University of Ottawa

Automated classification with robotic manipulator

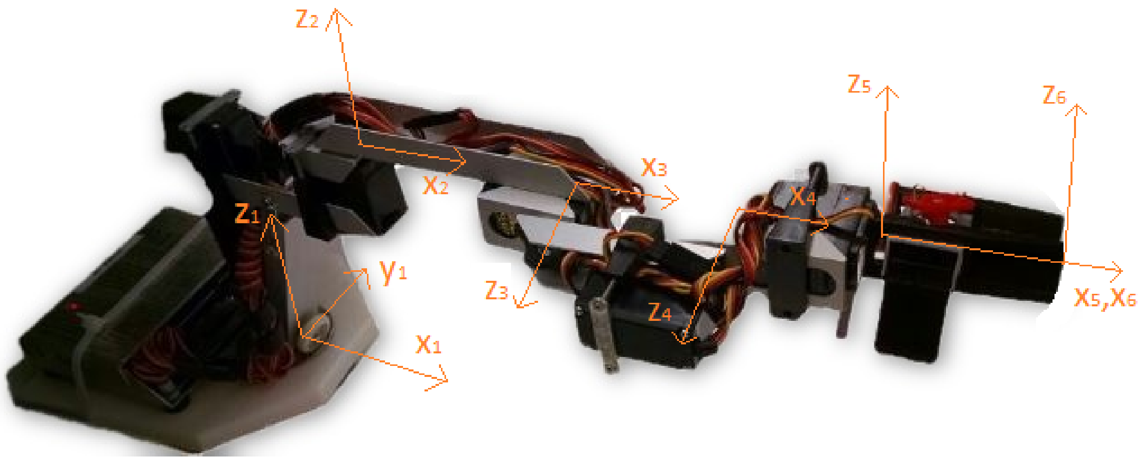
Michael Asafo 8277354
Maike Schröder 898079

Content

1. Robot modeling

a. Reference frames and model

The selection of reference frames for each joint was done according to the Denavit–Hartenberg convention. The first set of axis was selected with z_1 pointing upwards on joint 1 and with the origin attached to the table. In this way, the reference frames for the objects that will be defined later can have a z coordinate of 0, meaning they are considered to be in the table. The next sets of axes, from joints 2 to 5, were selected according to the convention. The origin of reference frame 6 was considered to be in the tip of the gripper, so that when z_6=0 in respect to z_1, the gripper is touching the table and can grab an object. The reference frames selection is shown in the next figure.



The Denavit–Hartenberg parameters for the robot's model are:

Joint Number	l_i (mm)	d_i	α_i	θ_i
1	95	155	0	θ
2	100	0	90	θ
3	60	0	0	θ
4	0	0	90	θ +90
5	0	110	0	θ

b. Forward kinematic model

In order to express the position and orientation of the gripper with respect to the base of the ROBIX, a forward kinematic model was found using the following development of A matrices, with an evolving reference frame, for each reference frame:

1. Rotation around the z_i axis of an angle θ_i;
2. Translation along the z_i axis of a distance d_i;
3. Translation along the x_{i+1} axis of a distance l_i;
4. Rotation around the x_{i+1} axis of an angle α_i.

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & l_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & l_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrices obtained are:

$$A_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 95\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & 95\sin\theta_1 \\ 0 & 0 & 1 & 155 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 100\cos\theta_2 \\ \sin\theta_2 & 0 & \cos\theta_2 & 100\sin\theta_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 60\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 60\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4 = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 110 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The product of the A matrices and simplification of $A_{gripper/base}$ were done using Matlab:

$$A_{gripper/base} = A_1 * A_2 * A_3 * A_4 * A_5$$

$$A_{gripper/base} = \begin{bmatrix} \beta & \chi & \cos\theta_A * \sin\theta_B & \lambda \\ \delta & \varepsilon & \sin\theta_A * \sin\theta_B & \xi \\ \sin\theta_B * \cos\theta_5 & -\sin\theta_B * \sin\theta_5 & -\cos\theta_B & 60\sin\theta_3 - 110\cos\theta_B + 155 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Where:}$$
$$\beta = \sin\theta_A * \sin\theta_5 + \cos\theta_A * \cos\theta_B * \cos\theta_5$$
$$\chi = \sin\theta_A * \cos\theta_5 - \cos\theta_A * \cos\theta_B * \sin\theta_5$$
$$\delta = \cos\theta_B * \sin\theta_A * \cos\theta_5 - \cos\theta_A * \sin\theta_5$$
$$\varepsilon = -\cos\theta_A * \cos\theta_5 - \cos\theta_B * \sin\theta_A * \sin\theta_5$$
$$\lambda = 100\cos\theta_A + 95\cos\theta_1 + 100\cos\theta_A * \sin\theta_B + 60\cos\theta_A * \cos\theta_3$$
$$\xi = 100\sin\theta_A + 95\sin\theta_1 + 100\sin\theta_A * \sin\theta_B + 60\sin\theta_A * \cos\theta_3$$
$$\theta_A = \theta_1 + \theta_2$$
$$\theta_B = \theta_3 + \theta_4$$

c. Implementation of forward kinematics

The code for the development of the forward kinematic model was implemented in a Matlab script, with a procedure similar to the one found in Inverse_22_11.m. The function sym('…') was used to make each of the Θ_i a symbolic variable, in order to obtain the generic expressions shown before.

The model was validated by introducing arbitrary angle values to each joint into the Matlab script to obtain different $A_{gripper/base}$ matrices. Then, a conversion of the selected angles was done to transform them from degrees to units with which the ROBIX controller can operate each joint. This was done following the next equation:

$$\text{joint}_i = \frac{\theta_i - b_i}{m_i}$$
$$\text{Where:}$$
$$b_1 = -5; b_2 = 5; b_3 = 10; b_4 = 10; b_5 = -20$$
$$m_1 = \frac{-165}{2800}; m_2 = \frac{-9}{140}; m_3 = \frac{-9}{140}; m_4 = \frac{17}{280}; m_5 = \frac{-9}{165}$$

The m_i values allow the conversion from degrees to ROBIX parameters, while the b_i terms represent an offset in degrees, because when the robot is in the rest position, joints are not completely at 0°.

