Technical University of Denmark

34753 - Robotics

Project assignment: Educational Robotics Arm

Group 22:
Anna Myken (s203857) [illegible]
Christian Casarotto (s223302) [illegible]
Christoffer Kofoed Christensen (s205186) [illegible]
Francesco Centomo (s223588) [illegible]
Jacob Sangkoyo Gramtorp (s153036) [illegible]

Fall, 2023

Introduction

This report concerns the educational robotics arm in the course ”Robotics” (course no. 34753). All problems have been completed as a joint effort of all five group members, which is also stated at each problem.

Part 1: Kinematics

Solving the kinematics of a robot is the first step in being able to control it effectively. To achieve this, mathematical methods such as the Denavit-Hartenberg convention are used to describe the spatial relationships between different parts of the robot. This convention provides a systematic way to represent the transformation matrices for each joint, allowing the user to understand and compute the position and orientation of the robot’s end effector.

Problem 1

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

Using figure one from the assignment, it is possible to extract the characteristic Denavit-Hartenberg parameters for the given robot and for the given reference frames. This is done in matrix 1, where the last element of the robot is in this case its stylus.
In this first part of the report, the angles will be referred to as θ_0, θ_1, θ_2 and θ_3 and the lengths of the robot arm are indicated as L_0, L_1, L_2 and L_3.

$$DH = \begin{pmatrix} \theta_0 & 50 & 0 & \frac{\pi}{2} \\ \theta_1 & 0 & 93 & 0 \\ \theta_2 & 0 & 93 & 0 \\ \theta_3 & 0 & 50 & 0 \end{pmatrix} \tag{1}$$

The resulting forward kinematic can be seen in 2. The initial or world frame is expressed with *0*, and the end effector frame is therefore *4*. Also, a short notation for sinus and cosines of the angles is adopted in the considered equations, and this will be used throughout the whole report.

$$T_4^0 = \begin{bmatrix} c_{123}c_0 & -s_{123}c_0 & s_0 & c_0\left(50c_{123} + 93c_{12} + 93c_1\right) \\ c_{123}s_0 & -s_{123}s_0 & -c_0 & -s_0\left(50c_{123} + 93c_{12} + 93c_1\right) \\ s_{123} & c_{123} & 0 & 50s_{123} + 93s_{12} + 93s_1 + 50 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Due to the fact that the camera position included a translation along the y axes, it was not possible to use the Denavit-Hartenberg convention to describe its position. This meant that a simply homogeneous matrix, T_5^4 in equation 3, will be multiplied to matrix 2 to describe the camera position. This is done, obtaining the matrix T_5^0 in equation 4.

$$T_5^4 = \begin{pmatrix} 1.0 & 0 & 0 & -15.0 \\ 0 & 1.0 & 0 & 45.0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \tag{3}$$

$$T_5^0 = \begin{bmatrix} c_{123}c_0 & -s_{123}c_0 & s_0 & c_0\left(35c_{123} + -45s_{123} + 93c_{12} + 93c_1\right) \\ c_{123}s_0 & -s_{123}s_0 & -c_0 & -s_0\left(35c_{123} - 45s_{123} + 93c_{12} + 93c_1\right) \\ s_{123} & c_{123} & 0 & 45s_{123} + 35s_{123} + 93s_{12} + 93s_1 + 50 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Problem 2

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

For the inverse kinematics function $q = f(x_4^0, o_4^0)$ the solution is found geometrically.
The process start by splitting the problem into 3 sub-problems:

1. Finding the algebraic expression for θ_1 and θ_2 by solving the inverse kinematics of a RR arm in 2D space.
2. Finding the algebraic expression for θ_3 by inserting ϕ as the angle of end-effector in the home coordinate system.
3. Finding the algebraic expression for θ_0 and thereby converting the function from 2D to 3D space.

θ_1 and θ_2

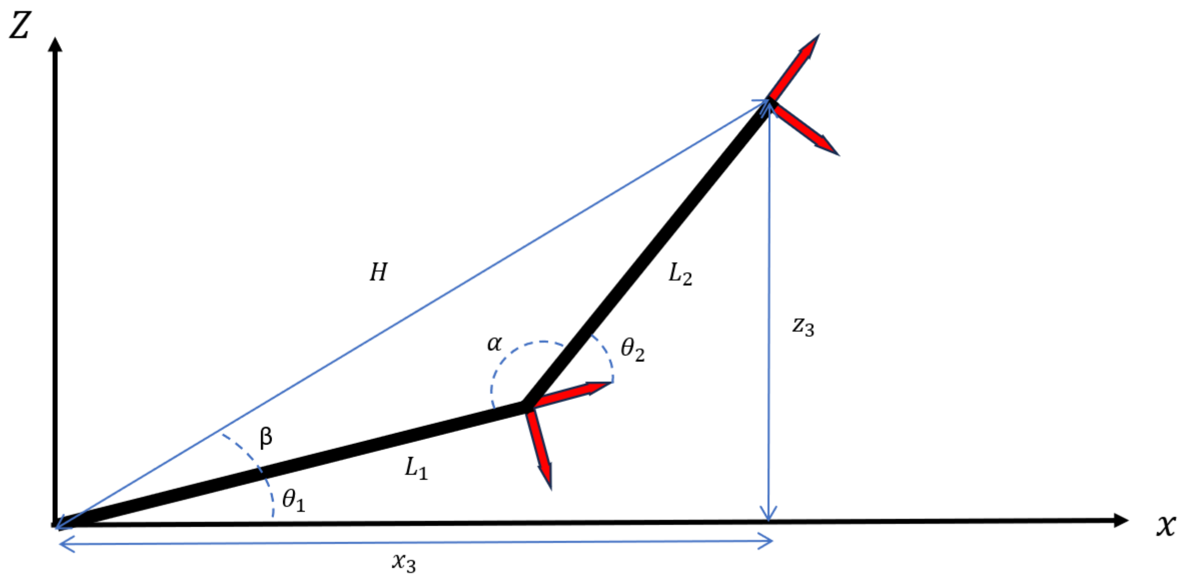

Figure 1: RR robot arm

Finding θ_2
Using Pythagoras theorem and law of cosines the following expression are set up for the angle α.

$$\cos(\alpha) = \frac{L_1^2 + L_2^2 - x_3^2 - z_3^2}{2L_1L_2} \tag{5}$$

From figure fig. 1 the following relationship between α and θ are derived.

$$\theta_2 + \alpha = \pi \tag{6}$$

$$\cos(\theta_2) = \cos(\pi - \alpha) = \cos(\pi)\cos(\alpha) + \sin(\pi)\sin(\alpha) = -\cos(\alpha) \tag{7}$$

There are two solutions for the above (8) and (9)

$$\theta_2 = \cos^{-1}\left(\frac{x_3^2 + z_3^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \tag{8}$$

$$\theta_2 = \pi - \cos^{-1}\left(\frac{L_1^2 + L_2^2 - x_3^2 - z_3^2}{2L_1L_2}\right) \tag{9}$$

The solution in 8 is chosen. However the arccos gives two solutions for every angle of θ_2 making it impossible control the robot when ex. $\cos \pm\pi/4$ gives the same result.
Instead one can use $\theta_2 = atan2(\sin\theta_2, \cos\theta_2)$ and the relation: $\cos^2\theta + \sin^2\theta = 1 \Rightarrow \sin\theta = \pm\sqrt{1-\cos^2\theta}$ to get the expression for θ_2 (10)

$$\theta_2 = \operatorname{atan2}\left(\pm\sqrt{1-\cos^2\left(\frac{x_3^2 + z_3^2 - L_1^2 - L_2^2}{2L_1L_2}\right)}, \frac{x_3^2 + z_3^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \tag{10}$$

Notice the $\pm$ in the expression gives two solutions for the angle, a positive and a negative solution. These result corresponds to two configurations of the robot arm: the "elbow up" configuration if θ_2 is negative and the "elbow down" configuration when θ_2 is positive.
Finding θ_1:
Again looking at fig.1 it is evident that the relationship between θ_1 and the position of the second link is given by:

$$\tan(\theta_1 - \beta) = \left(\frac{z}{x}\right) \tag{11}$$

$$\theta_1 = \tan^{-1}\left(\frac{z}{x}\right) - \beta \tag{12}$$

An expression for β can easily be found since θ_2 is known:

$$\beta = \tan^{-1}\left(\frac{L_2\sin(\theta_2)}{L_1 + L_2\cos(\theta_2)}\right) \tag{13}$$

The final expression for θ_1

$$\theta_1 = \tan^{-1}\left(\frac{z}{x}\right) - \tan^{-1}\left(\frac{L_2\sin(\theta_2)}{L_1 + L_2\cos(\theta_2)}\right) \tag{14}$$

θ_3

Adding the link L_3 makes the arm redundant as it now has three degrees of freedom, moving in a 2D space. Thus the arm can reach any point within its workspace with an near infinite different configurations. To define a specific configuration of the robot arm, a new variable is introduced to the inverse kinematic function: ϕ which is the angle of the stylus in the home coordinate system.

