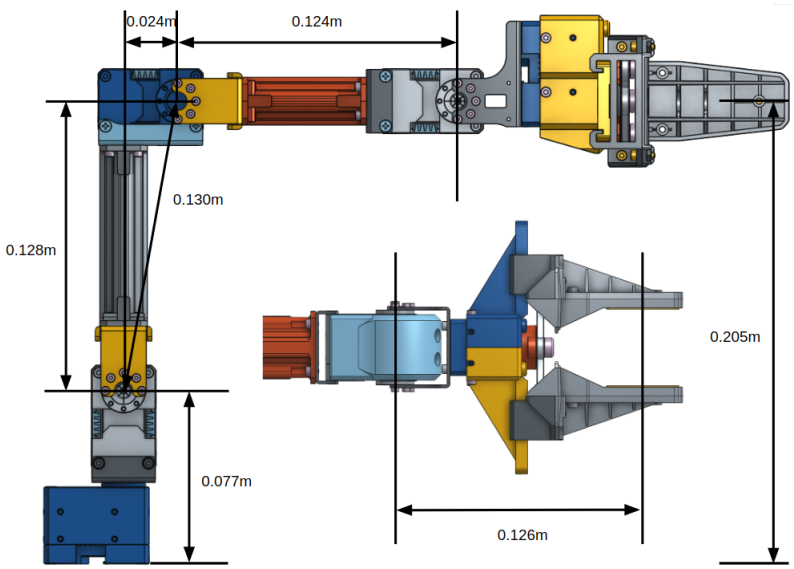

Figure 1: Hardware Specs (Robotis)

This robot cannot be analysed as a normal one, yet it can be seen that in the joint q_2 there is a triangle drawn, which refers to the length of the dynamical motor, which is not symmetrical. This means that if the first joint q_1 has turned 180º, the robot cannot reach the same point in both ways, as shown in the following picture:

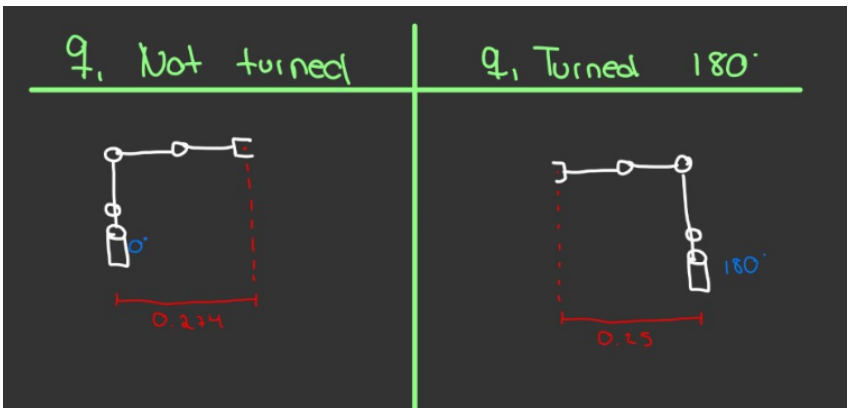

Figure 2: q_1 analysis.

The Denavit-Hartenberg analysis can be shown in the following image.

