NORTHEASTERN UNIVERSITY

EECE 5580: CLASSICAL CONTROL SYSTEMS

FINAL PROJECT

Teaching Robots to See

David A. Sevilla Cazes

April 22, 2016

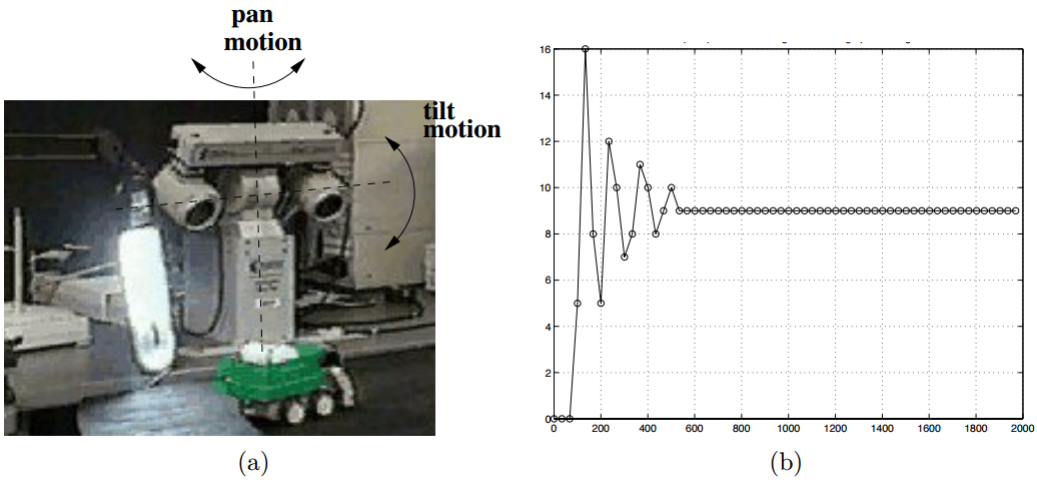
Abstract

This project explores classical control systems concepts that where cover by EECE 5580 by designing a controller that meet the performance specs for a robot vision system. The mathematical model for the system is derived from experimental data that contains a significant delay between the input signal and the system output. A controller is synthesized that yields a zero steady state error tracking, an system overshoot lower than 10% taking care that the controller output signal is within the operating specification of the robot head. Simulations are performed to validate the designed controller and the impact of neglecting the time delay is .

1 Introduction

Recent hardware developments have rendered controlled active vision a viable option for a broad range of problems, spanning applications as diverse as Intelligent Vehicle Highway Systems, roboticassisted surgery, 3D reconstruction, inspection, vision assisted grasping, MEMS microassembly and automated spacecraft docking. However, realizing this potential requires having a framework for synthesizing robust active vision systems, capable of moving beyond carefully controlled environments. In addition, in order to fully exploit the capabilities of newly available hardware, the control and computer vision aspects of the problem must be addressed jointly. The goal of the EECE5580 project is to briefly explore some of the issues involved in this problem. [1]

2 Problem Description



(a) (b)

Figure 1: (a) The Bisight Head, (b) Step Response Experiment

Figure 1: (a) shows a stereo head currently available at Northeasterns Robust Systems Lab. This head is capable of pan, tilt and vergence motion (it has 4 degrees of freedom), but for the purpose of this project we will concentrate on pan (i.e. horizontal) motion. Figure 1 (b) shows the result of a step response experiment for the combined head/image processing system. The input here is a step of magnitude 110 counts (encoder units for the head motor) applied at T = 0, and the output is in pixels. The actual data, corresponding to a sampling time Ts = 33.4ms is given in Table 1 that is located in Appendix. Note that the system has a time delay due to the image acquisition and processing time.
The EECE5580 project consists in designing a pan controller to track (with zero steady state error) an uncooperative target. [1]

2.1 System identification

The experimental data form Table 1 was used to obtain the model approximation of the plant neglecting the time delay. The experimental data with out the time delay was run through MATLAB `ident`

Time (msecs)	Position (pixels)
0	0
33.4	0
66.7	0
100.1	5
133.5	16
166.9	8
200.2	5
233.6	12
267.0	10
300.4	7
333.7	8
367.1	11
400.5	10
433.8	8
467.2	9
500.6	10
534.0	9
567.3	9
600.7	9
634.1	9
667.5	9
700.8	9
734.2	9
767.6	9
801.0	9