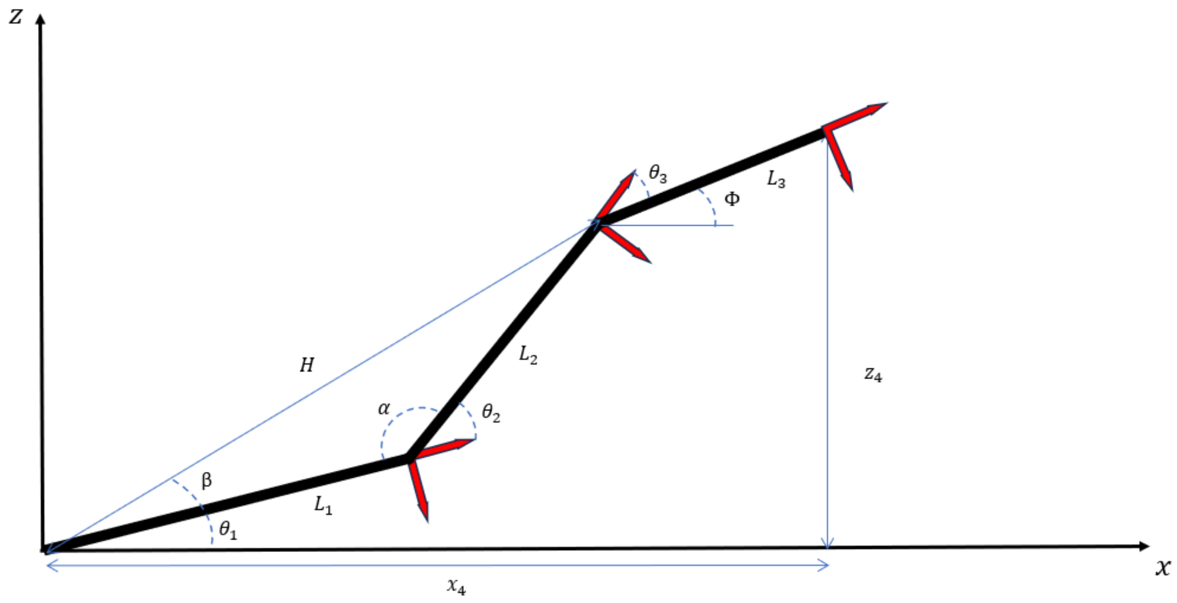

Figure 2: RR robot arm

In fig. 2 the three linked robot arm are represented. The relationship between the angle of the robot links can be manipulated to an expression for θ_3:

$$\theta_3 = \phi - \theta_1 - \theta_2 \tag{15}$$

θ_0

Since θ_0 is the only variable angle rotating around the z-axes of the origin coordinate system, it can simply be defined as:

$$\theta_0 = \text{atan2}\left(y_4, x_4\right) \tag{16}$$

Where y_4 and x_4 are the coordinates of the end effector point in the xy-plane.

Building the function

Now the inverse kinematics is solved for all motor angles of the robot arm. The next task is to set up the function described in the problem description:

$$q = [q_1, q_2, q_3, q_4]^T = f(x_4^0, o_4^0) \tag{17}$$

The first input of the function is x_4^0 from which only the last element needs to be satisfied. Looking at (2) it is possible to see that this element is:

$$\cos\phi = \cos\theta_1 + \theta_2 + \theta_3 \tag{18}$$

Thus one simply needs to input $cos\phi = x_4^0(3)$

$$\phi = atan2(\sqrt{1 - \cos x_4^0(3)}^2, \cos x_4^0(3)) \tag{19}$$

For the second input of the function the position of the point of the stylus is known. This relates to the geometric point a follows:

$$o_4^0 = \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} \tag{20}$$

As the angles $[\theta_1, \theta_2]$ are found from x_3 and z_3, these need to be found from the input.

$$x_3 = \sqrt{(o_4^0(1)^2 + o_4^0(2)^2)} - L_3 cos(\theta_1 + \theta_2 + \theta_3) \tag{21}$$

$$z_3 = o_4^0(3) - L_3 sin(\theta_1 + \theta_2 + \theta_3) - L_0 \tag{22}$$

Notice in (22) the first link of the robot arm: L_0 was subtracted. That is because this link was not included when the inverse kinematics was solved earlier, as the home coordinate system was placed at θ_1. The above expressions is now inserted into (10), (14),(15) and (16) to get (17).

$$f\left(x_4^0(3), o_4^0\right) = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \Rightarrow f\left(\phi, \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix}\right) = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(\sqrt{1-\cos(y_4)}, \cos(x_4)\right) \\ \tan^{-1}\left(\frac{z_3}{x_3}\right) - \tan^{-1}\left(\frac{L_2\sin(\theta_2)}{L_1+L_2\cos(\theta_2)}\right) - \frac{\pi}{2} \\ \text{atan2}\left(\pm\sqrt{1-\cos^2\left(\frac{{x_3}^2+{z_3}^2-L_1^2-L_2^2}{2L_1L_2}\right)}, \frac{{x_3}^2+{z_3}^2-L_1^2-L_2^2}{2L_1L_2}\right) \\ \phi - \theta_1 - \theta_2 \end{bmatrix} \tag{23}$$

Problem 3

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

Found the inverse kinematics associated with the robot arm, it is now possible to find the robot configurations in terms of joint angles associated with a certain end effector position. For example the robot could track in the 3D space a circle with its stylus tip. A circle with center p_c^0 and radius R is given by equation 24.

$$p^0(\varphi) = p_c^0 + R \begin{bmatrix} 0 \\ \cos(\varphi) \\ \sin(\varphi) \end{bmatrix} \tag{24}$$

The needed variables for the circle definition are collected below:

$$q^{(j)} = f(x_4^0 = [?,?,0], o_4^0 = p^0(\varphi_j)), R = 32, p_c^0 = [150, 0, 120]^T, j = 0, 1, ..., 36 \tag{25}$$

The process consists first in finding the end effector coordinates to compute the circle, keeping the end effector horizontal. Having then the needed sets of final coordinates for the 36 equidistant points and the stylus horizontal $x_4^0 = [?,?,0]$ it is possible to compute the inverse kinematics of the robot for each configuration. The resulting sets of robot configurations are collected in table 1.

$x \cdot 2\pi/36$	q_1	q_2	q_3	q_4
0	0.2102	1.4302	-1.6699	0.2397
1	0.2071	1.4437	-1.6242	0.1805
2	0.1978	1.4556	-1.5789	0.1233

$x \cdot 2\pi/36$	q_1	q_2	q_3	q_4
3	0.1827	1.4656	-1.5354	0.0698
4	0.1620	1.4737	-1.4952	0.0214
5	0.1363	1.4801	-1.4597	-0.0204
6	0.1063	1.4848	-1.4304	-0.0544
7	0.0728	1.4880	-1.4084	-0.0795
8	0.0370	1.4898	-1.3949	-0.0950
9	0.0000	1.4904	-1.3903	-0.1002
10	-0.0370	1.4898	-1.3949	-0.0950
11	-0.0728	1.4880	-1.4084	-0.0795
12	-0.1063	1.4848	-1.4304	-0.0544
13	-0.1363	1.4801	-1.4597	-0.0204
14	-0.1620	1.4737	-1.4952	0.0214
15	-0.1827	1.4656	-1.5354	0.0698
16	-0.1978	1.4556	-1.5789	0.1233
17	-0.2071	1.4437	-1.6242	0.1805
18	-0.2102	1.4302	-1.6699	0.2397
19	-0.2071	1.4152	-1.7146	0.2994
20	-0.1978	1.3992	-1.7571	0.3579
21	-0.1827	1.3829	-1.7963	0.4134
22	-0.1620	1.3669	-1.8311	0.4643
23	-0.1363	1.3521	-1.8608	0.5087
24	-0.1063	1.3393	-1.8846	0.5452
25	-0.0728	1.3295	-1.9019	0.5724
26	-0.0370	1.3233	-1.9125	0.5892
27	-0.0000	1.3212	-1.9160	0.5949
28	0.0370	1.3233	-1.9125	0.5892
29	0.0728	1.3295	-1.9019	0.5724
30	0.1063	1.3393	-1.8846	0.5452
31	0.1363	1.3521	-1.8608	0.5087
32	0.1620	1.3669	-1.8311	0.4643
33	0.1827	1.3829	-1.7963	0.4134
34	0.1978	1.3992	-1.7571	0.3579
35	0.2071	1.4152	-1.7146	0.2994
36	0.2102	1.4302	-1.6699	0.2397

Table 1: Problem 3 robot configurations

Problem 4

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

The objective of this problem was to determine the Jacobian for the robot end-effector as well as the Jacobian for the robot camera. From now on in the report the angles will be referred to as θ_1, θ_2, θ_3 and θ_4.