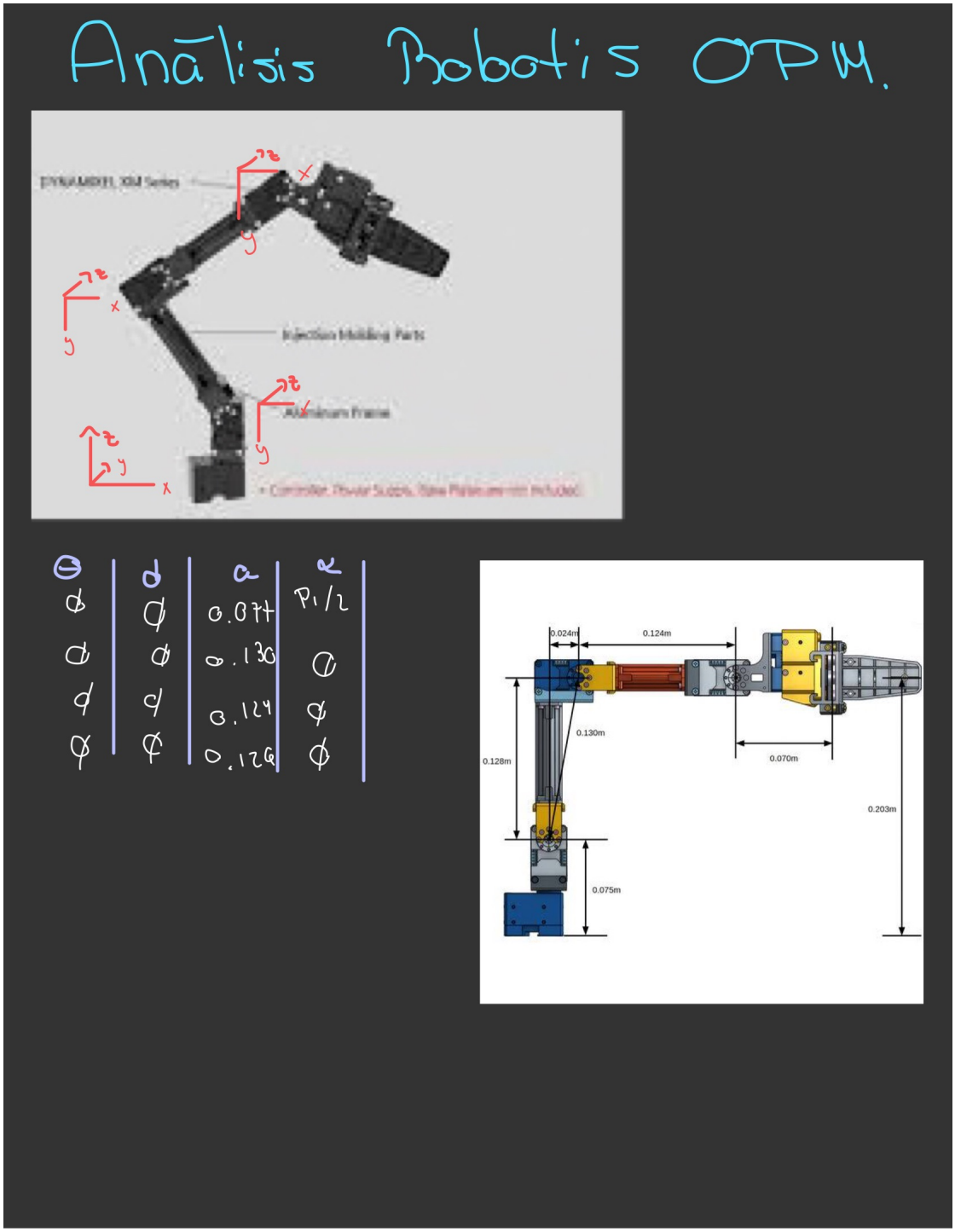

Figure 3: Denavit-Hartenberg analysis

Where the rotation and translation in the X and Z domain can be expressed in the following chart:

θ	d	a	α
0	0	0.077	$\frac{-\pi}{2}$
0	0	0.130	0
0	0	0.124	0
0	0	0.126	0

Where "θ" is the rotation in Z, "d" is Z's translation, "a" is X's translation, and "α" is X's rotation. As for the Matlab simulation and code, it can be seen here:

```
syms q1 q2 q3 q4

% crear el robot
R(1) = Link([0 0.077 0 pi/2 0]);
R(2) = Link([0 0 0.130 0 0]);
R(2).offset = 1.389282;
R(3) = Link([0 0 0.124 0 0]);
R(3).offset = -1.389282;
R(4) = Link([0 0 0.126 0 0]);

% unir eslabones y crear robot
robot_1 = SerialLink(R, "name", "RobotisX");
robot_1.teach
```

