Optimal Control

Integrated Master Degree in Mechanical Engineering

Scientific Area of Control, Automation, and Industrial Informatics

Target Tracking with Pan-Tilt

Author:
Luís Miranda, 81089
Diogo Catarino, 81772
Reon Fujieda, 89120

Instructor:
Prof. Paulo Oliveira

October 27, 2020

Contents

1 Introduction

Our aim is to model and control an acquisition and tracking system of a static target, in an optimal way. This system is built on top of a mobile robot "Rasteirinho" and will track the target using a combination of horizontal and vertical movement (Pan-Tilt). These movements are executed with the help of two servomotor through the acquisition of data from an end-effector camera. The main purpose of the proposed system is to keep the camera's central point fixed on the target.

In a first phase, the systems and sub-systems should be studied and modeled. After the control problem is formulated, the systems characteristics such as stability, controllability and observability will be analyzed and discussed. A Simulink model should be implemented as well. Lastly, the frequency and time responses, both in open-loop and in closed-loop, should be verified.

In a second phase, partial goals of the control system to be developed will be identified and a classical control solution will be proposed, namely based on pole placement, PID and lead-lag compensators. The performance of the solutions proposed, under realistic disturbances, will be verified and matching those in the real system/sensor package.

In a third phase, a functional to be optimized will be proposed. A Linear Quadratic Regulator will be used to control the system in an optimal way. After that, a State Observer will be design in order to every state variables be available. The performance of the solutions proposed, under realistic disturbances, will be verified and matching those in the real system/sensor package.

2 Formulation of the Problem

2.1 System dynamics

To solve the problem in hand the angular position values for the Pan-Tilt system must be obtained from the information obtained by the camera. Given the relative position of the target to the "rasteirinho", the desired angles to keep the target centered are easily taken as it is shown on figure 1.

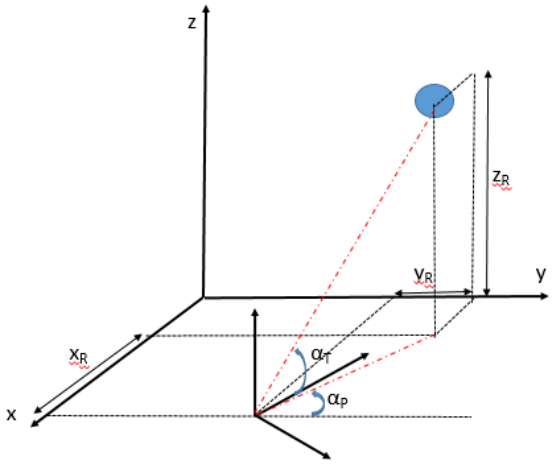


Figure 1: Target and "rasteirinho" coordinates

$$\alpha_{pan} = \arctan(\frac{x_R}{Y_R}) \tag{1}$$

$$\alpha_{tilt} = \arctan(\frac{Z_R}{\sqrt{X_R^2 + Y_R^2}}) \tag{2}$$

Using the camera we can calculate the coordenates needed by measuring, in pixels, the distance from the extremities of a L-sized square to the center of the image. Observing how the camera works in figure 2 the equations for transforming from virtual to real coordenates are (in which f is the focal distance of the camera):

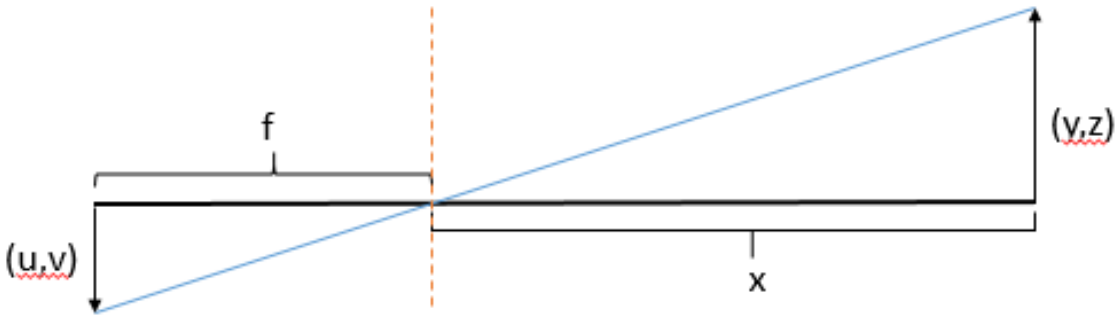


Figure 2: Camera functionality

$$u = f\frac{Y_C}{X_C} \tag{3}$$

$$v = f\frac{Z_C}{X_C} \tag{4}$$

Relating the camera reference to the "rasteirinho" reference according to the transformation seen in the Pan-Tilt Model section, the coordinates (x_C, y_C, z_C) can be calculated from the coordinates (x_R, y_R, z_R) and replacing on equations 3 and 4 the extremities of the target on the image will be given by:

$$\begin{cases} \begin{bmatrix} 0 \\ \frac{L}{2} \end{bmatrix} = \begin{bmatrix} x_R \\ z_R \end{bmatrix} + \begin{bmatrix} c_{\alpha_T} & -s_{\alpha_T} \\ s_{\alpha_T} & c_{\alpha_T} \end{bmatrix} \begin{bmatrix} x_{C_1} \\ z_{C_1} \end{bmatrix} \\ \begin{bmatrix} 0 \\ -\frac{L}{2} \end{bmatrix} = \begin{bmatrix} x_R \\ z_R \end{bmatrix} + \begin{bmatrix} c_{\alpha_T} & -s_{\alpha_T} \\ s_{\alpha_T} & c_{\alpha_T} \end{bmatrix} \begin{bmatrix} x_{C_2} \\ z_{C_2} \end{bmatrix} \end{cases} \tag{5}$$

$$\begin{cases} u_1 = f\frac{x_R s(\alpha_P - \alpha_R) + \frac{L}{2} c(\alpha_P - \alpha_R) - y_R c(\alpha_P - \alpha_R)}{-x_R c(\alpha_P - \alpha_R) + \frac{L}{2} s(\alpha_P - \alpha_R) - y_R s(\alpha_P - \alpha_R)} \\ u_2 = f\frac{x_R s(\alpha_P - \alpha_R) - \frac{L}{2} c(\alpha_P - \alpha_R) - y_R c(\alpha_P - \alpha_R)}{-x_R c(\alpha_P - \alpha_R) - \frac{L}{2} s(\alpha_P - \alpha_R) - y_R s(\alpha_P - \alpha_R)} \end{cases} \tag{6}$$

$$\begin{cases} v_1 = f\frac{x_R s_{\alpha_T} + \frac{L}{2} c_{\alpha_T} - z_R c_{\alpha_T}}{-x_R c_{\alpha_T} + \frac{L}{2} s_{\alpha_T} - z_R s_{\alpha_T}} \\ v_2 = f\frac{x_R s_{\alpha_T} - \frac{L}{2} c_{\alpha_T} - z_R c_{\alpha_T}}{-x_R c_{\alpha_T} - \frac{L}{2} s_{\alpha_T} - z_R s_{\alpha_T}} \end{cases} \tag{7}$$

At last, rearranging the previous equations, (x_R, y_R, z_R) can be calculated based on u1, u2, v1, v2 and the current angular position of the camera (α_{pan}, α_{tilt} and rotation of the "rasteririnho") which are used in equations 1 and 2 to calculate the desired angles of pan and tilt. Note that u_1 and u_2 are obtained in the same way as v_1 and v_2 only replacing α_{pan} by α_{tilt}

$$\begin{cases} x_R = \frac{(-fc_{\alpha_p} - u_2 s_{\alpha_p})(f^2 c_{\alpha_p} s_{\alpha_p} + f(c^2_{\alpha_p}(u_1+u_2) - u_1) - s_{\alpha_p} c_{\alpha_p} u_1 u_2)L}{f(fs_{\alpha_p} + u_2 c_{\alpha_p})(u_1 - u_2)} \\ y_R = \frac{-(f^2 s_{2\alpha_p} + f(2c^2_{\alpha_p} - 1)(u_1+u_2) - s_{2\alpha_p} u_1 u_2)L}{2f(u_1 - u_2)} \\ z_R = \frac{-(f^2 s_{2\alpha_t} + f(2c^2_{\alpha_t} - 1)(v_1+v_2) - s_{2\alpha_t} v_1 v_2)L}{2f(v_1 - v_2)} \end{cases} \tag{8}$$

2.2 Pan-Tilt Model

The camera model has 2 rotation degrees of freedom: rotation around the vertical and horizontal axis, which are called *pan* and *tilt*, correspondingly. The pan axis is collinear with the camera Z-axis, for a zero tilt angle. The tilt axis is collinear with the Y-axis, which is parallel to the ground.