Determining the Jacobian for the end-effector

In order to determine the Jacobian for the end-effector, the positions of origin $o_0, o_1, o_2, o_3, o_4 = o_n$ were necessary as well as the z-axes z_0, z_1, z_2 and z_3.
The positions of origin were found by determining the transformation matrices T_1^0,T_2^1,T_3^2 and T_4^3 and then extracting the points of origin from the last columns of these. This process resulted in the following four sets of coordinates:

$$o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, o_1 = \begin{bmatrix} 0 \\ 0 \\ 50 \end{bmatrix}, o_2 = \begin{pmatrix} 93\,\cos(\theta_1)\,\cos(\theta_2) \\ 93\,\cos(\theta_2)\,\sin(\theta_1) \\ 93\,\sin(\theta_2) + 50 \end{pmatrix} \tag{26}$$

$$o_3 = \begin{pmatrix} 93\cos(\theta_1)(\cos(\theta_2+\theta_3)+\cos(\theta_2)) \\ 93\sin(\theta_1)(\cos(\theta_2+\theta_3)+\cos(\theta_2)) \\ 93\sin(\theta_2+\theta_3)+93\sin(\theta_2)+50 \end{pmatrix} \tag{27}$$

$$o_4 = o_n = \begin{pmatrix} \cos(\theta_1)\,(50\,\cos(\theta_2+\theta_3+\theta_4)+93\,\cos(\theta_2+\theta_3)+93\,\cos(\theta_2)) \\ \sin(\theta_1)\,(50\,\cos(\theta_2+\theta_3+\theta_4)+93\,\cos(\theta_2+\theta_3)+93\,\cos(\theta_2)) \\ 50\,\sin(\theta_2+\theta_3+\theta_4)+93\,\sin(\theta_2+\theta_3)+93\,\sin(\theta_2)+50 \end{pmatrix} \tag{28}$$

Since the only occurring rotation around the x-axis happens when going from frame {0} to frame {1}, the vectors z_2 and z_3 were necessarily equal to the z_1 vector. The necessary z-axes were calculated as shown:

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad z_1 = z_2 = z_3 = \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \tag{29}$$

Since the four joints of the robot were all revolute, the four columns of the Jacobian could now be determined as shown below. The position of the origins are omitted as already presented in equations 26 to 26.

$$J_1 = \begin{bmatrix} z_0 \times (o_4 - o_0) \\ z_0 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (o_n - o_0) \\ \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \tag{30}$$

$$J_2 = \begin{bmatrix} z_1 \times (o_4 - o_1) \\ z_1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \times (o_n - o_1) \\ \\ \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \end{bmatrix} \tag{31}$$

$$J_3 = \begin{bmatrix} z_2 \times (o_4 - o_2) \\ z_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \times (o_n - o_2) \\ \\ \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \end{bmatrix} \tag{32}$$

$$J_4 = \begin{bmatrix} z_3 \times (o_4 - o_3) \\ z_3 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \times (o_n - o_3) \\ \\ \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} \end{bmatrix} \tag{33}$$

$$J_1 = \begin{pmatrix} -\sin(\theta_1)\ (50\cos(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2)) \\ \cos(\theta_1)\ (50\cos(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2)) \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{34}$$

$$J_2 = \begin{pmatrix} -\cos(\theta_1)\ (50\sin(\theta_2+\theta_3+\theta_4) + 93\sin(\theta_2+\theta_3) + 93\sin(\theta_2)) \\ -\sin(\theta_1)\ (50\sin(\theta_2+\theta_3+\theta_4) + 93\sin(\theta_2+\theta_3) + 93\sin(\theta_2)) \\ 50\cos(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2) \\ \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{pmatrix} \tag{35}$$

$$J_3 = \begin{pmatrix} -\cos(\theta_1)\ (50\sin(\theta_2+\theta_3+\theta_4) + 93\sin(\theta_2+\theta_3)) \\ -\sin(\theta_1)\ (50\sin(\theta_2+\theta_3+\theta_4) + 93\sin(\theta_2+\theta_3)) \\ 50\cos(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) \\ \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{pmatrix} \tag{36}$$

$$J_4 = \begin{pmatrix} -50\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_1) \\ -50\sin(\theta_2+\theta_3+\theta_4)\sin(\theta_1) \\ 50\cos(\theta_2+\theta_3+\theta_4) \\ \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{pmatrix} \tag{37}$$

In order to set up the final Jacobian for the end effector the four columns were appended as follows:

$$J = \begin{bmatrix} J1 & J2 & J3 & J4 \end{bmatrix} \tag{38}$$

Determining the Jacobian for the camera

The process of determining the Jacobian for the camera was similar to the process of determining the Jacobian for the stylus, the only difference being in the components $o_4 = o_n$, due to the nature of the transformation matrix from frame 0 to frame 4 (T_4^0) being different in the case where the end effector is the camera. The position of origin $o_4 = o_n$ was found by extracting the last column of the transformation matrix T_4^0, which resulted in the following:

$$o_n = o_4 = \begin{pmatrix} \cos(\theta_1)\ (35\cos(\theta_2+\theta_3+\theta_4) - 45\sin(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2)) \\ \sin(\theta_1)\ (35\cos(\theta_2+\theta_3+\theta_4) - 45\sin(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2)) \\ 93\sin(\theta_2+\theta_3) + 93\sin(\theta_2) + 5\sqrt{130}\cos\left(\theta_2+\theta_3+\theta_4 - \operatorname{atan}\left(\frac{7}{9}\right)\right) + 50 \end{pmatrix} \tag{39}$$

The result of this point changing propagates through all the calculations, resulting in the following four columns of the Jacobian matrix:

$$J1 = \begin{pmatrix} -\sin(\theta_1)\ (35\cos(\theta_2+\theta_3+\theta_4) - 45\sin(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2)) \\ \cos(\theta_1)\ (35\cos(\theta_2+\theta_3+\theta_4) - 45\sin(\theta_2+\theta_3+\theta_4) + 93\cos(\theta_2+\theta_3) + 93\cos(\theta_2)) \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{40}$$

$$J2 = \begin{pmatrix} -\cos(\theta_1)\ \left(93\ \sin(\theta_2+\theta_3) + 93\ \sin(\theta_2) + 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 - \text{atan}\left(\frac{7}{9}\right)\right)\right) \\ -\sin(\theta_1)\ \left(93\ \sin(\theta_2+\theta_3) + 93\ \sin(\theta_2) + 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 - \text{atan}\left(\frac{7}{9}\right)\right)\right) \\ 93\ \cos(\theta_2+\theta_3) + 93\ \cos(\theta_2) + 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 + \text{atan}\left(\frac{9}{7}\right)\right) \\ \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{pmatrix} \tag{41}$$

$$J3 = \begin{pmatrix} -\cos(\theta_1)\ \left(93\ \sin(\theta_2+\theta_3) + 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 - \text{atan}\left(\frac{7}{9}\right)\right)\right) \\ -\sin(\theta_1)\ \left(93\ \sin(\theta_2+\theta_3) + 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 - \text{atan}\left(\frac{7}{9}\right)\right)\right) \\ 93\ \cos(\theta_2+\theta_3) + 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 + \text{atan}\left(\frac{9}{7}\right)\right) \\ \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{pmatrix} \tag{42}$$

$$J4 = \begin{pmatrix} -\cos(\theta_1)(45\cos(\theta_2+\theta_3+\theta_4) + 35\sin(\theta_2+\theta_3+\theta_4)) \\ -\sin(\theta_1)(45\cos(\theta_2+\theta_3+\theta_4) + 35\sin(\theta_2+\theta_3+\theta_4)) \\ 5\sqrt{130}\ \cos\left(\theta_2+\theta_3+\theta_4 + \text{atan}\left(\frac{9}{7}\right)\right) \\ \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{pmatrix} \tag{43}$$

The final Jacobian is again the four columns J1, J2, J3 and J4 appended as shown:

$$J = \begin{bmatrix} J1 & J2 & J3 & J4 \end{bmatrix} \tag{44}$$

Numerical results of the Jacobians along the path studies in problem 3

After calculating the two Jacobians, the aim is now to determine the numerical results for the two Jacbians using four points along the circle path from problem 3. The points in question were at $\varphi = 0$, $\varphi = \frac{\pi}{2}$, $\varphi = \pi$ and $\varphi = \frac{3\pi}{2}$ at which the robot joint configuration were as follows:

$x2\pi/36$	q_1	q_2	q_3	q_4
0	0.2102	1.4302	-1.6699	0.2397
9	0.0000	1.4904	-1.3903	-0.1002
18	-0.2102	1.4302	-1.6699	0.2397
27	-0.0000	1.3212	-1.9160	0.5949

Table 2: Robot configurations for the four points

Substituting $\theta_1, \theta_2, \theta_3$ and θ_4 with their numerical values from table 3 the following numerical results of the Jacobians are achieved (please note the Jacobian for the end effector stylus and the camera will be denoted J1 and J2 respectively):

$\varphi = 0$

$$J1 = \begin{bmatrix} -32.0022 & -68.4623 & 21.5933 & 0.0000 \\ 149.9976 & -14.6065 & 4.6070 & 0.0000 \\ 0.0000 & 153.3735 & 140.3411 & 50.0000 \\ 0.0000 & 0.2087 & 0.2087 & 0.2087 \\ 0.0000 & -0.9780 & -0.9780 & -0.9780 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{45}$$

$$J2 = \begin{bmatrix} -28.8724 & -112.4718 & -22.4163 & -44.0095 \\ 135.3278 & -23.9960 & -4.7825 & -9.3895 \\ 0.0000 & 138.3735 & 125.3411 & 35.0000 \\ 0.0000 & 0.2087 & 0.2087 & 0.2087 \\ 0.0000 & -0.9780 & -0.9780 & -0.9780 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{46}$$

