Mobile Manipulator Control

António Nunes Henriques
antoniohenriques0000@gmail.com

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

June 2017

Abstract

The present thesis describes the progresses towards the development of a mobile 3D printer's prototype, whose goal is to be able to print with unlimited area. The printer is composed of a robotic arm, mounted on top of the platform of a differential drive robot. Posteriorly, the head of the 3D printer will be installed on the tip of the arm. As in any other 3D printing, after the creation of a virtual model of the desired part in CAD software a unique trajectory will be generated for that specific object, taking the printer's specifications into consideration. If followed with enough accuracy and precision by the system's end-effector, it will allow the head of the printer to deposit the numerous layers of material as pretended, leading to the creation of the desired item. The main focus of this dissertation is the tracking of this trajectory. It is intended that this process develops with extreme precision, so as to ensure the created objects are as close to their virtual models as possible. Several studies conducted in diverse areas such as Robotics, Control and Computer Vision will be presented, and whose goal when put together is to make the robot move in the desired way. In the end, the results of the developed algorithm's implementation will be presented.
Keywords: 3D printing, Control, Mobile Manipulator, Computer Vision, Nonholonomic Systems

1. Introduction

Considered by many to be one of the backbones of a 4^{th} Industrial Revolution ([9], [12] to quote a few), 3D printing is a rapidly growing technology that is becoming more commonplace by the day. Despite its recent emergence around the world, there are still considerable limitations associated with this technology. Arguably one of the most significant ones is that the printing area of a regular 3D printer is somewhat limited, which restricts the size of the components to be built. For the time being, there are already solutions to deal with this problem, but they mostly involve building massive 3D printers, capable of conceiving components of substantial size. Obviously this is an expensive solution and one with completely prohibitive prices to the majority of people. When it comes to having an affordable and accessible way to build a 3D printer with unlimited printing area, there are still no feasible solutions available.

1.1. State of the Art

In the beginning of this section, the results of an extensive research aiming to find existing solutions to the small-scale mobile 3D printer are presented. In a later stage, a brief description of the structure of the studied robot is presented.

As expected, it was challenging to find existing work on this particular field of study. However, a very similar robot to the studied one was developed by industrial engineers working at *NEXT* (Núcleo de Experimentacão Tridimensional) and *LIFE* (Laboratório de Interfaces Físicas Experimentais), both academic laboratories belonging to PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro). Although it was not possible to find very specific details regarding their project, one major difference stands out when comparing both robots: theirs uses omni-directional wheels, while the robot considered in this thesis is composed of a robotic arm mounted on top of a differential-drive platform. Information and videos regarding this project can be found in [7] and [5].

The mobile manipulator can be divided into two different parts: the mobile platform and the robotic arm.

1.1.1. Mobile Manipulator

The platform (also know as *Rasteirinho*) is a differential-drive robot. This means that it is composed of two wheels, whose wheel axis is the same. Due to the way that they are attached to the platform, it is not possible for the wheels to turn. Consequently, if both wheels have the same velocity in the same direction a purely linear movement will

occur. In contrast, considering the case where both wheels have the same velocity but in opposite directions, a purely rotational movement will occur. Naturally, if the wheels have different velocities a combination of linear and rotational movements will take place. Each of the wheels is connected to a *EMG30* motor and both the motors are controlled by a *MD-25* board.

1.1.2. Robotic Arm

The robotic arm is composed of a revolute joint, followed by two prismatic ones. A system similar to the worm and wheel was set up in the mobile platform, but with a plastic gear and a threaded rod with equivalent pitch instead of the custom made metallic set. A *EMG30* motor was again chosen to be the actuator of this joint, and was attached to the rod. This whole arrangement forms the revolute joint of the system. Both of the ensuing prismatic joints are assembled using the same system: a spindle with one *EMG30* motor at one end and a bearing at the other. Similarly to the mobile platform, all the motors actuating these joints are controlled by *MD-25* boards. In this case, one board controls the revolute joint and the other controls the prismatic ones.

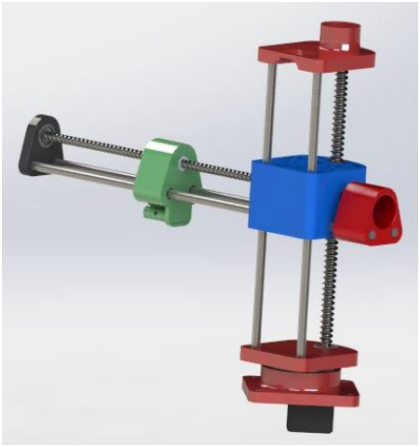

Figure 1: Robotic Arm

1.1.3. Experimental Setup

All the computational work necessary to ensure that the end-effector follows the desired trajectory (computation of the trajectory, control, among others) will be executed in a computer. The computer will communicate via *USB* with an *Arduino* board, fixed to the platform of the *Rasteirinho*. The role of this board is simply to make a connection between all of the *MD-25* boards attached to the motors and the computer. The board receives the inputs to provide to the motors and then reads the sensors' values (encoders) that will be sent to the computer to generate new inputs for the motors. In a later stage of the project, the encoders corresponding to the platform's wheels will not be used. Two cameras will be installed in the mobile platform, and will communicate via *USB* directly with the computer to provide the platform's position feedback signals. The whole system will be powered by a transformer, that provides current directly to one of the *MD-25* boards. In turn, this board is connected to a breadboard that will provide power to all the other boards (including the *Arduino*).