Table 1: Step Response Data [1]

When using MATLAB `ident` function it was assumed that first nonzero position value is captured at $t = 14.4$ ms (the signal crosses zero at $t = 19$ ms) for a 2 poles, 1 zeros system, obtaining the plant's transfer function:

$$G_{ol} = \frac{2.067s + 247.5}{s^2 + 14.44s + 3013} \tag{1}$$

Figure 2 shows the step response of the estimated transfer function (solid line) on top of the time delay neglected experimental data for a 2 poles 2 zero and 2 poles 1 zero approximations.
When analyzing the step response of Figure 2 it is assumed that the 2 poles 1 zero approximation is a good model for the robot vision system.

2.2 Open Loop Dynamics

Once we have identified a good model for our system we have to do some open loop dynamics analysis of the plant. This is done by finding the system poles and zero location and computing the system's root locus to achieve and understanding of the open loop dynamics when the input gain in modified. Figure 3 shows the system's open loop dynamics layout.

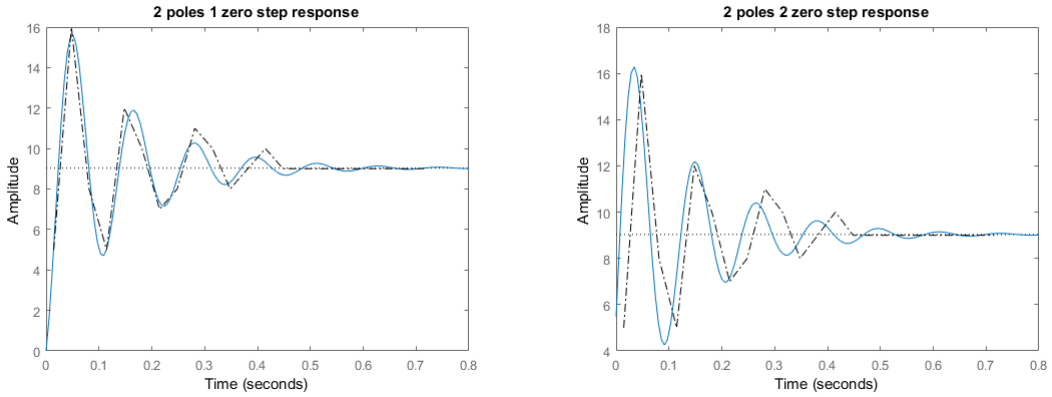


Figure 2: Model estimation step response: 2 poles and 1 zeros (left); 2 poles 2 zeros (right)

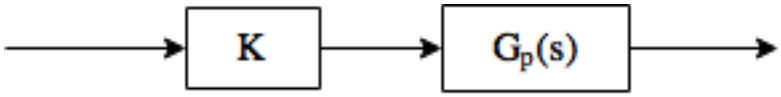


Figure 3: Open Loop

From the plant transfer function G_{ol} (eq. 1) has a zero at $s_z = 119.74$ and that the two poles are located at $s_p = -7.22 \pm j54.41$.
The plant characteristic equation is:

$$Q(s) = s^2 + 14.44 + 3013 = 0 \tag{2}$$

and that the characteristic equation of a second order system is:

$$Q(s) = s^2 + 2\zeta\omega_n + \omega_n^2 = 0 \tag{3}$$

from equations 2 and 3 we can obtain the damping ration $\zeta = 0.1315$ and that the system natural frequency $\omega_n = 54.89$ and remembering that the percentage overshoot can be calculated as:

$$M_{p\%} = \frac{-\zeta\pi}{\sqrt{1 - zeta^2}} \times 100 \tag{4}$$

The open loop system will yield a 66% overshoot which is out of the performance specifications. Using equation 4 it is calculated that the damping ratio has to be greater than 0.5912 ($\zeta = 0.5912$) to have achieve performance overshoot.
Figure 4 shows the plant root locus corroborating the minimum conditions that the system should hold to satisfy the overshoot specifications however this system would not have a zero state tracking error.