By considering α_{pan} as the pan angle and α_{tilt} as the tilt angle, we can write the tilt transformation matrix as the following,

$$T_{tilt}(\alpha_{tilt}) = Rot_Y(\alpha_{tilt}) = \begin{bmatrix} cos(\alpha_{tilt}) & 0 & sin(\alpha_{tilt}) \\ 0 & 1 & 0 \\ -sin(\alpha_{tilt}) & 0 & cos(\alpha_{tilt}) \end{bmatrix} \tag{9}$$

The pan movement is the rotation about the Z-axis, for zero tilt angle. If the tilt angle is not zero, then the rotation about the vertical axis become composed by three rotations. The first one is the rotation of -α_{tilt} about the Y-axis aligning the Z-axis with the vertical axis. Then, the camera rotates with α_{pan} about the Z-axis. At last, a rotation of α_{tilt} about the Y-axis is made in order to return the camera to its tilt position. The representation of this pan transformation matrix is the following,

$$T_{pan}(\alpha_{pan}) = Rot_Y(\alpha_{tilt}) \cdot Rot_Z(\alpha_{pan}) \cdot Rot_Y(-\alpha_{tilt}) =$$

$$\begin{bmatrix} cos(\alpha_{tilt}) & 0 & sin(\alpha_{tilt}) \\ 0 & 1 & 0 \\ -sin(\alpha_{tilt}) & 0 & cos(\alpha_{tilt}) \end{bmatrix} \begin{bmatrix} cos(\alpha_{pan}) & -sin(\alpha_{pan}) & 0 \\ sin(\alpha_{pan}) & cos(\alpha_{pan}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\alpha_{tilt}) & 0 & -sin(\alpha_{tilt}) \\ 0 & 1 & 0 \\ sin(\alpha_{tilt}) & 0 & cos(\alpha_{tilt}) \end{bmatrix} \tag{10}$$

The camera pose can be described by the pair of angles α_{pan} and α_{tilt} with respect to the initial position. Due to the relative movement, the world seen by the camera moves in the opposite. Therefor, the transformation matrix from the initial camera coordinates to the new ones is $T_{world}(\alpha_{tilt}; \alpha_{pan}) = Rot_Y(-\alpha_{tilt}) \cdot Rot_Z(-\alpha_{pan})$

$$T_{world}(\alpha_{tilt}; \alpha_{pan}) =$$

$$\begin{bmatrix} cos(\alpha_{tilt})cos(\alpha_{pan}) & cos(\alpha_{tilt})sin(\alpha_{pan}) & -sin(\alpha_{tilt}) \\ -sin(\alpha_{pan}) & cos(\alpha_{pan}) & 0 \\ sin(\alpha_{tilt})cos(\alpha_{pan}) & sin(\alpha_{tilt})sin(\alpha_{pan}) & cos(\alpha_{tilt}) \end{bmatrix} \tag{11}$$

2.3 Simulink

From the variables $\alpha_{tilt}, \alpha_{pan}, \alpha_R$ and the spacial coordinates of "rasteirinho" it was possible to create a Simulink file to model the "rasteirinho", as can be seen on figure 3.

- The first MATLAB box, real2image, contains equations 3 and 4: given the five variables above mentioned, it is possible to obtain the coordinates u1, u2, v1 and v2 from the target.
- The second MATLAB box, image2real, contains equation 8: from the image position (u,v) calculated before, it is possible to obtain the "rasteirinho" position (x_R, y_R, z_R) and an update at angles α_{tilt} and α_{pan}

Concerning the servo motors, another Simulink was created, as can be seen on figure 4, whose inputs are the angles calculated above.

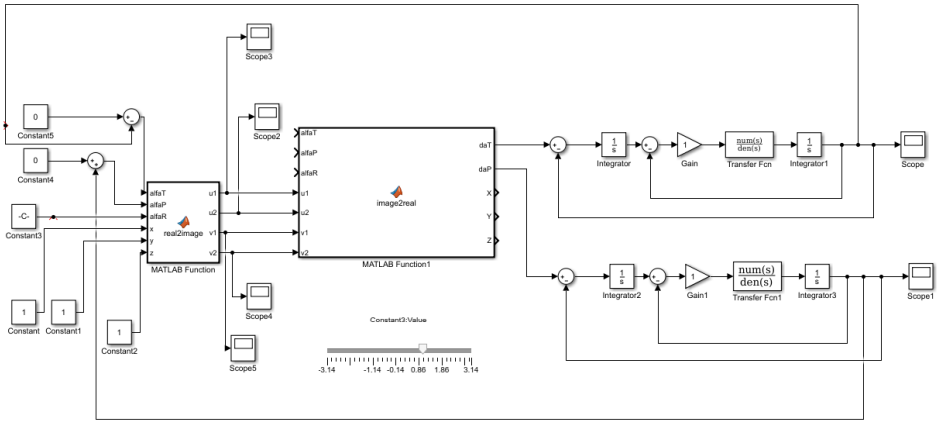


Figure 3: "Rasteirinho" and camera

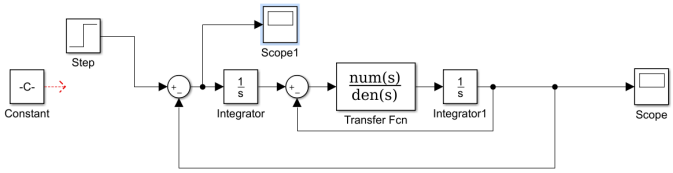


Figure 4: Servo

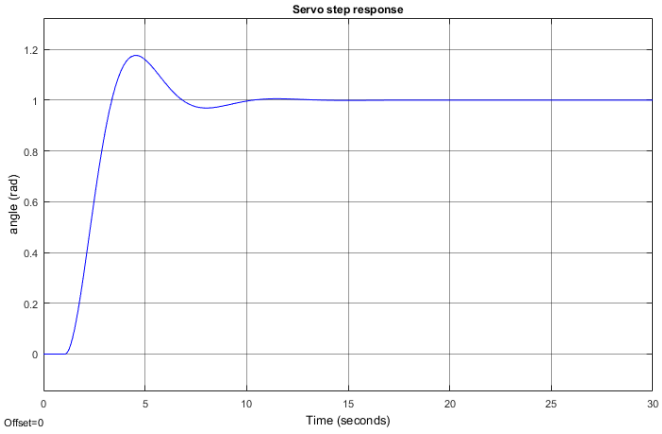


Figure 5: Servo step response

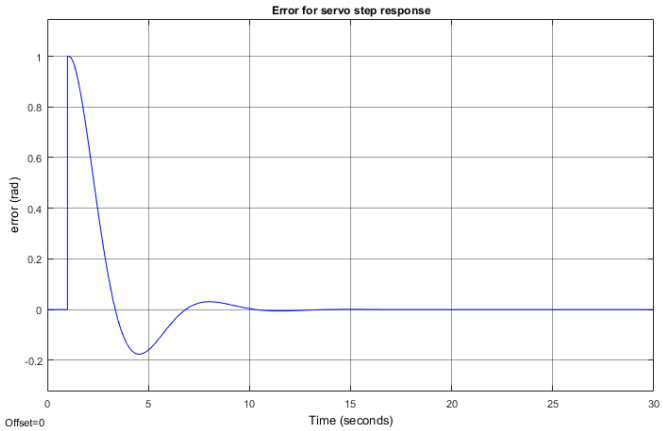


Figure 6: Servo output error in a step response

A step input was simulated, and the servo response and servo response error can be seen on figures 5 and 6.

2.4 System Characteristics

Based on previous papers on this subject, a block diagram for the servo motors was drawn as well as its transfer function.

$$G(s) = \frac{657.92}{s^3 + 40s^2 + 657.92s} \tag{12}$$

Given the transfer function it is possible to represent the system in state space. To do that it is known that:

$$\frac{Y(s)}{G(s)} = \frac{b_0 s^n + b_1 s^{n-1} + b_2 s^{n-2} + ... + b_{n-1}s + b_n}{a_0 s^n + a_1 s^{n-1} + a_2 s^{n-2} + ... + a_{n-1}s + a_n} \tag{13}$$

Calculating the a and b polynomials we can then have a state space representation of the system (in the canonical controllable form):

$$\begin{cases} a_0 = 1 \\ a_1 = 40 \\ a_2 = 657.92 \\ a_3 = 0 \end{cases} \tag{14}$$

$$\begin{cases} b_0 = 0 \\ b_1 = 0 \\ b_2 = 0 \\ b_3 = 657.92 \end{cases} \tag{15}$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & -657.92 & -40\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} u \quad ; y = \begin{bmatrix}657.92 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix} \quad (16)$$

The system can now be checked for controlability and observability by calculating their respective matrices.

$$C_s = \begin{bmatrix}B & \vdots & AB & \vdots & \dots & \vdots & A^{n-1}B\end{bmatrix} = \begin{bmatrix}0 & 0 & 1\\ 0 & 1 & -40\\ 1 & -40 & 942.08\end{bmatrix} \quad (17)$$

$$O_s = \begin{bmatrix} C \\ \cdots \\ CA \\ \cdots \\ \vdots \\ \cdots \\ CA^{n-1}\end{bmatrix} = \begin{bmatrix}657.92 & 0 & 0\\ 0 & 657.92 & 0\\ 0 & 0 & 657.92\end{bmatrix} \quad (18)$$

Both these matrices are full rank (rank = 3) which means that the system is completely controllable and observable.
The systems time and frequency responses in both open loop and closed-loop can be analyzed to gather more information on the system.