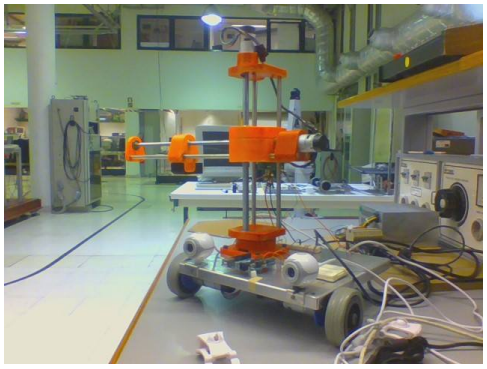

Figure 2: Final Assembly

2. Background

2.1. *Denavit-Hartenberg parameters*

The first item to consider is the definition of the robot's operational (ζ) and generalized coordinates (q):

$$\begin{bmatrix} x_q \\ y_q \\ \theta_q \\ rot_q \\ z_q \\ r_q \end{bmatrix} = q \quad \begin{bmatrix} x \\ y \\ z \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \zeta \tag{1}$$

x, y and z characterize the end-effector's position, whereas x_q and y_q represent the position of the central point of the wheel axis of the mobile platform and z_q represents the position of the robot's vertical joint. The last three parameters of the operational coordinates are defined as a ZYX set of *Euler* angles, θ_q is the angle made by the platform regarding the initial reference frame and rot_q and r_q are the coordinates associated with the robot's revolute and horizontal prismatic joints, respectively.

An initial fixed reference frame will be considered, with its Z axis pointing upwards. Its X axis is perpendicular to the wheel axis, pointing in the driving direction of the robot. The Y axis is parallel to the ground plane, but parallel to the wheel axis (perpendicular to the X axis). It is important to

note that these axis are fixed, even if the platform's position changes.
The parameters will be chosen so as to meet the following requirement: The frame corresponding to the beginning of the manipulator (located at its base) will have its Y axis parallel to the axis that passes through the wheel centres, parallel to the axis of the horizontal prismatic joint in its initial position (as represented in figure 3). This will allow us to posteriorly define the parameters of the manipulator with greater ease. It is important to point out that number of lines in the estimated *Denavit-Hartenberg* parameters is superior to the number of links. Had the conventional *Denavit-Hartenberg* approach been taken, this would not happen. However, and taking into account the fact that we are dealing with a mobile manipulator, the choice of including the additional links was made so as to facilitate the definition of the platform's final reference frame.

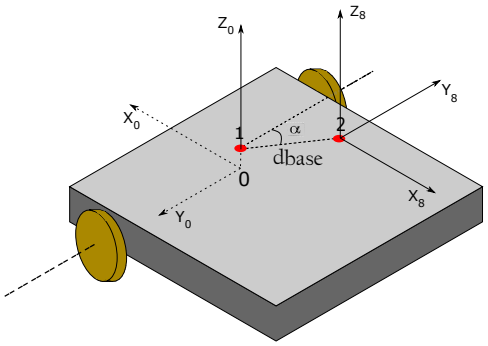

Figure 3: Base of the Manipulator Location

As mentioned, the manipulator consists of one revolute joint and two prismatic ones. Adopting the convention defined in [10], two cubes will be used to represent the prismatic joints and a cylinder will be used to represent the revolute one. A diagram of the robotic arm is presented:

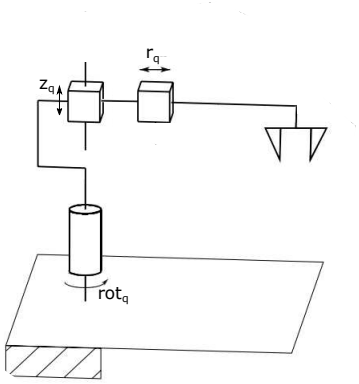

Figure 4: Diagram of the Robotic Arm

The obtained parameters are the following:

Table 1: *Denavit-Hartenberg* parameters - Mobile Manipulator

Link	d_i	θ_i	a_i	α_i
1	0	0	0	$-\frac{\pi}{2}$
2	y_q	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}$
3	x_q	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}$
4	0	$-\frac{\pi}{2}$	0	0
5	$hplat$	0	0	0
6	0	$\theta_q - \alpha$	0	$\frac{\pi}{2}$
7	$dbase$	0	0	$-\frac{\pi}{2}$
8	0	$-\pi + \alpha$	0	0
9	0	rot_q	0	0
10	z_q	0	0	$-\frac{\pi}{2}$
11	r_q	0	0	$-\frac{\pi}{2}$
12	ee	0	0	0

Where $hplat$ is the height of the mobile platform and α, $dbase$ define the angle and distance between the centre of the wheels' axis and the base of the manipulator, respectively.

