5-2011

High Resolution Vision-Based Servomechanism Using a Dynamic Target with Application to CNC Machines

Carlos Montes-solano
Clemson University, carlmontes@gmail.com

HIGH RESOLUTION VISION-BASED SERVOMECHANISM USING A DYNAMIC TARGET WITH APPLICATION TO CNC MACHINES

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

---

by
Carlos A. Montes-Solano
May 2011

Accepted by:
Dr. John C. Ziegert, Committee Chair
Dr. Laine Mears
Dr. Ardalan Vahidi
Dr. Darren Dawson

HIGH RESOLUTION VISION-BASED SERVOMECHANISM USING A DYNAMIC TARGET WITH APPLICATION TO CNC MACHINES

by

Carlos A. Montes-Solano

ABSTRACT

This dissertation introduces a novel three dimensional vision-based servomechanism with application to real time position control for manufacturing equipment, such as Computer Numerical Control (CNC) machine tools. The proposed system directly observes the multi-dimensional position of a point on the moving tool relative to a fixed ground, thus bypassing the inaccurate kinematic model normally used to convert axis sensor-readings into an estimate of the tool position.

A charge-coupled device (CCD camera) is used as the position transducer, which directly measures the current position error of the tool referenced to an absolute coordinate system. Due to the direct-sensing nature of the transducer no geometric error compensation is required. Two new signal processing algorithms, based on a recursive Newton-Raphson optimization routine, are developed to process the input data collected through digital imaging. The algorithms allow simultaneous high-precision position and orientation estimation from single readings. The desired displacement command of the tool in a planar environment is emulated, in one end of the kinematic chain, by an active element or active target pattern on a liquid-crystal display (LCD). On the other end of the kinematic chain the digital camera observes the active target and provides visual

feedback information utilized for position control of the tool. Implementation is carried out on an XYθ_Z stage, which is position with high resolution.

The introduction of the camera into the control loop yields a visual servo architecture; the dynamic problems and stability assessment of which are analyzed in depth for the case study of the *single CAM- single image processing thread*-configuration. Finally, two new command generation protocols are explained for full implementation of the proposed structure in real-time control applications. Command issuing resolutions do not depend upon the size of the smallest element of the grid/display being imaged, but can instead be determined in accordance with the sensor's resolution.

DEDICATION

I would like to initiate this section by thanking God for he has given me the opportunity, the capability and motivation to reach this very important stage of my professional career. I also dedicate this dissertation to my parents, Luis A. Montes Pico and Mayela Solano Cascante; your good advice and guidance were key for the completion of each one of the rigorous tasks that preceded this dissertation. I would also like to express my sincere thanks to my brother Luis A. Montes Solano and my two sisters Monica Montes Solano and Carolina Montes Solano. I thank all of you for the unconditional support throughout my years in graduate school. You made this often times difficult journey feel like a rather pleasant professional enrichment experience.

This work is also dedicated to all my other family members and friends, who always made an extra effort to make me aware of their presence and support and whose advice I have always held in high regard.

I now look forward to what should come next, and hope we can all share any positive outcome that might result after the conclusion of this cycle.

ACKNOWLEDGMENTS

It is a great honor for me to conclude my Ph.D. degree under the guidance of my mentor and academic advisor Dr. John C. Ziegert. Your knowledgeable input in the fields of Metrology, Machine Tool Design, System Dynamics and Controls were fundamental to my professional growth as a graduate student and to any contribution to Science and Engineering resulting from the present research. You also gave me freedom to investigate different branches in Engineering that were of my personal interest, even in instances when these topics were not directly related to our specific research subject. Your goal was always clear: guide me to become a researcher. Thank you.

****

The work presented in this thesis is the final outcome from a joint research collaboration with Dr. John C. Ziegert, Dr. Laine Mears, and Chan Wong. I would like to express my deepest gratitude to Dr. Laine Mears for all the technical input he provided during the development of this project. Thank you also for your very specific and relevant comments on chapters four through six of this thesis. I also thank Chan Wong, with whom I shared laboratory duties and who contributed significant content in the technical documents published during the project. It was a pleasure for me to work with such a talented group, and it is thanks to them that this dissertation was made possible.

****

I am very grateful to my committee member Dr. Ardalan Vahidi, from the Mechanical Engineering department at Clemson University, for all the graduate level

lectures on Systems and Controls he imparted, and that eventually became an important part of my final curriculum. I also want to express my sincere gratitude to Dr. Pierluigi Pisu for all the assistance he provided in the field of Advanced Control Systems and for his always positive attitude during our extracurricular research meetings. Your lectures on Sliding Mode Control and Hardware Fault Diagnosis highly influenced my research interests on these theoretical fields.

Additionally, I would like to acknowledge my colleagues Vincent Lee, Yousef Qaroush and Evan Lowe from the Mechanical Engineering Department at Clemson. Our interactions and discussions on different research topics and your kind attitude rendered an enjoyable atmosphere, which helped me carry through with my day-by-day tasks.

****

Special thanks are also extended to professor Hernan Robles from the Electrical Engineering department at the University of Costa Rica, for his reviews on an early version of the thesis proposal, document that preceded this dissertation. I also acknowledge all the professors from the Electrical Engineering Department at the University of Costa Rica, who helped me build a strong Engineering basis and encouraged me to pursue an advanced degree in Engineering.

****

I would like to thank Dr. Robert Lippert and Dr. Santiago Santi-Urena. Your hard work in the recruiting department at Clemson University has been a fundamental contributor to the academic growth of our great college.

****

The present research would not have been possible without the financial aid from the United States National Science Foundation. This Ph.D. dissertation is based upon work supported by the National Science Foundation under Grant No. 0800507. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER ONE

INTRODUCTION

In the manufacturing industry vast resources, both economic and human, are invested in the creation of new technologies to increase production rates while maintaining or increasing quality levels. The present research introduces a new automation method for manufacturing equipment, more precisely for machine tools. Although the main focus will reside in the mathematical derivation and implementation of a new accurate position control method, it is important to initiate the analysis by studying the fundamentals behind the construction and operation of common machine tools. In addition, a short historical background of the evolution of machine tools and their economical impact in the last two centuries is considered, in order to fully understand the overall relevance of new technologies in manufacturing.

The review presented next, briefly covering the historical and economical background of machine tools and some machine tool fundamentals, should serve not only as an introduction but also to denote the motivation for the development of the current research. It is important to bear in mind that the latest advances in machine tool metrology and position control systems are not the center point of this chapter, as they are for Chapter Three (Background Material), where the state of the art in the field under analysis is described.

Historical Background and the Introduction of Numerical Control

In the journal "Technological change in the machine tool industry, 1840-1910", by Nathan Rosenberg [1], the author states that between the second half of the 19th century and the beginning of the 20th century more and more importance was given to manufactured products, whereas other areas that had presented clear advances in previous years, like construction, started to be pushed into the background. Products were no longer generated based only on market needs but, in addition, were also required to meet a set of specifications. The fact that tolerance levels were being pushed up meant that more technological development was required. Moreover, companies in general became more specialized, which motivated manufacturing industries to improve their techniques. From an economic perspective, the rapid evolution of the manufacturing industry during the period mentioned above, according to Rosenberg, can be summarized in the flowchart shown in Figure 1.1. The economic chain effect resulting from the emergence of better manufacturing processes and technologies consequently increased technological innovation rates. Furthermore, the capital saved by reducing production costs actually represented savings for the whole economy.

Although the entire analysis is conducted for a period between the 19th and 20th centuries, a key conclusion can be extracted from Rosenberg's article; one that certainly serves as a motivation for a more engineering oriented project in the field of machine tool development. Rosenberg's article concludes that a big part of the economic growth of a specific region lies in the ability of the manufacturing sector to produce higher quality machine technology, and to constantly adapt according to new technological challenges.

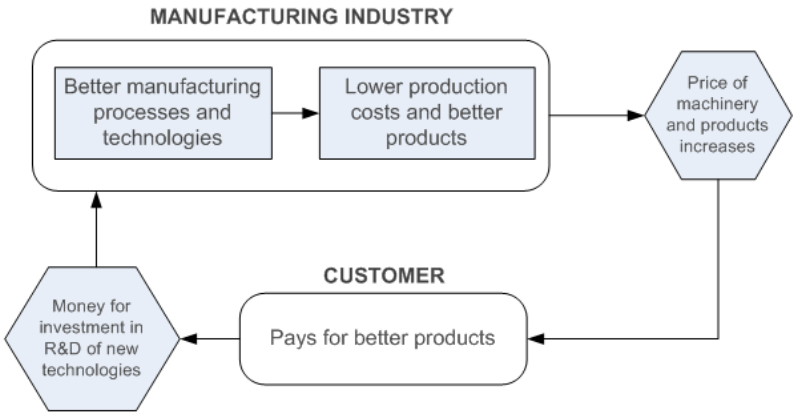


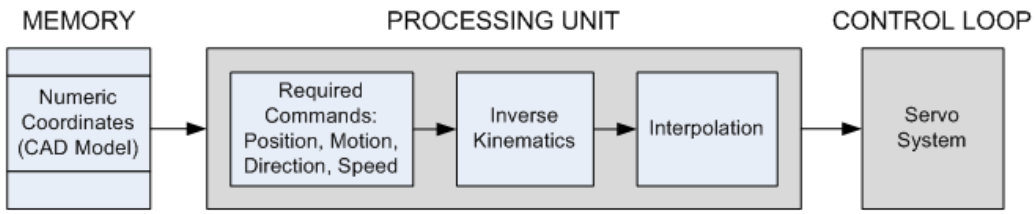
Figure 1.1: Economic flowchart in manufacturing industry (1840-1910).

In more recent years, the construction of more sophisticated mechanical and electromechanical equipment such as aircrafts, arms and transportation vehicles, took tolerance levels of machined parts to new levels. In many cases, and even more so today, more than 60% of the metal had to be removed through a machine tool to generate a small part for an airplane. The accuracy of these pieces had to be extremely high as human lives depended on them. Numerical control (NC) was born in the late 1940s and early 1950s as a result of these new technological requirements. Further research during the 1950s brought the introduction of new position control techniques to increase production rates through automation. The development of classical control algorithms along with the rapid growth of the computer industry during the following three decades gave birth to the so called computer numerical control (CNC) machine, which presented clear accuracy and efficiency advantages with respect to its predecessors.

All relevant parameters in a NC machine, such as position and motion commands, direction of motion and spindle speed are given through a sequence of numbers. The process of generating a part initiates by defining its geometrical properties. This step

usually includes a CAD model with specific units and dimensions. The data is fed into a processing unit (Figure 1.2 (a)), which computes the required coordinates and other commands for each individual axis. If the machine tool axes are built so as to be aligned with a Cartesian *X*-*Y*-*Z* axis, then the motion coordinates initially computed by the processing unit can be sent directly to the control system after interpolation, and only accuracy errors affecting the machine tool (geometric errors, thermal errors, tool wear, control system errors, etc.) would have to be compensated. However, aside from three translational axes, common five-axis machines present two rotational axes. Hence, displacement commands cannot be fed directly into the control system. Moreover, a kinematic model of the machine is required to map these displacement commands into real machine motions. The process of going from normal Cartesian coordinates to machine tool coordinates for each individual axis is defined as the inverse kinematic calculation. The reverse from that, going from axis coordinates to Cartesian coordinates, is called the forward kinematic calculation [2, 3].

(a)



(b)

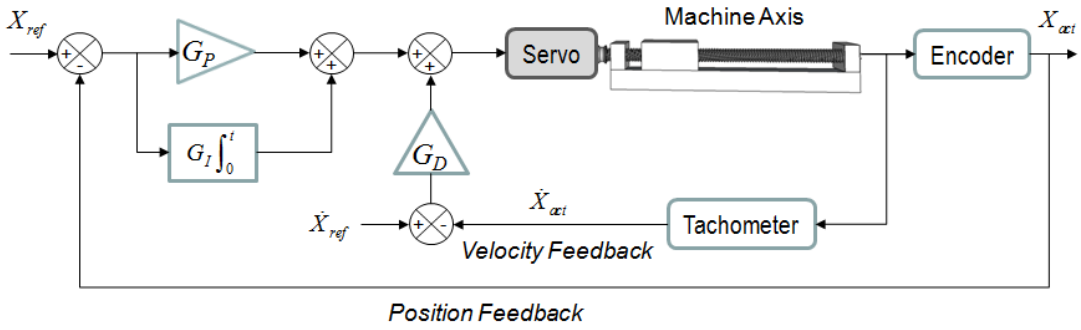


Figure 1.2: (a) Main electronic components of NC machine, (b) servo system under PID control.

In summary, compensation of the errors affecting machine tool accuracy as well as coordinate transformations of motion commands are performed through a complex kinematic model. Furthermore, from Figure 1.2 (a) it is clear that error compensation is carried out outside the control loop. If no on-line parameter estimation is performed, these models are only valid for a specific number of machine operations after which, costly maintenance or calibration procedures are required. Figure 1.2 (b) shows a typical single-axis servo mechanism operating under proportional-integral-derivative (PID) control. It can be seen that the position and velocity reference signals, X_{ref} and $\dot{X}_{ref}$, are determined by the processing unit of Figure 1.2 (a).

Machine Tools

A special characteristic that distinguishes machine tools from other types of machinery is the design criterion considered during fabrication. While the design of airplanes, ships and most construction vehicles is based on a strength criterion, machine tools are largely overdimensioned with respect to this measure. High accuracy requirements, sometimes on the order of micrometers or even smaller, make stiffness and rigidity the primary concern for engineers when designing a machine tool.

The components found in a machine tool can be classified into two main categories: structural and electrical components. Structural components can be further divided into two groups: stationary parts such as beds, columns and pedestals, and moving parts such as tables, carriages, slides and saddles (Figure 1.3). The second group performs coordinate motions and allows relative motion between the spindle and the workpiece. The interaction between stationary and moving parts as well as each individual part by itself must ensure high accuracy during motion (dynamics) and high stiffness (stationary) when the structure undergoes machining forces (cutting forces).

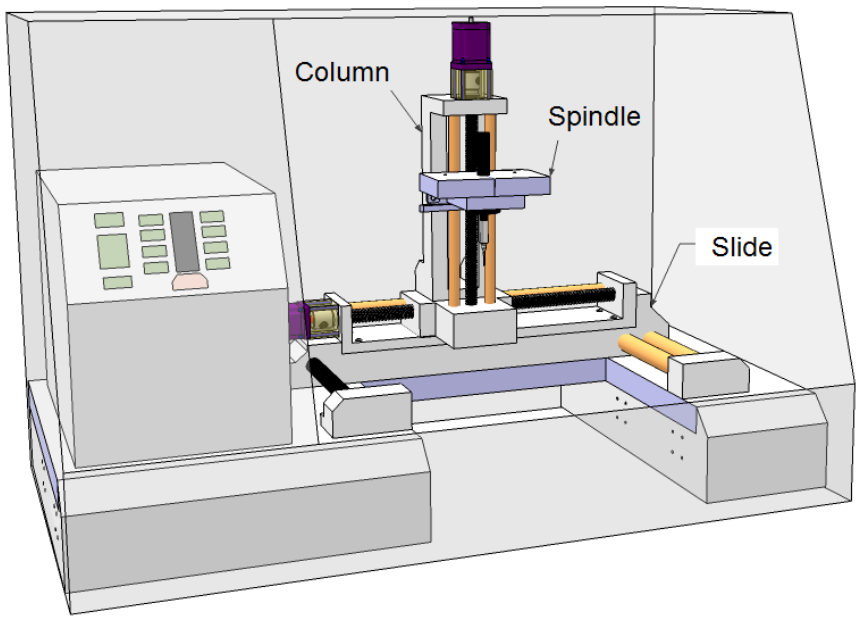


Figure 1.3: CAD model of a simplified three-axis CNC machine tool.

The second category, electric components, can also be further divided into two groups: drives and controls. The drives, which are basically motors and transmissions, are in charge of providing the torque required for the rotation of the spindle and, in automated machines, the forces used to produce the actual motion of the axes. The second group within the electric-component category is controls. Control systems act like automatic switches that either provide or cut the electric power fed to the drives to generate the motions in the desired directions and speeds.

Accuracy

After both structural and operational fundamentals of NC machines have been laid out, the final step of this introductory analysis concerns machine tool accuracy. In order to fully define machine tool accuracy this concept should be studied in conjunction

with another main concept in the field of precision engineering: workpiece accuracy [2]. Workpiece accuracy is mainly determined by dimensional errors of the workpiece itself, whereas machine tool accuracy is characterized by errors of motion and is highly affected by how the tool is manufactured. The diagram in Figure 1.4 shows the different factors affecting accuracy. It should be observed that machine tool accuracy, expressed as "Geometric Accuracy of the Machine Tool", is only one of the inputs that affect workpiece accuracy. A short outline highlighting the most important details of some of the factors in the diagram is presented next [4].

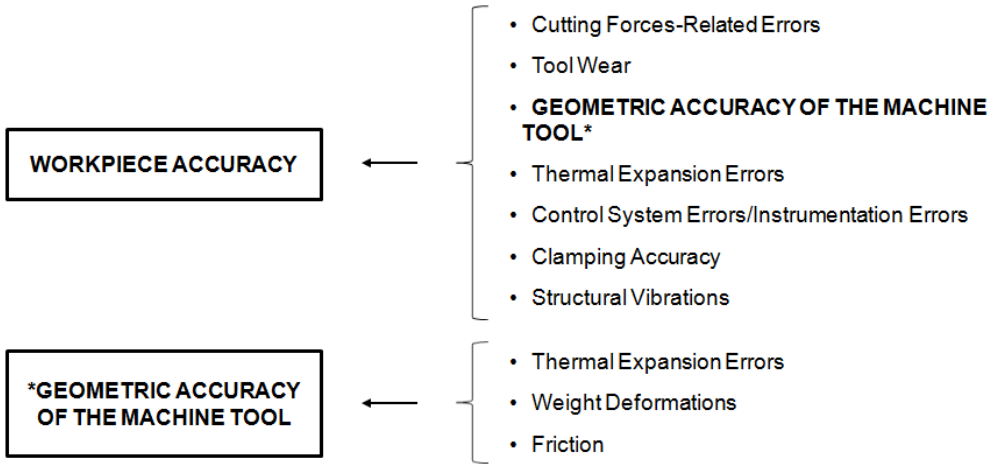


Figure 1.4: Machine tool- and workpiece-accuracy diagram.

External Loads-Related Errors (Weight Deformations, Cutting Forces)

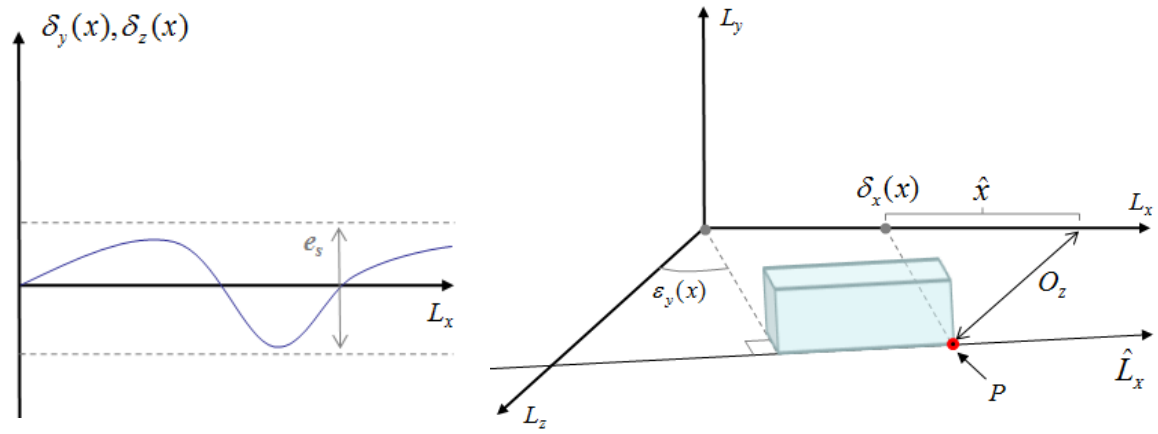
This first category includes error sources that influence both workpiece accuracy and geometric accuracy. Weight deformations constitute structural changes in the form of the machine, due to gravitational forces of some parts of the machine (tables, columns,

etc) acting on other parts over the machine life. This issue is usually addressed during the design process of the tool when the stiffness requirements are considered.

On the other hand, cutting forces affecting workpiece accuracy are typically determined experimentally based on measurements from several machines with similar characteristics. High speed cutting processes have come to mitigate, to a certain extent, errors caused by cutting forces.

Geometric Accuracy

Geometric accuracy of the tool is measured based on the errors of motion resulting from geometric imperfections of the individual parts of the machine (beds, columns, tables, carriages, slides, guideways and saddles). Some of the most important measurements used to describe geometric accuracy are: displacement errors along the axis, mutual squareness between guideways (axes), straightness of motion, and angular errors of motion. As an example consider a three-axis Cartesian machine, with a reference coordinate system L_X-L_Y-L_Z which passes through the center of the working space. Deviations in the *Y* direction when the machine is moving in the *X* direction along L_X are denoted by $\delta_y(x)$, where the subscript indicates the direction in which the deviation occurs and the variable inside the parenthesis indicates the direction of motion. $\delta_y(x)$ represents an error of straightness of motion *X* in direction *Y*. Figure 1.5 (a) shows how the straightness error (e_s) of motion *X* in *Y* or *Z* directions is measured. Similarly, deviations in direction *X* during motion in *X* are denoted by $\delta_x(x)$ and are simply called positioning errors.



(a) (b)

Figure 1.5: (a) Straightness error of motion *X* and (b) translative error difference due to angular error motion.

Aside from the three translative errors mentioned above affecting each individual axis, angular error motions also influence machine tool accuracy. These angular errors, known as roll, yaw and pitch are denoted by $\varepsilon_x(x), \varepsilon_y(x)$ and $\varepsilon_z(x)$ respectively, where again the subscript indicates the axis around which the rotation occurs and the variable inside the parenthesis indicates the direction of motion. As an example, consider again a reference coordinate system L_X-L_Y-L_Z shown in Figure 1.5 (b), and a point *P* that moves along a secondary axis $\hat{L}_x$ that is not parallel to L_X. Then the positioning error $\delta_x(x)$ maps to $\delta_x(x)_{\hat{L}x}$ at point *P* along $\hat{L}_X$ through

$$\delta_x(x)_{\hat{L}x} = \delta_x(x) + \hat{X} \tag{1.1}$$

If the offset distance between L_X and $\hat{L}_X$ at *P* is found to be O_Z and $\sin\left(\varepsilon_y(x)\right)$ is approximated by $\varepsilon_y(x)$, assuming a small angular error, then

$$\delta_x(x)_{\hat{L}x} = \delta_x(x) + O_Z \varepsilon_y(x) \tag{1.2}$$

It should be noted that if the measuring point changes, all calibration performed for the previous point must be readjusted based on the new offset distance. This is of course a tedious process. Geometric errors are usually compensated through calibration.

Thermal Expansion Errors

Temperature variations in the working environment, mainly originated from internal heat sources such as friction in the spindle bearing, the cutting process and electric components, impact both the geometric accuracy of the tool as well as the workpiece. A noticeable problem arises from different expansion coefficients associated to different metals. These errors are compensated by controlling the temperature of the working environment, using materials with low expansion coefficients and through the use of high pressure coolant pumps during the milling process.

Control System Errors/Instrumentation Errors

Most position sensors used in the control system of a given axis rely on interpolation algorithms to achieve high resolution levels. Furthermore quadrature signals, commonly used by linear glass scales and rotary encoders, are in many cases assumed to be perfect 90-degrees-out-of-phase sinusoids. The position of a slide, the spindle itself or the angle of a leadscrew are obtained by measuring the amplitude of the two sinusoids simultaneously. The result is a two dimensional vector which is interpolated along a perfect Lissajou curve to estimate the final value.

Errors Caused by Dynamic Forces (Structural Vibrations and Friction)

The actual contact between the cutting tool and the workpiece causes vibrations during the cutting process, which affect surface finish. A clear solution for reducing structural vibrations arises by increasing machine stiffness. This, however, increases manufacturing costs and reduces the ability of the control system and the drives to perform high speed motions. Dynamic forces are also responsible for frictional forces acting against machine motions. Although frictional forces help attenuate structural vibrations, they also cause machine wear and surface deformations which affect machine tool accuracy.

Outline

In most manufacturing machines, the position of the tool is determined through sensors that do not measure the tool position directly, but compute the position in an indirect matter. The rotating angle of a leadscrew or the actual displacement of the sensor along a linear scale are common measurements used by the controller, together with a kinematic model of the machine, to estimate position. This approach is subject to instrumentation errors; in addition the kinematic model does not exactly describe the actual machine due to imperfect straightness of the axis guideways, non-squareness of their motion directions, and thermal variations with time resulting in positioning errors. In this work, a new vision-based position control method for machine tools is investigated, with the intent of addressing geometric errors, thermal expansion errors and tool wear errors. The proposed system accomplishes this by direct observation of the multi-dimensional position of a point on the moving tool relative to a fixed ground, thus

bypassing the inaccurate kinematic model normally used to convert axis sensor-readings into an estimate of the tool position. The procedure is tested on an XYθ_Z stage, which is positioned with high resolution using direct position sensing.

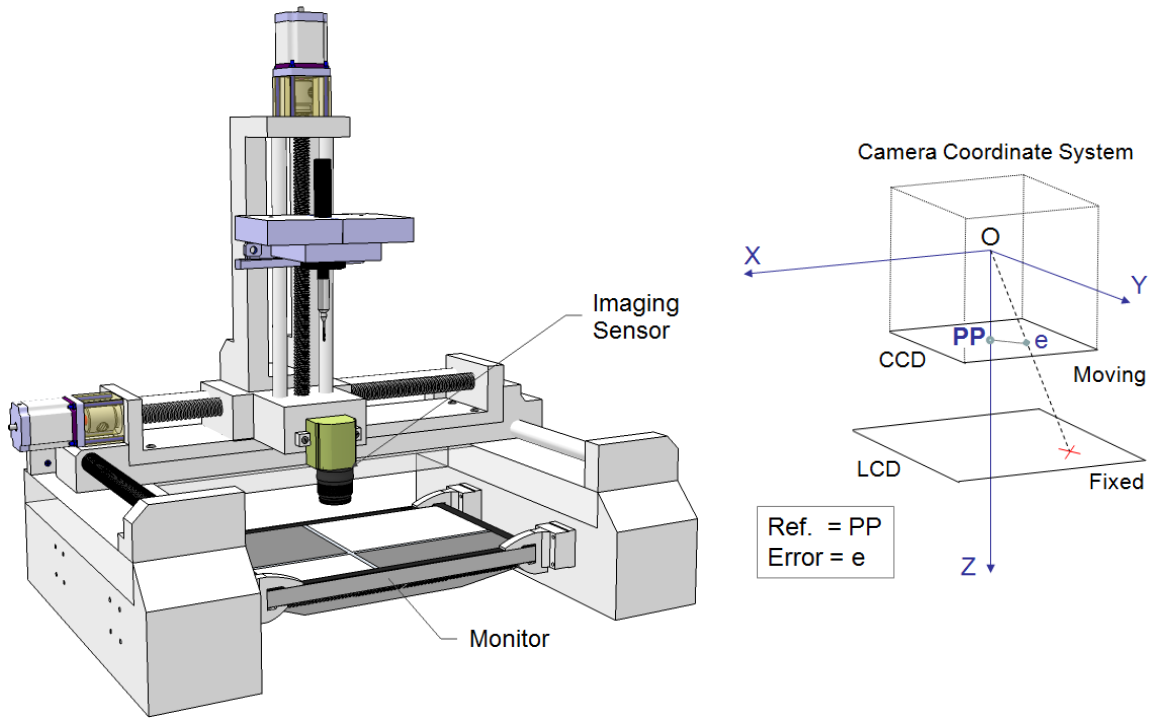
The core of this dissertation is presented in Chapter Four through Chapter Six. In Chapter Four a mathematical model of the direct-sensing position transducer, part of novel control system, is introduced. The problem is first approached by determining the physical relationship between the system parameters and the theoretically achievable resolution. Then, two new high resolution signal processing algorithms are presented, which are used to process the input data collected through digital imaging. Chapter Five is dedicated to understanding the overall control system, and assessing stability. Finally in Chapter Six, two new command issuing protocols are explained, which allow practical implementation of the methods in Chapter Four in real-time position control applications.

CHAPTER TWO

RESEARCH OBJECTIVE AND CONTRIBUTION

The objective of this project is the development of a signal processing algorithm suitable to retrieve high-precision position and orientation information in a new three-degrees-of-freedom (3-DOF) vision-based position control system for machine tools. The desired displacement command of the tool in a planar environment is emulated, in one end of the kinematic chain, by an active element or active target pattern on a liquid-crystal display (LCD). On the other end of the end of the kinematic chain, a charge-coupled device (CCD camera) is located so that the camera plane is parallel to the LCD. The CCD sensor observes the active target, which is dynamically modulated by the user (Figure 2.1 (a)). If the principal point PP (i.e. point where the optical axis intersects with the image plane) on the CCD (Figure 2.1 (b)) is regarded as the reference, then a position error vector of the target can be defined at any time t as $\mathbf{e(t)}=(x(t),y(t),\theta(t))_{Target}$, where θ is the orientation of the target with respect to the Z-axis. For this work, only planar positioning and in-plane orientation are considered. Motions in the Z direction must be monitored by another position feedback system. In the case of the tool in Figure 2.1 (a) the sensor is attached to the bottom of the column that holds the spindle and moves freely in the XY-plane based upon control action. Conventional systems sense displacements of individual axes and estimate the relative positions of tool and workpiece from the kinematic model. The new sensor directly observes from one end of the kinematic chain (the tool) to the other (the workpiece), and therefore doesn't require a kinematic model. The current position error of the tool is calculated referenced to the camera coordinate

system; the control system then acts as a planar tracking device, which moves the reference coordinate system (camera and therefore also the spindle) to bring the position error to zero. Due to the direct-sensing nature of the position transducer no geometric error compensation is required. A command generation protocol exists between the processing unit and the control system, where the processing unit reads the desired motion trajectory from memory and calculates the required coordinates, target shape and pixel intensities, and finally displays the motion command as a sequence of images on the LCD panel. The action of arbitrarily locating the dynamic target at some point on the LCD and consequently determining its position and orientation in the motion plane through the imaging sensor will be denoted as modulated target positioning, or MTP. The engineering challenge that such an active imaging system presents is to establish an appropriate signal processing algorithm for the purpose of simultaneous-axis positioning capable of achieving the desired resolution levels. Moreover, implementation of MTP also implies the need of an in depth study of the visual servo system from a control perspective, where relevant issues such as stability and performance optimization are analyzed.



(a) (b)

Figure 2.1: (a) Three-axis Cartesian machine tool with active target and 2D vision-based position control system, and (b) position error represented on pinhole camera model.

The control system found in most NC machines comprises mainly four components: displacement sensors, drives, a controller and a leadscrew-slide mechanism to generate the actual motion. Rotary or linear encoders, glass-scales and interferometers are common sensors used in these applications. Figure 2.2 (a) shows a typical 2 dimensional command generation diagram found in NC machine tools and a simplified version of the control loop. As seen the desired trajectory, $\mathbf{(x,y)_{des}}$, is computed outside the control loop and so is any error compensation used to correct errors inherent in the machine kinematic model and enhance the positioning accuracy. Hence, the controller cannot correct for external error sources such as geometric errors, thermal deviations, deflections under moving weights, etc., and is sensitive to changes in the value of the

desired command, $(\hat{x}, \hat{y})_{des}$, only. On the other hand, Figure 2.2 (b) shows the control loop configuration resulting by introducing the pixel display and the digital camera into the system. It should be noticed that the position of the tool is not measured indirectly through rotary or linear encoders, but it is measured directly with respect to the position of the target displayed on the LCD. The correcting action previously performed by the processing unit is no longer conducted outside the control loop. Instead, motion instructions are given by moving or modulating the dynamic target on the LCD, and thereby creating a position error between the target and a reference point in the camera plane. This implies that commands are processed as normal control errors inside the control loop, and any geometric or thermal errors of the machine will manifest themselves as a motion of the target image relative to the camera reference point, thus allowing the controller to compensate these errors and reject disturbances. The transfer function $\mathbf{H_m(s)}$ is used to translate the desired motion path from Cartesian coordinates to the corresponding target shape and pixel intensities. This setup presents clear cost advantages due to a reduction in the number of sensors per axis. However, from a control system perspective the diagram in Figure 2.2 (b) possesses a drawback with respect to the ordinary configuration, namely a control loop time delay due to slow hardware frame rate and long image processing times. This issue is addressed in Chapter Five.

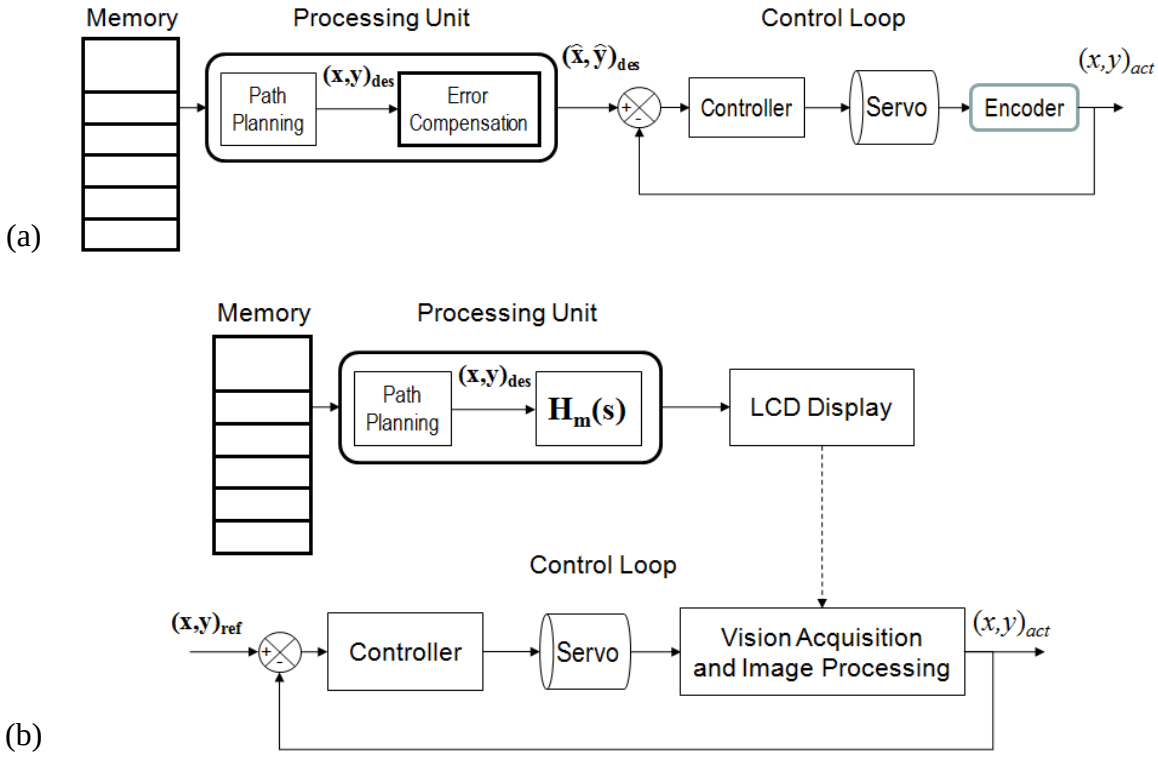


Figure 2.2: (a) Typical command generation diagram with simplified 2D control system, and (b) new command generation diagram through MTP.

The scheme resulting from the introduction of the digital camera and the display could be compared with the well-known visual servoing control scheme. Visual servoing has been widely used in the field of robotics for motion control. Feedback information regarding the position and orientation of the tool with respect to a reference frame (e.g. sensor frame), also known as pose, is estimated through a camera. Comparison between subsequent images allows the computation of coordinate transformation matrices, which represent translations and rotations of the robot within the working space. One of the key differences between visual servoing and MTP is the use of dynamically modulated target objects instead of common real-world fixed-shape objects. A dynamic object can be

generated on a LCD display and provides the user with a higher number of parameters to generate motion commands.

The question concerning the representation of motion commands by means of a dynamic target on a pixel display with specific application to machine tools has not been answered in the past. The new camera-display configuration, as it is contemplated here, utilizes different target formation methodologies and provides a new protocol theoretically independent of kinematic models. The isolated concept of two dimensional absolute positioning through an optical transducer and an illuminated grid, however, has indeed been analyzed by researchers. US-patent 6,765,195 [5] proposes a two dimensional absolute optical encoding method, where a known dotted pattern on a glass (or just transparent) surface is attached to the workpiece. A light source illuminates the known pattern, which is imaged by a stationary camera. A similar approach is presented in US-patent 6,937,349 [6] where a two dimensional scale is also illuminated by a light source. The light is then reflected onto the optical sensor according to the pattern on the scale. Aside from hardware setup, the biggest difference between these two patents appears to be the layout of the target that is utilized for position estimation: the first patent uses a dotted grid-like scale whereas the second patent uses a repeated quasi-random pattern. Of course, this also implies that two different image processing algorithms are proposed in these patents, in order to achieve the claimed resolution. None of these methods suggest the use of in-plane absolute positioning for real-time motion control or contemplates the use of dynamic modulation of pixel intensities in the target image.

Perhaps the most relevant approach to be considered, which exhibits clear conceptual similarities when compared with the proposed vision-based positioning system, is the one described in [7]. There, the two dimensional location of a mathematically calculated 3-tone target on a pixel grid is determined based on data acquired from a digital camera. In this case, target pixels have a 1:1 relationship with respect to sensor pixels, which somewhat reduces the problem to an alignment issue. The main idea is to rapidly and accurately determine the position of a known target on a larger image. This is achieved by mathematically generating a 3-tone deterministic pattern, considering only three grayscale intensities: white, gray and black. The two step process consists of a coarse target location, where the target coordinates in the plane are estimated with low accuracy, and a fine identification approach, which uses linear regression and the a priori knowledge of the intensity pattern to calculate the alignment deviation between the target pixels and the sensor pixels with high accuracy. Again, this method is not intended for position control applications, and much less for command issuing as the target under analysis is not only static, but its shape and pixel intensities do not change with time. Hence, it can be concluded that the target is not dynamic. The combination of three dimensional (3D) direct position sensing, dynamically modulated targets for the construction of visual displacement commands and an adequate control strategy, as proposed by MTP, is a unique and innovative approach that has the potential of reducing some of the common accuracy errors found in NC machines.

The contribution to the scientific and engineering communities arising from the current research project resides in the study and complete development of a new vision-

based position control servomechanism to be used in complex electromechanical systems such as high-precision NC machines. Specific contributions include two new image processing algorithms for real-time identification of a known pattern on an active display, two new command issuing protocols for in-plane absolute positioning and an in-depth analysis of the integrated system dynamics. Major problems such as unwanted perturbations in the sensor measurements due to noise, as well as destabilizing factors in the integrated control system are addressed. An appropriate strategy for implementation is presented. Full implementation of MTP is carried out on an XYθ_Z stage, which is positioned with high resolution.

CHAPTER THREE

BACKGROUND MATERIAL

This chapter provides an insight into some of the latest advances in machine tool metrology, image processing algorithms and visual servo control that are considered of interest for the project under development. Special attention is given to vision-based techniques for 2D and 3D displacement estimation and their application to position control. Each section in this chapter is self-explanatory and the topics do not follow a specific order. The review provided here should serve as an informative basis for the remaining chapters of this dissertation.

Error Compensation in Machine Tools

Different error sources (geometric errors, tool wear, thermally induced errors, etc.) affecting machine tool accuracy were discussed in Chapter One. The interaction between these errors and the machine tool components map onto undesired inaccurate behaviors in the machine's output motion [8]. This phenomenon, called error mapping, has lead scientists to develop error compensation techniques to achieve high accuracy standards.

Not only in machine tools, but also in robotics and many other mechatronic systems which operate under closed loop position control, position information is provided by one or several sensors. Errors associated with sensors are classified under instrumentation errors. Common position transducers such as rotary encoders or interferometers capture and process quadrature signals, which are used to compute

position. More precisely, two sinusoidal signals with a 90˚ phase shift with respect to each other and with a period $T = \lambda$ are generated. The way in which these signals are created and how they are read depend on the type of sensor. In the case of linear glass scales, a regular grid or grating is printed in a glass surface along the axis. A moving sensor produces a laser beam, which is projected onto the glass and the reflected light pattern is used to produce the sinusoids. In the case of optical rotary encoders, the pattern is printed on a disc (in a similar way as for the glass scale) that is attached to a rotating shaft, whose angle is to be measured. If, for example, the rotary encoder has *K* pitches in one rotation, a simultaneous reading of the sinusoids at any given location allows the sensor to determine the position of the shaft with an accuracy of $360/KN^{\circ}$, where *N* is the number of quantized sectors one rotation is divided into. Among errors sources in these sensors is the fact that the sinusoids are not exactly 90˚ out of phase; in addition, the two signals might not have the exact same amplitudes. Different methods have been presented to address these issues [9, 10]. Newer approaches aimed at improving accuracy in quadrature signal-readings, such as those described in [11] and [12], generate higher order sinusoids from the two measured signals. These higher order sinusoids are analyzed as binary pulses for easier and more accurate position estimation.

In order to increase the accuracy of machine tools, hardware improvement and software error compensation methods are desired. A variety of error compensation solutions, such as Homogeneous Transformation Matrices (HTM), rigid body kinematics, artificial intelligence- thermal modeling-control and software compensation techniques have been developed to address these issues [13-16]. A common pattern of most error

compensation algorithms is the tedious task of creating geometric and thermal models and performing measurements required to parameterize these models [17-19]. As more and more sensors are added to the machine, calibration and tuning become more complicated tasks [20]. The selection of calibration instruments, sensor placement, and the complexity of thermal deformations represent major challenges in improving the accuracy of the system [21].

In [22] two models, one for geometric error compensation and another one for cutting force induced error compensation, are approximated through a back-propagation neural network. The models are later combined into a single one to simultaneously compensate for geometric and cutting force induced errors in a 3-axis CNC milling machine. Experimental tests for linear displacements of the tool revealed a decrease in the average error from 100's of micrometers to approximately 40 μm or less. In slot cutting, the cutting force error was reduced from a range of approximately 347-237 μm to a range of 24-7 μm. Other studies have also used neural networks to address and compensate errors in machine tools [23-25]. Although these errors are indeed reduced, it takes several days to collect the data just to initiate the training process of the neural network [26].

In the field of machine tool calibration specific sensors, such as Heidenhain's grid encoder plate -KGM 181, KGM 182- [27], have been developed. Accuracy and repeatability of the machine, operated with common sensors, is tested and compared against data collected through such external sensors. The grid encoder provides accurate direct sensing capabilities that can be used to compare current position data acquired

through the built-in machine sensors with data obtained through this external transducer. Deviations between the data sets are used to calibrate the machine. The direct sensing nature of Heidenhain's transducer, which exhibits similarities with respect to the proposed LCD-camera setup of this project, is less vulnerable to geometric errors and thermally induced errors of the machine. Typical tests that can be conducted with the grid encoder are: circular interpolation, free-form and step response tests.

High Resolution Image Processing

In order to accomplish high resolution feature extraction from digital images three key aspects in the image acquisition process must be considered: resolution, contrast and distortion. Resolution is the ability to resolve a feature in the object of interest. Contrast determines how effective grayscale differences are detected by the camera, where a 100% contrast scenario is that where the systems detects white as white, and black as black in the digital image. Finally, image distortion corresponds to optical errors (aberrations) that affect the mapping of points in a given scene to points in the 2D image plane. These last errors are usually corrected through camera calibration techniques [28, 29].

The resolution of a digital camera is constrained to the discrete size of its smallest element, known as picture element or pixel. In order to improve optical resolutions beyond intensity-weighted distributions, different methods have been implemented to represent the data in a continuous form [30]. In [31] b-splines are utilized to continuously represent the digital image. Positive results are particularly evident in magnified images where resolution is increased without data losses. The algorithm consists of a series of iterative filters applied to the original image. The study of sub-pixel resolution techniques

has also play a key role in astronomy. In [32] the centroid calculation is applied to the discrete data in the digital image in order to accurately locate the position of stars to sub-pixel resolution. Such results gained in the field of astronomy are readily applicable in vision-aided manufacturing for high resolution positioning.

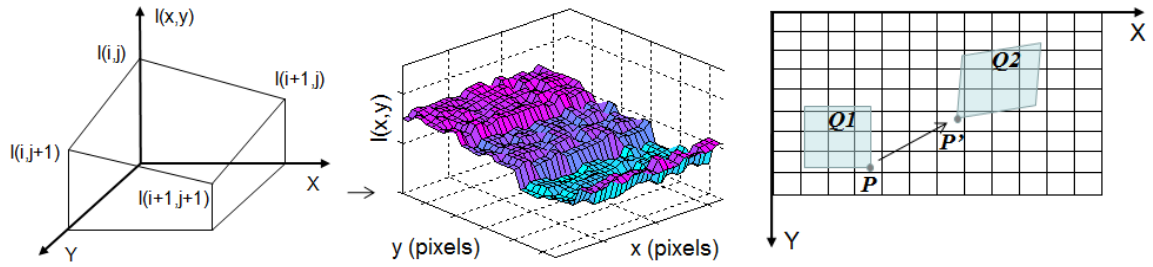
Displacement Estimation through Image Processing

Improvements in image processing and optics have helped engineers carry out part measurements and object detection with high resolution using non-intrusive techniques. Some of the latest techniques are described next.

Digital Image Correlation

Digital image correlation (DIC) methods have been used for more than two decades to perform different types of mechanical measurements [33-37]. Some of the most recent applications of this method are in-plane strain deformation measurements [38], thermal deformations [39], 3D surface characterization [40] and microscopic stereo vision [41].

The principle behind DIC is to perform measurements by numerically correlating the undisplaced/undeformed digitized intensity image of an object with subsequent images of the same object. The initial image is analyzed as a 3D discrete surface (intensity plotted against *XY*-image plane). A more useful representation of the same data is achieved by applying bilinear interpolation in small square areas, represented by 4 pixels. The result is a three dimensional continuous curve, with continuous partial derivatives every four pixels only (Figure 3.1 (a)).



(a) (b)

Figure 3.1: (a) Bilinear interpolation. (b) Deformation and displacement measurements through DIC.

Consider for example a square *Q1* that undergoes a deformation and a translation resulting in square *Q2* (Figure 3.1 (b)). It is assumed that there exists a homogeneous linear mapping between small segments in *Q1* and small segments of the same size in *Q2*. The goal of the algorithm comprises finding this homogeneous transformation and thereby estimating the type and size of the deformation. Then, for a given point *P* in Figure 3.1 (b) the form of the homogeneous linear mapping is assumed to be

$$P' = P + (u)_x + (u)_y + \frac{\partial\{u_i(u)\}}{\partial x}dx + \frac{\partial\{u_i(u)\}}{\partial y}dy, \quad i = 1,2 \tag{3.1}$$

where u is the displacement vector and $\partial u_i/\partial x$, $\partial u_i/\partial y$ are the components of the deformation gradient. Equation (3.1) defines the transformation of the set of all points in *Q1*, denoted as $\mathbf{x_{Q1}}$, onto the set of all points in *Q2*, denoted as $\mathbf{x_{Q2}}$. The image correlation algorithm becomes an optimization problem, where the goal is to calculate the value of six mapping parameters ($u_{x,y}$, $\partial u_i/\partial x$ and $\partial u_i/\partial y$) that define the functional C in

$$C = \min_{\left(u_x, u_y, \frac{\partial\{u_i(u)\}}{\partial x}, \frac{\partial\{u_i(u)\}}{\partial y}\right)} \iint_{\mathbf{M}} \left[\mathbf{P}(\mathbf{x_{Q2}}) - \mathbf{P'}(\mathbf{x_{Q1}})\right]^2 d\mathbf{x}, \quad i = 1,2 \tag{3.2}$$

where $\mathbf{M}$ is the set containing the coordinates of all points in $\mathbf{x_{Q1}}$ and $\mathbf{x_{Q2}}$.

Edge-based Motion Analysis

Edge detection methods have played a key role in feature extraction in the field of image processing. Ranging from highly elaborate algorithms such as *Canny* [42], to more simplistic such as *Prewitt* [43], these methods have been commonly used to perform shape and object detection in digital images. In [44] displacement of the object of interest is studied based on the localization of the edges in the image plane. Moreover, edge finding is performed at different resolution levels; this is, the image is represented at different lower resolution levels in a pyramid structure. The analysis at lower resolutions increments computational speeds, due to a reduction of data elements. An initial frame is used as the reference point. The location of the edges in subsequent images at multi-resolution levels compared with the reference frame allows the estimation of object displacement.

An alternative representation of the edges is obtained in the frequency domain, which can also be used for motion detection. In [45] researchers study the effects of the wavelet transform in regions containing edges. Considering that in the neighborhood of an edge the first derivative modulus corresponds to a local maximum, investigators utilize the first-derivative of a Gaussian wavelet for the multiscale approach. A scale-space representation of the image is obtained, and edges are located based on local maxima of the wavelet transform. This, in turn, allows the extraction of points from object edges in a robust matter. These points are tracked over a sequence of images and displacement is thereby calculated.

Frequency Analysis for Displacement Estimation

A key characteristic of the wavelet transformation is that it provides spatial information about specific features (frequency content) of the signal under analysis. If the spatial-frequency content of a reference target image is known, this information can be correlated with the location of the same frequency content on a displaced image of the same object. In doing so, motion of a target image can be estimated. A mathematical property can be used to better describe the same concept. Consider a point $P(x)$, which is translated by Δx along the X-axis. This can be represented by the convolution

$$\hat{P}(x) = P(x) * \delta(x - \Delta x) \tag{3.3}$$

where $\delta(x)$ is the Dirac's delta function. Applying the Fourier transformation to (3.3) yields

$$\begin{aligned} \Im\left\{\hat{P}(x)\right\} &= \Im\left\{P(x)\right\} * \Im\left\{\delta(x - \Delta x)\right\} \\ &= P(v_x) e^{-2\pi j v_x \Delta x} \end{aligned} \tag{3.4}$$

where $P(v_x)$ is the Fourier transform of $P(x)$. Evidently, a translation is represented by a phase change in the frequency domain. This property is used in [46] to develop a phase-sensitive technique to detect displacement of an object. A known frequency pattern is fixed on a flat surface of the object, which is imaged by a digital sensor. The wavelet transform is used to analyze the image and extract the initial position of the target with high resolution. The frequency content of subsequent images of the displaced object is correlated with the reference to estimate motion. A more powerful method, also based on the same space-frequency principle, is explained in [47] to perform six-degrees-of-freedom measurements.