$\varphi = \frac{\pi}{2}$:

$$J1 = \begin{bmatrix} 0.0000 & -101.9884 & -9.2888 & 0.0000 \\ 150.0033 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 150.0033 & 142.5345 & 50.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.0000 & -1.0000 & -1.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{47}$$

$$J2 = \begin{bmatrix} 0.0000 & -146.9899 & -54.2903 & -44.9965 \\ 135.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 135.0078 & 127.5390 & 35.0045 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.0000 & -1.0000 & -1.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{48}$$

$\varphi = \pi$:

$$J1 = \begin{bmatrix} 32.0022 & -68.4623 & 21.5933 & 0.0000 \\ 150.000 & 14.6065 & -4.6070 & 0.0000 \\ 0.0000 & 153.3735 & 140.3411 & 50.0000 \\ 0.0000 & -0.2087 & -0.2087 & -0.2087 \\ 0.0000 & -0.9780 & -0.9780 & -0.9780 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{49}$$

$$J2 = \begin{bmatrix} 28.8724 & -112.4718 & -22.4163 & -44.0095 \\ 135.3278 & 23.9960 & 4.7825 & 9.3895 \\ 0.0000 & 138.3735 & 125.3411 & 35.0000 \\ 0.0000 & -0.2087 & -0.2087 & -0.2087 \\ 0.0000 & -0.9780 & -0.9780 & -0.9780 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{50}$$

$\varphi = \frac{3\pi}{2}$:

$$J1 = \begin{bmatrix} 0.0000 & -38.0112 & 52.1069 & -0.0000 \\ 150.000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 150.0004 & 127.0282 & 50.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.0000 & -1.0000 & -1.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{51}$$

$$J2 = \begin{bmatrix} 0.0000 & -83.0097 & 7.1084 & -45.0035 \\ 135.000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 134.9959 & 112.0237 & 34.9955 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & -1.0000 & -1.0000 & -1.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \tag{52}$$

Problem 5

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

Having now found the Jacobians, it is possible to control the robot movement with precision. For example, it is possible to impose a certain end effector velocity to a specific robot configuration, finding the inverse velocity.
For example, it is possible to find the joint velocities $\dot{q}$ for the robot configuration corresponding to an angle of $\psi = \pi/2$ along the circle from problem 3. If here one wants to impose a linear velocity of $v_0^4 = [0, -3, 0]$ mm/s and angular velocity $\dot{x}_4 = [/, /, 0]$, some calculations are needed.
First of all, the interest is not in all the angular velocities of the end effector, but only in the last component. This means that the equation will only have one exact solution, as the unknowns are four, $\dot{q}^{4\times1}$. The vector of known end effector velocities, named ξ can then be computed as follows.

$$\xi = [v_x,\, v_y,\, v_z,\, \omega^*]^T \quad \text{and} \quad \dot{q} = [\dot{\theta}_1,\, \dot{\theta}_2,\, \dot{\theta}_3,\, \dot{\theta}_4]^T$$

For a set of angular velocities ω, the given end effector angular velocities $\dot{x}_4$ are given as in equation 53, using the fact that the frame of the end effector is rotated with a rotation matrix R with respect to the global axis, where in the considered case the R matrix is found below.

$$R = \begin{bmatrix} 0.978 & 0 & 0.209 \\ 0.209 & 0 & -0.978 \\ 0 & 1.0 & 0 \end{bmatrix} \qquad \dot{x} = \begin{pmatrix} R_{zx}\omega_y - R_{yx}\omega_z \\ R_{xx}\omega_z - R_{zx}\omega_x \\ R_{yx}\omega_x - R_{xx}\omega_y \end{pmatrix} \tag{53}$$

As the interest is only in the last element of $\dot{x}_4$ in this case, these considerations can be used to adapt the Jacobian matrix to the desired system.
Extracting the Jacobian corresponding to the $\psi = \pi/2$ configuration is indeed not sufficient, as this needs to be reduced to a four by four matrix. All the systems is meant for the four unknowns. To reduce the J matrix one could use a matrix as the M matrix below multiplied to the Jacobian, obtaining the reduced J in equation 54. Following this procedure the system becomes the one in equation 55.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_{yx} & -R_{xx} & 0 \end{bmatrix} \qquad J_{\text{red}} = \begin{bmatrix} 0 & -100 & -9.3 & 0 \\ 150 & 0 & 0 & 0 \\ 0 & 150 & 140 & 50 \\ 0 & 0.98 & 0.98 & 0.98 \end{bmatrix} \tag{54}$$

$$\xi^{4\times1} = M^{4\times6} \cdot J^{6\times6} \cdot \dot{q}^{4\times1} \tag{55}$$

Now that all the elements in equation 55 are present, the reduced Jacobian can be inverted to find $\dot{q}$. This can also be done in Matlab using the backslash operator. The resulting velocities are:

$$\dot{q} = [-0.02,\, 0,\, 0,\, 0]^T \quad [rad/s]$$

This result seems logical, as the considered point is for $\phi = \pi/2$, which means that the robot in this configuration is pointed along its x axis, and the only movement in that precise point is the rotation of the whole robot using the first motor. For this reason the only active rotation of $\dot{q}$ is $\dot{\theta}_1$.
The correctness of the result can also be seen by simply following the inverse process, multiplying the reduced J matrix by the obtained velocities and comparing the result with the initial ξ vector.

Part 2: Trajectory planning

The trajectory found in problem 3 is of course only the ideal trajectory for the robot to compute the circle. In reality, the robot will move with a certain speed and will have to conduct the path in a given time.
The real trajectory is then found by interpolation, using a number of knot points along the path and imposing speed and acceleration of the robot in those points.

Problem 6

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%) Using five knot points,

respectively at $\psi = \left[0, \pi/2, \pi, 3\pi/2, 2\pi\right]$, and imposing speeds of the end effector as seen below, it is possible to find interpolation polynomials for the four segments that go from one knot point to the other. Here accelerations are null.

$$
\begin{aligned}
v(t_A = 0) &= \left[0, 0, 0\right]^T \text{ mm/s} \\
v(t_A = 2) = v(t_B = 0) &= \left[0, -27, 0\right]^T \text{ mm/s} \\
v(t_B = 2) = v(t_C = 0) &= \left[0, 0, -27\right]^T \text{ mm/s} \\
v(t_C = 2) = v(t_D = 0) &= \left[0, 27, 0\right]^T \text{ mm/s} \\
v(t_D = 2) &= \left[0, 0, 0\right]^T \text{ mm/s}
\end{aligned}
$$

These velocity elements must be converted from end effector velocities to joint velocities $\dot{q}$ with the process described in problem 5. The position element of the free coordinates, q, are the one found in problem 3 for the given angles.
The interpolation polynomials for the four segments are found by quintic interpolation as seen in 56. This requires 6 boundary conditions: The configurations for the position, the angular velocity of each joint and acceleration of each joint for the start and the end-point of each trajectory segment.

$$
q(t_A) = \begin{bmatrix} A_{15}t^5 + A_{14}t^4 + A_{13}t^3 + A_{12}t^2 + A_{11}t + A_{10} \\ A_{25}t^5 + A_{24}t^4 + A_{23}t^3 + A_{22}t^2 + A_{21}t + A_{20} \\ A_{35}t^5 + A_{34}t^4 + A_{33}t^3 + A_{32}t^2 + A_{31}t + A_{30} \\ A_{45}t^5 + A_{44}t^4 + A_{43}t^3 + A_{42}t^2 + A_{41}t + A_{40} \end{bmatrix} \tag{56}
$$

Only the analysis for segment A will be shown here as the process is identical for each segment.

$$
q^{(0)} = \begin{bmatrix} 0.2102 \\ -0.3523 \\ -1.1777 \\ 2.03 \end{bmatrix} \tag{57}
$$

$$
q^{(9)} = \begin{bmatrix} 0 \\ 0.689 \\ -1.0961 \\ 1.7852 \end{bmatrix} \tag{58}
$$

$$
\dot{q}^{(9)} = \begin{bmatrix} -0.18 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{59}
$$

The start velocity is $v^0 = [0, 0, 0]^T$

$$
\dot{q}^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{60}
$$

The acceleration at each point is set to 0:

$$\ddot{q}^{(0)} = \ddot{q}^{(9)} \begin{bmatrix} -0.02 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{61}$$

Solving the 6 equations for the coefficients and constants.

$$q(t_A = 0) = q^{(0)} \tag{62}$$
$$q(t_A = 2) = q^{(9)} \tag{63}$$
$$\dot{q}(t_A = 0) = \dot{q}^{(0)} \tag{64}$$
$$\dot{q}(t_A = 2) = \dot{q}^{(9)} \tag{65}$$
$$\ddot{q}(t_A = 0) = \ddot{q}^{(0)} \tag{66}$$
$$\ddot{q}(t_A = 2) = \ddot{q}^{(9)} \tag{67}$$