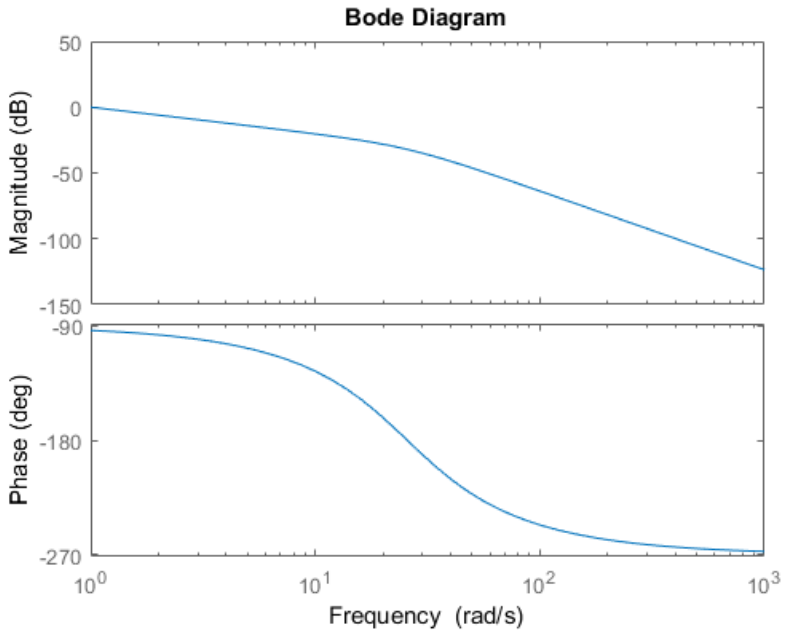


Figure 7: Open loop bode diagram

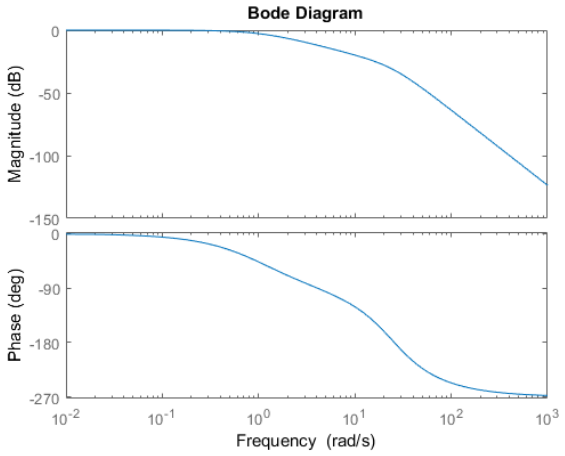


Figure 8: Closed loop bode diagram

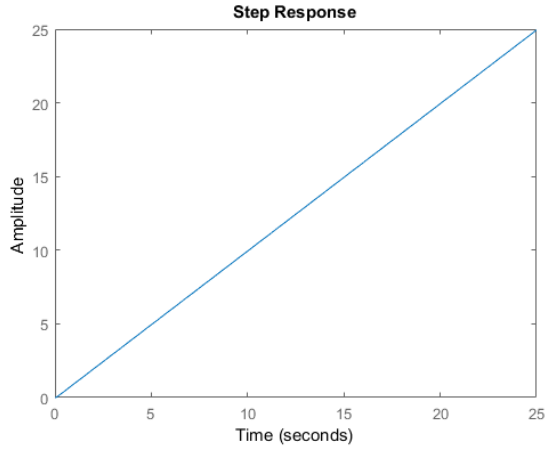


Figure 9: Open loop step response

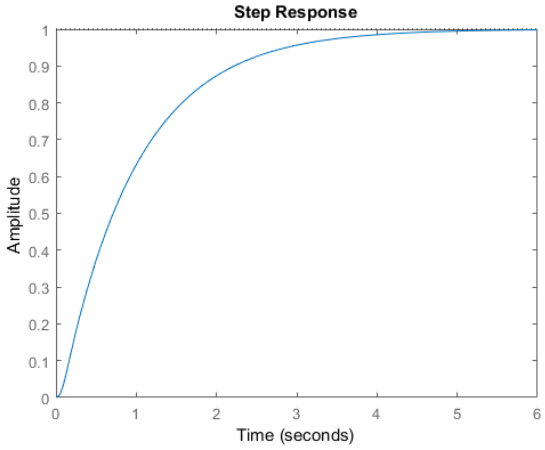


Figure 10: Closed loop step response

As expected the step response of the system in open loop is unstable given the presence of an integrator (pole at s=0).
The system in open loop is marginally stable having 2 conjugated complex poles and another one in the origin. In closed loop (with unitary feedback) it is stable and we can see that has no stationary error because of the integrator. Parameters such as gain and phase margins and rise time can be improved according to the control methods implemented.

3 Classical Control Approach

For now we will be looking for classical control solutions, namely based on PID, Lead-Lag Compensator and Pole Placement. The control will only be applied to the modeled system, which we have seen before it is fully controllable.

3.1 PID Solution

In this approach, we tried to get the lowest settling time and overshoot as possible, for each controller and watch its effect on the target tracking (camera coordinates)

For a P controller (figure 11), which represents a static gain, after some experiments a gain value of $K = 6.5$ was chose. This resulted in a system with a 0.6 percent of overshoot and a settling time of 0.35 seconds.

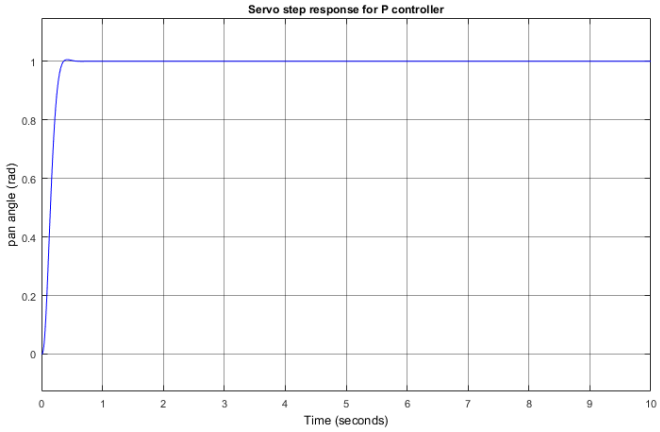
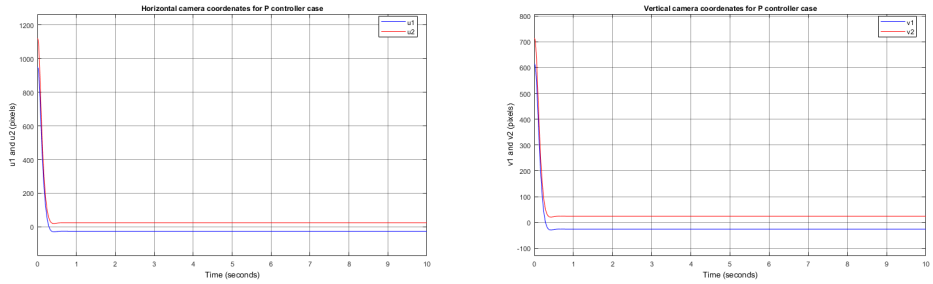


Figure 11: Step response for the P controller



(a) (b)

Figure 12: Camera coordinates when using a P controller

A PID controller (figure 13) has the following representation,

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

For this controller we chose the parameters values as $N = 3518$, $D = 1.846$, $I = 4.329$ and $P = 16.57$. This resulted in a system with a settling time of 0.9 seconds with less than 12 percent overshoot.