Direct Object Sensing

Direct position sensing has been used in the past for two dimensional absolute positioning. Among the main approaches developed for this purpose are two US patents [5, 6], described in Chapter Two. Another relevant method, also described in Chapter Two, is presented in [7]. In this case, a camera is used to retrieve position information of a known target on a larger image displayed on a pixel grid. This is achieved by mathematically generating a 3-tone deterministic pattern, considering only three grayscale intensities: white, gray and black.

Visual Servoing

In *Visual Control of Robot Manipulators* [48] Peter I. Corke, one of the biggest contributors of the visual servoing theory, states

> "Visual servoing is the fusion of results from many elemental areas including high speed image processing, kinematics, dynamics, control theory, and real time computing."… "The task in visual servoing is to control the *pose* of the robot's end effector using visual information, *features*, extracted from the image."

The beginning of visual servoing dates back to the early 1970s, when initial documents on this topic were published. Initial applications in this field ranged from pick-and-drop object manipulation [49] to more complex research providing some analytical details on the dynamics problems in real-time visual servoing [50].

In the 1990's the study of visual servoing took an important turn when more attention started to be paid to the dynamic effects of the electromechanical plant to be controlled and the digital camera, which limit output performance[51]. It is further argued

that the focus of previous literature was set only on the kinematics of the visual control system. Moreover, in [52] the effects of the plant- and vision sensor-dynamics are closely analyzed to achieve optimal performance. To this effect, the use of inner loops to regulate output torques and motor speeds, as well as the development of compensators permit a significant increase in performance of the visual servo system. Different controllers are tested and the visual loop is modeled as a linear system. Tuning is carried out through pole placement in the root locus plot. A detailed tutorial on visual servoing, covering the dynamics effects of robotic manipulators and other relevant topics associated to image formation, correlation and coordinate transformation can be found in [53].

In much newer applications, position control through vision feedback has been implemented in more complex architectures which usually include models of the plant dynamics to compensate for delays in the feedback signal. For instance, in [54] a non-linear model is used to parameterize the pose of a target of interest with respect to the digital camera mounted at end effector of a robotic arm. The states of the non-linear model are calculated in each iteration based on the images collected through the digital camera. Experimental results demonstrate that the robot is capable of following a linear 3D path with the desired resolution though vision feedback. The use of optimal controllers in systems where both a mathematical representation of the open loop plant as well as the reference input signal to the control system are known a priori, has the potential of allowing anticipative correcting action through the control effort [55, 56]. Moreover, different approaches have been developed for optimal controllers that take into account a time delay in the feedback loop [57-59], which makes them attractive for

vision-aided control applications. In [60] the image of a human hand is captured and processed to determine its outline; this information is quantified and fed as an input to a path planning procedure for a motion system. In [61] a six-degrees-of-freedom robot manipulator is operated under a visually controlled environment. The bandwidth of the servo loop is optimized by considering the dynamics of the manipulator and closely controlling the speed of motion. A generalized predictive controller based on a cost function, penalizing the error between the current state and the reference state and the control effort over a finite horizon, is proposed.

A slightly different control structure from the regular visual servo loop is presented in [62]. Six cameras surrounding the robot manipulator are used to generate the desired reference signal to the control system, as opposed to being used to generate the feedback signal. Regular encoders allow closed loop control based on the visual reference signal. A timing model is proposed to characterize the time delay associated with the camera dynamics and successful asynchronous camera-control loop operation is demonstrated. Along a similar trend of modified visual servo loops is the one presented in [63]. Visual servo control is used to enhance position accuracy of a preexisting robot manipulator, where coarse positioning occurs through regular encoder-feedback and fine positioning of the manipulator tweezers occurs based on imaging data collected from a digital camera with a microscopic lens.

CHAPTER FOUR

3D ABSOLUTE ENCODING

The current research is implemented on an XY table that allows two dimensional motion. An LCD screen is placed upside-down on top of the table and moves with the table (Figure 4.1 (a)). Although displacement is achieved through ordinary leadscrews connected to servo motors, the position of the stage is determined through a CCD monochrome digital camera that is fixed to the support structure of the stage and aimed at the LCD (Figure 4.1 (b)). The motion of the camera is constrained within the XY plane, i.e. it provides a stationary reference point to measure planar displacement. A third degree of freedom is introduced through a servo-driven rotary camera mount (Figure 4.2 (a)), which allows rotation of the camera (i.e. rotation of the reference frame) along the *Z*-axis. The names "stage/table", "XY stage/table", and "XYθ_Z stage/table" will be used from this point on to refer to the three-degrees-of-freedom servomechanism. The camera, then, acts as an absolute three-degrees-of-freedom displacement transducer. Ideally the point where the rotation axis Z_R intersects the image plane should be regarded as the reference point or origin for in-plane measurements, in order for the system to successfully measure the absolute rotation θ_Z, (Figure 4.2 (b). In other words, the reference point on the image plane should correspond to the principal point, PP. However, it has been demonstrated that for some applications the calculation of the exact location of the principal point is irrelevant, and that an estimate value of its in-plane position can be used for camera modeling [64]. For the algorithms developed in this chapter, the 2D center point of the exposed CCD chip is considered to be the reference or

origin of the image plane. Furthermore, from this point on it will be assumed that the principal point and the center point of the exposed CCD chip are the same point, and therefore the words “principal point” (or its acronym “PP”) and “CCD center point” will be used interchangeably to refer to the reference point used for in-plane measurements. An XYZθ_X adjustable stage (Figure 4.2 (a)), also part of the camera mount, is utilized to fine-tune the position of PP with respect to the rotation axis of the rotary stage such that Z_R is guaranteed to pass through PP; this is denoted as the coaxial requirement.

The main reason for using a monochrome- instead of a color-camera is due to the fact that CCD cameras do not really detect color. The wavelength information (i.e. color information) is actually lost during the photons-to-electrons conversion that takes place when the digital image is created. Color is actually generated through Bayer filtering and intensity interpolation between color transitions. This filtering process inevitably brings some data losses. Given the high (sub-pixel) resolution requirements of the current application, this filtering is considered unacceptable and therefore a monochrome camera is chosen.

(a)

(b)

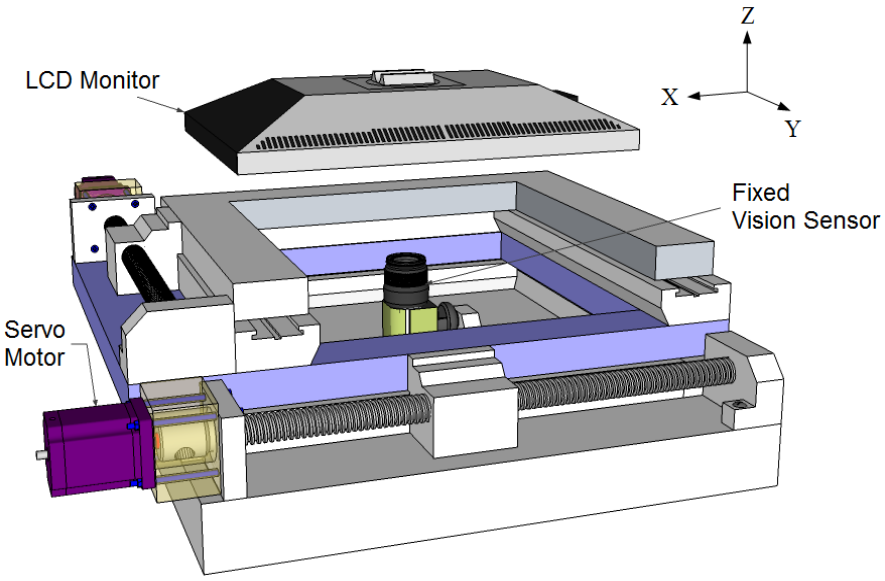


Figure 4.1: (a) 3-axis stage with LCD monitor and (b) 3D CAD model of experimental system.

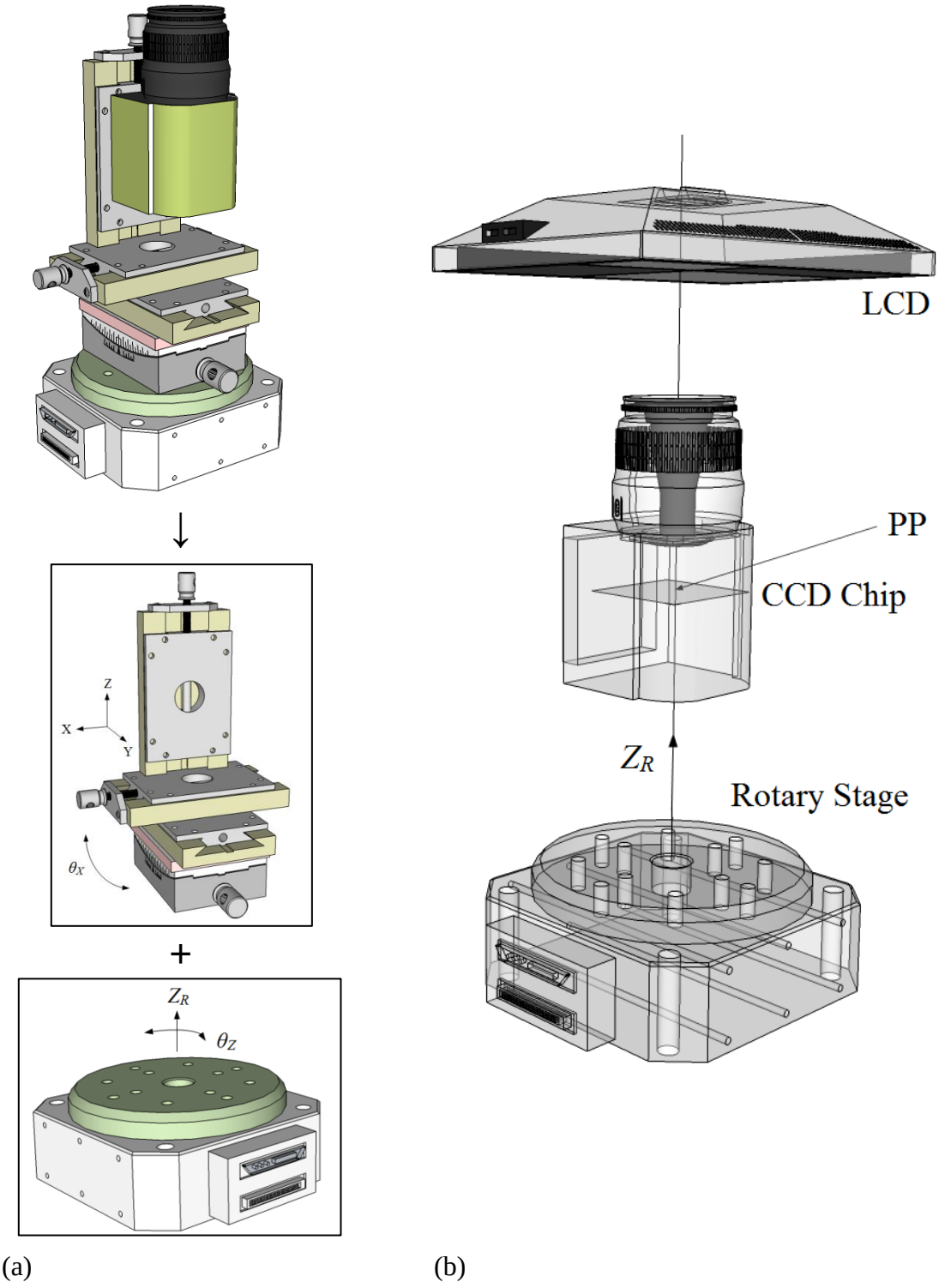


Figure 4.2: (a) Camera mount components: XYZθ_X adjustable stage and rotary stage. (b) Coaxial CCD-rotary stage requirement.

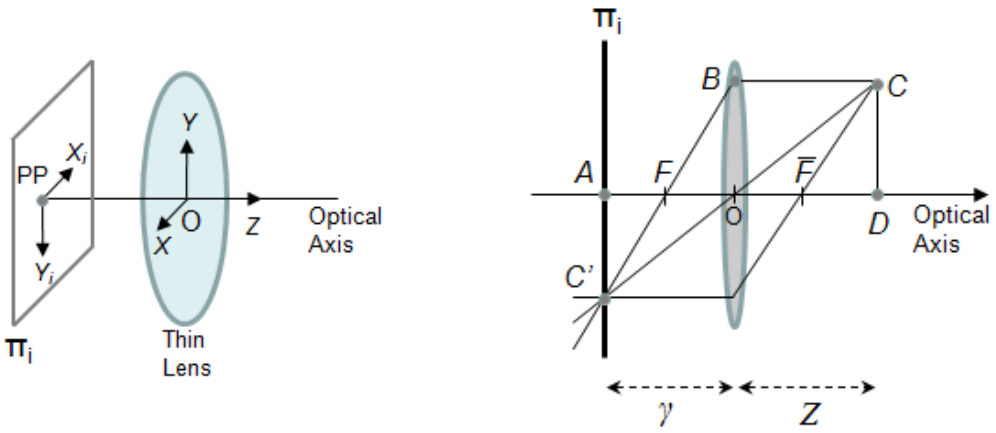
Perhaps the main limitation of the methods introduced later in this chapter is due to distortion and other physical errors associated with the optical elements of the vision system. For this, camera calibration techniques and other modeling procedures might be required. These issues have been addressed many times in previous literature [65-68]. The focus of this chapter is to achieve the desired resolution levels for the proposed platform and, later, use this information to successfully direct-command the stage to perform in-plane displacements in the micron order (Chapter Six). The first section of this chapter - The Optical Sensor and the Dynamic Target - describes some necessary concepts related to vision sensor modeling, and the main features of dynamic targets. The remaining sections of the chapter are dedicated to the development of the image processing algorithms required for MTP implementation.

The Optical Sensor and the Dynamic Target

The Optical Sensor: Perspective Projection

One of the key factors of using a digital camera for position feedback is the challenge of achieving an accurate mapping between real world coordinates and sensor coordinates. This process usually initiates by considering an ideal lens, or thin lens, which leads to two main assumptions: 1. The angles and diameters of the light rays that pass through the lens are sufficiently small and 2. Optical errors and other geometrical aberrations can be neglected. These assumptions are often referred to as the Gauss or paraxial approximation. The physical characteristics of the thin lens are mainly governed by the fact that the distance along the optical axis between the two outer surfaces is negligible with respect to the focal length of the lens. This implies that any ray passing

through the center of the lens is not deviated and all parallel rays (with respect to the optical axis) are deviated as to pass through the focal point of the lens. Lenses where the set of assumptions mentioned before are not valid are called thick lenses and their optical characteristics are different from those for the thin lens.



(a) (b)

Figure 4.3: (a) Optical system consisting of image plane coordinate system and camera coordinate system. (b) The thin lens.

The approximations resulting from the use of the thin lens permit the creation of an imaging model based on simple geometry, namely the well known pinhole model or perspective projection model [69]. Figure 4.3 (a) shows an optical system, where the X_i and Y_i axes represent the image plane Π_i, and (X,Y,Z) represents the camera coordinate system with the Z-axis coinciding with the optical axis. The point where the optical axis intersects the image plane is known as the principal point (PP) of the image plane. For a control point C in Figure 4.3 (b) it is easy to see that

$$\begin{cases} \dfrac{AC'}{DC} = \dfrac{OA}{OD} \\ OB = DC \\ \dfrac{AC'}{OB} = \dfrac{FA}{OF} \end{cases} \tag{4.1}$$

And since $FA=OA-OF$ it is clear that

$$\frac{OA}{OD} = \frac{OA - OF}{OF} \tag{4.2}$$

Now if OA and OD are replaced by the corresponding absolute distances γ and Z and $f = OF$ then, after rearranging, the previous equation can be written as

$$\frac{1}{f} = \frac{1}{Z} + \frac{1}{\gamma} \tag{4.3}$$

Equation (4.3) is the lensmaker's equation and it is used to describe the ideal mapping of an object point. Further derivation of (4.3) yields the typical focal length equation for an experimental vision system (see appendix A). If the analysis is extended to consider a point **C** in space, with camera coordinates (x_{cam},y_{cam},z_{cam}), the corresponding mapping to the image plane is denoted by point **C'** and the following perspective projection equality, in matrix notation, holds

$$\mathbf{C'} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -x_{cam}\dfrac{\gamma}{z_{cam}} \\ -y_{cam}\dfrac{\gamma}{z_{cam}} \\ -\gamma \end{bmatrix} = \begin{bmatrix} -\dfrac{\gamma}{z_{cam}} & 0 & 0 \\ 0 & -\dfrac{\gamma}{z_{cam}} & 0 \\ 0 & 0 & -\gamma \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} \tag{4.4}$$

From (4.3) it should be noted that if $Z \to \infty$ (or $Z >> \gamma$) then $f \to \gamma$, which leads to the commonly used perspective projection model

$$\mathbf{C}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -\frac{f}{z_{cam}} & 0 & 0 \\ 0 & -\frac{f}{z_{cam}} & 0 \\ 0 & 0 & -f \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} \tag{4.5}$$

If the image coordinate frame (X_i, Y_i) is taken as the reference, then point **C** simply maps to $(x'_i, y'_i) = \left(x_{cam} \, f \big/ z_{cam} , y_{cam} \, f \big/ z_{cam} \right)$, as seen in Figure 4.4 (a). The fact that the direction of the X_i and Y_i axes is opposite to the direction of the X and Y axes is found to be a convenient construction, as it avoids the need for negative signs in the coordinate transformation process. Consider now a translation of size $2f$ of the image plane, Π_i, along the Z-axis in the positive direction as shown in Figure 4.4 (b). This model, known as the Front Image Plane or Symmetry Interpretation, is widely used in computer vision as it provides a straight forward mapping of the scene being imaged, where no inversion of coordinate axes is required [70]. It is important to observe that the relationships given by (4.4) and (4.5) still hold for this approach. From this point on any analysis concerning the vision system will be conducted based on the Front Image Plane model. Moreover, for simplicity purposes, the Front Image Plane model will be referred to as the pinhole camera model or the perspective projection model.

In general a point in space with world coordinates (x'_w, y'_w, z'_w) undergoes a transformation, from world coordinates to camera coordinates, of the form

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} = \mathbf{R} \begin{bmatrix} x'_w \\ y'_w \\ z'_w \end{bmatrix} + \mathbf{T} \tag{4.6}$$

where the rotation matrix $\mathbf{R} \in \mathbb{R}^{3x3}$, and the translation vector $\mathbf{T} \in \mathbb{R}^{3x1}$ are commonly known as the extrinsic parameters of the vision system (Figure 4.4 (c)). Assuming that both the CCD chip and the LCD are perpendicular with respect to the optical axis, it is possible to define the location of a point on the LCD, with LCD-coordinates (x_{LCD}, y_{LCD}), referenced to the image plane through

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = M \left(\mathbf{R}_{\theta \mathbf{Z}} \begin{bmatrix} x'_{LCD} \\ y'_{LCD} \end{bmatrix} + \mathbf{T}_{\mathbf{LCD}} \right) \tag{4.7}$$

where $M = f / z'_{LCD}$ is the optical magnification, $\mathbf{R}_{\theta \mathbf{Z}} \in \mathbb{R}^{2x2}$ is the rotation matrix, and $\mathbf{T}_{\mathbf{LCD}} \in \mathbb{R}^{2x1}$ is the translation vector (Figure 4.4 (d)).

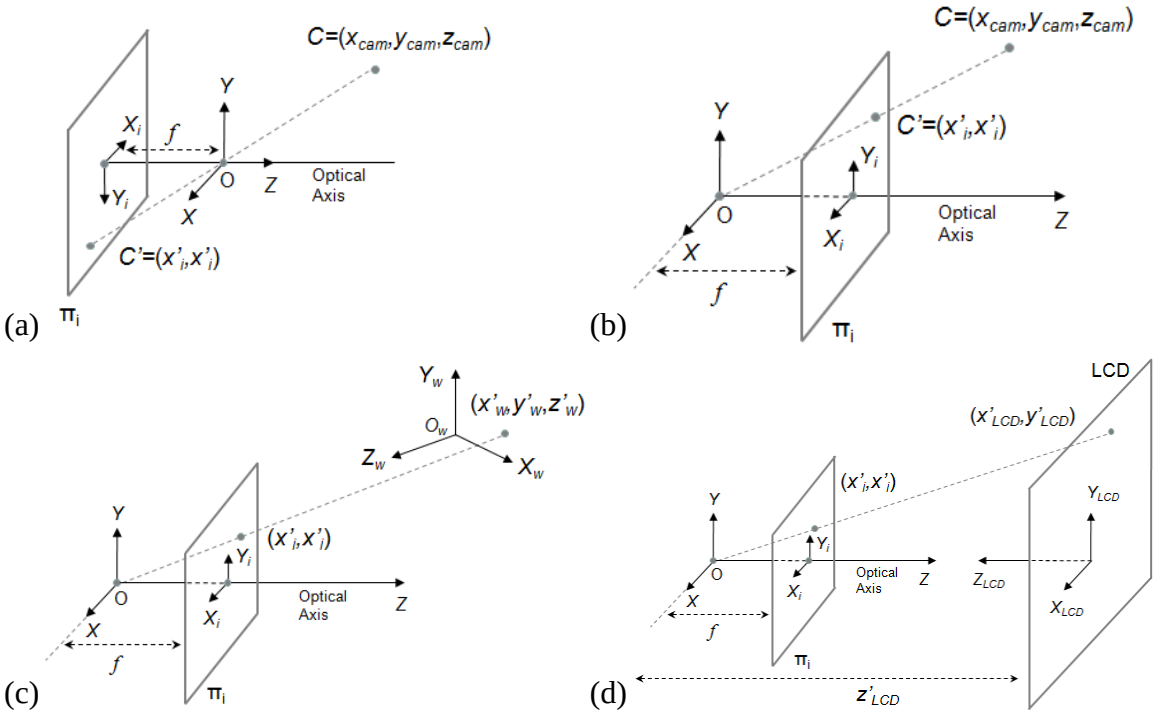


Figure 4.4: (a) Perspective projection model. (b) Front Image Plane model. (c) World-to-camera coordinate transformation. (d) LCD-to-image plane coordinate transformation.

The Dynamic Target

In the stage setup described before, motion commands are provided through a moving active target displayed on an LCD screen. The position of the target on the LCD is controlled by the user through software. If the principal point PP on the CCD (Figure 4.5) is regarded as the reference, then an in-plane position error vector can be defined at any time *t*, with respect to this reference. For instance, if the target in Figure 4.5 was purposely located at point **C** on the LCD, then the corresponding error vector would be ***e*(*t*)** = **C'** on the CCD. This vector can be used as the control loop position error, where the controller's goal is to bring the target to the reference position, i.e. PP, and control its orientation in the motion plane.

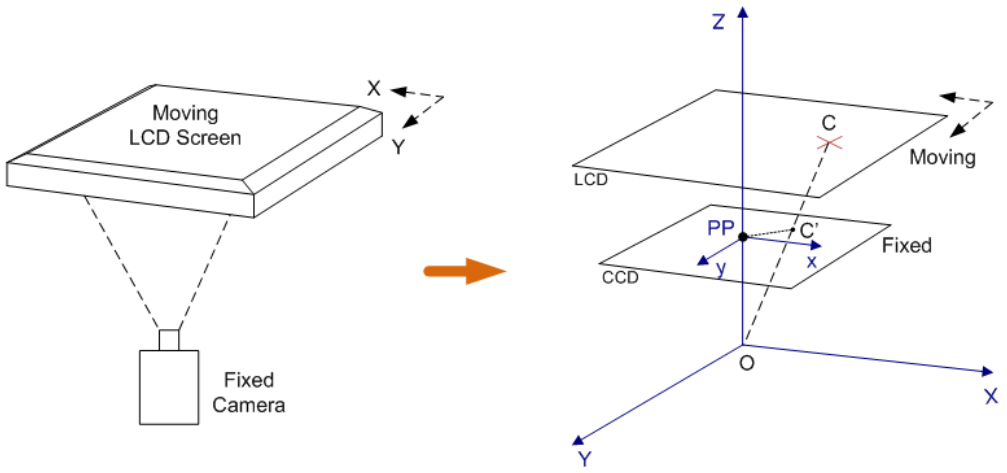


Figure 4.5: Tracking error based on target location and orientation.

It is important at this point to make a distinction between LCD pixels and CCD pixels. As it was stated at the beginning of this chapter, CCD pixels only perceive light intensities reflected from a given scene while color information is lost. This does not

mean that color patterns should not be used to form the target on the screen. Figure 4.6 shows one LCD pixel of size 294x294µm when inspected under a microscope with a 90x magnification. The possibility to individually generate blue, green or red from a single LCD pixel gives the user control not only over the intensity pattern of that pixel, but also over the physical (horizontal) location of this intensity to within 1/3 of the pixel. The intensity pattern on the LCD screen then maps to a grayscale image on the CCD.

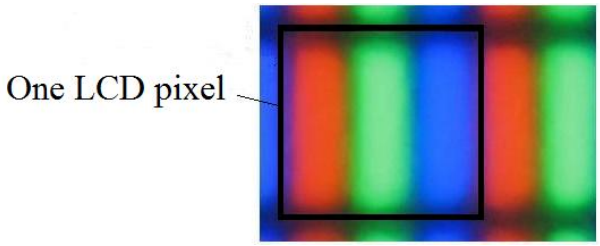


Figure 4.6: 90x magnification of one LCD pixel.

MTP – Method 1: Cross-Hairs

Given some a priori information, the position of a known target is determined and tracked over time. The target is represented by the intersection of two perpendicular lines, i.e. a cross-hairs target. The procedure for locating the target, once captured as a digital image, is described in three steps: 1. Fine point location and definition of the line sets; 2. Constrained curve fitting through Newton-Raphson method; and 3. Estimation of pixel size and optical magnification using four reference elements (the image processing application covering all three steps was programmed in C language; the complete code can be found in appendix B). Before these three steps are analyzed, the criteria for selecting the intensity format for the cross-hairs target on the LCD and its specific geometry are presented.

The Intensity Format and Target Geometry on LCD

The initial task consists in identifying the optimal combination of pixel intensities for the generation of the target image. On an 8-bit display, pixel intensities vary from 0 (lowest, black) to 255 (highest: R, G or B). Based on this intensity range two types of formats are tested: white target (R=255, G=255 and B=255) over black background and black target over white background (Figure 4.7). Experimental data is collected using bitmap (BMP) images, in order to minimize information losses due to image compression techniques. The grayscale intensity in the 3D plot in Figure 4.7 (b) is inverted (e.g. pixels with values of 255 are set to 0 and pixels with values of 0 are set to 255) for better visualization of the image contrast. In general, the contrast that a given imaging system is

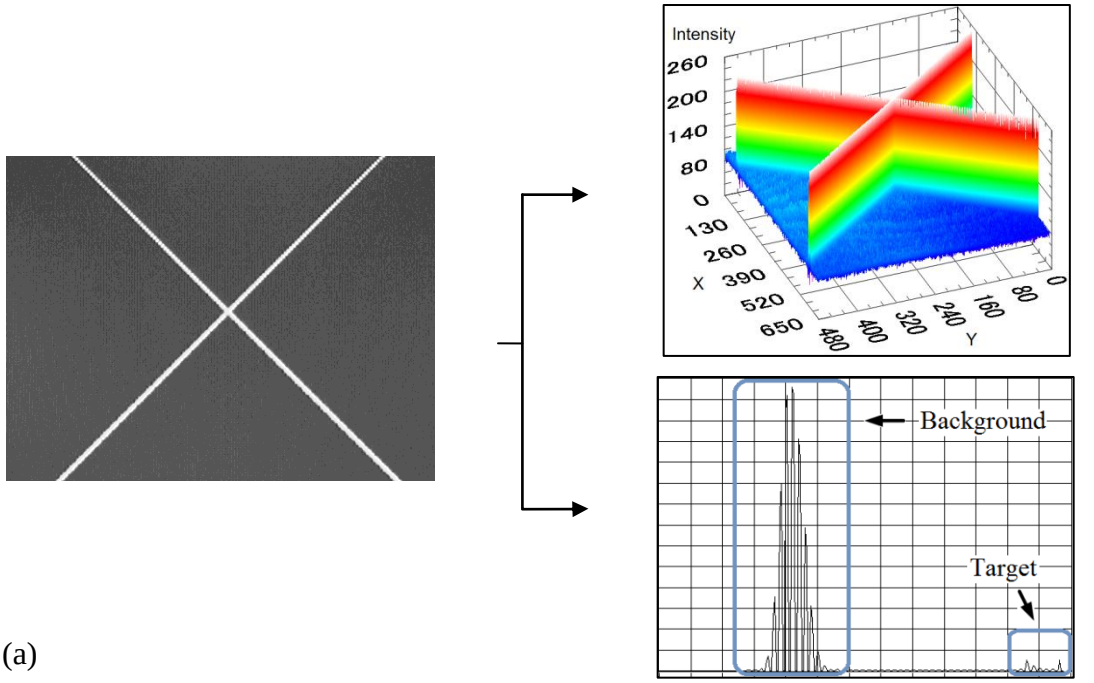
capable of reproducing on a specific image can be calculated based on percentages through [71]

$$Contrast = \frac{\left(I_{\max} - I_{\min}\right)}{\left(I_{\max} + I_{\min}\right)} \cdot 100\% \tag{4.8}$$

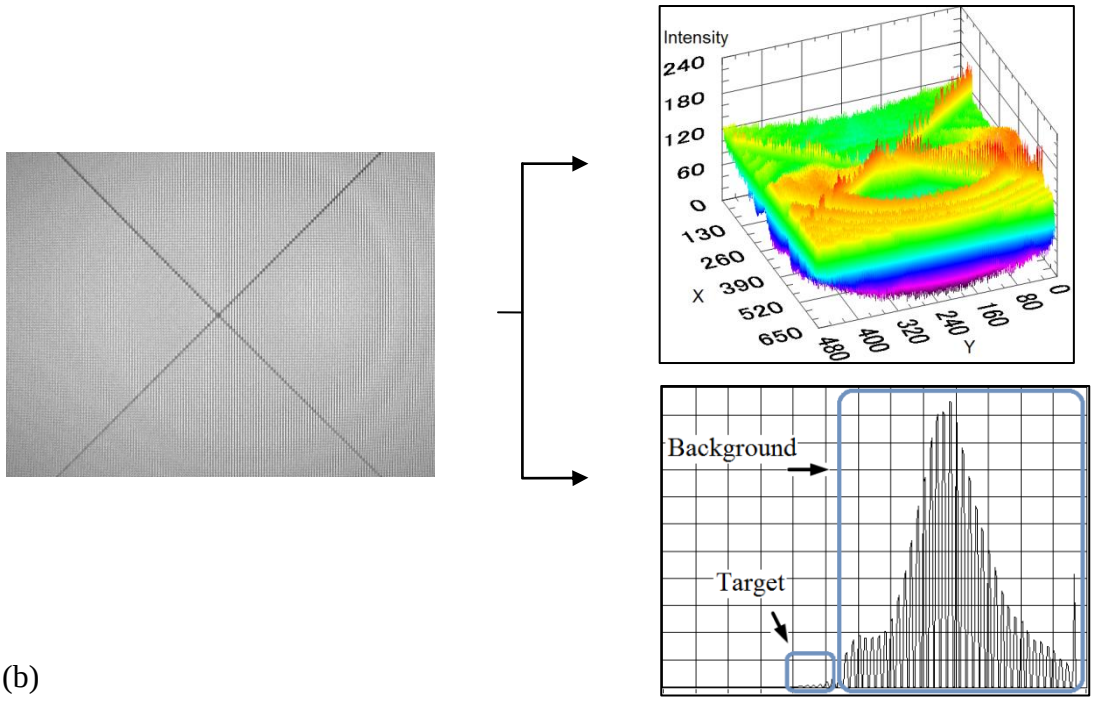
The contrast percentages obtained for the two formats, summarized in Table 4.1, indicate that more information can be extracted from the white target over black background. The histograms in Figure 4.7 also suggest an easier identification of the white target from the image background, when compared to the black target. Hence, the white target is selected for experimental tests.

Table 4.1: Contrast percentages for target formats.

Format	**Black Target**	**White Target**
Contrast	65.4%	84.7%



(a)



(b)

Figure 4.7: Target formats: (a) White target over black background with corresponding 3D intensity plot and histogram; (b) Black target over white background with corresponding 3D inverted-intensity plot and histogram.

Considering the construction of an LCD pixel, it is desired to take advantage of the possibility of arbitrarily locating and moving a given intensity to within 1/3 of the pixel along the LCD *X*-axis. Figure 4.8 shows a cross-hairs target, where the vertical line is generated by lighting up a single stripe (R, G or B) of one column of pixels on the LCD. In general, pixel intensities on the vertical line can be purposely adjusted to display known patterns, e.g. a discrete Gaussian curve. This capability, however, does not apply to the horizontal line of the target, which has a width of one full LCD pixel. In conclusion, the user can move the target by increments as small as 1/3 of a pixel along the LCD *X*-axis, and by increments as small as one pixel along the LCD *Y*-axis by simply illuminating the next individually addressable element on each axis. From a control perspective, this aspect clearly limits the capability of commanding horizontal displacements smaller than 1/3 of a pixel, and vertical displacements smaller than one whole pixel. This issue is addressed in Chapter Six, where two methods for commanding planar displacements on the micron order are presented.

Aside from the cross-hairs, the target image also includes four reference elements displayed close to the intersection. The location of the reference elements is symmetric with respect to the target's vertical line, and asymmetric with respect to the horizontal line. This layout allows the computation of the absolute target orientation θ_Z within 360°, i.e. the calculation of $\theta_Z \; mod(2\pi)$. The reference elements are also used to compute the optical magnification of the imaging system in step 3.

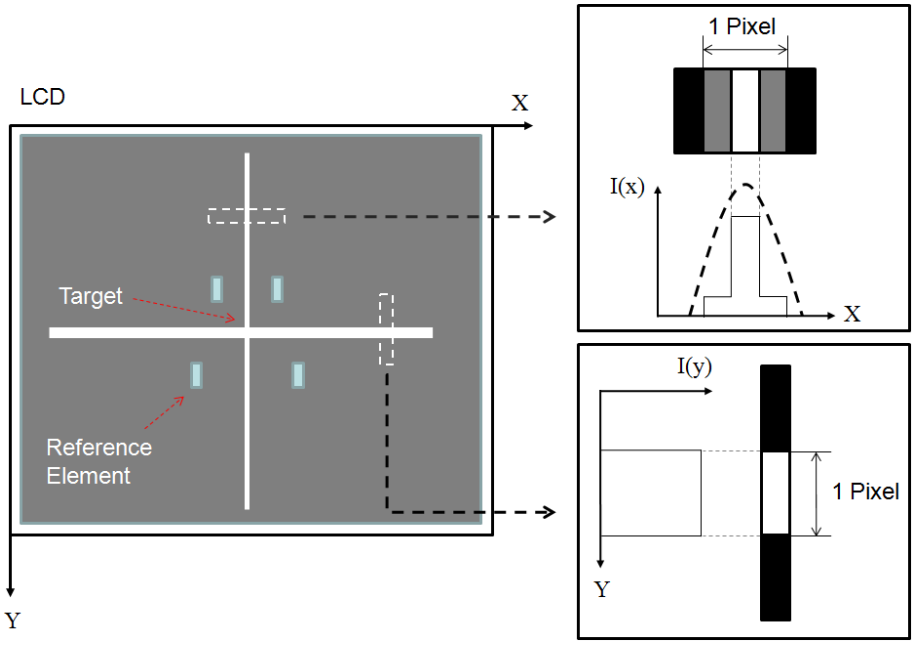


Figure 4.8: Pixelated dynamic target on LCD with four Reference elements.

Cross-Hairs Target Identification through Newton-Raphson Method

The procedure for locating the target, once captured as a digital image though the CCD monochrome camera, is described in three steps.

Step 1: Fine point location and definition of the line sets

It is desired to perform sub-pixel resolution measurements on the digital image, by defining a feature that combines both the grayscale intensity of the target and its location within the image plane. A practical solution arises by computing the intensity weighted center of mass, or centroid [72]. For a continuous three dimensional real function, $z_I = f(x, y)$, with $x \in [a,b], a,b \in \mathbb{R}$, and y fixed, i.e. $y=i$, $i \in \mathbb{R}$, the centroid along the X-axis, denoted by $X_{C,i}$, can be obtained from

$$X_{C,i} = \frac{\int_a^b xf(x,i)\,dx}{\int_a^b f(x,i)\,dx} \tag{4.9}$$

For a discrete-valued function such as an *m*-by-*m* image, $I_{mxm}(x,y)$, where *x* varies in a discrete manner over a horizontal array of pixels $x \in [0, m-1]$, and *y* is fixed to the current row under analysis e.g. *y=i*, the center of mass is

$$X_{C,i} = \frac{\sum_{x=0}^{m-1} xI(x,i)}{\sum_{x=0}^{m-1} I(x,i)} \tag{4.10}$$

Equation (4.10) renders a procedure for conducting sub-pixel resolution measurements along a given row of pixels (*X*-axis) in the image plane. An analogous equation can be obtain for performing sub-pixel resolution measurements along a given column of pixels (*Y*-axis), by simply fixing *x* to the column under analysis and letting *y* vary over a vertical window of pixels.

Sub-pixel resolution on the target image is achieved by calculating the intensity weighted centroid, using (4.10), around an intensity transition. An intensity transition of interest is defined as a vertical or horizontal transition that starts in a black region, goes through a saturation point at 255 and finally goes back to a black region. Figure 4.9 shows a target image, where the target is represented by two black (instead of white) perpendicular lines over a white background for demonstration purposes. Two transitions of interest are highlighted in this image: one horizontal and one vertical. If a horizontal transition is found in a given row of pixels, the *x*-coordinate of the point to be stored in memory will correspond to the horizontal centroid around this transition. The *y*-

coordinate assigned to this point will just be the discrete value of the row under analysis. For the case of vertical intensity transitions, the *y*-coordinate of the point will correspond to the vertical centroid around this transition; the *x*-coordinate of this same point will just be the discrete value of the column under analysis.

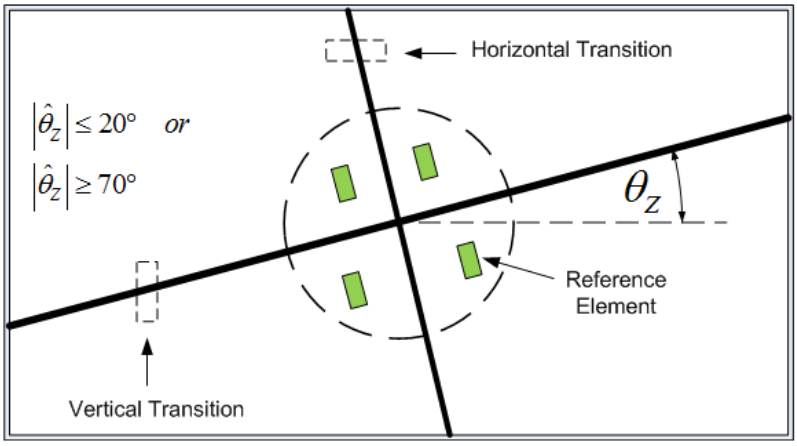


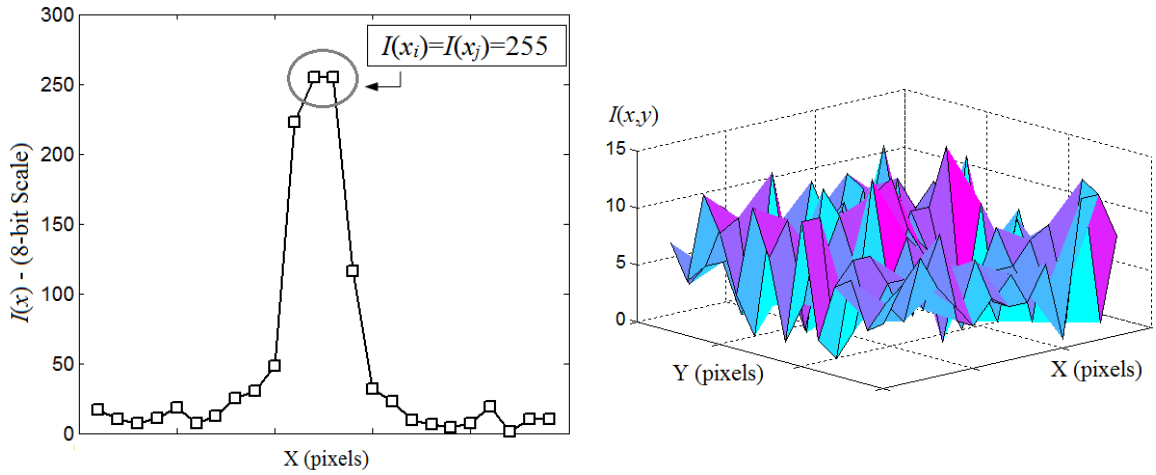
Figure 4.9: Target image with inverted grayscale intensities for demonstration purposes.

Remark 4.1: Noise filtering in black regions

Figure 4.10 (a) presents a horizontal intensity transition, taken from an experimental sample. It is important to notice that pixel intensities of 255 are easily detected on the CCD, when imaging white LCD pixels. Furthermore, regions containing pixel intensities of 255 seem to be less affected by noise. Then the key question to be answered is: how big should the pixel window be for the calculation of a horizontal (or vertical) centroid around an intensity transition?

A horizontal pixel-window should start (going from left to right) at a low-intensity pixel, go through one or more saturation pixels and end at a black pixel. As seen, high pixel intensities are easily identified by the sensor. Then, attention must be paid to the processing of black regions. Figure 4.10 (b) shows a black area as detected by the camera. The information content varying at high frequencies within a range of intensities from 0 to 15 is clear evidence of the presence of noise. Then, if the values in such an area were considered when computing the centroid, the precision of the measurement would be compromised. A solution arises by setting a low-

intensity threshold slightly above the maximum intensity of the noisy pattern shown in Figure 4.10 (b), and only including those neighboring pixels around the intensity peak with intensities above this threshold for the calculation of the centroid.



(a) (b)

Figure 4.10: Experimental data captured using a CCD camera: (a) Horizontal pixel window covering intensity transition around target pixels, and (b) 3D sample of a black (low-intensity) region.

By following the procedure explained before, a set of experimental points associated with the two perpendicular lines in the target image can be collected. It is even possible to utilize a subset of all these experimental points, that is only an arbitrary small number of points computed at the beginning of the routine, to estimate the intersection of the cross-hairs before the entire image has been analyzed. Based on the initial estimate, data points can classified according to their proximity to the intersection point. Transitions close to the cross-hairs intersection estimate (inside the dashed circle in Figure 4.9) are disregarded, as they require additional processing and might even introduce errors into the measurements, if not handled appropriately. The intersection

estimate also helps locate the reference elements, which are purposely placed close to the real intersection.

In general, an algorithm could search for both vertical and horizontal transitions at all times. This would be a sub-optimal approach, though, as it would increase processing times. If, for example, $\theta_Z = 45°$, in Figure 4.9 then the image processing algorithm could focus on searching horizontal transitions only, and not waste time in searching for vertical transitions. However, it is impossible to know the value of θ_Z before the target has been fully identified, i.e. before all the transition points have been found. Similarly as with the intersection estimate, this dilemma is solved by assuming that the first K transitions, associated with the first K points found in the digital image correspond to horizontal transitions. Based on these first points, an estimated value of θ_Z, namely $\hat{\theta}_Z$, can be computed. If $\left|\hat{\theta}_Z\right| \leq 20°$ or $\left|\hat{\theta}_Z\right| \geq 70°$, for $\theta_Z \in [-90°, 90°]$, then the image processing algorithm searches for both horizontal and vertical intensity transitions. Only horizontal transitions are examined, otherwise. The threshold value of 20° is selected based on trial and error tests using experimental data. Once the transition points are calculated they are separated in two sets, *DS*1 and *DS*2, associated to two different lines.

Step 2: Constrained curve fitting through Newton-Raphson method

The goal is to find the location of the target through the calculation of two straight lines that best fit the data collected in the previous step. The intersection of these lines in the image plane is regarded as the target location. The two data sets *DS*1 and *DS*2 are known to contain n_1 and n_2 data points, respectively. The best fit lines are constrained to

be perpendicular with respect to each other. The previous statement is equivalent to finding $\boldsymbol{y}^{model}$, where

$$\mathbf{y}^{\mathbf{model}} = \begin{cases} a_0 + a_1 x, & a_0, a_1 \in \mathbb{R} \text{ and } x \in DS1 \\ a_2 - \dfrac{1}{a_1} x, & a_2 \in \mathbb{R} \text{ and } x \in DS2 \end{cases} \tag{4.11}$$

such that

$$S(a_0, a_1, a_2) = \sum_{data} \left(\mathbf{y}_i - \mathbf{y}^{\mathbf{model}} \right)^2, \quad \mathbf{y} \in \{DS1, DS2\} \tag{4.12}$$

is minimized. Derivation of (4.12) yields

$$\begin{aligned} S(a_0, a_1, a_2) &= \sum_{DS1} \left(y_i - y^{model} \right)^2 + \sum_{DS2} \left(y_i - y^{model} \right)^2 \\ &= \sum_{DS1} \left(y_i - a_0 - a_1 x_i \right)^2 + \sum_{DS2} \left(y_i - a_2 + \frac{1}{a_1} x_i \right)^2 \end{aligned} \tag{4.13}$$

The minimum is obtained by finding a_0, a_1, and a_2 such that $\partial S / \partial a_0 = 0$, $\partial S / \partial a_1 = 0$ and $\partial S / \partial a_2 = 0$. The first partial derivative is calculated as follows

$$\frac{\partial S}{\partial a_0} = \sum_{DS1} 2\left(y_i - a_0 - a_1 x_i \right)(-1) + \sum_{DS2} 0 = 0 \tag{4.14}$$

which yields the linear equation

$$n_1 a_0 + a_1 \sum_{DS1} x_i = \sum_{DS1} y_i \tag{4.15}$$

and since

$$\frac{\partial^2 S}{\partial a_0^2} = n_1 > 0 \tag{4.16}$$

it is clear that the values of a_0 and a_1 satisfying (4.15) indeed correspond to a minimum.

Similarly, a second nonlinear equation is obtained from $\partial S / \partial a_2 = 0$, namely

$$n_2 a_2 - \frac{1}{a_1}\sum_{DS2} x_i = \sum_{DS2} y_i \tag{4.17}$$

Again, it is easy to see that the values of a_1 and a_2 satisfying (4.17) correspond to a minimum since

$$\frac{\partial^2 S}{\partial a_2^2} = n_2 > 0 \tag{4.18}$$

A third equation is obtained from $\partial S / \partial a_1 = 0$, i.e.

$$\frac{\partial S}{\partial a_1} = \sum_{DS1} 2\left(y_i - a_0 - a_1 x_i\right)\left(-x_i\right) + \sum_{DS2} 2\left(y_i - a_2 + \frac{1}{a_1}x_i\right)\left(-\frac{x_i}{a_1^2}\right) = 0 \tag{4.19}$$

which, after rearranging, yields the final nonlinear equation

$$a_0\left(\sum_{DS1} x_i\right) + a_1\left(\sum_{DS1} x_i^2\right) - \left(\sum_{DS1} x_i y_i\right) + \frac{a_2}{a_1^2}\left(\sum_{DS2} x_i\right) - \frac{1}{a_1^3}\left(\sum_{DS2} x_i^2\right) - \frac{1}{a_1^2}\left(\sum_{DS2} x_i y_i\right) = 0 \tag{4.20}$$

Now,

$$\frac{\partial^2 S}{\partial a_1^2} = \left(\sum_{DS1} x_i^2\right) - 2\frac{a_2}{a_1^3}\left(\sum_{DS2} x_i\right) + 3\frac{1}{a_1^4}\left(\sum_{DS2} x_i^2\right) + 2\frac{1}{a_1^3}\left(\sum_{DS2} x_i y_i\right) \tag{4.21}$$

and therefore a minimum occurs as long as

$$\left(\sum_{DS1} x_i^2\right) + 3\frac{1}{a_1^4}\left(\sum_{DS2} x_i^2\right) + 2\frac{1}{a_1^3}\left(\sum_{DS2} x_i y_i\right) > 2\frac{a_2}{a_1^3}\left(\sum_{DS2} x_i\right) \tag{4.22}$$

Equations (4.15), (4.17) and (4.20) form a set of nonlinear equations that must be solved in order to fully determine the model given in (4.11). By defining the constant values for *DS*1 and *DS*2 as

$$\begin{cases} v_1 = \sum_{DS1} x_i \\ v_2 = \sum_{DS1} y_i \\ v_3 = \sum_{DS1} x_i y_i \\ v_4 = \sum_{DS1} x_i^2 \end{cases}; \quad \begin{cases} w_1 = \sum_{DS2} x_i \\ w_2 = \sum_{DS2} y_i \\ w_3 = \sum_{DS2} x_i y_i \\ w_4 = \sum_{DS2} x_i^2 \end{cases} \tag{4.23}$$

from (4.15) and (4.17), respectively, it is clear that

$$a_0 = \frac{1}{n_1}\left(v_2 - a_1 v_1\right) \tag{4.24}$$

$$a_2 = \frac{1}{n_2}\left(w_2 + \frac{1}{a_1} w_1\right) \tag{4.25}$$

Inserting (4.24) and (4.25) into (4.20) and rearranging yields an equation of the form $f(a_1)=0$, where f is assumed to have a continuous first derivative f'. This is

$$f(a_1) = \frac{v_1 v_2}{n_1} - v_3 + a_1\left(v_4 - \frac{v_1^2}{n_1}\right) + \frac{1}{a_1^2}\left(\frac{w_1 w_2}{n_2} - w_3\right) + \frac{1}{a_1^3}\left(\frac{w_1^2}{n_2} - w_4\right) = 0 \tag{4.26}$$

and therefore

$$f' = \frac{df(a_1)}{da_1} = v_4 - \frac{v_1^2}{n_1} + \frac{2}{a_1^3}\left(w_3 - \frac{w_1 w_2}{n_2}\right) + \frac{3}{a_1^4}\left(w_4 - \frac{w_1^2}{n_2}\right) \tag{4.27}$$

Equation (4.26) is solved using Newton-Raphson iterative method, which implies calculating

$$a_1(n+1) = a_1(n) - \frac{f(a_1(n))}{f'(a_1(n))}, \quad n = 1, 2, ..., k \tag{4.28}$$

until

$$\left|a_1(n+1) - a_1(n)\right| \le \varepsilon \left|a_1(n)\right| \tag{4.29}$$

where $a_1(n)$ represents a constant value given to the variable a_1, k is the total number of iterations, ε is an arbitrary threshold and the factor $|a_1(n)|$ in (4.29) is required in case of zeros of very large or very small absolute value [73]. Although the criterion by itself does not imply convergence, in the current problem it is assumed that as $|a_1(n+1)-a_1(n)| \to 0$ the system converges.