The resulting polynomials for the four segments:

$$q(t_A) = \begin{bmatrix} -0.0056592\,t^5 + 0.039546\,t^4 - 0.082728\,t^3 + 0.21018 \\ 0.011298\,t^5 - 0.05649\,t^4 + 0.07532\,t^3 + 1.4302 \\ 0.052433\,t^5 - 0.26217\,t^4 + 0.34955\,t^3 - 1.6699 \\ -0.063731\,t^5 + 0.31866\,t^4 - 0.42487\,t^3 + 0.23973 \end{bmatrix} \tag{68}$$

$$q(t_B) = \begin{bmatrix} -0.0056592\,t^5 + 0.017046\,t^4 + 0.0072718\,t^3 - 0.18\,t + 1.3063\text{e-}17 \\ -0.0028359\,t^5 + 0.017\,t^4 - 0.030189\,t^3 + 3.8248\text{e-}18\,t + 1.4904 \\ -0.025608\,t^5 + 0.13698\,t^4 - 0.20649\,t^3 - 4.0252\text{e-}18\,t - 1.3903 \\ 0.063731\,t^5 - 0.31866\,t^4 + 0.42487\,t^3 - 0.10017 \end{bmatrix} \tag{69}$$

$$q(t_C) = \begin{bmatrix} 0.0056592\,t^5 - 0.039546\,t^4 + 0.082728\,t^3 - 8.0803\text{e-}19\,t - 0.21018 \\ -0.011977\,t^5 + 0.057064\,t^4 - 0.068563\,t^3 - 0.045131\,t + 1.4302 \\ -0.019325\,t^5 + 0.087682\,t^4 - 0.093065\,t^3 - 0.14307\,t - 1.6699 \\ 0.066589\,t^5 - 0.33294\,t^4 + 0.44392\,t^3 + 0.23973 \end{bmatrix} \tag{70}$$

$$q(t_D) = \begin{bmatrix} 0.0056592\,t^5 - 0.017046\,t^4 - 0.0072718\,t^3 + 0.18\,t - 3.9189\text{e-}17 \\ 0.020439\,t^5 - 0.1022\,t^4 + 0.13626\,t^3 + 1.0629\text{e-}17\,t + 1.3212 \\ 0.04615\,t^5 - 0.23075\,t^4 + 0.30766\,t^3 - 1.2552\text{e-}17\,t - 1.916 \\ -0.066589\,t^5 + 0.33294\,t^4 - 0.44392\,t^3 + 0.59487 \end{bmatrix} \tag{71}$$

Problem 7

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

When interpolation polynomials for the trajectory has been obtained, the desired time steps t were inserted to compute the trajectory. Using 5 knot points and a total of 100 timesteps, so 25 points for each segment, gave the approximate trajectory displayed in figure 3. In the plot, the segments corresponding to each vector of interpolation polynomials are highlighted in different colors, and it is evident by the comparison with the theoretical trajectory that the robot is not perfectly able to follow the desired path.

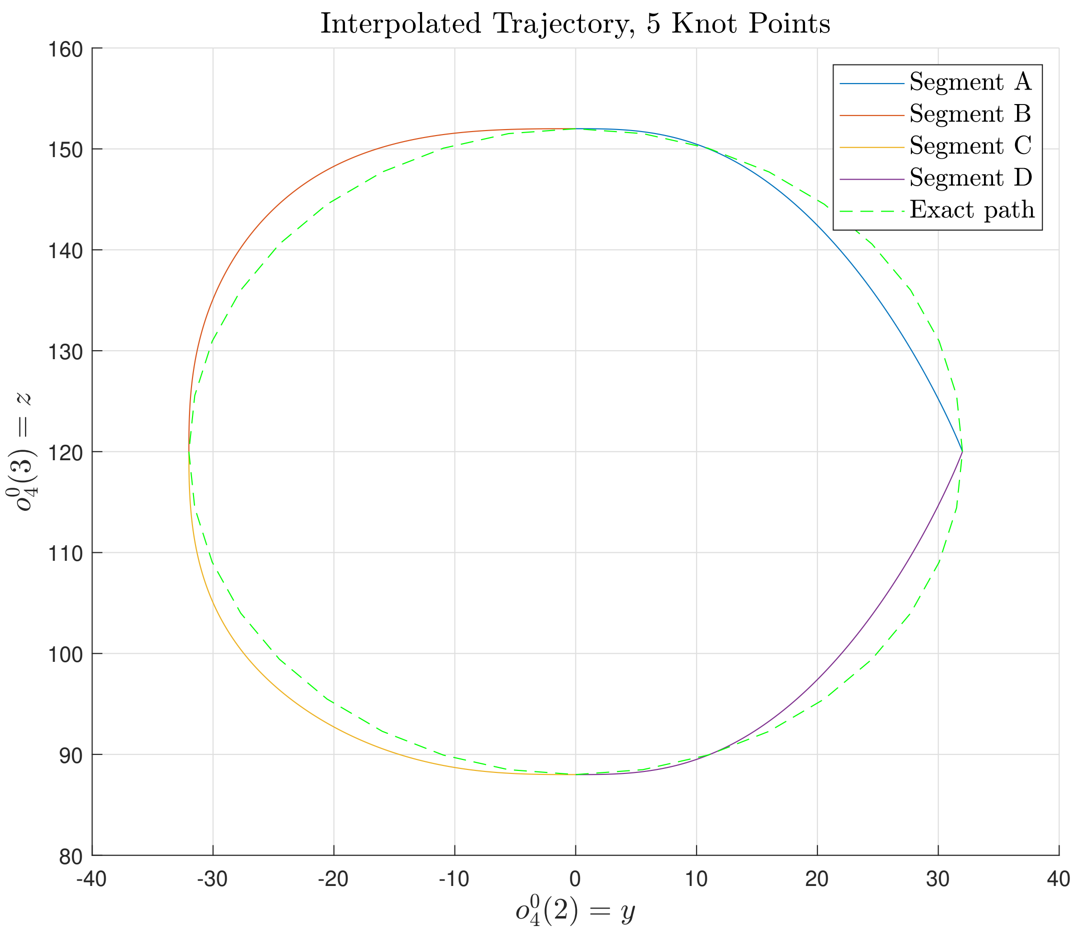

Figure 3: Interpolated trajectory and exact trajectory overlayed

It is possible to improve the precision of the robot trajectory by using more knot points. This is of course more computationally demanding, but precision is often determinant in robot applications, so a total of 21 knot points were created, using using inverse kinematics, obtaining the plot in figure 4.
This new interpolated path matches with great precision the desired path of the circle, given by the inverse kinematics of problem 3.

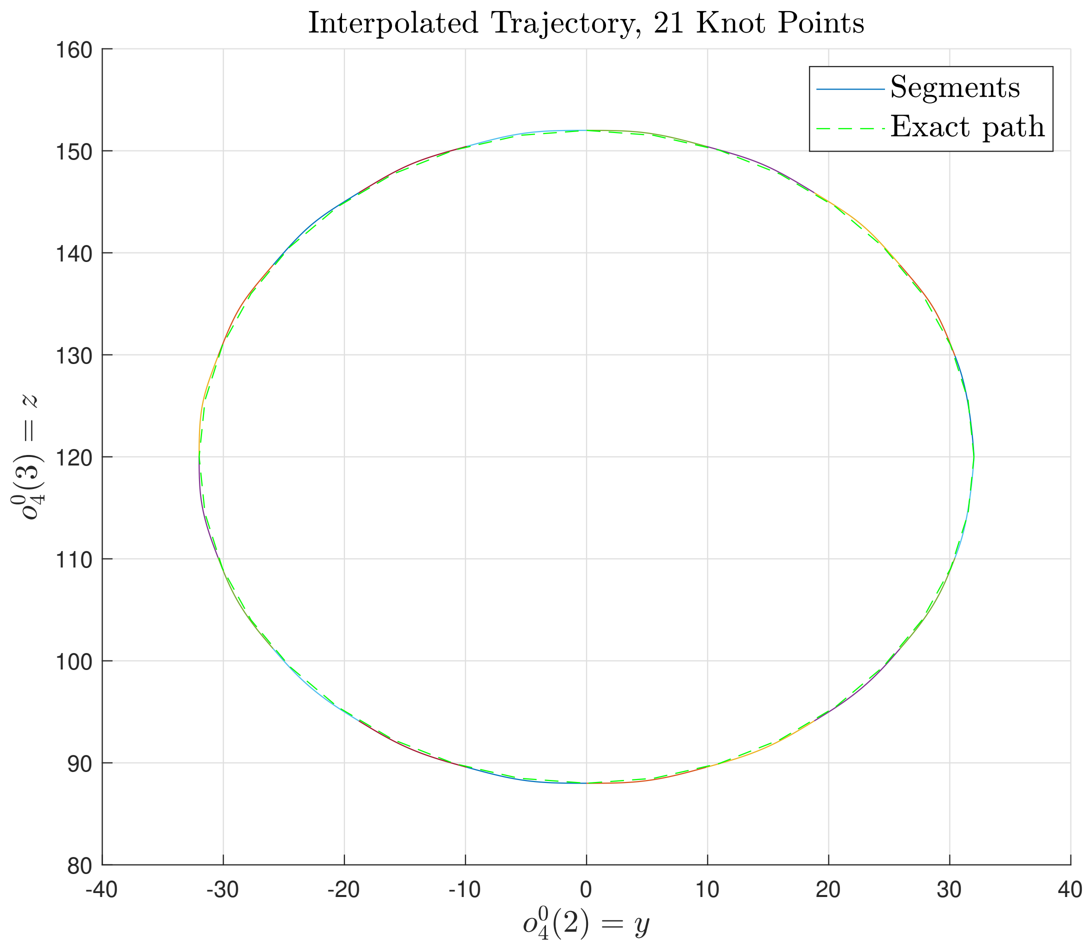

Figure 4: Interpolated trajectory and exact trajectory overlayed

Part 3: Singularities and statics

Problem 8

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%) The objective of this problem was

