1st Lab Assignment - Serial Manipulators

Miguel Graça (90142), Ricardo Santos (90178), Inês Ferreira (90395)
Professor João Sequeira

***Abstract*— Serial manipulators are a category of robot manipulator that consist of several links connected in series by various types of joints, typically revolute or prismatic. Serial manipulators have been used in manufacturing industries for certain production tasks, such as assembling and handling materials. The aim of this assignment is to analyze the Niryo One serial manipulator, using a simulator, in a tilling task, consisting in moving 4 plastic blocks from an origin position to a target area.**

I. Introduction

The following report is organized into several sections. In ***System Architecture***, a detailed overview of the developed system for simulation is given. In ***Direct Kinematics*** the frame system for the robotic manipulator is defined and the transformation matrix that maps joints to a pose is calculated. In ***Inverse Kinematics***, we describe how to find the joints corresponding to a given pose. In ***Robot Trajectory***, the calculation of the poses that constitute the full trajectory is explained. The correspondence between orientation angles of the gripper and different frames systems is also studied. In ***Kinematics Testing***, the proposed approaches for Direct and Inverse Kinematics are evaluated by comparing them to built-in functions in the simulator *api*. In ***Analysis of the Effect of Uncertainties***, the impact of uncertainties in the joint values is studied, emphasizing how this can affect the gripper position and the robot performance in the tilling task. Finally, in ***Scalability and Future Work***, we present how our studies can translate to the real robot and some other studies we would like to do in it.

II. System Architecture

Figure 1 depicts the developed system for the simulation of a tilling task, using the Niryo One serial manipulator. The desired trajectory for the robot is read from a file (*positions_file*), using the function GET_TRAJECTORY, which outputs a set of poses. Each pose is characterized by position (x,y,z) and orientation (α,β,γ).

For each desired pose, the function INVERSEKINEMATICS is applied to obtain the corresponding set of joints. Within this function, INVERSEKINEMATICSAUX is run two times, with the second run using the obtained solutions from the first run. A set of four possible solutions is obtained, of which the best one is chosen, using the function SELECT_SOLUTION.

When uncertainties are considered in the simulation, gaussian noise is added to these joints; otherwise, the joints are fed directly to the Niryo One function MOVE_JOINTS. This function applies the joint values to the robot to make it move. This movement can then be seen in simulation, on RVIZ.

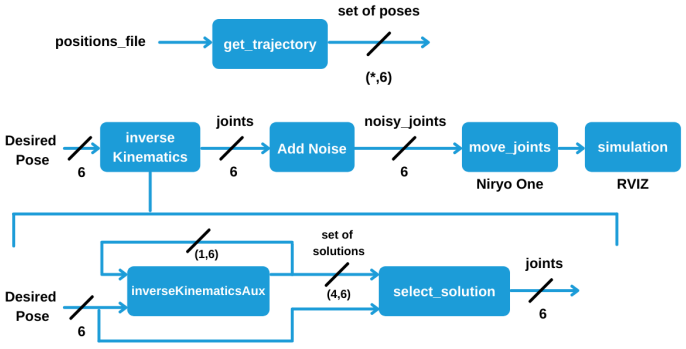

Fig. 1: Overall System Architecture

III. Direct Kinematics

Using the Denavit-Hartenberg convention for frame assignments, the chosen frame system is presented in Figure 2 and the respective dimensions in Table I.

TABLE I: Physical Dimensions of the Robotic Manipulator

	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
Distance (mm)	103	80	210	30	41.5	180	23.7	5.5

Note that some of the defined frames are equivalent to frames in the simulator of the robotic manipulator. For example, frame 0 is equivalent to the frames labeled as *world* or *base_link* in the simulator. The frame in the gripper, frame 6, is the same as the frame *hand_link* in the simulator.

The goal of Direct Kinematics is, given the joint values, to get the pose of the gripper with respect to world coordinates. Therefore, Direct Kinematics returns the coordinates and position of frame 6 with respect to frame 0. Table II depicts the relationships between frames, obtained after defining an appropriate frame system. This mapping is necessary to find the complete transformation matrix between frames 6 and 0.