Figure 4: Denavit-Hartenberg Matlab code

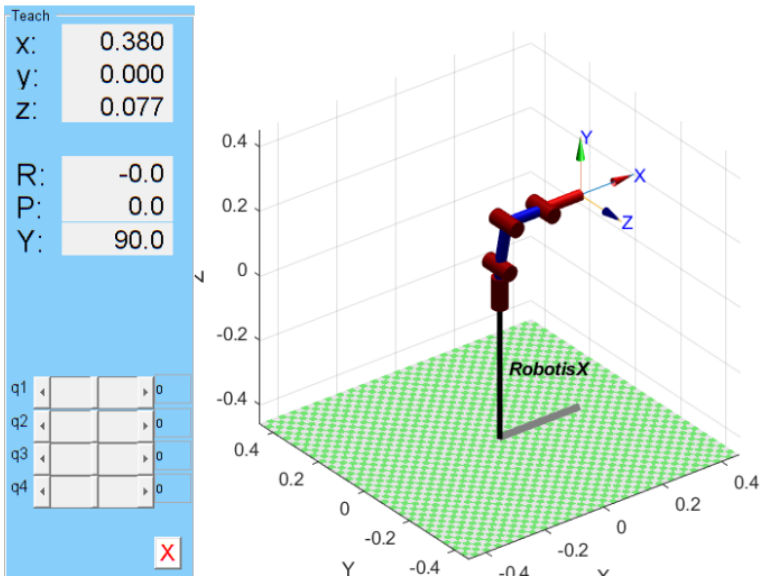

Figure 5: Denavit-Hartenberg Matlab simulation

5.2 Direct kinematics

The transformation matrices had to be made to analyse the robot's direct kinematics. The above offset angle has to be taken into account. That angle can be calculated with: $\tan^{-1}(\frac{0.024}{0.126})$ and results in 10.6197, but the angle of interest is the complementary one, $10.6197 - 90 = 79.3803$

```
% obtener cinemática directa por medio de funciones
R(1).A(q1);
R(2).A(q2);
R(3).A(q3);
R(4).A(q4);

% obtener cinemática directa de forma manual
dmxl_1 = trotz(q1) * transl([0, 0, 0.077]) * transl([0, 0, 0]) * trotx(pi/2)
dmxl_2 = trotz(q2 + deg2rad(79.3803)) * transl([0, 0, 0]) * transl([0.130, 0, 0]) * trotx(0)
dmxl_3 = trotz(q3 - deg2rad(79.3803)) * transl([0, 0, 0]) * transl([0.124, 0, 0]) * trotx(0)
dmxl_4 = trotz(q4) * transl([0, 0, 0]) * transl([0.126, 0, 0]) * trotx(0)
```

Figure 6: Direct Kinematics matrices

The matrices output is as follows:

$$\begin{pmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & \frac{77}{1000} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos\left(q_2 + \frac{6239501278081591}{4503599627370496}\right) & -\sin\left(q_2 + \frac{6239501278081591}{4503599627370496}\right) & 0 & \frac{13\cos\left(q_2+\frac{6239501278081591}{4503599627370496}\right)}{100} \\ \sin\left(q_2 + \frac{6239501278081591}{4503599627370496}\right) & \cos\left(q_2 + \frac{6239501278081591}{4503599627370496}\right) & 0 & \frac{13\sin\left(q_2+\frac{6239501278081591}{4503599627370496}\right)}{100} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos\left(q_3 - \frac{6239501278081591}{4503599627370496}\right) & -\sin\left(q_3 - \frac{6239501278081591}{4503599627370496}\right) & 0 & \frac{31\cos\left(q_3-\frac{6239501278081591}{4503599627370496}\right)}{250} \\ \sin\left(q_3 - \frac{6239501278081591}{4503599627370496}\right) & \cos\left(q_3 - \frac{6239501278081591}{4503599627370496}\right) & 0 & \frac{31\sin\left(q_3-\frac{6239501278081591}{4503599627370496}\right)}{250} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 & \frac{13\cos(q_2)}{100} \\ \sin(q_2) & \cos(q_2) & 0 & \frac{13\sin(q_2)}{100} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Using Peter Corke's command "fkine", the whole transformation matrix can be seen; this matrix has to be the same as the product of the four from the last matrices.

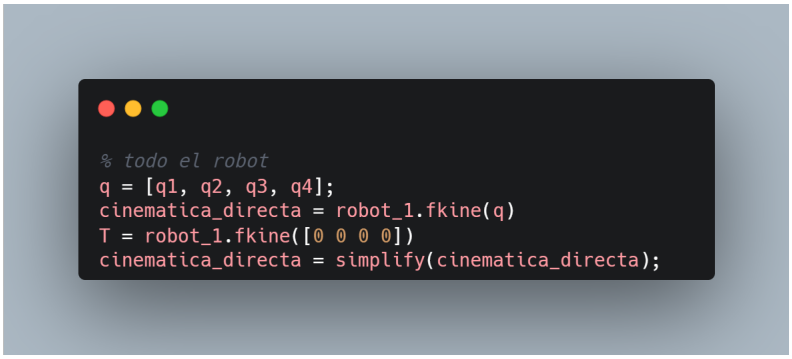

Figure 7: Direct Kinematics matrices

With an output like this one:

```
T =
[     1,     0,     0, 0.2735]
[     0,     0,    -1,      0]
[     0,     1,     0, 0.2049]
[     0,     0,     0,      1]
```

Figure 8: Direct Kinematics matrices

5.3 Inverse Kinematics

Now that the direct kinematics analysis is done, it is necessary to make an inverse kinematics analysis. This kinematics will help to determine the position that the joints need to be, in order for the tool of the robot to be in a specific position. First, it is necessary to understand how this inverse kinematics analysis is made. In this case, the method of simultaneous equations is chosen. Let's take into consideration the homogeneous transformation matrix of every joint of the robot; the steps for the method are as the following:

$$T = {}^{0}A_1 * {}^{1}A_2 * {}^{2}A_3$$

$$\left({}^{0}\mathbf{A}_1\right)^{-1}{}^{0}\mathbf{T}_3 = {}^{1}\mathbf{A}_2\,{}^{2}\mathbf{A}_3$$

$$\left({}^{1}\mathbf{A}_2\right)^{-1}\left({}^{0}\mathbf{A}_1\right)^{-1}\mathbf{T} = {}^{2}\mathbf{A}_3 =$$

Figure 9: Steps for inverse kinematics

Notice the order in which the matrix is shown in Figure 5; the direct kinematics gave a T matrix which is the transformation from the first to the final joint. What this method proposes is to go back into the transformation matrix, inverting its components one by one, thus obtaining equations that depend on q1, q2, and q3. Firstly, let us write manually the transformation matrix used for this analysis:

```
dmxl_1 = trotz(q1) * transl([0, 0, 0.077]) * transl([0, 0, 0]) *
trotx(pi/2)
dmxl_2 = trotz(q2) * transl([0, 0, 0]) * transl([0.130, 0, 0]) * trotx(0)
dmxl_3 = trotz(q3) * transl([0, 0, 0]) * transl([0.124, 0, 0]) * trotx(0)
dmxl_4 = trotz(q4) * transl([0, 0, 0]) * transl([0.126, 0, 0]) * trotx(0)
```

Figure 10: Transformation matrix for each joint

And the following matrix are the results of the transformation matrix of the code: (Note that inverse kinematics will be calculated WITHOUT the offset angle) $\begin{pmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & \frac{77}{1000} \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$$\begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 & \frac{13\cos(q_2)}{100} \\ \sin(q_2) & \cos(q_2) & 0 & \frac{13\sin(q_2)}{100} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos(q_3) & -\sin(q_3) & 0 & \frac{31\cos(q_3)}{250} \\ \sin(q_3) & \cos(q_3) & 0 & \frac{31\sin(q_3)}{250} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \cos(q_4) & -\sin(q_4) & 0 & \frac{63\cos(q_4)}{500} \\ \sin(q_4) & \cos(q_4) & 0 & \frac{63\sin(q_4)}{500} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now according to the data obtained from the direct kinematics as the positions, let's define the position desired for the tool of the robot, also the orientation extracted from the transformation matrix. It's important to declare the noap matrix, which could represent a symbolic version of our T matrix:

```
syms nx ny nz ox oy oz ax ay az px py pz

coords = [0.2735, 0, 0.2049];
px = coords(1);
py = coords(2);
pz = coords(3);
ax = 0;
ay = -1;
az = 0;

noap_ = [nx ox ax px; ny oy ay py; nz oz az pz; 0 0 0 1];
```

Figure 11: Declaring position, orientation values, and noap matrix

For the first step to achieve the inverse kinematics, let's follow the first step stated in Figure 7, multiplying the inverse matrix of the first joint by the transformation matrix which in this case will be noap's matrix; in this case, some of the values are already known, declared in the code of Figure 9:

```
p1_izq = inv(dmxl_1) * noap_;
p1_der = dmxl_2 * dmxl_3 * dmxl_4;
p1_izq = simplify(p1_izq);
p1_der = simplify(p1_der);

ec_1 = p1_izq == p1_der;

punto_1 = ec_1(3,4);
a_1 = -py;
b_1 = px;

sol_1 = [atan2(a_1, -b_1), atan2(-a_1, b_1)]
```

Figure 12: First step for inverse kinematics

Notice that the first part of the code represents the multiplication between the first joint inverse matrix and the noap matrix, this result is equalised with the product of the other joint's transformation matrix. And from this, the equation to obtain the angle q1 of the first joint can be obtained; first declaring the values of a and b for the trigonometric properties and obtaining two solutions for the first joint:

```
sol_1 = 1×2
   -3.1416         0
```

Figure 13: First step for inverse kinematics

Moving to the second step, the second joint transformation matrix has to pass to the left part of the equation, applying the inverse to it and multiplying by the noap matrix and the inverse of the first joint transformation matrix:

```
% encontrar la matriz que dependa de q1 y q2
p2_izq = inv(dmxl_2) * inv(dmxl_1) * noap_;
p2_der = dmxl_3 * dmxl_4;
p2_izq = simplify(p2_izq);
p2_der = simplify(p2_der);

% definir segunda ecuacion
ec_2 = p2_izq == p2_der;

% tomaremos las coordenadas:
punto_2 = ec_2(2,3);

% evaluar con q1 para obtener q2
eval_2 = [az * cos(q2) - ax * sin(q2), % para q1 = 0
          az * cos(q2) - ax * sin(-q2)]; % para q1 = -pi;
% evaluado sale az*cos(q2) - ax*sin(q2) = 0
a_2 = az;
b_2 = -ax;
sol_2 = [atan2(a_2, -b_2), atan2(-a_2, b_2), atan2(a_2, -b_2),
atan2(-a_2, b_2)]
```

Figure 14: Second step for inverse kinematics

Notice the first part of the code, where the inverse matrix is multiplied and both parts of the equation are equaled, the principal goal is to find an equation that is constituted into q1 and q2 terms, because q1 is already known q2 can be calculated with this equation:

ec_2 =

$$\begin{pmatrix} nz\sin(q_2)+nx\cos(q_1)\cos(q_2)+ny\cos(q_2)\sin(q_1)=\cos(q_3+q_4) & oz\sin(q_2)+ox\cos(q_1)\cos(q_2)+oy\cos(q_2)\sin(q_1)=-\sin(q_3+q_4) \\ nz\cos(q_2)-nx\cos(q_1)\sin(q_2)-ny\sin(q_1)\sin(q_2)=\sin(q_3+q_4) & oz\cos(q_2)-ox\cos(q_1)\sin(q_2)-oy\sin(q_1)\sin(q_2)=\cos(q_3+q_4) \\ nx\sin(q_1)-ny\cos(q_1)=0 & ox\sin(q_1)-oy\cos(q_1)=0 \\ 0=0 & 0=0 \end{pmatrix}$$

Figure 15: Second step equations matrix results from first part

$$\begin{pmatrix} -\cos(q_2)\sin(q_1)=0 & \frac{1279\sin(q_2)}{10000}+\frac{547\cos(q_1)\cos(q_2)}{2000}-\frac{13}{100}=\frac{63\cos(q_3+q_4)}{500}+\frac{31\cos(q_3)}{250} \\ \sin(q_1)\sin(q_2)=0 & \frac{1279\cos(q_2)}{10000}-\frac{547\cos(q_1)\sin(q_2)}{2000}=\frac{63\sin(q_3+q_4)}{500}+\frac{31\sin(q_3)}{250} \\ \cos(q_1)=1 & \frac{547\sin(q_1)}{2000}=0 \\ 0=0 & 1=1 \end{pmatrix}$$

Figure 16: Second step equations matrix's results (second part)

The equation selected for q2 is the equation (2,3), because of the simplicity and because it is in terms of q1 and q2, so taking this point, and evaluating the values of the orientation; using trigonometry properties, the following results are given:

punto_2 = $\sin(q_1)\sin(q_2)=0$

eval_2 =

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

sol_2 = 1×4

0	0	0	0

Figure 17: Solution for q2 evaluation ec2 with q1

For the third and last step, we have to pass the almost last joint inverse matrix to the left and extract q3 with the equation that the matrix has:

```
% encontrar la matriz que dependa de q1, q2 y q3
p3_izq = inv(dmxl_3) * inv(dmxl_2) * inv(dmxl_1) * noap_;
p3_der = dmxl_4;
p3_izq = simplify(p3_izq);
p3_der = simplify(p3_der);

% definir tercera ecuacion
ec_3 = p3_izq == p3_der

% tomaremos las coordenadas: (3,4)

punto_3 = simplify(ec_3(2,3));

% evaluar con q1 para obtener q3
eval_3 = [az*cos(0)*cos(q3) - az*sin(0)*sin(q3) -
ax*cos(0)*cos(0)*sin(q3) - ax*cos(0)*cos(q3)*sin(0) -
ay*cos(0)*sin(0)*sin(q3) - ay*cos(q3)*sin(0)*sin(0)]
% evaluar
a_3 = az;
b_3 = -ax;
sol_3 = [atan2(a_3, -b_3), atan2(-a_3, b_3)]
```

Figure 18: Third step for the inverse matrix

Notice that the first part of the code is similar to the other steps, we are applying this to the almost last joint of the robot equalising all the inverse products of the matrix to the last joint of the system:

ec_3 =

$$\begin{pmatrix} nz\cos(q_2)\sin(q_3) + nz\cos(q_3)\sin(q_2) + nx\cos(q_1)\cos(q_2)\cos(q_3) + ny\cos(q_2)\cos(q_3)\sin(q_1) - nx\cos(q_1)\sin(q_2)\sin(q_3) - ny\sin(q_1)\sin(q_2)\sin(q_3) = \cos(q_4) \\ nz\cos(q_2)\cos(q_3) - nz\sin(q_2)\sin(q_3) - nx\cos(q_1)\cos(q_2)\sin(q_3) - nx\cos(q_1)\cos(q_3)\sin(q_2) - ny\cos(q_2)\sin(q_1)\sin(q_3) - ny\cos(q_3)\sin(q_1)\sin(q_2) = \sin(q_4) \\ nx\sin(q_1) - ny\cos(q_1) = 0 \\ 0 = 0 \end{pmatrix}$$

Figure 19: Equation matrix for the third equation first part

$$\text{oz}\cos(q_2)\sin(q_3) + \text{oz}\cos(q_3)\sin(q_2) + \text{ox}\cos(q_1)\cos(q_2)\cos(q_3) + \text{oy}\cos(q_2)\cos(q_3)\sin(q_1) - \text{ox}\cos(q_1)\sin(q_2)\sin(q_3) - \text{oy}\sin(q_1)\sin(q_2)\sin(q_3) = -\sin(q_4)$$

$$\text{oz}\cos(q_2)\cos(q_3) - \text{oz}\sin(q_2)\sin(q_3) - \text{ox}\cos(q_1)\cos(q_2)\sin(q_3) - \text{ox}\cos(q_1)\cos(q_3)\sin(q_2) - \text{oy}\cos(q_2)\sin(q_1)\sin(q_3) - \text{oy}\cos(q_3)\sin(q_1)\sin(q_2) = \cos(q_4)$$

$$\text{ox}\sin(q_1) - \text{oy}\cos(q_1) = 0$$

$$0 = 0$$

Figure 20: Equation matrix for the third equation second part

$$\begin{pmatrix} -\cos(q_2+q_3)\sin(q_1) = 0 & \frac{1279\cos(q_2)\sin(q_3)}{10000} - \frac{13\cos(q_3)}{100} + \frac{1279\cos(q_3)\sin(q_2)}{10000} + \frac{547\cos(q_1)\cos(q_2)\cos(q_3)}{2000} - \frac{547\cos(q_1)\sin(q_2)\sin(q_3)}{2000} - \frac{31}{250} = \frac{63\cos(q_4)}{500} \\ \sin(q_2+q_3)\sin(q_1) = 0 & \frac{13\sin(q_3)}{100} + \frac{1279\cos(q_2)\cos(q_3)}{10000} - \frac{1279\sin(q_2)\sin(q_3)}{10000} - \frac{547\cos(q_1)\cos(q_2)\sin(q_3)}{2000} - \frac{547\cos(q_1)\cos(q_3)\sin(q_2)}{2000} = \frac{63\sin(q_4)}{500} \\ \cos(q_1) = 1 & \frac{547\sin(q_1)}{2000} = 0 \\ 0 = 0 & 1 = 1 \end{pmatrix}$$

Figure 21: Equation matrix for the third equation third part

For the code of Figure 16, we establish the desired equation for the solutions to q3, using trigonometric properties and evaluating the values of the q1 and q2 solutions, we could get the result:

```
eval_3 = 0


sol_3 = 1×2
     0     0
```

Figure 22: Results for q3 evaluating q1 and q2

Angles of q1, q2 and q3 are now resolved, the equation required for q4 can be extracted with the matrix from step three.

```
% tomaremos las coordenadas: (1,4)
q1 = 0;
q2 = q2 + 1.3892;
q3 =q3 - 1.3892;

punto_4 = simplify(ec_3(1,4))
punto_4 = eval(punto_4)
punto_4 = simplify(punto_4)

% evaluar con q1 para obtener q4
eval_4 = []

% evaluar
b_4 = 10/63;
sol_4 = [atan2(sqrt(1-b_4^2), b_4), atan2(-sqrt(1-b_4^2), b_4)]
rad2deg(sol_4)
```

Figure 23: Code to obtain q4

In the first part of the code in Figure 21 we are adding the offset angle of the second and third joint, because of this offset angle, the final joint has to correct the course in order to arrive at the desired point:

$$\texttt{punto\_4} = 1300\cos(q_3) + 1260\cos(q_4) + 2735\cos(q_1)\sin(q_2)\sin(q_3) + 1240 = 1279\cos(q_2)\sin(q_3) + 1279\cos(q_3)\sin(q_2) + 2735\cos(q_1)\cos(q_2)\cos(q_3)$$

$$\texttt{punto\_4} =$$

$$1260\cos(q_4) - \frac{10300352721658129}{8796093022208} = \frac{6277287927730065}{70368744177664}$$

$$\texttt{punto\_4} =$$

$$\cos(q_4) = \frac{88680109700995097}{88664617663856640}$$

```
eval_4 =

     []

sol_4 = 1×2
     0     0
```

Figure 24: q4 output

5.3.1 Inverse Kinemtics with Software

Peter Corke's toolbox has a way to obtain the inverse kinematics with pure software, to understand how this works, an SE3 has to be passed as the first input, the second important input is the mask, it establishes how many degrees of freedom (DOF) the robot will have, in this case, it has 4, so the mask is [111100] with the mask always having a length of 6 and the number different than 0 are the ammount of joints.

Figure 25: Inverse kinematics with Peter Corke's toolbox

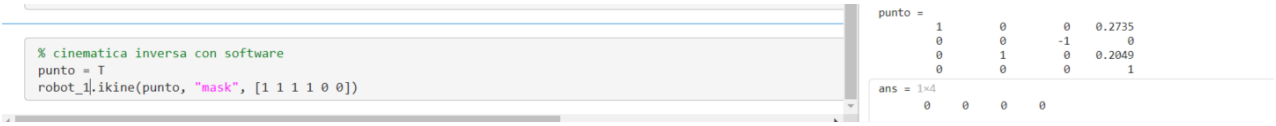

Figure 26: Inverse kinematics with Peter Corke's toolbox confirmation

5.3.2 Inverse Kinematics verification

To verify that the calculations were correct, the robot was plotted with the values given for the specific 0 point: [0.2735, 0, 0.2049] being the joint values: [0, 1.389282, -1.389282, 0]. Note that these values only work for a robot with NO OFFSET, meaning that the offset has to be set manually whenever a calculation is made, the following image and code will show the verification.

```
% crear el robot
R(1) = Link([0 0.077 0 pi/2 0]);
R(2) = Link([0 0 0.130 0 0]);
%R(2).offset = 1.389282;
R(3) = Link([0 0 0.124 0 0]);
%R(3).offset = -1.389282;
R(4) = Link([0 0 0.126 0 0]);

% unir eslabones y crear robot
robot_2 = SerialLink(R, "name", "RobotisX");

q1 = 0;
q2 = 1.389282;
q3 = -1.389282;
q4 = 0;
robot_2.plot([q1, q2, q3, q4])
```

Figure 27: Verification code.

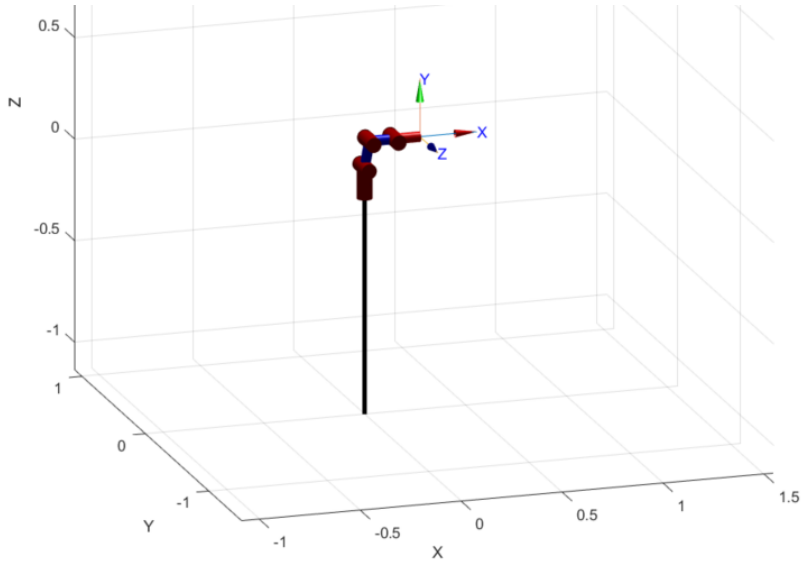

Figure 28: Robot with the angles of the joints pointing to the 0 position.

5.4 Problem solution approach

To resolve the approach to the solution of the problem, we are going to use the Matlab software where we will develop the elementary part of the project, the results will be obtained through the analysis of direct

and inverse kinematics where they will be ordered according to the required format.

Using the steps of Denavit-Hartenberg was how the approach to the solution of our first objective began, for this the type of analysis used in this robot was different from any other since in the design analysis of its joints, it was possible to realise that when It is not possible to reach a reference in two ways since it is not symmetrical. Combining in the analysis of the robot, the analysis of Denavit-Hartenberg allowed us to develop a code in Matlab that allows simulating a rotation in the X, Y, and Z axes, with this achieved, it was possible to continue with the inverse kinematics.

With the help of Peter Corke's ToolBox, the transformation matrices were made to have an analysis of what inverse kinematics is.

After completing the direct kinematics analysis, the next step is to perform an inverse kinematics analysis to determine the required joint positions for the robot tool to be in a specific position. The chosen method for this robot is the method of the simultaneous equation. To perform this analysis, the homogeneous transformation matrix for each joint must be considered. The process involves going backwards through the transformation matrix, inverting one matrix at a time to obtain equations that depend on q1, q2, and q3. The order of the matrices is important, as the direct kinematics analysis provided a T matrix that transforms from the first to the final joint. The transformation matrix used for this analysis can be written manually.

In the initial section of the code, the focus is on multiplying the inverse matrices and equating both sides of the equation. The primary objective is to derive an equation that consists of q1 and q2 terms. Since the value of q1 is already known, the equation can be utilised to calculate q2.