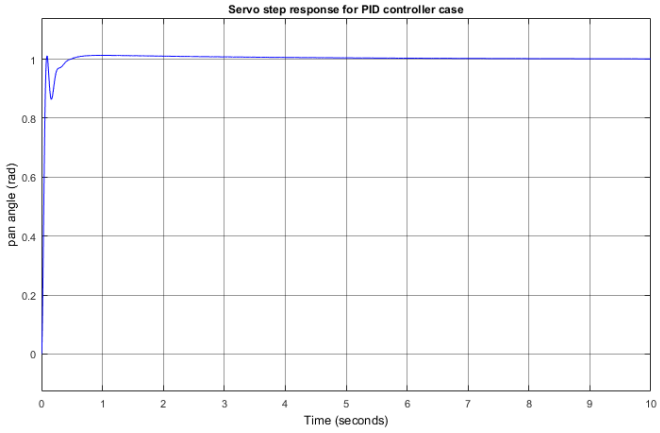
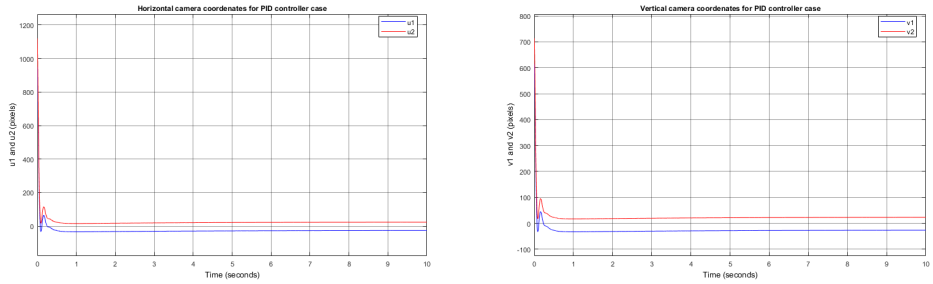


Figure 13: Step response for the PID controller



(a) (b)

Figure 14: Camera coordinates when using a PID controller

Analyzing both solutions we can see that the P controller fits the constraints having less settling time and less overshoot. We also considered PD and PI controllers but the results were worse than the PID controller, meaning not suitable.

3.2 Lead-Lag Compensator Solution

In this approach, we tried to get the lowest settling time and overshoot as possible, for each controller and also observe its effect on the target tracking (camera coordinates).

A Lead Compensator improves the transient response of the system and has the following representation,

$$\frac{\alpha s + a}{s + a}$$

For this compensator we chose $\alpha = 6$ and $a = 0.05$. This ended up in a system with null overshoot and a settling time 0.5 of seconds, represented in the figure 15.

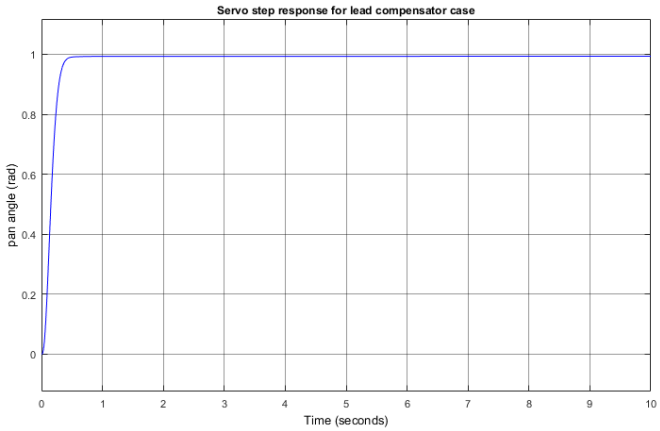
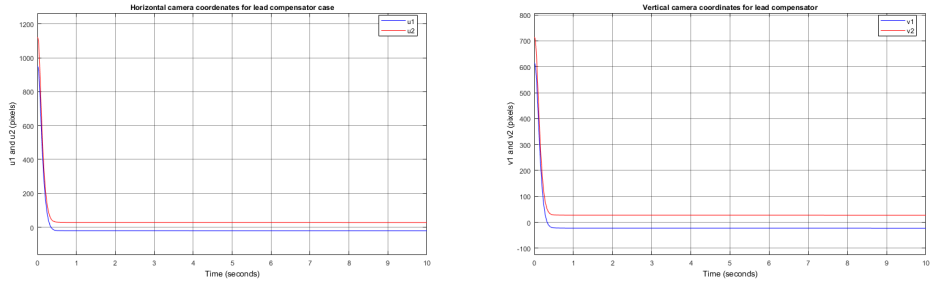


Figure 15: Step response for the lead compensator



(a) (b)

Figure 16: Camera coordinates when using a lead compensator

A Lag Compensator improves the stationary response of the system and has the following representation,

$$\frac{s+a}{s+\frac{a}{\alpha}}$$

For this compensator we chose $\alpha = 5$ and $a = 200$. This ended up in a system with 0.7 percent overshoot and a settling time 0.4 of seconds, represented in the figure 17.

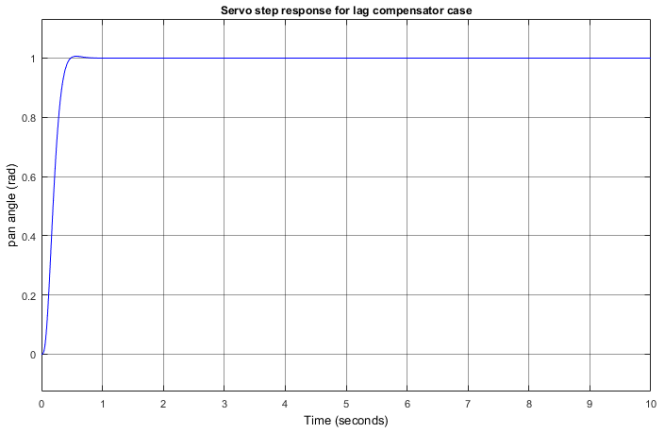
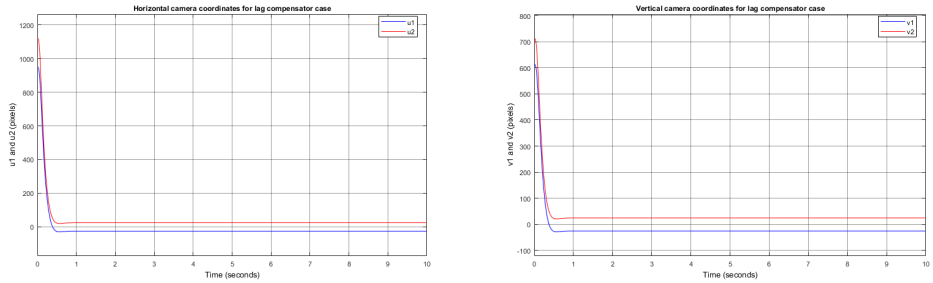


Figure 17: Step response for the lag compensator



(a) (b)

Figure 18: Camera coordinates when using a lead compensator

From the previous analysis, the most suitable is the Lag Compensator having less settling time and because lead compensator doesn't eliminate the position stationary error.

3.3 Pole Placement Solution

In order to our C matrix become, $C = [1 \;\; 0 \;\; 0]$, both A and B matrices becomes respectively,

$$A = \begin{bmatrix} 0 & 657.92 & 0 \\ 0 & 0 & 1 \\ 0 & -657.92 & -40 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Since the system is completely state controllable we can apply the Ackermann's Formula which can be rewritten as,

$$K = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} B & AB & A^2B \end{bmatrix}^{-1} \phi(A) \tag{19}$$

Three solutions were obtained. For the first case, the desired poles were

$$s = -22, \quad s = -20, \quad s = -18$$

We made this choice because we wanted null overshoot and the lowest settling time possible. This resulted in, $K = [12.0379 \;\; 538.08 \;\; 20.0]$ with a settling time of 0.4 seconds. Since the gain values were unrealistic we decided to decrease the values by choosing another set of poles (second case, figure 19)

$$s = -20, \quad s = -15, \quad s = -10$$

Which resulted in, $K = [4.5598 \;\; -7.92 \;\; 5.0]$. On this case, we increased the settling time to 0.6 seconds. We decided to keep the desired poles on the real axis so that we don't have overshoot at all.

In a third case (figure 21, we decided to let our system have the a bit of overshoot but trying to have the lowest settling time possible as well. This way we chose the desired poles as,

$$s = -11.3 + 11.7i, \quad s = -11.3 - 11.7i, \quad s = -18$$

The result was, $K = [7.2386 \;\; 13.46 \;\; 0.6]$, a settling time of 0.3 seconds with less than 2 percent overshoot and the most practical and realistic values of gains.