TABLE II: Denavit-Hartenberg Table

Frames	a_{i-1}	α_{i-1}	d_i	θ_i
$0 \rightarrow 1$	0	0	$L_1 + L_2$	θ_1
$1 \rightarrow 2$	0	$\pi/2$	0	$\theta_2 + \pi/2$
$2 \rightarrow 3$	L_3	0	0	θ_3
$3 \rightarrow 4$	L_4	$\pi/2$	0	θ_4
$4 \rightarrow 5$	0	$-\pi/2$	$L_5 + L_6$	θ_5
$5 \rightarrow 6$	$-L_8$	$\pi/2$	L_7	$\theta_6 - \pi/2$

To obtain the transformation matrices for each pair of consecutive frames, the matrix which maps frame i to frame $i-1$, as described below, is applied.

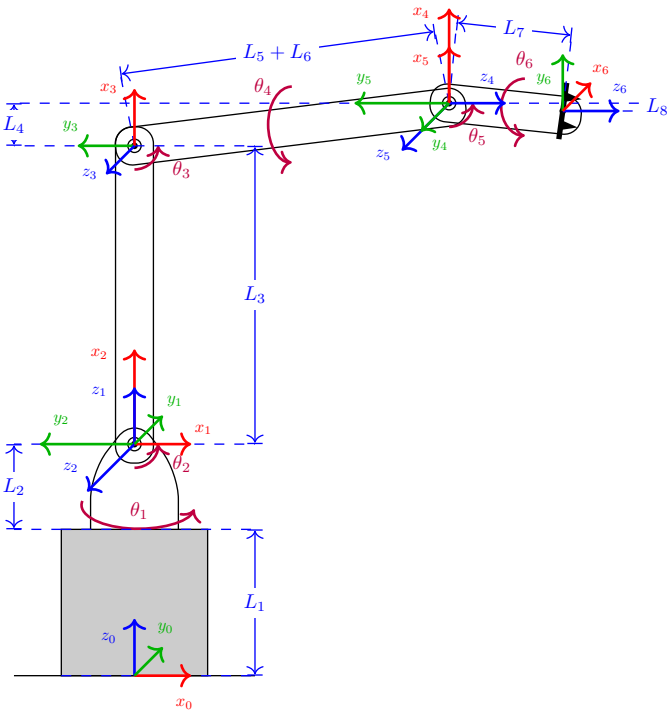

Fig. 2: Frames used for the Direct Kinematics

$$ {}^{i-1}_{i}T = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\ s_{\theta_i}c_{\alpha_{i-1}} & c_{\theta_i}c_{\alpha_{i-1}} & -s_{\alpha_{i-1}} & -s_{\alpha_{i-1}}d_i \\ s_{\theta_i}s_{\alpha_{i-1}} & c_{\theta_i}s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & c_{\alpha_{i-1}}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{1}T = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 & 0 \\ 0 & 0 & 1 & L_1 + L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} -s_{\theta_2} & -c_{\theta_2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c_{\theta_2} & -s_{\theta_2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & L_3 \\ s_{\theta_3} & c_{\theta_3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} c_{\theta_4} & -s_{\theta_4} & 0 & L_4 \\ 0 & 0 & -1 & -(L_5 + L_6) \\ s_{\theta_4} & c_{\theta_4} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_{\theta_5} & -c_{\theta_5} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{6}T = \begin{bmatrix} s_{\theta_6} & c_{\theta_6} & 0 & -L_8 \\ 0 & 0 & -1 & -L_7 \\ -c_{\theta_6} & s_{\theta_6} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The final transformation matrix is given by:

$$ {}^{0}_{6}T = {}^{0}_{1}T \; {}^{1}_{2}T \; {}^{2}_{3}T \; {}^{3}_{4}T \; {}^{4}_{5}T \; {}^{5}_{6}T $$

To find the coordinate values of the gripper using a certain set of joint angles, these must be substituted in matrix ${}^{0}_{6}T$. This matrix has a defined structure

$$ T = \left(\begin{array}{c|c} R & P \\ \hline 0 & 1 \end{array} \right) $$

where R is the 3 by 3 rotation matrix that defines the gripper orientation with respect to the world frame and $P = (x, y, z)^T$ are the gripper's coordinates in the world frame.