2. Inverse kinematics

a. Closed-form solution for each angle, development and results

After calculating the A matrix in the forward kinematics, equations have to be derived to calculate the (controllable) angles needed to result in an aimed A matrix.

Therefore the A matrix is simplified and the following equations are used:

Using $\theta_A = \theta_1 + \theta_2$ and $\theta_B = \theta_3 + \theta_4$

$$A(1,1) = \sin sin\left(\theta_A\right) \bullet \sin sin\left(\theta_5\right) + \cos cos\left(\theta_A\right) \bullet \cos cos\left(\theta_B\right) \bullet \cos cos\left(\theta_5\right) \quad (1)$$

$$A(2,1) = \sin sin\left(\theta_A\right) \bullet \cos cos\left(\theta_B\right) \bullet \cos cos\left(\theta_5\right) - \cos cos\left(\theta_A\right) \bullet \sin sin\left(\theta_5\right) \quad (2)$$

$$A(3,1) = \sin sin\left(\theta_B\right) \bullet \cos cos\left(\theta_5\right) \quad (3)$$

$$A(1,2) = \sin sin\left(\theta_A\right) \bullet \cos cos\left(\theta_5\right) - \cos cos\left(\theta_A\right) \bullet \cos cos\left(\theta_B\right) \bullet \sin sin\left(\theta_5\right) \quad (4)$$

$$A(2,2) = - \cos cos\left(\theta_A\right) \bullet \cos cos\left(\theta_5\right) - \sin sin\left(\theta_A\right) \bullet \cos cos\left(\theta_B\right) \bullet \sin sin\left(\theta_5\right) \quad (5)$$

$$A(3,2) = - \sin sin\left(\theta_B\right) \bullet \sin sin\left(\theta_5\right) \quad (6)$$

$$A(1,3) = \cos cos\left(\theta_A\right) \bullet \sin sin\left(\theta_B\right) \quad (7)$$

$$A(2,3) = \sin sin\left(\theta_A\right) \bullet \sin sin\left(\theta_B\right) \quad (8)$$

$$A(3,3) = - \cos cos\left(\theta_B\right) \quad (9)$$

$$A(1,4) = l_2 \cos cos\left(\theta_A\right) + \left(d_3 + d_4\right) \sin sin\left(\theta_A\right) + l_1 \cos cos\left(\theta_1\right) + d_5 \cos cos\left(\theta_A\right) sin(\theta_B) + l_3 cos(\theta_A) cos(\theta_3 \quad (10)$$

$$A(2,4) = l_2 \sin sin\left(\theta_A\right) - \left(d_3 + d_4\right) \cos cos\left(\theta_A\right) + l_1 \sin sin\left(\theta_1\right) + d_5 \sin sin\left(\theta_A\right) sin(\theta_B) + l_3 sin(\theta_A) cos(\theta_3) \quad (11)$$

$$A(3,4) = (d_1 + d_2 - d_5 \cos cos\left(\theta_B\right) + l_3 sin(\theta_3) \quad (12)$$

Divide (8)/(7):

$$\frac{A(2,3)}{A(1,3)} = \frac{sin(\theta_A)}{cos(\theta_A)} = \tan tan\, \theta_A$$

$$\boxed{\theta_A = \left(\frac{A(2,3)}{A(1,3)}\right)\ or\ \theta_A = atan(\frac{A(2,3)}{A(1,3)})\text{+pi}}$$

Two possible solutions for θ_A.

$$(7)^2 + (8)^2 = sin(\theta_B)^2$$

$$\frac{\sqrt{A(1,3)^2 + A(2,3)^2}}{-A(3,3)} = \frac{sin(\theta_B)}{cos(\theta_B)} = tan(\theta_B)$$

$$\boxed{\begin{gathered}\theta_B = \left(\frac{\sqrt{A(1,3)^2 + A(2,3)^2}}{-A(3,3)}\right),\ \theta_B = \left(\frac{\sqrt{A(1,3)^2 + A(2,3)^2}}{-A(3,3)}\right) + pi, \\ \theta_B = \left(\frac{-\sqrt{A(1,3)^2 + A(2,3)^2}}{-A(3,3)}\right)\ or\ \theta_B = \left(\frac{-\sqrt{A(1,3)^2 + A(2,3)^2}}{-A(3,3)}\right) + pi\end{gathered}}$$

That makes four possible solutions for θ_B.

$$\tan tan\left(\theta_5\right) = \frac{-A(3,2)}{A(3,1)}$$

$$\boxed{\theta_5 = \left(\frac{-A(3,2)}{A(3,1)}\right) or\ \theta_5 = atan(\frac{-A(3,2)}{A(3,1)})\text{+pi}}$$

$$\boxed{\begin{gathered}\theta_3 = asin(\frac{A(3,4) - d_1 - d_2 + d_5 cos\theta_B}{l_3})\text{ or }\theta_3 = pi - asin(\frac{A(3,4) - d_1 - d_2 + d_5 cos\theta_B}{l_3}) \\ \theta_4 = \theta_B - \theta_3\end{gathered}}$$

$$\theta_1 = atan(\frac{A(2,4) - l_2 \text{sin}sin\,\theta_A + d_4 \text{cos}cos\,\theta_A + d_3 cos\theta_A - d_5 \text{sin}sin\,\theta_A\ \text{sin}sin\,\theta_B - l_3 \text{sin}sin\,\theta_A\ \text{cos}cos\,\theta_3}{A(1,4) - l_2 \text{cos}cos\,\theta_A - d_3 \text{sin}sin\,\theta_A - d_4 sin\theta_A - d_5 \text{cos}cos\,\theta_A\ \text{sin}sin\,\theta_B - l_3 \text{cos}cos\,\theta_3\ \text{cos}cos\,\theta_A})$$

$$\boxed{\theta_2 = \theta_A - \theta_1}$$

To be able to put the gripper in the orientation of facing down (as it has to in order to pick up objects) the special case has to be considered that the A matrix looks as follows:

$$A_{\text{target}} = \begin{bmatrix} \sim & \sim & 0 & \sim \\ \sim & \sim & 0 & \sim \\ 0 & 0 & 1 & \sim \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can simplify the found equations even a little more by substitute

$$\cos cos\left(\theta_B\right) = 1 \; or \; \cos cos\left(\theta_B\right) =- 1$$