2.2. Direct Kinematics

The Direct Kinematics process consists in the computation of the robot's end-effector operational coordinates (ζ) from its matching generalized coordinates (q). The direct kinematics of the robot can be obtained through the multiplication of the several transformation matrices corresponding to each of the lines forming the *Denavit-Hartenberg* parameters table, as expressed by the following equation:

$$T_n^0(q) = A_1^0(q_1)A_2^1(q_2)...A_n^{n-1}(q_n) \tag{2}$$

Each of this matrices ($A_i^{i-1}(q_i)$) can be defined as:

$$\begin{bmatrix} cos(\theta_i) & -sin(\theta_i) * cos(\alpha_i) & sin(\theta_i) * sin(\alpha_i) & a_i * cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i) * cos(\alpha_i) & -cos(\theta_i) * sin(\alpha_i) & a_i * sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Alternatively, the direct kinematic equations can also be determined through an analysis of the robot's configuration. Using this method, the following equations are obtained:

$$\begin{aligned} \zeta_1 &= x_q + dbase * cos(-\alpha - \frac{\pi}{2} + \theta_q) + r_q * cos(-\frac{\pi}{2} + \theta_q + rot_q) \\ \zeta_2 &= y_q + dbase * sin(-\alpha - \frac{\pi}{2} + \theta_q) + r_q * sin(-\frac{\pi}{2} + \theta_q + rot_q) \\ \zeta_3 &= z_q + hplat + ee \\ \zeta_4 &= -\pi + \theta_q + rot_q \\ \zeta_5 &= 0 \\ \zeta_6 &= \pi \end{aligned} \tag{4}$$

In order to validate the above shown equations, several sets of operational coordinates were computed using different sets of joint coordinates. As expected, the obtained results were exactly the same despite the used method, thus confirming the validity of the model.

2.3. Non-holonomic constraint and Jacobian

Differentiating the equations obtained in 4, the Jacobian of the system can be calculated:

$$J_{ij} = \frac{\delta f_i}{\delta q_j} \tag{5}$$

However, analysing the mobile platform's configuration we can verify that there is a restriction that keeps it from moving in the direction of the axis that passes through both wheel centres (in the Y axis direction). It is named the *"rolling without slipping condition"*. It is a non-holonomic constraint and as such doesn't imply loss of accessibility in the *Rasteirinho's* configuration space. This restriction can be represented by the following equation:

$$\dot{x_q} * sin(\theta_q) - \dot{y_q} * cos(\theta_q) = 0 \tag{6}$$

Having this limitation in mind, it becomes important to define the platform's velocity in terms of linear and angular velocity (η), instead of $\dot{x_q}$, $\dot{y_q}$ and $\dot{\theta_q}$. Based on [1], the following relationship is established:

$$\dot{q} = S(q) * \eta \tag{7}$$

It can also be expressed by:

$$\begin{bmatrix} \dot{x}_q \\ \dot{y}_q \\ \dot{\theta}_q \\ \dot{rot}_q \\ \dot{z}_q \\ \dot{r}_q \end{bmatrix} = \begin{bmatrix} cos(\theta_q) & 0 & 0 & 0 & 0 \\ sin(\theta_q) & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \dot{v}_q \\ \dot{\theta}_q \\ \dot{rot}_q \\ \dot{z}_q \\ \dot{r}_q \end{bmatrix} \tag{8}$$

From this point on, η velocities will be named *quasi-velocities* of the system.

The newly defined $S(q)$ matrix will establish a functional relationship between the generalized coordinates velocities and the *quasi-velocities*. As such, it is also possible to define a Jacobian matrix that relates the *quasi-velocities* with the operational coordinates velocities:

$$J_{qv}(q) = J(q) * S(q) \tag{9}$$

Which allows us to obtain:

$$\dot{\zeta} = J_{qv}(q) * \eta \tag{10}$$

2.4. *Inverse Kinematics*

In the first place, the inverse of the Jacobian Matrix needs to be defined. Being a non square matrix, a direct inversion is not possible. Alternatively, we can define its *Right Pseudoinverse*. This solution locally minimizes the norm of the joint velocities. However, in the neighbourhood of singularities, the solutions obtained by multiplying this matrix by the operational velocities might not be viable (too high). To work around this problem, we add an additional term to the equation. By doing so, the *Damped Least Squares Inverse* is defined [10]:

$$J^{-1} = J^T(JJ^T + k^2 * I)^{-1} \tag{11}$$

This method allows a smoother evolution of the obtained velocities, especially in the neighbourhood of singularities. However, it also increases the error between the desired and the computed velocities. It is therefore extremely important to have in mind these issues when selecting the damping factor k during testing, so we can achieve the correct balance between them. It is important to mention that usually the gain k is not considered constant, but rather a significantly small value that increases in the neighbourhood of singularities. Naturally, had this variable gain been considered the obtained trajectory would be closer to the desired one. However, and taking into account the already satisfactory results obtained with a constant gain, the variable gain approach was not considered.

The chosen Inverse Kinematics method is presented in [10]. It can be defined by the following equation:

$$\dot{q} = J^{-1}(q)(\dot{\zeta}_d + K_P e) + (I - J^{-1}J)\dot{q}_0 \tag{12}$$

Which is the result of approaching the Inverse Kinematics of the studied redundant robot as a Linear Programming problem, with the following objective function and restriction, respectively:

$$g'(\dot{q}) = \frac{1}{2} * (\dot{q} - \dot{q}_0)^T(\dot{q} - \dot{q}_0) \tag{13}$$

$$\dot{q} = J^{-1}(q)\dot{\zeta_r} \tag{14}$$

Due to ease of implementation, the second term of equation 12 is dropped. Defining the velocity error as:

$$\dot{e} = \dot{\zeta}_d - J(q)\dot{q} \tag{15}$$

It is possible to solve 15 and 12 after the second term is dropped for $\dot{q}$. Putting the resulting equations together, the following result is achieved:

$$\dot{e} + K_P e = 0 \tag{16}$$

Which means that the velocity error converges to zero, proving that the above system is asymptotically stable.