The following pseudocode for calculating the best fit perpendicular-lines and determining the position of the target in the *XY*-plane is proposed. Pseudocode:

1. Find the initial generic lines $\hat{y}_1 = m_1 x + b_1$ and $\hat{y}_2 = m_2 x + b_2$ through linear regression applied independently to *DS*1 and *DS*2, respectively.
2. Set $a_1(0)=m_1$ and define the variables *slope_a*$_1$ and *Status*="No Success"
3. ***For j*=1 until *j*=*k*** (*k* is determined experimentally)
   Compute $f'(a(j-1))$
   ***If* $f'(a(j-1))=0$**
   Status="No Success"
   Terminate For Loop
   End of *If*
   Else
   Compute: $a_1(j) = a_1(j-1) - \dfrac{f(a_1(j-1))}{f'(a_1(j-1))}$
   If $|a_1(j) - a_1(j-1)| \le \varepsilon |a_1(j-1)|$
   slope_a$_1$=$a_1(j)$
   Status="Success"
   Terminate For Loop
   End of *If*
   End of *Else*
   End of *For*
4. **If *Status*="No Success"**
   Find the target location through the intersection of $\hat{y}_1$ and $\hat{y}_2$
   End of *If*
   Else
   Set a_1= *slope_a*$_1$ in (4.24) and (4.25) to find a_0 and a_2, respectively. Then use these values to fully determine the model given in (4.11). Finally, calculate the position of the target as the intersection of the lines defined by (4.11).
   End of *Else*

A relevant characteristic of the previous algorithm is that constants v_1-v_4 and w_1-w_4 can also be used to compute the generic lines $\hat{y}_1$ and $\hat{y}_2$ and therefore no significant extra computational effort is required for this initial step. If convergence has not been reached after k iterations in point 3, the algorithm still guarantees the calculation of the target location using $\hat{y}_1$ and $\hat{y}_2$ as shown in the first line of point 4 in the pseudocode. This, of course, implies a small sacrifice in the accuracy of the estimation.

Finally, the relative target orientation given in radians for a range of $[-\pi/2,\pi/2]$, is defined as

$$\tilde{\theta}_Z = \tan^{-1}\left(\frac{1}{a_1}\right) \tag{4.30}$$

The absolute target orientation, θ_Z, for a range of $[0,2\pi]$ is determined based on the location of the four reference elements and the value of $\tilde{\theta}_Z$.

Step 3: Estimation of pixel size and optical magnification through reference elements

An estimate value of the optical magnification is obtained by computing the distances between the four reference elements on the CCD and comparing it with the known LCD spacing between these points. The locations of the reference elements on the target image are calculated with sub-pixel resolution using (4.10) over the areas containing these elements. This calculation yields the coordinates of the four points on the image plane: **P1=**$[x_1,y_1]^T$, **P2=**$[x_2,y_2]^T$, **P3=**$[x_3,y_3]^T$ and **P4=**$[x_4,y_4]^T$ **(**Figure 4.11**)**. These coordinates are mapped to LCD coordinates through a two-step transformation process. The first step is characterized by

$$\begin{cases} \hat{\mathbf{P}}_1 = S\mathbf{R}_{\theta Z}\mathbf{P}_1 = [\hat{x}_1, \hat{y}_1]^T \\ \hat{\mathbf{P}}_2 = S\mathbf{R}_{\theta Z}\mathbf{P}_2 = [\hat{x}_2, \hat{y}_2]^T \\ \hat{\mathbf{P}}_3 = S\mathbf{R}_{\theta Z}\mathbf{P}_3 = [\hat{x}_3, \hat{y}_3]^T \\ \hat{\mathbf{P}}_4 = S\mathbf{R}_{\theta Z}\mathbf{P}_4 = [\hat{x}_4, \hat{y}_4]^T \end{cases} \tag{4.31}$$

The scaling factor *S*, in (4.31), is used to convert the coordinates from pixel- to µm-units, assuming square CCD pixels of size *S*x*S* µm. If the CCD pixels are not square, *S* must be replaced by a 2-by-2 diagonal matrix with entries equal to the length and height of a CCD pixel. The rotation matrix, $\mathbf{R}_{\theta Z}$, is necessary since the distances on the CCD and LCD are compared at the vector level, i.e. absolute Euclidian distances are not used. The second step of the transformation process is simply the multiplication of the points $\hat{\mathbf{P}}\mathbf{1} - \hat{\mathbf{P}}\mathbf{4}$ by the inverse of the optical magnification, defined as $M = f / z'_{LCD}$, where *f* is the focal length and z'_{LCD} is the out-of-plane distance between the lens and the LCD. The inverse of *M* is utilized in this case since the LCD is taken as the reference frame. The distances between the reference elements on the LCD, given in LCD pixel-units, are known to be $\Delta X1_{LCD}$, $\Delta X2_{LCD}$, $\Delta Y1_{LCD}$, $\Delta Y2_{LCD}$ with $\Delta Y1_{LCD}=\Delta Y2_{LCD}$. The LCD pixel size, in µm, is *H*x*V*. The following set of equations results from the perspective projection model

$$\begin{cases} (\hat{x}_2 - \hat{x}_1) M^{-1} = \Delta X1_{LCD} H \\ (\hat{x}_4 - \hat{x}_3) M^{-1} = \Delta X2_{LCD} H \\ (\hat{y}_3 - \hat{y}_1) M^{-1} = \Delta Y1_{LCD} V \\ (\hat{y}_4 - \hat{y}_2) M^{-1} = \Delta Y1_{LCD} V \end{cases} \tag{4.32}$$

It should be observed that the two-step transformation process does not require the translation vector introduced in (4.7). The reason for this is due to the fact that (4.32)

describes a relationship between CCD distances (left-hand side of the set of equalities) and LCD distances (right-hand side of the set of equalities), and not between coordinates.

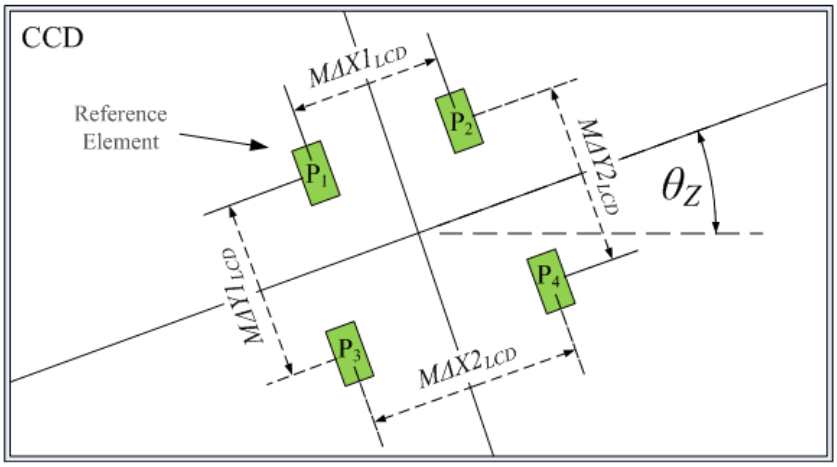


Figure 4.11: System magnification calculated based on reference elements.

Equation (4.32) can be written in the matrix form as

$$\begin{bmatrix} \Delta\hat{x}_1 & -\Delta X1_{LCD} & 0 \\ \Delta\hat{x}_2 & -\Delta X2_{LCD} & 0 \\ \Delta\hat{y}_1 & 0 & -\Delta Y1_{LCD} \\ \Delta\hat{y}_2 & 0 & -\Delta Y1_{LCD} \end{bmatrix} \begin{bmatrix} M^{-1} \\ H \\ V \end{bmatrix} = \hat{\mathbf{A}}\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.33}$$

where **u** is the vector of unknown parameters and $\hat{\mathbf{A}}$ is the matrix containing the known- and measured-distances; the latter are redefined as $\Delta\hat{x}_1 = (\hat{x}_2 - \hat{x}_1), \Delta\hat{x}_2 = (\hat{x}_4 - \hat{x}_3), \Delta\hat{y}_1 = (\hat{y}_3 - \hat{y}_1)$ and $\Delta\hat{y}_2 = (\hat{y}_4 - \hat{y}_2)$. Analytically, it is possible to reduce $\hat{\mathbf{A}}$ to at least $\mathbf{A} \in \mathbb{R}^{3x3}$. In general, if matrix **A** is full rank, then there are no non-trivial solutions to the system of equations. Considering that the values $\Delta\hat{x}_1, \Delta\hat{x}_2, \Delta\hat{y}_1$ and $\Delta\hat{y}_2$ are affected by noise in the measuring process, it should come as

no surprise that rank(**A**)=3. One way to approach this problem is by sacrificing one of the unknown parameters, and giving it a fixed value. Then, the other two unknown parameters can be calculated through a regular optimization method. In this case the horizontal length of an LCD pixel, *H*, commonly known as dot-pitch, is acquired from the LCD manufacturer and the following quadratic function is minimized

$$f(\mathbf{v}) = \left\| \mathbf{w} + \mathbf{B}\mathbf{v} \right\|^2 \tag{4.34}$$

where

$$\mathbf{w} = \begin{bmatrix} -\Delta X1_{LCD} \\ -\Delta X2_{LCD} \\ 0 \\ 0 \end{bmatrix} H; \quad \mathbf{B} = \begin{bmatrix} \Delta\hat{x}_1 & 0 \\ \Delta\hat{x}_2 & 0 \\ \Delta\hat{y}_1 & -\Delta Y1_{LCD} \\ \Delta\hat{y}_2 & -\Delta Y1_{LCD} \end{bmatrix}; \quad \mathbf{v} = \begin{bmatrix} M^{-1} \\ V \end{bmatrix} \tag{4.35}$$

The minimum for the function in (4.34) is almost the same as the solution of the regular least-squares problem, with the only difference being a negative sign preceding the right-hand side of the equation. This is,

$$\mathbf{v} = -\left(\mathbf{B}^{\mathbf{T}}\mathbf{B}\right)^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{w} \tag{4.36}$$

Experimental Results

The ability of the image processing algorithm to detect displacements of the target along the *XY*-plane is tested. The test consists in moving the cross-hairs (Figure 4.12) on the LCD screen by discrete increments of sizes 1/3 of an LCD pixel along the LCD *X*-axis, and one full LCD pixel along the LCD *Y*-axis, while maintaining the stage still. The target orientation is set to a value as close to zero as possible, such that horizontal

displacements along the LCD (LCD *X*-axis) also map to horizontal displacements along the image plane. The same applies to the vertical displacement test.

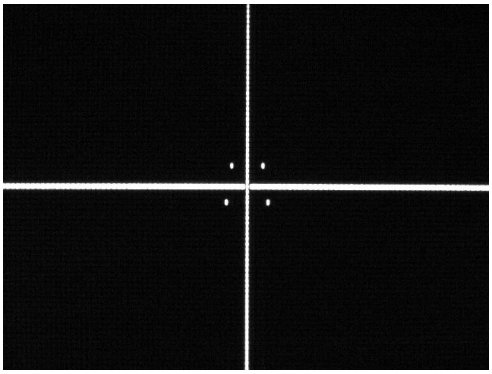


Figure 4.12: Experimental cross-hairs target.

Experimental results are listed in Table 4.2 and Table 4.3, rounded to two decimal places. At each step a 100 image sample is collected and the location of the target on each image is computed with respect to the reference point (PP). The sample mean is regarded as the location of the target at each step. The first row lists the known displacement of the target on the LCD in LCD pixels. The unit of length for the known displacement in Table 4.2, second row, is introduced by considering the known LCD dot pitch (horizontal length of the pixel, *H*), taken from the manufacturer specifications, which in this case is known to be 294 μm. In Table 4.3, second row, the unit of length is introduced through the vertical length of one the LCD pixel, *V*, calculated through step 3. The experimental value, averaged from all samples, is found to be $V = 293.81$ μm and is considered to be the actual increment size for displacements along the *Y*-axis.

The last two rows in Table 4.2 and Table 4.3 (labeled as CCD Sensor) list the results of the measurements as observed by the vision sensor, after applying Method 1 to

the sample images. The direct output from the image processing algorithm is given in CCD pixels, and is shown in the third row of these tables. These values must be mapped to real world coordinates. Assuming $\theta_Z = 0$, this mapping requires the third row to be multiplied by the CCD pixel size to convert to camera coordinates in microns. Then, this value is multiplied once again by the inverse of the optical magnification to obtain the results listed in row four in both tables. The results in the last row of both tables indicate the actual displacement that the camera is able to detect through Method 1. The CCD pixel size for this calculation is taken from the camera manufacturer and for the experimental setup is known to be 5.6x5.6 μm.

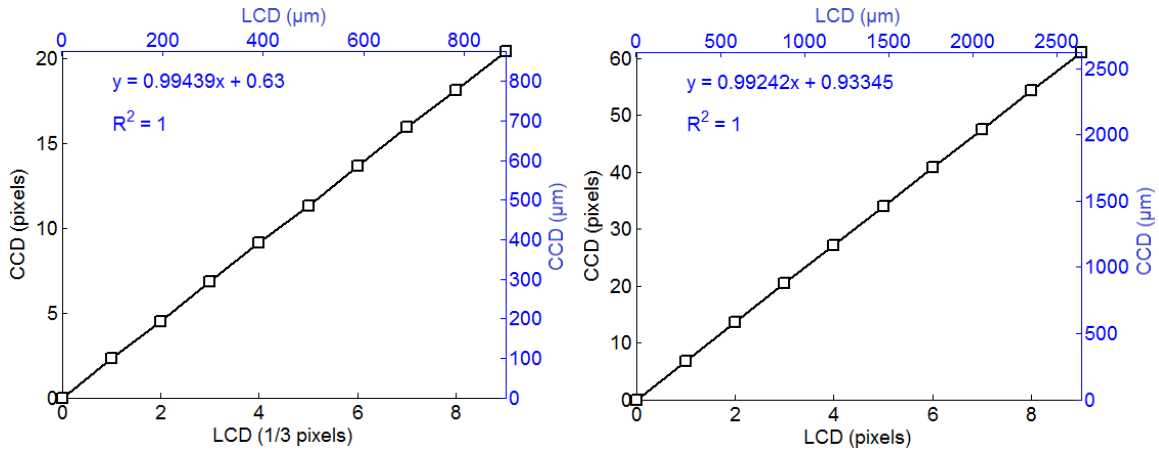
Table 4.2: *X*-coordinates of the cross-hairs, calculated using Method 1, for 10 discrete motion steps of the target along LCD *X*-axis. The averages are each associated to a 100 image sample collected at each step. The inverse of the optical magnification, calculated through step 3 and averaged from all samples, is M^{-1}= 7.6641.

	Units	Data									
Target on LCD	(1/3 LCD pixels)	0	1	2	3	4	5	6	7	8	9
	(μm)	0.00	98.00	196.00	294.00	392.00	490.00	588.00	686.00	784.00	882.00
CCD Sensor	(pixels)	0.00	2.34	4.49	6.82	9.14	11.32	13.64	15.97	18.14	20.45
	(μm)	0.00	100.56	192.86	292.71	392.45	485.89	585.34	685.57	778.62	877.55

Table 4.3: *Y*-coordinates of the cross-hairs, calculated using Method 1, for 10 discrete motion steps of the target along LCD *Y*-axis. The averages are each associated to a 100 image sample collected at each step. The inverse of the optical magnification, calculated through step 3 and averaged from all samples, is M^{-1}= 7.6654.

	Units	Data									
Target on LCD	(pixels)	0	1	2	3	4	5	6	7	8	9
	(μm)	0.00	293.81	587.62	881.43	1175.24	1469.05	1762.86	2056.67	2350.48	2644.29
CCD Sensor	(pixels)	0.00	6.81	13.61	20.40	27.21	34.01	40.81	47.54	54.34	61.15
	(μm)	0.00	292.17	584.14	875.89	1168.05	1460.03	1751.82	2040.92	2332.59	2624.95

The plots in Figure 4.13 summarize the results from Table 4.2 and Table 4.3. The displacement of the target as detected by the sensor (*Y*-axes) is compared against the known displacement (*X*-axes). The regression lines in both plots are calculated using the data given in microns (blue axes). The proximity to a unitary value in both slopes (0.99439 and 0.99242) indicates almost ideal 1-to-1 mapping from actual to recorded distances. Further, both lines present a correlation coefficient of exactly 1, which implies that there exists a desired linear relationship between the actual displacement and the sensor data.



(a) (b)

Figure 4.13: Experimental plots. Data taken from (a) Table 4.2 and (b) Table 4.3

The ability of the image processing algorithm to measure target orientation is also tested. The camera mount is purposely rotated by random steps for this test, and at each point a 100 image sample is collected for offline analysis. In addition, the angle of rotation of the camera with respect to the stage is recorded through a rotary encoder, which provides a secondary reading for comparison purposes. The first two rows in Table

4.4 show the target orientation recorded using the rotary encoder. The last two rows list the average orientation of each 100 image sample, after being processed using Method 1. The results from Table 4.4 are plotted in Figure 4.14. The best fit line exhibits once again almost ideal 1-to-1 mapping between encoder and camera readings and has a correlation coefficient of exactly 1.

Table 4.4: θ-coordinates of the cross-hairs, calculated using Method 1, for 6 discrete target orientations with respect to the CCD coordinate frame. The averages are each associated to a 100 image sample.

	Units	Data					
Encoder	(counts)	1	6719	13267	19462	24059	38996
	(deg)	-1.53	5.19	11.74	17.93	22.53	37.47
CCD Sensor	(rads)	-0.03	0.09	0.20	0.31	0.39	0.65
	(deg)	-1.53	5.19	11.74	17.93	22.53	37.47

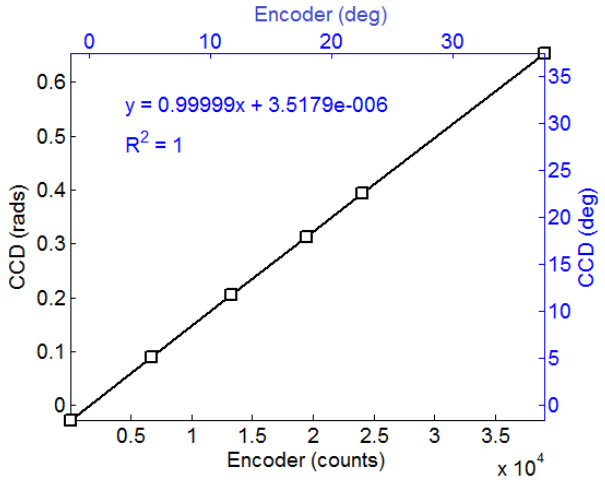


Figure 4.14: Comparison of encoder and vision sensor readings for different target orientations. Data taken from Table 4.4.

The number of iterations for convergence in the Newton-Raphson routine, for the data in Table 4.2, was found to lie within the interval [0,17] for ε = 6E-8 in (4.29). For the data in Table 4.3 the number of iterations for convergence was found to lie within the interval [0,5] for ε = 7.2E-7 in (4.29). The average off-line processing frequency for all 2000 images associated to the samples in Table 4.2 and Table 4.3, was found to be 4.4 Hz when utilizing a 64-bit Windows OS machine with an Intel(R) Core(TM) i3 M330 @2.13GHz, 4GB RAM.

The regression lines in Figure 4.13 and Figure 4.14 evidently exhibit a desired linear behavior. Nevertheless, if the goal is to track a continuously moving target, then the system must be capable of accurately computing the position and orientation of the former using a single image at each point in time. Therefore, no averaging between information extracted from consecutive images can take place, as each new frame might correspond to different target locations on the LCD. It is therefore necessary to study the level of uncertainty present in individual observations and then determine the resolution of the 3D position sensor.

The experimental resolution of the position transducer and the level of uncertainty, u, present in the measurements are determined through a statistical analysis. For this purpose, a large sample affected by Gaussian noise is collected. The assumption that the experimental data is normally distributed allows the calculation of the resolution with an associated level of confidence. A procedure for determining measurement uncertainty, u (denoted in the following quoted text as u_j), associated with a single

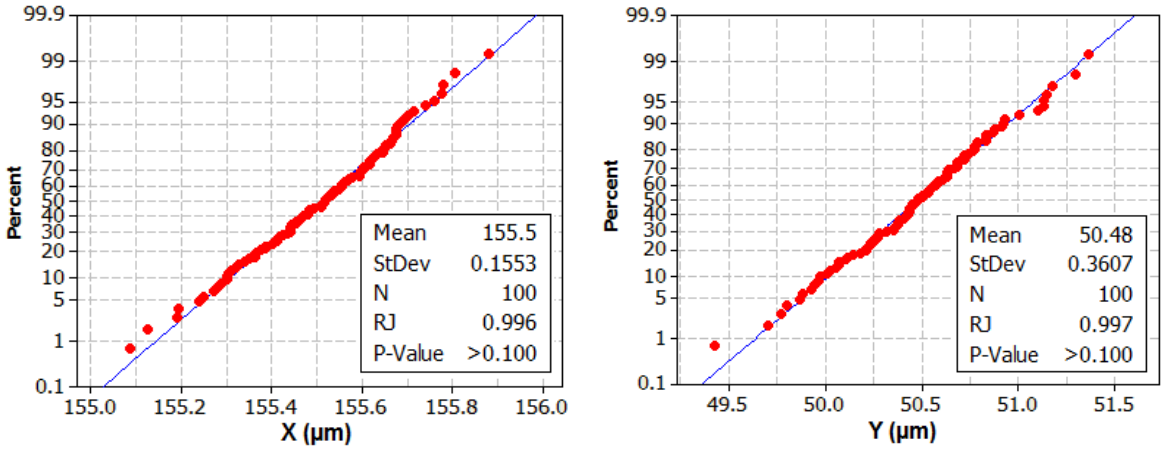
observation belonging to a normally distributed sample is presented in *The NIST Reference on Constants, Units and Uncertainty* [74], which states:

> "If the quantity in question is modeled by a normal probability distribution, there are no finite limits that will contain 100% of its possible values. However, plus and minus 3 standard deviations about the mean of a normal distribution corresponds to 99.73% limits. Thus, if the limits a_- and a_+ of a normally distributed quantity with mean $(a_+ + a_-)/2$ are considered to contain "almost all" of the possible values of the quantity, that is, approximately 99.73% of them, then u_j is approximately $a/3$, where $a = (a_+ - a_-)/2$ is the half-width of the interval."

Following the previous procedure the resolution, *r*, is defined based on the range $\mu \pm 3\sigma$, where μ is the sample's mean and σ is the standard deviation, provided that almost all possible values (99.73%) of the measurand are covered within this range. This is $r = 6\sigma$. The level of uncertainty in a single observation is $u = 3\sigma$.

A 100 image sample is collected. The target's pose in all 100 images is calculated through Method 1 and can be defined as a 3D-coordinate array, $\mathbf{Pose} = \{(x_1, y_1, \theta_1), (x_2, y_2, \theta_2), ..., (x_{100}, y_{100}, \theta_{100})\}$. This array is separated in three sub-arrays **X**, **Y** and **θ**, containing only the *x*-, *y*- and θ-coordinates of all points, respectively. The units of **X** and **Y** are given in μm (LCD-real world coordinates) whereas the units of **θ** are given in radians. First, the hypothesis that the data sets **X**, **Y** and **θ** are normally distributed must be tested. A common procedure for testing this hypothesis is through a normal probability plot. A normal probability plot displays the z-values (along plot *Y*-axis), calculated for the percentiles of each experimental data point, against the corresponding data points (plot *X*-axis). Further details for generating the probability plot are omitted as they can be found in other documents (see chapter 10 in [75]). The

experimental values are compared against a theoretical normal distribution. If the experimental values are indeed normally distributed, they should follow an approximately linear trend. Deviations from the linear behavior suggest non-normality of the distribution. Figure 4.15 shows the probability plots for **X** and **Y**, with correlation coefficients of 0.996 and 0.997, respectively, indicating a strong linear behavior. The hypothesis that the data is normal cannot be rejected with the current results, i.e. it is not wrong to assume from the current results that the data sets **X** and **Y** are normally distributed.



(a) (b)

Figure 4.15: Normal probability plots for the sub-arrays: (a) **X** and (b) **Y**.

Figure 4.16 shows the distribution plots for the data sets **X** and **Y**. The resolution for **X** is $r_x = 6\sigma_x$, whereas the resolution for **Y** is $r_y = 6\sigma_y$. The lower sample limit (LSL)- and upper sample limit (USL)-markers indicate that the experimental distributions

span over $\mu_x \pm 3\sigma_x$ and $\mu_y \pm 3\sigma_y$, approximately. The corresponding resolutions are $r_x = 0.93$ µm and $r_y = 2.16$ µm.

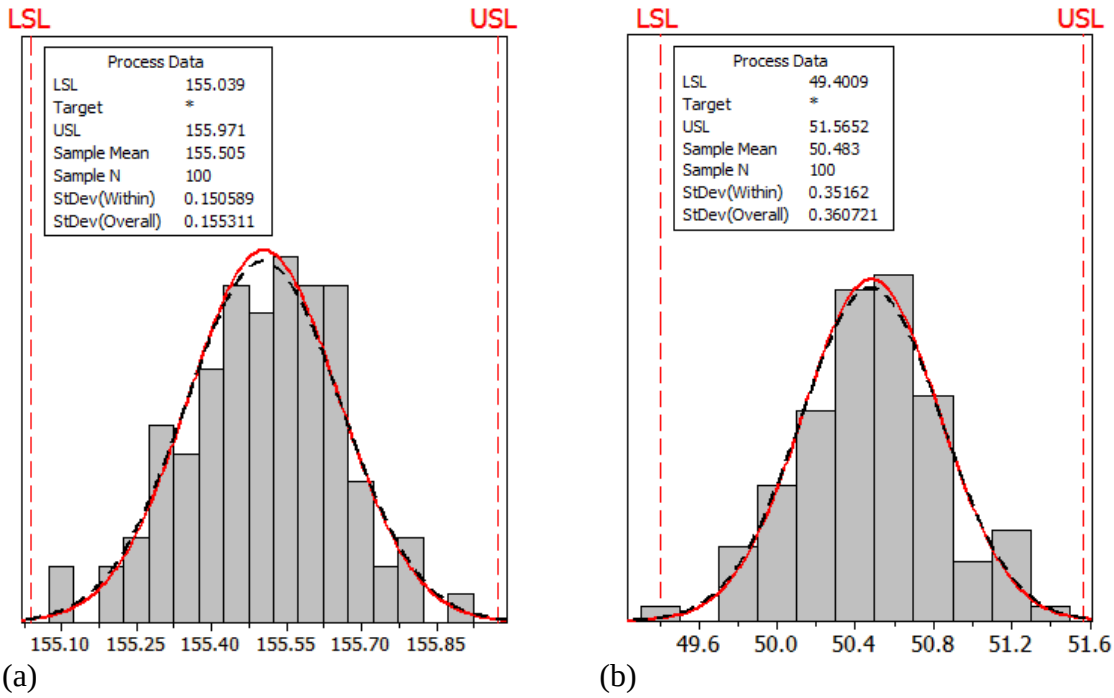
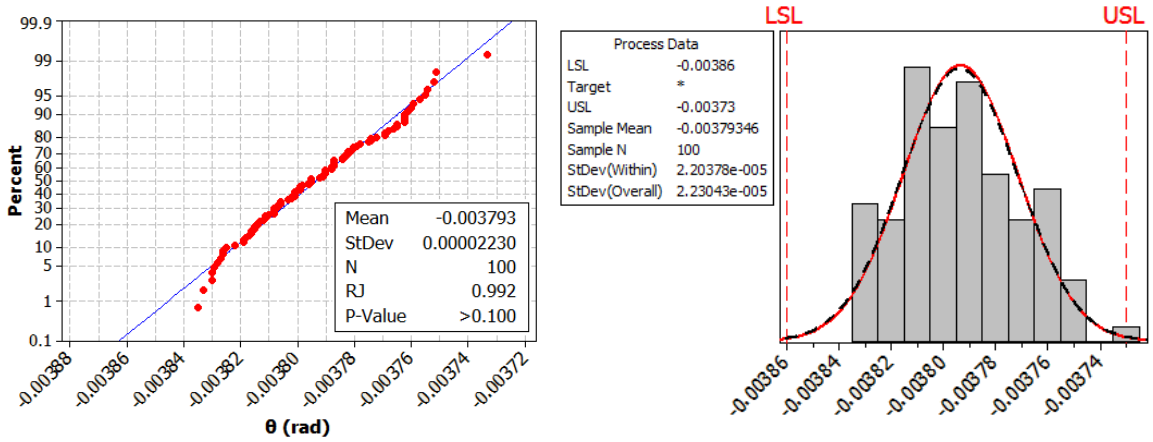


Figure 4.16: (a) **X**-sample distribution measured in µm. $\text{LSL} = \mu_x - 3\sigma_x$; $\text{USL} = \mu_x + 3\sigma_x$. (b) **Y**-sample distribution measured in µm. $\text{LSL} = \mu_y - 3\sigma_y$; $\text{USL} = \mu_y + 3\sigma_y$.

Figure 4.17 (a) shows the normal probability plot for **θ** with a correlation coefficient of 0.992 also demonstrating strong linear behavior. The LSL- and USL-markers in Figure 4.17 (b) indicate that the experimental distribution spans over $r_\theta = 6\sigma_\theta$, approximately. The corresponding resolution is r_θ = 0.001368 radians, or 0.078355°. Resolution results for all 3 degrees of freedom of the experimental system are summarized in Table 4.5.



(a) (b)

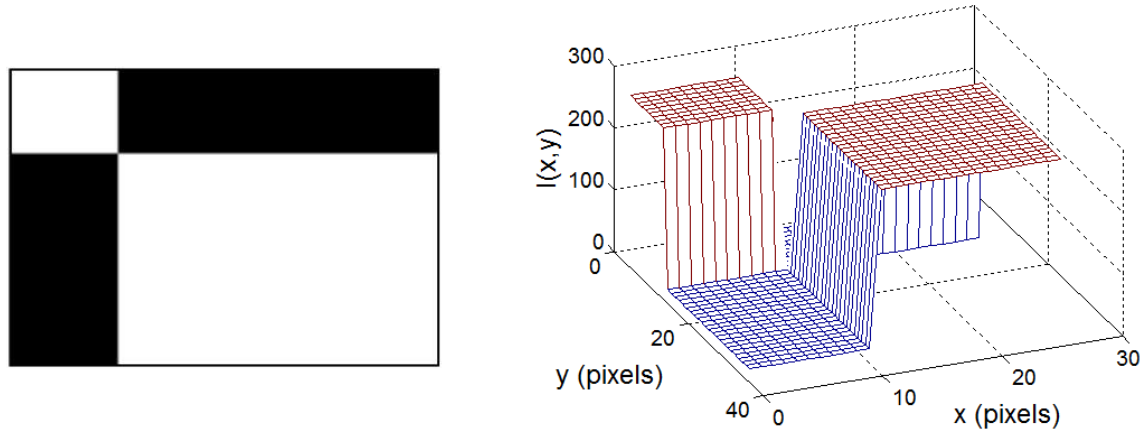
Figure 4.17: (a) Normal probability plot for sub-array **θ**. (b) **θ**-sample distribution measured in radians. $\mathrm{LSL} = \mu_\theta - 3\sigma_\theta$; $\mathrm{USL} = \mu_\theta + 3\sigma_\theta$.

Table 4.5: Experimental resolution. Lateral (X and Y) resolution results are calculated for a CCD pixel size of 5.6x5.6 μm and after applying the inverse magnification factor, M^{-1}= 7.6641.

		X (um)	**Y (um)**	**θ (rads)**	**θ (deg)**
Sample's Std.		0.15531092	0.360721	2.23E-05	0.001278
Resolution	r_x	**0.931865521**	-	-	-
	r_y	-	**2.164324**	-	-
	r_θ	-	-	**0.000134**	**0.007668**

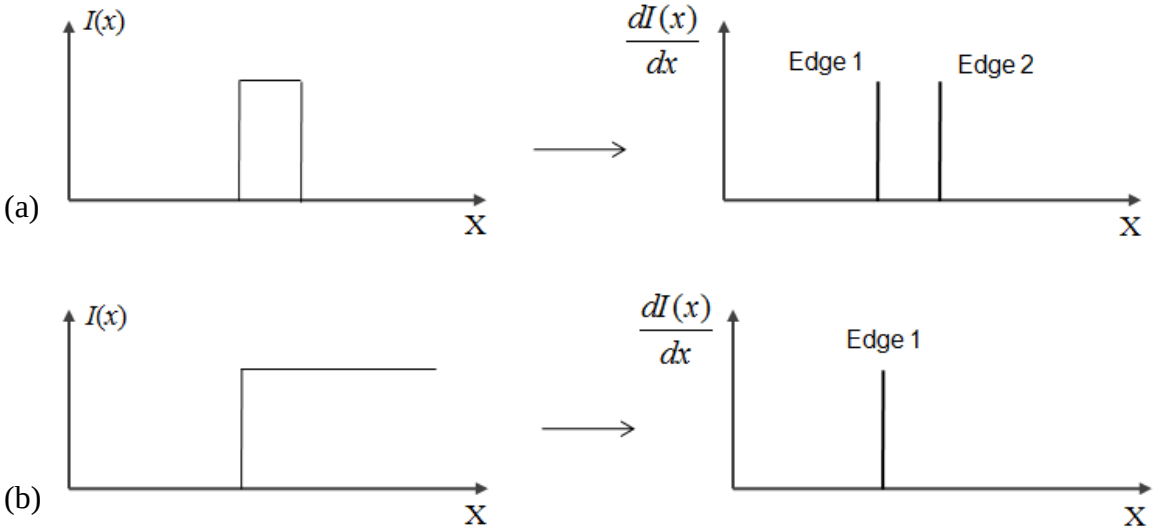
MTP – Method 2: Checkered Pattern Identification

A second algorithm is proposed, where the object of interest on the display is embodied by four homogeneous grayscale-intensity areas, forming a checkered pattern (Figure 4.18 (a)). The main idea is to identify the location of the edges with high accuracy, correlate this data with two perpendicular lines and then, again, define the position of the target as the intersection of these lines. In the previous section it was demonstrated that the intensity-weighted centroid formula presented an appropriate method for performing sub-pixel resolution measurements, when detecting a white cross-hairs over black background. However, the centroid formula is not suitable for the checkered pattern and a different approach must be taken. An important feature of the checkered pattern is the type of intensity transitions present in the image. This aspect is demonstrated when calculating the intensity gradient along the image *X*-axis (or *Y*-axis for vertical transitions), as shown in Figure 4.19. In order to estimate the position of the midpoint of an intensity impulse function, such as those in the cross-hairs target introduced in Method 1, an image processing algorithm based on edge-detection would have to deal with inaccuracies in the location of two edges (Figure 4.19 (a)). The checkered pattern presents potential advantages in this regard, as only one intensity transition happens around the pixels of interest (Figure 4.19 (b)).



(a) (b)

Figure 4.18: (a) Theoretical active target represented by a checkered pattern and (b) corresponding 3D intensity plot.



(a)

(b)

Figure 4.19: (a) Intensity impulse function and corresponding 2D gradient. (b) Intensity step function and corresponding 2D gradient.

The checkered pattern method initiates with the calculation of the image gradient, for coarse-location of the edges. An edge is defined as a transition starting at a low-intensity pixel and ending at a high-intensity pixel (or the opposite of this: starting at a saturation point and ending at a low-intensity pixel), within a given pixel window. Fine point identification is performed by fitting a 2D Gaussian curve, to a horizontal or vertical array of pixels around an intensity peak in the gradient image. The selection of the Gaussian for curve fitting follows directly from the 2D bell-shape patterns (along a row or column of pixels) found in the gradient image around the target edges. The mean of the best-fit Gaussian curve is regarded as the edge location within the vertical or horizontal pixel window. The identification of all edges in the target image leads to the definition of two data sets, associated with two lines. The final step consists in determining the two perpendicular lines that best fit the two data sets. This last step is the same as step 2 in Method 1. Again, from the physical construction of the LCD pixels, it is anticipated that the smallest displacement of the LCD target will be constrained to 1/3 of a pixel along the *X*-axis, and one whole pixel along the *Y*-axis.

Step 1: Coarse point check through image gradient

The first step is the computation of the image gradient. Although there are highly elaborated gradient filters, this step is intended for coarse point location only. A simple gradient operator, such as *Prewitt*, satisfies this initial criterion. Figure 4.20 shows a digital image of the experimental checkered pattern, as well as its 3D representation and corresponding gradient image.

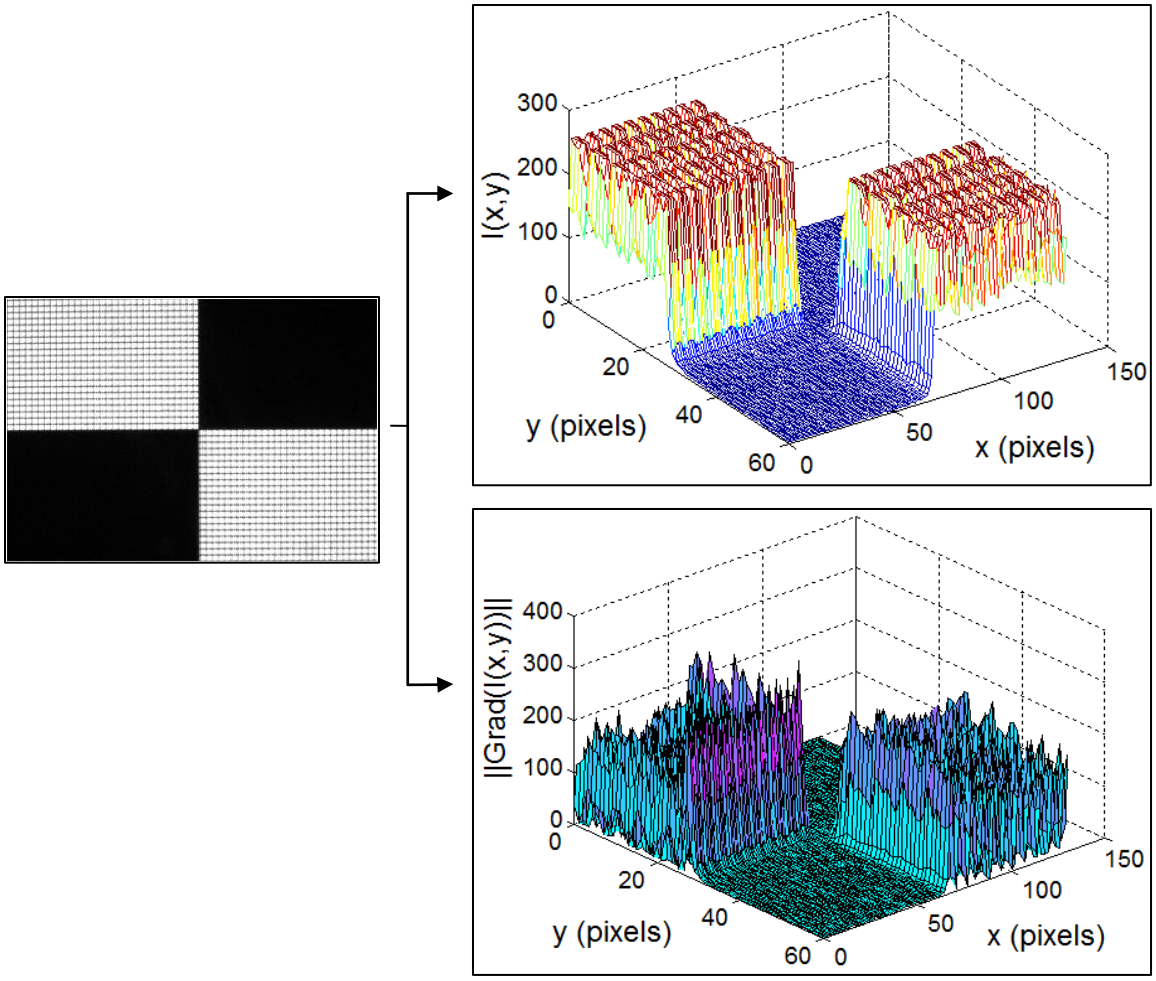


Figure 4.20: Snapshot of experimental target image (middle-left); corresponding 3D intensity plot (top-right) and corresponding 3D gradient plot (bottom-right).

Remark 4.2: LCD pixel gaps

The 3D gradient graph in Figure 4.20 exhibits local maxima, not only around the target edges, but also in areas that map from high intensities pixels (white regions) on the LCD. This phenomenon reveals that the camera is able to recognize the small low intensity (black) gaps that physically separate LCD pixels, when sensing high intensity patterns on the LCD. This is also confirmed in the 3D intensity plot, where local minima are found in regions that map from high intensity LCD pixels. A robust algorithm is required to distinguish between real target edges and LCD pixels gaps, without data losses.

A valid 1D edge is defined in the gradient image as an array of pixels which, when read from left to right (row along *X*-axis) or from top to bottom (column along *Y*-axis), contains: a pixel with an intensity below a predefined low-intensity threshold, followed by a second pixel with an intensity above the predefined low-intensity threshold; then, (one or several pixels later) a gradient peak and finally (one or several pixels later) another pixel with an intensity below the low-intensity threshold, preceded by another one with an intensity above it. Figure 4.21 shows three 2D gradient curves, extracted from three consecutive rows (Row 1, Row 2 and Row 3) of an experimental target image. The pixels inside these rows contain information about a horizontal edge on the target image. Based on the previous definition, it can be stated that Row 2 presents a valid edge, starting at the fifth pixel (pixels in this figure are equally spaced, by increments of 1) and ending at the twelfth pixel (or thirteenth, depending the arbitrary value of the low-intensity threshold). Likewise, Row 3 presents a valid edge starting at the sixth pixel and ending at the twelfth pixel. It is worth pointing out that the behavior of the discrete gradient values in rows 2 and 3 follow a similar pattern; this characteristic is desired as it can help repeatability in the curve-fitting process performed later in step 2. On the contrary, the gradient curve associated to Row 1 does not satisfy the definition of a valid edge. The local maxima found in the area enclosed by the two arrows, in Figure 4.21, are clearly related to intensity transitions mapped from LCD pixel gaps. These transitions can be thought of as noise, and should not be confused with target edges for measurement purposes.

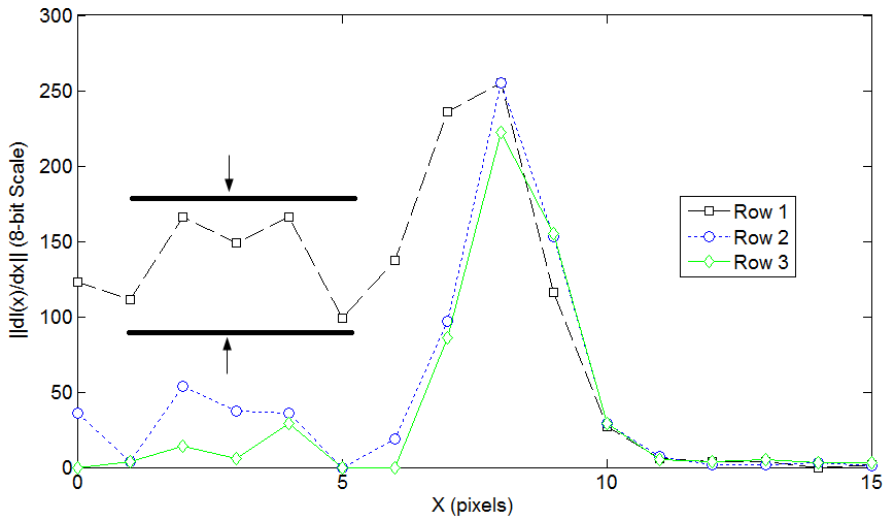


Figure 4.21: 2D gradient curves for three consecutive rows of a checkered pattern image, covering a transition area of interest.

Step 2: Fine edge locus though Gaussian-curve fitting

A fine edge location is achieved by calculating the 2D Gaussian curves that best fit the valid horizontal or vertical edges, in the gradient image (details on how to calculate a best fit Gaussian curve from experimental data are found in appendix C). The 2D Gaussian function is known to have the form

$$f(x, A_G, \mu_x, \sigma) = A_G e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}} \tag{4.37}$$

with mean μ_x, standard deviation σ_x and maximum amplitude A_G. The independent variable x in (4.37) contains the X-coordinates of the pixels covering a valid horizontal edge defined in the gradient image. Then, for a valid horizontal edge located at row i, the coordinates of the edge on the image plane are (μ_x,i), where μ_x is the mean of the Gaussian curve that best fits the data points. The same procedure is followed when calculating the position of a vertical edge with sub-pixel resolution, but using pixels within vertical edges and fixing the value of the x-coordinate to the column under analysis. Therefore, a vertical edge located at column j, would be represented by the coordinates (j,μ_y), where μ_y is the mean of the best fit Gaussian curve. Similar to Method 1, edges close to the intersection of the checkered pattern must be disregarded to avoid confusion and increase image processing times.

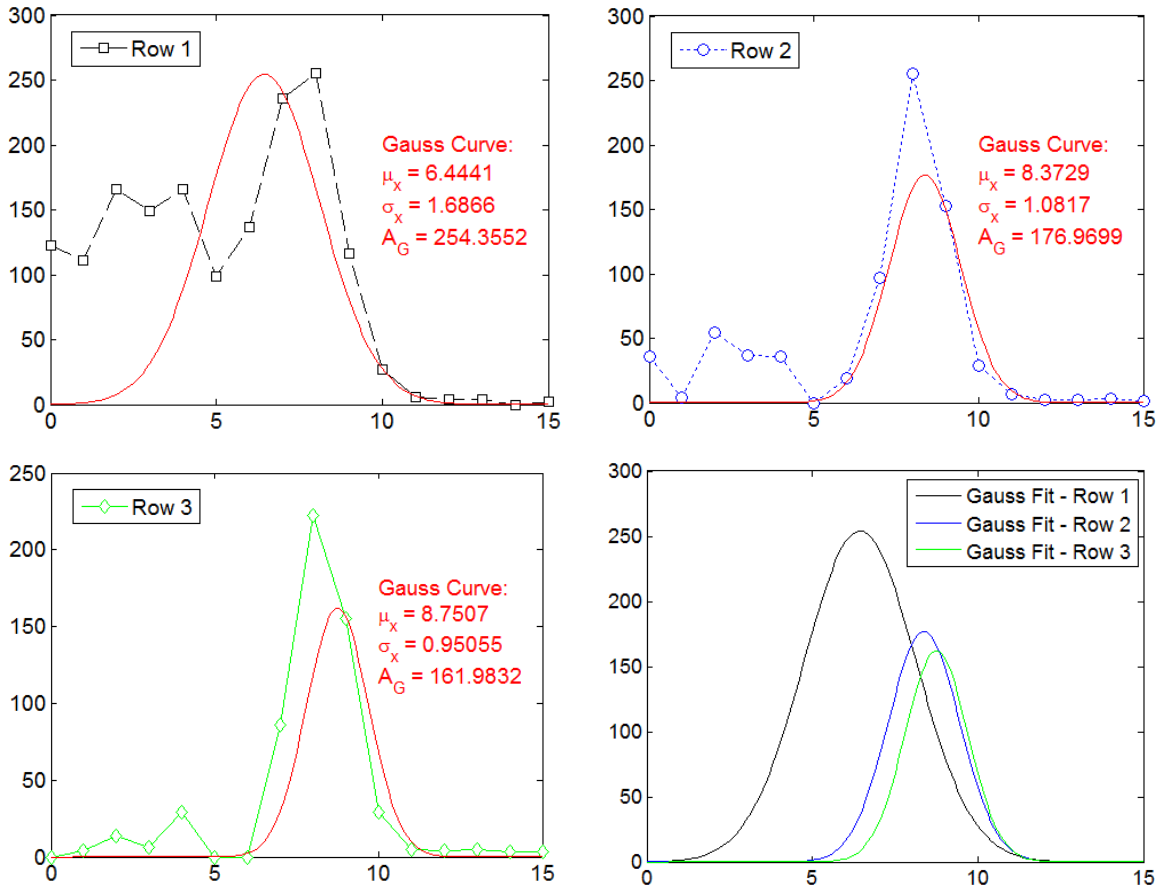


Figure 4.22: Results of Gaussian curve fitting applied to the gradient curves in Figure 4.21.

The Gaussian curve fitting procedure is applied to the three rows of pixels presented in the gradient plots of Figure 4.21. The results are shown in Figure 4.22. According to the definition of a valid edge, given in Remark 4.2, row 1 (upper left) is not to be classified as an edge, as it is not clear which pixels should be included in the curve fitting process. Nevertheless, a pixel window $x \in [4,12]$ is selected for curve fitting in row 1, for comparison purposes. A gradient low-intensity threshold of 4 is set to define the pixel intervals for the edges in rows 2 and 3. This means that an edge in the gradient image should start at a pixel where the gradient ($\|dI(x)/dx\|$) is equal or lower than 4, it

should then go through a local gradient maximum, and finally go back to a pixel with a gradient value of 4 or lower. As seen, the value of μ_x for row 1 (6.4441) is highly shifted towards the left in the *X*-axis, with respect to the values of μ_x for the other two rows (8.3729 and 8.7507), which is a consequence of incorrect edge identification. The definition of a "valid edge" helps to better locate the points where the checkered pattern transitions from one intensity quadrant to another, while avoiding confusions associated with the mapping of LCD pixel gaps. This is expected to improve repeatability in the measurements. Once the coordinates of all valid edges have been identified, these points are separated in two sets, *DSE*1 and *DSE*2 associated to two different lines.

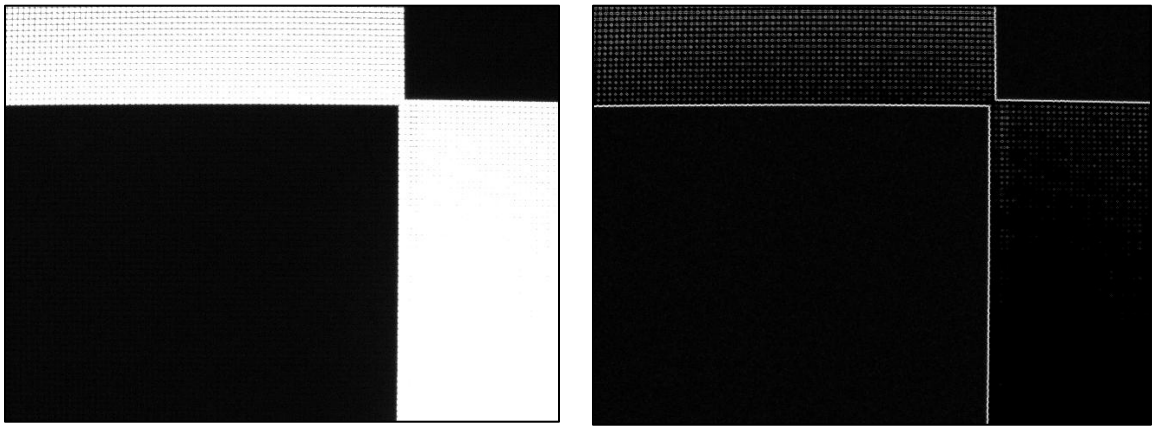
Step 3: Constrained curve fitting and target identification

The location of the target is obtained by calculating the two perpendicular lines that best fit the data sets *DSE*1 and *DSE*2 collected in the previous step, and then analytically computing the intersection of these lines. This procedure is the same as the one explained in step 2 for Method 1. The relative target orientation, for Method 2, is calculated for a range of $[-\pi/2,\pi/2]$ through (4.30).