$$\sin sin\left(\theta_B\right) = 0$$

In this case θ_A cannot be calculated like above because A(1,3)=0. In this case we calculate the angles in a different order as follows:

$$\boxed{\theta_B = 0 \text{ or } \theta_B = pi}$$

$$\boxed{\theta_3 = asin\left(\frac{A(3,4) - d_1 - d_2 + d_5 cos(\theta_B)}{l_3}\right) \text{ or } \theta_3 = pi - \theta_3}$$

$$\boxed{\theta_4 = \theta_B - \theta_3}$$

It is more complex to calculate θ_2

We substitute:

$$a = l_2 + l_3 cos\theta_3$$

$$b = d_3 + d_4$$

$$c = l_1$$

$$U = 2ac$$

$$V = 2bc$$

$$W = A(1,4)^2 + A(2,4)^2 - a^2 - b^2 - c^2$$

$$T = \frac{V}{U+W} + (\frac{V}{U+W})^2 - \frac{W-U}{W+U} \; or \; T = \frac{V}{U+W} - (\frac{V}{U+W})^2 - \frac{W-U}{W+U}$$

$$\theta_2 = 2(T) \; or \; \theta_2 = 2(pi - (T))$$

θ_A is calculated similarily

$$U2 = l_2 + l_1 \cos cos\, \theta_2 \; + l_3 cos\theta_3$$

$$V2 = d_3 + d_4 + l_1 sin\theta_2$$

$$W2 = A(1,4)$$

$$T2 = \frac{V2}{U2+W2} + (\frac{V2}{U2+W2})^2 - \frac{W2-U2}{W2+U2} \; or \; T2 = \frac{V2}{U2+W2} - (\frac{V2}{U2+W2})^2 - \frac{W2-U2}{W2+U2}$$

$$\theta_A = 2(T2) \; or \; \theta_A = 2(pi - (T2))$$

$$\theta_1 = \theta_A - \theta_2$$

b. Implementation an validation

The implementation in the code is based on the assumption, that in the end we just need the workspace of every angle to be between -60 and +60.

So basically a trial and error method is used. Randomly a first solution is picked and checked, if the angles calculated are in this range and if the position (A(1,4), A(2,4), A(3,4)) is reached. If not other solutions is tried until a solution is found that satisfies our requirements.

If we cannot find any solution, we assume that the given target matrix is out of our defined workspace.

The model is validated by trying the inverse model for different target matrices. The angles needed are calculated using the inverse model and then first tried if the forward model would give calculate the aimed matrix. After that we tried setting the robot to a specific position and orientation and measured if it is in the aimed pose.

c. Conversion angles/steps

To go from degrees to motor steps (units) the motors are put to the extreme values (-1400 and 1400) and the angle range is measured.

This conversion was already described in section 1-c.

d. Minimum/maximum rotation angles for each joint

Angle	Minimum	Maximum
Theta 1	-87.5°	77.5°
Theta 2	-85°	90°
Theta 3	-80°	100°
Theta 4	-75°	95°
Theta 5	-96°	56°

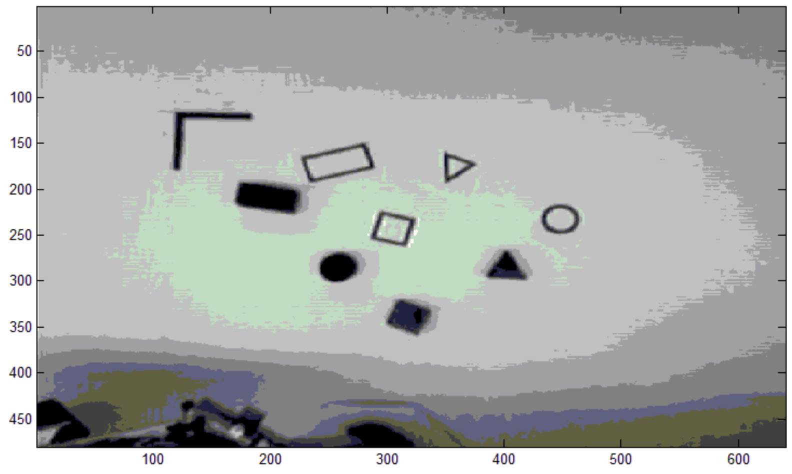
e. Selection among multiple possible solutions

The selection works by trial and error as described above.

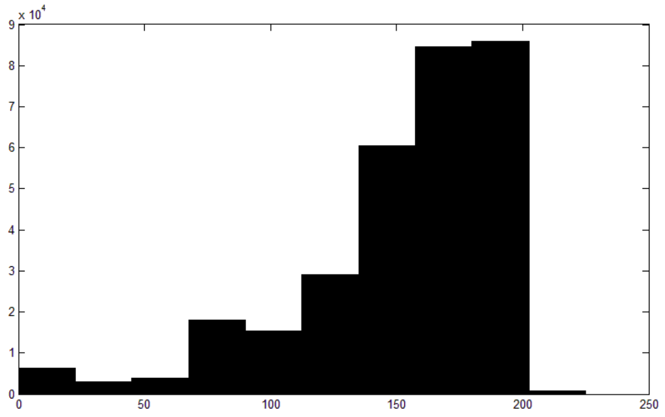
3. Image processing

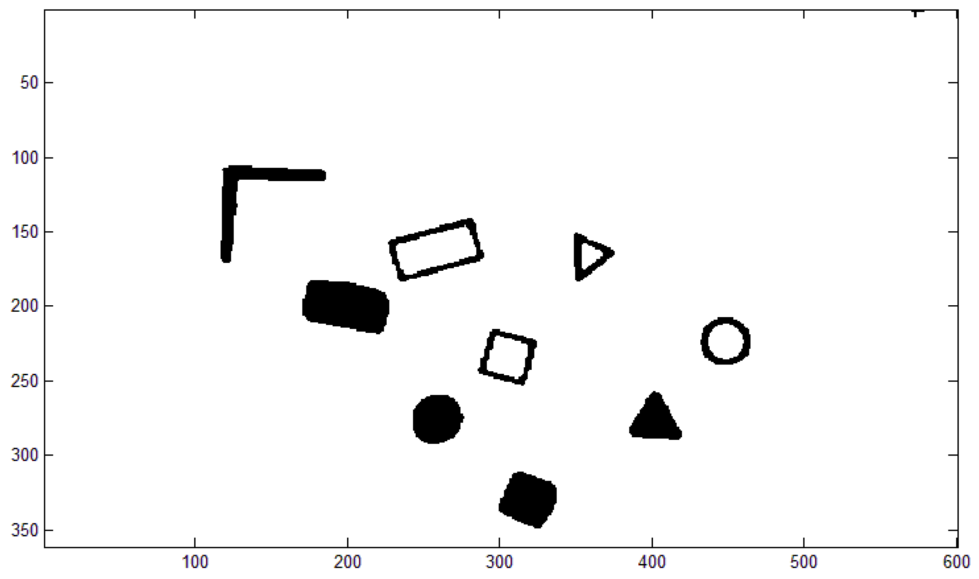
a. Image capture

For the image processing the bmp image is loaded with the function imread(). After that the image is converted to a grayscale image (using rgb2gray()) which is then filtered by a median filter 3x3 to eliminate salt-and-pepper noise.



To create digital image the threshold value needs to be found. Therefore a histogram is created and a threshold value can be derived that distinguishes object pixel from background pixel. A threshold value of 130 is chosen.





The image is converted to a digital (0: background or 1: object) picture by assigning a 1 to every pixel with a value greater than the threshold and assigning a 0 to every pixel with a value smaller than the threshold value.

b. Scaling factor estimation

By measuring the real length of the reference marker size with a measuring tape and counting the belonging pixels in the image a scaling factor can be calculated to convert the distances from pixels in the picture to length in the real world.

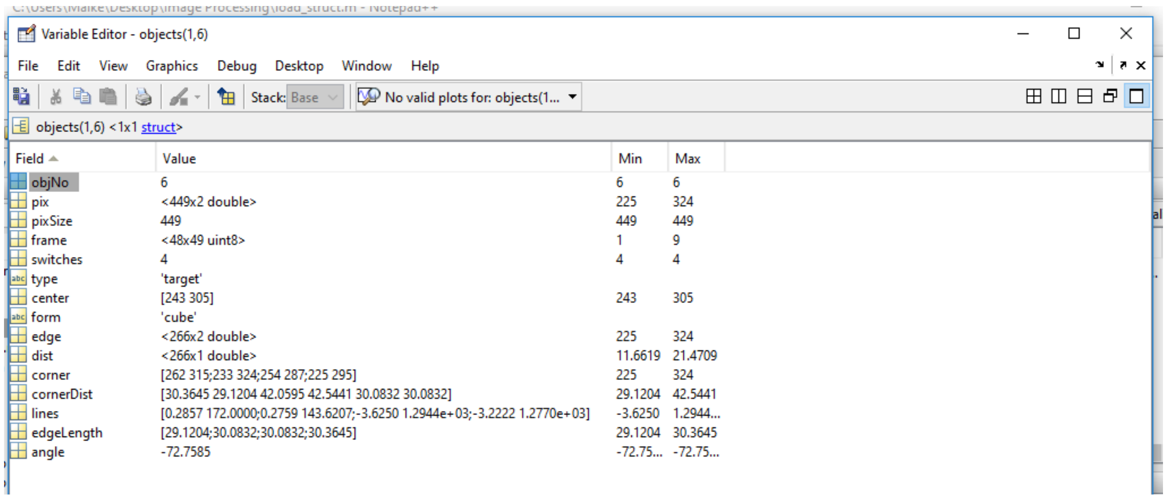
Estimated Scaling Factor=50mm/68pixels

c. Segmentation and recognition of objects in various poses and 4 possible shapes

For segmentation the digital image is processed in a way that there are different numbers assigned to the pixels belonging to different objects. In the code it is done with two for loops, that run raw by raw and column by column and detect if an object pixels has more object pixels around it. If so, they are all assigned the same object number.

After that the pixels (of a frame that contents one object) and their positions are saved in a structure. The structure *objects* will content the same number of entries as there are objects in the picture. In this way every information can be captured and assigned to its object.

At this point of the code it is also detected if an object is full or if it is a target. For that purpose the raw in the middle of the frame containing the object is analysed. The number of changes from background to object and object to background is counted. If there are 4 it is a target, if there are 2 the object is full.


C:\Users\Maike\Desktop\Image Processing\load_struct.m - Notepad++
Variable Editor - objects(1,6)
File Edit View Graphics Debug Desktop Window Help
Stack: Base
No valid plots for: objects(1...

objects(1,6) <1x1 struct>

Field	Value	Min	Max
objNo	6	6	6
pix	<449x2 double>	225	324
pixSize	449	449	449
frame	<48x49 uint8>	1	9
switches	4	4	4
type	'target'		
center	[243 305]	243	305
form	'cube'		
edge	<266x2 double>	225	324
dist	<266x1 double>	11.6619	21.4709
corner	[262 315;233 324;254 287;225 295]	225	324
cornerDist	[30.3645 29.1204 42.0595 42.5441 30.0832 30.0832]	29.1204	42.5441
lines	[0.2857 172.0000;0.2759 143.6207;-3.6250 1.2944e+03;-3.2222 1.2770e+03]	-3.6250	1.2944...
edgeLength	[29.1204;30.0832;30.0832;30.3645]	29.1204	30.3645
angle	-72.7585	-72.75...	-72.75...

d. Object’s position estimation

Now, the centre of every object can be determined by calculating the mean x- and y-values of the object pixels’ positions. The position of every object’s centre is also captured in the structure objects.