2.5. Sensors

2.5.1. Encoders

Initially, all of the robot's joints were controlled by encoders. However, there is considerable lack of precision inherent to the calculation of positions and velocities through odometry. Several sources of imprecisions may occur, such as:

- Error accumulation, especially noticeable in big trajectories
- Wheel slipping is not taken into consideration
- Precision is highly dependant on human made measurements, such as the mechanical gains and the distance between both wheels

The use of encoders in the manipulator's joints is not as problematic as in the wheels, seeing that in this case problems like wheel slipping or lack of adhesion of the wheels to the floor due to unbalanced weight of the robot do not occur. This precision was corroborated by human made measures during the simulations. Obviously, in latter stage of the project this will be changed but for the time being, the use of encoders as a feedback sensor for the joints is acceptable to validate the model.

2.5.2. Mobile Platform Positioning System

Two cameras and two targets are going to be used to estimate the *Rasteirinho's* position. Naturally, the logical progression is to first transform the points defined in Image Plane reference frame coordinates to Camera frame reference coordinates, to posteriorly enable the transformation of these points into world reference frame points. The scenario was manipulated in a way both camera centres and both target centroids were placed in the same plane. In other words, the camera centres and the targets' centroids were placed at approximately the same height. Consequently, only the cameras' XZ plane will be considered (the Y axis is pointing upwards). The centre of camera one (the left one) is considered to be the origin of the camera frame. x_1 is the x coordinate in the image plane (in pixels) of any point P and x_2 is the x coordinate of the same point in camera two.The goal is to find the X and Z coordinates of both target centroids and the first step is to determine Z , as defined in 5. The calculations to do so are based on the work presented in [3]. Consider the following image: Where b is the distance between the two cameras. To calculate the Z coordinate of point P we considerer the equations of both lines represented in Figure 5, as defined in the pinhole model:

$$\begin{aligned} Z &= \frac{f}{x_1} * X, \\ Z &= \frac{f}{x_2} * X - \frac{f}{x_2} * b. \end{aligned} \tag{17}$$

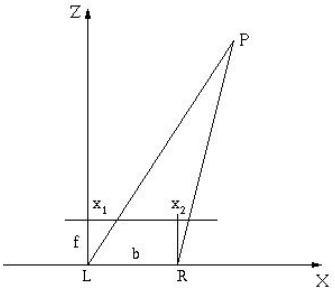

Figure 5: Point in Stereo Cameras [3]

The capital letters X and Z represent the coordinates in the camera frame. Solving both these equations for X and putting them together we end up with:

$$\begin{aligned} &\frac{x_1}{f} * Z = (Z + \frac{f * b}{x_2}) * \frac{x_2}{f} \Leftrightarrow, \\ &\Leftrightarrow Z = \frac{f * b}{x_1 - x_2}. \end{aligned} \tag{18}$$

Having the coordinates of the two target centroids in both cameras, it is possible to calculate the Z coordinate of the first target (Z_1) and of the second (Z_2). Having both of these distances and $\triangle\, x$ (the distance between the two targets), θ_q (the robot's angular position regarding the world coordinates frame) can be calculated from the following equation:

$$\theta_q = \arcsin(\frac{\Delta Z}{\triangle\, x}). \tag{19}$$

The distance of the centre of camera 1 to the first target can also be calculated using the pinhole model formula:

$$X = \frac{x * Z}{f}. \tag{20}$$

After this step, the transformation from Image Plane reference frame coordinates to Camera reference frame coordinates is complete. Henceforth, the computation of the world's coordinates of the platform is relatively straightforward. As previously defined, X and Z are coordinates regarding the camera reference frame (only absolute values are considered). Two auxiliary variables are calculated:

$$h = \sqrt{X^2 + Z^2} \tag{21}$$

$$\gamma = atan\frac{X}{Z} \tag{22}$$

The final x_q and y_q values computation vary according to the signals of θ_q and X. They can be calculated the following way:

$$\phi = \theta_q \pm \gamma \tag{23}$$

$$x_q = h * cos(\phi) \quad (24)$$

$$y_q = \pm h * sin(\phi) \quad (25)$$

ϕ is an auxiliary variable that represents the angle between point P and the camera's Z axis represented in 5. One more detail is worth mentioning: the calculated position represents the world coordinates of the left camera. An adjustment to consider the central point between the two wheels can be made through the following equation:

$$(x_q, y_q) = (x_q, y_q) + (\frac{b}{2} * cos(\theta), \frac{b}{2} * sin(\theta)). \quad (26)$$

2.6. Control Algorithms

Due to the nonholonomic nature of the robot, two different control methods will be applied: one for the mobile platform's wheels and another one to the remaining joints.