Experimental Results

In theory the checkered pattern target presents several advantages with respect to the white cross-hairs, especially when using the local intensity gradient instead of the centroid formula for retrieving sub-pixel resolution information. Experimentally, however, some difficulties associated with inconsistent intensity-readings are encountered depending on the location of the target with respect to the CCD frame.

The problem is related to the amount of white area versus black area that the sensor sees at a given time, depending on the location of the target. If the in-plane position of the target is such that the intersection of the four quadrants is close to a corner of the CCD chip, the ratio between white LCD pixels and black LCD pixels detected by the sensor is not close to one. Whenever a large dark (low-intensity) region covers the majority of the FOV (as shown in Figure 4.23 (a)), the camera pixels become highly sensitive to any light-emitting sources almost regardless of the intensity of the source. In such cases, the CCD pixels are easily saturated yielding high intensity contrasts in the target image; this, in turn, results in strong peaks around the target edges in the gradient image (Figure 4.23 (b)). This effect is demonstrated in the 2D gradient plot of Figure 4.24; as can be observed, high gradient peaks above 200 are easily retrieved from the gradient image around an edge while LCD pixel gaps are nearly undetectable.



(a) (b)

Figure 4.23: (a) Snapshot 1 of experimental checkered pattern target, taken using monochrome camera with lens aperture setting 1. (b) Corresponding gradient image.

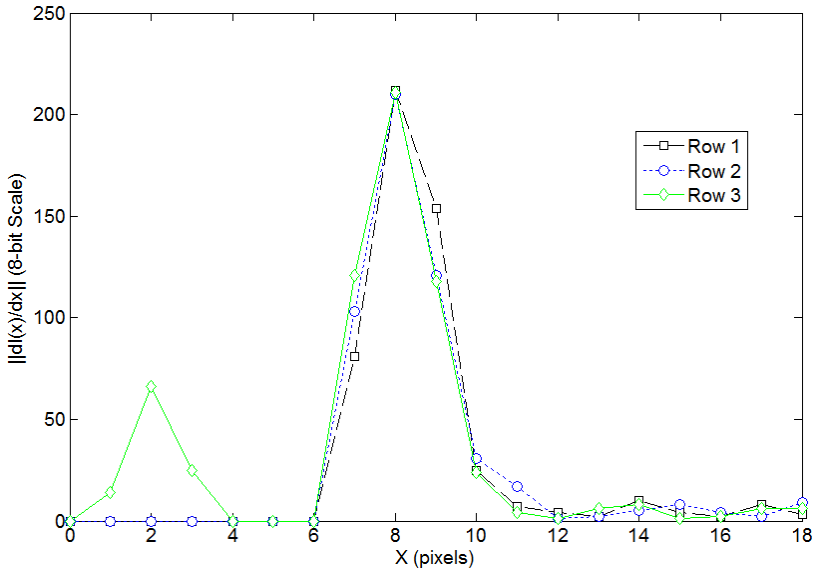
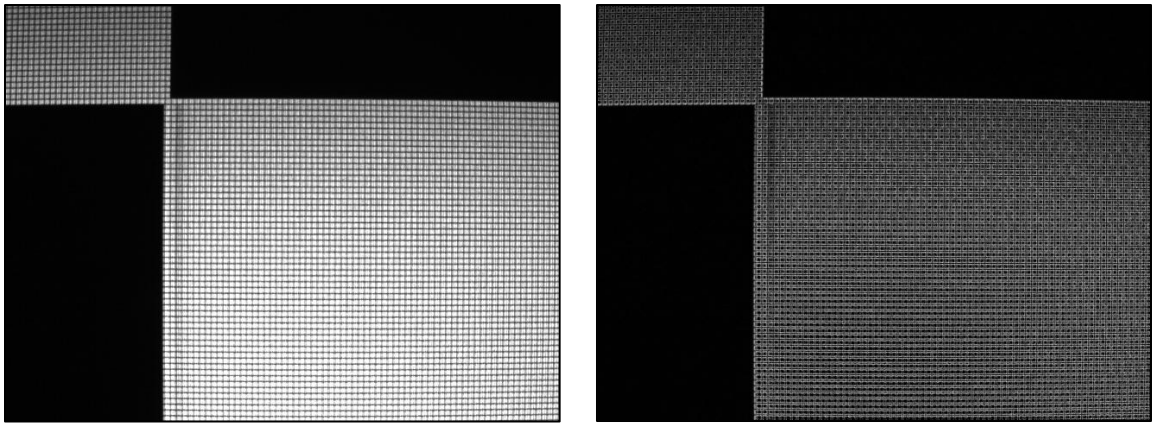


Figure 4.24: 2D gradient curves for three consecutive rows, extract from Figure 4.23 (b), around an edge.

On the other hand, whenever a bright (high-intensity) region covers the majority of the FOV, the overall light intensity perceived by the camera is high, and saturation does not occur as easily. Figure 4.25 (a) shows a snapshot of the same target presented in Figure 4.23, taken with the same camera and with the same lens aperture (lens aperture setting 1), but located at a different position with respect to the CCD frame. A simple visual comparison between Figure 4.23 (b) and Figure 4.25 (b) indicates inconsistent intensity mappings depending on the location of the checkered pattern with respect to the image plane. A close look at the 2D gradient curves, in Figure 4.26, also reveals intensity gradient peaks lower than 120, which implies low contrast in the original image. Finally, LCD pixel gaps are easily detected in the digital image, presenting another discrepancy in the mapping.



(a) (b)

Figure 4.25: (a) Snapshot 2 of experimental checkered pattern target, taken using monochrome camera with lens aperture setting 1. (b) Corresponding gradient image.

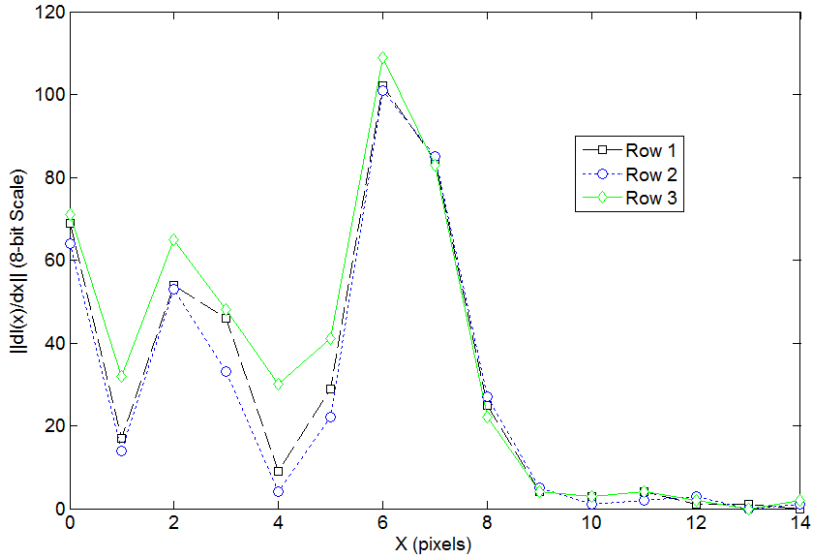


Figure 4.26: 2D gradient curves for three consecutive rows, extract from Figure 4.25 (b), around an edge.

Hence, for this particular target, the sensor presents intensity inconsistencies when detecting the same target located at different positions with respect to the camera FOV. The white cross-hairs over a black background, on the contrary, offers a much more

homogenous pattern for the camera to detect. Consistent intensity readings are certainly a desired characteristic as accuracy and repeatability of the sub-pixel resolution algorithm depend on it.

Concluding Remarks

Two new methods for measuring the 3D absolute position of a dynamic target displayed on an LCD display were presented. Experimental results for the cross-hairs target show that measurement resolutions on the order of less than 2.5 microns are reliably achieved for planar motion, and smaller than 0.008° for in-plane rotation measurements. Although the image processing algorithm is capable of such resolution levels, displacement increments of the target image on the LCD are constrained to the size of an LCD pixel. Therefore, if motion commands are to be issued through dynamic target positioning for the new position control system described in Chapter Two, additional elements of the overall control scheme must be studied and utilized to achieve this goal. This topic is discussed in Chapter Six, where two new methods for sub-pixel command issuing are explained. In addition, experimental tests reveal better results when imaging a white cross-hairs over black background on the LCD screen through a CCD monochrome camera, compared with the checkered pattern. The cross-hairs pattern exhibits a more homogeneous intensity pattern to be mapped onto the CCD chip.

CHAPTER FIVE

MODELING OF SYSTEM DYNAMICS AND THE MULTI-FREQUENCY FEEDBACK LOOP

The use of MTP presents clear advantages with respect to ordinary position control systems as it provides a robust way for estimating position without being affected by geometric errors of the machine. However, this method also presents a big challenge that must be overcome, in order to successfully achieve high performance levels. From a control system perspective, the biggest problem resides in the latency of the closed loop feedback signal, due to low frame acquisition rates and high image processing times.

The goal of this chapter is to study the control system as a whole, and identify a specific strategy that will allow successful implementation of MTP in real machines. The first section - Stage Modeling and System Identification - is dedicated to the derivation of a mathematical model of the test-bed stage. In the second section - System Integration - full attention is given to the dynamic problems in visual servoing. Stability and regular operation of the control system are studied in the last section - Case Study: Model-based Vision Control System using a *Single CAM- Single Image Processing Thread*-Configuration.

Stage Modeling and System Identification

Servo Motor Modeling

One of the main goals of the current project is to develop a control strategy that does not fully rely on a kinematic model of the machine to estimate the position of the point of interest (POI). However, as will be explained later, a mathematical representation of the machine is still required for prototyping and implementation. In general, a linear time-invariant (LTI) multiple-input multiple-output (MIMO) system can be represented in the frequency domain through $\mathbf{X(s)} = \mathbf{G(s)V(s)}$, where the output vector **X(s)** has *n* elements (i.e. there are *n* outputs), and the input vector **V(s)** has *m* elements (i.e. there are *m* inputs). It follows that

$$\mathbf{X(s)} = \begin{bmatrix} X_1(s) \\ X_2(s) \\ \vdots \\ X_n(s) \end{bmatrix}; \quad \mathbf{V(s)} = \begin{bmatrix} V_1(s) \\ \vdots \\ V_m(s) \end{bmatrix}; \quad \mathbf{G(s)} = \begin{bmatrix} G_{11}(s) & \cdots & G_{1m}(s) \\ \vdots & & \\ G_{n1}(s) & \cdots & G_{nm}(s) \end{bmatrix} \tag{5.1}$$

where the transfer function for the *i*-th subsystem is defined as $X_i(s) = \mathbf{G_i(s)V(s)}$. Given the fact that the dynamics of individual axes on the stage are decoupled, it is possible to rewrite the equation for the experimental MIMO system as

$$\begin{bmatrix} X_1(s) \\ X_2(s) \\ X_3(s) \end{bmatrix} = \begin{bmatrix} G_1(s) & 0 & 0 \\ 0 & G_2(s) & 0 \\ 0 & 0 & G_3(s) \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \\ V_3(s) \end{bmatrix} \tag{5.2}$$

where each axis is a SISO systems of the form $X_i(s) = G_i(s)V_i(s)$. The output vector in (5.2) actually represents the three motion axes of the stage, i.e.

$\begin{bmatrix} X_1 & X_2 & X_3 \end{bmatrix}^T = \begin{bmatrix} X & Y & \theta_Z \end{bmatrix}^T$. The analysis that follows is conducted for one axis (*X*-axis), only. Modeling and other considerations for the second- and third-axes (*Y*-axis and θ_Z-axis) can be obtained in a similar way as for the first one.

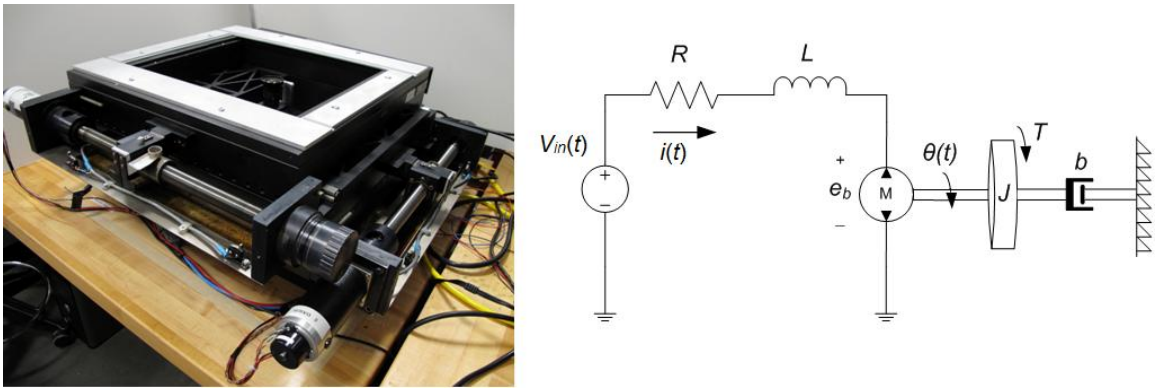


Figure 5.1: Single axis model.

Figure 5.1 shows a mechatronic model for a servo motor used to describe the dynamics of a single axis. The system, as shown in Figure 5.1, consists of an *RL* electric circuit with a power source V_{in}, a motor powered by the circuit, an inertial mass *J* subject to the motor output torque *T*, and a damper with damping coefficient *b*. The system's equations are

$$\begin{cases} V_{in} - Ri - \dfrac{di}{dt}L - e_b = 0 \\ T - b\dfrac{d\theta}{dt} = J\dfrac{d^2\theta}{dt^2} \\ T = K_t i \\ e_b = K_b \dfrac{d\theta}{dt} \end{cases} \tag{5.3}$$

with units as shown in Table 5.1.

Table 5.1: System parameters with corresponding units.

Symbol	Units	Symbol	Units
V_{in}	*V*	θ	*rad*
R	*ohms*	*b*	*N-m-s*
i	*A*	*J*	$Kg\text{-}m^2$
L	*H*	K_t	*N-m/A*
e_b	*V*	K_b	*V-s*
T	*N-m*		

The first two equations in (5.3) are derived by applying Kirchhoff's voltage law to the circuit under analysis and Newton's second law for rotation to the inertial mass, respectively. The last two well-known equalities describe the relationship between the output torque and the circuit current through the torque constant K_t, and the relationship between the motor back EMF voltage and the angular velocity through the back EMF constant K_b. A transfer function for the system is obtained by defining the angular rotation of the shaft, θ, as the output, taking the input voltage as the input of the system and assuming zero initial conditions. The result is a third order system, namely

$$\frac{\theta(s)}{V_{in}(s)} = \frac{K_t}{LJs^3 + (RJ + Lb)s^2 + (Rb + K_b K_t)s} \tag{5.4}$$

Assuming that the pole associated with the inductor dynamics is at least five times faster than the dominant system poles, this pole can be neglected and the system is reduced to a second order one. Then, the model for the servo dynamics becomes

$$\hat{G}_\theta(s) = \frac{\hat{\theta}(s)}{V_{in}(s)} = \frac{K_t}{RJs^2 + (Rb + K_b K_t)s} \tag{5.5}$$

System Identification: *ARX* Model

This section presents the results obtained from the *ARX* system identification (SI) method, available in the LABVIEW SI Toolkit. In order to determine the parameters of the mathematical model, the servo motor must be stimulated by a known rich signal that should excite a wide range of frequencies. Ideally, the input signal should satisfy the persistent excitation (PE) condition, which would guarantee convergence of model parameters to the real parameters of the plant as $t \rightarrow \infty$ (see chapter 4, section 4.3 in [76]). However, in the last section of this chapter it will be demonstrated that this strong requisite can be relaxed, and stability of the integrated system can be guaranteed as long as the error between the model and the plant is bounded.

For this test, a rotary encoder is utilized to record the output signal, i.e. the rotational position of the shaft; the encoder sampling frequency is 1 kHz. The selected test signal is the chirp function shown in Figure 5.2, which excites the following frequencies: 20Hz, 12.5Hz, 10Hz, 3.3Hz, 2Hz, 1.25Hz, 1Hz and 0.5Hz. Each frequency is held constant for a period of 3 seconds, except for the last two (1Hz and 0.5Hz) which are held constant for 6 seconds each. A 30-second time frame is tested. After recording the system response to the input signal, an offline SI analysis is conducted.

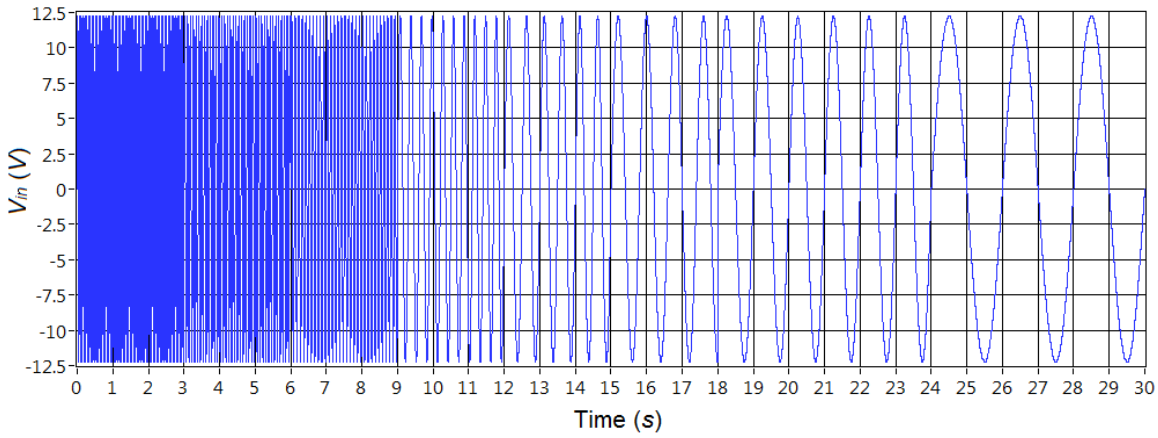


Figure 5.2: Excitation signal.

The *ARX* built-in function in the LABVIEW SI Toolkit provides a direct estimate of the model coefficients in the discrete domain. Before the results are presented, it is necessary to find a discrete representation of (5.5), which will allow an appropriate comparison between the output parameters from the built-in function and those in (5.5). In the time domain, (5.5) can be expressed as

$$RJ\ddot{\hat{\theta}}(t)+\left(Rb+K_bK_t\right)\dot{\hat{\theta}}(t)=K_tV_{in}(t) \tag{5.6}$$

which is a second order ODE (with $\dot{\hat{\theta}}(t)=d\hat{\theta}/dt$, $\ddot{\hat{\theta}}(t)=d^2\hat{\theta}/dt^2$). Using Euler's backward approximation it is possible to write

$$RJ\frac{\left[\hat{\theta}(n)-2\hat{\theta}(n-1)+\hat{\theta}(n-2)\right]}{\Delta t^2}+\left(Rb+K_bK_t\right)\frac{\left[\hat{\theta}(n)-\hat{\theta}(n-1)\right]}{\Delta t}=K_tV_{in}(n) \tag{5.7}$$

which after rearranging can be written in the z-domain as

$$\left[1+a_1z^{-1}+a_2z^{-2}\right]\hat{\theta}(z)=b_0V_{in}(z) \tag{5.8}$$

where

$$\begin{cases} a_1 = \dfrac{[-2RJ - \Delta t(Rb + K_b K_t)]}{[RJ + \Delta t(Rb + K_b K_t)]} \\ a_2 = \dfrac{RJ}{[RJ + \Delta t(Rb + K_b K_t)]} \\ b_0 = \dfrac{\Delta t^2 K_t}{[RJ + \Delta t(Rb + K_b K_t)]} \end{cases} \tag{5.9}$$

From (5.8) it is possible to directly relate the second order model with the output model of the *ARX* function, which is of the form

$$\left[1\text{-}1.91913z^{-1} + 0.919123z^{-2}\right]\hat{\theta}(z) = 0.000892098V_{in}(z) \tag{5.10}$$

The results are summarized in Table 5.2.

Table 5.2: Experimental results from *ARX* function.

a_1	-1.92 E+00
a_2	9.19 E-01
b_0	8.92 E-04

Figure 5.3 shows the real output of the plant (recorded with the encoder) and the output of the discrete model when excited with the chirp function. The average absolute error over the 30-second test period is 29.3101 rad. This value is not necessarily a good indicator of the SI performance. The behavior of the error, between the real plant and the model, over time is usually a better indicator to assess performance of the SI method and is closely related to stability in model-aided control systems. These topics are covered in the last section of this chapter.

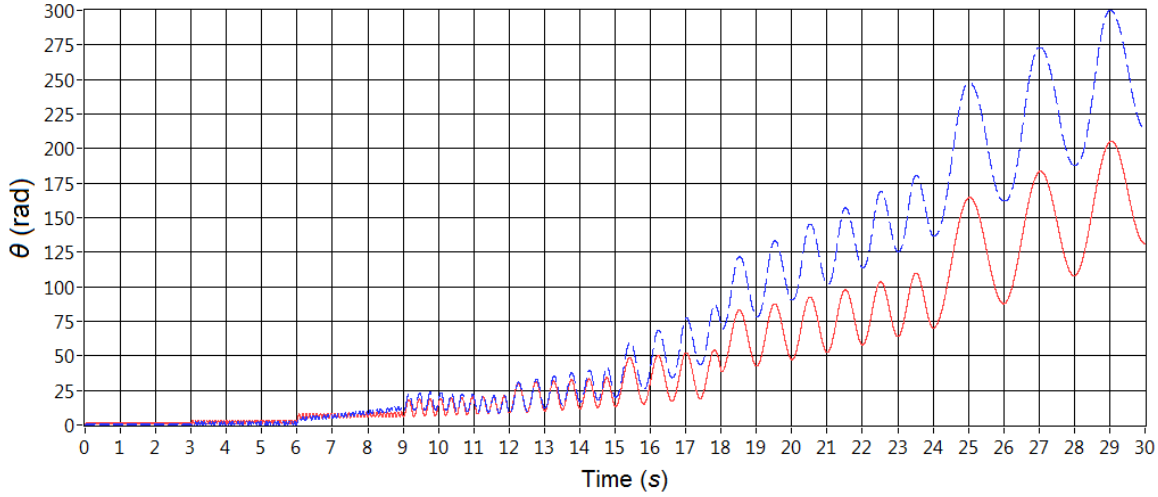


Figure 5.3: Plant (dashed line) response and *ARX* model (continuous line) response to chirp stimulus signal.

System Integration

The structure of the overall positioning system is shown in Figure 5.4, where $C(s)$ is the controller's transfer function and $G_\theta(s)$ is the ideal transfer function that describes the mathematical behavior of the motor with no errors. The coordinate transformation, from angular motion of the shaft to planar displacement of the stage, is assumed to be governed by the scalar factor L_{CAM} with units of µm/rad. This factor is, of course, not required when modeling the rotary stage, i.e. θ_Z. Experimental results using a laser interferometer demonstrated that the angular rotation of the shaft is related to the planar displacement of the stage by a constant factor $L_{CAM} = 1000/8\pi$ µm/rad.

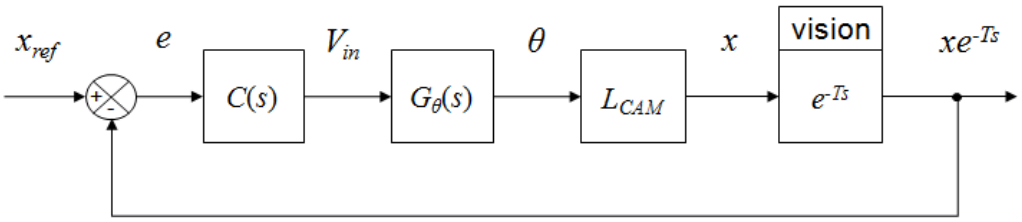


Figure 5.4: Vision-based control system.

The final block in the diagram in Figure 5.4 is used to model the camera in the control loop. In this setup, the camera is represented by a time delay of T seconds associated with the sensor's low frame rate and long image processing times. However, additional elements must be analyzed in order to properly understand the effects of the camera dynamics on the overall system. Let Δt_c be the sampling period of the controller given in seconds, and $f_c = 1/\Delta t_c$ be the controller's operating frequency given in Hz. Based on the information given by the manufacturer, the camera image acquisition sampling period is known to have a value of Δt_{CAM} seconds. Hence, $f_{CAM} = 1/\Delta t_{CAM}$ is the frequency at which images are acquired. The highest value between f_c and f_{CAM} is taken as the nominal frequency of the system.

Two case studies are analyzed based on how the images acquired by the camera are handled. The first case is the *single CAM- multiple image processing threads*-configuration. This case is shown in Figure 5.5, where all the images acquired by the camera are stored in a buffer that operates as a queue (based on the first-in first-out - FIFO - principle), without losses. The images are then processed by an array of graphic processing units (GPU), operating in parallel. In general, the vision block in either one of

the case studies consists of a digital camera, a buffer or a stack and one or multiple GPU units.

In the first case, if $f_{CAM} = f_c$ and the number of parallel GPU units is enough to maintain an update rate equal to f_{CAM} in the feedback signal, then the feedback signal is only affected by an initial delay associated with the time required to process the first image. In Figure 5.5 this delay is represented in the discrete domain by k samples. If $f_{CAM} < f_c$, the controller frequency f_c is taken as the system's nominal frequency and the feedback signal from the vision block is not only delayed, but is also intermittent with respect to the controller's output signal.

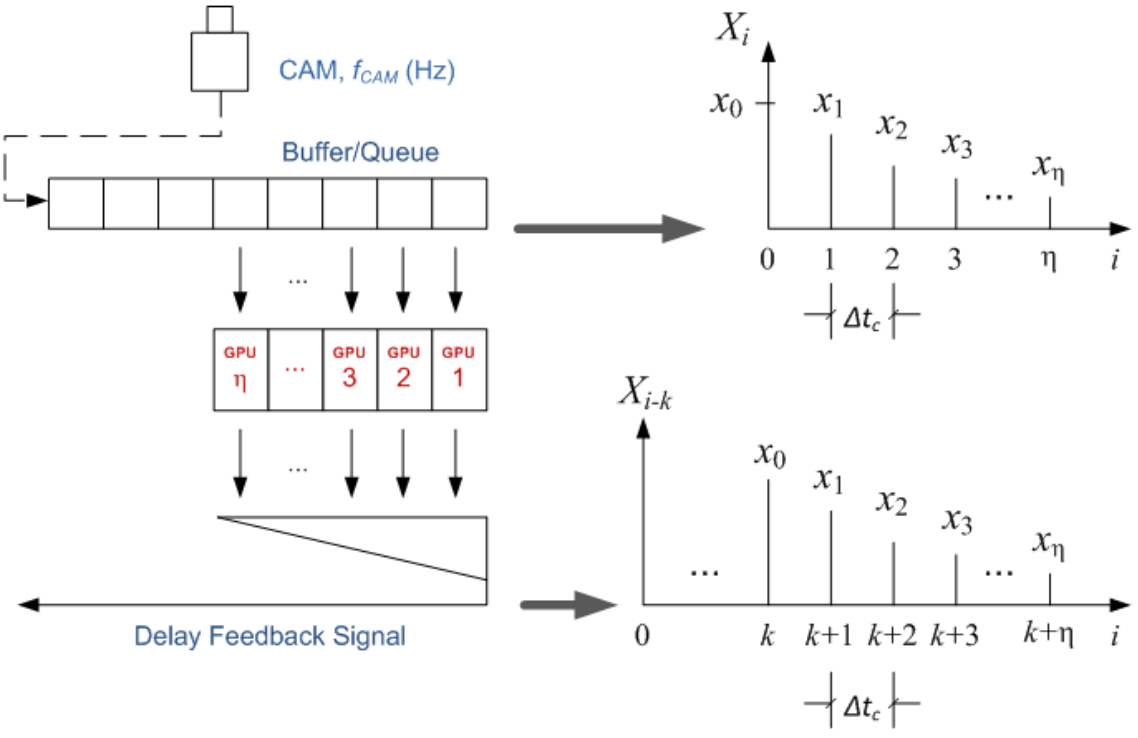


Figure 5.5: *Single CAM- multiple image processing threads*-configuration. The feedback signal is delay in time, but has an updated rate equal to the controller's frequency; i.e. it is continuous with respect to the controller's output signal.

The second case under analysis is the *single CAM- single image processing thread*-configuration depicted in Figure 5.6, where a single GPU is used to process the images. Though this case further compromises stability of the control system, it is actually one of the most common configurations found in the literature due to hardware availability and low costs. The images acquired by the camera are stored in a stack, as opposed to a queue utilized in the first configuration. Even though the camera is capable of acquiring images at f_{CAM} Hz, the GPU can only process the data and make it available to the control loop at a much lower frequency. This is usually attributed to the complexity of the image processing algorithm and hardware limitations. Then, the GPU reads the stack based on the last-in first-out (LIFO) principle every time it finishes processing one image; all other images that are stored in the stack in between GPU readings are lost. The consequence of this configuration is an intermittent and delayed feedback signal, even when $f_{CAM} = f_c$.

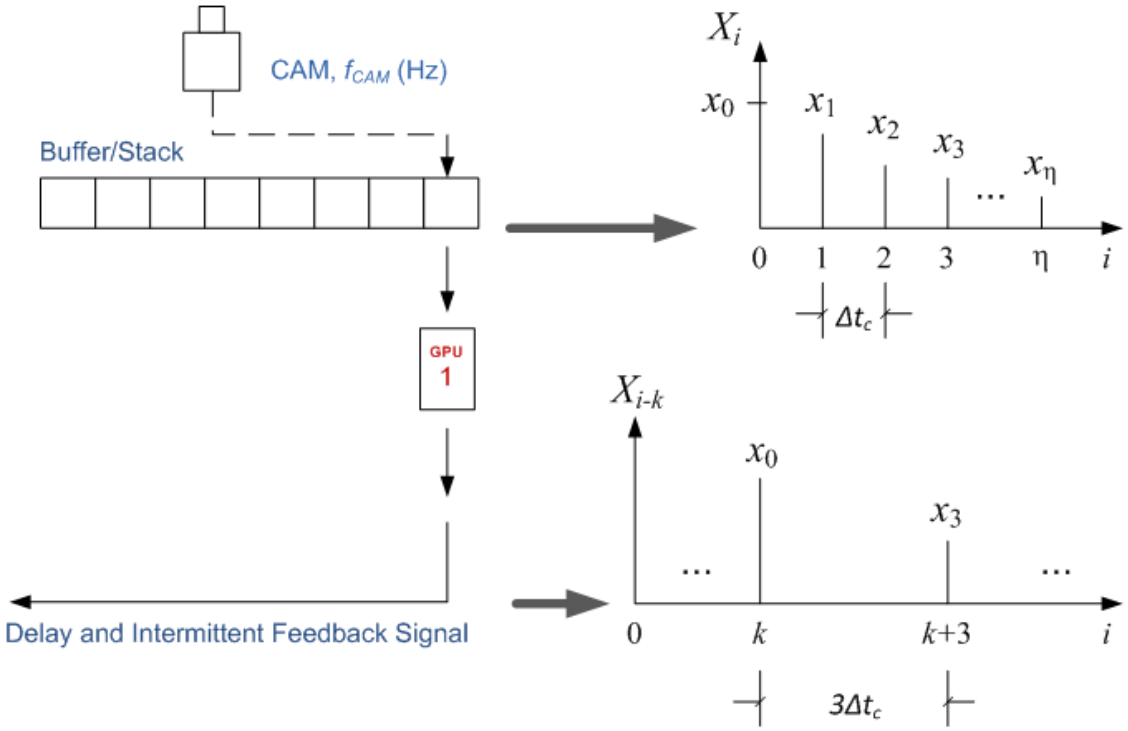


Figure 5.6: *Single CAM- single image processing thread*-configuration. The feedback signal is delay in time and is intermittent with respect to the controller's output signal. In this example the frequency of the feedback signal is three times smaller than f_c.

Based on the two configurations in Figure 5.5 and Figure 5.6, the following two definitions follow.

Definition 5.1: Delay Feedback Signal

> The feedback signal is said to be delayed if the information carried by this signal is recorded at some time t_r, but is only available to the control system at some later time t_a, with $t_a > t_r$. Further, this signal is updated in a discrete matter at the same frequency as the system's nominal frequency, $f_{nominal}$, containing at each discrete point new data.

Definition 5.2: Intermittent Feedback Signal

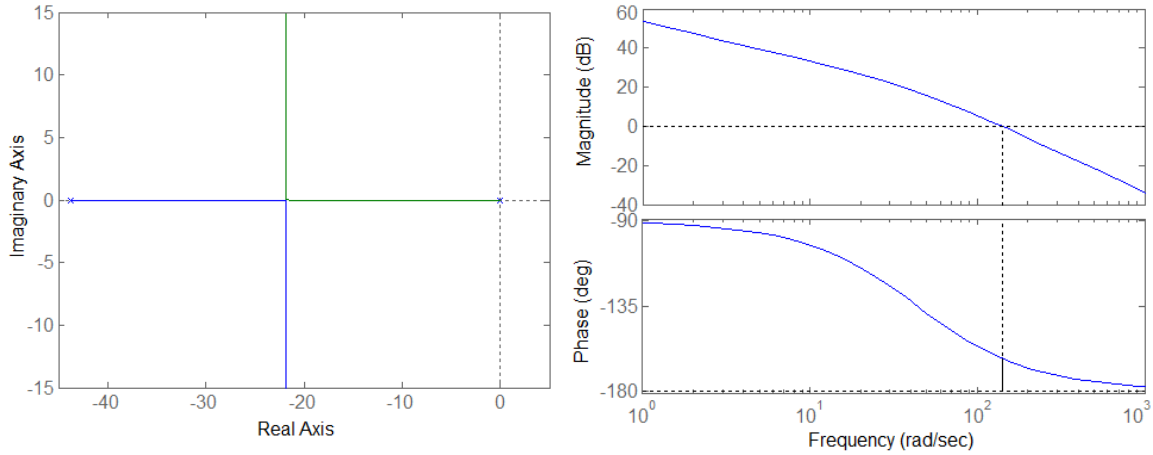
> The feedback signal is said to be intermittent, if the frequency at which this signal is updated is lower than the system's nominal frequency.

Stability Analysis of Time-delay System

Though not always correct, the assumption that the feedback signal is only delayed in time and not intermittent is quite convenient for mathematical analysis and has therefore been followed in most visual servoing literature currently available [61, 77-79]. This assumption is also considered valid in this section in order to isolate and better understand the effects of delays in the control loop. Issues related to intermittencies in the feedback signal, and the multi-frequency nature of the visual servo loop when incorporating a model-based predictor to increase performance, are discussed afterwards.

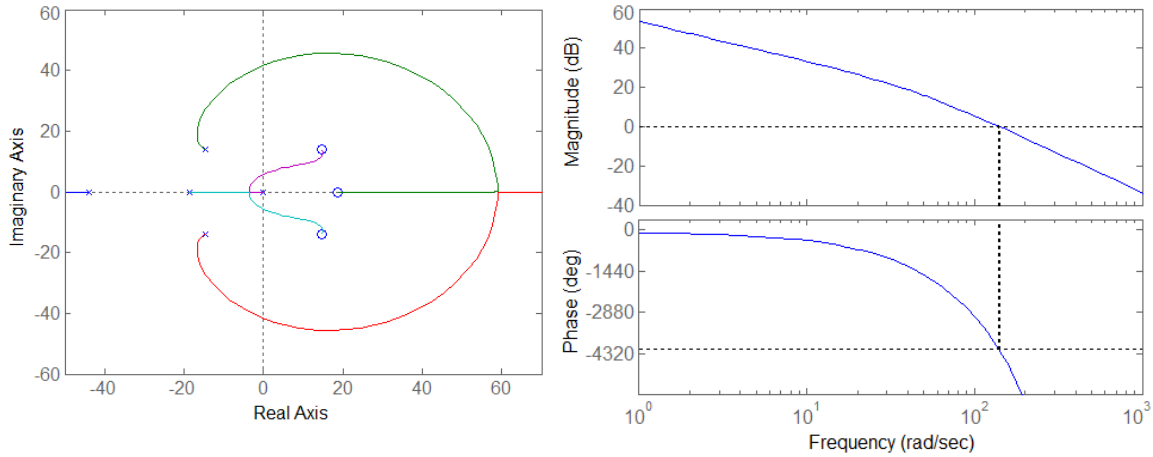
In order to properly examine the consequences of latencies in vision-based control, a mathematical representation of the plant is required. For analysis purposes it is assumed that the experimental model given by (5.10) characterizes the real servo motor with no errors and L_{CAM}, as defined earlier, also represents the coordinate transformation from radians to linear motion on the stage without errors. Figure 5.7 shows the root locus and bode plots for the system of Figure 5.4 without considering the delay and using a proportional controller. The system's roots never leave the left-hand side plane in (a), indicating that the system is stable for any gain value that the controller might take. This result is the same obtained from the bode plot in (b), showing infinite gain margin. The previous analysis is repeated in Figure 5.8 but this time considering the 0.25-seconds time delay. The value for the time delay is chosen based on the offline processing frequencies for the cross-hairs target. The negative gain margin in Figure 5.8 (b), G_m = -0.17dB (0.98 linear units), already indicates that the resulting delayed system is unstable and that a proportional gain smaller or equal to 0.98 should be added to achieve stability.

The delay in the root locus plot of Figure 5.8 (a) is approximated by a third order polynomial using the *pade* function in MATLAB. This approximation yields even worse results, suggesting that the delayed system is stable for proportional gains smaller than 0.0123 (smaller gain before system poles go into the right-hand side plane). It should be noted, though, that such mathematical approximations for simulating time delays are not considered appropriate for assessing system stability [80]; therefore, more importance should be given to the bode plot in Figure 5.8 (b) when evaluating stability.



(a) (b)

Figure 5.7: Experimental non-delayed plant: (a) root locus and (b) bode plot indicating infinite gain margin and P_m=17.3° (at 140 rad/sec).



(a) (b)

Figure 5.8: Experimental plant with a 0.25sec delay: (a) root locus and (b) bode plot indicating G_m=-0.17dB (at 139 rad/sec) and P_m=-41.1° (at 140 rad/sec).

The results presented in Figure 5.8 are confirmed experimentally when operating the stage under closed loop control using the cross-hairs target. Even when testing with a PI controller, with small proportional gains to assure stability and with slightly bigger integral gains to try to more rapidly bring the steady-state error to zero, the overall response is still sluggish and far away from the desired performance levels. It is important to stress that one of the main goals of the present project, aside from achieving the desired resolution levels through MTP, is to successfully create a new position control system to be used in machine tools. Thus, the overall system performance is not only determined by how fast the system reacts to the input reference, but more so by how accurate the resulting output motion is with respect to the reference signal. In fact, a constant velocity lag with respect to the reference signal does not affect the accuracy of the machine motion as long as this lag is maintained constant for all axes. Oscillations in the response, on the other hand, certainly represent an undesired behavior of a cutting

tool and therefore must be avoided. This, of course, implies that the dominant poles of the integrated system should only have negative real parts and no complex parts. One accurate observation that can be made from Figure 5.8 (a), also confirmed experimentally, is that without a proper control strategy the delayed system would present complex-conjugate dominant poles (i.e. oscillations) almost for any proportional controller gain.

Case Study: Model-based Vision Control System using a *Single CAM- Single Image Processing Thread*-Configuration

Single-Axis Position Control using a Smith Predictor Scheme

Time delays in vision-based control systems have commonly been addressed through a Smith predictor scheme (see section 2.2 in [81], chapter 5 in [82] and [80, 83-86]). Figure 5.9 shows the augmented control structure, where $\hat{T}$ is an estimate of the real delay T, and S is the feedback signal. The plant dynamics $G_1(s)$ are modeled through the transfer function $\hat{G}_1(s)$. Setting $x = \hat{X}_1(s)$ and $V_{in} = V_1(s)$, the relationship between the input voltage and the linear displacement of the stage along the X-axis is

$$\hat{G}_1(s) = \frac{\hat{X}_1(s)}{V_1(s)} = L_{CAM}\hat{G}_\theta(s) \tag{5.11}$$

where $\hat{G}_\theta(s)$ is described by (5.5).

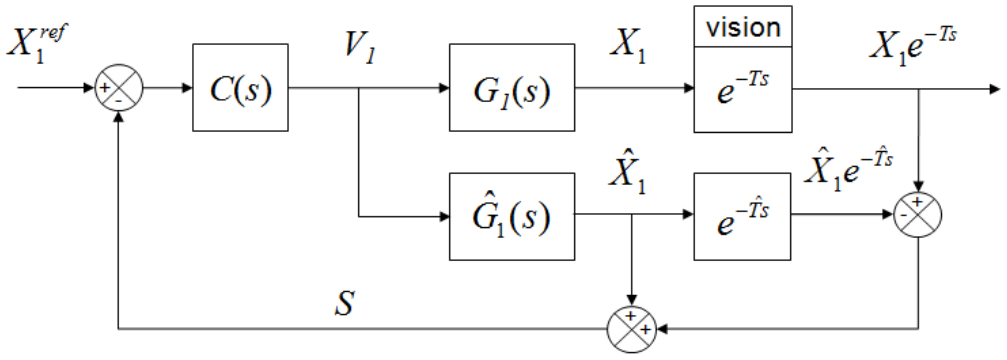


Figure 5.9: Proposed Smith predictor control structure for experimental implementation.

The proposed control scheme is expected to provide a robust basis for implementing MPT. An interesting characteristic about the diagram of Figure 5.9 is that the controller type is not specified. Hence, different controllers can be tested for different applications.

The Multi-frequency Control System

The Smith predictor scheme has been implemented in single-rate systems, where the assumption of a delayed, but still continuous, feedback signal is valid. Among the most relevant in depth analyses on the multi-frequency phenomenon (considering both the controller's- as well as the vision block-frequencies) in visual servoing are the ones presented in [51, 87, 88]. Simulation and experimental results are provided in [88] and [51] to compare the behavior of the single-rate model, operating at the vision block frequency, with the behavior of the multi-rate system (both the controller's- as well as the vision block-frequencies are taken into account). This approach, however, is focused on developing the necessary feedforward and feedback compensators to force the resulting multi-rate system to behave very close to the single-rate one. As such, the latency in feedback signal can indeed be treated as a unitary delay, and therefore it is modeled in the

z-domain through z^{-1}. If the controller's frequency is assumed to be the nominal frequency of operation, then the feedback signal from the vision block is intermittent. It will be shown that for the model-based visual-servo loop, this intermittently updated signal can be set to maintain its last value in-between updates; hence making it continuous with respect to the controller's output signal. This approach is adopted in [89]:

> "In the case of an intermittent feedback, the delayed measurement is available only once every n time steps and model errors can only be corrected every n time steps. One solution is to store the correction factor and use it unchanged until the next measurement becomes available."

In [90] and [91] the authors deal with one type of control system, with similar characteristics as the one currently under analysis. The system is denoted as Model-based Networked Control System (MB-NCS). Given the fact that a network is used as the means to carry the feedback data, there exists a desire to reduce the amount of information that is sent over the network to prevent overload. Hence, two periods are purposely defined: one period when the feedback loop is not closed and the system operates based on the synthetic signal generated by the model, and a second period when the loop is closed and the data collected from the actual sensors is sent over the network and is used as the feedback signal. The system is therefore considered to be intermittent, although during both operating periods the feedback signal is continuous with respect to the controller's output signal (taken as the reference signal). Stability of MB-NCS is demonstrated in these documents. However, the nature of MB-NCS differs from the system under analysis in that, in the former, the intermittent pattern is purposely introduced into the system and the feedback signal (whether actual or synthetic) is always

continuous with respect to the controller's output signal. In the current case study only the synthetic feedback signal is continuous whereas the actual feedback signal, provided by the vision block, is intermittent. The intermittent pattern is not deliberately added, but is inherited from the proposed architecture.

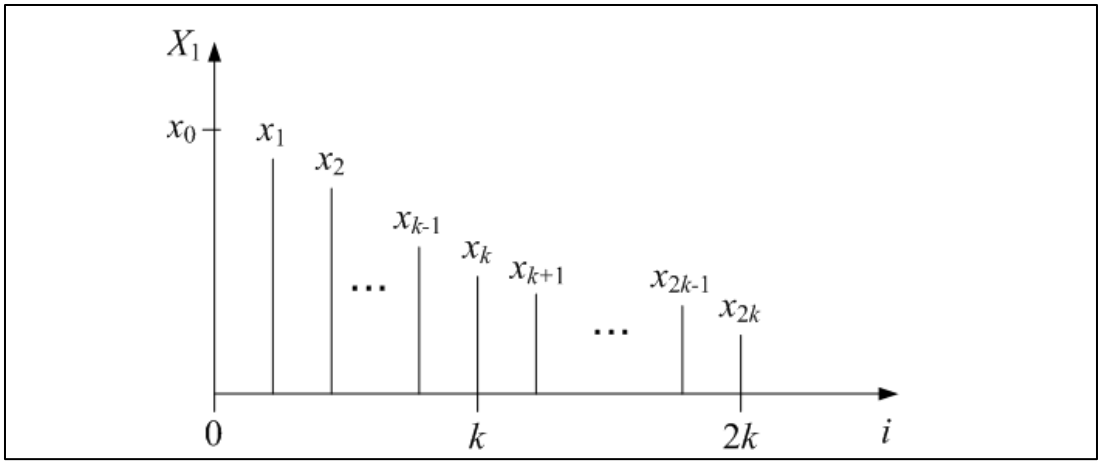
Previously, the controller's sampling period was denoted as Δt_c, given in seconds, and $f_c = 1/\Delta t_c$ was defined as the controller's operating frequency in Hz. For the configuration under analysis, the time required by the vision block to acquire and process a single target image is taken as one and is assumed to have a constant value of Δt^{Ac+P} seconds. Hence, the vision block update frequency is defined as $f_{V.B.} = 1/\Delta t^{Ac+P}$ Hz. The frequency at which the camera is capable of acquiring and storing images in the stack, f_{CAM}, is irrelevant in the *single CAM- single image processing thread*-configuration as long as this frequency is at least double the frequency at which the GPU processes the digital images. Although one is likely to think that this condition is set to prevent aliasing when tracking the target on the LCD, this is actually not true as this ratio only applies as a precautionary measure between the camera sampling frequency and the GPU stack reading-frequency, and has nothing to do with the frequency at which the target image changes on the LCD. In Chapter Six it will be demonstrated that the aliasing problem is completely avoided in the experimental setup by updating the initial conditions of the model $\hat{G}_1$ every time the target image is displaced on the LCD. It is assumed now that $\Delta t^{Ac+P} = Q\Delta t_c$, where $Q > 1$ is a real constant, and therefore $f_{V.B.} = f_c/Q$ given in Hz. The first two graphs in Figure 5.10 ((a) and (b)) show the real position of the plant, X_1, and

the estimated position from the model, $\hat{X}_1$, sampled at the controller's frequency, f_c, and plotted against the discrete variable *i*. In the absence of delays and in an ideal single rate system the feedback signal, *S* in Figure 5.9, would be $s_i = \hat{x}_i + \left(x_i - \hat{x}_i\right) = x_i$. The real position, however, obtained from the camera is available only every *k* samples, and provides information about the plant *k* samples late. Then, for instance, at $i = k$ the estimated feedback signal, $\hat{S}$, is $\hat{s}_k = \hat{x}_k + \left(x_0 - \hat{x}_0\right)$ (Figure 5.10 (c)). The approximate correcting factor $\left(x_0 - \hat{x}_0\right)$ is applied during the entire interval $i \in \left[k, 2k\right)$ until a new feedback point is generated by the vision block at $i = 2k$. If the approximation is maintained, then in general

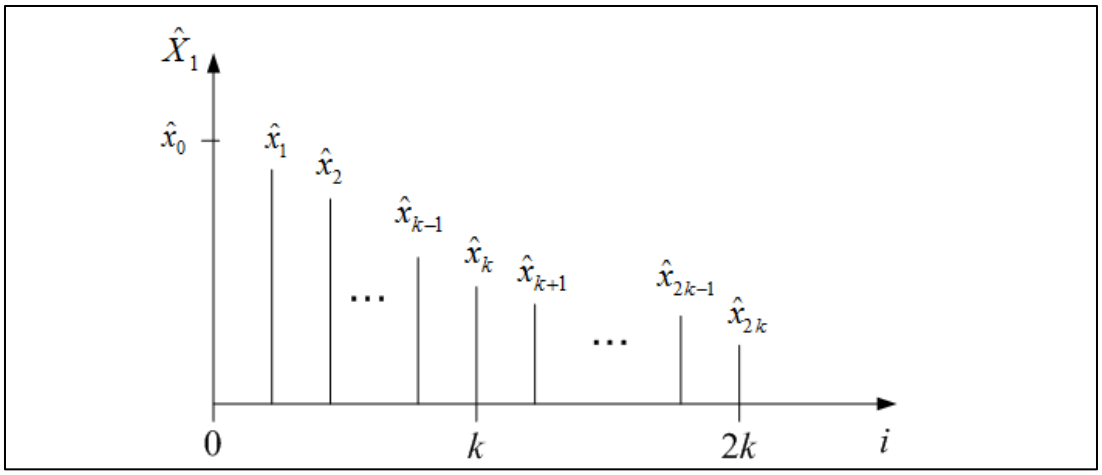
$$\hat{s}_i = \hat{x}_i + \left(x_{(N-1)k} - \hat{x}_{(N-1)k}\right) \tag{5.12}$$

where $i \in \left[Nk, (N+1)k\right)$ and $N \in \mathbb{N}_0$. To the author's best knowledge, stability of the model-aided vision servo loop, affected by delays and intermittencies in the vision block feedback signal, has not been studied in the past. Stability of the overall system during $i \in \left[Nk, (N+1)k\right)$, denoted here as the faster time loop and corresponding to operation at the controller's frequency in-between vision block updates, is demonstrated next. Stability of the overall system over the slower time loop *Nk* (for $N \in \mathbb{N}_0$), where the transition of *i* from $i \in \left[Nk, (N+1)k\right)$ to $i \in \left[(N+1)k, (N+2)k\right)$ must be considered, is left as future work.