3 Controller Synthesis

In the previous section was stated that the open loop dynamics can be manipulated in order to satisfy the overshoot performance specification. However in order to achieve a zero state tracking error the open loop system has to be of type 1. It is important to remark that a type one open

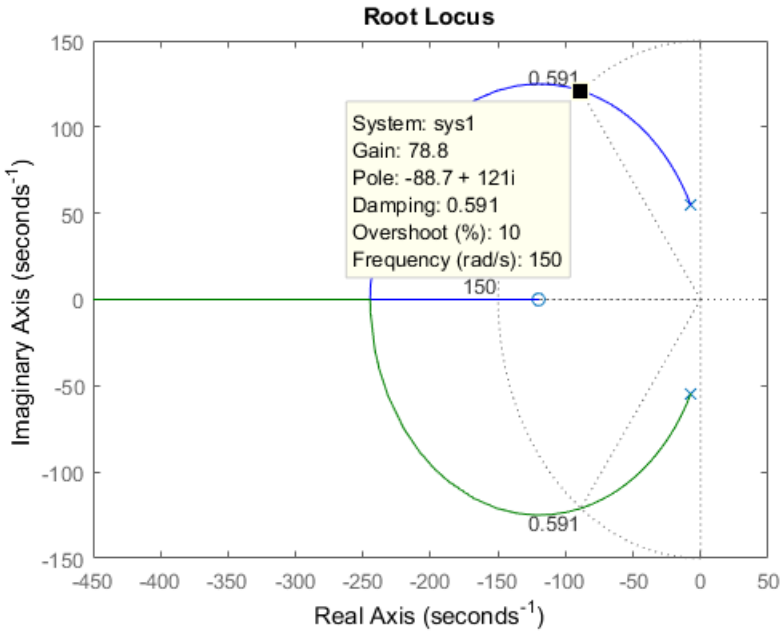


Figure 4: Plant Root Locus

loop system is unstable and can only be stable under a close loop configuration as shown in Figure 3.
Where G_p is the plant transfer function shown in equation 1 and $C(s)$ is a type one controller to be synthesized.

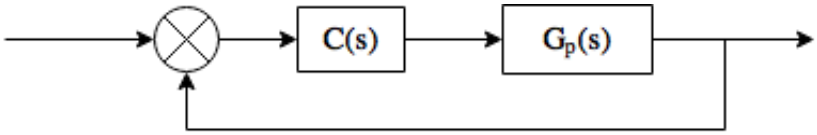


Figure 5: PID Controller

The first approach to synthesized the controller is to make the open loop a type 1 system. This can be achieved with a phase lead, a PI or a PID controller. So a PI controller is attempted with the form:

$$C(s) = K_p + \frac{K_i}{s} \tag{5}$$

$$C(s) = \frac{K_p s + K_i}{s} \tag{6}$$

$$C(s) = \frac{K_p(s+a)}{s}; \text{ Where } a = \frac{K_i}{K_p} \tag{7}$$

From equation 7 the PI controller adds a pole at $s = 0$ and a zero at $s = -a$.
As a first approach the PI controller zero is placed at an arbitrary position to the right of the plants zero (equation 1) that is located approximately at $s = -120$. The PI controller zero is

therefore located at the arbitrary position of $s = -60$ and its pole is located at $s = 0$. The gain K_p is set to a value close to what was reported from the open loop dynamics root locus (Figure 4) at a value of $K_p = 80$. Figure 6 shows the step response to the close loop system described in this paragraph.

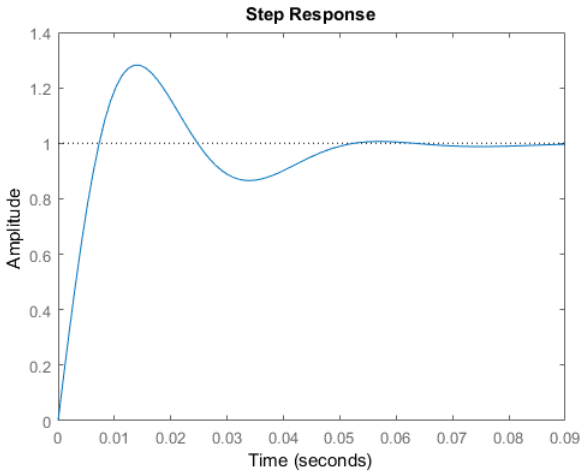


Figure 6: PI Step Response with Kp=80 and a=60

Figure 6 shows that the system has now a zero tracking error, however using MATLABS `stepinfo` function it is verified that the overshoot has a value of 28% and a settling time of 484ms. This can be corrected using an heuristic method that follows the the guidelines of table 2