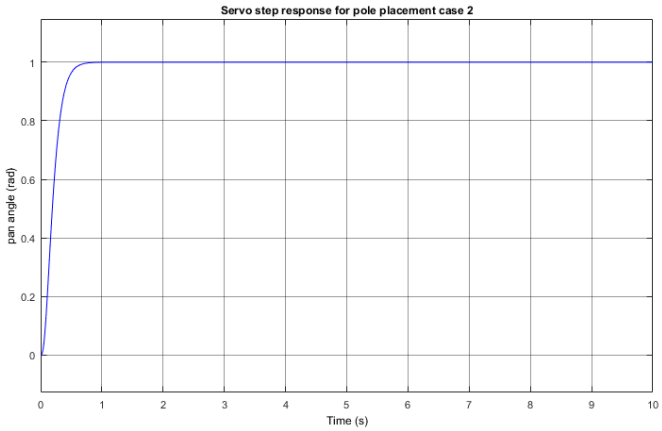
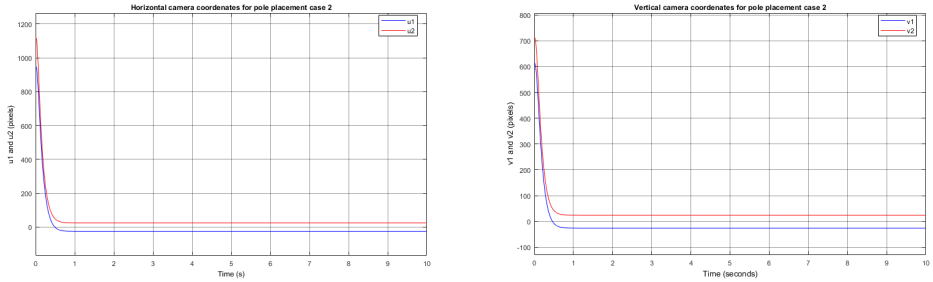


Figure 19: Step response for 2nd Case Pole Placement



(a) (b)

Figure 20: Camera coordinates for 2nd case pole placement

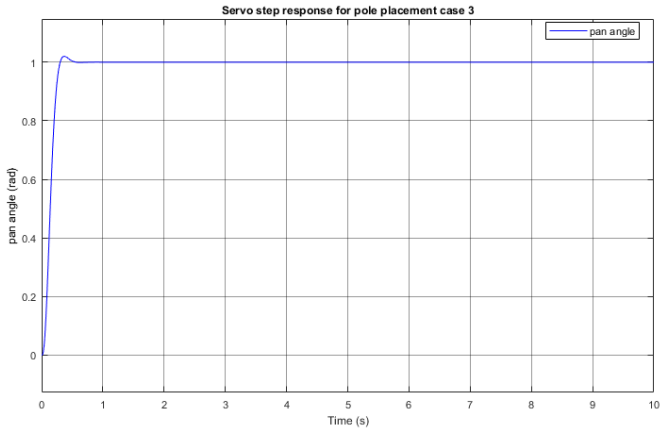
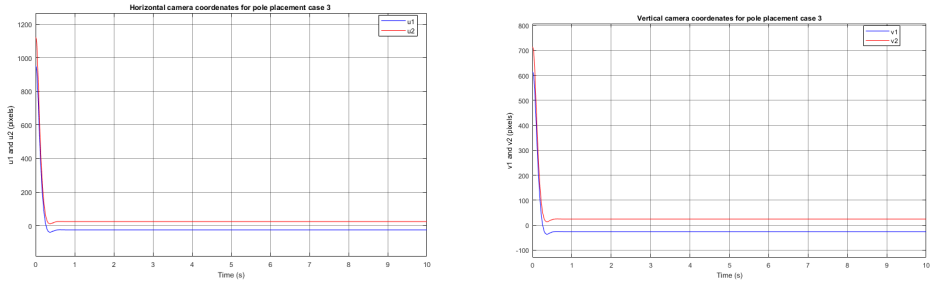


Figure 21: Step response for 3rd Case Pole Placement



(a) (b)

Figure 22: Camera coordinates for 3rd case Pole placement

4 Optimal Control Approach

4.1 Linear Quadratic Regulator

The main goal of the control problem is to design an optimal controller. To do so, we need to optimize a cost function. For our problem, the cost function will have both the optimal state-tracking control and minimum-energy control problems.

$$J(t) = \int_{t0}^{tf} (x^* Q x + u^* R u) dt \tag{20}$$

Since this performance index will have a quadratic form, to guarantee mathematical tractability, the linear quadratic optimal regulator (LQR) can be implemented.

The optimal control law is given by,

$$u(t) = -Kx(t) = -R^{-1}B^{T}Px(t) \tag{21}$$

and the optimal matrix K is given by,

$$K = R^{-1}B^{T}P \tag{22}$$

The matrix P must satisfy the Reduced-Matrix Riccati Equation,

$$A^{T}P + PA - PBR^{-1}B^{T}P + Q = 0 \tag{23}$$

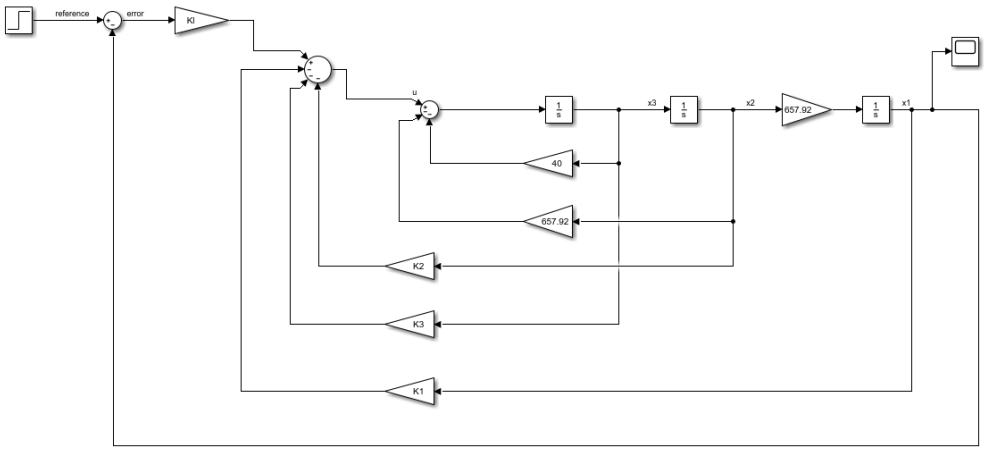


Figure 23: LQR implementation in Simulink

Two cases where taken in consideration,

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \quad R_1 = 0.01 \quad ; \quad Q_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \quad R_2 = 0.1$$

The gain matrices obtain are the following:

$$K_1 = [23.0859 \quad 766.3468 \quad 16.8568 \quad -100]$$

$$K_2 = [10.4315 \quad 370.0216 \quad 8.4772 \quad -31.6228]$$

The first case, figures 25 and 26, is the one with best performance but the gain matrix have a very big number. Because of that we decided take in consideration another case, a more practical one, in which the gain matrix was lower to a more decent value. This harmed the performance but not significantly, figures 27 and 28.

4.2 Design of State Observer

The model of the observer is the following,

$$\dot{\hat{X}} = (A - LC)\hat{X} + Bu + Ly \tag{24}$$

The close loop characteristic equation is given by,

$$|sI - (A - LC)| = 0 \tag{25}$$

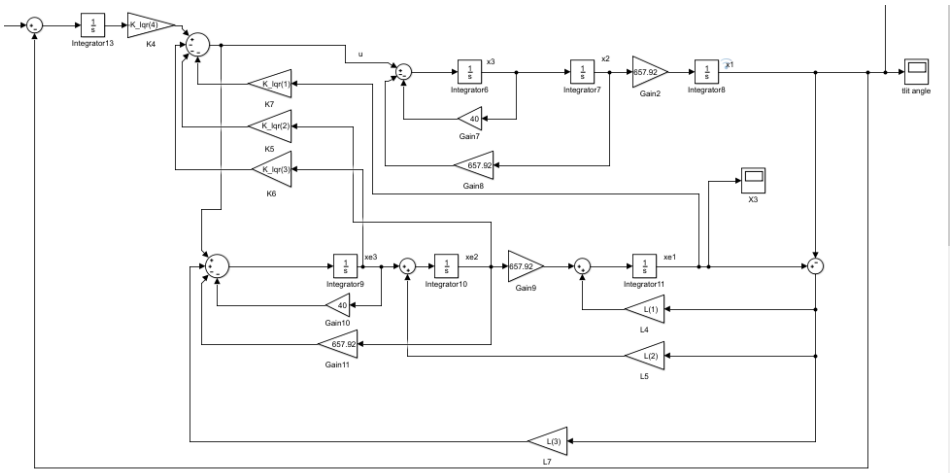


Figure 24: State Observer implementation in Simulink

The result was, $L = [113 \quad 3.9884 \quad -70.9937]$

Implementing the two gain matrices from the LQR into the State Observer we get the following step responses and camera coordinates,