(a)



(b)



(c)

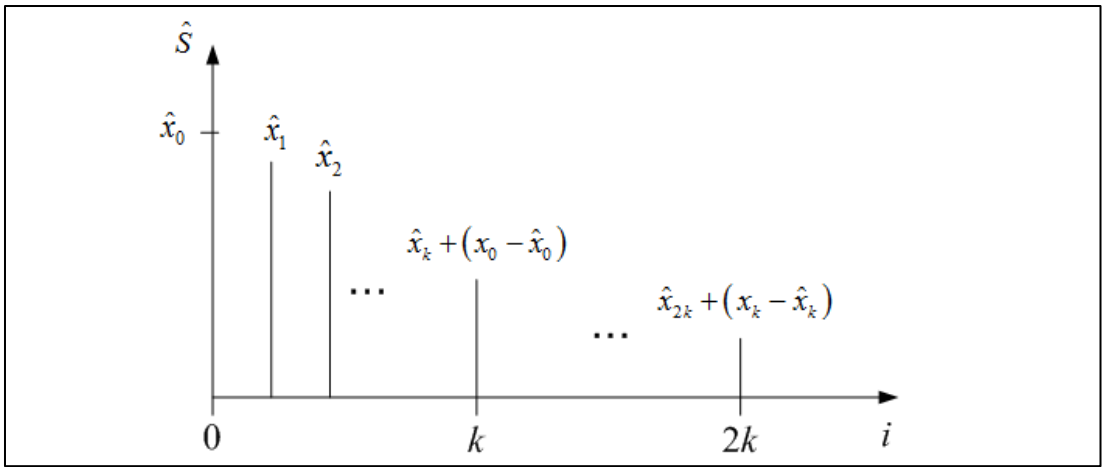


Figure 5.10: Signals sampled at f_c Hz: (a) Real position, (b) estimated position and (c) estimated feedback signal.

Let the error between the model output and the real plant be defined in the discrete domain as $d_i = x_i - \hat{x}_i$. Hence, during the interval $i \in [Nk, (N+1)k)$, $\hat{s}_i = \hat{x}_i + d_{(N-1)k}$ (Figure 5.11).

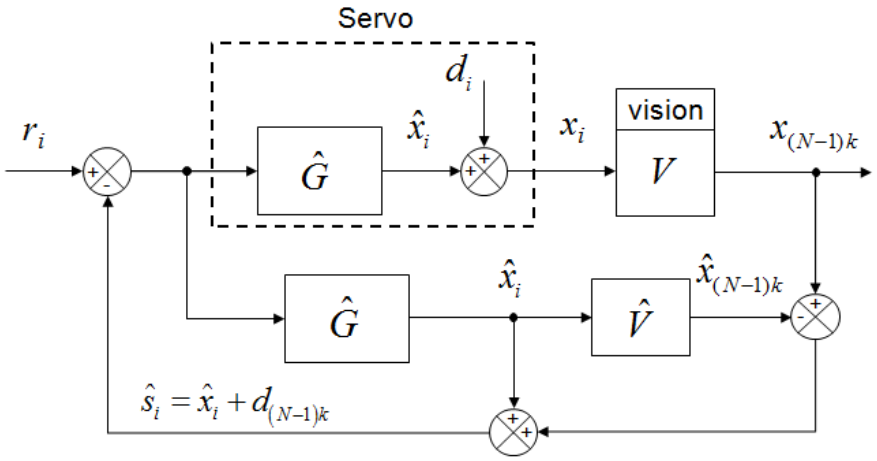


Figure 5.11: Smith predictor control scheme without the controller.

Now, let the model in Figure 5.11 be described in the discrete domain by

$$\begin{bmatrix} \hat{x}_i \\ \hat{x}_{i+1} \end{bmatrix} = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} \\ \hat{a}_{21} & \hat{a}_{22} \end{bmatrix} \begin{bmatrix} \hat{x}_{i-1} \\ \hat{x}_i \end{bmatrix} + \begin{bmatrix} \hat{b}_0 \\ \hat{b}_1 \end{bmatrix} u_i \tag{5.13}$$

with parameters $\hat{a}_{11} - \hat{a}_{22}, \hat{b}_0, \hat{b}_1 \in \mathbb{R}$, and input variable u. Under closed loop operation, the input variable becomes $u_i = r_i - \hat{s}_i = r_i - \left(\hat{x}_i + d_{(N-1)k}\right)$ and therefore

$$\begin{bmatrix} \hat{x}_i \\ \hat{x}_{i+1} \end{bmatrix} = \hat{\mathbf{A}}_{aug} \begin{bmatrix} \hat{x}_{i-1} \\ \hat{x}_i \end{bmatrix} + \hat{\mathbf{B}}_{aug} \begin{bmatrix} r_i \\ d_{(N-1)k} \end{bmatrix} \tag{5.14}$$

with

$$\hat{\mathbf{A}}_{aug} = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} - \hat{b}_0 \\ \hat{a}_{21} & \hat{a}_{22} - \hat{b}_1 \end{bmatrix}; \quad \hat{\mathbf{B}}_{aug} = \begin{bmatrix} \hat{b}_0 & -\hat{b}_0 \\ \hat{b}_1 & -\hat{b}_1 \end{bmatrix} \tag{5.15}$$

Bounded-input bounded-state stability of the closed-loop system in (5.14) is only guaranteed if the Eigen-values of $\hat{\mathbf{A}}_{aug}$ lie inside the unitary circle in the complex plane, provided that both inputs r_i and $d_{(N\text{-}1)k}$ are bounded. If the previous conditions are satisfied, the output of the plant defined as

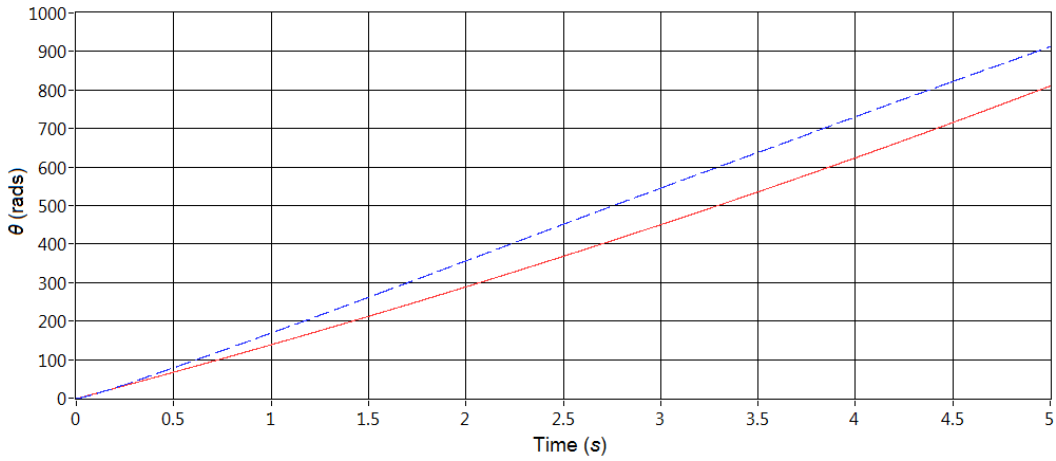
$$x_i = \hat{x}_i + d_i \tag{5.16}$$

will also be stable, again under the assumption that d_i is bounded during $i \in \left[Nk, (N+1)k \right)$. It can be concluded that stable operation of the multi-rate system proposed here, requires the system identification method to guarantee that the error between the real output and the estimated output remains bounded; this implies $d_i \in D$, D an invariant set. If the controller is also considered, then the Eigen-values of the augmented closed-loop model, including the controller, must lie within the unitary circle in the complex plane (for a discrete system). Based on the definition it should also be noted that in the absence of delays the sub-optimal feedback signal, $\hat{S}$, approaches the ideal feedback signal, *S*, as $Q \rightarrow 1$ (where *Q* defines the relationship $\Delta t_{CAM} = Q\Delta t_c$). This happens when the frequency of the vision block approaches the controller's frequency. For the feedback signal this implies $\lim_{Q \to 1} \hat{S} = S$.

An experimental test is conducted to validate the performance of the system identification procedure in the first section of this chapter - Stage Modeling and System Identification. The real servo motor and the model given in (5.10) are excited by a step

input voltage $V_{in}(t) = 5V, t > 0$ for a period of 5 seconds. The two output signals are shown in Figure 5.12 (a); the absolute difference between these signals (the error) is shown in (b). The plot in Figure 5.12 (b) suggests that the experimental steady-state error does satisfy the boundedness condition as, after some time (around 4 seconds), it stops growing and seems to remain below an upper bound.

(a)



(b)

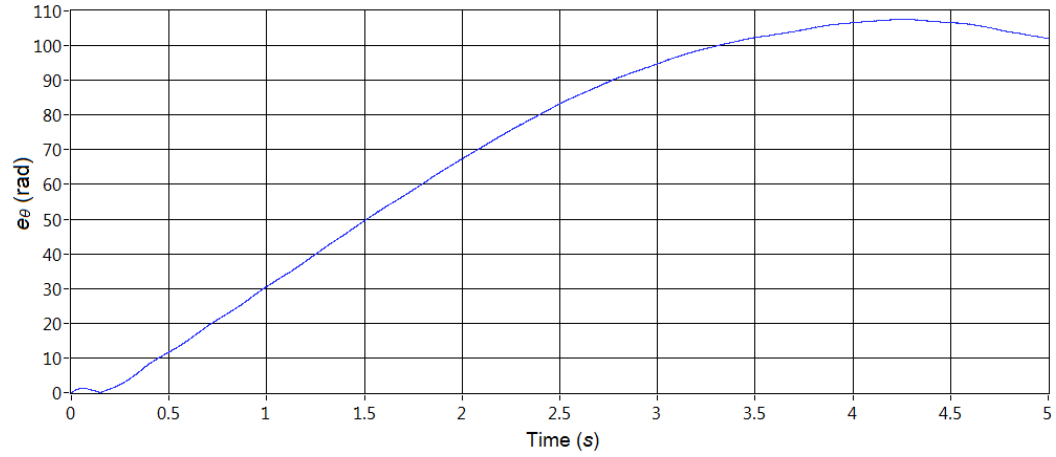


Figure 5.12: (a) Step response: plant (dashed line) and *ARX* model (continuous line). (b) Error signal, e_θ, calculated from step response.

Concluding Remarks

The functionality of the Smith predictor scheme in visual servo-mechanisms has been described, including both analytical and practical details. Common dynamic problems present in most visual servo systems, such as such as delays and discontinuities in the feedback signal associated with the imaging sensor, were isolated and analyzed. Moreover, the requirements for stability were presented.

The approximations and assumptions made in this chapter proved to be valid from a stability stand point. However, the implications of these in the performance of high resolution position control applications are still to be determined. These aspects are studied in Chapter Six.

CHAPTER SIX

COMMAND ISSUING PROTOCOLS FOR HIGH RESOLUTION IN-PLANE POSITION CONTROL

This chapter presents two new methods for generating motion commands using MPT Method 1, explained in Chapter Four. A distinction is made here between rotational commands and in-plane commands. Namely, rotational commands can be generated by simply setting the reference input signal to the control system to the desired value, not by rotating the displayed target pattern on the LCD screen. In-plane motion instructions, on the other hand, are given by moving or modulating the dynamic target on the LCD, and thereby creating a position error between the target and the principal point in the camera plane. The focus of this chapter is set on developing two command strategies for 2D motion, which are expected to optimize the use of the direct sensing capabilities of the sensor. The structure of the integrated single-axis control system, in the discrete domain, is shown in Figure 6.1. It is important to stress that the metrics for measuring system performance in visual servoing are not necessarily the same as those used in regular control loops [52]. Performance of regular control systems has been commonly determined based on bandwidth. Performance of model-aided vision-based systems is usually measured through other parameters, given the known bandwidth limitations of such systems. The parameter selected for assessing system performance in the proposed position control system aided by visual feedback is accuracy of output motion with respect to the desired trajectory.

A relevant observation to be made about visual-aided robotics, where the camera is mounted at the end effector of the tool (also denoted as in-hand camera robotics), is that the camera sees the position of the tool relative to the current fixed position of the object (see for example [92]). Motion of the robot does not affect the absolute position of the object being imaged. Considering the fact that the LCD is mounted on the moving stage, in the experimental setup presented in Chapter Four, it would be wrong to assume that the absolute position of the target displayed on the LCD remains constant. In this case, it is actually the reference coordinate system (the camera) the one that remains constant in the absence of rotations (θ_Z = 0°), and the camera sees $\overline{x}_i$ (see Figure 6.1) where

$$\overline{x}_i = x_i^{Target} - x_i \tag{6.1}$$

x_i^{Target} in (6.1) is the displacement of the target referenced to the LCD frame, with $x_0^{Target} = 0$; x_i is the position of the stage with $x_0 = 0$. The analysis presented here is one-dimensional (only the *X*-axis is considered); however, the same results and considerations apply to both axes. From (6.1) it is clear that the displacement of the stage is governed by

$$x_i = x_i^{Target} - \overline{x}_i \tag{6.2}$$

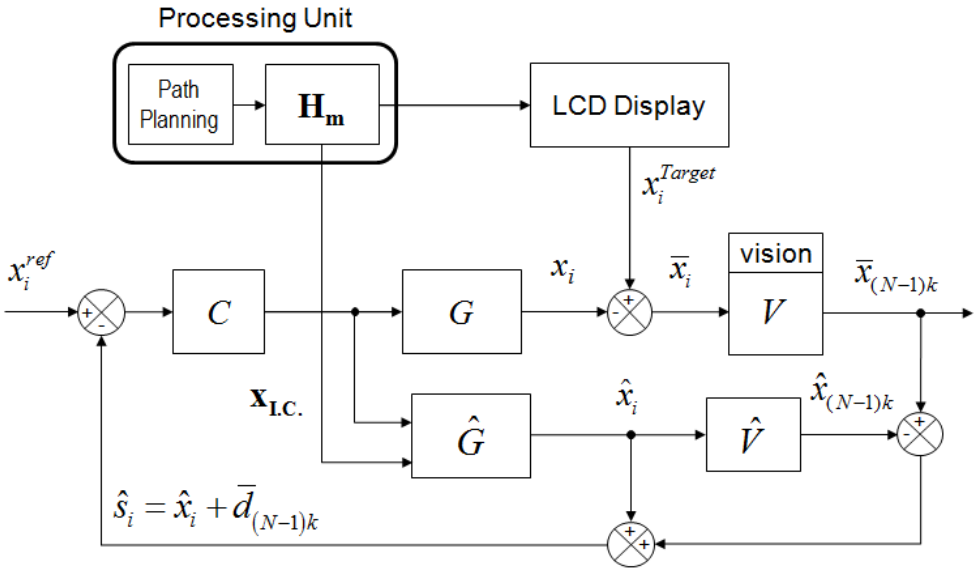


Figure 6.1: Proposed Smith predictor control structure for experimental implementation of MTP, where $i \in \lfloor Nk, (N+1)k \rfloor$ and $N \in \mathbb{N}_0$.

Experimental results in Chapter Four demonstrated that sensor resolutions on the order of 2.5 µm can be reliably achieved through MPT Method 1. However, the smallest displacement increment in the position of the target on the LCD is limited by the size of one LCD pixel (or 1/3 of an LCD pixel, when considering horizontal displacement commands), which is considerably larger than the achievable resolution. Therefore, the primary goal is to command stage displacements as small as 2.5 µm, under the constraint that the value of x_i^{Target} in (6.2) is always a multiple of one LCD pixel. Two methods are presented next for generating displacement commands as small as 2.5 µm. The functionality of the first method, denoted as Embedded Digital Coding Protocol, is almost the same as that of a regular visual-tracker; i.e. the reference input signal in Figure 6.1, x_i^{ref}, is always fixed to the principal point on the image plane. A digital code, embedded in the cross-hairs target image, is used to represent a desired offset value that is added to

$\bar{x}_i$. The second method combines visual-tracking and regular servo control, where the former implies the tracking of the target on the LCD and the latter implies a non-fixed reference. This method is denominated as Hybrid Command Generation Protocol. Other relevant modeling and synchronization considerations are also contemplated in this chapter, which lead to full MTP implementation.

Command Issuing Protocols

Embedded Digital Coding Protocol

The idea behind this method is to combine both the analog cross-hairs signal with a digital code embedded in the target image. The position of the target is still determined through MPT Method 1 and the control system operates as a tracking device.

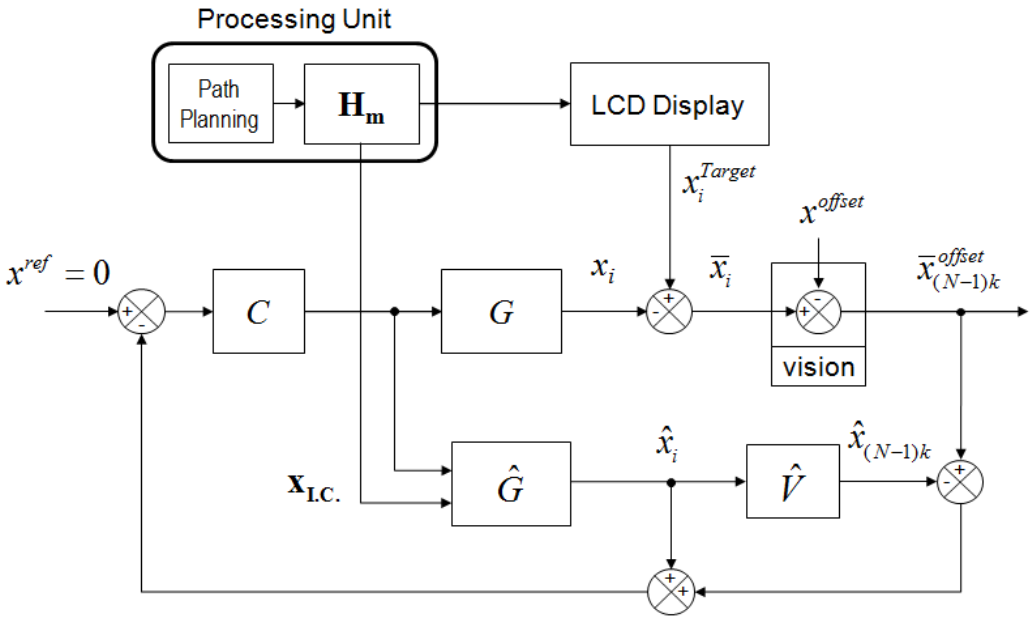


Figure 6.2: Single axis command issuing through image-embedded digital offset code.

A 2D offset value is purposely subtracted from the position of the target as imaged by the camera, to generate tracking errors smaller than one LCD pixel (or 1/3 of an LCD pixel for horizontal displacement commands), as shown in Figure 6.3. The magnitude of the offset is defined through a digital code embedded in the target image and can be thought of as a signal entering the control loop through the vision block as shown in Figure 6.2. The position of the stage is redefined as

$$x_i = x_i^{Target} - x_i^{offset} - \overline{x}_i \tag{6.3}$$

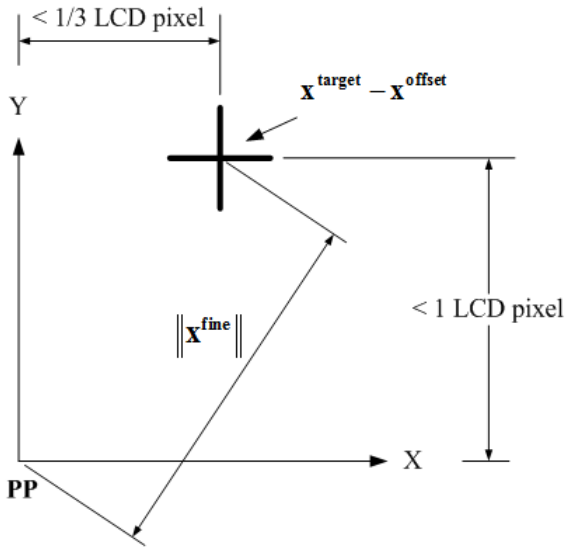
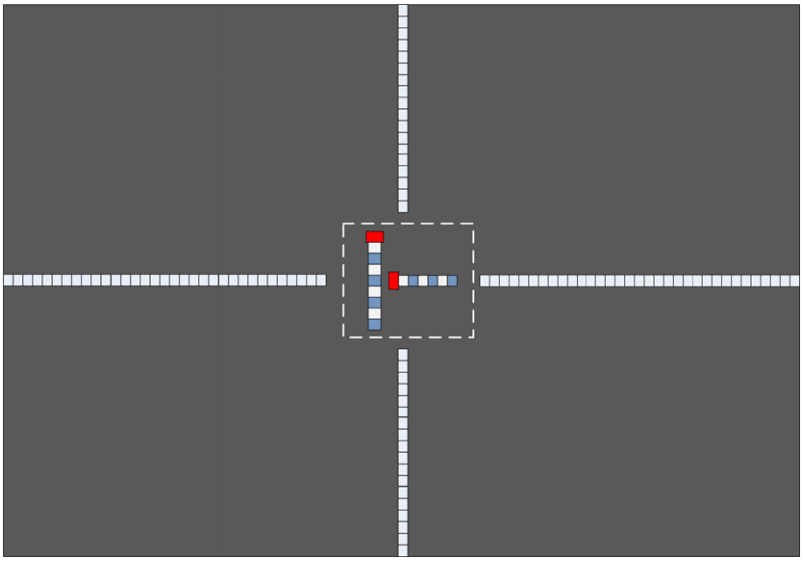


Figure 6.3: Tracking 2D error represented on image plane.

The cross-hairs target in Figure 6.4 (a) presents an embedded digital code in the area enclosed by the dashed line. The offset value for horizontal displacement commands is generated through the array of pixels labeled as "Binary Code – LCD *X*-axis" in Figure 6.4 (b). Similarly, the offset value for vertical displacement commands is generated through the array of pixels labeled as "Binary Code – LCD *Y*-axis". Different colors are

used for illustration purposes to highlight the startbits used to identify the beginning of the sequence, the word-bits used to represent the offset value, and the parity-bits at the end of the sequence used to check for errors.

(a)



(b)

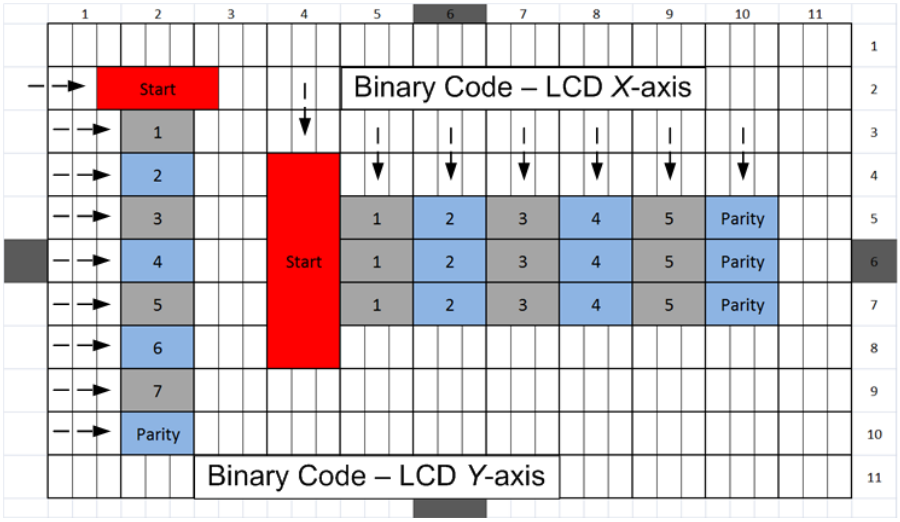


Figure 6.4: (a) Theoretical LCD target with binary code; (b) zoomed-in 11x11 pixel-area highlighted in theoretical target though dashed line.

If the camera orientation is aligned with the LCD, i.e. if $\theta_Z = 0$, then the binary code along the LCD *X*-axis must be read column-wise on the digital image, as demonstrated by the small dashed arrows in Figure 6.4 (b). As with any other digital signal read in time, or in space as is the case here, the first step consists in indentifying the startbits of the sequence. Once the startbits have been identified (read), the remaining bits on the signal are read in discrete steps proceeding from left to right as illustrated by the dashed arrows on the same figure. The most significant bit follows right after the startbits. The size of the discrete step that determines the location on the image where the next reading must be performed (i.e. the spacing between the dashed lines in Figure 6.4 (b) where a bit-reading occurs) is based on the LCD pixel size and the optical magnification of the vision system; therefore these values must be known a priori. The binary code along the LCD *Y*-axis, on the other hand, must be read row-wise on the digital image assuming $\theta_Z = 0$. As shown in Chapter Four, an estimated value of the target's orientation with respect to the camera coordinate system is known before the entire image is processed. Hence for $\theta_Z \neq 0$, the reading of the digital code must simply follow the estimated target orientation.

It is required to represent offset values that are multiples of 2.5 μm, the achievable sensor resolution. The total number of pixels needed to generate the offset depends on the LCD pixel geometry. For horizontal displacement commands it is known that the smallest displacement of the target on the LCD is 1/3 of an LCD pixel. Therefore, offset values in (6.3) should range from 1/3 of a pixel to 2.5 μm. For the case of vertical displacement commands, the range of offset values must cover from 1 LCD pixel to 2.5

μm. Table 6.1 shows the number of bits required to span the experimental ranges, for an LCD pixel size of 294x294 μm.

Table 6.1: Digital code range.

	Offset Range [μm]	Experimental Pixel Size (294x294 μm)	Required # of codes (rounded to biggest integer)	Minimum # of bits
X-axis	1/3 LCD pixel - 2.5 μm	98 – 2.5 = 95.5 μm	Int(95.5/3) = 32	5
Y-axis	1 LCD pixel - 2.5 μm	294 – 2.5 = 291.5 μm	Int(291.5/3) = 98	7

The intensity modulation corresponding to each binary code is presented in Figure 6.5. Every binary code consists of a sequence of bits which are never the same. The startbits in Figure 6.5, which are always the same and act as a reference for the readings, require 5 pixels (bits). Binary codes 1 and 0, used to represent the parity-bits and the commands ranging according to the values in Table 6.1, only require 3 pixels (bits).

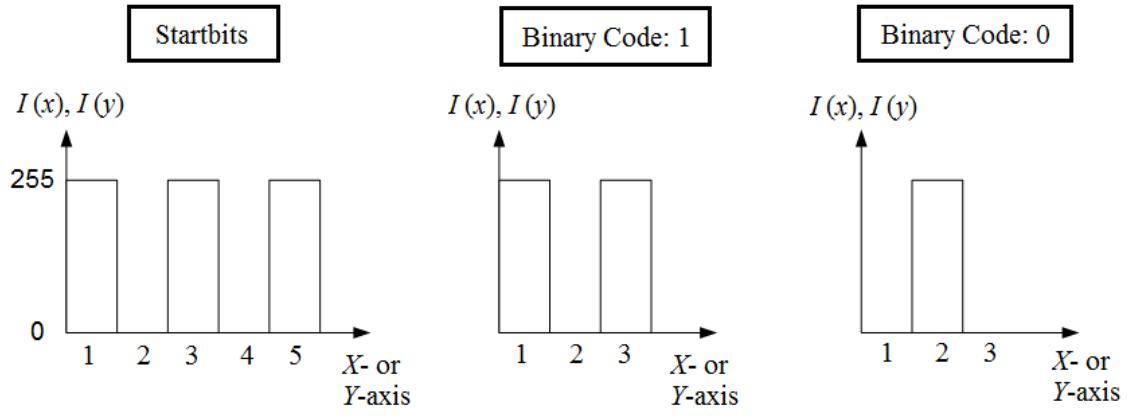


Figure 6.5: Intensity modulation.

Excluding the startbits each binary code read column-wise for the digital command expressed along the LCD *X*-axis, is made of three pixels, each of them

corresponding to a bit (Figure 6.4 (b)). Again excluding the startbits, each binary code read row-wise for the digital command expressed along the LCD *Y*-axis, is made of a single pixel; the three stripes of which correspond to a bit. This topology follows, of course, from the physical characteristics of LCD pixels, explained in Chapter Four. Consequently, the startbits for the LCD *X*-axis digital code require 5 pixels (5 bits), whereas the startbits for the LCD *Y*-axis digital code require at least 2 pixels, or 3 if the code is centered at given pixel column as shown Figure 6.4 (b). The parity-bits should be set to binary code 1 if the number of binary ones in the offset word is even; the parity-bits should be set to binary code 0 otherwise. Parity bits are commonly used for error detection in the reading of digital signals.

Hybrid Command Generation Protocol

In the hybrid command generation protocol, commands equal to a multiple of one LCD pixel (or 1/3 of a pixel for the case of horizontal displacement commands) are issued by moving the target on the LCD by that specific distance, and setting the reference on the image plane to the origin (i.e. PP). In this case the control system operates as a regular tracking system. For commands smaller than one LCD pixel or not equal to an integer multiple of pixels, the target is displaced by the smallest integer number of pixels that is larger than the desired commanded displacement and the reference signal is set to a value different than PP; the value of the reference is less than 1 LCD pixel size. Figure 6.6 shows the proposed control structure, where the reference signal to the control system is defined as $x^{ref} = x^{Target} - x^{fine}$. The value x^{fine} is the desired

displacement command, where $\left|x^{fine}\right|<1$ LCD pixel. Convergence of x (position of the stage) to x^{fine} when setting $x^{ref}=x^{Target}-x^{fine}$ is demonstrated in the next section.

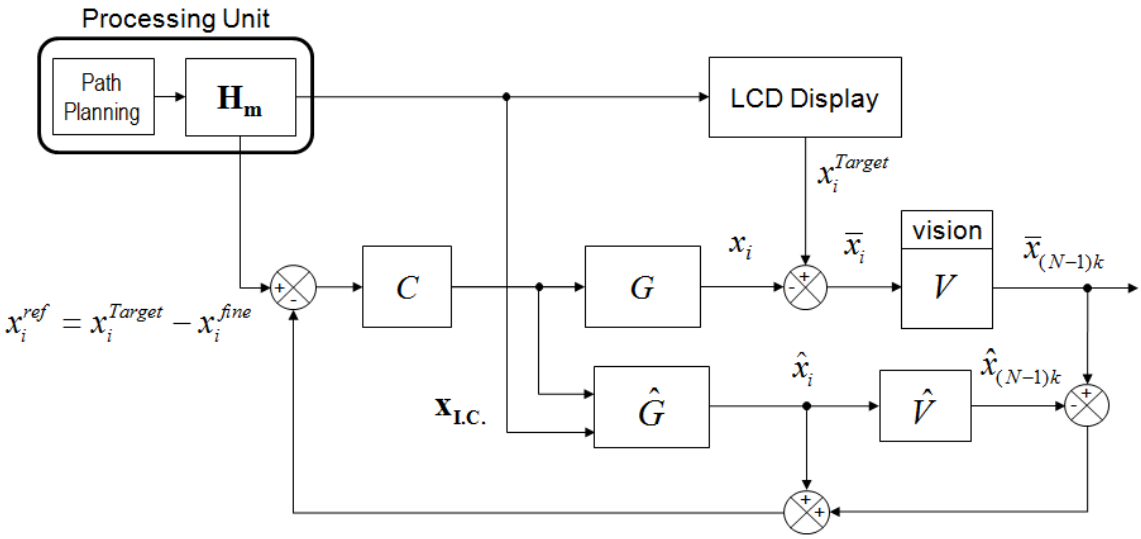


Figure 6.6: Single axis command issuing through hybrid protocol.

Consider, for example, an array of 2D displacement commands $\mathbf{X^{fine}}=\left\{\mathbf{x_1^{fine}},\mathbf{x_2^{fine}},\mathbf{x_3^{fine}},\mathbf{x_4^{fine}},\mathbf{x_5^{fine}}\right\}$ as the one in Figure 6.7. The time diagram in Figure 6.8 shows the times at which each command in the 2D displacement array is issued to the control loop for both protocols (e.g. $\mathbf{x_1^{fine}}$ is issued at t_{x1}). Commands issued through the embedded digital coding protocol are read only after the vision block has acquired and processed the corresponding target image; in the diagram this equals to Δt^{Ac+P} seconds. This means that fine commands in the embedded digital coding protocol enter the control loop through the vision block and are threfore issued at the vision block frequency. Commands issued through the hybrid method enter the control loop through the reference

signal; this permits operation at the controller's frequency during the processing of the entire set.

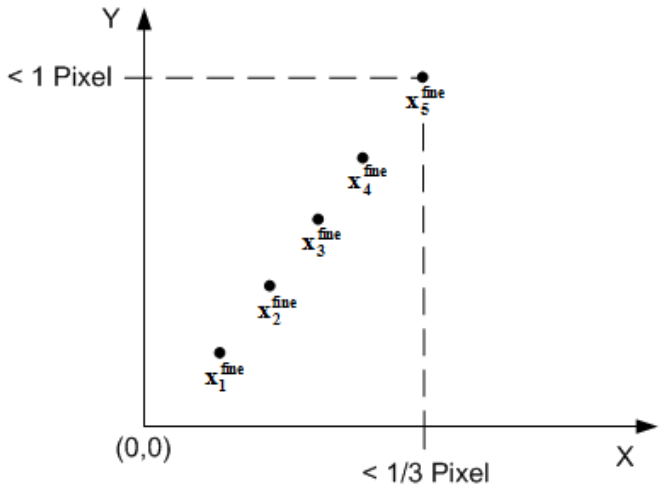


Figure 6.7: Array of 2D displacement commands composed of five commands with entries smaller than 1/3 of an LCD pixel in *X*, and 1 full pixel in *Y*.

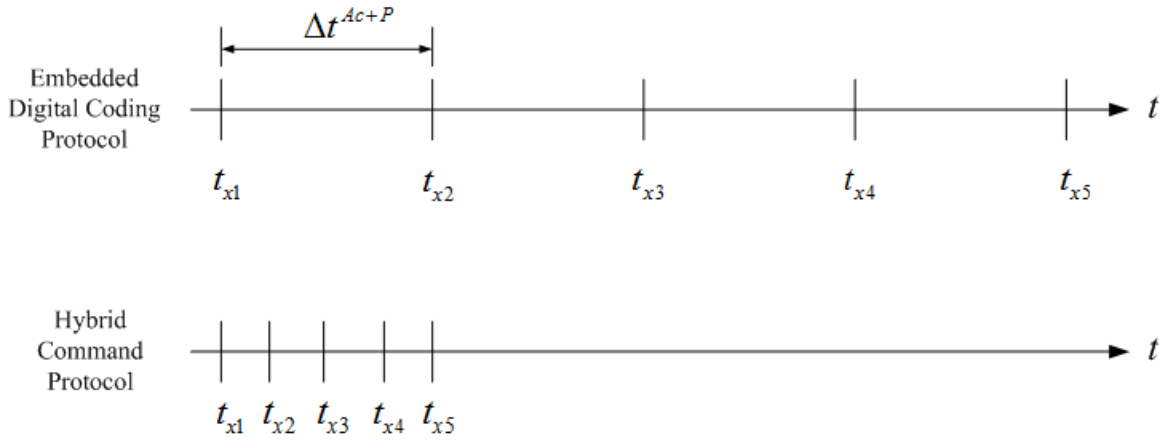


Figure 6.8: Comparison of two time diagrams associated with the two new command generation methods when issuing the 2D displacement commands shown in Figure 6.7.

$\mathbf{H}_m$, in Figure 6.2 and Figure 6.6, is used to calculate vectors $\mathbf{x}^{Target}$ and $\mathbf{x}^{fine}$, according to a predefined path plan. Simulation results for both methods are presented

later in this chapter to demonstrate what vectors $\mathbf{x}^{\mathbf{Target}}$ and $\mathbf{x}^{\mathbf{fine}}$ look like for a generic path plan. Experimental results are also presented for in-plane motion control using the hybrid command generation protocol, only. Before this, it is important to identify the necessary conditions for the model to provide adequate estimates of the plant states at each point in time, for both methods. At the same time, full implementation of MTP requires dedicated hardware equipment running independent processing threads. These threads share information over a dedicated network, and must be synchronized for correct operation. The next two sections cover some important details on modeling and the multi-thread processing configuration required for experimental implementation of MTP.

Modeling Considerations

The implications of delays and intermittencies in the feedback signal were studied in detail in Chapter Five and will not be considered in the rest of the current chapter. Full attention is now paid to introducing the necessary adaptations to make the estimated value $\hat{x}_i$ behave as $\overline{x}_i$ in Figure 6.1.

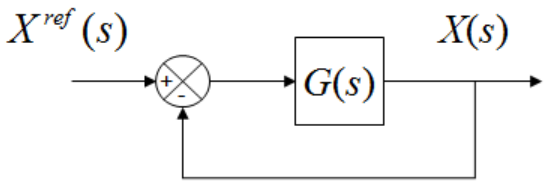


Figure 6.9: Single-axis servo motor operating under closed-loop control.

Let a system, utilized for closed-loop position control of a servo motor, be represented by the block diagram in Figure 6.9. From Chapter Five (see (5.5) and (5.11)) it is known that

$$G(s) = \frac{X(s)}{V(s)} = \frac{b_0}{s^2 + a_1 s} \tag{6.4}$$

with $b_0, a_1 \in \mathbb{R}$. Therefore

$$\frac{X(s)}{X^{ref}(s)} = \frac{b_0}{s^2 + a_1 s + b_0} \tag{6.5}$$

The differential equation associated with (6.5) is

$$\ddot{x}(t) + a_1 \dot{x}(t) + b_0 x(t) = b_0 x^{ref}(t) \tag{6.6}$$

with $\dot{x}(t) = dx/dt$, $\ddot{x}(t) = d^2x/dt^2$. The general solution to (6.6) is

$$x(t) = C_0 e^{r_1 t} + C_1 e^{r_2 t} + x^{ref} \tag{6.7}$$

with $C_0, C_1, r_1, r_2 \in \mathbb{R}$, $r_1, r_2 \leq 0$, $r_1 \neq r_2$, assuming a stable non-oscillatory system.

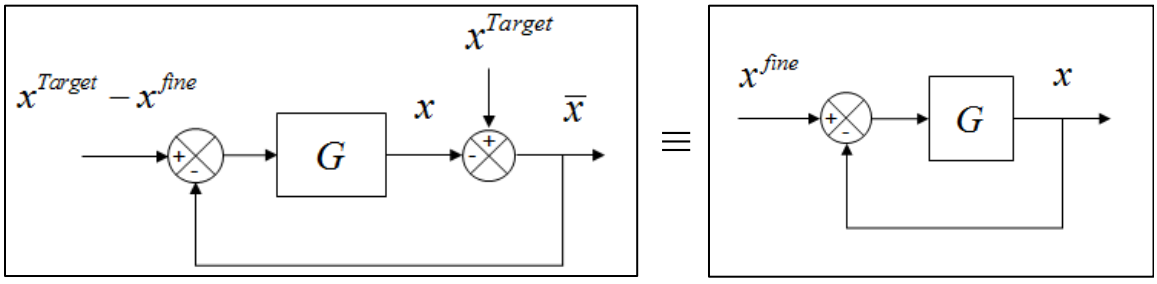


Figure 6.10: Simplified visual servo loop.

For analysis purposes, let the hybrid protocol be chosen for command issuing at this point. Ignoring the Smith predictor, the controller and the effects of the camera

dynamics, the visual servo loop of Figure 6.1, can be represented by either one of the diagrams in Figure 6.10 with $x^{ref} = x^{Target} - x^{fine}$. The value x^{fine} is the desired displacement command, where $\left|x^{fine}\right| < 1$ LCD pixel. The structure of the right-hand side diagram in Figure 6.10 is equivalent to the one in Figure 6.9, and therefore its general solution has the same form as (6.7). Given the initial conditions $x(0) = 0, \dot{x}(0) = 0$, it follows that $C_0 = \left(r_2 / r_1 - r_2\right) x^{fine}$ and $C_1 = \left(r_1 / r_2 - r_1\right) x^{fine}$. Hence,

$$x(t) = x^{fine}\left(\frac{r_2}{r_1 - r_2} e^{r_1 t} + \frac{r_1}{r_2 - r_1} e^{r_2 t} + 1\right) \tag{6.8}$$

Equation (6.8) indicates that the position of the stage along the *X*-axis converges exponentially to x^{fine}, as desired. From the left-hand side diagram in Figure 6.10, it follows that

$$\bar{x}(t) = x^{Target} - x^{fine}\left(\frac{r_2}{r_1 - r_2} e^{r_1 t} + \frac{r_1}{r_2 - r_1} e^{r_2 t} + 1\right) \tag{6.9}$$

The value of $\bar{x}$ is now well defined for the required reference signal $x^{ref} = x^{Target} - x^{fine}$ to command the stage to move by x^{fine} μm. It is now necessary to determine the conditions to make the model $\hat{G}$, in Figure 6.1, provide an appropriate estimate of $\bar{x}$. Let it be assumed that the model $\hat{G}$ satisfies the boundedness condition given at the end of Chapter Five, which guarantees stability of the augmented multi-frequency loop (this assumption could not be contradicted based on the experimental test presented in Figure 5.12 in Chapter Five and is therefore still considered valid). Further,

let the differential equation associated with $\hat{G}$, for a reference signal $x^{ref} = x^{Target} - x^{fine}$, be

$$\hat{x}(t) = \hat{C}_0 e^{\hat{r}_1 t} + \hat{C}_1 e^{\hat{r}_2 t} + \left(x^{Target} - x^{fine} \right) \tag{6.10}$$

with $\hat{C}_0, \hat{C}_1, \hat{r}_1, \hat{r}_2 \in \mathbb{R},\ \hat{r}_1, \hat{r}_2 \leq 0,\ \hat{r}_1 \neq \hat{r}_2,\ \left\| r_1 - \hat{r}_1 \right\|^2 < B1,\ \left\| r_2 - \hat{r}_2 \right\|^2 < B2,\ B1, B2 \in \mathbb{R}^+$. For initial conditions $\hat{x}(0) = x^{Target}$, $\dot{\hat{x}}(0) = 0$, it is possible to calculate $\hat{C}_0 = -\left(\hat{r}_2 \big/ \hat{r}_1 - \hat{r}_2 \right) x^{fine}$ and $\hat{C}_1 = -\left(\hat{r}_1 \big/ \hat{r}_2 - \hat{r}_1 \right) x^{fine}$, and the solution to (6.10) becomes

$$\hat{x}(t) = x^{Target} - x^{fine} \left(\frac{\hat{r}_2}{\hat{r}_1 - \hat{r}_2} e^{\hat{r}_1 t} + \frac{\hat{r}_1}{\hat{r}_2 - \hat{r}_1} e^{\hat{r}_2 t} + 1 \right) \tag{6.11}$$

A direct comparison between (6.9) and (6.11) demonstrates that, if the initial conditions of the model $\hat{G}$ are updated according to the latest position of the target as imaged by the camera, x^{Target}, then $\hat{x}(t)$ is indeed an estimate of $\bar{x}(t)$. It is worth pointing out that x^{fine} can be calculated in advance for all values of $\mathbf{x}^{\mathbf{Target}}$ belonging to a predefined path. The input vectors $\mathbf{x}_{\mathbf{I.C.}} = \left[x(0) \quad \dot{x}(0) \right]^{\mathrm{T}}$ in Figure 6.1, Figure 6.2 and Figure 6.6 illustrate the initialization of the model states. Similarly, for the embedded digital coding method it can be demonstrated that the initial conditions of the model must be set to $\hat{x}(0) = x^{Target} - x^{offset}$, $\dot{\hat{x}}(0) = 0$, in order for $\hat{x}$ to be an estimate of $\bar{x}^{offset}(t)$ in Figure 6.2.

In general a 2-D displacement command, $\mathbf{x}^{\mathbf{target}}$, must consider the orientation of the camera coordinate frame with respect to the stage, θ_Z. Therefore, a displacement

command $\mathbf{x}^{\mathbf{target}}$ to be issued through the cross-hairs target on the LCD requires a displacement of the target image, $\tilde{\mathbf{x}}^{\mathbf{target}}$, referenced to the LCD plane. These two quantities are related through a rotation matrix $\mathbf{R}_{\theta Z}$, i.e. $\mathbf{x}^{\mathbf{target}} = \mathbf{R}_{\theta Z}\tilde{\mathbf{x}}^{\mathbf{target}}$.

Multi-thread Synchronization

It is important to stress that the update in the model states should only occur right after the position of the cross-hairs has changed with respect to the LCD frame. Hence, adequate command issuing through the proposed methods can only be achieved if the host thread, in charge of generating and displaying the dynamic target, and the control thread, where the model-based closed-loop system is run, are synchronized (Figure 6.11). The control thread in the experimental setup runs on a real-time controller with an integrated Field Programmable Gate Array (FPGA); the controller is utilized to drive all three servo motors associated with the three motion axes. Image acquisition is achieved using a Firewire camera connected to a second dedicated hardware-device, where the vision thread runs. Finally, the host thread is run on a stand-alone computer that generates the required target patterns to be displayed on the LCD screen. Communication between the three components (threads) is achieved through a dedicated Ethernet network. Appendix D provides more details on the components associated with each thread, as well as the corresponding software applications programmed using LABVIEW software.

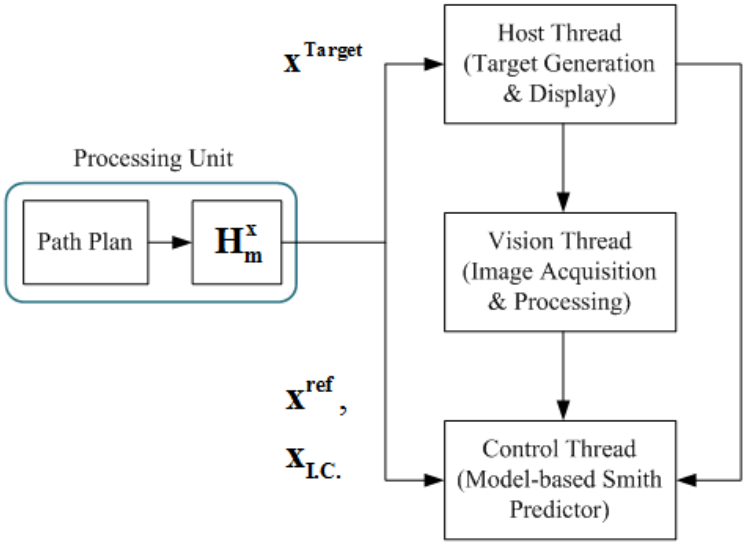


Figure 6.11: Communication scheme of the three independent threads required for MTP implementation. Only one axis is considered for illustration purposes.

In practice, synchronization must be enforced between the host thread and the control thread. Once a new target imaged has been displayed, the host thread must notify the control thread so that the latter updates both the initial conditions of the model and the reference signal (the reference signal must be updated only in the hybrid command issuing protocol when $x^{ref} \neq 0$). The time diagram in Figure 6.12 shows how the three processing threads operate in a synchronized matter. Communication times between threads are considered to be small, with respect to other processing times, and are assumed to be negligible in the analysis that follows. Before t_0 the position of the target image, as displayed on the LCD, must coincide with PP on the image plane. At t_0 the host thread displays a new target image, x^{Target}. Once the image has been updated on the LCD, the vision thread starts the acquisition process at t_0^D, and $\mathbf{x}_{\mathbf{LC.}}$ and x^{ref} are updated at t_0^{update} in the control thread (with $t_0^D = t_0^{update}$). At this point the stage starts moving

according to the initial conditions of the model, and the control effort tries to bring the model states to the reference position. If the resulting velocity from the control effort is too high the tracking error will be reduced to zero rapidly. In Figure 6.12 it takes the control system Δt_0^{fine} seconds to bring the tracking error to zero, according to the model feedback signal. The stage then waits motionless for Δt_0^{wait} seconds until a new target image is displayed at t_1. This wait time is actually a drawback as it introduces discontinuities in the axis velocity (simulation results, presented later, validate this claim).

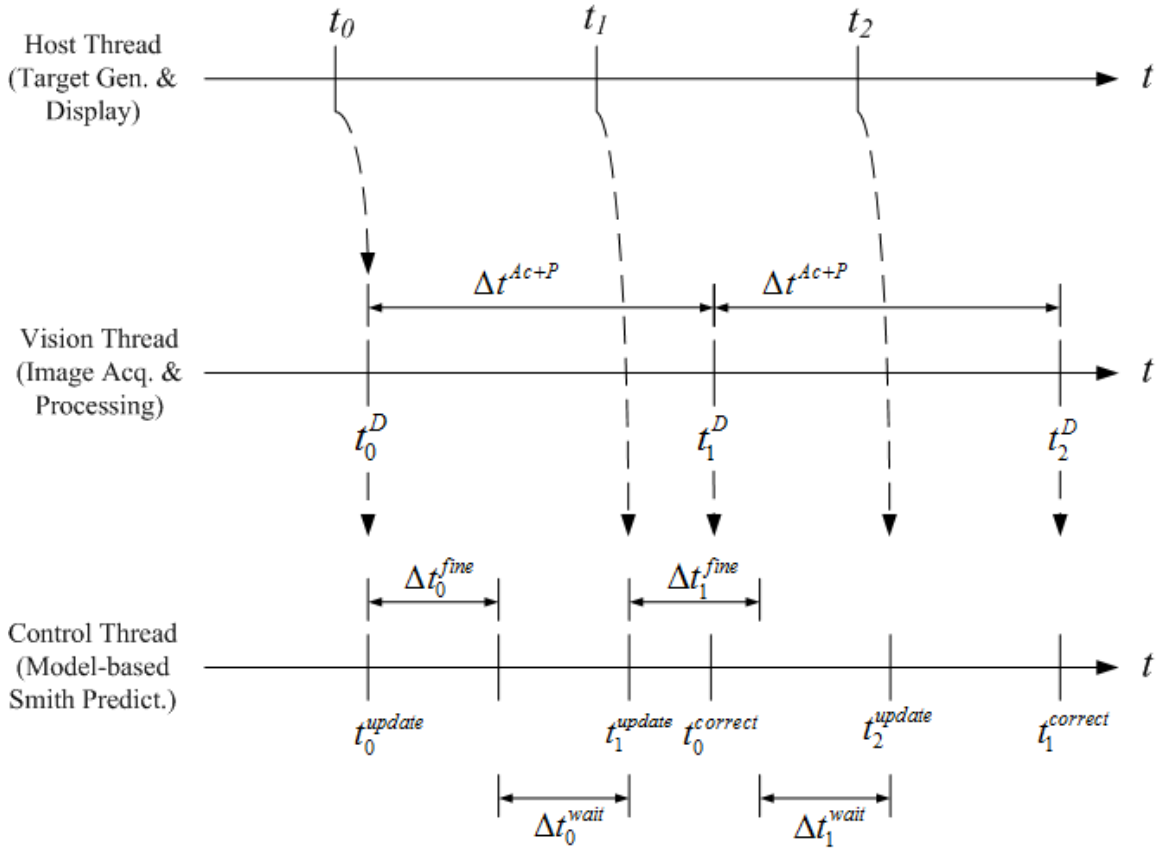


Figure 6.12: Multi-thread time diagram.