to investigate the presence of singularities along the paths from problem 3, the ideal path of the robot, and 6, the one found by interpolation. This was done by plotting the condition number of the Jacobian matrix for each of the 37 robot configurations, in figures 5 and 6. In these plots, the unit of the system were kept as millimeters.
The condition number is defined as the ratio of the largest singular value to the smallest singular value. It is a measure of how sensitive the solution of a system of linear equations is to changes in the input, and therefore gives a good indication of how close the robot is to a singular configuration.
In Matlab, it is possible to find the condition number of a given matrix using the *"cond"* function.

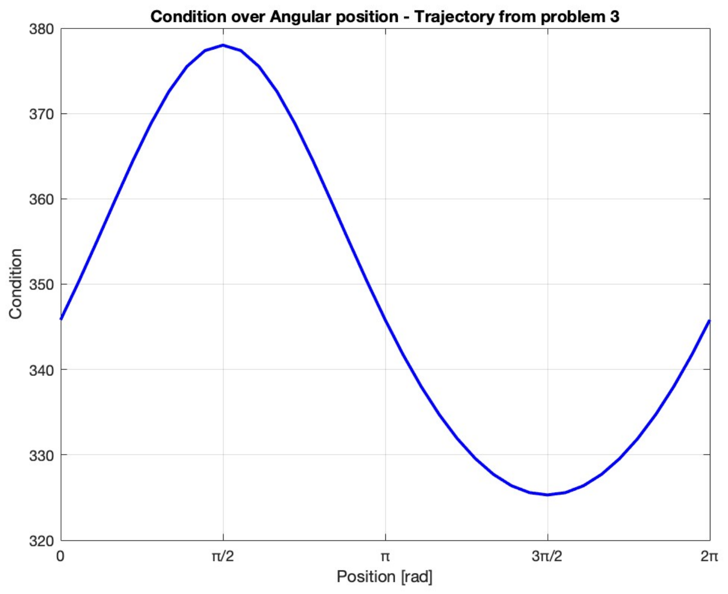

Figure 5: Condition number along the circle path from problem 3

Taking for example the plot in figure 5, it is evident that the condition value of the Jacobian has no peaks along the desired path, and therefore the robot will not encounter any singularity. In the second plot of figure 6 the results are similar, but the values do not change as smoothly, as the trajectory is in this case an approximation given by interpolation.

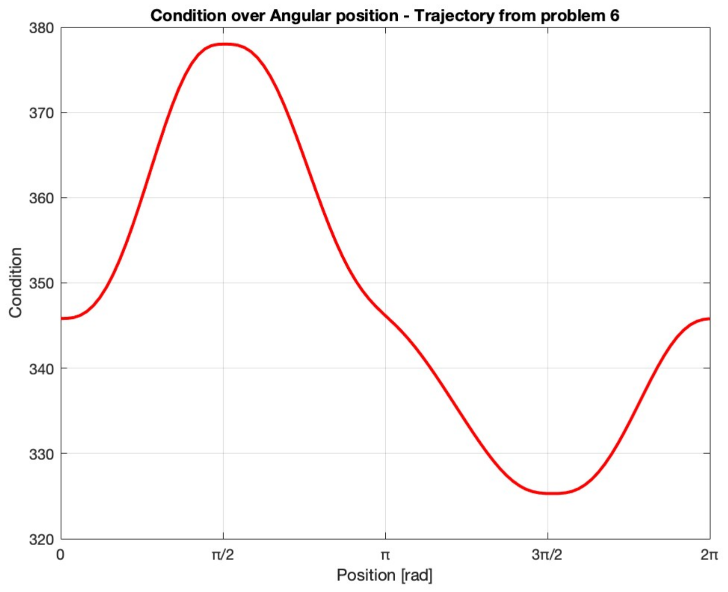

Figure 6: Condition number along the circle path from problem 6

Problem 9

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%) As part of the static analysis of the robot, it is possible to find the static torques associated with a certain configuration of the robot and for a certain applied force. This is done with equation 72, where the robot configuration is of course given by the free coordinates in q, and J is the Jacobian of the robot, for that specific q.

$$\tau = J^T(q)F \tag{72}$$

Here F is the vector collecting the forces applied to the end effector, where the first three elements are forces along the global axes, and the last three elements are torques around the same axes. In this case the own mass of the robot arm is neglected, and the applied load is simply a 1N weight at the end effector, pointing down along the z_0 axes, so the F vector is:

$$F = [0 \;\; 0 \;\; -1 \;\; 0 \;\; 0 \;\; 0]^T$$

Equation 72 can be solved for each interpolated configuration of the robot q, found in problem 6 to draw a circle with the robot. This allows to see the evolution of the static torque needed by each motor to hold the 1N load at the end effector, for each interpolated position. Performing these calculation allows to create the plot in figure 7 below.

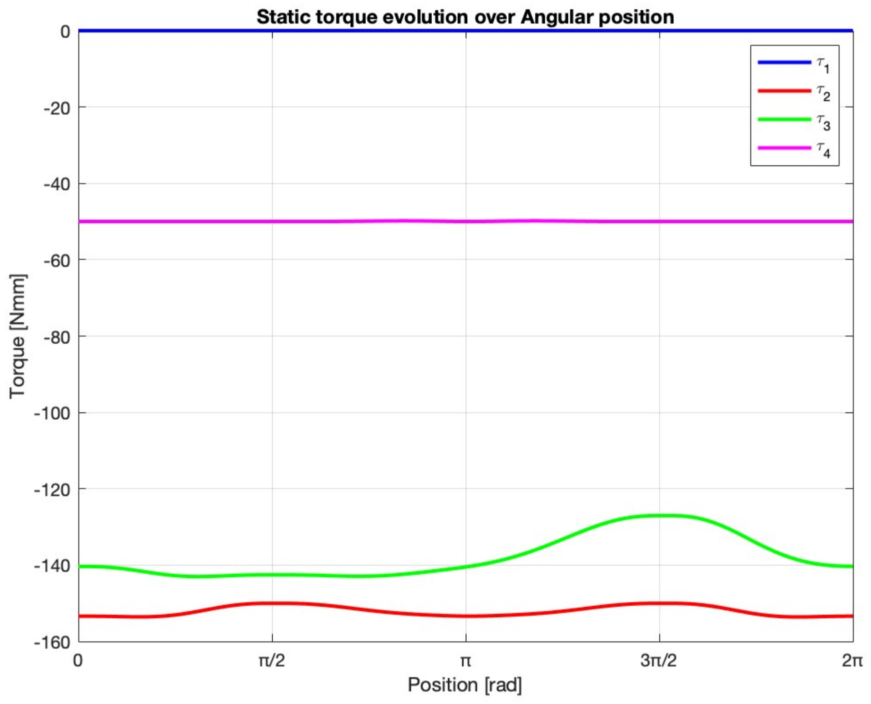

Figure 7: Static torque of each motor to compute trajectory from problem 6

Here the torque of the first actuator, τ_1 is null throughout the whole trajectory. This is logical, as the load is held by the other robot links, while the first link is positioned without any load applied, in statics. The second torque τ_2 is roughly constant, while the other two actuators need a higher torque, up to -160 Nmm, to hold the 1N load in the desired position.
https://www.overleaf.com/project/65254bda6839135978bf46e8

Part 4: Dynamics

Problem 10

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%) Knowing the dynamics of a robot

is fundamental for its control. For example, forces and torques applied by the motors are strictly dependent on the robot dynamics, therefore having a good dynamics model results in a more accurate control.
The dynamics system for the robot arm in its standard form is found as follows:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{73}$$

The first element in equation 73 is the inertia matrix of the robotic manipulator, often indicated with the letter D. This is a function of the free coordinates of the robot q.
Finding D involves finding the Jacobian matrices corresponding to the centers of mass of each robot link, to then split the Jacobians in the part associated with linear velocities, J_v and angular velocities J_ω. These are used as follows to find the D matrix:

$$D(q) = \sum_{i=1}^{n} m_i J_{v,c_i}^T J_{v,c_i} + J_{\omega_i}^T R_i I_i R_i^T J_{\omega_i} \tag{74}$$

First of all one can find the positions of the centers of mass based on the dimensions given in the assignment, visible in figure 8.

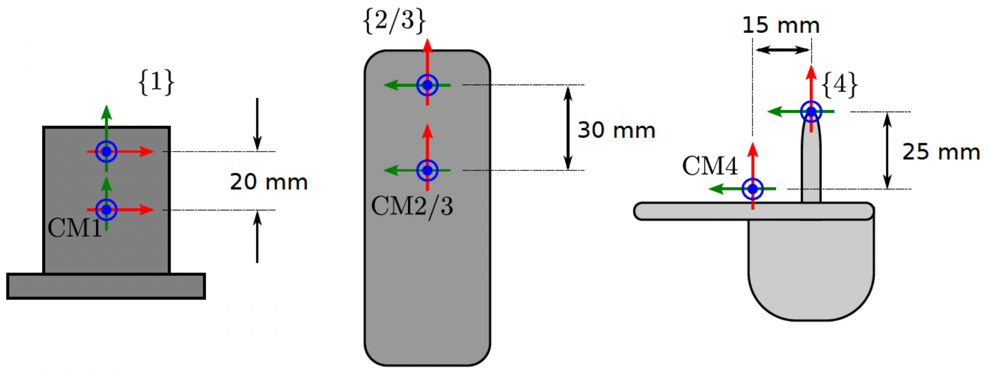

Figure 8: Centers of mass and principal axes of inertia of links 1 to 4, from assignment

As the centers of mass are simply a translation of the original link frames, one can simply apply a translation matrix to frames one to four, finding the centers of mass. Examples of center of mass coordinates are given below for the first three.

$$\mathbf{o}_{\text{cm1}} = \begin{bmatrix} 0 \\ 0 \\ 0.03 \end{bmatrix}$$

$$\mathbf{o}_{\text{cm2}} = \begin{bmatrix} 0.063c12 \\ 0.063c2s1 \\ 0.063s2 + 0.05 \end{bmatrix}$$

$$\mathbf{o}_{\text{cm3}} = \begin{bmatrix} 0.093c12 - 0.063c1s23 + 0.063c12c3 \\ 0.093c2s1 - 0.063s12s3 + 0.063c23s1 \\ 0.093s2 + 0.063c2s3 + 0.063c3s2 + 0.05 \end{bmatrix}$$

With these, the Jacobian matrices for the four centers of mass are soon found following the procedure described in previous chapters (see section). The matrices are not reported in this document as too cumbersome.
The inertia tensors for each robot link were given in the assignment as function of I_0. Finding this I_0 is straight forward, it is sufficient to consider the first link of the robot, whose inertia matrix is given below, and compute the moment of inertia of the link with respect to the vertical axes, to compare it with the given one in the matrix.

$$D_1 = \begin{bmatrix} I_0 & 0 & 0 \\ 0 & 0.4 \cdot I_0 & 0 \\ 0 & 0 & 0.9 \cdot I_0 \end{bmatrix}$$

The vertical axes of the first link is the y axes, therefore the needed part of the matrix is $D_1(2,2)$. The moment of inertia of the link can be found as a combination of the moment of inertia of the link itself and the disk to which its fixed. In the equations below R indicated the ratio of the disk, and a and b indicate the side lengths of the first robot link. The mass was given as an overall mass of $60g$. This was divided in between the disk, $10g$ and the link, $50g$.

$$\begin{aligned} I_{link} &= \frac{1}{12} m_{link}(a^2 + b^2) \\ I_{disk} &= \frac{1}{2} m_{disk} R^2 \\ I_{total} &= I_{link} + I_{disk} = 0.4 \cdot I_0 \end{aligned}$$

Solving this simple equation, one can find the value of I_0, which is $I_0 = 6.515 \cdot 10^{-5}$ kgm^2.
With I_0, the inertia matrices with respect to the center of mass of the links are soon found by substitution of I_0 in the matrices given in the assignment. These can be rotated as follows to refer to the global axes instead.

$$J_i = R_i I_i R_i^T$$

The rotated matrices are used in equation 73 to find the D matrix.Having now all the components of equation 74, $D(q)$ can be calculated using Matlab. The result is not reported here as very cumbersome.
After the inertia matrix of the robot D, the next step involves calculating the C matrix, composed of Christoffel symbols c_{ijk}. Following equation 75, the matrix's rows and columns are indicated with k and j respectively, and the Christoffel symbols are found by derivation of the elements of the inertia matrix with respect to the free coordinates.
A summation along the first index of Christoffel symbols, multiplied by the velocity vector $\dot{q}$ gives the result. In the considered robot, the free coordinates are four, so C will be a 4x4 matrix.

$$[C(q,\dot{q})]_{kj} = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i = \sum_{i=1}^{n} \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k} \right) \dot{q}_i \tag{75}$$

The last element needed to compute equation 73 and find τ is the g vector. First the potential energy associated with the robot is calculated with equation 76. Here g is simply the gravity, which in the reference system used for the robot is pointing down along the z axes.

$$g = \begin{bmatrix} 0 & 0 & -9.81^T \end{bmatrix}$$

$$P = -\sum_{i=1}^{n} m_i g^T r_{ci}(q) \tag{76}$$

Found P, this can be used in equation 77 to find **g**, by derivation of the potential energy with respect to the free coordinates q.

$$\mathbf{g} = \begin{bmatrix} g_1(q) \\ \dots \\ g_n(q) \end{bmatrix} \text{ where } g_k = \frac{\partial \mathcal{P}}{\partial q_k} \tag{77}$$

It is now possible to compute equation 73, and find the required torque for each link motor to compute a certain trajectory. The output of the equation will be a vector where each element is the torque needed by one of the robot actuators.
As an example, the needed torque to compute the circular trajectory found in problem 6 is calculated. This can be done by simply finding an array of positions, velocities and accelerations interpolating the given points in the trajectory.
Substituting these values in the equation found for τ the, torque is found and plotted in figure 9.

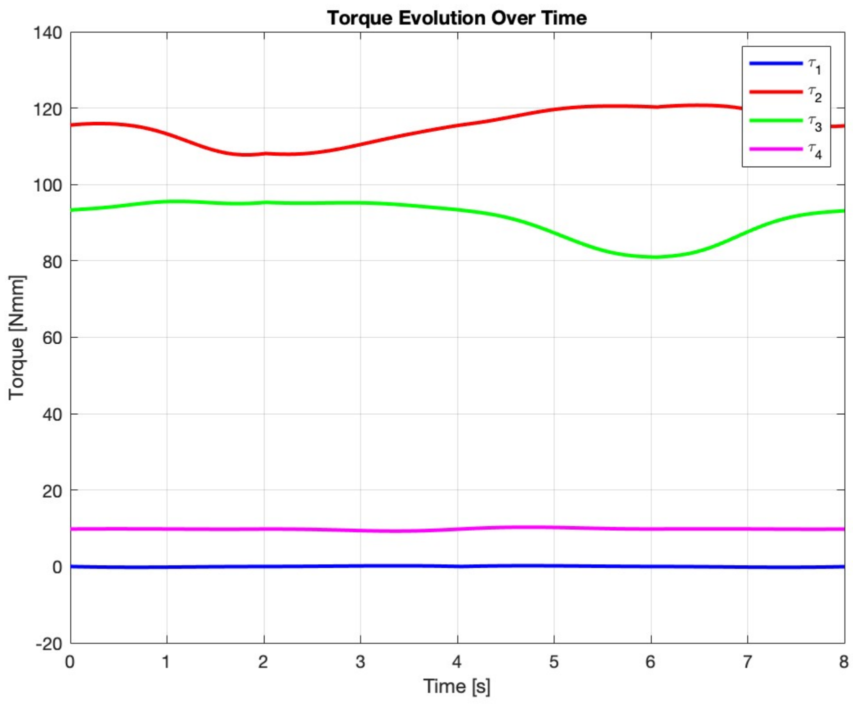

Figure 9: Dynamic torque of each motor to compute trajectory from problem 6

It is soon evident that the plot is similar to the one obtained for the static system. If in the static system τ_1 was null, here the torque associated with the first actuator has to fight the inertia, and is therefore not perfectly zero. The torque from the last actuator, τ_4, is relatively low compared to the two middle links. This was also seen in the static system, and the reason for it is that the end effector is kept horizontal and does not move much in drawing the circle. Torques τ_2 and τ_3 are higher than the previous two, with values in between 80 [Nmm] and 120 [Nmm].