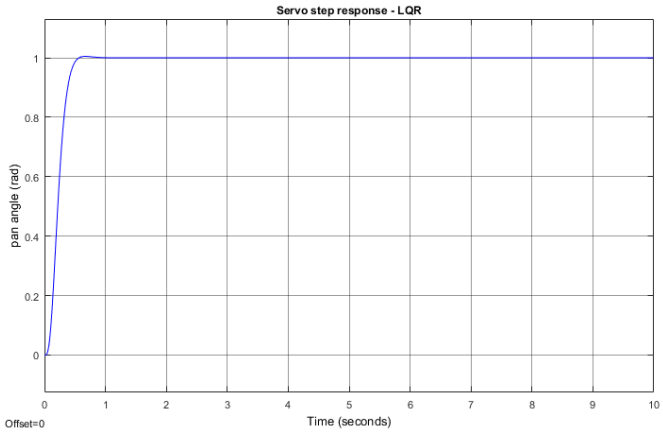
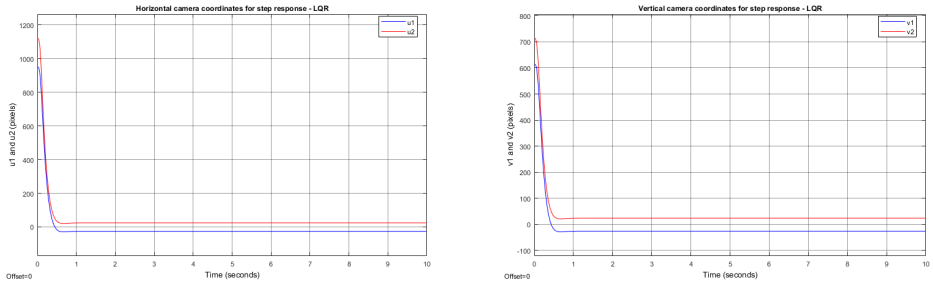


Figure 25: Servo step response using optimal control (R = 0.01)



(a) (b)

Figure 26: Camera coordinates for step response using optimal control (R=0.01)

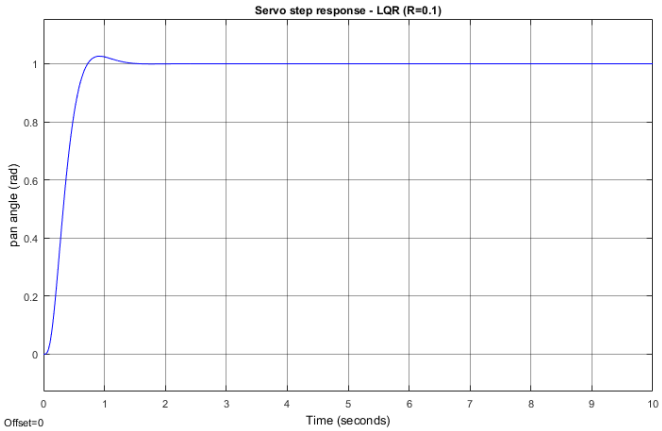
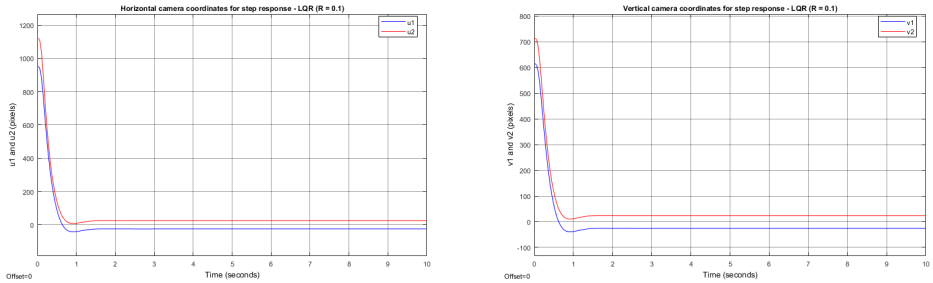


Figure 27: Servo step response using optimal control (R = 0.1)



(a) (b)

Figure 28: Camera coordinates for step response using optimal control (R=0.1)

5 Continuous Time Simulation

Having designed the controllers we can now perform a simulation with them in order to observe their behavior. To do so a rectilinear trajectory and a circular trajectory was created and the camera coordinates observed.

Since the trajectories do not apply only a step as input, an additional integrator was added to minimize possible errors. In the optimal control phase this was already considered but not in the pole placement approach.

Using the Ackermann's for formula with corrected A and B matrices, we were able to determinate the gains which needed to be used. Considering

the third case and adding $s = -15$ to the other desired poles have $K = [K_1 \quad K_2 \quad K_3 \quad -k_1] = [22.1785 \quad 607.42 \quad 15.2 \quad -106.1666]$

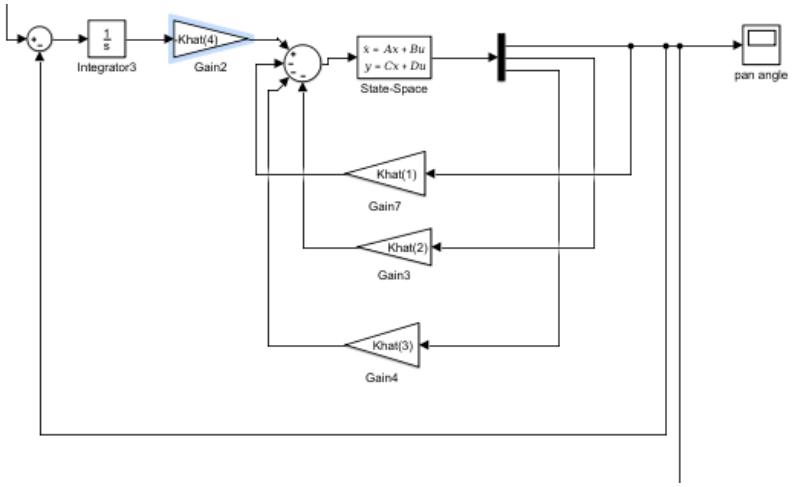
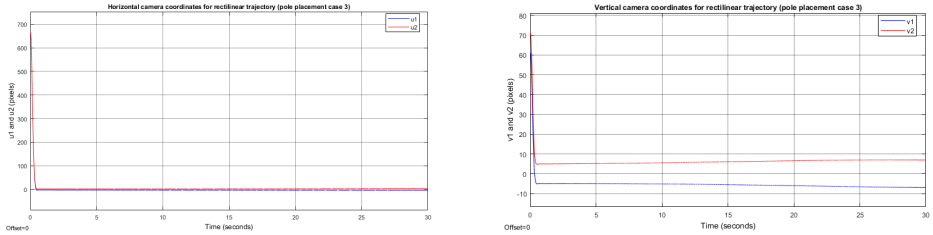
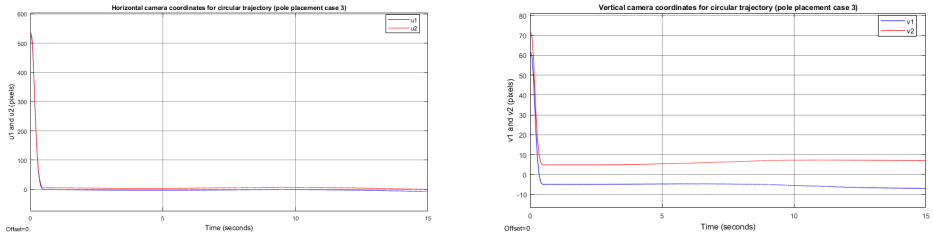


Figure 29: Servo with integrator control using pole placement approach



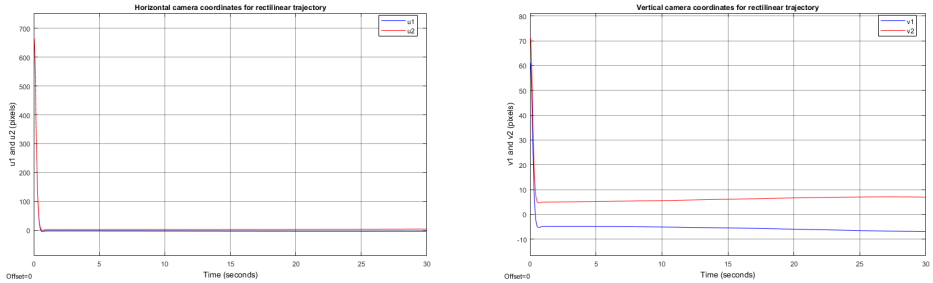
(a) (b)

Figure 30: Camera coordinates for a rectilinear trajectory using pole placement control



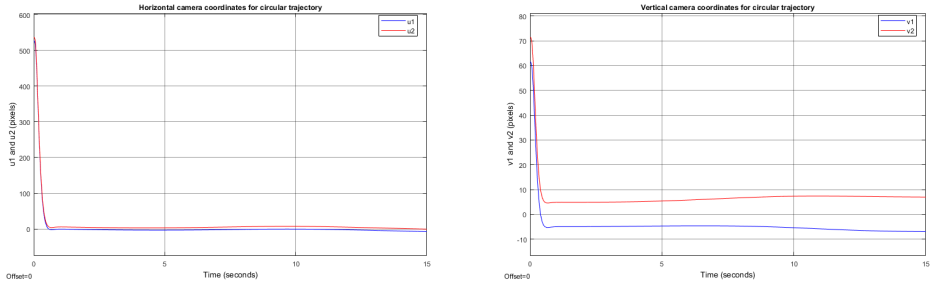
(a) (b)

Figure 31: Camera coordinates for a circular trajectory using pole placement control



(a) (b)

Figure 32: Camera coordinates for a rectilinear trajectory using optimal control



(a) (b)

Figure 33: Camera coordinates for a circular trajectory using optimal control

From the pictures above we can see that (both in the pole placement

approach and the optimal control) for the rectilinear trajectory the camera coordinates became symmetrical meaning that the camera would be centered with the target achieving the project's goal.

For the circular trajectory this goal was also achieved but with more difficulty. Nevertheless the error was never bigger than 1%.

6 Laboratory Work

In this chapter, the final tasks will be completed in order to the systems and the controllers previously simulated can be tested in a real prototype.

6.1 Stochastic Discrete Time System

In the servos state-space representation, the plant noise(*w)* and the output noise(θ) was introduced,

$$\dot{X} = Ax(t) + Bu(t) + w(t)$$
$$y(t) = Cx(t) + Du(t) + \theta(t)$$

Then the State-Space representation in discrete time becomes,

$$X(k+1) = Gx(k) + Hu(k) + w(k)$$
$$y(k+1) = Cx(k) + Du(k) + \theta(k)$$

The matrices G and H are obtained by,

$$G(T) = e^{AT} \qquad H = (T)\int_0^T e^{AT} d\tau B$$

with a sample time of T=1/20 (this value was chosen because the frame rate of the webcam is 20 frames per second). Using the *c2d Matlab function* the Stochastic discrete time state space representation is the following,

$$\begin{bmatrix} \dot{x}_1(k+1) \\ \dot{x}_2(k+1) \\ \dot{x}_3(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 27.4380 & 0.4149 \\ 0 & 0.5811 & 0.0165 \\ 0 & -10.8425 & -0.0741 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0.0083 \\ 6.3060 \cdot 10^{-4} \\ 0.0165 \end{bmatrix} u(k) + w(k)$$

$$y(k+1) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \theta(k)$$

6.2 Linear Quadratic Gaussian Control and Kalman Filter

The optimal control function results of the minimization of the Performance Index. The optimal control law is given by,

$$u(k) = -\hat{X}(k|k) \tag{26}$$

and the optimal matrix K is given by,

$$(B'PB + R)^{-1}B'PA \tag{27}$$

The matrix P must satisfy the steady state solution of the Riccati equation,

$$P - A'PA + A'PB(B'PB + R)^{-1}B'PA - Q = 0 \tag{28}$$

The Kalman-Bucy filter was used for the optimal state estimation design. The stationary Kalman filter gain matrix is given by,

$$L = GPC'(CPC' + \Theta)^{-1} \tag{29}$$

where P is the solution of the discrete algebraic Riccati equation given by,

$$G(P - PC')(CPC' + \Theta)^{-1}CP)G' + W = 0 \tag{30}$$

Finally, by introducing the equation $\tilde{x} = x - \hat{x}$, the equations can be written as

$$\begin{bmatrix} x(k+1) \\ \tilde{x}(k+1) \end{bmatrix} = \begin{bmatrix} G - FL & FL \\ 0 & G - KC \end{bmatrix} \begin{bmatrix} x(k) \\ \tilde{x}(k) \end{bmatrix} + \begin{bmatrix} I \\ I \end{bmatrix} w(k) + \begin{bmatrix} 0 \\ -K \end{bmatrix} \theta(k) \tag{31}$$

To determine both the L and K matrix gains we used the Matlab functions lqrd and kalman. The results obtained were $K = [K1 \ K2 \ K3 \ -k1] = [17,19 \ 583,71 \ 12,83 \ -66,58]$ and $K = [K1 \ K2 \ K3 \ -k1] = [17,19 \ 583,71 \ 12,83 \ -66,58]$ and $L = [L1 \ L2 \ L3 \ ] = [0,022 \ 2,61e - 06hspace2mm - 7,18e - 05]$. To implement in simulink we used the Kalman filter block.

6.3 Discrete model simulation

To test the discrete model created we used the same trajectories as before (one rectilinear and one circular). Bellow we can see the results obtained.

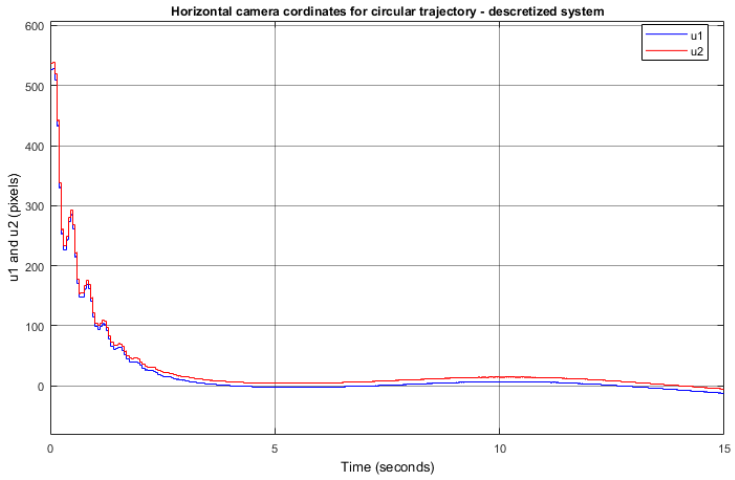


Figure 34: Horizontal camera coordinates for a circular trajectory

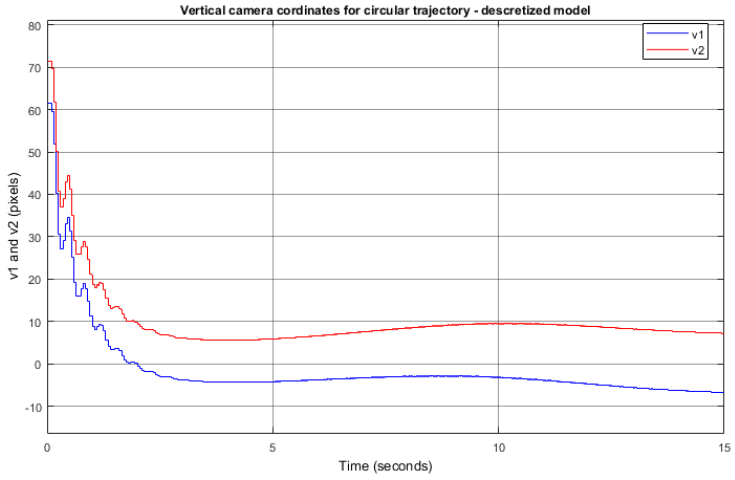


Figure 35: Vertical camera coordinates for a circular trajectory

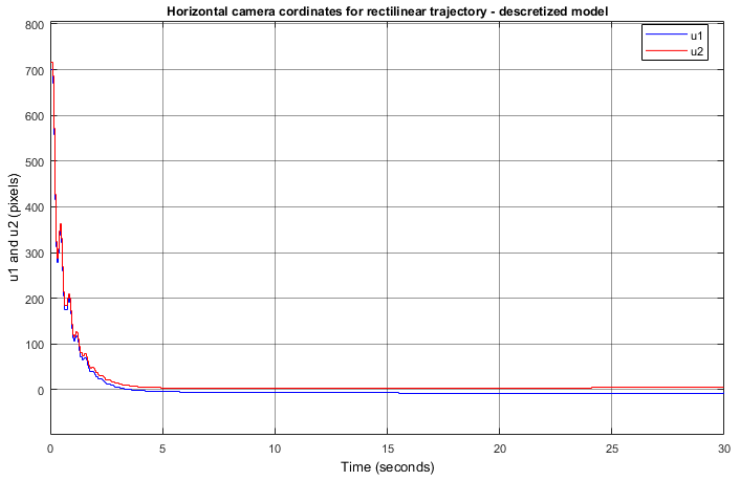


Figure 36: Horizontal camera coordinates for a rectilinear trajectory

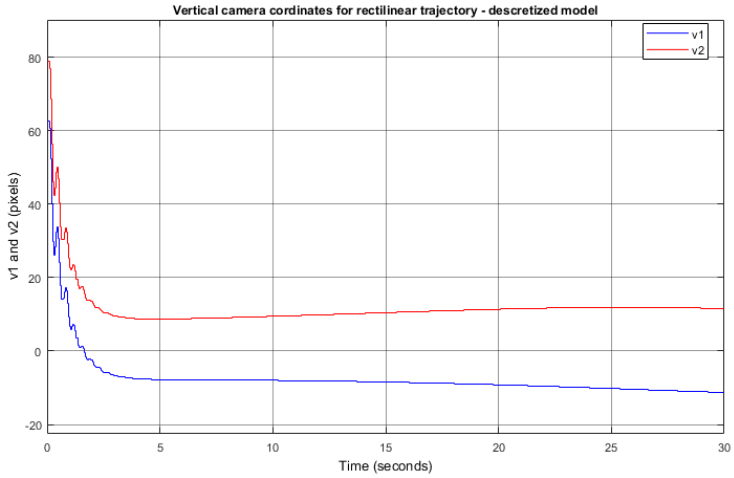
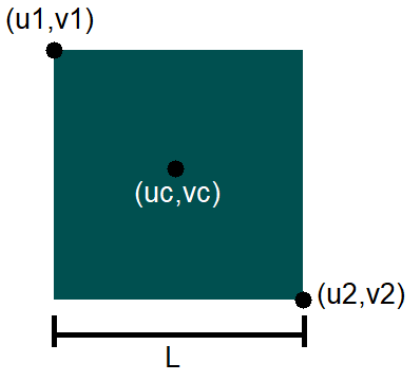


Figure 37: Horizontal camera coordinates for a rectilinear trajectory

Just as the continuous model the results were a bit worse in the circular trajectory but both were able to follow the target, which was the goal. So we can say that these simulations validate the model.

6.4 Target acquisition

A Simulink file was created to find the target, which consists of a green-blue square (HSV = [180, 100, 100], RGB = 008080). By filtering what is recorded by the camera to a binary scale, the target is located and some parameters are calculated (centroid, perimeter and target count, the latter one in order to assure there is no false target detected), and finally 4 coordinates (upper left and lower right coordinates points) are calculated as can be seen below.



P = Perimeter
Centroid coordinates = (u_c, v_c)
L = Square length = P/4
(u_1, v_1) = (u_c - L/2, v_c - L/2)
(u_2, v_2) = (u_c + L/2, v_c + L/2)

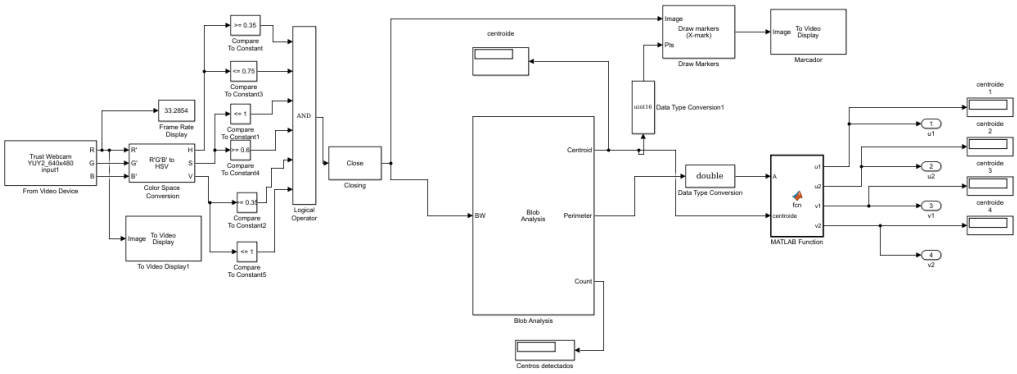


Figure 38: Target acquisition Simulink

6.5 Real Prototype

The Arduino Uno was used so that we could control the servomotors and consequently the webcam. Hardware toolboxes were used in order to control the servomotors. The connections were based on the following diagram from Mathworks website,

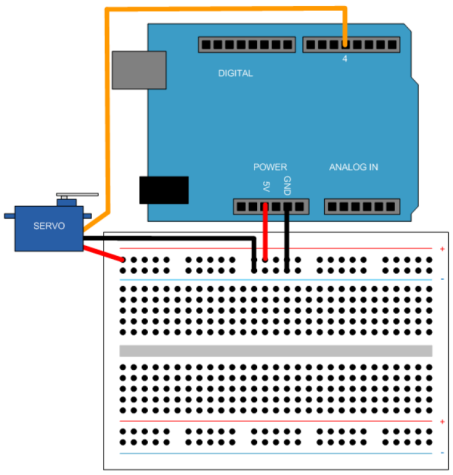


Figure 39: Arduino and Servo Diagram Connection

The real prototype can be seen in the following,

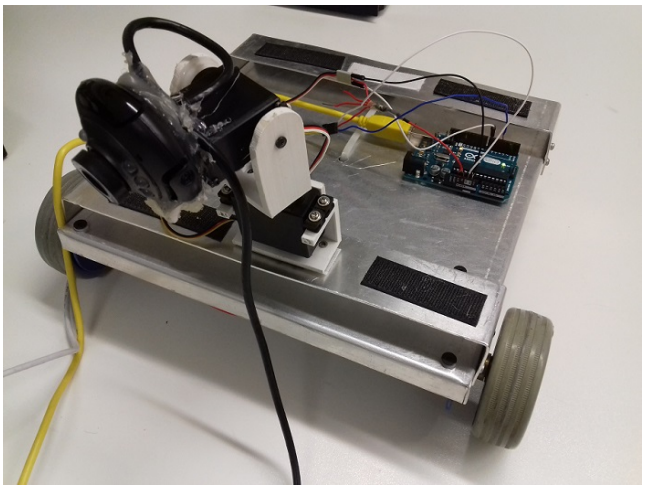


Figure 40: Real Prototype

Finally, the full/complete Simulink model of the project is the following,

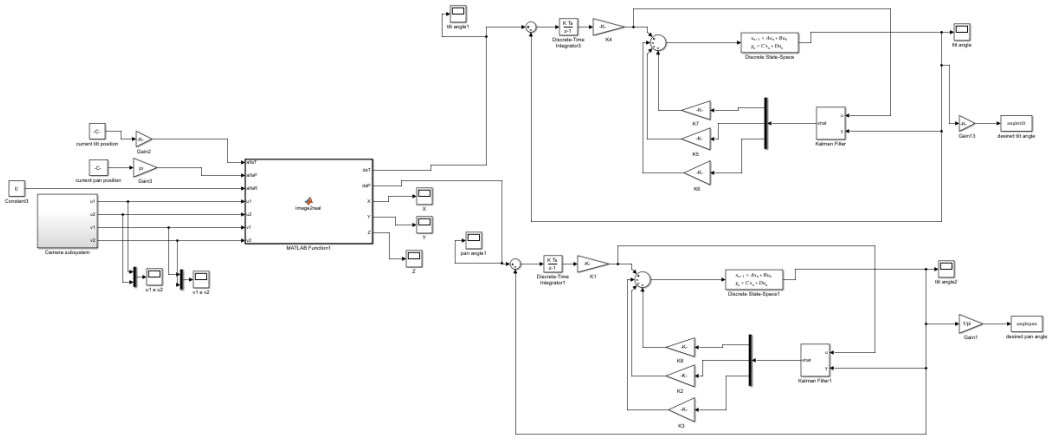


Figure 41: Full/complete Simulink model

7 Conclusion

The explained model and discussion above about the system characteristics results in the projected system. As expected the system is linear, observable, controllable and stable in closed-loop.

Referring to the classical controllers, we arrived at three possible solutions using a P controller, a Lag Compensator and the Pole Placement method. Although these are able to control the system properly, they are not optimal and therefor we used the Linear Quadratic Regulator(LQR) to obtain the optimal solution.

A State Observer was designed in order to every state variables become available. Implementing the LQR into the State Observer we verified its solution using a rectilinear and a circular trajectory.

The system was discretized and stochastic discrete time system was successfully obtained. The Linear Quadratic Gaussian Control (LQG) was designed using the Kalman filter. The Simulink models were validated by comparing them with the continuous time and respective solutions. Finally, we are going to test our solutions, in the lab, with the real prototype.

References

[1] Ogata, K., *Modern Control Engineering*. Prentice Hall, 5th edition, 2010.

[2] Brogan, W. L., *Modern Control Theory*. Prentice Hall, 3rd edition, 1990.

[3] Daniel, D., Armando, A., and Ramos, H., *Target Tracking with Pan-Tilt*. Instituto Superior Técnico, 2015.

[4] Park, J., Hwang, W., Bahn, W., Lee, C., Kim, T., Shaikh, M., Kim, K. and Cho, D., *Pan/Tilt Camera Control for Vision Tracking System Based on the Robot Motion and Vision Information*. IFAC 18th World Congress, 2011.

[5] Nasiri, N., *Camera-based 3D Object Tracking and Following Mobile Robot*. Swinburne University of Technology, 2006.

[6] *www.arduino.cc/*

[7] *www.mathworks.com/help/supportpkg/arduinoio.html*