Hence, there exists a need to either reduce the output velocity, which would in turn increase Δt_0^{fine} and reduce Δt_0^{wait}, or increase the rate at which the target image is displaced on the LCD, or both. If, for a given desired trajectory, the frequency at which the target image is displaced on the LCD is too high, the control system would operate based mainly on the synthetic feedback signal over the entire path, not taking full advantage of the direct sensing capabilities of the vision sensor. On the other hand, low target update frequencies would cause long wait times making the output motion intermittent and yielding discontinuities in the velocity curve. Ideally, the position of the target on the LCD should be updated at a similar rate as the vision block update frequency. In Figure 6.12 the time required by the vision block to acquire and process a target image is assumed to be constant and has a value of Δt^{Ac+P} seconds; hence, the vision block update frequency is approximately $1/\Delta t^{Ac+P}$ Hz. In such case, when tracking the target over a predefined path plan it is desired to enforce

$$\frac{f_{T.P.}}{f_{V.B.}} \approx 1 \tag{6.12}$$

where $f_{T.P.}$ is the frequency at which the target position is updated on the LCD, and $f_{V.B.} = 1/\Delta t^{Ac+P}$. In practice, $f_{V.B.}$ is not constant and an average can be calculated from experimental data and used for this purpose. In addition to condition (6.12), the velocity of the output motion must also be regulated to increase the value of Δt_0^{fine}, in Figure 6.12, and maintain continuous motion, i.e. avoid discontinuities in the output velocity. If these two conditions are enforced, the output motion and output velocity are expected to be smooth curves over a predefined path plan.

The time diagram in Figure 6.13 shows ideal operation of the three parallel threads with no wait times. It is important to point out that the vision thread captures and processes target images whenever these are available and the correcting factor, $\bar{d}_{(N-1)k} = \bar{x}_{(N-1)k} - \hat{x}_{(N-1)k}$ (see Figure 6.1), occurs asynchronously without affecting the operation of the other two threads. For example, at t_0^{update} in Figure 6.13 the control thread records the current state of the model, $\hat{x}_0$. During the interval $\left[t_0^{update}, t_0^{correct}\right)$ the control loop operates based on the model feedback signal, only. At $t_0^{correct}$ the vision block provides feedback about the actual state of the plant at t_0^{update}, namely $\bar{x}_0$. During the interval $\left[t_0^{correct}, t_1^{correct}\right)$ a correcting factor of $\bar{d}_0 = \bar{x}_0 - \hat{x}_0$ is applied to the model feedback signal. It should be observed that t_1^{update}, the time associated with the second update on the target image, occurs before $t_0^{correct}$. From a stability stand point this aspect does not affect the control loop. In fact, if $t_0^{correct}$ would have occurred before t_1^{update} in the time diagram, it would have only caused the correcting factor to happen at an earlier point in time. Similarly, at $t_0^{correct}$ the control thread stores the current state of the model, $\hat{x}_1$, which is later used during the interval $\left[t_1^{correct}, t_2^{correct}\right)$ to correct for modeling discrepancies.

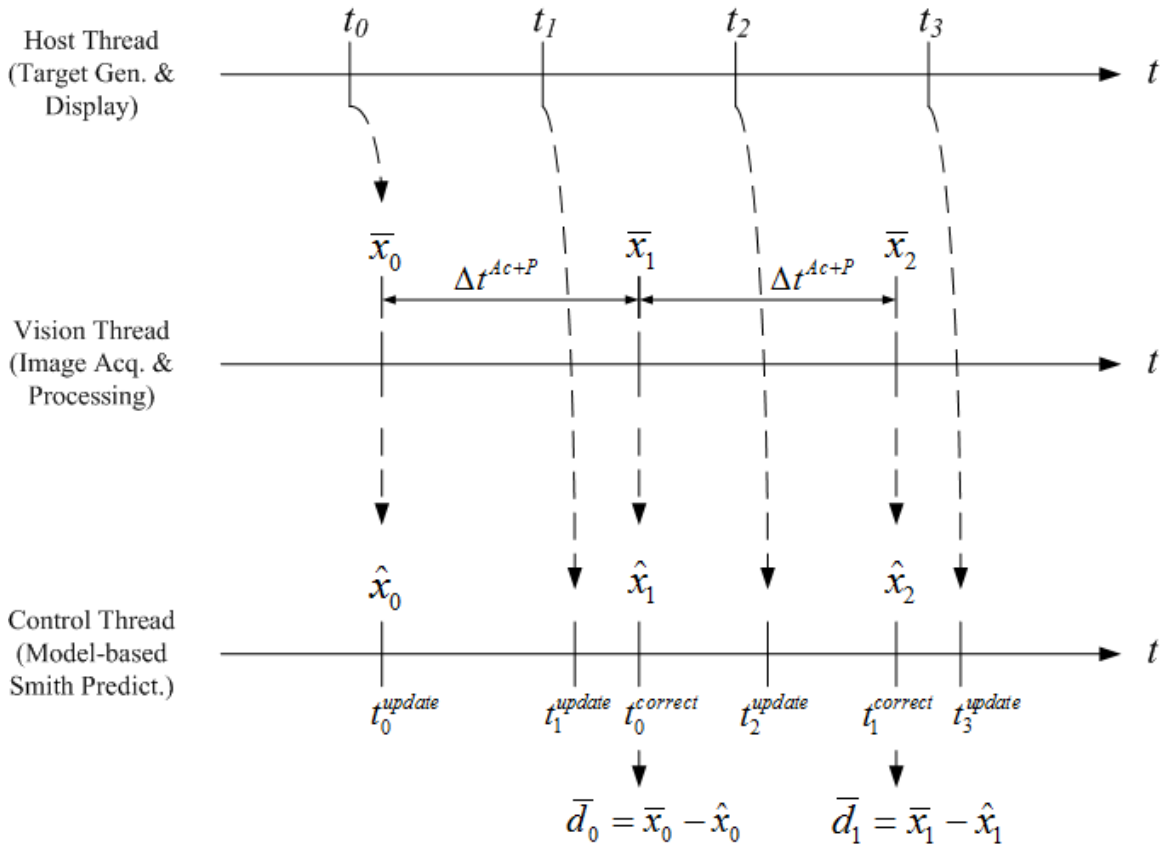


Figure 6.13: Time diagram – asynchronous modeling correction.

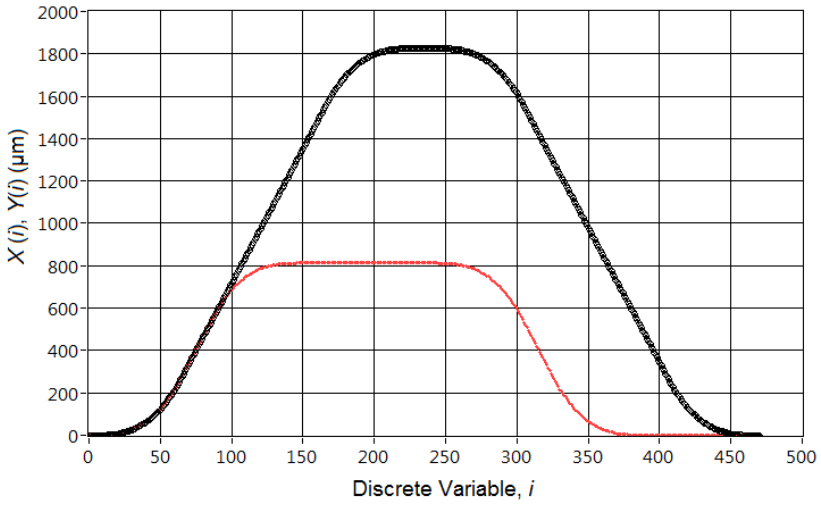
Remark 6.1: Bandwidth of the Model-aided Vision-based Control Loop

In order to increase the control system bandwidth, while satisfying condition (6.12) and avoiding intermittent motion, it is necessary to reduce the image acquisition and image processing times associated with the vision block.

Simulation Results

A basic path plan for fine in-plane displacement is generated. Figure 6.14 shows the desired motion for each axis in (a), and the resulting path in the motion plane in (b).

(a)



(b)

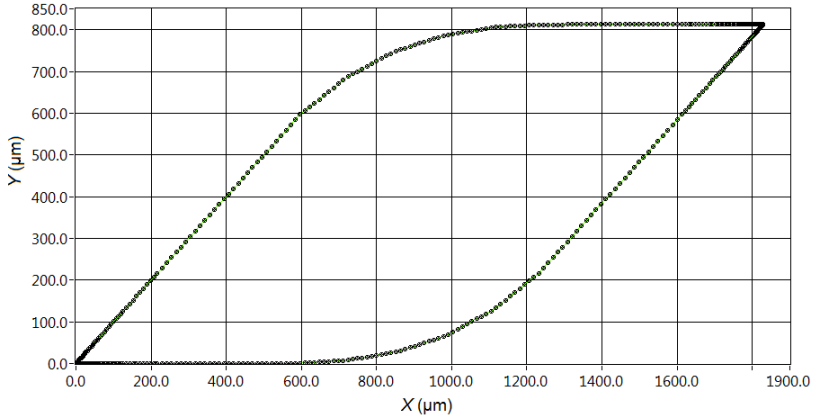
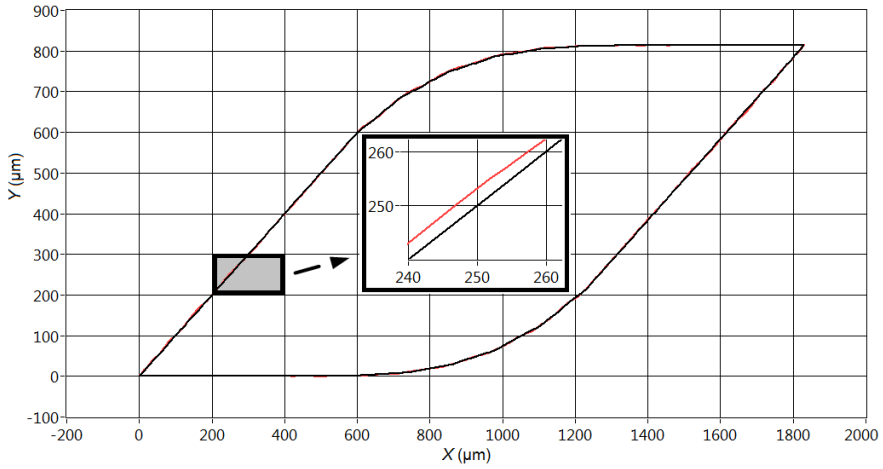


Figure 6.14: Experimental path plan. (a) Arrays *X* (thick trace) and *Y* (thin trace) plotted against the discrete variable *i*, with each array containing 471 points. (b) Array *Y* plotted against array *X*.

Simulation results presented next are characterized by the selection of specific vision block frequencies for each test. Hence, the first step consists in defining $f_{V.B.}$ and then setting $f_{T.P.}$ to a slightly smaller value, such that (6.12) is satisfied. Discrepancies between the model and the real plant are not accounted for in simulation. An initial simulation test is carried out for an ideal single rate system, where the vision block is capable of operating at the controller's frequency. These results are used to compare the performance and tuning parameters of the ideal system with those of the simulated and experimental systems subject to multiple feedback rates. The results of the ideal system are shown in Figure 6.15, when using a PI controller with gains P_x=200 and $T_{i,x}$=0.001min (*X*-axis); P_y=400 and $T_{i,y}$=0.001min (*Y*-axis). The maximum deviation between the desired path and the output path is found to be less than 3 μm, and is highlighted by the rectangle in (a). The output velocity of the *X*-axis is shown in (b).

(a)



(b)

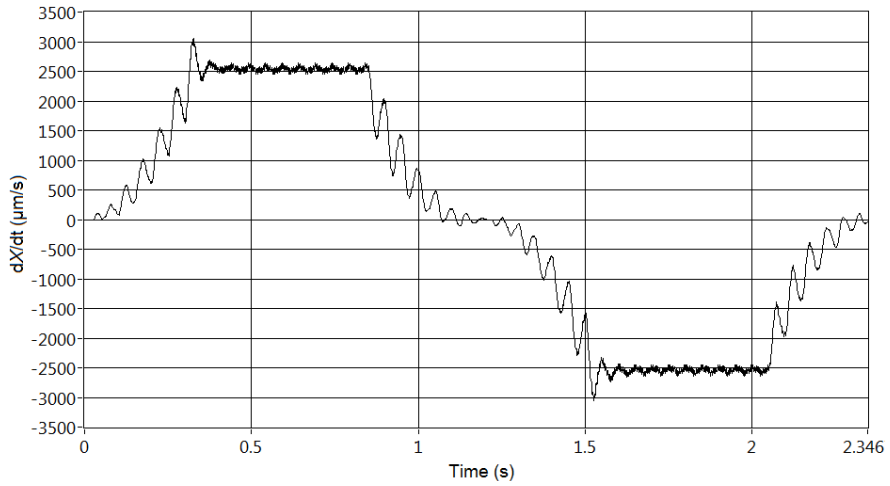


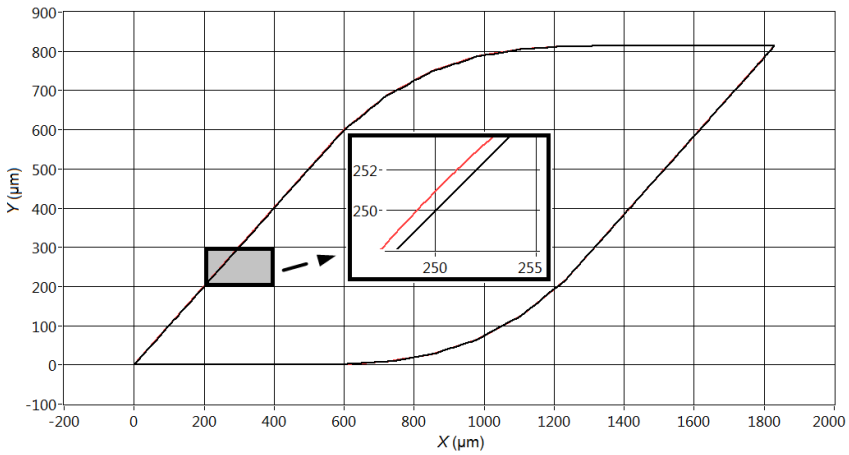
Figure 6.15: Simulation results for an ideal single-rate system operating at 1 kHz with PI Gains: P_x=200 and $T_{i,x}$=0.001min; P_y=400 and $T_{i,y}$=0.001min. (a) Desired in-plane motion and simulation output plotted on same graph. Highlighted rectangular area shows biggest deviation between desired and simulated motion. (b) Output velocity, *X*-axis.

Embedded Digital Coding Protocol – Simulation Results

Simulation results in Figure 6.16 correspond to a system where commands are issued by displaying a target image that carries an embedded digital code. The controller in this simulation runs at 1 kHz while the vision block is only capable of providing a feedback at 2.5 Hz. The graph in (a) shows that it is possible to obtain a similar output motion as the one for the ideal case through a significant decrease in the controller gains.

The effects of long wait times are evident in (b). As expected, the velocity curve is highly discontinuous. Moreover, it takes approximately 188 seconds for the *X*-axis to travel through the entire path; it only takes less than 2.4 seconds for the ideal single-rate position loop to cover the same displacement (Figure 6.15 (b)).

(a)



(b)

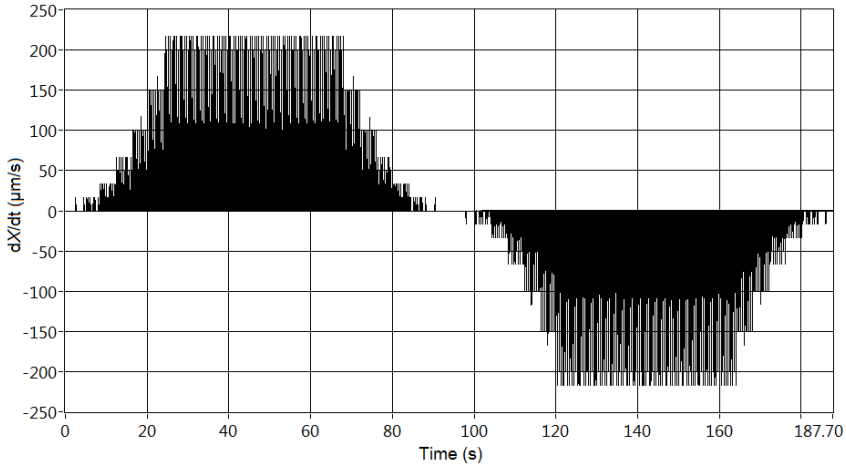
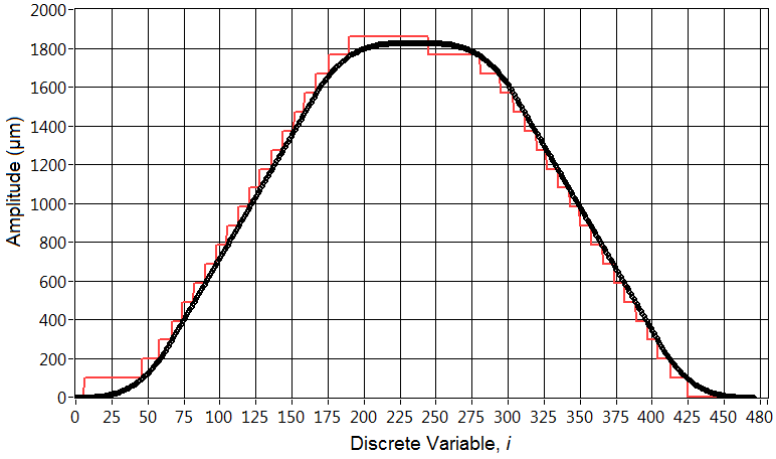


Figure 6.16: Simulation results for a multi-frequency system, where the controller runs at 1 kHz, and the vision block provides feedback signals at 2.5 Hz. PI gains: P_x=20 and $T_{i,x}$=0.1min; P_y=20 and $T_{i,y}$=0.1min. (a) Desired in-plane motion and simulation output plotted on same graph. Highlighted rectangular area shows biggest deviation between desired and simulated motion. (b) Output velocity, *X*-axis.

Hybrid Command Issuing – Simulation Results

The path plan for *X* and *Y* displacement utilized in this test is the same as the one used in the previous method, and is presented in Figure 6.17 (a) and Figure 6.18 (a), respectively (continuous traces). The discontinuous traces in these plots correspond to the 2D position of the target image on the LCD, required to represent $\mathbf{x}^{\mathbf{Target}}$ throughout the path for the hybrid command protocol. Figure 6.17 (b) and Figure 6.18 (b) show the values of the reference signals for the *X*- and *Y*-axes to the control system. As expected, the reference signals are bounded by the size of an LCD pixel: ± 98 μm (1/3 of a pixel) for *X*-axis motion and ± 294 μm (1 full pixel) for *Y*-axis motion.

(a)



(b)

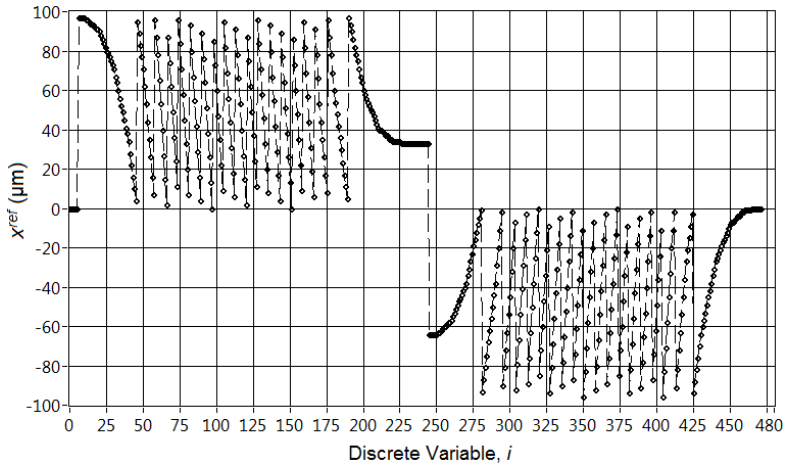
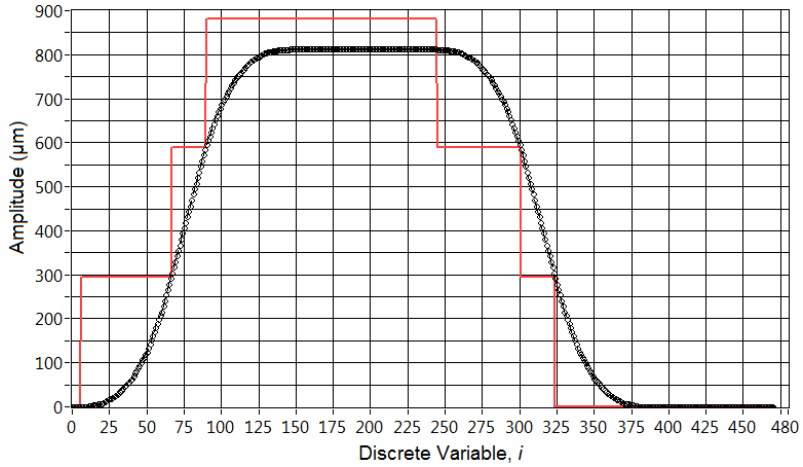


Figure 6.17: (a) *X*-axis path plan (continuous trace) and corresponding target position on LCD. (b) Reference control-signal required for path plan.

(a)



(b)

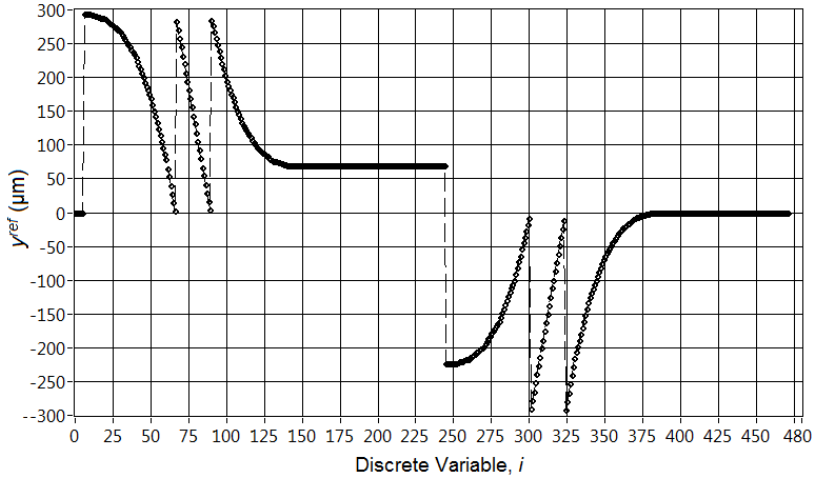


Figure 6.18: (a) *Y*-axis path plan (continuous trace) and corresponding target position on LCD. (b) Reference control-signal required for path plan.

Simulation is conducted using again a controller operating at 1 KHz and a vision-block providing feedback at 2.5 Hz. Figure 6.19 (a) shows a similar result as the one obtained for the embedded digital code, when using the same PI gains. The biggest difference between the two command protocols is evident when comparing the velocity response of the stage *X*-axis; clearly the wait times during which the stage remains stationary are highly reduced with the hybrid method. This in turn yields fewer discontinuities in the velocity pattern as illustrated in Figure 6.19 (b). The magnitude of

the velocity peaks in (b), which is around 1500 µm/s, is also closer to the continuous magnitude of the ideal velocity pattern in Figure 6.15 (b), which is approximately 2500 µm/s. The total time for the control system to complete the path along the *X*-axis is approximately 34.5 seconds, which is again an improvement with respect to the 188 seconds (approximately) required by the embedded digital code method.

(a)

(b)

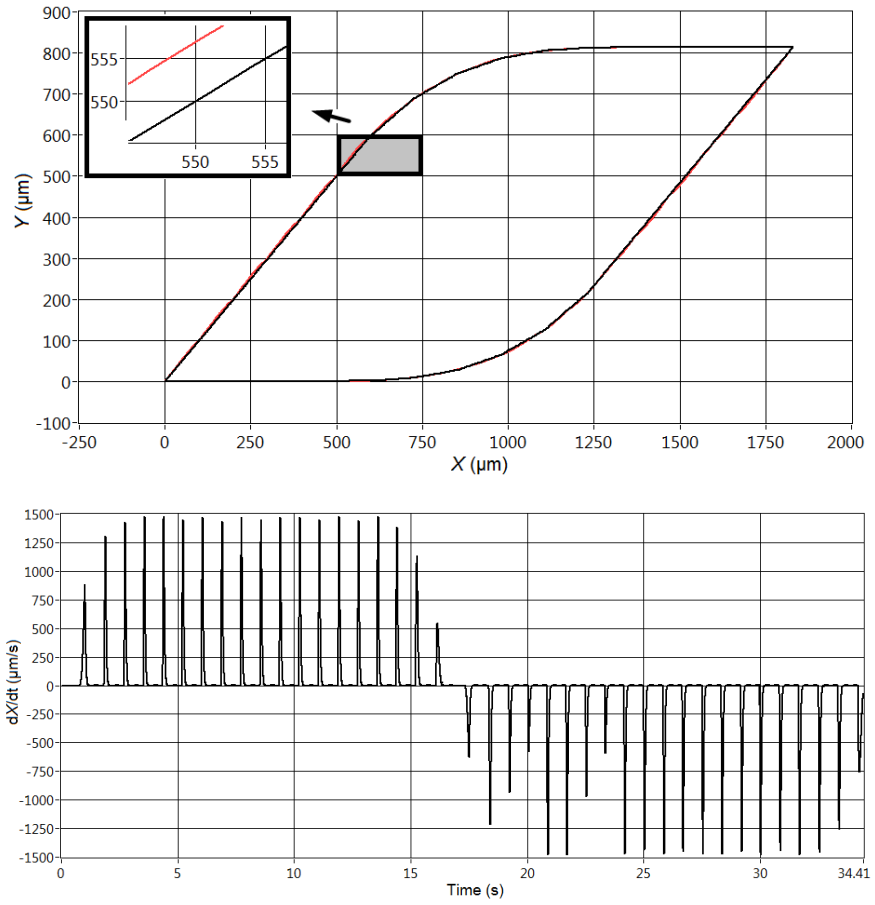


Figure 6.19: Simulation results for a multi-frequency system, where the controller runs at 1 kHz, and the vision block provides feedback signals at 2.5 Hz. PI gains: P_x=20 and $T_{i,x}$=0min; P_y=20 and $T_{i,y}$=0.1min. (a) Desired in-plane motion and simulation output plotted on same graph. Highlighted rectangular area shows biggest deviation between desired and simulated motion. (b) Output velocity, *X*-axis.

Experimental Implementation: Hybrid Command Issuing

Modeling and Operation at Low Speeds

The linear model for the servo motors, obtained through system identification in Chapter Five, is actually a mathematical representation of the table dynamics (the rotary stage, part of the camera mount, is not considered here). The model inertial mass represents the equivalent reflected inertia of the entire structure and includes the inertias of the leadscrew and couplings, the motor rotor, and the masses of the table and drive nut coupled through the mechanical drive ratio defined by the pitch of the leadscrew (Figure 6.20). Deviations between the linear model and the real electromechanical plant are small at medium output motion speeds. At lower speeds, however, the un-modeled nonlinear dynamics associated with friction between moving parts causes the model output to be less accurate with respect to the real plant. As stated in [93], friction has been commonly modeled through static + kinetic + viscous components, in engineering literature:

> "There are many models built to explain the friction phenomenon. In engineering the classical static + kinetic + viscous friction model is commonly used. This model has basically three components that are Coulomb friction, viscous component and static force..." "First, Coulomb friction depends only on the sign of the velocity. On the other hand viscous friction, which is caused by the usage of lubricant, is proportional with the velocity. Static friction represents the force needed to initiate motion from rest and it is usually larger than Coulomb friction. If a system is characterized by combination of viscous friction Coulomb friction and static friction, it will produce a friction characteristic as function of system velocity." (see Figure 6.21).

Additional details on modeling tools and compensation in machines subject to friction can be found in [94].

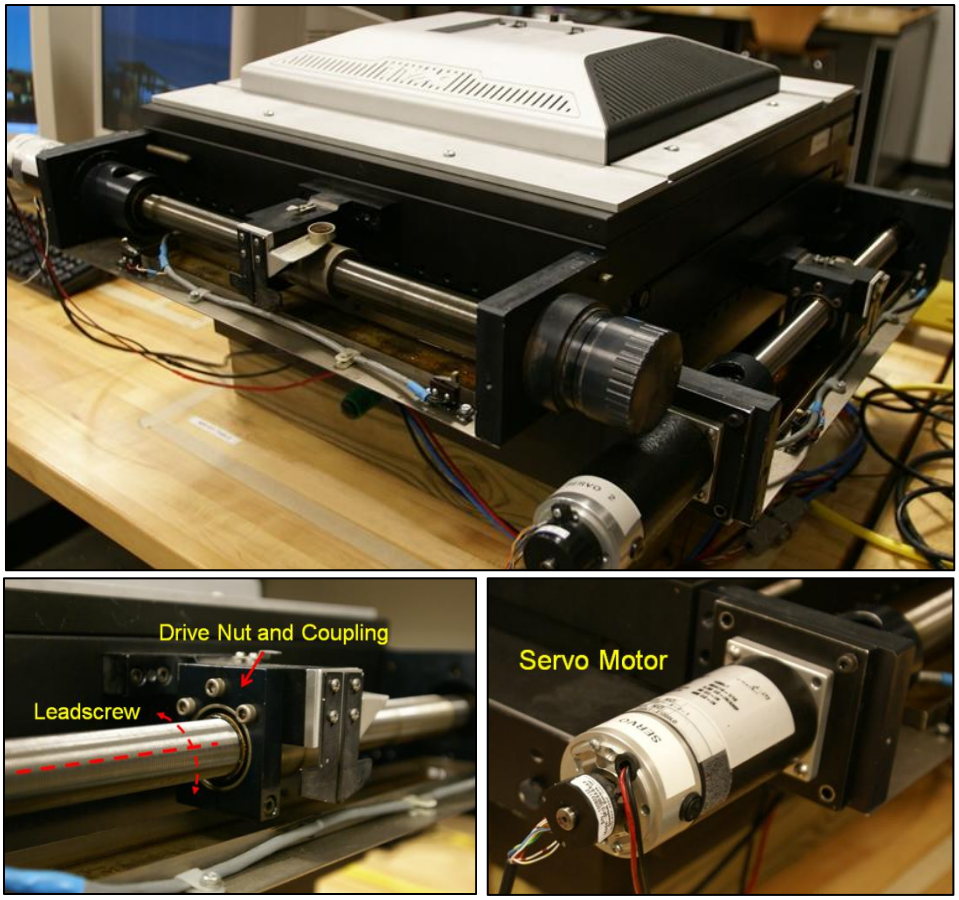


Figure 6.20: Servo motor, leadscrew and couplings of experimental table.

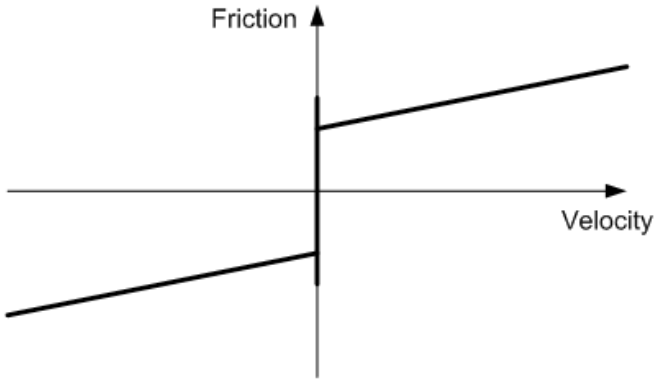


Figure 6.21: Friction vs. velocity curve.

If the input voltage to the servo motor is not enough to produce sufficient torque to overcome static friction and initiate motion and no friction modeling is conducted, the control effort only affects the model states while the actual tool remains stationary. Let the very moment when the vision block provides feedback about the relative position of the stage, with respect to the camera frame, be considered as a reference point in time. To be consistent with previous nomenclature, let this point be denoted as $t_0^{correct}$. Further, let it be assumed that the error between the actual relative position of the stage (feedback signal) and the control reference is large. During the interval $\left[t_0^{correct}, t_1^{correct}\right)$ the model states are driven such that the control error is minimized according to the controller gains. If the control effort is not sufficient to make the stage move but is enough to bring the error between the model output and the control reference to zero before $t_1^{correct}$, then the control effort would stop at some point before $t_1^{correct}$ without causing any changes to the position of the tool. Before $t_1^{correct}$, right after the control error has been reduced to zero, the accumulative counter associated with the controller's integral action is reset. Hence, not even the integral action in charge of reducing the steady state error would provide the additional increase in output voltage to make the stage move. At $t_1^{correct}$ the vision block indicates that the position error remains the same as the one reported at $t_0^{correct}$, and the same cycle begins once again with no effect on the actual position of the stage. A direct consequence of not addressing friction in model-based control applications with intermittent and delay feedback, is that the control system might enter an infinite loop

where the tool remains stationary and only the model states are modified by the controller.

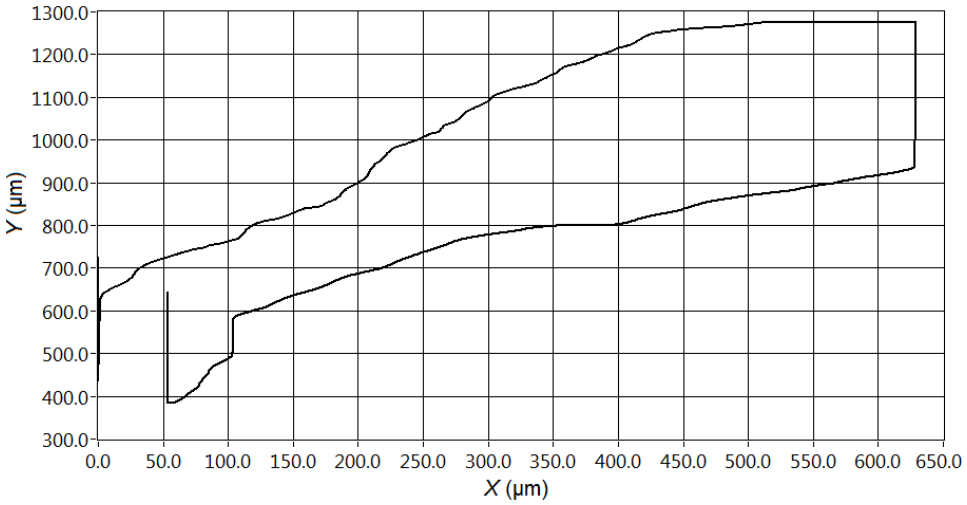


Figure 6.22: Experimental in-plane motion when using a voltage threshold of 1.25 V (same for both axes) to account for friction nonlinearities.

A simple experimental test is run in the XYθ_Z stage to determine a threshold value for the controller's output voltage required to generate sufficient torque in the motor to overcome static friction. The test consists of recording the input voltage to the servos needed to initiate motion from a standstill, and repeat this procedure several times to generate a sample. An average is then calculated from the sample and is set as the voltage threshold required to initiate motion. If the controller's output voltage is below the threshold the model states are copied from the previous iteration emulating motionless conditions of the table. On the other hand, if the output voltage is above the threshold the model obtained in Chapter Five is used to estimate the current position of the plant. Figure 6.22 shows the results obtained when setting a threshold voltage of 1.25 V on the

controllers output signals for both axes. As expected, this approach is insufficient to properly describe the behavior of the plant at low speeds, and additional measures are required.

Nonlinear Modeling through Leadscrew Rotating Threshold

Ideally, friction should be accounted for through modeling. An alternative approach is adopted, where a rotary encoder is utilized to identify the point when static friction is overcome and motion initiates. The nonlinear model operates based on the encoder feedback signal, denoted as *EN* (Figure 6.23). If *EN* does not change between subsequent iterations, the model states are simply copied from the previous iteration. On the other hand if $\Delta EN \neq 0$, the linear model obtained in Chapter Five is used to estimate the current position of the plant in-between vision block updates. Figure 6.23 shows the schematic of the nonlinear model aided by the encoder signal. The model $\hat{G}$ is expressed in the state space form and in the discrete domain as

$$\begin{aligned} \hat{\mathbf{x}}_{\mathbf{i+1}} &= \mathbf{A}\hat{\mathbf{x}}_{\mathbf{i}} + \mathbf{B}V_{in,i} \\ \hat{x}_i &= \mathbf{C}\hat{\mathbf{x}}_{\mathbf{i}} \end{aligned} \tag{6.13}$$

where matrix $\mathbf{A}_{2x2}$ and vectors $\mathbf{B}_{2x1}, \mathbf{C}_{1x2}$ have constant entries (LTI model). The nonlinear operation of the model is summarized through

$$\begin{cases} \hat{x}_i = \hat{x}_{i-1}, & \Delta EN = 0 \\ \hat{x}_i = \mathbf{C}\hat{\mathbf{x}}_{\mathbf{i}}, & \Delta EN \neq 0 \end{cases} \tag{6.14}$$

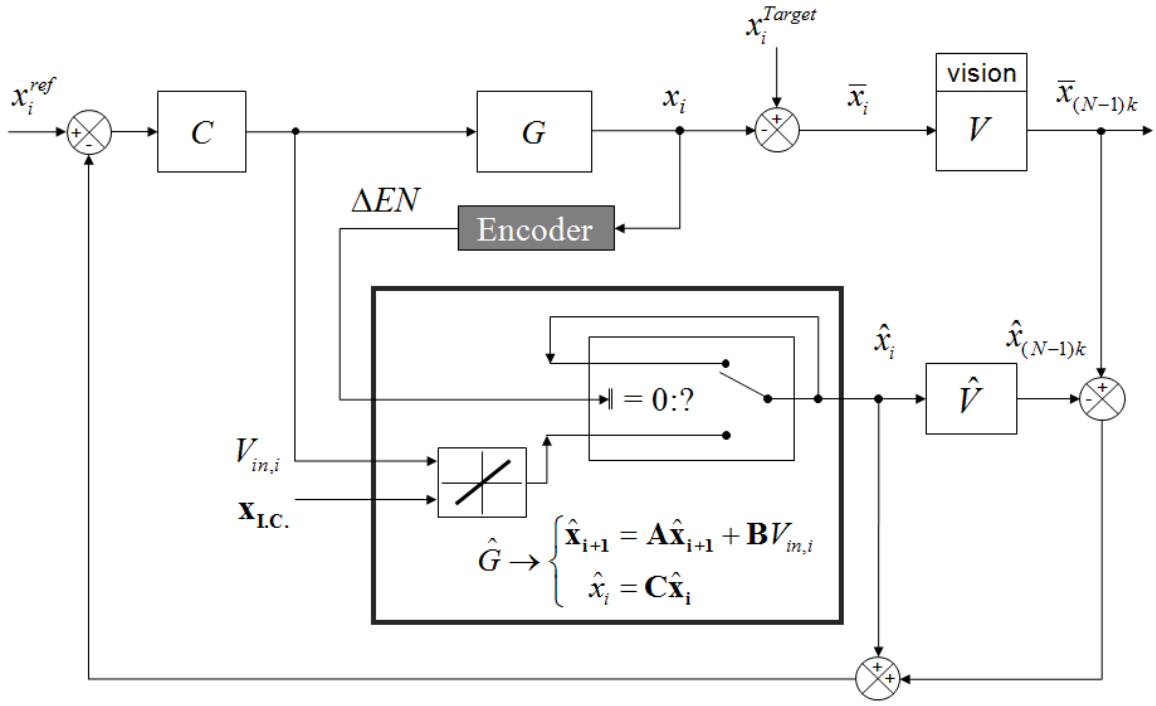


Figure 6.23: Nonlinear modeling aided by auxiliary encoder input.

It is important to point out here that the encoder signal is not used as position feedback to the controller. Therefore, the direct sensing nature of the vision-based control loop is not altered, nor is it enhanced through the encoder in Figure 6.23.

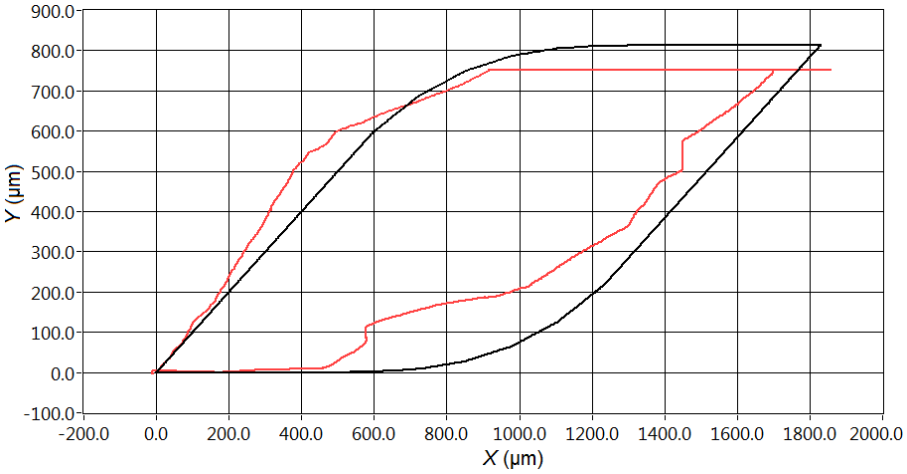
Experimental Results: Non-compensated System

A stock system, like the one in Figure 6.23, is tested using the hybrid command issuing protocol. Initial experimental results of the stock system, when using the same controller gains as the ones used for simulation in Figure 6.19, were found to be extremely inaccurate. Therefore, experimental controller gains are further decreased to P_x = 5, T_{ix} = 0.01 min, T_{dx} = 0.007 min, P_y = 5 and T_{dy} = 0.007 min. The experimental update frequency of the target on the LCD is $f_{T.P.}$ = 0.572785 Hz, whereas the experimental (average) vision block feedback frequency is $f_{V.B.}$ = 0.872723 Hz, yielding

$f_{T.P.}/f_{V.B.} = 0.656$. Though all experimental results presented next correspond to closed loop operation of the stage through MTP and using the hybrid command issuing protocol, the experimental data is actually collected through rotary encoders attached directly to the stage axes. This allows evaluation of performance of the proposed system by means of external sensors. As such, these rotary encoders are only used for data collection and offline analysis.

In order to avoid integral wind-up associated with the controller's integral gain, a reset threshold of 1 rad (approximately 40 um) is set on the accumulative counter. Integral wind-up usually occurs when a large reference is issued and the controller's integral term accumulates a large error while the control system tries to follow the reference. Such large accumulated value usually causes the control variable to overshoot. Despite taking this precaution, the 2D output displacement of the stage shown in Figure 6.24 (a) is still highly deviated from the desired hysteresis pattern and further modifications to the experimental configuration are required. The velocity pattern in Figure 6.24 (b) also demonstrates that the control loop is affected by noise. This aspect is in part attributed to the controller's derivative gain (T_{dx}=0.007). The derivative gain, however, is required to slow down the output motion and avoid wait times, i.e. discontinuities in the velocity pattern. If no penalty is imposed on the output velocity through the derivative gain, the output velocity curve would present more discontinuities and could look similar to the curve in Figure 6.16 (b).

(a)



(b)

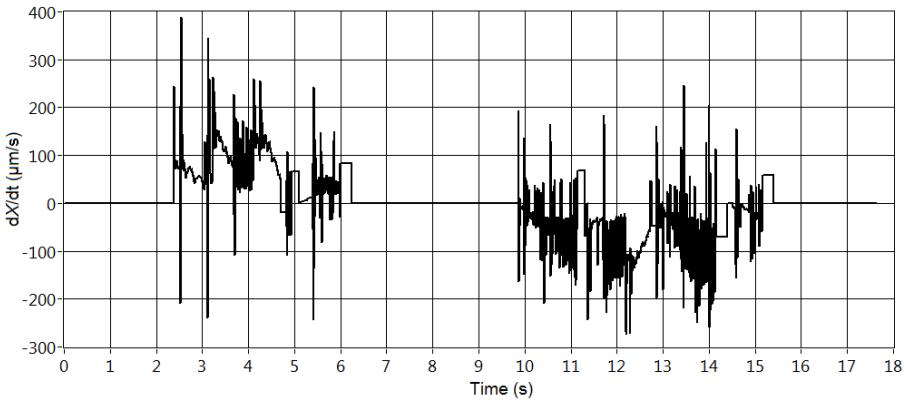


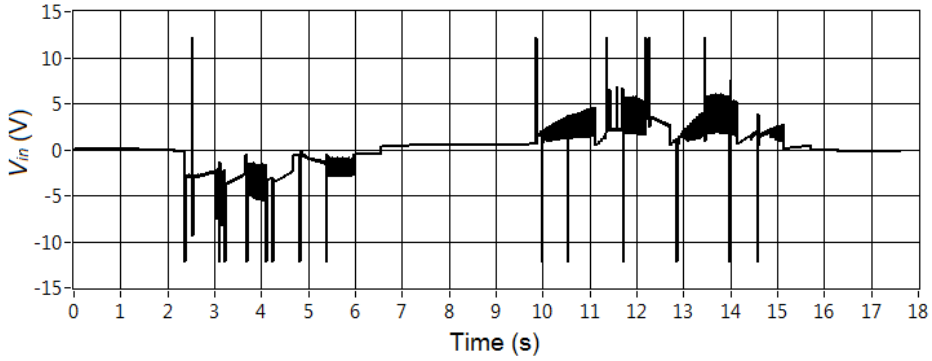
Figure 6.24: Experimental results of the stock system, where the controller runs at 1 kHz, and the vision block provides feedback signals at 0.872723 Hz. Controller gains: P_x=5, T_{ix}=0.01min, T_{dx}=0.007min, P_y=5 and T_{dy}=0.007min. (a) Desired and experimental in-plane motion plotted on same graph. (b) Output velocity, *X*-axis.

Compensated Experimental Control Loop

Complex electromechanical systems are often operated under PID control. Among the drawbacks of simple PID control is the fact that such controllers are affected by noise, which in turn affects stability and more important accuracy in position control systems. This aspect is evident in the graphs in Figure 6.25, which correspond to the controller output voltage for both *X*- and *Y*-axes of the non-compensated system

presented in the previous section. As seen, the controllers output signals change rapidly from negative to positive and reach saturation easily (saturation occurs at ±12.24 V).

(a)



(b)

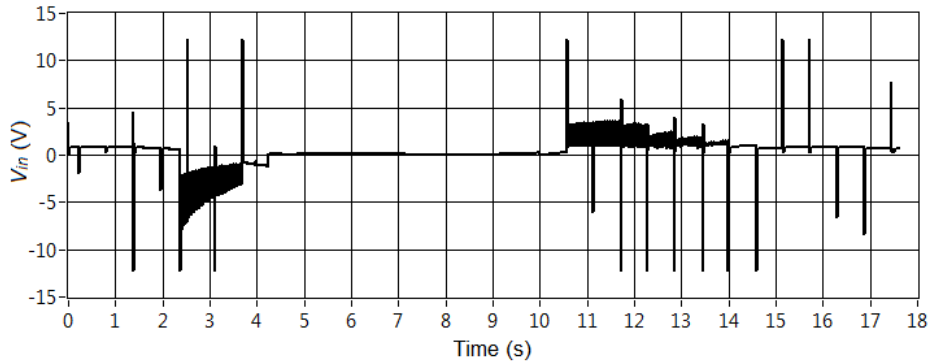


Figure 6.25: Controller output voltage for non-compensated system, with $V_{in,max}$ = 12.24 V (experimental results of non-compensated system are shown in Figure 6.24). (a) Controller output for *X*-axis. (b) Controller output for *Y*-axis.

In order to increase noise rejection and improve system performance a Lead-Lag compensator is introduced into the control scheme, as shown in Figure 6.26. Lead-Lag compensators are commonly used to improve transient response, steady-state response and stability in control systems [95]. Although the response of such compensators is known to be worse than common PID, as it sometimes decreases control system bandwidth, this is less relevant in vision-based control systems given that these systems

are already slow and their performance is not always measured according to a bandwidth criterion [52].

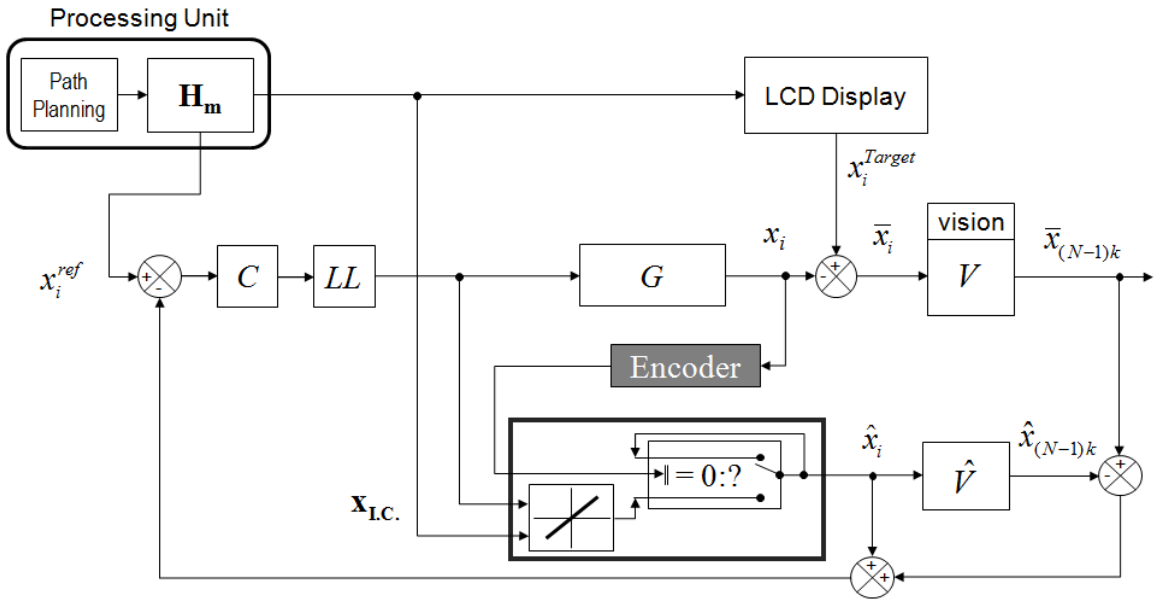
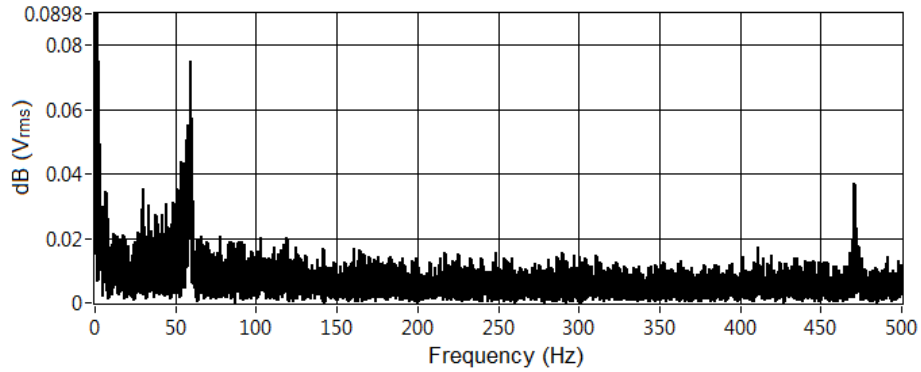


Figure 6.26: Single-axis control system scheme compensated through a Lead-Lag compensator, *LL*.

The design process of a Lead-Lag compensator initiates with the calculation of the lead portion, which is intended to improve the transient response so that certain performance parameters are met. Then, the steady-state error is addressed and improved through the design of the lag portion, which yields an increase in the steady-state gain. It is important to remember at this point that the focus of the vision-aided control system, operated through MTP, is resolution and not necessarily high-speed tracking. Therefore, the design criterion of the Lead-Lag compensator in the current application is focused on rejecting noise and adding stability to the final control scheme. The steady-state error and the transient response are addressed through regular PID control action.

(a)



(b)

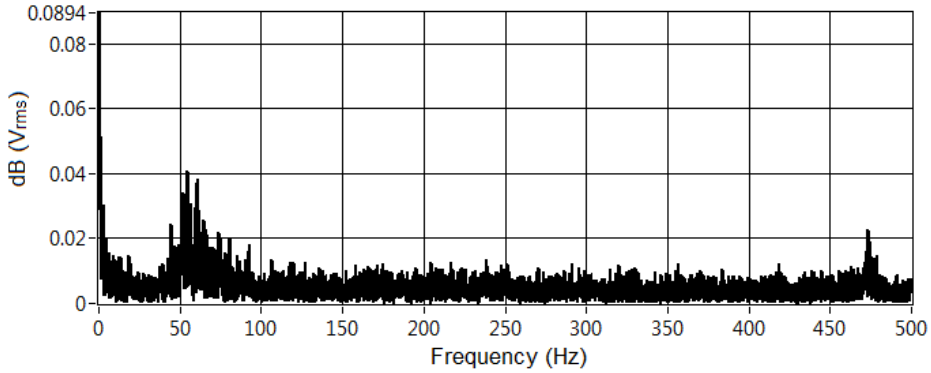


Figure 6.27: FFT of the signals shown in Figure 6.25. (a) FFT of controller output voltage shown in Figure 6.25 (a). (b) FFT of controller output voltage shown in Figure 6.25 (b).

Figure 6.27 shows the Fast Fourier Transforms (FFTs) of the controller output signals shown in Figure 6.25, recorded for the non-compensated system when following the hysteresis path plan. Both graphs present similar frequency content, with main frequencies around 475 Hz, 60 Hz and between 0 and 10 Hz. In an effort to aggressively minimize the effects of noise, the compensator (the same compensator is applied to both axes) is designed such that frequencies below 2.5 Hz are attenuated. The compensator in this case acts as a low-pass filter, and the cutoff frequency is set to approximately 0.5 Hz (3.14 rad/sec). The transfer function of the experimental compensator is (note that $1/3.14 \approx 3.005E^{-1}$)

$$LL(s) = \frac{9.98E^{-4}s + 1}{3.005E^{-1}s + 1} \tag{6.15}$$

The bode plot in Figure 6.28 demonstrates the response of the compensator in the frequency domain. As desired, the magnitude decays -3 dB at around 3 rad/sec. Since $9.98E^{-4} < 3.005E^{-1}$ and as seen in the phase plot, the compensator adds phase lag to the system. In the actual control thread, running in dedicated LABVIEW hardware, the equivalent Lead-Lag compensator gains are: gain = 1, lag time = 0.005 min, lead time = 0 min.

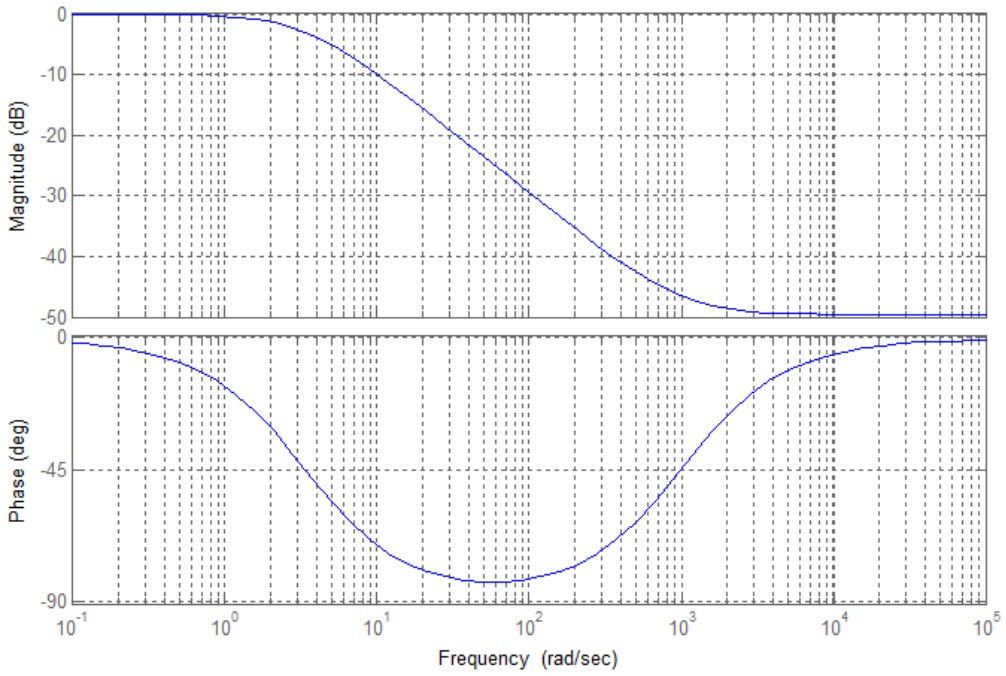


Figure 6.28: Lead-Lag compensator bode plot.

The *X*, *Y* vs. time graphs in Figure 6.29 show a clear improvement in the smoothness of the output motion through a compensation of the controller output signal. Let it be observed, also in the *X*, *Y* vs. time graphs, that it takes the compensated system

approximately double the time (approximately 36 seconds) to complete the path plan, compared to the non-compensated servo loop (approximately 18 seconds). This is due to an increase in the derivative gain (for both axes), from $T_d = 0.007$ min in the stock system to $T_d = 0.05$ min in the compensated one. Simulation results revealed a need to penalize output velocity, in order to obtain smooth velocity curves. However, velocity was difficult to control as an increase in the derivative action (i.e. increasing the penalty on output velocity) also meant an increased in noise levels. The Lead-Lag compensator allows higher control over output velocity through higher derivative gains, while reducing the effects of noise on the output displacement.

The experimental update frequency of the target on the LCD, in this case, is the same as the one for the stock system test; this is $f_{T.P.} = 0.572785$ Hz. The average vision block feedback frequency is found to have a value of $f_{V.B.} = 0.869087$ Hz; hence $f_{T.P.} / f_{V.B.} = 0.66$.

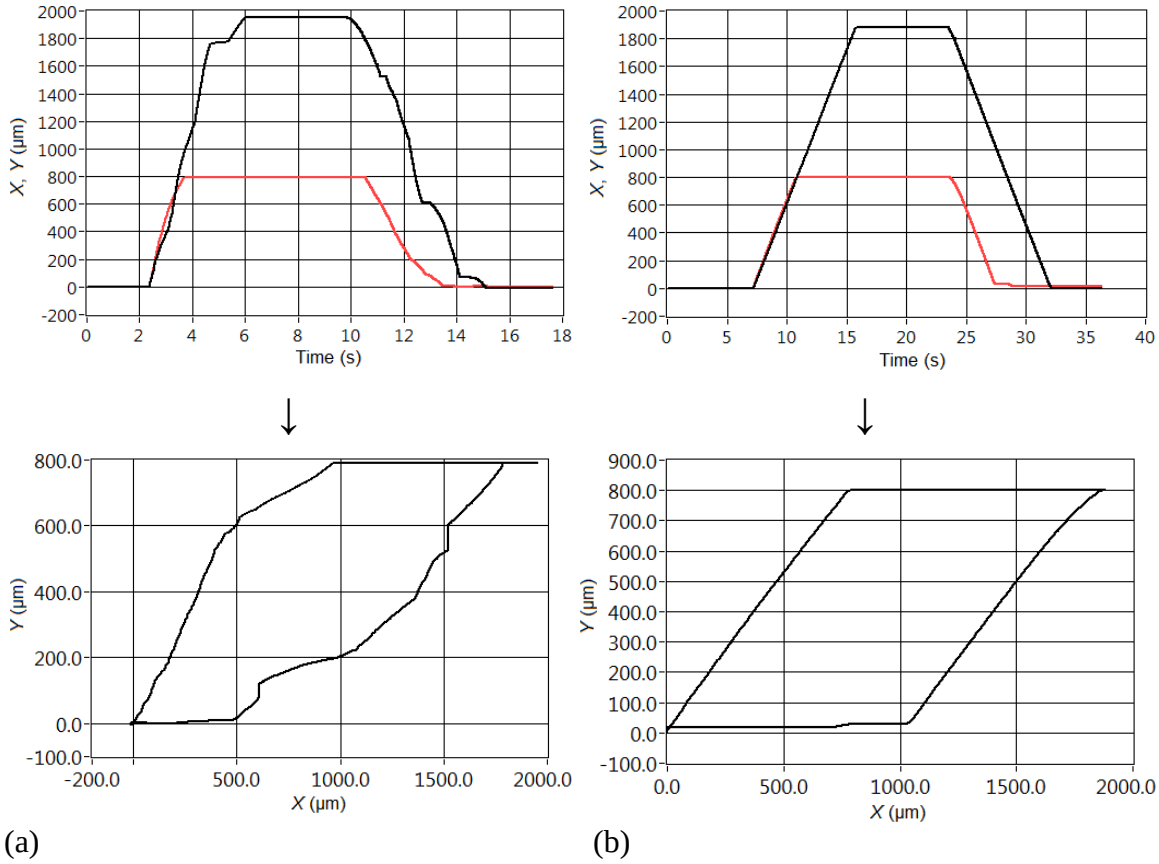


Figure 6.29: Comparison of output motion for the non-compensated and compensated systems, when operating the stage through MTP and using the hybrid command issuing protocol. *X, Y*- axes vs. time plot and in-plane displacement plot: (a) non-compensated system (see controller gains in Figure 6.24), (b) servo system compensated through Lead-Lag compensator. PID gains of compensated system (same for both axes): P = 5, T_i = 0.01min, T_d = 0.05 min. Lead-Lag compensator gains (same for both axes): gain = 1, lag time = 0.005 min, lead time = 0 min).

The result in Figure 6.30 (a) shows the actual displacement of the stage on the *XY*-plane (same as the one shown in Figure 6.29 (b), bottom) plotted on top of the desired hysteresis path. The highlighted rectangle in the top left corner in Figure 6.30 (a) indicates the presence of accuracy errors on the output motion on the order of approximately 25 µm, in areas where both axes move along a straight line. The largest error in the output planar motion is highlighted by the second rectangle in the bottom right corner, showing a deviation of approximately 40 µm in the *Y*-axis and a little less

than 100 μm in the *X*-axis. These errors are associated with the correcting factor (see Figure 6.1), $\bar{d}$, which is assumed to be constant in-between updates from the camera feedback signal.

The un-modeled error between the actual servo (plant) and its corresponding mathematical model was defined in Chapter Five as d_i. This variable was replaced at the beginning of the present chapter by $\bar{d}$, as it is the relative position of the stage with respect to the camera frame, $\bar{x}_i$, and not the absolute position of the stage, x_i, the one being modeled and tracked over time. Ideally, the control system should see a feedback signal $s_i = \bar{x}_i$, which would provide accurate information about the relative position of the stage for all $i \in \mathbb{N}_0$. For the current experimental case of the *single CAM- single image processing*-thread, the feedback signal has the form $\hat{s}_i = \hat{x}_i + \bar{d}_{(N-1)k}$, where $\bar{d}_{(N-1)k} = \bar{x}_{(N-1)k} - \hat{x}_{(N-1)k}$ for $N \in \mathbb{N}_0$ and $i \in \left[Nk, (N+1)k \right)$. The vision block was assumed to operate at constant frequency in Chapter Five, providing information about the position of the stage only every *k* samples (*k* = constant). Then, the error between the ideal and actual feedback signals can be defined as

$$\bar{e}_i = s_i - \hat{s}_i \tag{6.16}$$

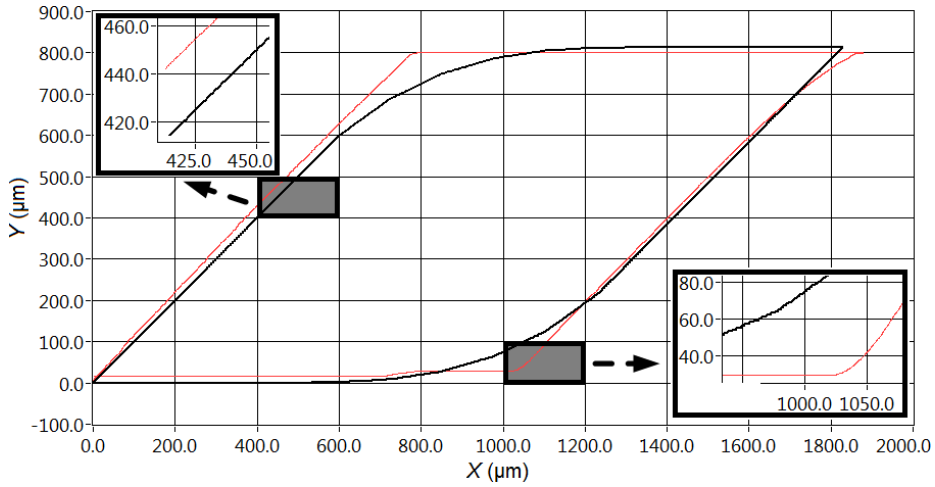
$$\bar{e}_i = \bar{x}_i - \left(\hat{x}_i + \bar{d}_{(N-1)k} \right) \tag{6.17}$$

$$\bar{e}_i = \bar{d}_i - \bar{d}_{(N-1)k} \tag{6.18}$$

Discrepancies between the ideal and actual feedback signals in model-aided vision-based control, given by (6.18), are responsible for inaccuracies in the output

displacement of the stage shown in Figure 6.30 (a). It is important to stress that even if the vision block was capable of operating at the controller's frequency (k = 1), i.e. without intermittencies, the error between the ideal and actual feedback signals would still be different from zero due to the processing delay. In such case, $\bar{e}_i = \bar{d}_i - \bar{d}_{(N-1)}$ for $i \in [N, N+1)$ and $N \in \mathbb{N}_0$. Error $\bar{e}_i$ would only be zero if no intermittencies and no delays were present in the vision block feedback signal; in such case, however, no modeling would be required. This is another fundamental limitation associated with delays and intermittencies in *single CAM- single processing thread*-visual servoing. Finally, the graph in Figure 6.30 (b) shows the positive effects of the Lead-Lag compensator allowing an increase in the controller's derivative gain, and resulting in an approximately continuous output velocity pattern.

(a)



(b)

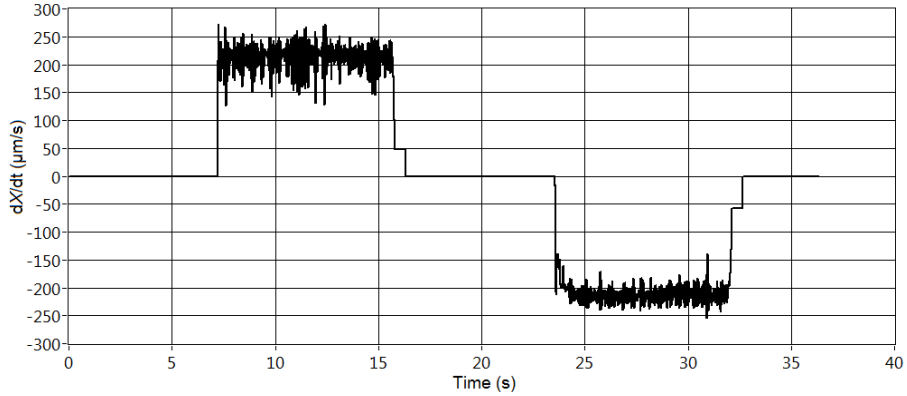


Figure 6.30: Experimental results of the compensated system, where the controller runs at 1 kHz, and the vision block provides feedback signals at 0.869087 Hz. PID gains (same for both axes): P = 5, T_i = 0.01min, T_d = 0.05 min. Lead-Lag compensator gains (same for both axes): gain = 1, lag time = 0.005 min, lead time = 0 min. (a) Desired and experimental in-plane motion plotted on same graph. (b) Output velocity, *X*-axis.

Concluding Remarks

Two command generation protocols have been presented for implementation of MTP in real-time control applications. Command issuing resolutions are no longer dependent upon the size of the smallest element of the grid/display being imaged, and therefore can be determined in accordance with the sensor's resolution. Adequate command issuing through the proposed methods requires synchronization between the host- and control-threads. Two conditions for achieving smooth output motion in the vision-based control system are identified: 1. The ratio between the frequency at which the imaged is displaced on the LCD and the frequency at which the vision block is capable of providing feedback information, should be approximately 1 ($f_{T.P.}/f_{V.B.} \approx 1$); 2. Velocity of the output motion must be regulated to avoid intermittent output motion. If the frequency of the vision block increases in condition one, this implies that the frequency at which the target is displaced on the LCD, and therefore the command issuing frequency, increases as well. As such, less velocity regulation (condition 2) must be enforced. Hence, conditions 1 and 2 are correlated through $f_{V.B.}$, which represents a performance dependence of vision-aided control systems on computational efficiency, including both hardware as well as software capabilities. As the frequency of the vision block approaches the controller's frequency, motion commands can be issued at faster rates and less velocity regulation is required. This highlights a bandwidth limitation of common vision-aided control applications; the use of multi-threaded dedicated hardware

equipment, together with high speed cameras running on high memory-bandwidth networks can help increase visual feedback rates, and improve performance levels.

Experimental results also confirm that appropriate operation of the model-aided visual servo loop relies highly on accurate modeling of the system dynamics. In particular, the case study of the *single CAM- single image processing thread*-configuration, when operating at low speeds the plant is subject to high friction coefficients. If no adequate friction modeling or friction compensation is performed when operating at low speeds, the control system can enter an infinite processing loop where the tool remains stationary and only the model states are modified by the controller.

Finally, the overall best achievable resolution for in-plane closed-loop positioning of the stage is found to be on the order of 100 μm (approximately). Analysis and experimental results expose a fundamental drawback in model-aided vision-based servo mechanisms when applied to high-resolution position control under the proposed conditions. Namely, the control feedback signal is not generated based upon current sensed data, but is approximated through modeling and old sensed data. Though this approximation would certainly be considered acceptable for macro-positioning applications, some micro- and, even more so, nano-positioning systems might require additional elements to aid in achieving the desired resolution levels. For such micro- and nano-scale systems a combination of common high-resolution sensors and vision-based control is suggested. The appeal and need for high-resolution imaging-based positioning systems, however, still remains due to the direct-sensing nature of its transducers, which

eliminates the need for common error compensation and calibration techniques in NC machines.

CHAPTER SEVEN

CONCLUSIONS

Final Remarks

The concept of a new three dimensional position feedback system using a digital camera aimed at an actively modulated image on an LCD screen with application to position control in NC machines was investigated and developed. Implementation was carried out on an XYθ_Z stage. The research work required an in-depth study and modeling of the camera dynamics and the stage dynamics, for optimal integration of these two components in a robust closed-loop control system. A mathematical representation of the image acquisition process was established based on a perspective projection model when using a thin lens. Given some a-priori information, the position of a known target on the display was determined with high accuracy through two new image processing algorithms. The first method was based on the detection of a cross-hairs represented by the intersection of two high-intensity lines over low-intensity background. The effects of noise were minimized by using a Newton-Raphson optimization technique to find the best two perpendicular lines that fitted the acquired data. Finally the intersection and orientation of the lines were calculated and correlated to the position of the tool for displacement and orientation measurement in the *XY*-plane. Experimental results showed that, although the pixel size of the display was close to 300x300μm, the procedure was able to reliably detect stage motions as small as 2.5 μm. A checkered pattern was introduced in the second method to define the object of interest. Two lines in the checkered pattern, forming again a cross-hairs, were in this case represented by the

edges in the digital image. Fine-edge location was carried out through the calculation of best-fit 2D Gaussian curves around the peaks on the gradient image. The means of the best-fit Gaussian functions in the gradient image were taken as the fine location of the edges in the original image. Once all points along the edges were determined with sub-pixel resolution, these points were split into two data sets associated to two perpendicular lines, and constrained curve fitting was once again applied to these sets to define the target location in the CCD plane.

The physical characteristics of the LCD pixels indicated that different color topologies could be used for finer representation of the target on the display. At the same time, the actual LCD pixel presented one of the barriers that needed to be overcome for implementation. Namely, commanding displacements on the order of 2.5 μm through a target image that could only be displaced on the LCD frame by increments of at least 100 times that value. Two new command generation protocols were developed to make full use of the direct sensing capabilities of the vision sensor, and still achieve command-resolutions of less than 1/100th of the pixel size on the display. Functionality of the command protocols required synchronization between the host thread, in charge of generating and displaying the target on the LCD, and the control thread, where the model-based PID control loop was run. Synchronization was achieved by appropriately updating the initial conditions of the MIMO plant-model as well as the control system set point, based on the known displacement of the target image on the display.

The integrated vision-based control loop presented time delays and intermittencies in the feedback signal, which clearly affected stability and performance of

the overall system. A Smith predictor topology was studied in order to address these issues. Both analytical and practical details about the functionality of the Smith predictor scheme in visual servo-mechanisms were explained. By isolating and analyzing the discontinuities and delays, it was possible to clarify ambiguities about real implementation of the multi-rate system. Finally, the requirements for stability during intermittent updates of the vision-feedback signal were presented.

The direct sensing nature of the control system introduced here implies that error compensation is no longer conducted outside the control loop. Geometric errors of the machine are detected by the sensor and are therefore compensated through control action. The need for complex kinematic models and high costs associated with error mapping and calibration procedures are highly reduced through the new configuration. Successful results in this work should serve in the design of true closed-loop motion control systems for simultaneous axis positioning of manufacturing equipment.

Though sensor resolutions smaller than 2.5 μm were obtained through adequate signal processing, the actual position control resolution for a predefined trajectory was found to be on the order of 100 μm (approximately). The gap between these two resolution values is attributed to inaccuracies in the real-time estimation of the feedback signal, based upon modeling and old sensed data; such estimation is needed to compensate for time delays and intermittencies in the vision block output signal. In this sense, performance levels in vision based control rely highly on a combination of efficient signal processing algorithms, as well as the development of more powerful real-time hardware.

Broader impacts of this research include: intelligent pattern generation for ultrafine position control; fundamental advancements in accuracy and direct control of multi-degree-of-freedom manufacturing equipment through visual detection of dynamic elements on a display; and the architecture of a new class of model-aided vision-based positioning systems with disparate update rates.

Original Contributions

The original contributions resulting at the end of this research work are [96-100]:

- A new position control configuration and command generation platform with application to machine tools.
- A new three dimensional absolute positioning sensor using off-the-shelf LCD monitor and CCD digital camera. The LCD was studied as a pixelated grid imaged by the camera, with the latter providing both in-plane position and orientation information of its relative location with respect to the former.
- A new approach for managing geometric errors, present in common machine tools due to inaccuracies in the construction of its individual parts. This was accomplished by direct sensing the position of the tool. Calibration procedures were thereby minimized and the dependence on kinematic models in a three dimensional working environment was eliminated.
- Two high resolution image processing methods for in-plane absolute positioning were presented. These methods, based on edge detection and continuous intensity pattern identification, utilize optimization techniques to increase signal-to-noise

ratio (SNR) and were used to perform high resolution measurements of an actively control target.

- Two new command generation protocols were introduced. Based on a desired path plan, the required reference signal to the control system and the 2D coordinates of the target image on the display, were generated following a well-defined set of constraints to guarantee the desired final motion.

Future Research

As stated in Chapter Four one of the biggest limitations of the signal processing algorithms in the present work are due to distortion and other physical errors associated with the optical elements of the vision system. An appropriate camera calibration procedure is suggested to be applied to the image acquisition process to decrease uncertainty in the measurements. Calibration of the actual physical dimensions of the LCD pixel and the uniformity of their pixel spacing is also required to establish the actual positioning accuracy of the system.

As it is known, a common LCD screen has refresh rates in the range of 50-100 Hz, while most digital cameras can only capture images in the range of 20-100 FPS. These rates are several orders of magnitude smaller than the processing rates of a common PID controller (normally in the order of KHz). The use of multi-threaded dedicated hardware equipment, together with high speed cameras running on high memory-bandwidth networks can certainly mitigate some of the problems encountered in

visual-aided applications and can help increase accuracy levels when these are used for position control.

From a theoretical point of view, the derivation of an analytical proof for stability of the multi-rate control loop considering the slower time-loop Nk (for N=1,2,3...) in Figure 5.10, where the transition of i should be analyzed from $i \in \lfloor Nk, (N+1)k \rangle$ to $i \in \lfloor (N+1)k, (N+2)k \rangle$, is recommended.

APPENDICES

Appendix A

Focal Length of a Camera with a Thin Lens

The formula commonly used to calculate the experimental focal length of a camera is derived next. The analysis is based on a paraxial approximation when dealing with a thin lens.

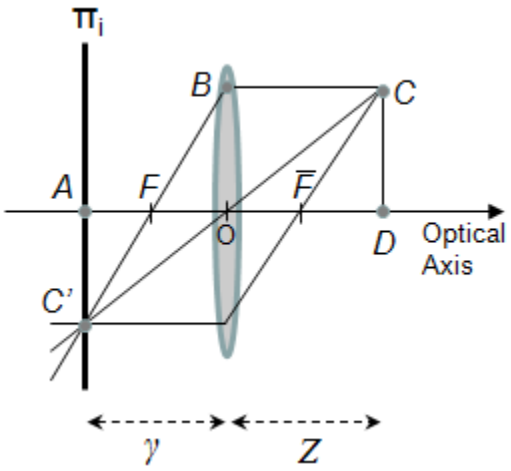


Figure A.1: The thin lens.

For the thin lens in Figure A.1, the following relationships hold

$$\frac{AC'}{DC} = \frac{OA}{OD} \tag{A.1}$$

$$OA = FA + OF \tag{A.2}$$

$$\frac{AC'}{DC} = \frac{FA}{OF} \tag{A.3}$$

From (A2) and (A3) it is clear that

$$OA = OF\left[\frac{AC'}{DC} + 1\right] \tag{A.4}$$

Inserting (A.4) into (A.1) and isolating *OF* yields

$$OF = \frac{AC'OD}{AC' + DC} \tag{A.5}$$

Now if *AC' = CCD Width*, *OD = Working Distance* (or *Z* in the pinhole camera model), *DC = Object Width* and *OF = f* then the previous equation can be written as

$$f = \frac{Working\ Distance * CCD\ Width}{Object\ Width + CCD\ Width} \tag{A.6}$$

where all units are in mm. Equation (A.6) is commonly used to determine the experimental focal length of a vision system.

Appendix B

Image Processing - MTP Method 1: C/C++ Code

The image processing code was originally programmed in C++ using a specific image processing library. The code was later modified to regular C code, such that no dedicated library was required, and in order for it to be integrated within the vision thread in a LABVIEW programming environment. The integration was accomplished through the *Formula Node* VI available in the programming module of the LABVIEW 8.6 functions palette. The *Formula Node* VI allows the insertion of text-based C code into LABVIEW. Figure B.1 shows the *Formula Node* VI, where the C code is run within the vision application (thread). The complete integrated code can be found in appendix D. Here, only the text-based code is presented, preceded by a brief description of the input and output variables (see input and output variables to the left and right margins of the *Formula Node* VI in Figure B.1) of the *Formula Node* VI.

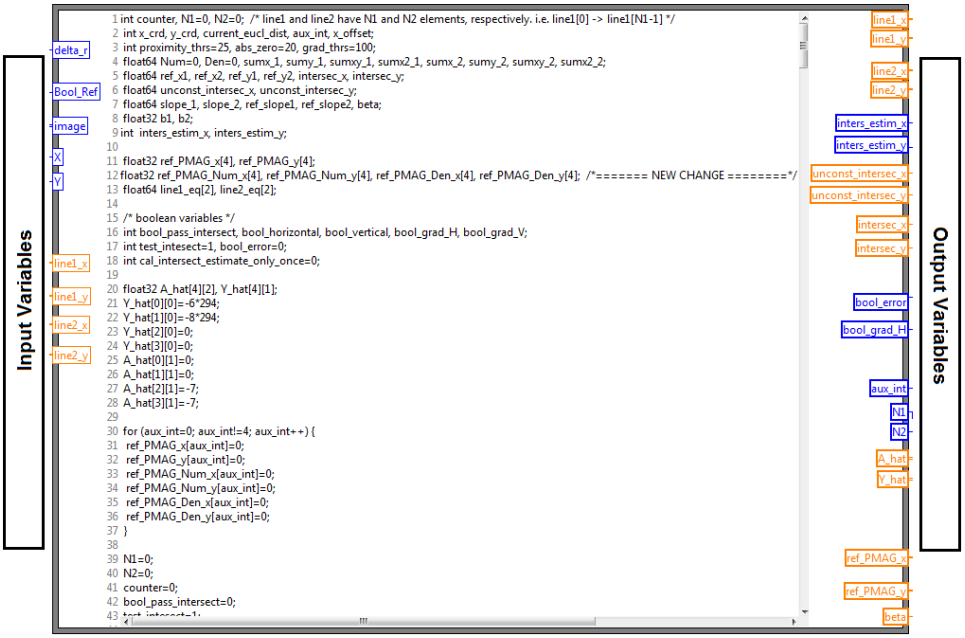


Figure B.1: *Formula Node* VI.

Input Variables

Variables `line1_x`, `line1_y`, `line2_x` and `line2_y` in Table B.1 can be defined inside the code and not as inputs, as they do not really provide input data to the code. The reason for them to be classified as inputs is related to a convenience factor that simplifies the interaction between the text code (C code) and the graphical code in LABVIEW programming environment. Variables `line1_x`, `line1_y`, `line2_x` and `line2_y` are used to store the coordinates of the experimental data points utilized in the constrained curve fitting procedure to determine the location of the target on the digital image with high resolution.

Table B.1: Input variables to the *Formula Node* VI.

Name	Type	Description
`delta_r`	integer	Variable used to specify the radius of the virtual circle that encloses the intersection of the cross-hairs target; pixels inside this circle are only considered for the calculation of the optical magnification through the reference elements.
`Bool_ref`	binary/ boolean	Variable used as an on/off switch to determine whether or not the optical magnification must be calculated. The calculation of the optical magnification requires a single iteration of the code. Once known, the procedure for calculating the optical magnification can be turned off to increase processing speeds.
`X`	integer	Specifies the horizontal size of the image to be processed.
`Y`	integer	Specifies the vertical size of the image to be processed.
`image`	[Y][X] integer matrix	Digital image.
`line1_x`	[arbitrary size, e.g.1500] floating point array	Empty array that, after the entire code is run, contains the floating point *x*-coordinates of all the points in the digital image that are associated with the first line of the cross-hairs target. Floating point precision is achieved through the centroid calculation explained in Method 1, step 1, in Chapter Four.
`line1_y`	[arbitrary size, e.g.1500] floating point array	Empty array that, after the entire code is run, contains the floating point *y*-coordinates of all the points in the digital image that are associated with the first line of the cross-hairs target.
`line2_x`	[arbitrary size, e.g.1500] floating point array	Same as `line1_x` but for the second line.
`line2_y`	[arbitrary size, e.g.1500] floating point array	Same as `line1_y` but for the second line.

Output Variables

Once the C code inside the *Formula Node* VI is executed, at the end of one iteration, the output variables contain the results from steps 1 and 2 (fine point location and constrained curve fitting) of MTP Method 1, and only part of the results from step 3. The output variables `A_hat` and `Y_hat` correspond to matrix **B** and vector **w**, respectively, in $\mathbf{v} = -\left(\mathbf{B}^{\mathbf{T}}\mathbf{B}\right)^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{w}$ (equation (4.36)). Equation (4.36) is solved outside the *Formula Node* VI through regular LABVIEW functions.

Table B. 2: Output Variables of the *Formula Node* VI.

Name	Type	Description
`inters_ estim_x`	integer	Initial *x*-coordinate estimate of cross-hairs intersection. This value is calculated at the beginning of the routine and is only required within the *Formula Node* VI for other calculations.
`inters_ estim_y`	integer	Initial *y*-coordinate estimate of cross-hairs intersection. This value is only required within the *Formula Node* VI for other calculations.
`unconst_ intersec_x`	float	Cross-hairs *x*-coordinate obtained through unconstrained curve fitting applied to the points in lines 1 and 2.
`unconst_ intersec_y`	float	Cross-hairs *y*-coordinate obtained through unconstrained curve fitting applied to the points in lines 1 and 2.
`intersec_x`	float	Cross-hairs *x*-coordinate obtained through curve fitting applied to the points in lines 1 and 2, under a perpendicular constraint.
`intersec_y`	float	Cross-hairs *y*-coordinate obtained through curve fitting applied to the points in lines 1 and 2, under a perpendicular constraint.
`line1_x`	[arbitrary size] floating point array	Array containing the floating point *x*-coordinates of all the points in the digital image that are associated with line 1 of the cross-hairs target.
`line1_y`	[arbitrary size] floating point array	Array containing the floating point *y*-coordinates of all the points in the digital image that are associated with line 1 of the cross-hairs target.
`line2_x`	[arbitrary size] floating point array	Same as `line1_x` but for line 2.
`line2_y`	[arbitrary size] floating point array	Same as `line1_y` but for line 2.
`bool_error`	binary/boolean	Boolean variable indicates whether or not the initial estimate of the cross-hairs target position (i.e. `inters_estim_x` and `inters_estim_y`) was successfully calculated at the beginning of the routine.
`bool_grad_H`	binary/boolean	Auxiliary variable indicates whether or not convergence in the Newton-Raphson iterative method was achieved at the end of one iteration.
`N1`	integer	Number of total points collected for line 1.
`N2`	integer	Number of total points collected for line 2.
`A_hat`	[4,2] floating point matrix	Matrix **B** in equation (4.36).
`Y_hat`	[4] floating point array	Vector **w** in equation (4.36).
`beta`	float	Used to define relative target orientation, $\tilde{\theta}_z$, through $\tilde{\theta}_z = \tan^{-1}(beta)$

C Code

Let it be observed that in some cases the syntax for defining variables in the LABVIEW *Formula Node* VI does not exactly coincide with the syntax used in regular C code. For instance, the definition of a floating point variable in C is performed through the embedded operators `float` or `double`, depending on the required precision. In the *Formula Node* VI floating point variables are defined through the operators `float`, `float32` and `float64`. Similar modifications might be required for other variable definitions if the following code is to be edited and run in a C/C++ compiler.

```
1.  target_detect.c – Program use to detect the cross-hairs target in the digital image
2.  through a Newton-Raphson iterative approach.
3.  Copyright (C) <2011> Carlos A. Montes-Solano carlosm@g.clemson.edu
4.
5.  This program is free software: you can redistribute it and/or modify
6.  it under the terms of the GNU General Public License as published by
7.  the Free Software Foundation, either version 3 of the License, or
8.  (at your option) any later version.
9.
10. This program is distributed in the hope that it will be useful,
11. but WITHOUT ANY WARRANTY; without even the implied warranty of
12. MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13. GNU General Public License for more details.
14.
15. You should have received a copy of the GNU General Public License
16. along with this program.  If not, see <http://www.gnu.org/licenses/>.
17.
18.
19.
20. int counter, N1=0, N2=0;     /* line1 and line2 have N1 and N2 elements,
    respectively. i.e. line1[0] -> line1[N1-1] */
21. int x_crd, y_crd, current_eucl_dist, aux_int, x_offset;
22. int proximity_thrs=25, abs_zero=20, grad_thrs=100;
23. float64 Num=0, Den=0, sumx_1, sumy_1, sumxy_1, sumx2_1, sumx_2, sumy_2, sumxy_2,
    sumx2_2;
24. float64 ref_x1, ref_x2, ref_y1, ref_y2, intersec_x, intersec_y;
25. float64 unconst_intersec_x, unconst_intersec_y;
26. float64 slope_1, slope_2, ref_slope1, ref_slope2, beta;
27. float32 b1, b2;
28. int  inters_estim_x, inters_estim_y;
29.
30. float32 ref_PMAG_x[4], ref_PMAG_y[4];
31. float32 ref_PMAG_Num_x[4], ref_PMAG_Num_y[4], ref_PMAG_Den_x[4], ref_PMAG_Den_y[4];
32. float64 line1_eq[2], line2_eq[2];
33. /* boolean variables */
34. int bool_pass_intersect, bool_horizontal, bool_vertical, bool_grad_H, bool_grad_V;
35. int test_intesect=1, bool_error=0;
36. int cal_intersect_estimate_only_once=0;
37.
38. float32 A_hat[4][2], Y_hat[4][1];
```

```
39. Y_hat[0][0]=-6*294;
40. Y_hat[1][0]=-8*294;
41. Y_hat[2][0]=0;
42. Y_hat[3][0]=0;
43. A_hat[0][1]=0;
44. A_hat[1][1]=0;
45. A_hat[2][1]=-7;
46. A_hat[3][1]=-7;
47.
48. for (aux_int=0; aux_int!=4; aux_int++) {
49.
50. ref_PMAG_x[aux_int]=0;
51. ref_PMAG_y[aux_int]=0;
52. ref_PMAG_Num_x[aux_int]=0;
53. ref_PMAG_Num_y[aux_int]=0;
54. ref_PMAG_Den_x[aux_int]=0;
55. ref_PMAG_Den_y[aux_int]=0;
56. }
57.
58. N1=0;
59. N2=0;
60. counter=0;
61. bool_pass_intersect=0;
62. test_intesect=1;
63. cal_intersect_estimate_only_once=0;
64. beta=1;
65.
66. for(counter=0; counter!=X*Y; counter++) {
67. x_crd=counter%X;
68. y_crd=counter/X;
69.
70. if(N1==10) {
71. beta=(line1_x[N1-1]-line1_x[0])/(line1_y[N1-1]-line1_y[0]);
72. }
73.
74. /* Calculating an estimate value of the intersection using initial points */
75. if(N1>=20 && cal_intersect_estimate_only_once==0) {
76. if(abs(beta)>=0.27) {
77. if(N1>10 && N2>10)
78. current_eucl_dist=abs(line1_x[N1-1]-line2_x[N2-1])+abs(line1_y[N1-1]-line2_y[N2-1]);
79. else
80. current_eucl_dist=proximity_thrs+1;
81. if(current_eucl_dist<proximity_thrs || N2>=15) {
82. slope_1=(line1_y[N1-1]-line1_y[0])/(line1_x[N1-1]-line1_x[0]);
83. b1=line1_y[N1-1]-line1_x[N1-1]*slope_1;
84. slope_2=(line2_y[N2-1]-line2_y[0])/(line2_x[N2-1]-line2_x[0]);
85. b2=line2_y[N2-1]-line2_x[N2-1]*slope_2;
86. inters_estim_x=(b2-b1)/(slope_1-slope_2);
87. inters_estim_y=slope_1*inters_estim_x+b1;
88. cal_intersect_estimate_only_once=1;
89. }
90. } /* end of if(fabs(beta)>=0.467) */
91. else {
92. if (abs(beta)<0.09) {
93. if(line1_x[N1-1]>X-line1_x[N1-1])
94. x_offset=-50;
95. else
96. x_offset=50;
97. }
98. else {
99. if (beta<0) {
100.  if (X-line1_x[N1-1]<52)
101.  x_offset=X-line1_x[N1-1]-3;
102.  else
103.  x_offset=50;
104.  }
105.  else {
```

```
106.  if (line1_x[N1-1]<52)
107.  x_offset=-line1_x[N1-1]+2;
108.  else
109.  x_offset=-50;
110.  }
111.  }
112.  bool_grad_H=1;   /* bool_grad_H is used as an auxiliary variable here */
113.  aux_int=line1_y[N1-1];
114.  while (bool_grad_H==1 && aux_int<Y-2) {
115.  aux_int=aux_int+2;
116.  current_eucl_dist=line1_x[N1-1];
117.  if(image[aux_int][current_eucl_dist+x_offset]>=grad_thrs) {
118.  bool_grad_H=0;
119.  }
120.  } /* end of while */
121.  if(bool_grad_H==0) {
122.  if (abs(beta)<0.05) {
123.  inters_estim_x=line1_x[N1-1];
124.  inters_estim_y=aux_int+2;     /* use aux_int+2 to improve approximation */
125.  }
126.  else {
127.  slope_1=(line1_y[N1-1]-line1_y[0])/(line1_x[N1-1]-line1_x[0]);
128.  b1=line1_y[N1-1]-line1_x[N1-1]*slope_1;
129.  slope_2=-1/slope_1;
130.  b2=aux_int+2-(current_eucl_dist+x_offset)*slope_2;
131.  inters_estim_x=(b2-b1)/(slope_1-slope_2);
132.  inters_estim_y=slope_1*inters_estim_x+b1;
133.  }
134.  cal_intersect_estimate_only_once=1;
135.  }
136.  Else
137.  bool_error=1;
138.  } /* end of else */
139.  } /* end of if(N1>30) */
140.  /*======================= End of estimation process ============================*/
141.
142.  bool_pass_intersect=0;
143.  if(cal_intersect_estimate_only_once==1 && abs(y_crd-inters_estim_y)<70 &&
    abs(x_crd-inters_estim_x)<70) {
144.  if(abs(inters_estim_x-x_crd)+abs(inters_estim_y-y_crd)<proximity_thrs+70) {
145.  if(test_intesect==1){
146.  /* Determining reference angle */
147.  if(abs(beta)>=0.177) {
148.  ref_x1=line1_x[N1-1];
149.  ref_y1=line1_y[N1-1];
150.  ref_x2=line2_x[N2-1];
151.  ref_y2=line2_y[N2-1];
152.  ref_slope1=(line1_y[N1-1]-line1_y[0])/(line1_x[N1-1]-line1_x[0]);
153.  ref_slope2=(line2_y[N2-1]-line2_y[0])/(line2_x[N2-1]-line2_x[0]);
154.  }
155.  test_intesect=0;
156.  }
157.  b1=inters_estim_x-x_crd;
158.  b2=inters_estim_y-y_crd;
159.  sumx_1=(b1*b1)+(b2*b2);
160.  current_eucl_dist=pow(sumx_1,0.5);
161.  bool_pass_intersect=1;
162.  if(current_eucl_dist<proximity_thrs+delta_r) {
163.  bool_horizontal=0;
164.  if (current_eucl_dist>proximity_thrs &&       image[y_crd][x_crd]>200 &&
    Bool_Ref==1) {
165.  if (abs(beta)>=0.177) {
166.  b1=inters_estim_y-ref_slope1*inters_estim_x;
167.  b2=inters_estim_y-ref_slope2*inters_estim_x;
168.  if(abs(y_crd-(ref_slope1*x_crd+b1))>2*proximity_thrs/3 && abs(y_crd-
    (ref_slope2*x_crd+b2))>2*proximity_thrs/3 && abs(x_crd-(y_crd-
```

```
    b1)/ref_slope1)>2*proximity_thrs/3 && abs(x_crd-(y_crd-
    b2)/ref_slope2)>2*proximity_thrs/3)
169.  bool_horizontal=1;
170.  /* end of if (fabs(beta)>=0.177) */
171.  else {
172.  if(abs(inters_estim_x-x_crd)>2*proximity_thrs/3 && abs(inters_estim_y-
    y_crd)>2*proximity_thrs/3)
173.  bool_horizontal=1;
174.  } /* end of else */
175.
176.  /* Calculate reference elements */
177.  if (bool_horizontal==1) {
178.
179.  if(ref_PMAG_x[0]==0 || (abs(x_crd-ref_PMAG_x[0])+abs(y_crd-
    ref_PMAG_y[0])<proximity_thrs)) {
180.  if(ref_PMAG_x[0]==0) {
181.  ref_PMAG_x[0]=x_crd;
182.  ref_PMAG_y[0]=y_crd;
183.  }
184.
185.  ref_PMAG_Num_x[0]+=image[y_crd][x_crd]*x_crd;
186.  ref_PMAG_Num_y[0]+=image[y_crd][x_crd]*y_crd;
187.  }
188.  else if(ref_PMAG_x[1]==0 || (abs(x_crd-ref_PMAG_x[1])+abs(y_crd-
    ref_PMAG_y[1])<proximity_thrs)) {
189.  if(ref_PMAG_x[1]==0) {
190.  ref_PMAG_x[1]=x_crd;
191.  ref_PMAG_y[1]=y_crd;
192.  }
193.
194.  ref_PMAG_Num_x[1]+=image[y_crd][x_crd]*x_crd;
195.  ref_PMAG_Num_y[1]+=image[y_crd][x_crd]*y_crd;
196.  }
197.  else if(ref_PMAG_x[2]==0 || (abs(x_crd-ref_PMAG_x[2])+abs(y_crd-
    ref_PMAG_y[2])<proximity_thrs)) {
198.  if(ref_PMAG_x[2]==0) {
199.  ref_PMAG_x[2]=x_crd;
200.  ref_PMAG_y[2]=y_crd;
201.  }
202.
203.  ref_PMAG_Num_x[2]+=image[y_crd][x_crd]*x_crd;
204.  ref_PMAG_Num_y[2]+=image[y_crd][x_crd]*y_crd;
205.  }
206.  else if(ref_PMAG_x[3]==0 || (abs(x_crd-ref_PMAG_x[3])+abs(y_crd-
    ref_PMAG_y[3])<proximity_thrs)) {
207.  if(ref_PMAG_x[3]==0) {
208.  ref_PMAG_x[3]=x_crd;
209.  ref_PMAG_y[3]=y_crd;
210.  }
211.  ref_PMAG_Num_x[3]+=image[y_crd][x_crd]*x_crd;
212.  ref_PMAG_Num_y[3]+=image[y_crd][x_crd]*y_crd;
213.  }
214.
215.  } /* end of reference elements' calculation */
216.  } /* end of if (current_eucl_dist>proximity_thrs && *p==255) */
217.
218.  x_crd=0; /* By setting the coordinates of the current pixel to zero, the pixel is
    disregarded by the rest of the code */
219.  y_crd=0;
220.
221.  } /* end of if(current_eucl_dist<proximity_thrs+25) */
222.  }
223.  } /* end of if(cal_intersect_estimate_only_once) */
224.
225.
226.  /* Determining the type of transition: horizontal or vertical */
227.  bool_horizontal=0;
```

```
228.  bool_vertical=0;
229.  bool_grad_H=0;
230.  bool_grad_V=0;
231.  if (x_crd<X-2 && x_crd>2 && y_crd<Y-2 && y_crd>2) {
232.  if (image[y_crd][x_crd]<abs_zero && image[y_crd][x_crd+1]>abs_zero &&
    image[y_crd][x_crd+2]>abs_zero) {
233.  aux_int=4;
234.  while (bool_horizontal==0 && x_crd+aux_int<X && aux_int<proximity_thrs+30) {
235.  if(image[y_crd][x_crd+aux_int-1]<abs_zero && image[y_crd][x_crd+aux_int]<abs_zero
    && image[y_crd-2][x_crd+aux_int]<abs_zero && image[y_crd+2][x_crd+aux_int]<abs_zero)
236.  bool_horizontal=1;
237.  if(abs(abs_zero-image[y_crd][x_crd+aux_int-1])>grad_thrs)
238.  bool_grad_H=1;
239.  aux_int++;
240.  }
241.  }
242.  else if (N1>10 && abs(beta)<0.467 && image[y_crd][x_crd]<abs_zero &&
    image[y_crd+1][x_crd]>abs_zero && image[y_crd+2][x_crd]>abs_zero) {
243.  aux_int=4;
244.  while (bool_vertical==0 && y_crd+aux_int<Y && aux_int<proximity_thrs+45) {
245.  if (image[y_crd+aux_int-1][x_crd]<abs_zero && image[y_crd+aux_int][x_crd]<abs_zero
    && image[y_crd+aux_int][x_crd-2]<abs_zero && image[y_crd+aux_int][x_crd+2]<abs_zero)
246.  bool_vertical=1;
247.  if(abs(abs_zero-image[y_crd+aux_int-1][x_crd])>grad_thrs)
248.  bool_grad_V=1;
249.  aux_int++;
250.  } /* end of while */
251.  }
252.  } /* end of simultaneous transition analysis */
253.
254.
255.  if (bool_vertical==1 && bool_grad_V==1) {
256.  if(x_crd-5>0 && x_crd+5<X && y_crd+4<Y) {
257.  if (image[y_crd+4][x_crd-5]>abs_zero && image[y_crd+4][x_crd+5]>abs_zero);
258.  else
259.  bool_vertical=0;
260.  }
261.  Else
262.  bool_vertical=0;
263.  }
264.
265.
266.  /****************** Calculating the centroid ******************/
267.  if ((bool_horizontal==1 && bool_grad_H==1) || (bool_vertical==1 && bool_grad_V==1))
    {
268.  bool_grad_H=0;
269.  bool_grad_V=0;
270.  Num=0;
271.  Den=0;
272.  aux_int=1;
273.  if (bool_horizontal==1) {
274.  while (x_crd+aux_int<X && image[y_crd][x_crd+aux_int]>abs_zero) {
275.
276.  if (image[y_crd][x_crd+aux_int]>abs_zero) {
277.  Num+=image[y_crd][x_crd+aux_int]*(x_crd+aux_int);
278.  Den+=image[y_crd][x_crd+aux_int];
279.  }
280.  if(abs(abs_zero-image[y_crd][x_crd+aux_int])>grad_thrs)
281.  bool_grad_H=1;
282.  aux_int++;
283.  }
284.  }
285.  else {
286.  while(y_crd+aux_int<Y && image[y_crd+aux_int][x_crd]>abs_zero) {
287.  if (image[y_crd+aux_int][x_crd]>abs_zero) {
288.  Num+=image[y_crd+aux_int][x_crd]*(y_crd+aux_int);
289.  Den+=image[y_crd+aux_int][x_crd];
```

```
290.  }
291.  if(abs(abs_zero-image[y_crd+aux_int][x_crd])>grad_thrs)
292.  bool_grad_V=1;
293.  aux_int++;
294.  }
295.  }
296.
297.  if (Num>0 && Den>0 && (bool_grad_H==1 || bool_grad_V==1)) {
298.  if (N1==0) {
299.  line1_x[N1]=Num/Den;
300.  line1_y[N1]=y_crd;
301.  N1++;
302.  } /* end of if */
303.  else {
304.  if (bool_horizontal==1) { /*intersect_x/y are used temporarily to calculate the
    centroid */
305.  intersec_x=Num/Den;
306.  intersec_y=y_crd;
307.  }
308.  else {
309.  intersec_x=x_crd;
310.  intersec_y=Num/Den;
311.  }
312.
313.  if(N1<11) {
314.  if (abs(intersec_x-line1_x[N1-1])+abs(intersec_y-line1_y[N1-1])<2*proximity_thrs) {
315.  line1_x[N1]=intersec_x;
316.  line1_y[N1]=intersec_y;
317.  N1++;
318.  }
319.  else {
320.  line2_x[N2]=intersec_x;
321.  line2_y[N2]=intersec_y;
322.  N2++;
323.  }
324.  }
325.  else if (abs(beta)<0.177) {
326.  if (bool_horizontal==1) {
327.  line1_x[N1]=intersec_x;
328.  line1_y[N1]=intersec_y;
329.  N1++;
330.  }
331.  else {
332.  line2_x[N2]=intersec_x;
333.  line2_y[N2]=intersec_y;
334.  N2++;
335.  }
336.  }
337.  else {
338.  if (N2==0 || bool_pass_intersect==0) {
339.  if (N2==0) {
340.  if (abs(intersec_x-line1_x[N1-1])+abs(intersec_y-line1_y[N1-1])<proximity_thrs) {
341.  line1_x[N1]=intersec_x;
342.  line1_y[N1]=intersec_y;
343.  N1++;
344.  }
345.  else {
346.  line2_x[N2]=intersec_x;
347.  line2_y[N2]=intersec_y;
348.  N2++;
349.  }
350.  }
351.  else if (abs(intersec_x-line1_x[N1-1])+abs(intersec_y-line1_y[N1-
    1])<abs(intersec_x-line2_x[N2-1])+abs(intersec_y-line2_y[N2-1])) {
352.  line1_x[N1]=intersec_x;
353.  line1_y[N1]=intersec_y;
354.  N1++;
```

```
355.  }
356.  else {
357.  line2_x[N2]=intersec_x;
358.  line2_y[N2]=intersec_y;
359.  N2++;
360.  }
361.  } /* end of if (N2==10 || bool_pass_intersect==0) */
362.  else if (bool_pass_intersect==1) {
363.  /* the following if/else statements are used to avoid dividing by very small
    numbers, which leads to erroneous results */
364.  if (abs(intersec_x-ref_x1)+abs(intersec_y-ref_y1)<proximity_thrs)
365.  slope_1=(intersec_y-line1_y[N1-10])/(intersec_x-line1_x[N1-10]);
366.  else
367.  slope_1=(intersec_y-ref_y1)/(intersec_x-ref_x1);
368.  if (abs(intersec_x-ref_x2)+abs(intersec_y-ref_y2)<proximity_thrs)
369.  slope_2=(intersec_y-line2_y[N2-10])/(intersec_x-line2_x[N2-10]);
370.  else
371.  slope_2=(intersec_y-ref_y2)/(intersec_x-ref_x2);
372.
373.  if (abs(ref_slope1-slope_1)<abs(ref_slope2-slope_2)) {
374.  line1_x[N1]=intersec_x;
375.  line1_y[N1]=intersec_y;
376.  N1++;
377.  }
378.  else {
379.  line2_x[N2]=intersec_x;
380.  line2_y[N2]=intersec_y;
381.  N2++;
382.  }
383.  }
384.  } /* end of else */
385.  } /* end of else */
386.  } /* end of if (Num>0 && Den>0) */
387.  } /* end of if ((bool_horizontal && bool_grad_H) || (bool_vertical && bool_grad_V))
    */
388.  /********************** End of centroid calculation ****************************/
389.
390.  } /* end of for (p=img.Begin()...) */
391.
392.
393.  /*==================== Curve Fitting ====================*/
394.  sumx_1=0;
395.  sumy_1=0;
396.  sumxy_1=0;
397.  sumx2_1=0;
398.  sumx_2=0;
399.  sumy_2=0;
400.  sumxy_2=0;
401.  sumx2_2=0;
402.  if (N1>N2)
403.  counter=N1;
404.  else
405.  counter=N2;
406.  for(aux_int=0; aux_int<counter; aux_int++) {
407.  if (aux_int<N1) {
408.  sumx_1+=line1_x[aux_int];
409.  sumy_1+=line1_y[aux_int];
410.  sumxy_1+=line1_x[aux_int]*line1_y[aux_int];
411.  sumx2_1+=pow(line1_x[aux_int],2);
412.  }
413.  if (aux_int<N2) {
414.  sumx_2+=line2_x[aux_int];
415.  sumy_2+=line2_y[aux_int];
416.  sumxy_2+=line2_x[aux_int]*line2_y[aux_int];
417.  sumx2_2+=pow(line2_x[aux_int],2);
418.  }
419.  }
```

```
420.  line1_eq[0]=((N1*sumxy_1)-(sumx_1*sumy_1))/((N1*sumx2_1)-pow(sumx_1,2));
421.  line1_eq[1]=(sumy_1-line1_eq[0]*sumx_1)/N1;
422.  Num=N1*sumxy_1-sumx_1*sumy_1;
423.  beta=1/line1_eq[0];
424.
425.  line2_eq[0]=((N2*sumxy_2)-(sumx_2*sumy_2))/((N2*sumx2_2)-pow(sumx_2,2));
426.  line2_eq[1]=(sumy_2-line2_eq[0]*sumx_2)/N2;
427.  Num=N2*sumxy_2-sumx_2*sumy_2;
428.
429.  unconst_intersec_x=(line2_eq[1]-line1_eq[1])/(line1_eq[0]-line2_eq[0]);
430.  unconst_intersec_y=line1_eq[0]*unconst_intersec_x+line1_eq[1];
431.  intersec_x=unconst_intersec_x;
432.  intersec_y=unconst_intersec_y;
433.
434.
435.  /* Calculation of reference elements for PMAG estimation */
436.  ref_PMAG_x[0]=ref_PMAG_Num_x[0]/ref_PMAG_Den_x[0];
437.  ref_PMAG_y[0]=ref_PMAG_Num_y[0]/ref_PMAG_Den_y[0];
438.  ref_PMAG_x[1]=ref_PMAG_Num_x[1]/ref_PMAG_Den_x[1];
439.  ref_PMAG_y[1]=ref_PMAG_Num_y[1]/ref_PMAG_Den_y[1];
440.  ref_PMAG_x[2]=ref_PMAG_Num_x[2]/ref_PMAG_Den_x[2];
441.  ref_PMAG_y[2]=ref_PMAG_Num_y[2]/ref_PMAG_Den_y[2];
442.  ref_PMAG_x[3]=ref_PMAG_Num_x[3]/ref_PMAG_Den_x[3];
443.  ref_PMAG_y[3]=ref_PMAG_Num_y[3]/ref_PMAG_Den_y[3];
444.
445.
446.  /* calculating x1 (x1 is the x-coordinate of the first reference pt after being
    rotated (corrected)) */
447.  b1=cos(atan(-line2_eq[0]))*ref_PMAG_x[0]-sin(atan(-line2_eq[0]))*ref_PMAG_y[0];
448.  /* calculating x2 */
449.  b2=cos(atan(-line2_eq[0]))*ref_PMAG_x[1]-sin(atan(-line2_eq[0]))*ref_PMAG_y[1];
450.  A_hat[0][0]=5.6*(b2-b1);      /* delta_x1 */
451.
452.  /* calculating x3 (x1 is the x-coordinate of the first reference pt after being
    rotated (corrected)) */
453.  b1=cos(atan(-line2_eq[0]))*ref_PMAG_x[2]-sin(atan(-line2_eq[0]))*ref_PMAG_y[2];
454.  /* calculating x4 */
455.  b2=cos(atan(-line2_eq[0]))*ref_PMAG_x[3]-sin(atan(-line2_eq[0]))*ref_PMAG_y[3];
456.  A_hat[1][0]=5.6*(b2-b1);      /* delta_x2 */
457.
458.  /* calculating y1 (y1 is the x-coordinate of the first reference pt after being
    rotated (corrected)) */
459.  b1=sin(atan(-line2_eq[0]))*ref_PMAG_x[0]+cos(atan(-line2_eq[0]))*ref_PMAG_y[0];
460.  /* calculating y3 */
461.  b2=sin(atan(-line2_eq[0]))*ref_PMAG_x[2]+cos(atan(-line2_eq[0]))*ref_PMAG_y[2];
462.  A_hat[2][0]=5.6*(b2-b1);      /* delta_y1 */
463.
464.  /* calculating y2 */
465.  b1=sin(atan(-line2_eq[0]))*ref_PMAG_x[1]+cos(atan(-line2_eq[0]))*ref_PMAG_y[1];
466.  /* calculating y4 */
467.  b2=sin(atan(-line2_eq[0]))*ref_PMAG_x[3]+cos(atan(-line2_eq[0]))*ref_PMAG_y[3];
468.  A_hat[3][0]=5.6*(b2-b1);      /* delta_y2 */
469.
470.
471.  /*=============== Constrained Curve Fitting (Newton Raphson) ==============*/
472.
473.  /**** define:  m1(n)=slope_1;
474.  m1(n+1)=slope_2
475.  f(m1)=ref_slope1
476.  f_prime(m1)=ref_slope2     ****/
477.
478.  bool_grad_H=0;
479.  float Newton_iteration_thrs=0.0000072;
480.  slope_1=line1_eq[0];
481.  for(aux_int=0; aux_int<20; aux_int++) {
482.  ref_slope2=sumx2_1-(sumx_1*sumx_1/N1)+(sumxy_2-
    sumx_2*sumy_2)*(2/pow(slope_1,3))+(3/pow(slope_1,4))*(sumx2_2-sumx_2*sumx_2/N2);
```

```
483.
484.  if(ref_slope2==0) {
485.  break;
486.  }
487.  else {
488.  ref_slope1=(sumx_1*sumy_1/N1)-sumxy_1+slope_1*(sumx2_1-
    sumx_1*sumx_1/N1)+(1/pow(slope_1,2))*((sumx_2*sumy_2/N2)-
    sumxy_2)+(1/pow(slope_1,3))*((sumx_2*sumx_2/N2)-sumx2_2);
489.  slope_2=slope_1-ref_slope1/ref_slope2;
490.  if (aux_int>0 && abs(slope_2-slope_1)>abs(slope_1-intersec_x)) {
491.  slope_1=intersec_x;
492.  break;
493.  }
494.  else if(abs(slope_2-slope_1)<Newton_iteration_thrs*abs(slope_1)) {
495.  bool_grad_H=1;
496.  slope_1=slope_2;
497.  break;
498.  }
499.
500.  intersec_x=slope_1;   /* m1(n-1) */
501.  slope_1=slope_2;    /* m1(n) */
502.  }
503.  } /* end of for */
504.
505.  /*================ End of Curve Fitting ==================*/
506.
507.  line1_eq[0]=slope_1;
508.  line1_eq[1]=(sumy_1-line1_eq[0]*sumx_1)/N1;
509.
510.  line2_eq[0]=-1/slope_1;
511.  line2_eq[1]=(sumy_2-line2_eq[0]*sumx_2)/N2;
512.
513.  intersec_x=(line2_eq[1]-line1_eq[1])/(line1_eq[0]-line2_eq[0]);
514.  intersec_y=line1_eq[0]*intersec_x+line1_eq[1];
```