Part 5: Computer vision and control

(Christian 20%, Francesco 20%, Christoffer 20%, Jacob 20%, Anna 20%)

The aim of part 5 of the assignment is to detect some red coloured ”smarties” among a cluster of colourful ”smarties” on a surface in front of it, and to probe all of them with its stylus.
In order to do so it is necessary to connect the robot arm to a laptop, in this way it will be possible to detect the camera mounted on the final part of the robotic arm.
At first the code was written in python but later an similar program was written in matlab, in order to make the integration with the movement code of the robot better. In order to control the robot and

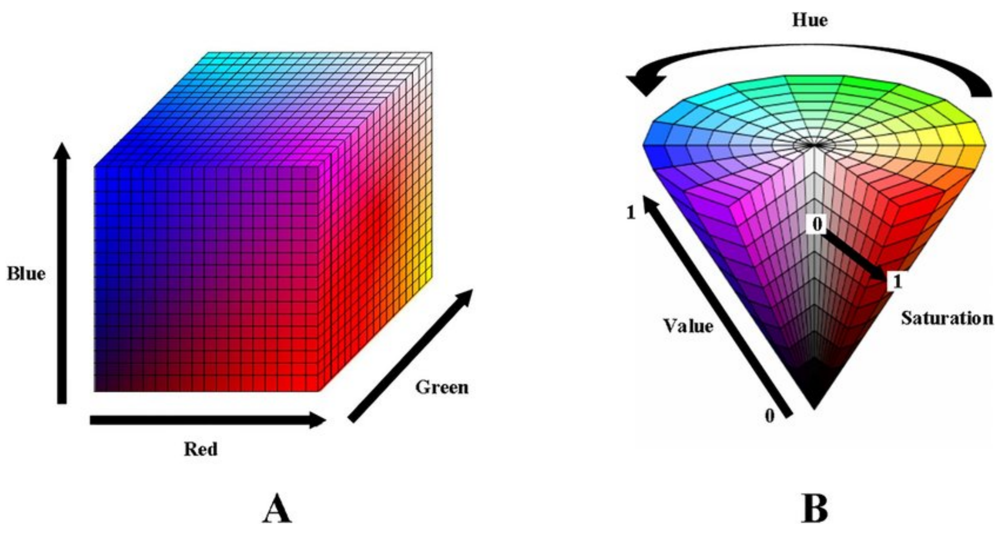

Figure 10: RGB and HSV color spaces representation (1)

implement the theoretical kinematics, inspiration was found in the git repository linked in the project description:
`https://github.com/MonsisGit/MyRobot`
Camera calibration was executed employing the integrated camera calibrator within the Image Processing App. The objective was to obtain the intrinsic parameters of the Logitech C270 camera. This process involved capturing 21 images of a checkerboard grid with known dimensions, taken from various angles. It is imperative to acknowledge that, as an aspect of laboratory testing practical work, the parameters utilized in the latest iteration of the code were derived from online specification sheets, as detailed in the reference list. A comparative analysis is presented in Table 3 below.

C270	Matlab	Online
Height/Width [pixels]	[960,1280]	[960,1280]
Pixel length [mm]	TBC	2.8*10$^-$3
Focal length [mm] [f_x,f_y,λ]	[1415,1429,TBC]	[1430,1430,4]
Principal point [pixels]	[683,522]	[620,480]

Table 3: Comparison of intrinsic values of C270

The code that actually performs the movements on the robot arm has been divided in three functions. Initially, the "Image" function captures an image. Subsequently, the "CreateMask" function generates a mask by employing selected RGB thresholds derived from the ColorThresholder App. These thresholds may exhibit slight variations contingent upon alterations in lighting conditions and exposure settings on the camera. Once the thresholds are established in the RGB channels, a mask is generated based on these values, followed by the removal of the background.
Thirdly, the "FindCoords" function processes the image obtained from the "CreateMask" function by converting it into black and white through binarization. Subsequently, the function searches for circularity within the binary image and determines a center point for any identified centroids. The resulting output, denoted as "CurrImageCoords," represents the pixel coordinates of the detected objects on the camera plane.

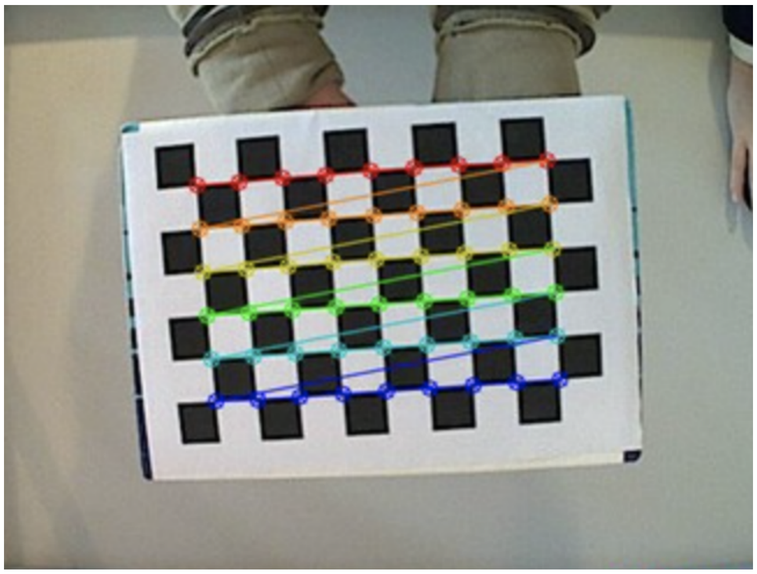

Figure 11: Camera calibration - detection of corners with opencv (2)

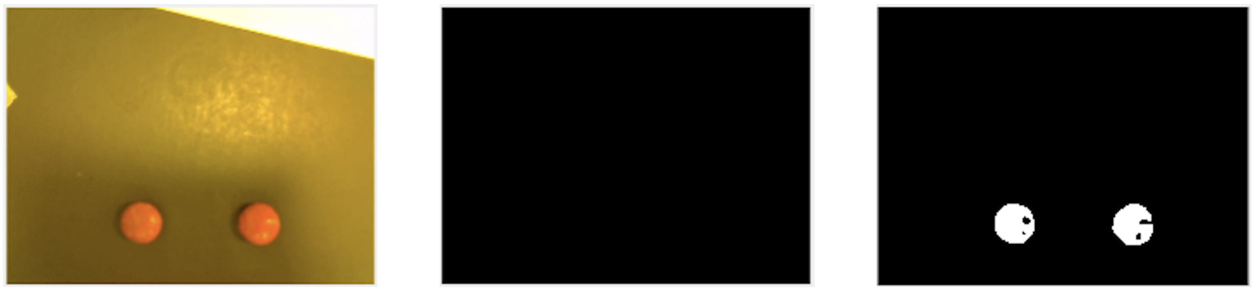

Figure 14: Stages of image processing

A computational transformation is undertaken to convert pixel coordinates to world frame coordinates using the following equations, leveraging intrinsic parameters:

$$r = -\frac{\lambda}{s_x}\frac{x}{z} + o_r \qquad c = -\frac{\lambda}{s_y}\frac{y}{z} + o_c \tag{78}$$

In the implemented camera test code, these computed coordinates are denoted as worldX and worldY. Upon obtaining these coordinates, an adjustment is introduced in both the X and Y directions to relocate the center of the pixel frame from the top-left corner to the middle. This involves determining the number of pixels, multiplying it by the length of one pixel, and subsequently dividing it by 2. These adjustments are represented as offsetx and offsety in the code.
These coordinates will then be used from the robot to position pointing to the detected objects, this will be done thanks to the MoveCartesian function incorporating the computed worldX and worldY coordinates.
Below specifications on the initially implemented Python script can be found:
The code, after a successful connection to the camera, starts by converting the image its capturing from RGB color space (red-green-blue) to HSV color space.
The code starts by connecting to the external camera indexed with 0 and acquire the image on which the script will perform the computer vision task. The image is initially converted from RGB (Red-Green-Blue) color space to the HSV (Hue-Saturation-Value) one.
Operating in this color space is particularly indicated for this kind of ”color detection” tasks, it allows the programmer to easily find the values of H - Hue, S - Saturation and V - Value needed to select a specific

range of colors. After some tests on the light setting of the area where the experiment has been carried out, the values that have been found adapt to fulfill the requirements for the tasks are :

$$lowerlimit = [0, 145, 145]; upperlimit = [10, 255, 255]; \tag{79}$$

The colour red of the smarties to be detected is found to be in this limits, these values allow the correct detection of the color, which may not be a simple task since there is a lot of noise from external light that makes it difficult to not detect similar colors like pink. Some attention on the setup and position of the robotic arm can save the programmers a lot of time and effort.
After having found this values a mask is created so that only the red zones will be detected, making it easy to work on the detected shapes.
Once the mask is created it is possible to iterate through the contours and operate the BLOB analysis (Binary large objects).
In this case it has been necessary to define a range between which the area of the detected objects should be in order to minimize the disturbance which can give as output some small clusters of pixel that may be in the same HSV values as the smarties caused by lighting environment.
It has not been necessary to operate on particular BLOB features since the only parameters of interest for this problem were the coordinates of the center of the smarties c_{x_i} and c_{y_i}.
These coordinates are then saved into the *cxArray* and *cyArray*, making it possible to indicate where the final position of the arm should be.
As previously stated, first the code was written in Python using the OpenCV package but later a similar program was written in matlab, in order to make the integration with the movement of the robot better.

1 Reference list

Sources for C270 camera Specifications:

- https://stargazerslounge.com/topic/244964-cheap-astrophotography-galileoscope-and-logitech-c270/
- https://horus.readthedocs.io/en/release-0.2/source/scanner-components/camera.html

Images:
(1) https://www.researchgate.net/figure/RGB-color-space-to-HSV-color-space-conversion_fig4_359058781
(2)https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html