For the X-position of object i:

```
xc=round(sum(objects(i).pix (:,1))/size(objects(i).pix,1));
```

For the Y-position of object i:

```
yc=round(sum(objects(i).pix(:,2))/size(objects(i).pix,1));
```

If the centre of the object belongs to the object you can also tell that it is an actual object. If the centre does belong to the background it is the marker.

f. Shape classification

In the beginning of shape classification the pixels belonging to the edges of an object are detected, by noticing if there is a change from background to object or the other way around, and saved in the structure.
After that the distances from the edge pixels to the objects centre are calculated. If they vary just in a range of 5 pixels, it can be assigned as a circle.
To distinguish between a quader (rectangular), a triangle and a cube (square) the corners are detected. The edge pixel with the maximum distance (calculated before) is assigned as a corner. After the first corner is detected, the distances of the corner and the pixels around the first corner are deleted. Then again the maximum of the left distances is assigned as the second corner. In this way four corners are detected. If the minimum distance between the corners is smaller than 15 pixels it is a triangle (because it just has three ‘real’ corners).
After that there are just cubes and quaders to be distinguished. You can tell by the maximum corner distances if it a quader (greater than 60 pixels) or a cube (around 4353 pixels)

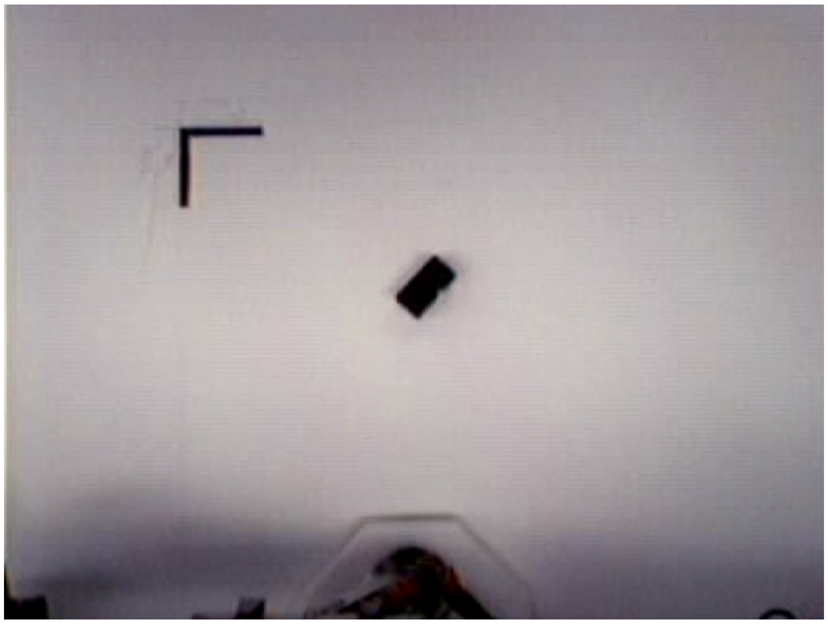
To avoid mistakes there are double checks in the end. It is checked if the frame sizes of the objects are right. For example the frame of an rectangular is bigger than the frame size of a circle.

e. Object’s orientation estimation

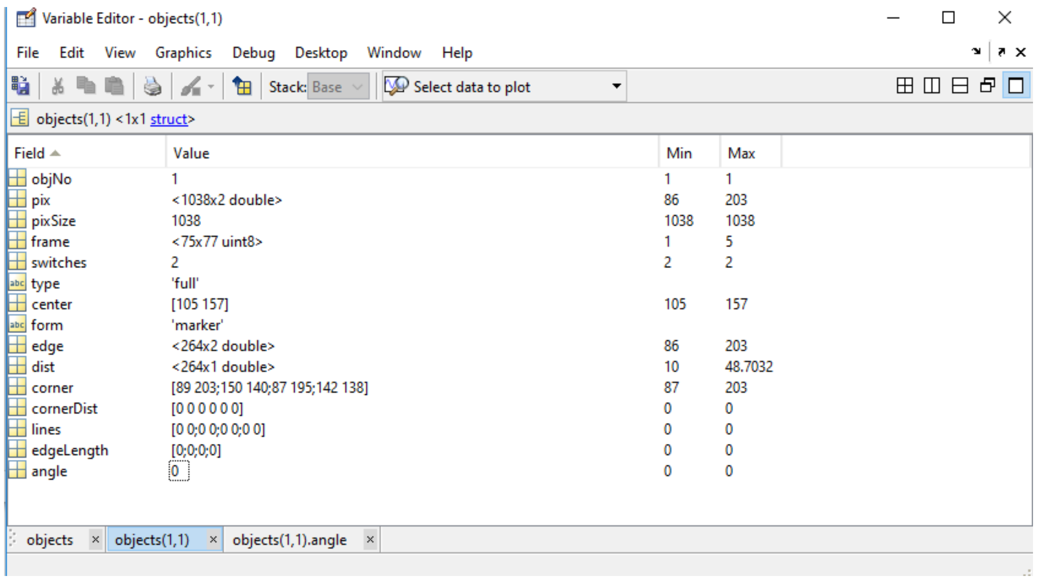
The orientation of cubes and quaders can be calculated by a linear regression through the (for example longest) edge. If you determine the atangens of the calculated slope of the edge you get the angle in which the object is orientated with respect to the reference marker.

g. Implementation and validation

After implementing the ideas described above we took a couple of sample pictures to validate that we can detect the right objects at the right place with the right classifications.

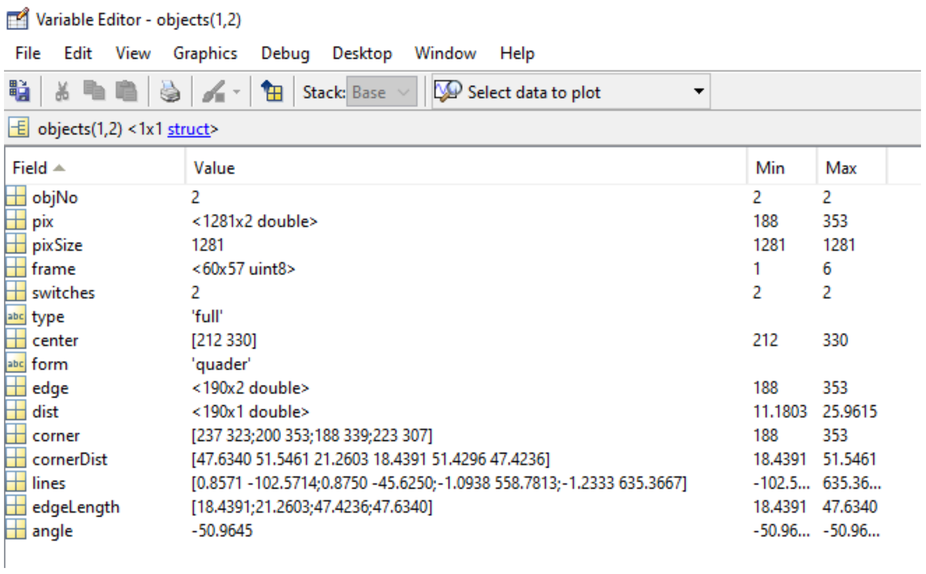


If we process this picture for example we get the following parameter for the marker:


Variable Editor - objects(1,1)
File Edit View Graphics Debug Desktop Window Help
Stack: Base
Select data to plot
objects(1,1) <1x1 struct>
objects
objects(1,1)
objects(1,1).angle

Field	Value	Min	Max
objNo	1	1	1
pix	<1038x2 double>	86	203
pixSize	1038	1038	1038
frame	<75x77 uint8>	1	5
switches	2	2	2
type	'full'		
center	[105 157]	105	157
form	'marker'		
edge	<264x2 double>	86	203
dist	<264x1 double>	10	48.7032
corner	[89 203;150 140;87 195;142 138]	87	203
cornerDist	[0 0 0 0 0 0]	0	0
lines	[0 0;0 0;0 0;0 0]	0	0
edgeLength	[0;0;0;0]	0	0
angle	0	0	0

And the following for the quader:


Variable Editor - objects(1,2)
File Edit View Graphics Debug Desktop Window Help
Stack: Base
Select data to plot
objects(1,2) <1x1 struct>

Field	Value	Min	Max
objNo	2	2	2
pix	<1281x2 double>	188	353
pixSize	1281	1281	1281
frame	<60x57 uint8>	1	6
switches	2	2	2
type	'full'		
center	[212 330]	212	330
form	'quader'		
edge	<190x2 double>	188	353
dist	<190x1 double>	11.1803	25.9615
corner	[237 323;200 353;188 339;223 307]	188	353
cornerDist	[47.6340 51.5461 21.2603 18.4391 51.4296 47.4236]	18.4391	51.5461
lines	[0.8571 -102.5714;0.8750 -45.6250;-1.0938 558.7813;-1.2333 635.3667]	-102.5...	635.36...
edgeLength	[18.4391;21.2603;47.4236;47.6340]	18.4391	47.6340
angle	-50.9645	-50.96...	-50.96...

This corresponds to the measurements done with the real set up. The conversion to millimeters is in a different code fragment.

4. Integration of modules

a. Generating ROBIX commands/script

The generation of macro instructions in ROBIX script language were developed upon the Robix.m and pick_place.m Matlab scripts. The first step was to use the fopen command to write a .txt file with the instructions for the robot.

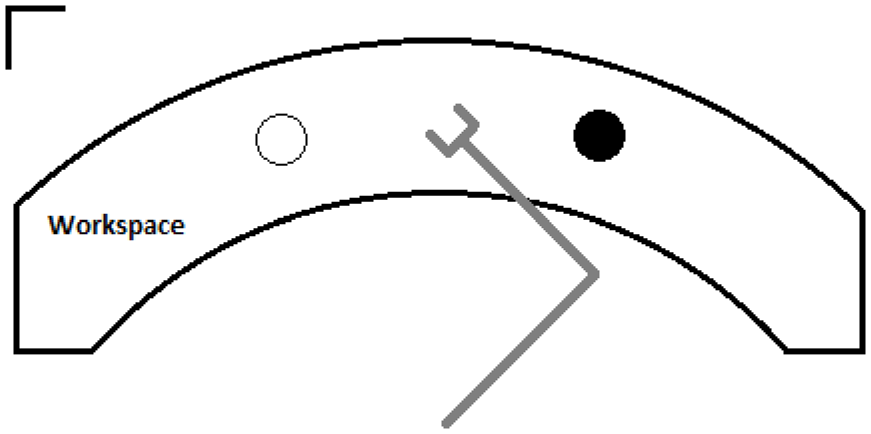
The pick_place.m script was used every time a new set of angles was calculated to execute a new manipulation task, either grabbing or dropping an object. It operates by first moving joints 1 and 2 at the same time to get an approximate position in the x and y axes. After that, it rotates joint 5 to align it with the object's orientation. Then, joints 3 and 4 move to get the final position over the object or target. A condition is checked to see if the gripper has to open or close depending on if the robot is currently carrying an object or not. Finally, joint 3 and 4 return to a zero position so that the ROBIX can move freely while performing the next operation.

The instructions are written in the text file using the fprintf command. ROBIX command move a_1 to x_1, a_2 to x_2; were used to move either one or to joints at the same time. An example of a .txt generated to pick up a circle and drop it in its target is provided next:

```
move 6 to -1300;
move 1 to 288, 2 to -778;
move 5 to -972;
move 3 to 918, 4 to -840;
move 6 to -250;
move 3 to 0, 4 to 0;
move 1 to 662, 2 to -622;
move 5 to -403;
move 3 to 918, 4 to -840;
move 6 to -1300;
move 3 to 0, 4 to 0;
```

b. Workspace definition and management within code

The workspace for the robot was defined by the solutions that can be obtained from the Inverse Kinematic Model, but is managed under the translation2.m script. The important configurations in which the robot actually needs to be able to operate are when the gripper is pointing downwards (when z_6=-z_1), in order to pick and place objects from above and when the gripper is pointing outside (when Θ_3=0 and Θ_4=0) to move above the objects.

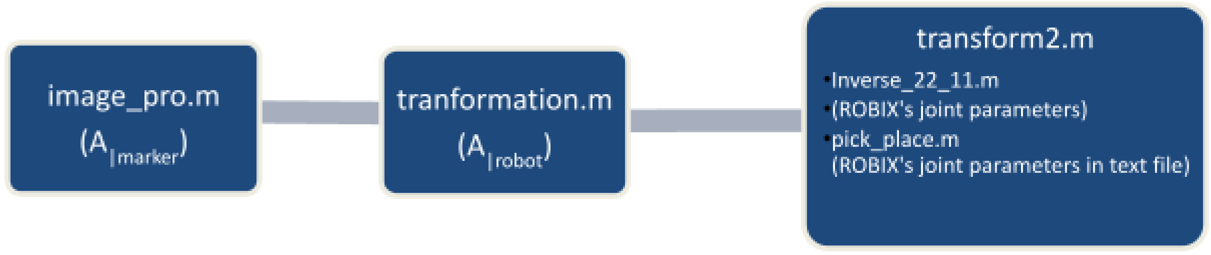


According to this, the workspace over the surface of the table was defined by a hollow semicircle including all the positions that the gripper can take when it is facing down and with an elevation going from z=0 mm to z=15 mm. These heights were selected taking into consideration that the objects have an average height of 20mm, so at least the gripper can't exceed an elevation of 15 mm in order to grab them adequately. To achieve the evaluation of the Inverse kinematics model with different heights, a loop was implemented in the code to change the value of the z parameter in the A matrix of each object (A(3,4)) from 0 to 15. In this way, the Inverse Kinematic Model can be solved with different values of height, because the robot's workspace is bigger if we take into consideration that the gripper is not touching the table. This happens because Θ_3+Θ_4 becomes bigger than -90° and the reach of the gripper grows.

The center of each object or target was used to determine if it is inside our outside the workspace. This was done by evaluating the results from the Inverse kinematics model, if the results had an angle for a joint which was outside of the range of -90°<Θ_i<90° (the maximum range of movement for each of the joints in the ROBIX), then the object was outside of the workspace and it was ignored.

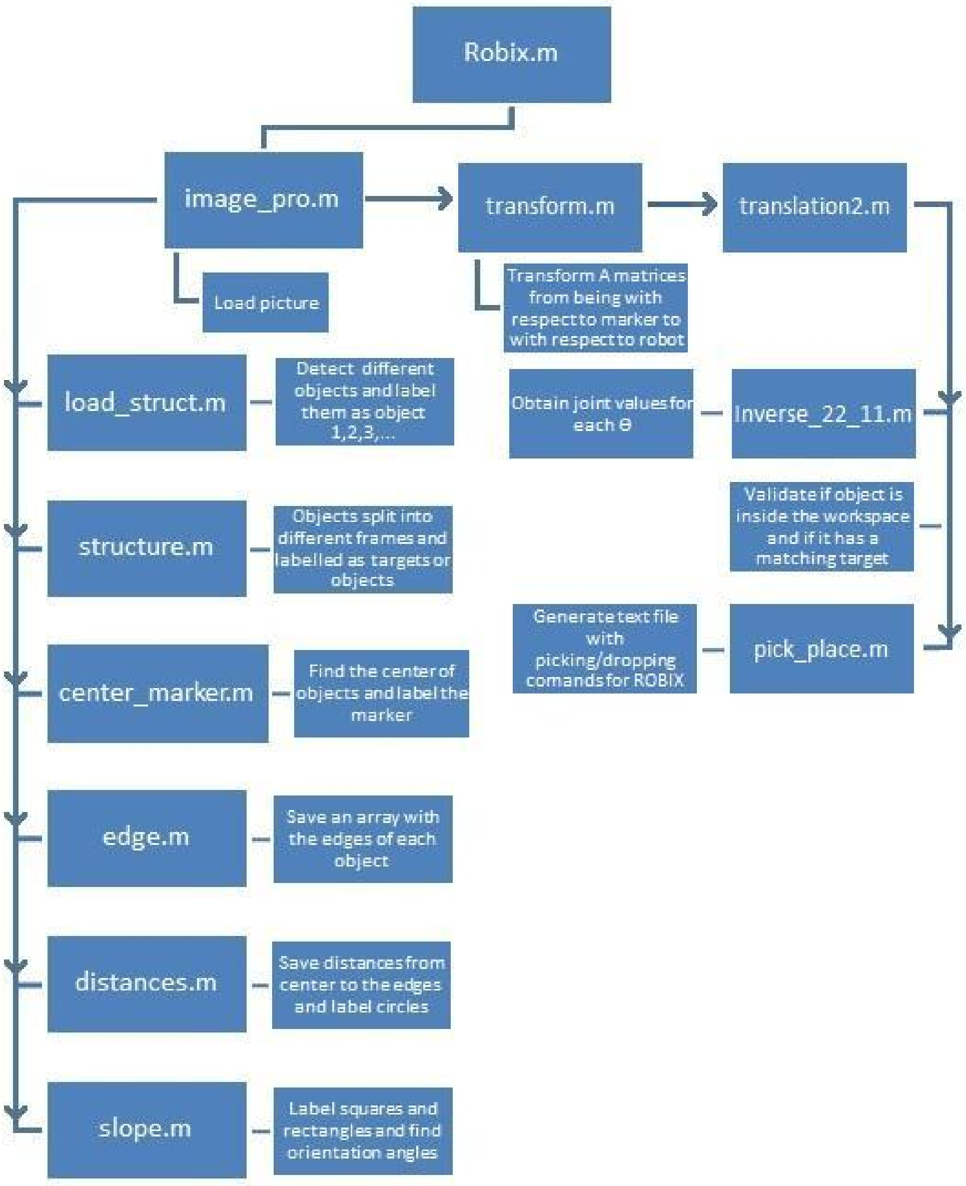
c. Transfer of information between vision, inverse kinematics and ROBIX

All the information obtained from the image processing is saved into a structure named "objects" with dimensions [1 X Number of objects]. After that, the transform.m script makes the scaling and transformations of the objects into the robot's point of view generating new A matrices for the objects and also saving them into the structure. After that, the translation2.m script identifies each object, one by one and runs the inverse kinematics model, executed in the Inverse_22_11.m script, which takes the information of the current A matrix and saves the resulting ROBIX joint parameters in another variable that is finally written in the text file which is going to be exported to the ROBIX's software.

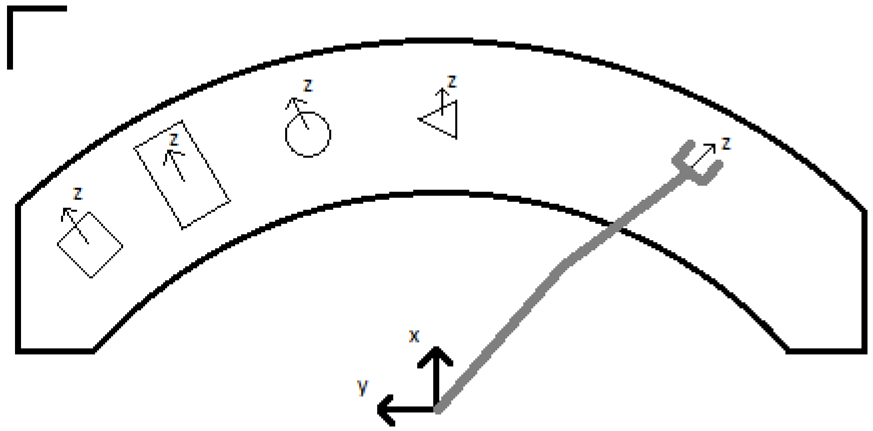


d. Implementation

The implementation of the code and order of operations was done according to the steps in the next diagram:


Robix.m
image_pro.m
transform.m
translation2.m
Load picture
Transform A matrices from being with respect to marker to with respect to robot
load_struct.m
Detect different objects and label them as object 1,2,3,...
Obtain joint values for each θ
Inverse_22_11.m
Validate if object is inside the workspace and if it has a matching target
structure.m
Objects split into different frames and labelled as targets or objects
Generate text file with picking/dropping comands for ROBIX
pick_place.m
center_marker.m
Find the center of objects and label the marker
edge.m
Save an array with the edges of each object
distances.m
Save distances from center to the edges and label circles
slope.m
Label squares and rectangles and find orientation angles

Each of the objects/targets was numbered from 1 to *n* according to their center x position with respect to the robot's base, being 1 the object located farther away from the base and *n* the one closer. If two objects have the same x coordinates then the same criteria is applied with respect to the y position. The path planning for the pick and place operations was done following this numeration. Starting from object 1, the code will check if it is an object or a target. If it is an object, joints 1, 2 and 5will move to approach the object with an orientation that leaves the gripper's x axis with an orientation that is normal to the semicircle constituting the workspace. This happens because even when the orientation of targets and objects is calculated in a previous step, it is not considered. Because of this, all objects and targets must have an imposed orientation which is also normal to the semicircle constituting the workspace as shown in the diagram below. After this alignment, joints 3 and 4 come down to grab the object. Joints 3 and 4 then return to the 0 position allowing the robot to move in a similar way to drop the object while moving above the other objects and avoiding a collision with them.



The controller includes a feature that will allow checking if an object or a target is outside of the workspace and also if there is an object which has no target with the same shape. According to this, the Robix.m script can return three kinds of warnings:

1. 'Warning, object number *i* is out of the workspace': Occurs when the result from the inverse kinematic model for the center of an object has angles outside of the range of $-90° < \Theta_i < 90°$ (the maximum range of movement for each of the joints in the ROBIX). In this case, the object is ignored.
2. 'Warning, target number *i* is out of the workspace': Occurs when the result from the inverse kinematic model for the center of a target has angles outside of the range of $-90° < \Theta_i < 90°$ (the maximum range of movement for each of the joints in the ROBIX). In this case, the ROBIX will pick the object associated with that shape and then drop it outside of the workspace.
3. 'Warning, there is no target detected for object number *i* ': Occurs when there is no target associated with the shape of the current object. In this case, the ROBIX will pick the object associated with that shape and then drop it outside of the workspace.

This warning messages are displayed in Matlab's Command Window and doesn't interfere with the other operations of the robot.

e. Performance analysis

The controller successfully allows the classification of all of the objects as objects or targets, and as circles, triangle, squares or rectangles. The image processing was tested with a lamp providing additional light to avoid the shadows and without it, having accurate results in the classification for both of the cases. There is no number of objects that can be processed, picked and placed as long as they can be placed within the workspace. If there is more than one target of the same shape, the one that has a bigger x coordinate with respect to the robot will be used. As mentioned before, even when the orientation of the objects is calculated it is not used and they need to have a certain orientation in order for the gripper to be able to grab them, if not the robot will get to the object's position but with an incorrect orientation and will not grab it.

The precision of the controller regarding the A matrices for objects and targets is very good. This was checked by comparing the x and y coordinates for the center of different objects with measurements done by hand, showing a precision of +/- 5 mm. Nevertheless, the position of the robot's gripper was not always accurate, this due to the inaccuracy of the ROBIX but also to the inaccurate calibration of the joints offsets. The smoothness of the approach was affected by the same issues and the grasp of objects was adequate, as they didn't fall when the robot was moving them towards the targets.

5. Discussion

a. Observations, difficulties and learning experience

The practical application of theoretical concepts like the direct and inverse models, the Denavit–Hartenberg parameters and the image processing was very important to comprehend both how they work and how to use them.

We realized that having the correct Denavit–Hartenberg parameters is crucial to obtain the solutions for the direct and inverse kinematic models, but that also the calculations of these could be simplified depending on the location of the axis for each joint. The biggest problem faced with the development of the project was, in fact, the solution of the inverse kinematic model. The problem was first approached by trying to find a general solution that could work for any angle. As this solution wasn´t found, restrictions were introduced. First, because none of the physical joints could reach a value that was outside of the range of $-90° < \Theta_i < 90°$. Then, for joints 1 and 2, the restriction in the range of $-60° < \Theta_i < 60°$ was applied, because angles outside of the range were hard to capture in the pictures taken with the camera. Finally, a restriction for $-90° < \Theta_3 + \Theta_4 < 90°$ was introduced. In the end, very few of this positions were used, the ones we are really interested in are when the gripper's z=-z from the robot's base, because this is the picking and placing position of the gripper. If we had only looked for the solution of these cases since the beginning, the difficulty and time spent in getting a very general solution could have been spared.

c. How to improve the solution

Apart from the imprecision of the ROBIX itself and its calibration, the best improvement that could be implemented to the solution is to do a better path planning which allows different objects' orientations. To do this, an extra step should be included in the actual path planning. First, we need to input the orientation angles (which are currently calculated but not used) into the A matrix of each object with respect to the robot's base. Then, the values of joints 1 and 2 will change, allowing the robot first to align the gripper with the orientation to the object and then to come down and grab it or drop it, contrasting with the simple orientation task that is currently used.

With this, the order in which objects are picked would also change to avoid collisions, because know objects could be approached from any direction. To implement this, the controller should analyze which objects have free space in the direction of their z axis and approach them by that face. Then, check for the next set of objects, taking into consideration that some obstacles have been removed.

The above shows that the step missing for the complete implementation of the project is a well done path planning, allowing objects and targets to have any orientation and also avoiding the possible collisions that this may cause.