Appendix C

Best Fit Gaussian Estimation through Polynomial Curve Fitting

The two dimensional Gaussian function is known to have the form

$$f(x, A_G, \mu_x, \sigma) = A_G e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}} \tag{C.1}$$

with x being the independent variable, mean μ_x, standard deviation σ_x and maximum amplitude A_G. Equation (C.1) can be converted into a second order polynomial through the application of the ln() function to both sides of the equation. This yields

$$\ln(f) = \ln(A_G) - \left(\frac{(x-\mu_x)^2}{2\sigma_x^2} \right) \tag{C.2}$$

$$\ln(f) = \left(\ln(A_G) - \frac{\mu_x^2}{2\sigma_x^2} \right) + \frac{\mu_x}{\sigma_x^2} x - \frac{1}{2\sigma_x^2} x^2 \tag{C.3}$$

Equation (C.3) can be expressed as $y_G = a + bx + cx^2$ (where $\ln(f) = y_G$), which implies that

$$\begin{cases} \sigma_x = \sqrt{-\dfrac{1}{2c}} \\ \mu_x = -\dfrac{b}{2c} \\ A_G = e^{\left(a - b^2/4c\right)} \end{cases} \tag{C.4}$$

The second order polynomial is now used to define a system of n equations $\mathbf{y_G} = \mathbf{\Lambda \tilde{a}}$, where $\mathbf{\Lambda}$ is an n-by-3 Vandermonde matrix and $\mathbf{\tilde{a}}$ is the vector containing the unknowns. This is

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix}, \quad \tilde{\mathbf{a}} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{C.5}$$

Common least-squares can now be applied to determine parameters *a*, *b* and *c* using experimental data. Let a set of experimental data be $\left\{\mathbf{x}^{\mathbf{exp}}, \mathbf{y}^{\mathbf{exp}}\right\}$, where $\mathbf{x}^{\mathbf{exp}} = \{x_1, x_2, x_3, ..., x_n\}$ and $\mathbf{y}^{\mathbf{exp}} = \{y_1, y_2, y_3, ..., y_n\}$. It is desired to find

$$\mathbf{e}(\tilde{\mathbf{a}}) = \min_{\tilde{\mathbf{a}}} \left\| \hat{\mathbf{y}}^{\mathbf{exp}} - \mathbf{y}_{\mathbf{G}} \right\|^2 \tag{C.6}$$

where $\hat{\mathbf{y}}^{\mathbf{exp}} = \ln\left(\mathbf{y}^{\mathbf{exp}}\right)$. The solution to (C.6) is known from the regular least-squares problem, and has the form

$$\tilde{\mathbf{a}} = \left(\boldsymbol{\Lambda}^T \boldsymbol{\Lambda}\right)^{-1} \boldsymbol{\Lambda}^T \hat{\mathbf{y}}^{\mathbf{exp}} \tag{C.7}$$

Once $\tilde{\mathbf{a}}$ is obtained, the parameters of the best fit Gaussian curve (μ_x, σ_x and A_G) can be calculated through (C.4).

Appendix D

Processing Threads: LABVIEW Code

Figure D. 1 shows the hardware configuration of the testbed. All three threads are enclosed by rectangular boxes. The control thread includes a National Instruments real time controller with a Field Programmable Gate Array (FPGA), two H-Bridge brushed DC servo drive modules and one brushless servo drive module. Axes motion is achieved through two DC brushed servo motors and one brushless rotary stage utilized in the camera mount. Three rotary encoders, one for each servo drive, are used according to the control configuration explained in Chapter Six. The entire control application (control thread), providing PWM input voltages to all servos, is run inside the controller. Controller's tasks include PID control, Lead-Lag compensation and system dynamics modeling.

Image acquisition and image processing are achieved, in the vision thread, using a Firewire camera connected to an NI Compact Vision System. In addition, the vision thread includes a megapixel lens. Finally, the components of the host thread are a stand-alone computer and an external LCD screen. Path planning, in the diagram of Figure D. 1 is performed off-line; the corresponding data files are loaded into the control- and host-threads for real-time operation. A detailed list of all hardware components utilized in the experimental implementation of MTP is found in Table D. 1. Communication between the different processing threads is accomplished through a dedicated Ethernet network.

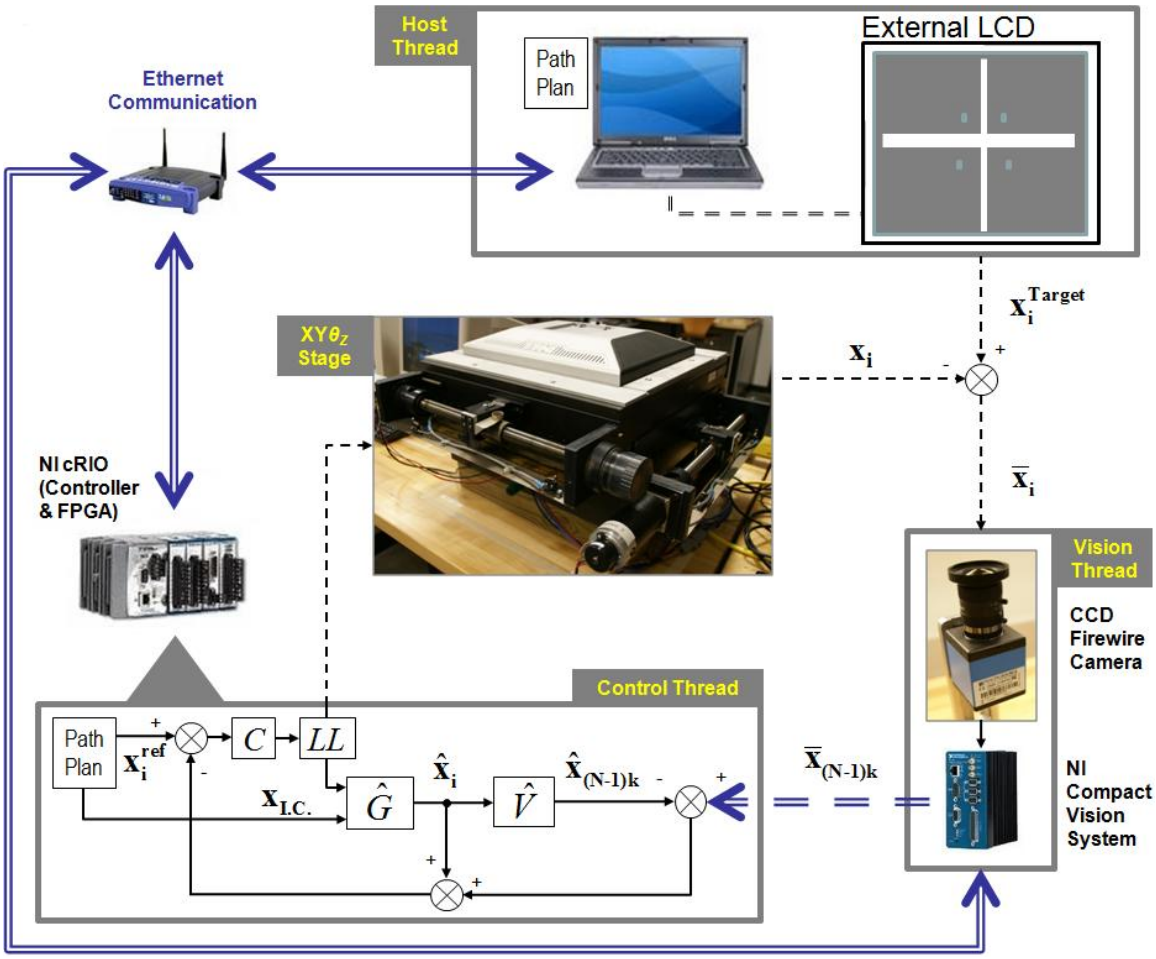


Figure D. 1: Experimental hardware configuration.

All three software applications associated with the three processing threads were programmed using the NI LABVIEW graphical programming environment [101]. The front panel and corresponding block diagrams developed for all three threads are shown in Figure D. 2, Figure D. 3 and Figure D. 4. This code can be downloaded from the author's personal website: **http://people.clemson.edu/~128218403**.

Table D. 1: Hardware components required for MTP experimental implementation.

Thread	Component (Model)	Manufacturer	No. of Components	Description
Control	NI 9014 CompactRIO	National Instruments	1	Embedded controller w/ FPGA, 400 MHz processor, 2 GB nonvolatile storage, 128 MB DRAM memory.
	NI 9505 Module	National Instruments	2	Full H-Bridge Brushed DC Servo Drive Module
	NI 9514 Module	National Instruments	1	C Series Servo Drive Interface with Encoder Feedback (Brushless Servo Drive)
	1050LT-E1000LD Servo Motor	Aerotech [102]	2	Brush, Rotary DC Servomotors
	1000-Counts Rotary Encoder	Generic	2	1000 counts/rotation Rotary Encoder
	ADRS-100-MA-AS Rotary Stage	Aerotech	1	Mechanical-Bearing Rotary Stage w/ Brushless Servo Motor
	MXH Multiplier Board	Aerotech	1	Programmable encoder multiplier with real-time encoder quadrature output (required for rotary stage encoder reading)
Vision	NI CVS 1456	National Instruments	1	Real-Time Compact Vision System for IEEE 1394a Cameras
	DMK 21BF04 CCD Firewire Camera	The Imaging Source [103]	1	1/4" CCD, progressive scan, resolution 640 x 480, protocol: DCAM 1.31, C/CS-mount (WITHOUT lens).
	H0514-MP Megapixel C-Mount Lens	Computar [104]	1	1/2" 5mm f1.4 w/locking Iris & Focus, Megapixel (C Mount)
Host	Desktop PC	N/A	1	N/A
	1907FPt External LCD	Dell [105]	1	19" LCD Monitor, VGA /DVI input

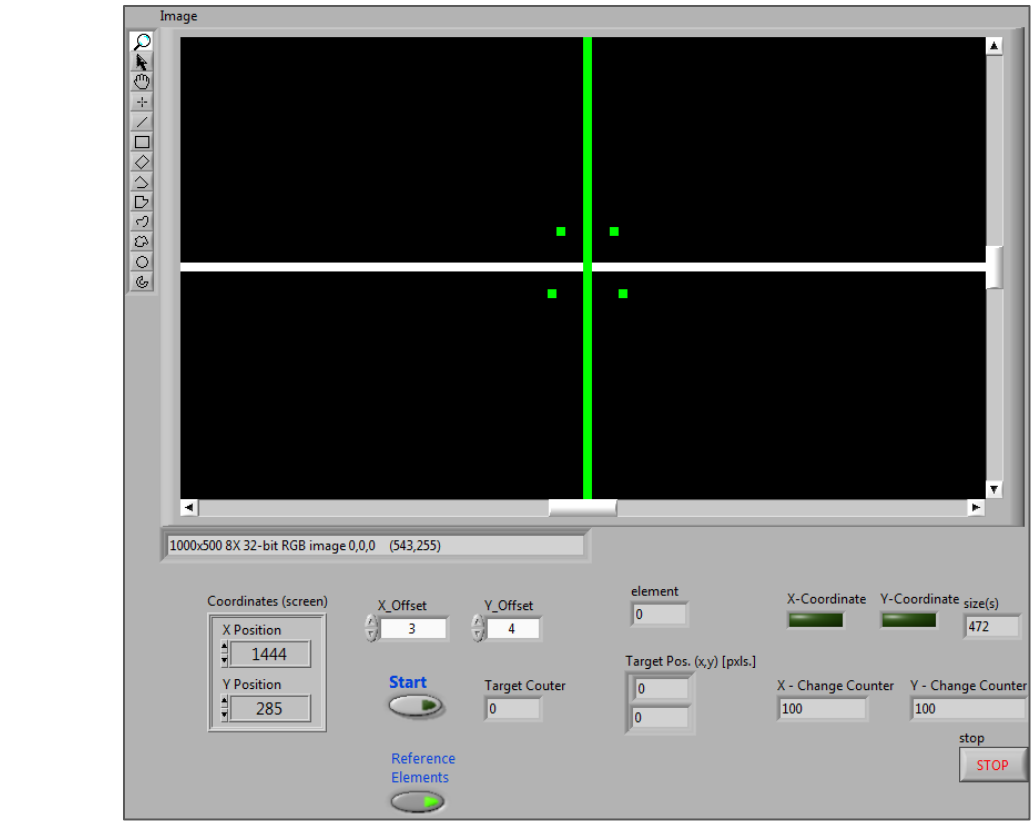


Figure D. 2: Host thread application: front panel (top) and block diagram (bottom).

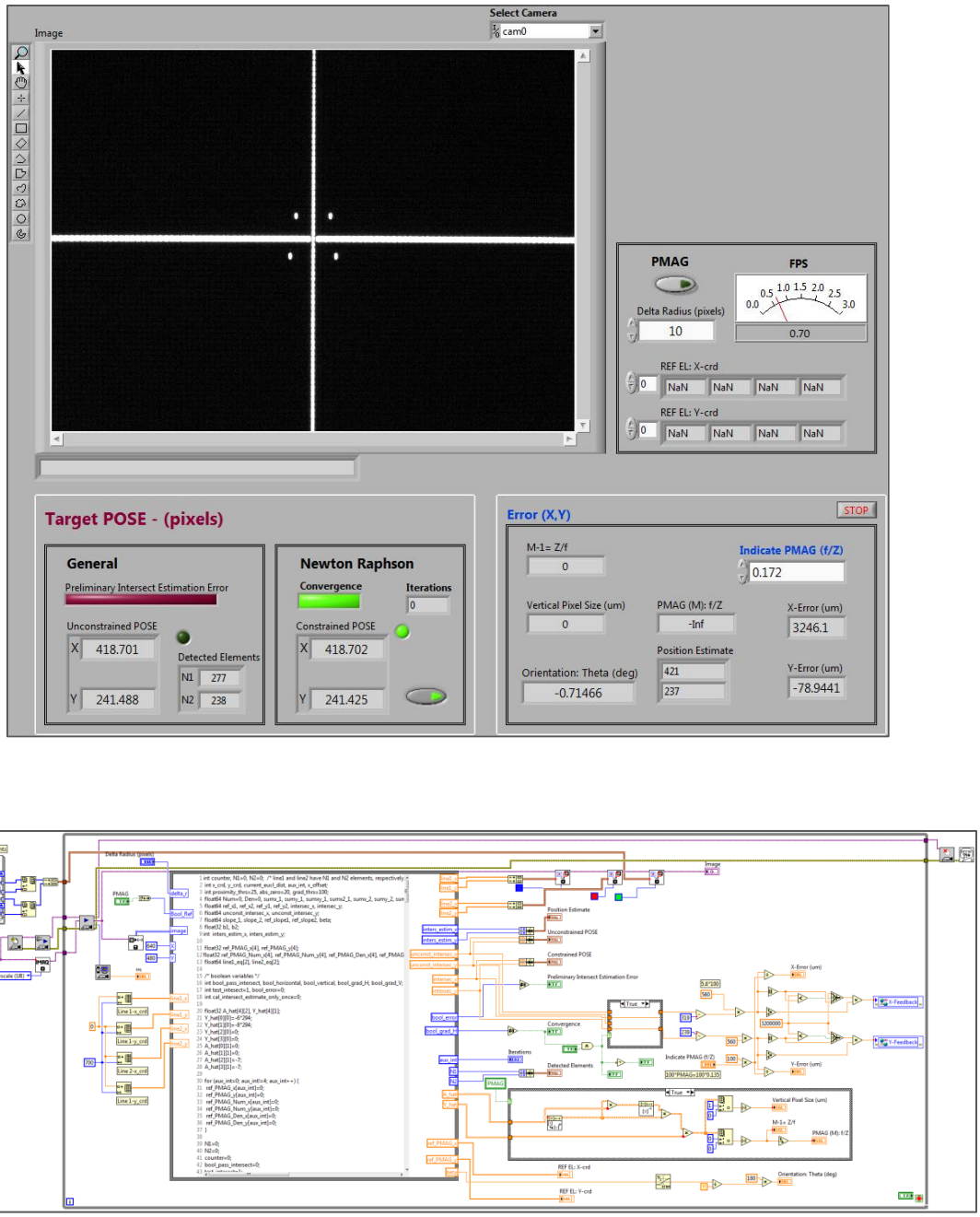


Figure D. 3: Vision thread application running on real-time GPU (NI CVS): front panel (top) and block diagram (bottom).

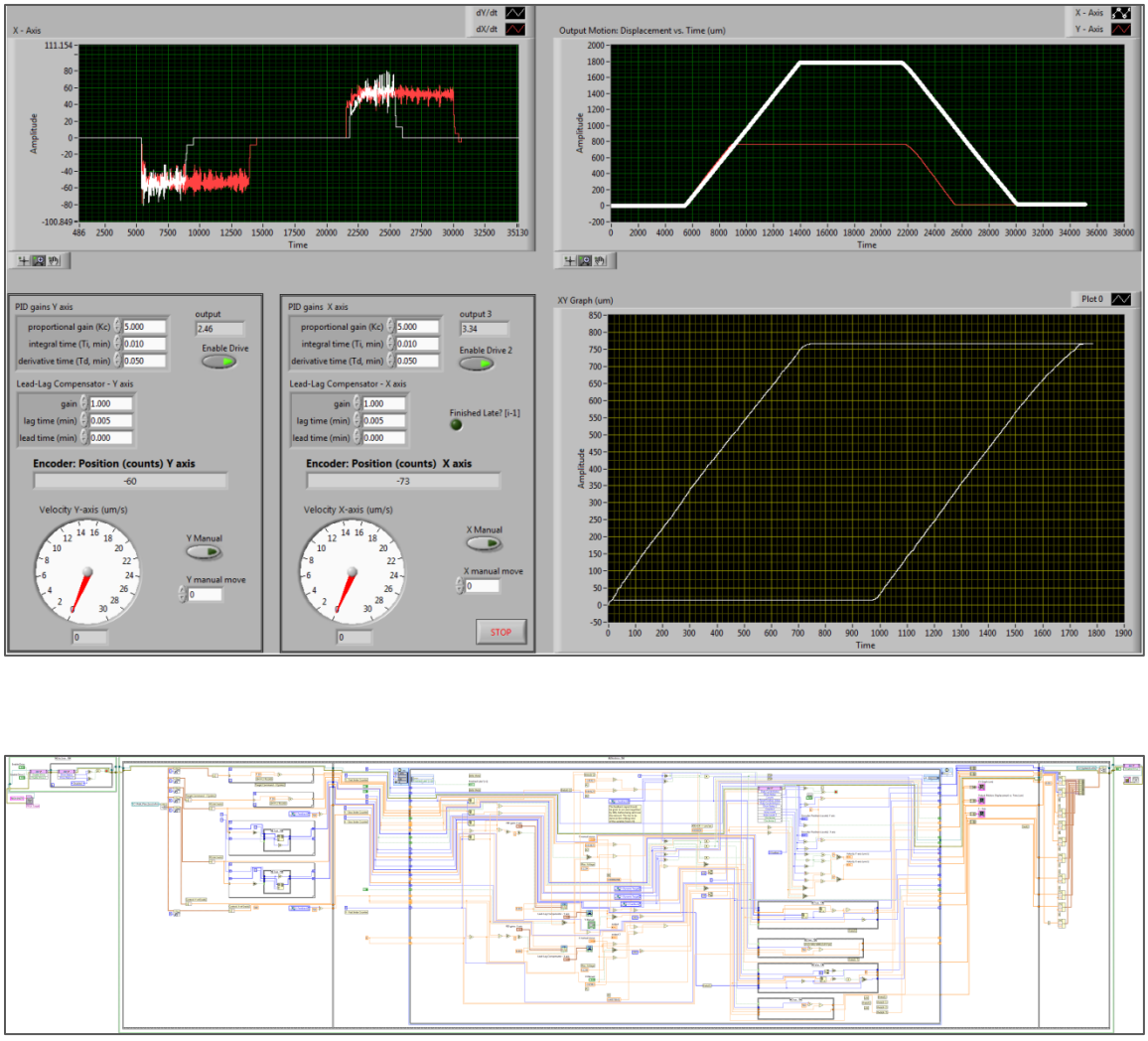


Figure D. 4: Control thread application on real-time controller (NI cRIO): front panel (top) and block diagram (bottom).

BIBLIOGRAPHY

[1] Rosenberg, N., 1963, "Technological Change in the Machine Tool Industry, 1840-1910," The Journal of Economic History, **23**(4) pp. 414-443.

[2] Tlusty, G., 2000, "Manufacturing Equipment and Processes," Prentice-Hall, Upper Saddle River, NJ, Section.

[3] Chang, T.C., Wysk, R.A., and Wang, H.P., 2005, "Computer-Aided Manufacturing (Prentice Hall International Series on Industrial and Systems Engineering)," Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

[4] Slocum, A.H., 1992, "Precision machine design," Society of Manufacturing.

[5] Leviton, D. B., 2004, "Method and Apparatus for Two-Dimensional Absolute Optical Encoding," Method and Apparatus for Two-Dimensional Absolute Optical Encoding.

[6] Jones, B. K., and Nahum, M., 2003, "Systems and Methods for Absolute Positioning using Repeated Quasirandom Pattern".

[7] Chu, D. C., Trujillo, H., Whitney, E., 2006, "Rapid High Resolution Absolute 2-D Encoding by Low Resolution Digital Imaging of a Mathematically Generated 3-Tone Target," Annual Meeting of the American Society for Precision Engineering.

[8] Dornfeld, D., and Lee, D.E., 2007, "Precision manufacturing," Springer Verlag.

[9] Heydemann, P. L. M., 1981, "Determination and Correction of Quadrature Fringe Measurement Errors in Interferometers," Applied Optics, **20**(19) pp. 3382-3384.

[10] Hagiwara, N., Suzuki, Y., and Murase, H., 1992, "A Method of Improving the Resolution and Accuracy of Rotary Encoders using a Code Compensation Technique," IEEE Transactions on Instrumentation and Measurement, **41**(1) pp. 98-101.

[11] Tan, K. K., Zhou, H. X., and Lee, T. H., 2002, "New Interpolation Method for Quadrature Encoder Signals," IEEE Transactions on Instrumentation and Measurement, **51**(5) pp. 1073-1079.

[12] Tan, K. K., and Tang, K. Z., 2005, "Adaptive Online Correction and Interpolation of Quadrature Encoder Signals using Radial Basis Functions," Control Systems Technology, IEEE Transactions on, **13**(3) pp. 370-377.

[13] Ramesh, R., Mannan, M. A., and Poo, A. N., 2000, "Error Compensation in Machine Tools--a Review:: Part I: Geometric, Cutting-Force Induced and Fixture-Dependent Errors," International Journal of Machine Tools and Manufacture, **40**(9) pp. 1235-1256.

[14] Tseng, P. C., 1997, "A Real-Time Thermal Inaccuracy Compensation Method on a Machining Centre," The International Journal of Advanced Manufacturing Technology, **13**(3) pp. 182-190.

[15] Bryan, J., 1990, "International Status of Thermal Error Research (1990)," CIRP Annals-Manufacturing Technology, **39**(2) pp. 645-656.

[16] Mize, C. D., and Ziegert, J. C., 2000, "Durability Evaluation of Software Error Correction on a Machining Center," International Journal of Machine Tools and Manufacture, **40**(10) pp. 1527-1534.

[17] Chen, J. S., Yuan, J., and Ni, J., 1996, "Thermal Error Modelling for Real-Time Error Compensation," The International Journal of Advanced Manufacturing Technology, **12**(4) pp. 266-275.

[18] Hocken, R. J., Trumper, D. L., and Wang, C., 2001, "Dynamics and Control of the UNCC/MIT Sub-Atomic Measuring Machine," CIRP Annals-Manufacturing Technology, **50**(1) pp. 373-376.

[19] Srinivasa, N., and Ziegert, J. C., 1996, "Automated Measurement and Compensation of Thermally Induced Error Maps in Machine Tools," Precision Engineering, **19**(2-3) pp. 112-132.

[20] Woody, B. A., Smith, K. S., Hocken, R. J., 2007, "A Technique for Enhancing Machine Tool Accuracy by Transferring the Metrology Reference from the Machine Tool to the Workpiece," Journal of Manufacturing Science and Engineering, **129**pp. 636.

[21] Ni, J., 1997, "CNC Machine Accuracy Enhancement through Real-Time Error Compensation," Journal of Manufacturing Science and Engineering, **119**pp. 717.

[22] Raksiri, C., and Parnichkun, M., 2004, "Geometric and Force Errors Compensation in a 3-Axis CNC Milling Machine," International Journal of Machine Tools and Manufacture, **44**(12-13) pp. 1283-1291.

[23] Mize, C. D., and Ziegert, J. C., 2000, "Neural Network Thermal Error Compensation of a Machining Center," Precision Engineering, **24**(4) pp. 338-346.

[24] Mou, J., 1997, "A Method of using Neural Networks and Inverse Kinematics for Machine Tools Error Estimation and Correction," Journal of Manufacturing Science and Engineering, **119**pp. 247.

[25] Srinivasa, N., and Ziegert, J. C., 2002, "An application of Fuzzy ARTMAP neural network to real-time learning and prediction of time-variant machine tool error maps," Proceedings of the Third IEEE Conference on Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence, pp. 1725-1730.

[26] Chen, J. S., 1995, "Computer-Aided Accuracy Enhancement for Multi-Axis CNC Machine Tool," International Journal of Machine Tools and Manufacture, **35**(4) pp. 593-605.

[27] Heidenhain GmbH, "Measuring Systems for Machine Tool Inspection and Acceptance Testing".

[28] Zhang, Z., 2002, "A Flexible New Technique for Camera Calibration," Pattern Analysis and Machine Intelligence, IEEE Transactions on, **22**(11) pp. 1330-1334.

[29] Zhang, Z., 1999, "Flexible camera calibration by viewing a plane from unknown orientations," iccv, published by the IEEE Computer Society.

[30] Parker, J. A., Kenyon, R. V., and Troxel, D. E., 2007, "Comparison of Interpolating Methods for Image Resampling," Medical Imaging, IEEE Transactions on, **2**(1) pp. 31-39.

[31] Keys, R., 1981, "Cubic Convolution Interpolation for Digital Image Processing," IEEE Transactions on Acoustics Speech and Signal Processing, **29**(6) pp. 1153-1160.

[32] Quine, B. M., Tarasyuk, V., Mebrahtu, H., 2007, "Determining Star-Image Location: A New Sub-Pixel Interpolation Technique to Process Image Centroids," Computer Physics Communications, **177**(9) pp. 700-706.

[33] Sutton, M. A., MingQi, C., Peters, W. H., 1986, "Application of an Optimized Digital Correlation Method to Planar Deformation Analysis," Image and Vision Computing, **4**(3) pp. 143-150.

[34] Sutton, M. A., Wolters, W. J., Peters, W. H., 1983, "Determination of Displacements using an Improved Digital Correlation Method," Image and Vision Computing, **1**(3) pp. 133-139.

[35] Bruck, H. A., McNeill, S. R., Sutton, M. A., 1989, "Digital Image Correlation using Newton-Raphson Method of Partial Differential Correction," Experimental Mechanics, **29**(3) pp. 261-267.

[36] Lu, H., and Cary, P. D., 2000, "Deformation Measurements by Digital Image Correlation: Implementation of a Second-Order Displacement Gradient," Experimental Mechanics, **40**(4) pp. 393-400.

[37] McNeill, S. R., Peters, W. H., and Sutton, M. A., 1987, "Estimation of Stress Intensity Factor by Digital Image Correlation," Engineering Fracture Mechanics, **28**(1) pp. 101-112.

[38] Hung, P. C., and Voloshin, A. S., 2003, "In-Plane Strain Measurement by Digital Image Correlation," Journal of the Brazilian Society of Mechanical Sciences and Engineering, **25**pp. 215-221.

[39] Sun, Y., Pang, J. H. L., Shi, X., 2006, "Thermal deformation measurement by digital image correlation method," Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. ITHERM'06. The Tenth Intersociety Conference, pp. 921-927.

[40] Lockwood, W. D., and Reynolds, A. P., 1999, "Use and Verification of Digital Image Correlation for Automated 3-D Surface Characterization in the Scanning Electron Microscope," Materials Characterization, **42**(2-3) pp. 123-134.

[41] Schreier, H. W., Garcia, D., and Sutton, M. A., 2004, "Advances in Light Microscope Stereo Vision," Experimental Mechanics, **44**(3) pp. 278-288.

[42] Canny, J., 1987, "A Computational Approach to Edge Detection," Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, pp. 184.

[43] Prewitt, J. M. S., 1970, "Object Enhancement and Extraction," Picture Processing and Psychopictorics, pp. 75-149.

[44] Song, S., Liao, M., and Qin, J., 1990, "Multiresolution Image Motion Detection and Displacement Estimation," Machine Vision and Applications, **3**(1) pp. 17-20.

[45] Fayolle, J., Ducottet, C., and Schon, J. P., 1998, "Application of Multiscale Characterization of Edges to Motion Determination," IEEE Transactions on Signal Processing, **46**(4) pp. 1174-1178.

[46] Sandoz, P., Ravassard, J. C., Dembelé, S., 2000, "Phase-Sensitive Vision Technique for High Accuracy Position Measurement of Moving Targets," IEEE Transactions on Instrumentation and Measurement, **49**(4) pp. 867-872.

[47] Sandoz, P., 2005, "Nanometric Position and Displacement Measurement of the Six Degrees of Freedom by Means of a Patterned Surface Element," Applied Optics, **44**(8) pp. 1449-1453.

[48] Corke, P. I., 1993, "Visual Control of Robot Manipulators - a Review," Visual Serving: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback, pp. 1.

[49] Shirai, Y., and Inoue, H., 1973, "Guiding a Robot by Visual Feedback in Assembling Tasks," Pattern Recognition, **5**(2) pp. 99-106.

[50] Hill, J., and Park, W., 1979, "Real time control of a robot with a mobile camera," 9th International Symposium on Industrial Robots, pp.233-246.

[51] Corke, P. I., Good, M. C., and CSIRO, P., 1992, "Dynamic effects in high-performance visual servoing," 1992 IEEE International Conference on Robotics and Automation, pp. 1838-1843.

[52] Corke, P. I., and Good, M. C., 1996, "Dynamic Effects in Visual Closed-Loop Systems," IEEE Transactions on Robotics and Automation, **12**(5) pp. 671-683.

[53] Hutchinson, S., Hager, G. D., and Corke, P. I., 1996, "A Tutorial on Visual Servo Control," IEEE Transactions on Robotics and Automation, **12**(5) pp. 651-670.

[54] Martinet, P., and Gallice, J., 2002, "Position based visual servoing using a non-linear approach," International Conference on Intelligent Robots and Systems, 1999. IROS'99. 1999 IEEE/RSJ, **1,** pp. 531-536.

[55] Bemporad, A., 2006, "Model predictive control design: new trends and tools," 2006 45th IEEE Conference on Decision and Control, pp. 6678-6683.

[56] Lewis, F.L., and Syrmos, V.L., 1995, "Optimal control," Wiley-Interscience.

[57] Pindyck, R., 1972, "The Discrete-Time Tracking Problem with a Time Delay in the Control," IEEE Transactions on Automatic Control, **17**(3) pp. 397-398.

[58] Karbassi, S. M., 2005, "Time-Optimal Control of Disturbance-Rejection Tracking Systems for Discrete-Time Time-Delayed Systems by State Feedback," Computers and Mathematics with Applications, **50**(8-9) pp. 1415-1424.

[59] Tang, G., Sun, H., and Pang, H., 2008, "Approximately Optimal Tracking Control for Discrete Time-Delay Systems with Disturbances," Progress in Natural Science, **18**(3) pp. 225-232.

[60] See, A., 2005, "Rapid Prototyping Design and Implementation of a Motion Control IntegratedWith an Inexpensive Machine Vision System," Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE, **3,** pp. 2065-2070.

[61] Gangloff, J. A., and de Mathelin, M. F., 2003, "High-Speed Visual Servoing of a 6-Dof Manipulator using Multivariable Predictive Control," Advanced Robotics, **17**(10) pp. 993-1021.

[62] Liu, Y., Hoover, A. W., and Walker, I. D., 2004, "A Timing Model for Vision-Based Control of Industrial Robot Manipulators," IEEE Transactions on Robotics, **20**(5) pp. 891-898.

[63] Sulzer, J., and Kovac, I., 2010, "Enhancement of Positioning Accuracy of Industrial Robots with a Reconfigurable Fine-Positioning Module," Precision Engineering, **34**(2) pp. 201-217.

[64] Pedro, A. R., 2002, "A Note on Principal Point Estimability," Proc. 16 th Int. Conf. Pattern Recognition (ICP), Citeseer.

[65] Brown, D. C., 1971, "Close-Range Camera Calibration," Photogrammetric Engineering, **37**(8) pp. 855-866.

[66] Gennery, D. B., 1979, "Stereo-Camera Calibration." Proc. Image Understanding Workshop.

[67] Isaguirre, A., Pu, P., and Summers, J., 1985, "A new development in camera calibration: Calibrating a pair of mobile vectors," Proc. Int. Conf. Robotics and Automation.

[68] Tsai, R., 1987, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology using Off-the-Shelf TV Cameras and Lenses," IEEE Journal of Robotics and Automation, **3**(4) pp. 323-344.

[69] Faugeras, O., 1993, "Three-dimensional computer vision: a geometric viewpoint," the MIT Press.

[70] Sutton, M.A., Orteu, J.J., and Schreier, H., 2009, "Image Correlation for Shape, Motion and Deformation Measurements: Basic Concepts, Theory and Applications," Springer Publishing Company, Incorporated.

[71] Edmund Optics, "Electronic Imaging Resource Guide," **2010**.

[72] Chen, C., and Lin, M., 2007, "An Improved Adaptive Centroid Estimation Algorithm," TENCON 2006. 2006 IEEE Region 10 Conference, pp. 1-4.

[73] Erwin, K., 1999, "Advanced engineering mathematics".

[74] B.N. Taylor, and P. J. Mohr, 2003, "The NIST Reference on Constants, Units and Uncertainty".

[75] Bury, K.V., 1999, "Statistical distributions in engineering," Cambridge Univ Pr.

[76] Ioannou, P.A., and Sun, J., 1996, "Robust adaptive control," Prentice Hall Englewood Cliffs, NJ.

[77] Xie, H., Sun, L., Rong, W., 2006, "Visual servoing with modified smith predictor for micromanipulation tasks," IEEE International Conference on Mechatronics and Automation, **1,** pp. 71-76.

[78] Abe, N., and Yamanaka, K., 2004, "Smith predictor control and internal model control-a tutorial," SICE 2003 Annual Conference, IEEE, **2,** pp. 1383-1387.

[79] Ramos, C., Martínez, M., Blasco, X., 2002, "Use of Filtered Smith Predictor in DMC".

[80] Bahill, A., 2002, "A Simple Adaptive Smith-Predictor for Controlling Time-Delay Systems: A Tutorial," Control Systems Magazine, IEEE, **3**(2) pp. 16-22.

[81] Zhong, Q.C., 2006, "Robust control of time-delay systems," Springer Verlag.

[82] Normey-Rico, J.E., and Camacho, E.F., 2007, "Control of dead-time processes," Springer Verlag.

[83] Watanabe, K., and Ito, M., 2002, "A Process-Model Control for Linear Systems with Delay," IEEE Transactions on Automatic Control, **26**(6) pp. 1261-1269.

[84] Astrom, K. J., Hang, C. C., and Lim, B. C., 2002, "A New Smith Predictor for Controlling a Process with an Integrator and Long Dead-Time," IEEE Transactions on Automatic Control, **39**(2) pp. 343-345.

[85] Matausek, M. R., and Micic, A. D., 2002, "A Modified Smith Predictor for Controlling a Process with an Integrator and Long Dead-Time," IEEE Transactions on Automatic Control, **41**(8) pp. 1199-1203.

[86] Zhang, W. D., and Sun, Y. X., 1996, "Modified Smith Predictor for Controlling integrator/time Delay Processes," Ind.Eng.Chem.Res, **35**(8) pp. 2769-2772.

[87] Sim, T. P., Hong, G. S., and Lim, K. B., 2002, "Multirate predictor control scheme for visual servo control," Control Theory and Applications, IEE Proceedings-IET, **149,** pp. 117-124.

[88] Corke, P. I., 1994, "High-Performance Visual Closed-Loop Robot Control," Ph.D. Dissertation, Dept. Mechanical and Manufacturing Engineering, University of Melbourne, Australia, July 1994.

[89] Koay, K., and Bugmann, G., 2004, "Compensating intermittent delayed visual feedback in robot navigation," Proceedings of the IEEE Conference on Decision and Control Including the Symposium on Adaptive Processes, Citeseer.

[90] Estrada, T., Lin, H., and Antsaklis, P. J., 2007, "Model-based control with intermittent feedback," MED'06. 14th Mediterranean Conference on Control and Automation, pp. 1-6.

[91] Estrada, T., and Antsaklis, P. J., 2008, "Stability of discrete-time plants using model-based control with intermittent feedback," 2008 16th Mediterranean Conference on Control and Automation, IEEE, pp. 1130-1136.

[92] Hashimoto, K., Ebine, T., and Kimura, H., 2002, "Visual Servoing with Hand-Eye Manipulator-Optimal Control Approach," IEEE Transactions on Robotics and Automation, **12**(5) pp. 766-774.

[93] Tijani, I. B., 2008, "Friction compensation for motion control system using multilayer feedforward network," ISMA 2008. 5th International Symposium on Mechatronics and Its Applications, IEEE, pp. 1-6.

[94] DE WIT, C., 1994, "A Survey of Models, Analysis Tools and Compensation Methods for the Control of Machines with Friction," Automatica, **30**(7) pp. 1083-1138.

[95] Nise, N.S., 2004, "Control systems engineering," Wiley.

[96] Montes, C., Wong, C., Ziegert, J. C., 2011, "Hybrid Command Issuing in 2D Visual Servomechanism," Proceedings of the International Conference on Sustainable Automotive Technologies, Springer.

[97] Montes, C., and Ziegert, J., 2010, "Vision-aided Position Control Method for Manufacturing Machines," Proceedings of the International Conference on Sustainable Automotive Technologies, Springer.

[98] Montes, C. A., Ziegert, J. C., and Mears, L., 2009, "Method to Measure Planar Displacement using Centroid Calculation," Proceedings of the 2009 North American Manufacturing Research Conference.

[99] Montes, C. A., Ziegert, J. C., Mears, L., 2010, " 2-D Absolute Positioning System for Real Time Control Applications," Proceedings of the 2010 Conference of the American Society for Precision Engineering.

[100] Wong, C., Montes, C. A., Mears, L., 2008, "A New Position Feedback Method for Manufacturing Equipment," Proceedings of the ASME International Manufacturing Science and Engineering Conference, pp. 111-120.

[101] © 2011 National Instruments Corporation, Austin, TX.

[102] © 2011 Aerotech, Inc.

[103] © 1991-2011 The Imaging Source Europe GmbH.

[104] 2011 CBC Group (AMERICA) Corp, "Computar".

[105] © 2011 Dell.