IV. Inverse Kinematics

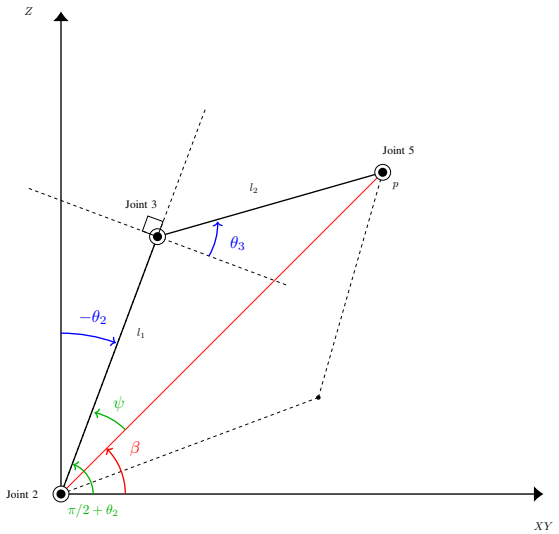

Fig. 3: Geometry of a RRR manipulator

Given a pose $(x, y, z, \alpha, \beta, \gamma)$, where x, y, z are Cartesian coordinates and α, β, γ are Euler angles related to the orientation, inverse kinematics determines the corresponding values in joint space. To place the end-effector in an arbitrary point, at least six degrees of freedom are necessary. The Niryo One robot satisfies this constraint. However, there is not a closed form solution for inverse kinematics for the Niryo One robot, mainly due to the distance L_8. Therefore, an approximate, iterative solution is considered.

In the developed inverse kinematics, joints 1 to 3 are calculated geometrically, while joints 4 to 6 are calculated algebraically. Joint 1 is given directly by the following equation

$$ \theta_1 = atan2(y, x) \tag{1} $$

where $atan2$ returns the arc-tangent, considering the appropriate quadrant. Joints 2 and 3 can be computed using planar geometry, given that the coordinates of joint 5, as depicted in figure 3, are known. There, $l_1 = L_3$ and $l_2 = \sqrt{L_4^2 + (L_5 + L_6)^2}$. Joint 2 is considered the origin of this auxiliary system of axis. The coordinates of joint 5 can be obtained taking into consideration that

$$ {}^{0}_{5}T = {}^{0}_{6}T \cdot ({}^{5}_{6}T)^{-1}. $$

Particularly, since we are only interested in the position, we can write

$$ {}^0_5P = {}^0_6P - {}^0_6R \cdot {}^5_6R^T \cdot {}^5_6P. $$

Simplifying,

$$ {}^0_5P = {}^0_6P - {}^0_6R \cdot [-L_8 s_{\theta_6} \quad -L_8 c_{\theta_6} \quad L_7]^T, \tag{2} $$

where ${}^0_6P$ and ${}^0_6R$ represent the desired pose (position and orientation, respectively), which is known. In order to obtain the coordinates of joint 5 in the frame considered in figure 3, $(0, 0, L_1 + L_2)^T$ is subtracted from ${}^0_5P$. An initial approximation, considering $L_8 = 0$, is made, eliminating the dependence in equation 2 of θ_6. Furthermore, it is evident by figure 3 that, geometrically, there are two solutions for the values of joints 2 and 3. However, for the problem at hand, only the solution with the marked geometry is useful, which leads to a single feasible solution for θ_2 and θ_3.

Applying the law of cosines to the angles of figure 3 and some algebraic manipulations, Equation 4 is obtained

$$ \theta_2 = \psi + \beta - \frac{\pi}{2}, \tag{3} $$

$$ \theta_3 = arcsin\Big(\frac{||p||^2 - l_1^2 - l_2^2}{2 l_1 l_2}\Big), \tag{4} $$

where $||p|| = \sqrt{p_{XY}^2 + p_Z^2}$, $\beta = atan2(p_Z, p_{XY})$ and $\psi = arccos(\frac{||p||^2 + l_1^2 - l_2^2}{2 l_1 ||p||})$. Finally, note that the value of θ_3 depicted in figure 3 is not exactly the value of joint 3. For that, the effect of the lengths of the robot arm, L_4, L_5 and L_6, must be subtracted. In fact, $\theta_3 = \theta_3 - arctan(L_4, L_5 + L_6)$.