	Rise Time	Over Shoot	Settling Time	SS error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	No Change

Table 2: Effects of K_p, K_i and K_d controller parameters [3]

After following heuristic guidelines from table 1 a PI controller was synthesized from equation 7 with $K_p = 14$ and $a = 30$ to form:

$$C(s) = \frac{14(s+30)}{s} \tag{8}$$

Forming a PI controller with proportional gain $K_p = 14$ and integral gain $K_i = 420$. Figure 7 shows the $C(s)G_p(s)$ open loop poles and zeros and Figure 9 shows the step controlled closed loop system step response. Using MATLAB's `stepinfo` it was verified that the system has an overshoot of 8.35%, and a settling time of 240ms bringing the overshoot into performance specifications and improving the settling time.

4 Analysis

Once the controller has been design a verification analysis is performed to validated that the system is within the specification. The analysis is performed with SIMULINK simulations.

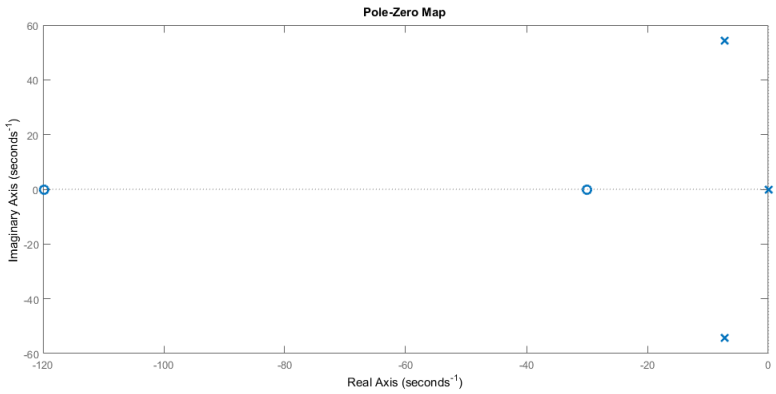


Figure 7: $C(s)G_p(s)$ poles and zeros map

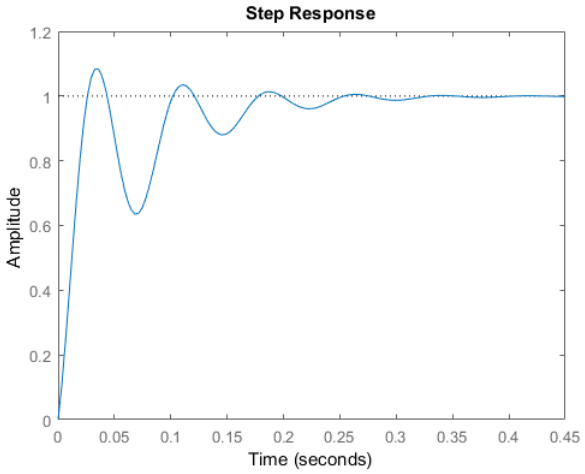


Figure 8: Controlled Close Loop Step Reponse

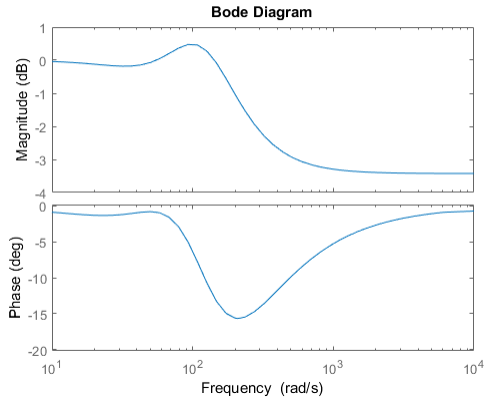


Figure 9: System frequency response

4.1 No Delay Simulation

Figure 10 shows the SIMULINK simulation diagram that was built to verify the results from the previews sections. With a step input with values from 0 to 9 pixels. The PID controller was set with a proportional gain (P) of 14 and integral gain (I) of 420. The derivative gain (D) was set to 0 and the filter coefficient (N) was left unchanged at 100. The plant transfer function was set in the Plant block as $G = \frac{2.067s+247.5}{s^2+14.44s+3013}$. Finally a scope is connected to the plant input to verify that the input to the robot head does not exceed 200 counts and a second scope is connected to the system output that should have a value of 9 pixels in steady state and an overshoot lower than 10%