2.6.1. Non-linear Model Based Predictive Control (Robot's Joints Control)

The control method presented in this section was developed by Adel Merabet and Jason Gu, and can be consulted in [6]. In order to estimate the robot's dynamic model the *Newton-Euler* method was applied. This recursive algorithm was chosen due to its computational efficiency. It is based on a balance of the actuating forces in each of the robot's links and it can be divided in two stages. The first consists of a forward recursion from the first joint to the end-effector, that calculates all the velocities and accelerations corresponding to each of the joints. In a second phase, a backward recursion takes place (from the end-effector to the first joint). This recursion determines all the forces and torques the joints are subjected to. Disregarding the effects of the viscous friction, the *Coulomb* friction and the forces and torques applied by the end-effector in the surrounding environment, we can simplify the dynamic of the system:

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (27)$$

However, there are several sources of uncertainties that can have an influence on the calculation of the various parameters in this equation. Of these uncertainties, the following stand out: modelling errors, unknown forces applied to the robot, computation errors or the disregard of several terms during the application of the *Newton-Euler* method. Having this limitation in mind, the objective is to define a control law $u(t)$ to be applied in real time. It is a predictive control method, which means that the control law $u(t)$ designed to follow a reference trajectory Y_d is calculated based on the predicted output of the system in the ensuing time step: $Y(t+\tau)$. The term τ designates the sample time. This is made through the minimization of a cost function, based on the quadratic form of the predicted error in $t + \tau$:

$$\kappa = \frac{1}{2}\int_0^{\tau_r}(Y(t+\tau) - Y_d(t+\tau))^T(Y(t+\tau) - Y_d(t+\tau))d\tau \quad (28)$$

Where

$$e_Y(t+\tau) = Y(t+\tau) - Y_d(t+\tau) = T(\tau)(Y(t) - Y_d(t)) \quad (29)$$

and

$$T(\tau) = \begin{pmatrix} I_{N_j \times N_j} & \tau * I_{N_j \times N_j} & \frac{\tau^2}{2} * I_{N_j \times N_j} \end{pmatrix} \quad (30)$$

After differentiated and matched to zero (along with a few other changes), the final control law is obtained:

$$\begin{aligned} u(t) = &-B_0(X_1)\{K_1(Y - Y_d) + K_2(\dot{Y} - \dot{Y}_d) \\ &- B_0^{-1}(C(X_1, X_2)X_2 + G(X_1)) - \ddot{Y}_d\} - \delta_{est}(t) \end{aligned} \quad (31)$$

The uncertainties can be estimated by:

$$\delta_{est}(t) = L\left(\dot{e}_Y(t) + K_2 e_Y(t) + K_1 \int e_Y(t)dt\right) \quad (32)$$

2.6.2. Nonholonomic Mobile Platform Control

In this section, the control law [8] applied to the mobile platform is presented, an alternative approach to the method presented in [4]. It is important to emphasize that this is the method chosen to control the platform only, and that the previously presented method (Non-linear Model Based Predictive Control) is used to control the manipulator's joints.

In order to define the tracking errors used to execute the control law, a new local coordinate system is defined. The Z axis will remain the same, but two new axis will be defined in the original XY plane: The D axis, parallel to the driving direction and the L axis, perpendicular to the driving direction. The new coordinate system is illustrated in the following image:

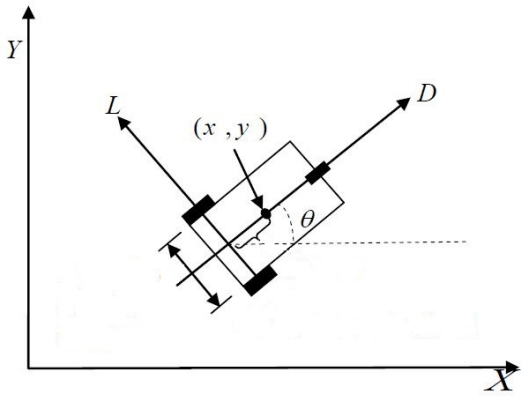

Figure 6: Driving Platform Coordinate System [8]

This transformation allows the definition of the position errors as the following:

$$e = \begin{bmatrix} e_D \\ e_L \\ e_{\theta_q} \end{bmatrix} = \begin{bmatrix} cos(\theta_q) & sin(\theta_q) & 0 \\ -sin(\theta_q) & cos(\theta_q) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{qd} - x_q \\ y_{qd} - y_q \\ \theta_{q_d} - \theta_q \end{bmatrix} \quad (33)$$

Where e_D represents the error in the driving direction and e_L the error in the lateral direction. The velocity errors can be defined as:

$$\dot{e} = \begin{bmatrix} \dot{e_D} \\ \dot{e_L} \\ \dot{e_{\theta q}} \end{bmatrix} = \begin{bmatrix} \dot{\theta_q} * e_L - v + v_d * cos(e_{\theta q}) \\ -\dot{\theta_q} * e_D + v_d * sin(e_{\theta q}) \\ \dot{\theta_{q_d}} - \dot{\theta_q} \end{bmatrix} \quad (34)$$

The proposed solution is a tracking controller, represented by the following equations:

$$\begin{bmatrix} v \\ \dot{\theta_q} \end{bmatrix} = \begin{bmatrix} v_d * cos(e_{\theta q}) + k_1 * e_D + k_4 * sign(e_D) * e_L^2 \\ \dot{\theta_{q_d}} + v_d * (k_2 * e_L + k_3 * sin(e_{\theta q})) \end{bmatrix} \quad (35)$$

With k_1, k_2, k_3 and k_4 being positive constants and

$$sign(e_D) = \begin{cases} -1, & e_D < 0 \\ 1, & e_D \geq 0 \end{cases} \quad (36)$$

These velocity values can be converted into β values to provide as input to the *Rasteirinho* as detailed in [11].

3. Implementation

A few aspects are worth mentioning regarding the implementation of the previously studied model in the real robot.

3.1. *Md-25* Boards and *EMG30* Motors

As explained before, the five motors incorporated in the robot are all driven by *MD-25* boards. These drivers already come with integrated speed control, achieved through the encoders' feedback of each motor. Consequently, when a certain set of velocities is intended, instead of directly controlling the matching voltages a 1 byte value that corresponds to the desired velocity should be provided to the board (out of 256 possible values). The selected manipulator control law (used only to manipulate the robotic arm) calculates the torques to apply to the various joints in each time step. Therefore, it becomes necessary to find the motor velocities that correspond to each of those torques, so that later the corresponding input byte can be computed. The calculation of this input to provide to the *Md-25* board is relatively straightforward in the mobile platform case, because the control law already provides the desired velocities, instead of torques.
Ignoring the influence of inductance in the motors' dynamics in an initial stage, the functional relationship between joint torques and the corresponding motor voltages can be defined as :

$$v = R_a * K_s^{-1} * K_r^{-1} * \tau + K_s * K_r * \dot{q} \quad (37)$$

Where R_a is the matrix containing the resistances of the motors. For ease of notation, K_s is the matrix that represents both the torque and the voltage constants, that are considered to be the same. K_r is defined as the matrix that relates the angular velocities of each motor with their respective joint velocities:

$$K_{r_i} \dot{q_i} = \dot{\vartheta}_{m_i} \quad (38)$$

Assuming a steady state regime, the function describing the relation between the voltage of a motor and its corresponding angular velocity is:

$$\dot{\vartheta}_m = \frac{K_s}{K_s^2 + R_a * B} + \frac{R * T_c}{K_s^2 + R_a * B} \quad (39)$$

The parameters required for this equation are estimated in [2]. After obtaining the desired motor angular velocity, its conversion to a β value to provide to the $MD - 25$ driver is necessary. Again, these transformations are only required when the robotic arm is considered, since the mobile platform control already provides linear and angular velocity inputs, that can be used directly to calculate the desired β inputs. This above mentioned conversion is described by the equation:

$$\beta = \frac{\dot{\vartheta}}{0,02546479 * 2 * \pi} \quad (40)$$

The obtained result is then rounded to the nearest integer, because the value sent to the board should be a whole number ranging from 0 to 255.

3.2. Trajectory

Considering that the sensoring system used for feedback is still being developed, it is easier to assess its precision if each of the joints moves individually (one at a time). For this reason, and just for the time being, a trajectory computed through the described Inverse Kinematics method was not used. Instead, a trajectory in the generalized coordinates space was created and applied directly to the robot:

- Holding still for $5s$, followed by a linear movement of the mobile platform ($10cm$ in $10s$)
- Rotation of the revolute joint ($0.25rad$ in $10s$), followed by its return to the original angular position in $10s$
- Rotation of the mobile platform ($0.0.1745rad$ or 10 degrees in $10s$)
- At the same time, both the horizontal and the vertical prismatic joints advance $2cm$ in $10s$
- Return of both prismatic joints to their original positions.
- Hold the final position for 5 seconds, resulting in a total trajectory time of 65 seconds.

All of the defined sub-trajectories (platform, revolute and prismatic joints) have a cubic time evolution of their respective coordinates. A more detailed explanation of the first presented point is in order. As it will be posteriorly approached in the Results section, the computation of the platform's θ_q angle is subjected to considerable fluctuation, which can affect a more precise calculation of both x_q and y_q. Therefore, a *Kalman* filter will be applied to the calculation of this variable. The considered filter computes the filtered values based on the previously obtained unfiltered ones, and so the during the first five seconds no movement is enforced on the platform so as to allow the value of θ_q to stabilize. These filters are also applied to the computation of x_q and y_q, naturally with different covariances.

4. Results

The errors associated with the manipulator's joints are the following:

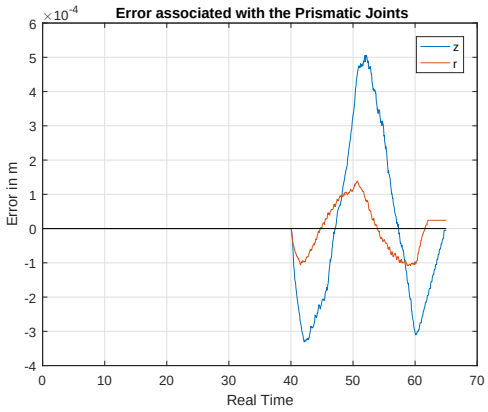

Figure 7: Manipulator's Prismatic Joints Errors

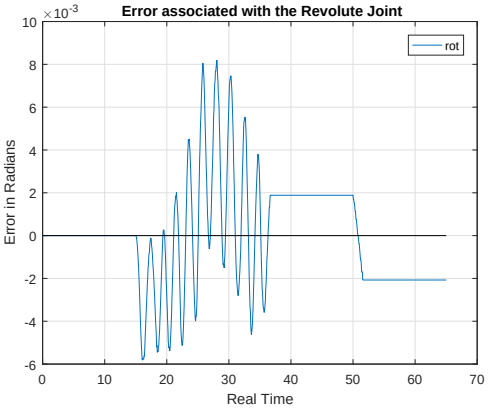

Figure 8: Manipulator's Revolute Joints Errors

The evolution of x_q throughout time is presented, as well as the corresponding error:

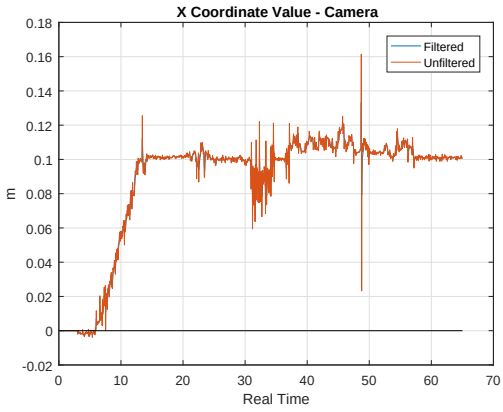

Figure 9: Evolution of x_q with time

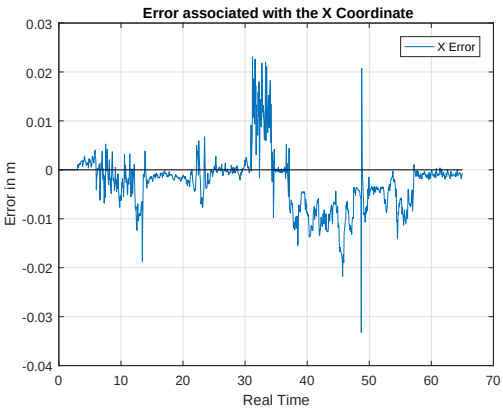

Figure 10: Error associated with x_q

The evolution of y_q throughout time is presented, as well as the corresponding error:

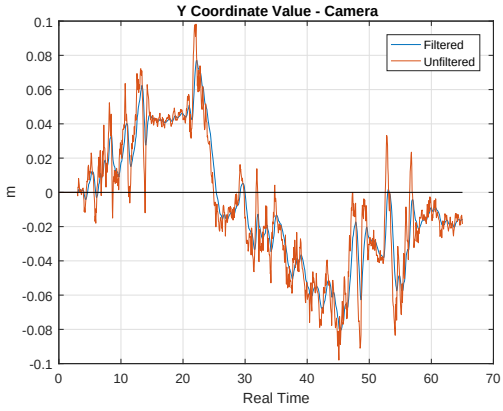

Figure 11: Evolution of y_q with time

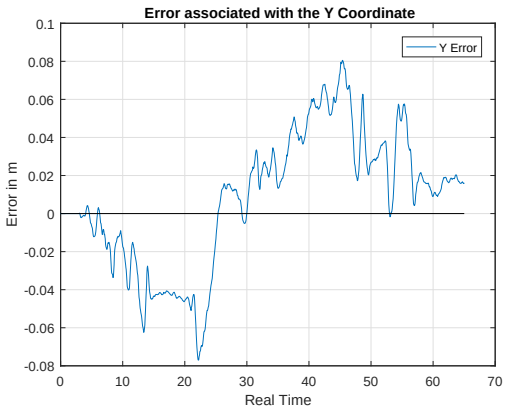

Figure 12: Error associated with y_q

Finally, the evolution of θ_q throughout time is presented, as well as the corresponding error:

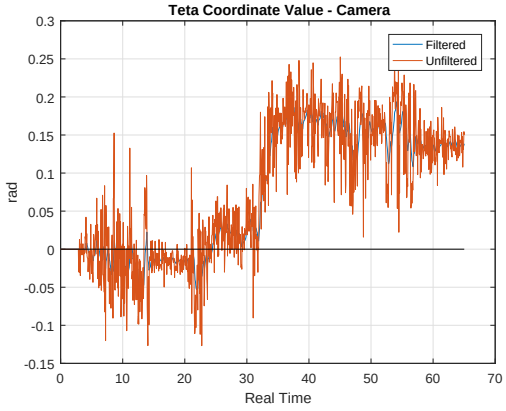

Figure 13: Evolution of θ_q with time

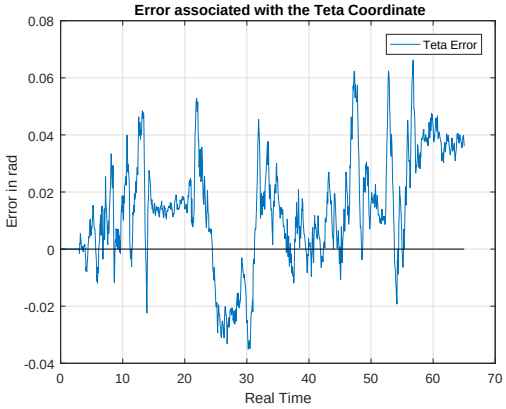

Figure 14: Error associated with θ_q

4.1. Discussion

The errors associated with the three manipulator joints were highly satisfactory. However, there a few remarks that can be made regarding the platform's results. First of all, the fact that the provided cameras' angle of view is fairly limited has to be taken into consideration. Given that the goal is to be able to print with an unlimited area and that the camera has to always be able to detect both targets, the following scenario adjustments had to be imposed:

- An effort was made so that the targets would remain as close as possible ($60mm$)
- The platform had to be as far as possible from the targets in the initial time step

Although this adjustments allowed a significant widening of the working area (again, the cameras always have to see both targets), one problem arose. During the computing of the centroids' targets, there are unavoidable fluctuations in the obtained pixel. Even if these fluctuations are not that significant (one or two pixels for example), allied with the cameras' lack of resolution they can cause the obtained angle value to oscillate tremendously (see the unfiltered results of the obtained angle in figure 13). It is important to emphasize that the fact that these fluctuations occur despite the correct calculation of the angle, as the oscillations always take place around their correct value (this was confirmed by human-made measures). Consequently, this issue can lead to oscillations in both the x_q and y_q values, due to the coordinate transformation that occurs when we pass from camera coordinates to world coordinates. Naturally, it affects the y_q coordinate more noticeably, given the fact that the initial distance from the camera to the first target is considerably bigger in the x_q coordinate than in y_q. This comparison can be made through the analysis of figures 9 and 11. It is evident that the fluctuations in both x_q and y_q occur proportionally to the fluctuations of θ_q. This theory is strongly corroborated by the fact that when the angle stabilizes at its final position, x_q stabilizes at $0.1m$ and y_q begins to stabilize at values close to 0. Despite this issue, the obtained results were quite positive.

5. Conclusions

5.1. Achievements

In this dissertation, significant progresses regarding the mobile 3D printer's project were made. A Direct kinematics model for the robot was developed, as well as an Inverse Kinematics one. A dynamic model was proposed for both the motors and the mobile manipulator. Different control laws were studied and applied to the manipulator and the platform, taking into consideration its nonholonomic constraint. A positioning system using stereo cameras was also developed, despite the previously discussed limitation. The model resulting from putting all of these points together lead to a satisfactory tracking of a desired predefined trajectory, presented in the previous section. All of this was

made taking into consideration the computational efficiency of the algorithm.

5.2. Future Work

As discussed, this is an ongoing project with still a considerable way to go before any actual 3D printing becomes a reality. The following suggestions for possible future work are made:

- Replacing the cameras with ones with a wider angle of view and resolution. This would significantly soften the angle fluctuation discussed in the Results section and would imply almost no changes in the developed code.Other strategies aiming to tackle this issue could also be relevant.
- Changing the positioning of the cameras so they could be used as feedback sensor for all the joints, not just the platform. Naturally, it would be important to deal with first point before moving on to this one.
- Control implementation without the *MD-25* boards, and the limitations that the use of these boards implies.
- Development of trajectory planning software
- Installation of a printing head

Acknowledgements

In this section, I would like to acknowledge all my lab mates, for all the support they have given me throughout the writing of this dissertation. In particular, I would like to thank my friends Francisco Fernandes, Tobias Pereira, Miguel Roque and Guilherme Leite for their endless patience towards me and their unwavering willingness to help whenever needed. Moreover, I would like express my sincere gratitude to all the people at *Instituto Superior Tcnico* who were a part of my academic journey. Namely, I would like to thank my supervisors Prof. Paulo Oliveira and Prof. Carlos Cardeira for setting such high standards and constantly pushing me to do better. Finally, I would like to thank my family and friends for their unconditional support, and for keeping sane throughout the writing of this dissertation.

References

[1] B. Bayle, M. Renaud, and J. Y. Fourquet. Nonholonomic mobile manipulators: Kinematics, velocities and redundancies. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 36(1):45–63, 2003.

[2] J. Gonçalves, J. Lima, P. J. Costa, and A. P. Moreira. Modeling and simulation of the EMG30 geared motor with encoder resorting to simtwo: The official robot@factory simulator. *Lecture Notes in Mechanical Engineering*, 7:307–314, 2013.

[3] L. Iocchi. Stereo Vision: Triangulation, 1998.

[4] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. *Proceedings., IEEE International Conference on Robotics and Automation*, 30(5):384–389, 1990.

[5] A. Liszewski. A Robotic 3D Printer Could Print Anything, Anywhere It Wants. *Gizmodo*, 2014.

[6] A. Merabet and J. Gu. World ' s largest Science , Technology & Medicine Open Access book publisher The Oil Palm Wastes in Malaysia. In *Robot Manipulators New Achievements*, chapter Advanced N, pages 107–128. InTech, 2010.

[7] L. Morris. A robotic 3D printer could print anything, anywhere it wants. *Electronic Engineering Journal*, 2014.

[8] I. M. H. Sanhoury, S. H. M. Amin, and A. R. Husain. Tracking Control of a Nonholonomic Wheeled Mobile Robot. 1(April):7–11, 2012.

[9] K. Schwab. The Fourth Industrial Revolution: what it means, how to respond. *World Economic Forum*, 2016.

[10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics - Modelling, Planning and Control*. Springer, 2009.

[11] D. Valério and C. Cardeira. Labguide2015 - Control Systems Lecture Slides. Technical report, Instituto Superior Técnico, Lisboa, 2015.

[12] Wikipedia. Fourth Industrial Revolution, 2017.