After computing joints 1 to 3, Equation 5 is obtained

$$ {}^3_6T = {}^0_3T^-1 \cdot {}^0_6T, \tag{5} $$

where the matrices in the right-hand side are known. For the given task, and since the gripper is desired to be facing down at all times, θ_4 can be forced to be 0 for every desired pose. So, matrix ${}^3_6T$ can also be written as

$$ {}^3_6T = \begin{bmatrix} c_{\theta_5} s_{\theta_6} & c_{\theta_5} c_{\theta_6} & s_{\theta_5} & L_7 s_{\theta_5} - L_8 c_{\theta_5} + L_4 \\ s_{\theta_5} s_{\theta_6} & s_{\theta_5} c_{\theta_6} & -c_{\theta_5} & -L_8 s_{\theta_5} - L_7 c_{\theta_5} - L_5 - L_6 \\ -c_{\theta_6} & s_{\theta_6} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

Equating 5 and 6, joint 5 is obtained directly from the element (2,3)

$$ \theta_5 = arccos(-{}^3_6T(2,3)) \ \vee \ \theta_5 = -arccos(-{}^3_6T(2,3)) \tag{7} $$

The value for joint 6 can also be obtained from the matrix ${}^3_6T$:

$$ \theta_6 = atan2\Big(\frac{{}^3_6T(0,0)}{cos(\theta_5)}, \frac{{}^3_6T(0,1)}{cos(\theta_5)}\Big). \tag{8} $$

Considering the physical limitations of the serial manipulator, not every possible angle is valid for the joints. Therefore, after calculation, for each joint, a comparison is done with the possible limits to ensure validation. These limits, depicted in Table III, were extracted from the simulator.

TABLE III: Joint Limits Table

Joint Angle	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Lower Limit (radians)	-3.05	-1.57	-1.40	-3.05	-1.75	-2.57
Upper Limit (radians)	3.05	0.64	1.57	3.05	1.92	2.57

The Inverse Kinematics is executed two times. On the first run, it outputs two solutions, depending on the sign of θ_5, and the 5.5 mm are not considered as an approximation. For each solution, the Inverse Kinematics is then run again, now using the value for θ_6 previously obtained and considering $L_8 = 5.5$ mm in equation 2. In total, four solutions are obtained and the final output is the best solution among these four. The best solution is verified by checking if the joints are within the physical limits and running the direct kinematics to compare the obtained position with the intended position. Two iterations were enough to yield position errors in the order of the tenths of millimeter.

V. Robot Trajectory

To define the robot trajectory, some measurements were taken in the lab. For this reason, the velcros attached to the lego pieces and to the gripper are considered to have a height of 4 mm. For all other dimensions, the provided lab layout is used. It is important to note that, to control a real robot, calibration should always be the first thing to do before any action.

The robot trajectory is initiated in the *home configuration*. It is the final pose of the robot after calibration and it is also the pose of the robot when the simulator is launched. The the robot motion starts by setting all joints to zero.

The process of picking up one piece from the origin area and placing it in the target area is made up by 6 poses - 3 poses to pick up the piece and 3 poses to place it. To pick up a piece, the gripper moves to the center of the origin area to a height a bit higher than the top lego. The arm velocity is slowed down and the gripper approaches the top piece.

The gripper stops when it hits a height difference of 4 mm from the velcro of the top piece. This is to account for the velcro in the gripper. Furthermore, since the velcro's strength was tested, the velcros should not be forced together. A weak connection is enough for the piece to be transported without falling. A complete connection between the velcros hinders the separation process and it may result in a decalibration of the robot. After connecting to the piece, the arm moves up to the same position where it was before picking the piece.

Notice that the x and y coordinates for these three poses in the origin area are the same and correspond to the center of the origin area ($x = 0.1075\ m$, $y = -0.180\ m$). The heights are calculated using the lego and velcro heights. The arm velocity is now set to the maximum before moving to the target area.