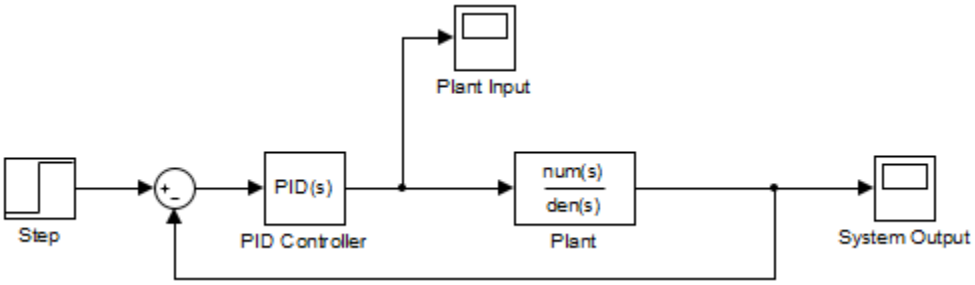


Figure 10: Simulink Model (No Delay)

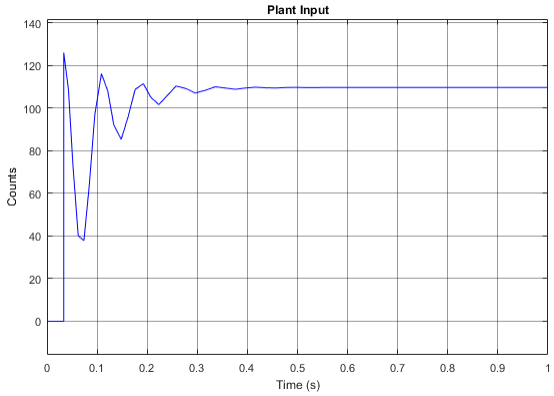


Figure 11: SIMULINK plant input signal

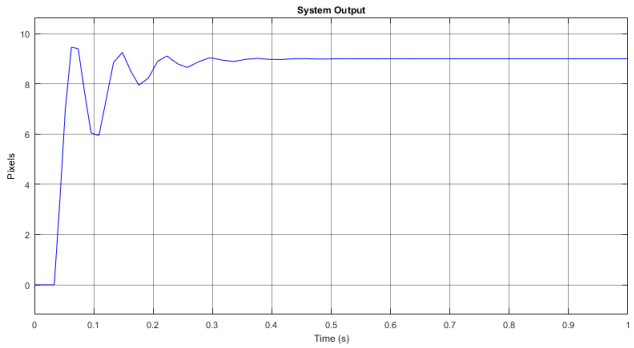


Figure 12: SIMULINK system output

From Figures 11 and 12 it is possible to conclude that for a system with no delays the controller that was design according to the performance specification since the plant input overshoot is slightly higher than 120 encoder unit counts and that the system output overshoot is lower than 9.9 pixels that mark a 10% overshoot.

4.2 Simulation with delay reincorporated

After proving that the controlled system works within the performance specifications it is of interest to know how the system would perform if we add back the time delay that was neglected form the experimental data.
Figure 13 shows the SIMULINK model of the system with an added time delay. This simulation was set up with the same parameters as the No Delay Simulation with and added time delay of 81.1 ms that corresponds tho the estimated moment in which the step response reacts to the input signal.(ie, 66.7 ms plus a 14.4 ms delay)

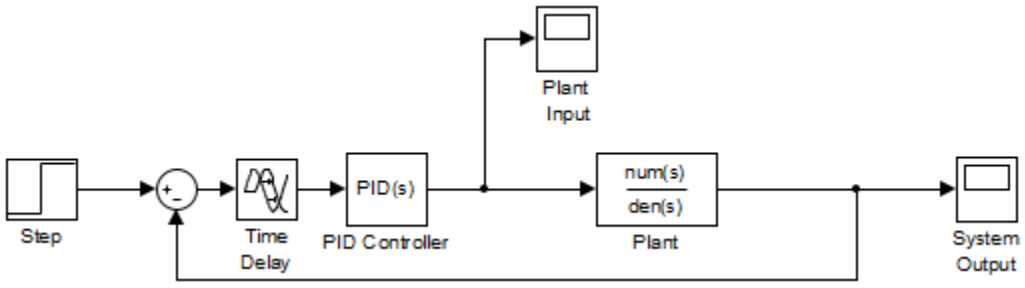


Figure 13: SIMULINK System Model (with time delay)

As it is expected Figure 14 shows that when a delay is added to a system the controller loses its ability to compensate and the system becomes highly unstable.

