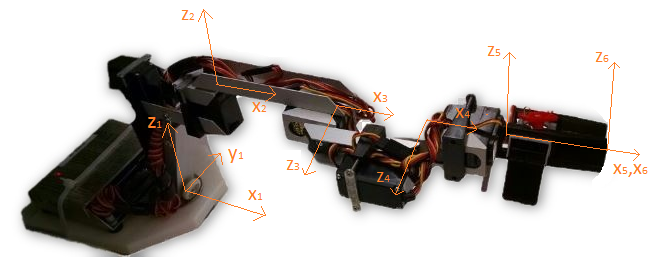
# 1. Robot modeling

### 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 z1 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 z6=0 in respect to z1, 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** | **li (mm)** | **di** | **αi** | **θi** |
| 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 zi axis of an angle θi;
2. Translation along the zi axis of a distance di;
3. Translation along the xi+1 axis of a distance li;
4. Rotation around the xi+1 axis of an angle αi.



The matrices obtained are:



The product of the A matrices and simplification of Agripper/base were done using Matlab:



### c. Implementation of forward kinematics (how was it done and validated?)

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 ϴi 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 Agripper/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:



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

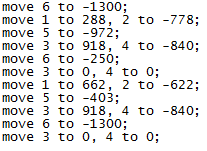
# 4. Integration of modules

### 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.

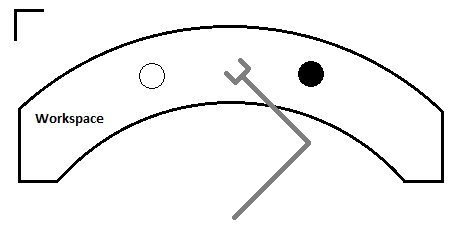
The pick\_place.m script was used every time a new set of angles was calculated to execute a new manipulation task, either grabbing or dropping an object. It operates by first moving joints 1 and 2 at the same time to get an approximate position in the x and y axes. After that, it rotates joint 5 to 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 a1 to x1, a2 to x2; 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:



### 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 z6=-z1), in order to pick and place objects from above and when the gripper is pointing outside (when ϴ3=0 and ϴ4=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 ϴ3+ϴ4 becomes bigger than -90° and the reach of the gripper grows.

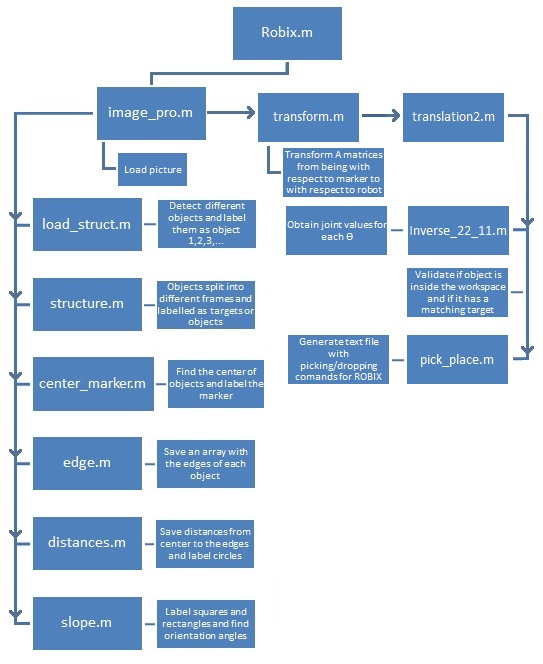
The center of each object or target was used to determine if it is inside our outside the workspace. This was done by evaluating the results from the Inverse kinematics model, if the results had an angle for a joint which was outside of the range of -90°<ϴi<90° (the maximum range of movement for each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

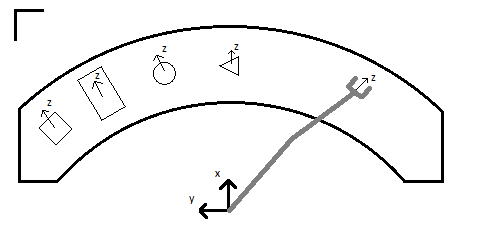
All the information obtained from the image processing is saved into a structure named “objects” with dimensions [1 X Number of objects]. After that, the transform.m script makes the scaling and transformations of the objects into the robot’s point of view generating new A matrices for the objects and 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 completed according to the steps in the next diagram:



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°<ϴi<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°<ϴi<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°<ϴi<90°. Then, for joints 1 and 2, the restriction in the range of -60°<ϴi<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°<ϴ3+ ϴ4<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.