In the final step of inverse kinematics, the second-to-last joint inverse matrix needs to be transferred to the left side, and q3 can be obtained using the equation present in the matrix. This step represents the third and final stage of the process.

To confirm the accuracy of the computations, the robot was graphed using the specified values. However, it should be noted that these values are only applicable to a robot that has no offset. Therefore, whenever a calculation is performed, the offset needs to be manually configured. The verification process will be demonstrated through the accompanying code and image.

5.5 Workspace

For the workspace of the robotic arm, a neural network was chosen and implemented on a specific piece of hardware it is "- - - -" This is because, due to the nature of the task, the robot does not initially know the position of the object it has to manipulate; it knows the general direction from which it will be presented; however, the distance and orientation is unknown and has to be measured after the object's presentation.

A camera will be mounted on a stand above the robot, pointing downwards for it to have a bi-dimensional view of the table/workspace of the robot. This way, the camera will not be affected by the movement of the arm or the movement of the presentation of the objects to be manipulated. Furthermore, in this way, it may be easier for the camera to present the data to the computer so it can translate the reference axis of the object concerning the world axis, which is at the base of the arm.

Finally, the limitations of the workspace will be calculated in the following way according to the previously calculated DH parameters: The code graphs the border limits of the workspace, meaning the maximum reach the robot has in the x, y, and z axis: Limits for y and x:

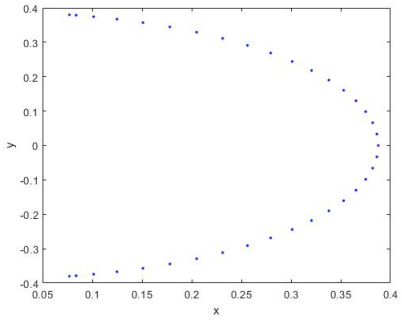

Figure 29: work-space's X and Y axis limits

limit for z= 0.38m Code:

```
clc
clear all
%% DH
R(1) = Link([0 0.077 0 pi/2 0], 'standard');
R(2) = Link([0 0 0.130 0 0],'standard');
R(2).offset = 1.389282;
R(3) = Link([0 0 0.124 0 0],'standard');
R(3).offset = -1.389282;
R(4) = Link([0 0 0.126 0 0],'standard');

KR= SerialLink (R);

conv= pi/180;

i=0;
%% angular joint freedom in degrees
for q1= -360*conv:5*conv:360*conv
    for q2= -180*conv:5*conv:180*conv
        for q3= -90*conv:5*conv:90*conv
            for q4= -90*conv:5*conv:90*conv
%% homogeneous matrices
             aAb=trotz(q1)*transl(0, 0, 0.077)*transl(0, 0, 0)*trotx(pi/2);
             bAc=trotz(q2)*transl(0, 0, 0)*transl(0, 0, 0.130)*trotx(0);
             cAd=trotz(q3)*transl(0, 0, 0)*transl(0, 0, 0.124)*trotx(0);
             dAe=trotz(q4)*transl(0, 0, 0)*transl(0, 0, 0.126)*trotx(0);
             aAe= aAb*bAc*cAd*dAe;
             i=i+1;
             %% points
              p=aAe(1:4,4);
              p1(i)=p(1);
              p2(i)=p(2);
              p3(i)=p(3);
        end
      end
    end
end
%% figures
figure(2)
plot ((p2.^2+p1.^2).^0.5,p3,'b.');
xlabel('x')
ylabel('z')

figure(3)
plot ((p1.^2+p3.^2).^0.5,p2,'b.');
xlabel('x')
ylabel('y')
```

Figure 30: work-space's code

As for the absolute workspace, the one that the team will work with is the base that it has, there's a square base that will mark the limits of the workspace, this being 30cm wide and 30cm long. With a height of 48cm.

6 Conclusions

As it was shown in this project the implementation of neural networks alongside vision enabling hardware opens the possibility to rely ever more often on robotic arms to carry on processes which could result repetitive or tiresome for human workers thus enabling workers to implement their time and resources in more dearing and complex projects. furthermore a none blind robotic arm can perform tasks like bin-picking faster than a worker could thus incrementing production for a manufacturing line.

To wrap it up we can conclude that robotic automatisation is an incredibly important tool which constantly proves it's value due to its ability to make processes more efficient or take control of processes altogether thus improving the quality of life for workers and consumers alike.

References

[1] Robotis. "ROBOTIS E-Manual." ROBOTIS E-Manual, `emanual.robotis.com/docs/en/platform/openmanipulator_x/specification/#hardware-specification`. Accessed 13 Apr. 2023.

[2] Russell. *S*urvey of bin-picking approaches using depth sensors. Robotics and Autonomous Systems, 121, 30-40.