The process of placing pieces is a bit different. It is also made up of three poses. The second and third are always, respectively, the final position of the piece and a position directly above it. The first position depends on the piece. The initial strategy is to approach the final position vertically. Due to the uncertainties in the placement of gripper, this only works for the first piece. After that, a simultaneously lateral and vertical strategy is applied. This can help to position pieces that were initially

misplaced in the correct position. Figure 4 shows a 2D view of the target area, divided in 4 small areas where the lego pieces should be placed. The arrows show how the final position of the pieces are approached by the gripper.

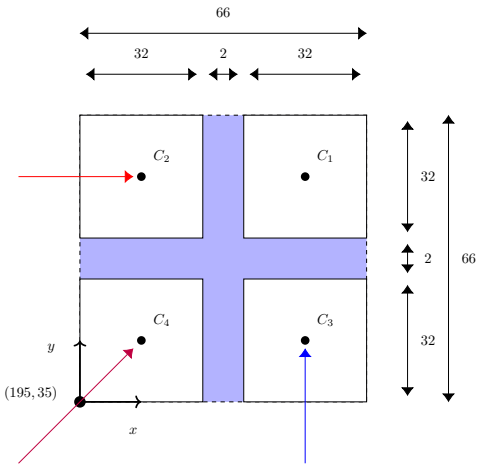

Fig. 4: Dimension of the target area and distribution of the pieces

Note that the pieces are placed the farthest away from themselves inside the target area to account for uncertainties in the gripper position. Table IV shows the centers of each piece.

TABLE IV: Final positions of the pieces in the target area

	C_1	C_2	C_3	C_4
$x(mm)$	245	211	245	211
$y(mm)$	85	85	51	51

When the gripper is placing the pieces, the velocity of the gripper is slowed down for a smoother separation of the velcros and less calibrations errors. After all pieces are positioned, the joints are again set to zero. Throughout the placement of every piece, the orientation of the gripper is set so that it is always facing down.

VI. Velcro Orientation

The lab experiments showed that the lego tower is more stable when the orientation of the velcros is alternated. Therefore, the orientation of the gripper for each piece is changed to match both velcros. The orientation of the gripper is controlled with the α, β and γ angles in the MOVE_POSE function.

However, a problem arises to control the orientation. The orientation angles used in the MOVE_POSE and GET_POSE functions do not match the orientation given by the final rotation matrix calculated with the Direct Kinematics. This is an important issue, as a systematic way to compare poses between the Direct Kinematics and the *api* functions is necessary. The function MOVE_POSE uses coordinates and orientation of the frame *tool_link* in the simulator. Since this frame has the same origin as frame 6, but a different axis, the x, y and z coordinates are the same as the ones given by the developed Direct Kinematics, but the orientation angles differ. Figure 5 depicts the correspondence between the velcro orientation, the axis in frame 6, and the axis in the *tool_link* used for the MOVE_POSE and GET_POSE functions.

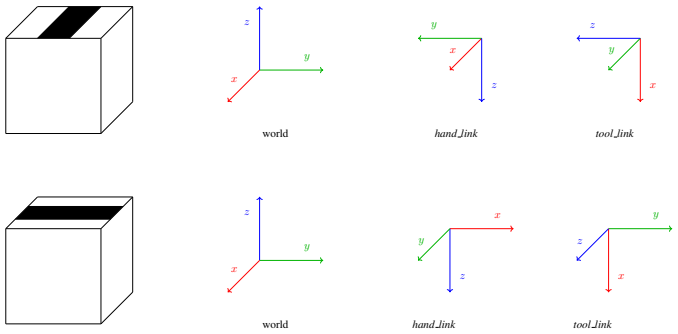

Fig. 5: Orientation of the world frame and the two frames in the gripper in function of the piece orientation.

The diagram above shows that the *hand_link* frame can be obtained by the following rotation matrices for the top and bottom pieces, respectively:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

These two matrices can be achieved using the fixed angle convention $\alpha = 0$, $\beta = 0$, $\gamma = \pi$ and $\alpha = \pi/2$, $\beta = 0$, $\gamma = \pi$. On the contrary, the orientation of the *tool_link* frame is obtained, for the top piece, rotating around x, y, z (fixed angles) with angles $\alpha = 0$, $\beta = \pi/2$, $\gamma = -\pi/2$, and for the bottom piece with angles $\alpha = 0$, $\beta = \pi/2$, $\gamma = 0$.

Therefore, to pick up a piece with the top orientation in Figure 5, the pose sent to the *move_pose* function should have orientation angles $\alpha = 0$, $\beta = \pi/2$, $\gamma = -\pi/2$. If the Direct Kinematics are used, the orientation angles of the pose are $\alpha = 0$, $\beta = 0$, $\gamma = \pi$. Since the Inverse Kinematics also uses the frame 6 as the gripper frame, to calculate the joints correspondent to a pose with this orientation, the orientation angles must be $\alpha = 0$, $\beta = 0$, $\gamma = \pi$.

On the other hand, to pick up a piece with the bottom orientation in Figure 5, the pose sent to the *move_pose* function should have orientation angles $\alpha = 0$, $\beta = \pi/2$, $\gamma = 0$. If the Direct Kinematics are used, the orientation angles of the pose are $\alpha = \pi/2$, $\beta = 0$, $\gamma = \pi$. Since the Inverse Kinematics also uses the frame 6 as the gripper frame, to calculate the joints correspondent to a pose with this orientation, the orientation angles must be $\alpha = \pi/2$, $\beta = 0$, $\gamma = \pi$.

VII. Kinematics Testing

To verify the validity of Direct Kinematics, the Niryo One simulator is used. The coordinates are estimated using two approaches. First, a pose with coordinates x, y, z and orientations α, β, γ is sent to the *api* function MOVE_POSE. The function GET_JOINTS is used to obtain the corresponding joints, which are used in the developed Direct Kinematics to estimate the pose. Second, instead of using the joints obtained with the GET_JOINTS function as input for the Direct Kinematics, joints estimated by Inverse Kinematics are used.

The values of the coordinates given as input to the MOVE_POSE function are considered "correct". The "estimated values" are the outputs of the final step in the previous procedures. The RMSE (Root Mean Squared Error) is calculated

using the 24 poses that compose the robot's trajectory. Table V presents the results for both procedures.

TABLE V: Root Mean Squared Error of the Coordinates

	x	y	z
Coordinates w/ Direct Kinematics ($\times 10^{-5}$)	6.12	6.45	5.00
Coordinates w/ Direct Kinematics using joints from Inverse Kinematics ($\times 10^{-5}$)	2.04	2.04	0

To evaluate the Inverse Kinematics, a similar procedure is employed. First, a pose with coordinates x, y, z and orientations α, β, γ is sent to the *api* function MOVE_POSE. The function GET_JOINTS is used to obtained the corresponding joints to the given pose. The Inverse Kinematics are used to calculate the joints corresponding to the initially defined pose. Second, instead of using as input for the Inverse Kinematics the original pose, a pose estimated by Direct Kinematics, whose input is the joints obtained from the GET_JOINTS function.

The values of the joints given by the GET_JOINTS function are considered "correct". The "estimated values" are the outputs of the final step in the previous procedures. The RMSE (Root Mean Squared Error) is calculated using the 24 poses that compose the robot's trajectory. Table VI presents the results for both procedures.

TABLE VI: Root Mean Squared Error of the Joints

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Joints w/ Inverse Kinematics ($\times 10^{-2}$)	0.24	0.02	0.07	9.20	0.09	9.42
Joints w/ Inverse Kinematics using coordinates from Direct Kinematics ($\times 10^{-2}$)	0.24	0.02	0.06	9.20	0.05	9.44

In both cases, the errors are, in general, small. For the position, the errors are inferior to the tenth of a millimeter, and in the same order of magnitude of that of the functions of the Niryo One *api*. For the joint angles, the errors are, in general, in the order of a tenth of a degree or smaller. For joints 4 and 6 the errors are bigger, since in the proposed inverse kinematics forces θ_4 to 0. As joints 4 and 6 are mostly related to the grippers orientation, the effect of setting θ_4 to 0 is compensated by θ_6. In general, the small errors validate the developed kinematics.

VIII. Analysis of the Effect of Uncertainties

During the studies in the lab, it became clear that the real robot manipulator has some uncertainties associated. As the function MOVE_POSE is executed with the same arguments several times, the gripper ends in a slightly different position on each run. This difference is in the order of the millimeters and is mainly explained by the backlash in the joint effector motors.

To model this effect, the desired position of the gripper is chosen and the Inverse Kinematics are used to calculate the corresponding joints. Noise is introduced in the joint values by applying a normal distribution, where the average value of each joint is the calculated joint value and the standard deviation is 0.01. This value was chosen such that the positions differ in some millimeters, as observed in the lab. Finally, the MOVE_JOINTS function is executed using the noisy joints.

To better visualise the effect of the added noise in the tilling task, a simple simulation is developed, in PYTHON. There, it is possible to see the initial and end positions of each block, as well as the orientation and position of the gripper throughout the task. When noise is added, it is possible to observe that the final positions of the blocks is off by some millimeters. The trajectory chosen for each block (see figure 4) aims to mitigate this effect: when being placed, a block can slightly push its neighbouring blocks, so that the relative position of each block remains the desired one.

Finally, table VIII depicts a study on how each joint uncertainty may affect the final pose. It is possible to conclude that, for small changes in the angles, the effect of joints 4 and 6 are similar and mostly affect the orientation. Joint 1 is the joint that mostly affects the Y position of the gripper, while joints 2 and 3 mostly affect the X and Z positions. Joint 5 also affects mostly the Z position.

TABLE VII: Initial Values

Joints	(X, Y, Z) (m)	(α, β, γ) (rad)
(0,0,0,0,0,0)	(0.2452, 0, 0.4175)	(1.5708, 0, 1.5708)

TABLE VIII: Uncertainty Table (L = 0.5 degrees)

Joints	(ΔX, ΔY, ΔZ) (m)	(Δ α, Δ β, Δ γ) (rad)
(L,0,0,0,0,0)	(0, 0.0021, 0)	(0.0087, 0, 0)
(-L,0,0,0,0,0)	(0, -0.0021, 0)	(-0.0087, 0, 0)
(0,L,0,0,0,0)	(-0.0021, 0, 0.0021)	(0, 0, -0.0087)
(0,-L,0,0,0,0)	(0.0021, 0, -0.0021)	(0, 0, 0.0087)
(0,0,L,0,0,0)	(-0.0002, 0, 0.0021)	(0, 0, -0.0087)
(0,0,-L,0,0,0)	(0.0002, 0, -0.0021)	(0, 0, 0.0087)
(0,0,0,L,0,0)	(0, 0, 0)	(0, -0.0087, 0)
(0,0,0,-L,0,0)	(0, 0, 0)	(0, 0,0087, 0)
(0,0,0,0,L,0)	(0, 0, 0.0002)	(0, 0, -0.0087)
(0,0,0,0,-L,0)	(0, 0, -0.0002)	(0, 0, 0.0087)
(0,0,0,0,0,L)	(0, 0, 0)	(0, -0.0087, 0)
(0,0,0,0,0,-L)	(0, 0, 0)	(0, 0.0087, 0)

IX. Scalability and Future Work

The developed work can be applied to a real tilling task, as it was confirmed in the laboratory, with the real robot. In the lab, the uncertainties were tackled by a trial and error approach. Small changes in the desired positions were applied and the function MOVE_POSE was used to move the robot to said position, until an accurate placement of blocks was consistently achieved. A more systematic approach would be to have a sensor, external to the robot, that would give the accurate position of the robot's end effector.

Therefore, before applying the proposed solution to a real problem, these questions would have to be taken into account, specially if precision is key. In the absence of a need for millimetric precision, and if blocks could push each other, as described in section "Analysis of the Effect of Uncertainties", the developed code could be applied as is.

Given the time and robot availability constraints, there were some questions that can still be studied further. Namely, in the lab, when calibrating the robot and, immediately after, setting all joints to zero, the values of the joints were not exactly zero. This would be a way to model the magnitude of the uncertainties for each joint. Also, having different uncertainties for each joint (that is, different values of standard deviation per joint), would allow for a more accurate tackling of the uncertainties.