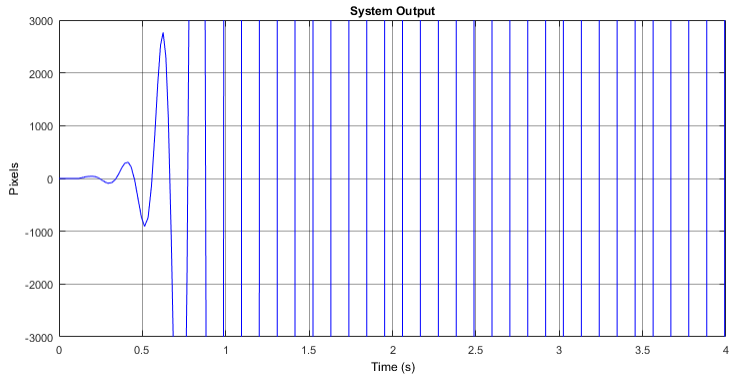


Figure 14: SIMULINK system output with delay

Since the delayed system is now unstable a new controller is needed. The design of the new controller was synthesizes following an heuristic method using the original controller as a starting seed. The resulting controller is a PI controller with $K_p = 1$ and $K_i = 100$ Figures 15 shows the new controller output that feeds the robot head input and 16 shows the system output.

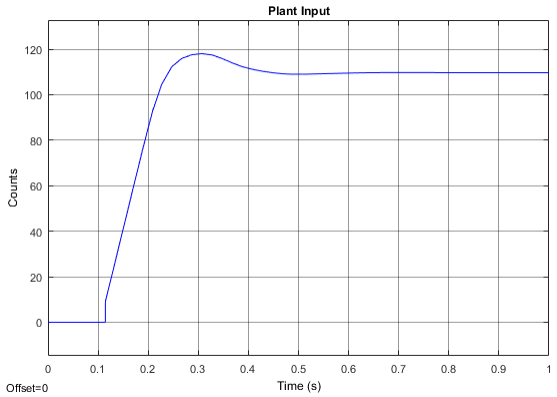


Figure 15: Delay Control Modified Plant Input

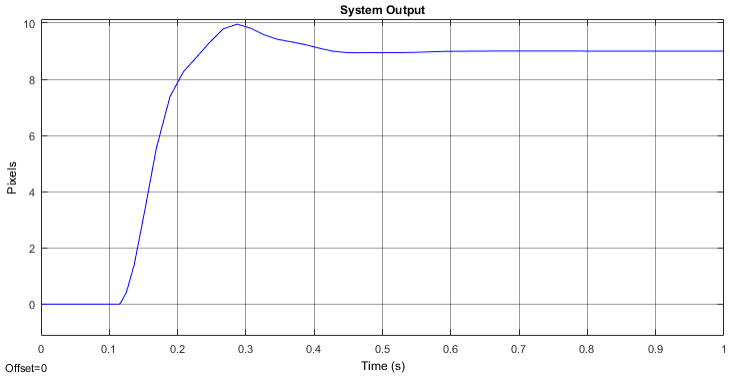


Figure 16: Delay Control Modified System Output

5 Conclusions

This project explores the implementation of a controller that has to meet some performance specification. Through out the completion of this project a series of classical control system tools where used that where cover by ECEE 5580 complete syllabus.
When designing the controller it was noted that given the tools available from ECEE 5580 it is probably more efficient to obtain the values of a PI/PID controller gains through an heuristic method rather than by finding an analytical approximation.
It was also observed that neglecting system delays might mathematically simplify a problem however delays should not be neglected since they will make a system unstable if the time delay is large enough since the measured error can easily lose its accuracy.
Finally if the step response data was corrupted $\pm$1pixel noise controller should be design to match all the performance specifications in the worst case scenario, ie. the design should be based on a step input of 10 pixels to match the overshoot specification.

References

[1] Mario Sznaier. *Final Project: "Teaching Robots to See"*. EECE 5580,Northeastern University, Massachusetts, Spring 2016.

[2] Phillips and Parr. *Feedback Control Systems*. Pearson. Ed. 5th. 2015.

[3] Control Tutorials for MATLAB and Simulink - Introduction: PID Controller Design, `http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID`