VISUAL SERVOING

MOK HENG CHONG
(B.Eng.(Hons.), NUS)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE
2003

ACKNOWLEDGEMENT

I would like to show my gratitude to my supervisor, Professor Poo Aun Neow for his patient and invaluable guidance through the years I have known him.

The other people that have in some ways or other help in my work are Assoc. Prof. Marcelo Ang, Xi Weiya, Myint Htoo Win and Sakthiyavan.

Finally, I would like to express my appreciation to my dear wife, Jayna for her encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURES

Symbol	Description
x	Incremental counter value
x_{ref}	Desired counter value
e	Counter error
K_p	Proportional gain
K_i	Integral gain
K_d	Differential gain
K_0, K_1, K_2	Control Constants
T	Period
V	Control Signal
m_{00}, m_{01}, m_{10}	Image Moments
P_r	Pixel value of red band of image
P_g	Pixel value of green band of image
P_b	Pixel value of blue band of image
i_c	X centroid value of child image
j_c	Y centroid value of child image

SUMMARY

Conventional industrial robots utilize close loop kinematics chain for position control. Although the end-effector of the robots can be positioned very accurately, the work pieces which the robots manipulate must also be positioned accurately with respect to the robot coordinate frame. This presents a problem when the work pieces are not positioned accurately or are shifted during operation. Wear over years of robot operation may also introduce some variation in the kinematics of the robot causing inaccurate positioning of the end-effector. Feedback control of the robot has so far been focused mainly on joint positions. In this dissertation, visual feedback control of a robot, better known as visual servoing, is explored.

In a visual servoing system, the pose of the robot end-effector with respect to the object of interest is determined using visual information from a camera. The camera can be a stationary one or fixed to the end-effector. The classification of visual feedback robotic systems falls into four basic structures, namely the Dynamic Positioned-based Look-and-Moved structure, the Dynamic Image-based Look-and-Moved structure, the Position-based Visual Servo structure and the Image-based Visual Servo structure.

Two visual tracking systems are developed. They are based on the Dynamic Position-based Look-and-Move structure and the Image-based Visual Servo structure. The system setup comprised of a custom-designed 2-axis pan-tilt robot, a color CCD camera, a frame grabber card and the host PC with real-time software installed. The

systems are to track a stationary or moving object using its pan-tilt axes, keeping the object of interest in its view at all times.

The Dynamic Position-based Look-and-Move system developed has a slower tracking speed than that of the Image-based Visual servo system. Its motion is not smooth when tracking a moving object. It is more suited for tracking stationary object. The Image-based Visual Servo system developed is capable of tracking a moving object at an angular speed of 38.16 degrees per second for pan and 66.24 degrees per second for tilt. This higher performance allows it to track a moving object reasonable well with smooth robotic movement.

Chapter 1 Introduction

This chapter provides the reader with some background information regarding the difficulties faced by present industrial robots and how the use of various sensor feedbacks may enhance the performance of these robots. It also discusses the previous work done by other researchers in the field of visual servoing and defines the objective and scope of the project. The last section of this chapter presents the outline of the dissertation.

1.1 History of Visual Servoing

The history of visual servoing goes back to the seventies. The term visual servoing is coin by J. Hill et W. T. Park [1] in 1979. At the time, the concept of robot began to be commonly accepted as the one of a machine liable to interact with its environment with some autonomy. Among the sensors available at the time, camera has fascinated researchers due to the large amount of information which could be extracted from the images. Giving visual information to a robot also brought its skill nearer to the level of human.

Owing to the amount of information provided by cameras, the initial ambition in robotics was generally to understand the world, in order to decide, through symbolic approaches, how to act. The idea of using cameras inside a control loop only came later. It is achieved by selecting specific "signals" from an image and using them as a sensor feedback associated with a dynamical system.

These two approaches grew separately from the eighties. The first was linked to the computer vision domain, mainly led by computer scientists and applied mathematicians. The latter gave rise to the visual servoing approach, investigated by people from the automatic control field.

1.2 Visual Servoing System

A visual servoing system is a feedback control system based on visual information. A visual servoing system is essential for autonomous robots working in unknown environments. A visual servoing system is composed of two processes, namely environment recognition and motion control. The feature-based approach in visual servoing provides a theoretical background for robustness against modeling errors.

Visual servoing is a rapidly maturing approach to the control of robot manipulators that is based on visual perception of robot and workpiece location. Visual servoing involves the use of one or more cameras and a computer vision system to control the position of the robot's end-effector relative to the workpiece as required by the task. It is a multi-disciplinary research area spanning computer vision, robotics, kinematics, dynamics, control and real-time systems.

The rapid growth in computing power and the falling cost of cameras and frame grabbers have created a growing interest in the application of visual servo system in the industry.

1.3 Previous Work

Research in computer vision has traditionally emphasized the paradigm of image understanding. However, some work has been reported towards the use of vision information for tracking [2], [8], [15], [18], [20], [21], [22]. In addition, some research [12], [23], [24] has also been conducted on the use of vision information in the dynamic feedback loop.

Hunt and Sanderson [18] presented algorithms for the visual tracking based on mathematical prediction of the position of the object's centroid. Their algorithm needed the computation of the co-ordinates of the centroid and could only track slow moving objects. Wesis *et al.* [23] proposed solutions to the problem of robotic visual tracking under the framework of model reference adaptive control. The proposed framework has been verified with a number of simulated examples.

Lee and Wohn [20] used image differencing technique to track the object. Luo *et al.* [21] presented a robot conveyer tracking system that incorporates a combination of visual and acoustic sensing. They used modified version of the Horn-Schunk [17] algorithm for the calculation of the optical flow at every pixel. This algorithm accomplished 1-D visual tracking and assumed knowledge of the conveyer's speed.

Goldenberg *et at.* [15] used the PIPE real-time image processing machine to do visual tracking. Their method used temporal filtering. Corke and Paul [9] used a feature-based method to drive the robotic device.

Allen et al. [2], [4] presented a method for real-time tracking that was based on the idea of spatio-temporal filtering [16]. In addition, Allen *et al.* [4] used the visual information provided by a pair of stationary cameras in order to track a moving object with a robot. Dickmanns and Zapp [10], [11] presented several methods (Kalman filters) for the integration of vision information in the feedback loop of various mechanical systems such as satellites and cars.

Tsakiris [22] proposed strategies for visual tracking that are based on the computation of the optical flow and the computation of the objects moments. Feddema *at al.* [12], [13], [14] emphasized the control aspect of the robotic visual tracking problem. Koivo and Houshangi [19] used adaptive control techniques in conjunction with the information provided by a stationary camera in order to control the robotic device.

1.4 Project Objective

The objective of the project is to develop a 2-DOF robotic visual servoing system based on the dynamic position-based look-and-move structure and the image-based visual servo structure.

1.5 Project Scope

The project scope includes the design and fabrication of the 2-DOF pan-tilt robot, the development of the computer vision feedback system, the development of the control

and real-time system and the final integration of the systems. System characteristic and performance of the two visual servoing systems are also studied.

1.6 Dissertation Outline

In Chapter 2 of the dissertation, the concept of vision-based robotic positioning system is presented. The various visual tracking techniques are also briefly discussed. Chapter 3 touches on the design and specification of the pan-tilt robot. The software design to control the robot is also presented. The vision system and the software design of the visual feedback are discussed in Chapter 4 of the dissertation. Chapter 5 presents the software consideration in the integration of the vision module and the robot control module. The system characteristic and performance of the two visual servoing structures are also discussed. Finally, Chapter 6 concludes the dissertation and list out some recommendations for future work in relevant areas.

Chapter 2 Concept of Visual Servoing

The role involved in a vision-based robotic positioning system is the control of the pose of the end-effector using visual information extracted from the image captured by the camera. Section 2.1 of this chapter explains the term "feature" and how it is being extracted. The relative pose is explained in Section 2.2 and the various visual tracking techniques are discussed in Section 2.3.

2.1 Feature and Feature Extraction

A feature is defined as any measurable relationship in an image [6]. Examples of features are holes, edges, corners, outlines or even the entire face area [5]. Three or more features are required to determine the full pose of an object, although generally four are required for a unique solution [3].

Features are extracted from the image by suitably designed algorithms. Depending on the selection of feature, the feature extraction algorithms may detect the centroid, area, perimeter or signature of the feature. Four typical steps involved in feature extraction are:

- Image grabbing
- Image thresholding
- Image cleaning
- Feature detection

2.2 Relative Pose Determination

From the extracted features, the relative pose (position and orientation) of the object can be computed. The relative pose of the object may be computed in the image plane and later transformed in to the world frame using a suitable transformation matrix. Pose determination is particularly important for position-based structures.

2.3 Visual Servoing Techniques

Sanderson and Weiss [7] introduced an important classification of visual servoing techniques in the eighties. The classification is represented in Figures 2.1, 2.2, 2.3, 2.4.

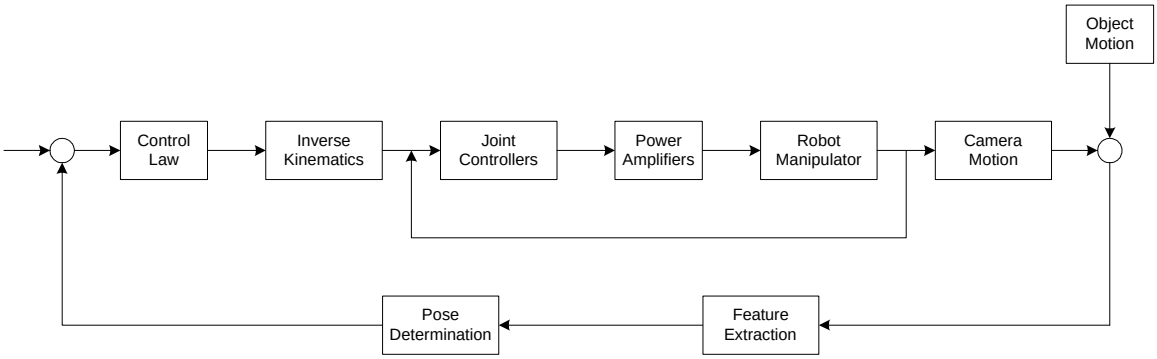

Figure 2.1: Dynamic Position-Based Look-and-Move Structure

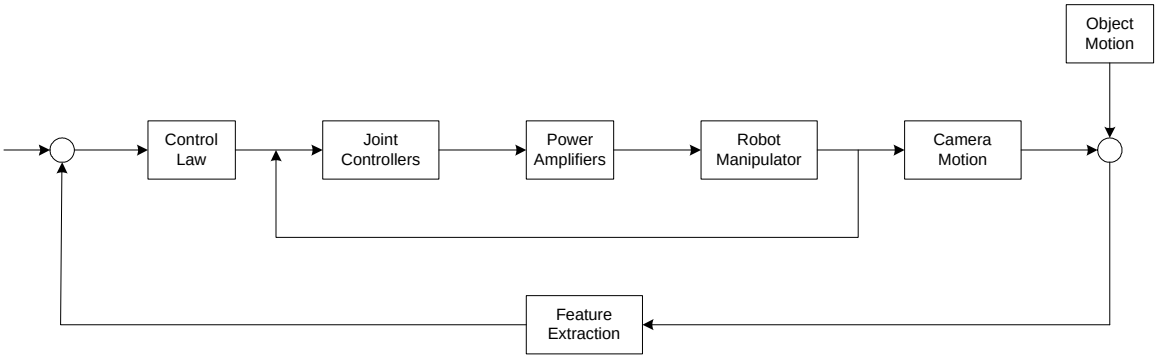

Figure 2.2: Dynamic Image-Based Look-and-Move Structure

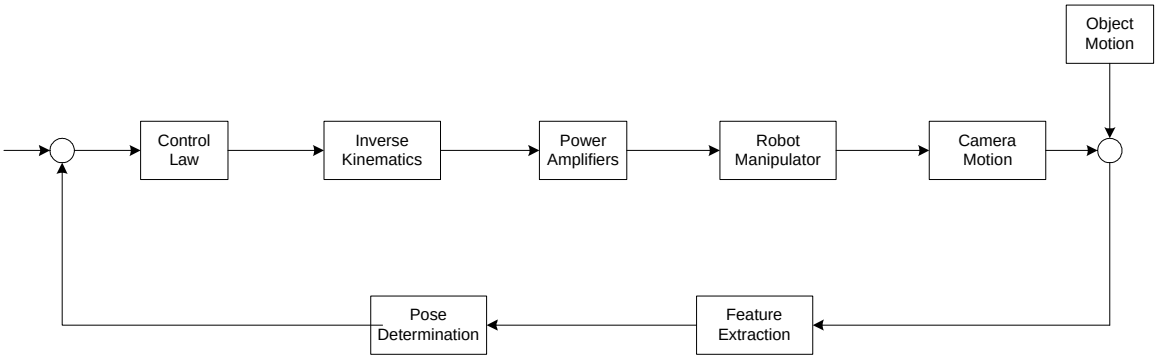

Figure 2.3: Position-Based Visual Servo (PBVS) Structure

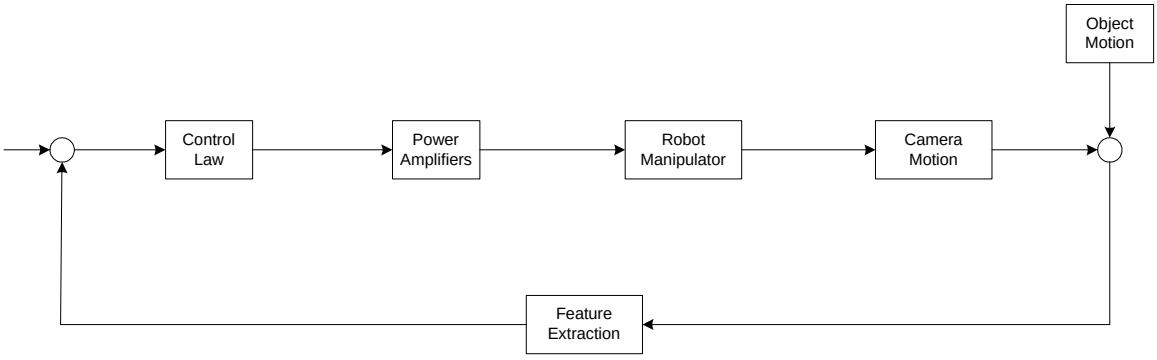

Figure 2.4: Image-Based Visual Servo (IBVS) Structure

In the Position-Based Visual Servo (PBVS) Structure, features are extracted from the image and used in conjunction with a geometric model of the object to determine the pose of the object with respect to the camera.

In the Image-Based Visual Servo (IBVS) Structure, the last step, i.e., the determination of pose is eliminated and robot control is done based on the image features directly. The IBVS structure reduces computational delay, eliminates the need for image interpretation and eradicates errors in sensor modeling and camera calibration when compared with the PBVS structure. However, the process is highly coupled and thus non-linear.

In the Look-and Move Structure, joint feedback is used. This additional loop will ensure the accurate positioning if the robot. However, with this additional loop, more processing overhead is incurred. As the name imply, the Look-and-Moved Structure requires the robot to come to a complete halt before the next "Look" is executed.

It is good to take note that the term "Visual Servoing" has been loosely used over the years. It is now used generally to describe robot control that utilizes visual feedback instead of the PBVS and IBVS techniques mentioned above.

Chapter 3 The Robot Control System

This Chapter presents to the robot control system. Instead of using commercial robots, a 2-DOF pan-tilt robot is designed and built.

3.1 The Pan-Tilt Robot

The main design consideration was to have it modular and at the same time compact. A single axis rotary table was first designed. Thereafter, a U-shaped structure was designed to mount a similar rotary table perpendicular to the first one. With the 2 axes of rotation orthogonal to one another, a pan-tilt motion similar to that of a human head can be achieved.

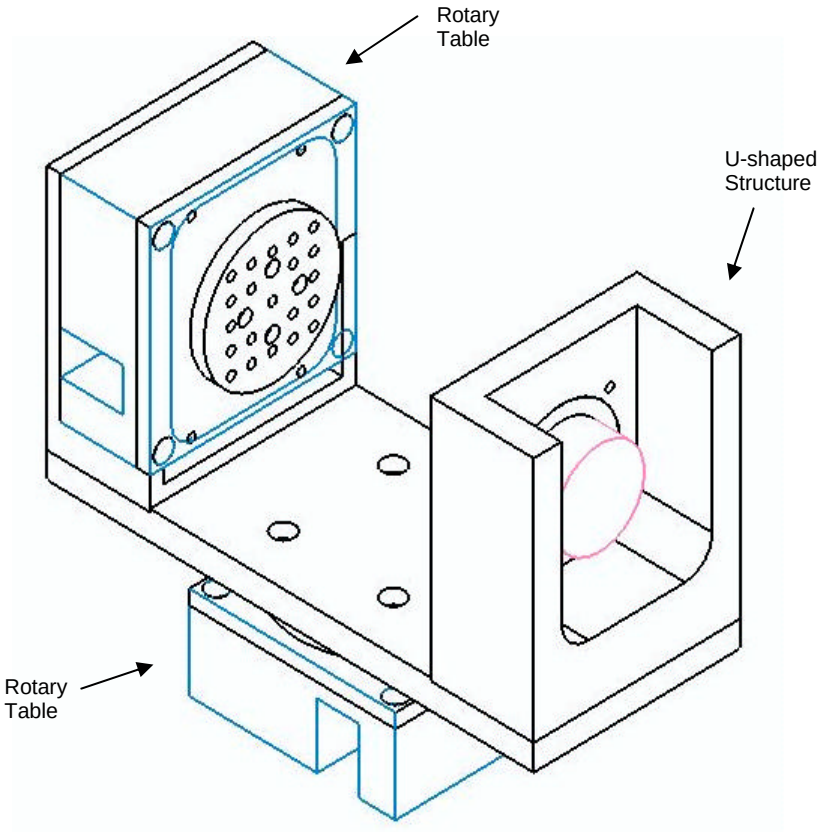

Figure 3.1: Pan-Tilt Robot

3.2 The Motors and Drivers

Panasonic MINAS A-Series AC Servo Motors and Drivers have been selected to drive the pan-tilt robot. The A-Series Motors provide a maximum speed of 5,000 rpm and an incremental type encoder with 2500 pulses per revolution resolution. The matching A Series Drivers may be controlled via Analog Voltage Signal (speed or torque), or two digital inputs (Step and Direction). The Drivers also feature the ability to switch between these control signals automatically. The specification of the two Motors and Drivers selected are shown in Table 3.1.

	Axis 0 (Pan)	**Axis 1 (Tilt)**
Motor Type	**MSMA042A**	**MSMA022A**
Voltage	200V	200V
Output Rating	400W	200W
Revolution Rating	3000r/min	3000r/min
Encoder Type	Incremental 2500P/r, 11 Wires	Incremental 2500P/r, 11 Wires
Driver Type	**MSDA043A1A**	**MSDA023A1A**
Input Voltage	200-230V	200-230V
Output Voltage	106V	92V
Input F.L.C.	1.8A	1.1A
Output F.L.C.	2.5A	1.6A
Power	400W	200W

Table 3.1: Specification of Motors and Drivers

3.3 Motion Control Board

The Servotogo (STG) Model II PC Card is selected for the control of the motors due to its low cost and flexibility. It is a basic AD/DA card with no on board controller. Most other PC-based Controller card has precompiled libraries which are not open-source. This makes the programming of the control algorithm less flexible. The specification of the STG Model II is shown in Table 3.2.

Encoder Input	• Up to 8 channels of encoder input • A, B, and Index signal inputs • 24 Bit counters, extendible to 32 bits in software • Single-ended or differential (RS422 compatible) input signals • The index pulse can be configured for normally high or normally low input
Analog Output	• Up to 8 channels of analog output • +10 Volt to -10 Volt span @ 28ma • 13 bit resolution
Digital Input and Output	• 32 bit Opto-22 compatible, configurable in various input and output combinations
Analog Input	• 8 Channels of analog input • 13 bit resolution • Configurable as +/- 10V or +/- 5V
Interval Timers	• Timer interval is programmable to 10 minutes in 25 microsecond increments • Capable of interrupting the PC
Battery Backup	• Input Used to maintain encoder counting capability in event of a power failure so that re-initialization is not required when power is removed.
Board Address Detection with IRQ Software Selectable	• Used to determine the board base address automatically without user interaction. • IRQ number is software selectable - no board jumper required
Watchdog Timer	• Can be used to initiate servo shut-down in the case of a catastrophic computer malfunction

Table 3.2: Specification of STG Motion Control Board

Basic algorithms have been written to access the ADs, DAs, I/Os and counters. These functions are compiled into library files (.lib) for ease of use when developing the control algorithm.

3.4 Real-Time System

Microsoft Windows operating systems like Windows NT, Windows 2000 and Windows XP are designed for the masses as home and office operating systems. The success of Windows operating system prompted uses that extend beyond its initial purposes. Developers of real-time applications desire the benefits that come from using Windows, but require a focused real-time development and execution environment in keeping with traditional Real-Time Operating System's (RTOS's) performance and determinism.

RTX enables Windows NT, Windows 2000 and Windows XP to address these needs. It allows real-time and non-real-time processing within the same computer, enables Windows to handle control oriented applications, leverages industry standard development and debugging tools.

RTX has a host of precompiled functions which allow users to read and write directly to a specific address in the computer. This enables the direct access of all the functionality of the STG card by simply reading from and writing to the appropriate STG addresses.

3.4.1 PID Control

A software PID Controller is implemented for the position control of the 2 axes. The position of the motor is read from the incremental counter and compared with the desired position. The error in the position is fed into the PID Controller. This PID loop runs at a frequency of 1 KHz. The Control Block Diagram is shown in Figure 3.2.

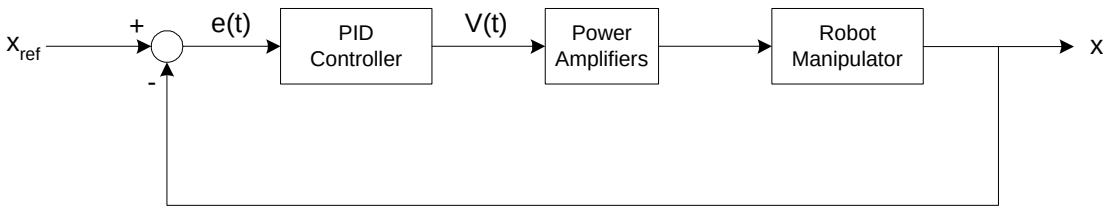

Figure 3.2: Control Block Diagram

The PID controller is implemented in C codes using the equations below.

$$V(t) = V(t-1) + K_0 e(t) + K_1 e(t-1) + K_2 e(t-2)$$

$$K_0 = K_p + K_i T + K_d / T$$

$$K_1 = -K_p - 2K_d / T$$

$$K_2 = K_d / T$$

$$e = x_{ref} - x$$

where T = sampling period

The PID values are tuned by observing the response of the motor on an oscilloscope when given a square wave input. PID values of 1.8, 1.8 and 0.0001 provide reasonable response.

Figures 3.3 and 3.4 are plots of the x and x_{ref} against time for duration of 500 milliseconds for the pan and tilt axis respectively.

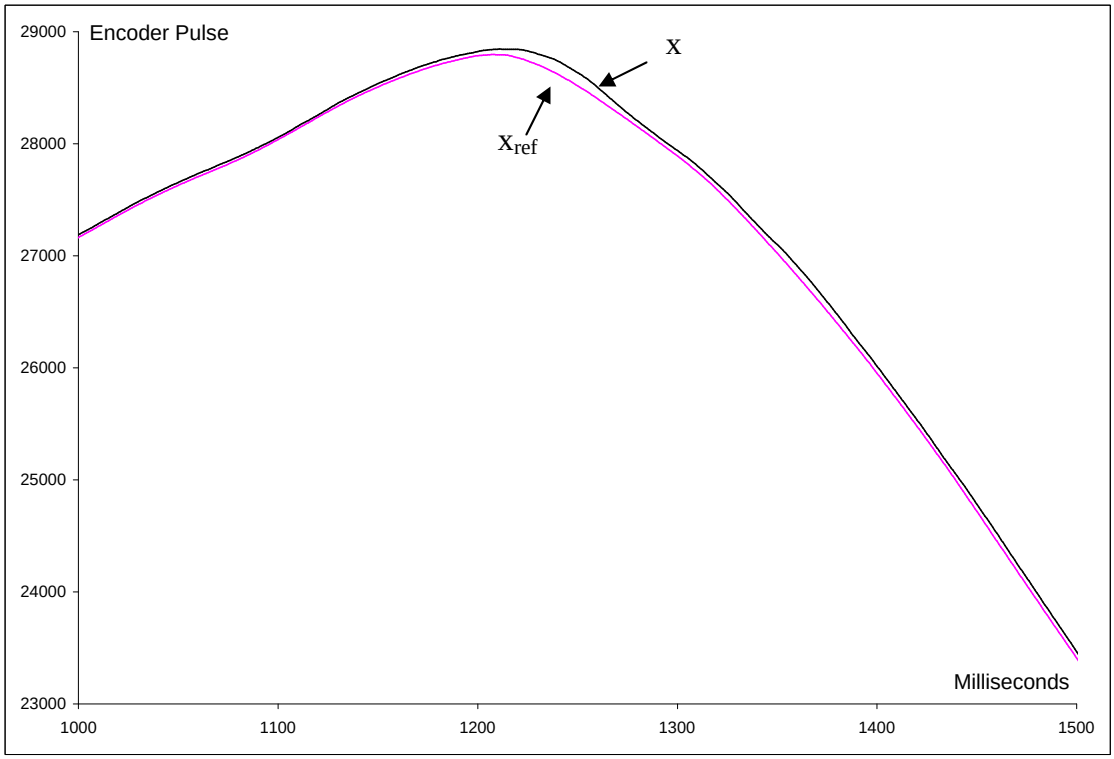

Figure 3.3 Plot of x and xref(encoder pulse) against time(ms) for axis 0 (pan)

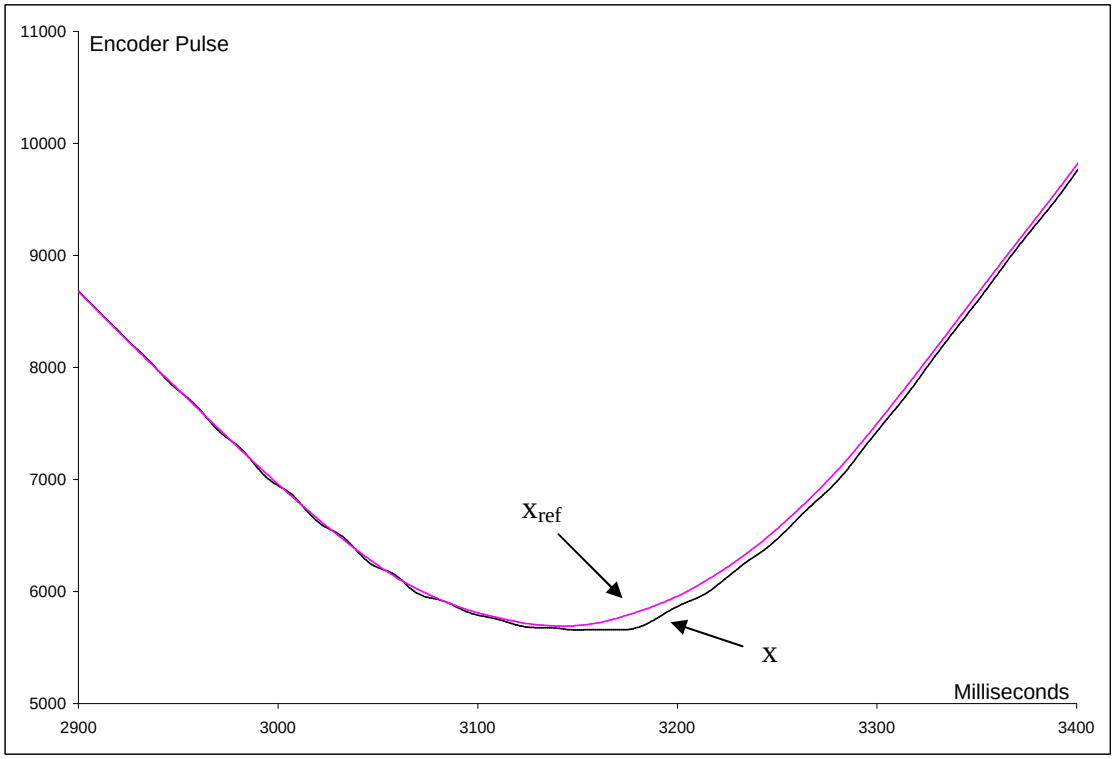

Figure 3.4: Plot of x and xref(encoder pulse) against time(ms) for axis 1 (tilt)

Figure 3.5 shows a sample C code of the real-time PID control loop that is runs at a frequency of 1 KHz.

```
// compute constant for PID
for(axis = 0; axis < NAXIS; axis++)
{
  data->K1[axis] = Kp[axis] + Ki[axis]*data->T + Kd[axis]/data->T;
  data->K2[axis] = -Kp[axis] - 2.0*Kd[axis]/data->T;
  data->K3[axis] = Kd[axis]/data->T;
}
while(1) // real-time loop
{
  for(axis = 0; axis < NAXIS; axis++)
  {
    data->X[axis] = RltReadCntr(axis);       // read counter
    XErrN[axis] = data->XRef[axis] - (double)data->X[axis];
    data->control[axis] = data->controlOld[axis]
                        + data->K1[axis] * XErrN[axis]
                        + data->K2[axis] * data->XErrN1[axis]
                        + data->K3[axis] * data->XErrN2[axis];

    DACout = (USHORT)(-data->control[axis]) + 0x1000;
    addpus=(PUSHORT)(STG_A + DAC0 + 2 * axis);
    RtWritePortUshort(addpus, DACout);       // writes to port

    data->controlOld[axis] = data->control[axis];
    data->XErrN2[axis] = data->XErrN1[axis];
    data->XErrN1[axis] = XErrN[axis];
  }
}
```

Figure 3.5: C Source of Real-Time PID Control Loop

The flowchart of the real-time robot control module is shown at Figure 3.6.

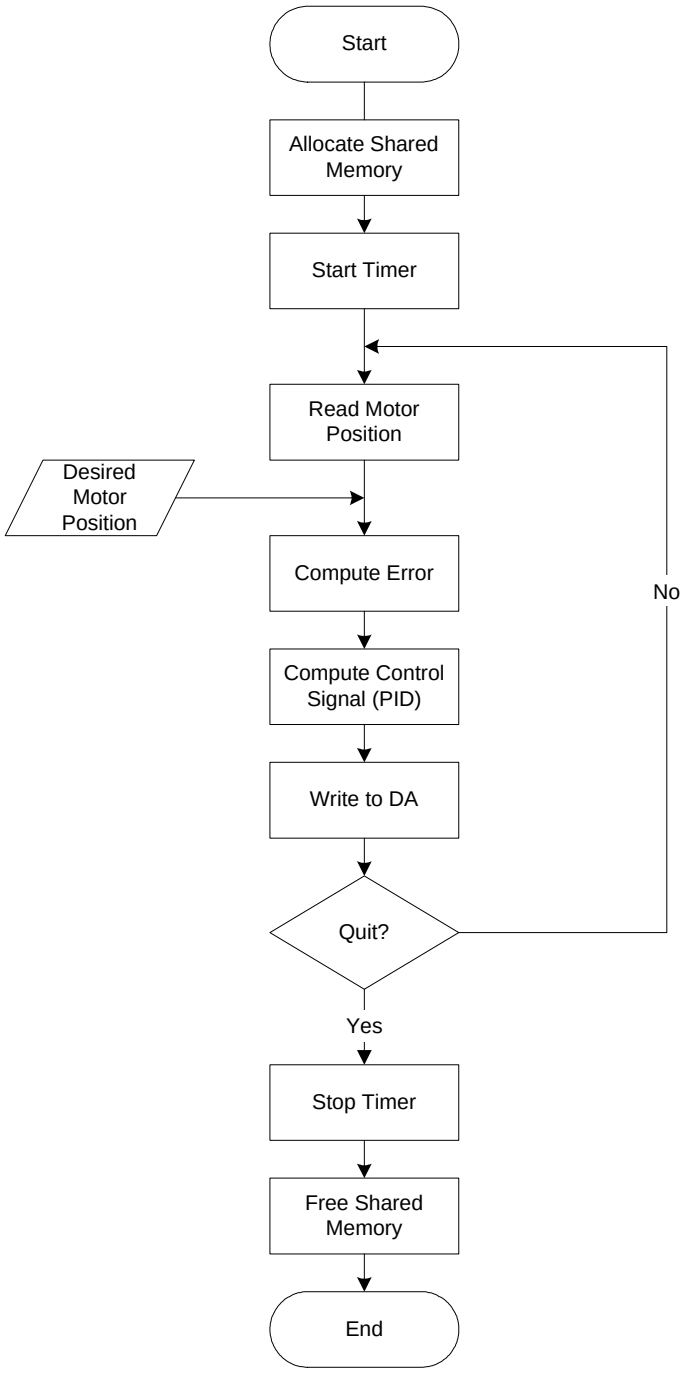

Figure 3.6: Flowchart of Real-time Control Module

Chapter 4 The Vision System

The vision system plays an important role in the visual servoing system. Section 4.1 outlines the camera technology, video standards and the camera used in this project. Section 4.2 presents the frame grabber card and the software used for the development.

4.1 Camera Technology

Early computer vision works are based on vidicon and thermionic tube image sensors. These devices are large and heavy, lack robustness and suffer from poor image stability and quality. In the mid eighties, most researchers switched to more robust cameras with solid state sensors.

4.1.1 Sensors

An interline transfer CCD (Charged Coupled Device) camera consists of rectangular array of photodiodes. Each photodiode accumulates a charge related to the time integrated incident illumination over it.

Incident photons generate electron-hole pairs in the semiconductor material and one of these charges, generally the electron, is captured by an electric field in a charge well. Charged packets are moved using multi-phase clock voltages applied to various gate

electrodes. Such a transport mechanism shifts the charge from the photodiode to the output amplifier.

Data (charge) is transferred out simultaneously by frames directly from the image sensors to their corresponding sensors registers. The output from the camera is always one frame behind the image being captured.

4.1.2 Video Standards and Interlaced Scanning

Broadcast and closed circuit televion industries dominate the manufacturing and consumption of video equipment. Thus most cameras used for machine vision work conform to television standards. The 4 most widely used standards are:

- RS170, scanning 525 lines per second, monochrome
- CCIR, scanning 625 lines per second, monochrome
- NTSC, scanning 525 lines per second, color
- PAL, scanning 625 lines per second, color

Interlaced Scanning is a scanning process in which all odd and then all even horizontal lines are alternately scanned. Adjacent lines belong to different fields. Two fields constitute a frame. Hence, a field is scanned at 60 Hz for RS170 and NTSC format, and 50 Hz for the CCIR and PAL format.

4.1.3 JAI MCL-1500 Color CCD Camera

Most vision applications have been developed in the grayscale domain. Due to the fall in cost of color CCD cameras and digitizers, and increasing PC processing power, it has become more viable to explore into the color domain. The depth of information extractable from environment becomes immense.

The MCL-1500 from JAI is a 1/3" CCD Camera with built-in 10-times motorized zoom lens. It also features an auto focus lens with auto iris and white balance. The most useful feature of this particular camera is the ability to manually control the zoom, focus, iris and white balance via the RS232 interface. The specification of the camera is shown in Table 4.1.

TV Standard	PAL
Pick-up device	1/3" Interline-transfer CCD
Effective pixels	752(H) x 582(V) 440,000 pixels
Scanning system	Horizontal:625 lines at 2:1 interlace Vertical: 450 lines
Scanning frequency	Horizontal: 15.625KHz Vertical: 50Hz
Sync. system	Internal/External by VS or Sync.
Sensitivity	1000 lux F4 (3200K)
Min. illumination on CCD sensor	0.05 lux F1.4 (AGC off), -50% Video level
S/N ratio	More than 46dB (AGC off)
Video Output	VBS 1.0Vp-p 75? , S-Video (Y/C)
White Balance	Auto, 4600K, Manual (2600 ~ 9000K)
AGC	ON/OFF
Back Light compensation	ON/OFF
Lens focal distance	F5.8 ~ 58mm
Min. object distance	Wide: 0.3m Tele: 1.0m
Lens Iris	Auto

Zoom Control	Remote/Manual
Focus Control	Auto/Remote/Manual
Serial Interface	RS-232C
Power Source	+12VDC±10%
Operating temperature	0°C ~ 40°C
Dimension (W x H x D)	55 x 60 x 95 mm
Weight	350g

Table 4.1: Specification of MCL-1500 JAI Color CCD Camera

Video Signal from the camera is sent to the frame grabber card through a coaxial RCA cable.

4.2 Machine Vision

Various machine vision techniques are used in the vision system of the project. This section discusses the digitization, the storage and the processing of the image. The hardware used for digitization and the software for the image processing are presented.

4.2.1 Image Digitization and Aspect Ratio

The digital image used in machine vision is a spatially sampled representation of the continuous image function. The first step in machine vision is to digitize the analog video signal that represents the image function. The signal is sampled, quantified and

stored in a two dimensional array in the memory. These samples are referred to as the gray-level values in a monochrome image and RGB values in a color image.

When acquiring an image of an object, each pixel depicts some real distance in width and height. Ideally, this distance is the same in both width and height, producing square pixels. However, after digitization, it is quite common for a pixel to show different length in each direction. The ratio of the pixel's width to its height is called the aspect ratio. A square in the real world will appear square in the image captured if the aspect ratio is 1.

4.2.2 Matrox Meteor II Frame Grabber Card

Matrox Meteor II is a standard monochrome and color analog frame grabber. This board is in a PCI form factor. It can acquire different types of standard video formats using its video decoder. The video decoder can accept composite (CVBS) and component (Y/C) video in NTSC/PAL formats, and convert it to RGB 8:8:8, YUV 4:2:2 or YUV 4:1:1, with either square pixels or CCIR-601 resolutions. It can also convert RS-170/CCIR video formats with square pixels or CCIR-601 resolutions. Matrox Meteor-II /Standard accepts an external trigger input, and can operate in next valid frame/field mode.

The Matrox Meteor II board allows the transfer of live video to Host memory or off-board display memory. To prevent loss of data during long bus-access latencies found in heavily loaded computer systems, the Matrox Meteor II board features 4 megabytes

of video transfer memory for temporary frame storage. The board is also equipped with the Matrox Video Interface ASIC (VIA), which acts as a video-to-PCI bridge.

4.2.3 Matrox Imaging Library

Matrox Imaging Library MIL-Lite is a development library which provides an extensive list of functions used to capture, process, analyze, transfer, display, and archive images. Processing and analysis operations include: spatial filtering operations, morphological operations, measurements, blob analysis, optical character recognition (OCR), pattern matching, matrix/bar code reading, and calibration.

4.2.4 Object Selection

A red painted surface is selected as the object to be tracked. Only the area and centroid of the red surface is of interest to us. The x-y coordinate of the centroid corresponds to the pan-tilt axis of the 2-DOF robot. The area information of the red painted surface can be used to control the zoom level of the CCD camera.

4.2.5 Software Design

Before elaborating on the software design, the coordinates of the image plane is defined in Figure 4.1.

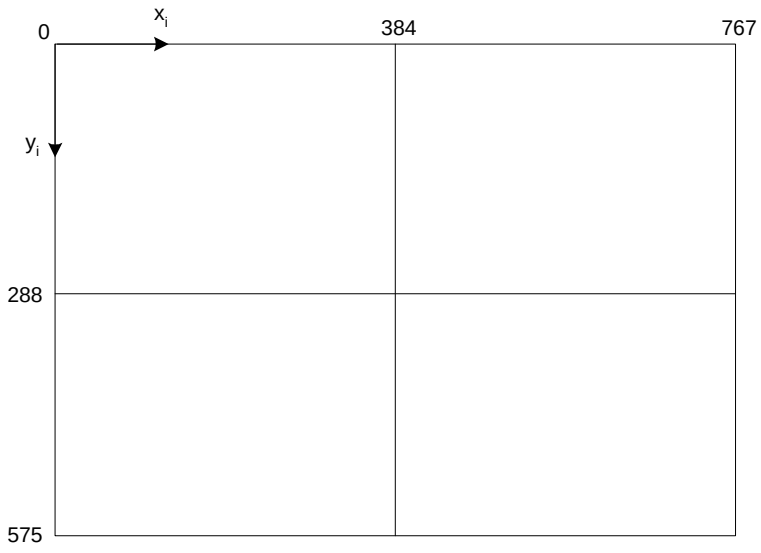

Figure 4.1: Image Plane Coordinate System

Color vision is adopted instead of the traditional monochrome vision. As cost of color camera and color frame grabber has dropped to the level of their monochrome counterparts, it is worthwhile to explore into the realm of color machine vision.

A color image can be represented in the Red/Green/Blue domain or the Hue/Saturation/Luminance domain. Since the raw data acquired from the frame-grabber is in the RGB domain, the software development is done in this domain.

Typical method to extract a red feature will use only the red band. A threshold value is set and which ever pixel of a value higher than the threshold value is considered red. The method of extracting a red feature will contain much “pepper noise” cause by very white areas. This method of red feature extraction will also be very sensitive to ambient lighting condition.

To solve this problem, we use a thresholding method that will discount the color of illumination and its intensity. This property of compensating for illumination is called color constancy. To achieve, color constancy in the feature, the ratios among the 3 color bands are used instead. These ratios are also known as chromaticity ratios. Many chromaticity ratios were tested. It was found out that the ratio between the red band and green band can define a red feature well. For the lighting condition of the setup, a threshold value of 1.2 for this ratio is able to extract the red feature. Further experiments on other ratios show that if another criterion using the blue band to red band ratio is added, better result is achieved. Hence we define a pixel to be red when,

$$\frac{P_r}{P_g} \geq 1.2 \text{ and } \frac{P_b}{P_r} \leq 0.9$$

The threshold values of 1.2 and 0.9 needed to be varied slightly to achieve optimal result under different lighting conditions. A separate program has been written to assist in the fine tuning of these values.

After thresholding, pixels containing the red feature are identified. The centroid of the red feature is determined by computing the moments. A sample calculation of the moments and centroid are illustrated based on a smaller child image that has already been threshold for red. The sample child image is illustrated in Figure 4.2.

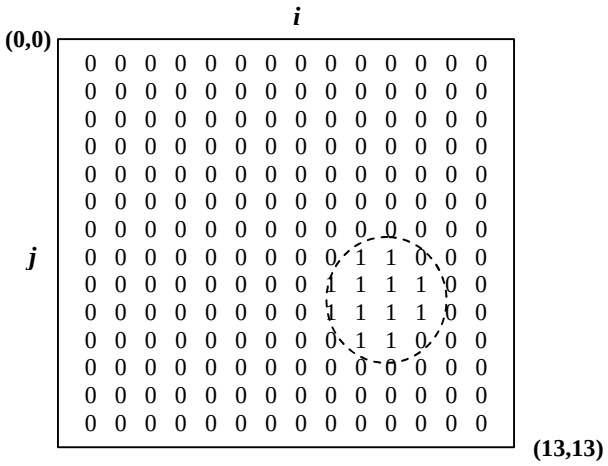

Figure 4.2: Sample Image after Thresholding

Calculation based on the sample child image in Figure 4.2 is demonstrated below.

$$m_{00} = \sum_{i,} \sum_{j \in R} 1 = 12$$

$$m_{10} = \sum_{i \in R} i = (8 \times 2) + (9 \times 4) + (10 \times 4) + (11 \times 2) = 114$$

$$m_{01} = \sum_{j \in R} j = (7 \times 2) + (8 \times 4) + (9 \times 4) + (10 \times 2) = 102$$

$$i_c = \frac{m_{10}}{m_{00}} = \frac{114}{12} = 9.5$$

$$j_c = \frac{m_{01}}{m_{00}} = \frac{102}{12} = 8.5$$

Hence, the centroid of the red feature is (9.5,8.5).

Thresholding the whole image in visual servoing application is sometimes unnecessary as the feature of interest in this case is always near the center of the image plane. In order to increase the sampling rate of the vision module, a smaller child image is

defined within the original image. Only the child image is being threshold and computed for centroid. This reduces the amount of data to be computed significantly.

The sampling rate can be further increased by processing 1 line for every 2 lines of data or 1 line for every 3 lines of data. It is observed that the centroid of the red feature is not affected even though fewer lines are processed. For a child image of 256 by 384, the sampling rate is about 28Hz when every line is processed. When every other line and every other 3 lines are processed, the sampling rate increases to about 86Hz and 139Hz respectively.

However, due to the PAL camera that is being used, the image buffer is only updated at a rate of 25 frames per second. Hence, as long as the computation of the centroid is faster than the camera refresh rate, no image information is lost.

The C code of the sampling loop of the vision module is shown in Figure 4.3. This loop includes the thresholding and centroid computation algorithm. Each color band of the image buffer is copied to an array using the MIL function, 'MbufGet', for easy data manipulation.

```
while(key!='x')
{
  if(kbhit())
    key=getch();
  MbufGet(MilRedBand, RedTemp);
  MbufGet(MilGreBand, GreTemp);
  MbufGet(MilBluBand, BluTemp);

  m00=0,
  m01=0,
  m10=0;

  for(j=0;j<CHeight;j=j+3)
  {
    for(i=0;i<CWidth;i=i+3)
    {
      Red = (double)RedTemp[CWidth*j+i]/(double)GreTemp[CWidth*j+i];
      Blu = (double)BluTemp[CWidth*j+i]/(double)RedTemp[CWidth*j+i];
      if(Red>=RED_THRESHOLD && Blu<=BLU_THRESHOLD)
      {
        m00++;
        m10 += j;
        m01 += i;
      }
    }
  }

  X = (double)m01/(double)m00;
  Y = (double)m10/(double)m00;

  PixErrN[0] = X + CHILD_START_X - IMAGE_WIDTH/2.0;
  PixErrN[1] = Y + CHILD_START_Y - IMAGE_HEIGHT/2.0;

}
```

Figure 4.3: C Code of Centroid Computation Loop

4.2.6 Second Order Low-Pass Filter

A second order low-pass filter is implemented to filter off high frequency variation of the centroid computed. This sudden change in centroid value can be caused by image noise or sudden acceleration of target. When the low-pass filter runs at a frequency higher than that of the camera frame rate, it also helps to smoothen the trajectory by interpolating the centroid values in between image frames.

Instead of using the centroid computed for the current sample, a weighted average of the current centroid and centroid of the previous two samples are used. The weights need to add up to 1. Weights of 0.4, 0.3, 0.3 respectively works well to smoothen the shift in centroid.

$$centroid = W_1 \times centroid(t) + W_2 \times centroid(t-1) + W_3 \times centroid(t-2)$$

Centroid values for a period of 1000 millisecond is collected and plotted in Figure 4.4 and Figure 4.5. The darker line is the centroid value after passing through the low-pass filter. It can be observed that the low-pass filter interpolates and smoothens the original line to a certain extend.

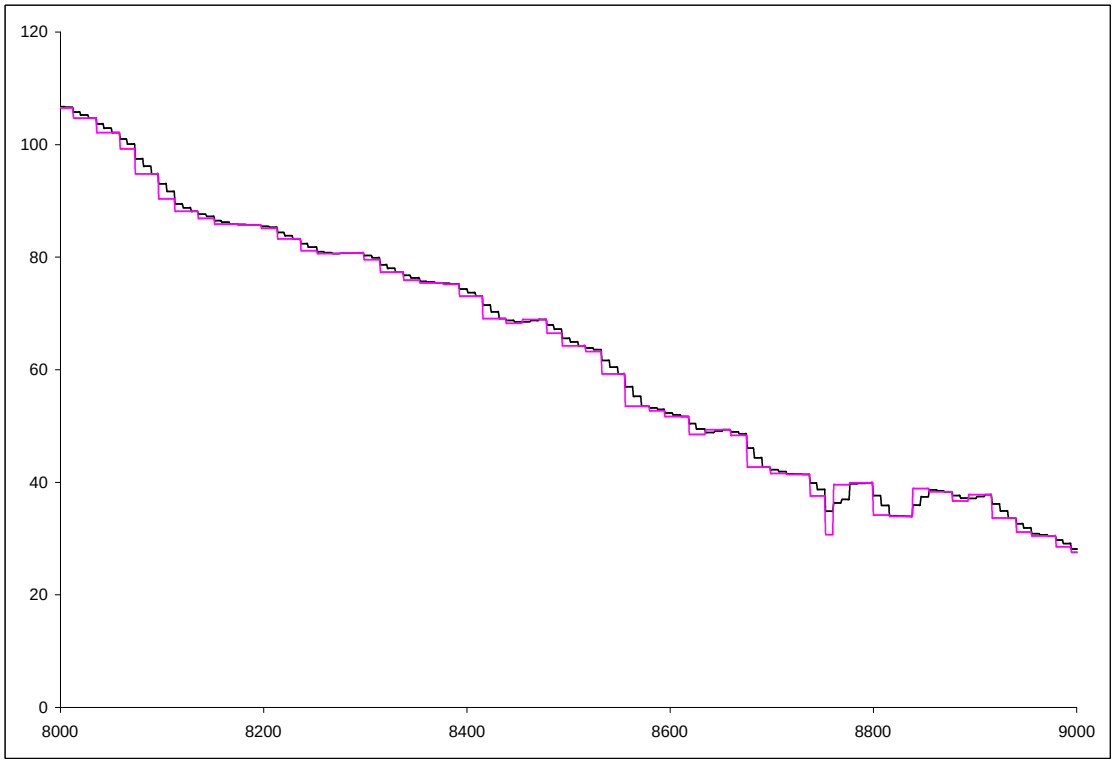

Figure 4.4: Plot of X Centroid Value with and without Low-Pass Filter against Time

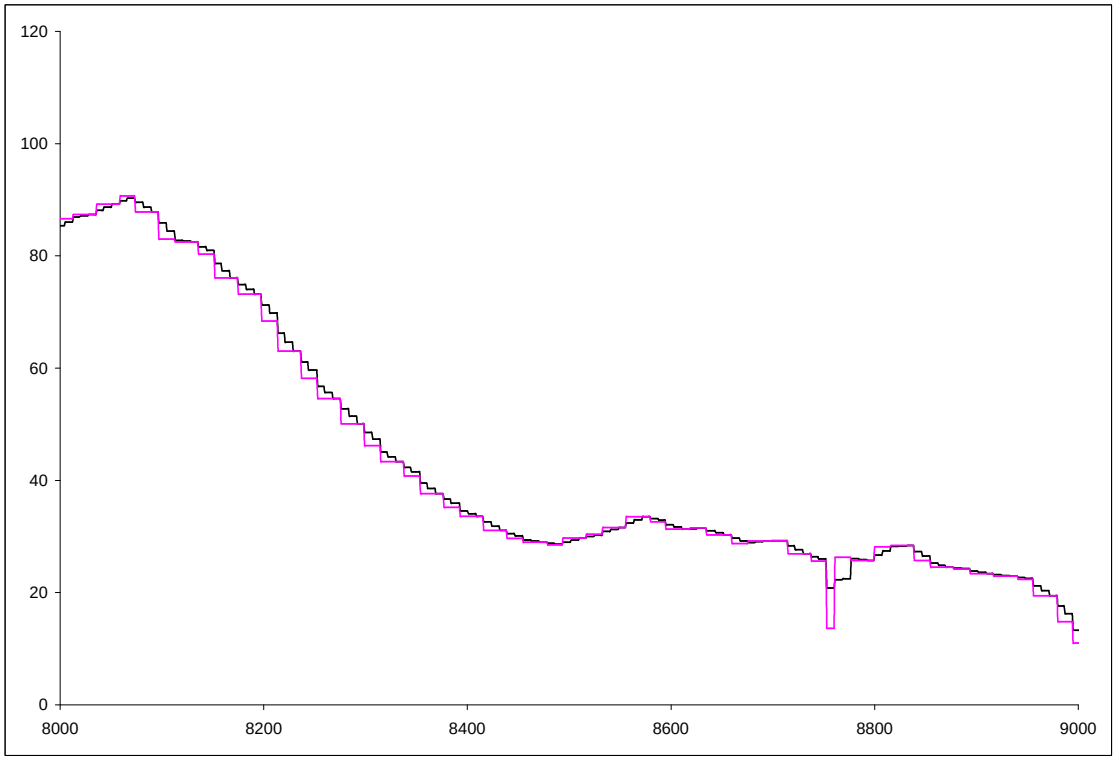

Figure 4.5: Plot of Y Centroid Value with and without Low-Pass Filter against Time

The second order low-pass filter is implemented into the vision feedback loop as illustrated in the sample C-code in Figure 4.6.

```
while(key!='x')
{
  if(kbhit())
    key=getch();
  MbufGet(MilRedBand, RedTemp);
  MbufGet(MilGreBand, GreTemp);
  MbufGet(MilBluBand, BluTemp);

  m00=0,
  m01=0,
  m10=0;

  for(j=0;j<CHeight;j=j+3)
  {
    for(i=0;i<CWidth;i=i+3)
    {
      Red = (double)RedTemp[CWidth*j+i]/(double)GreTemp[CWidth*j+i];
      Blu = (double)BluTemp[CWidth*j+i]/(double)RedTemp[CWidth*j+i];
      if(Red>=RED_THRESHOLD && Blu<=BLU_THRESHOLD)
      {
        m00++;
        m10 += j;
        m01 += i;
      }
    }
  }

  X = (double)m01/(double)m00;
  Y = (double)m10/(double)m00;

  PixErrN2[0] = PixErrN1[0];
  PixErrN1[0] = PixErrN[0];
  PixErrN[0] = X + CHILD_START_X - IMAGE_WIDTH/2.0;

  PixErrN2[1] = PixErrN1[1];
  PixErrN1[1] = PixErrN[1];
  PixErrN[1] = Y + CHILD_START_Y - IMAGE_HEIGHT/2.0;

  PixErrX = 0.4*PixErrN[0] + 0.3*PixErrN1[0] + 0.3*PixErrN2[0];
  PixErrY = 0.4*PixErrN[1] + 0.3*PixErrN1[1] + 0.3*PixErrN2[1];
}
```

Figure 4.6: C Code of Centroid Computation Loop with Second Order Filter

The flowchart of the vision module developed is illustrated in the Figure 4.7.

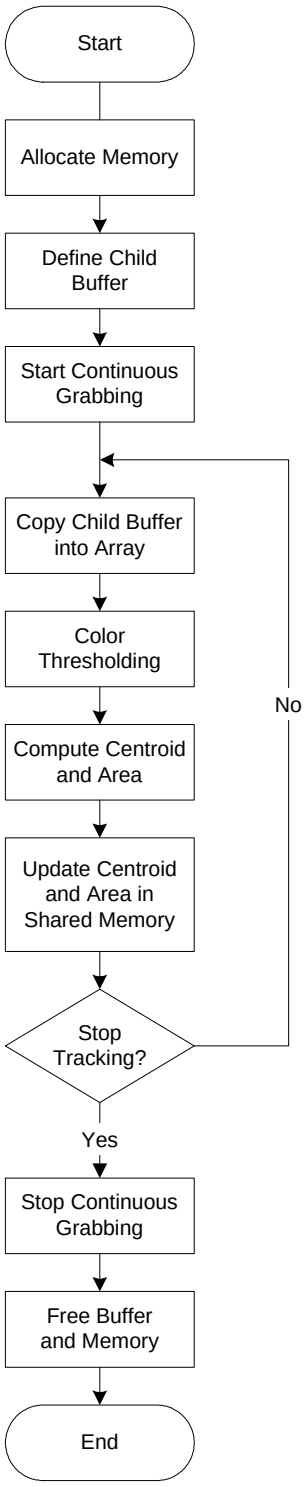

Figure 4.7: Flowchart of the Vision Module

Chapter 5 Visual Servoing

The robot control module controls the position of the robot axes given a reference position to move to. The vision module extracts the feature information and returns a position in the image plane. Visual Servoing is achieved by integrating the two modules into a structure with an inner loop of robot control and an outer loop of vision feedback. Two visual servoing structures are implemented. They are the Dynamic Position-Based Look-and-Move structure and the Image-Based Visual Servo structure. The details of the implementation are discussed in Sections 5.1 and 5.2. Section 5.3 will discuss the system characteristic and performance of each structure.

5.1 Dynamic Position-Based Look-and-Move Structure

5.1.1 Calibration

Calibration involves relating the pixels in the image plane to a length or angle in the world space. For the pan-tilt robot, calibration is done to find the factor relating the pixel and the angle. With this calibration factor, the relative pose of the object can be computed from the feature extracted.

To find this factor, the view angle for the camera is needed. A simple experiment is done to get the view angle. The real world points on the left and right edge of the image plane is marked. A flat wall is used for the real world points for easy marking. Motor 0 is then moved in such a way that the optical center is on the left real world

marker and the right real world marker. The encoder readings of these two positions are noted down.

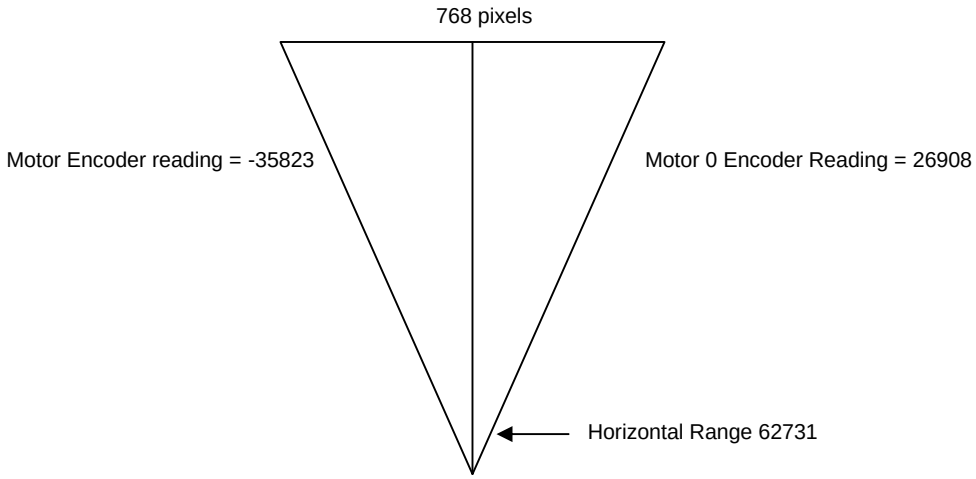

Figure 5.1: Calibration of Pan axis

The same is done for the tilt axis, i.e. motor 1. The calibration data are tabulated below. With a gear ratio of 1:50 and encoder count of 10,000 per motor shaft revolution, the range and degree per pixel are also computed for reference.

	Axis 0	**Axis 1**
Encoder Reading 1	-35823	-18154
Encoder Reading 2	26908	27784
Encoder Range (Angle Range)	62731 (45.17°)	45938 (33.08°)
No. of Pixels	768 (Horizontal)	576 (Vertical)
Encoder Count per pixel	81.68 (0.0588°)	79.75 (0.0574°)

Table 5.1: Calibration Data for Axis 0 and Axis 1

With the calibration factor for each of the axis, we can compute how much to move the motor upon detecting a shift in centroid value of the object detected.

5.1.2 Implementation

As mentioned in Section 3.4.1, a real-time position control using PID has been implemented for the Pan-Tilt Robot. This forms the robot control feedback loop. A visual feedback loop is added outside the robot control feedback loop to give visual feedback of the object to be track.

Shared memory buffer is created using RTX for data exchange between the real-time robot control loop and the non real-time vision feedback loop.

Figure 5.2 is the block diagram of the visual servoing structure that is adopted for the setup.

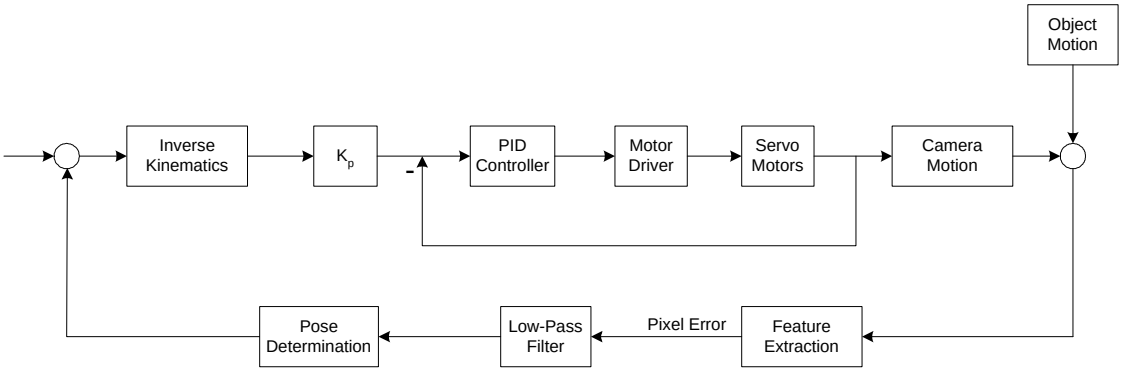

Figure 5.2: Dynamic Position-Based Look-and-Moved Structure

When some pixel error is detected between the camera optical center and the object centroid, this pixel error is multiplied by the calibration factor to get the relative pose of the object with respect to the camera. With this relative pose information, the robot is given a point-to-point move with a trapezoidal trajectory. Ideally, this point-to-point motion will position the optical center to that of the centroid. However, due to cylindrical image distortion, a single point-to-point move seldom aligns the optical center and the centroid. Any overshoot will cause unnecessary oscillations in the camera motion. A proportional gain is added to resolve this problem. More discussion on the system characteristic with different gain values are presented in Section 5.1.3.

The PID position control loop runs at 1 KHz real-time. The visual feedback loop is executed upon completion of camera motion, i.e., when the camera comes to a complete stop. The look-and-move structures are usually implemented on commercial robots with close loop joint controllers.

A sample C code of the position-based look-and-move visual servoing is shown in Figure 5.3. The flowchart of the system is shown in Figure 5.4.

```
while(key!='x')
{
  if(kbhit()) key=getch();
  MbufGet(MilRedBand, RedTemp);
  MbufGet(MilGreBand, GreTemp);
  MbufGet(MilBluBand, BluTemp);

  //****************thresholding of image***************
  m00=0,
  m01=0,
  m10=0;

  for(j=0;j<CHeight;j=j+3)
  {
    for(i=0;i<CWidth;i=i+3)
    {
        Red = (double)RedTemp[CWidth*j+i]/(double)GreTemp[CWidth*j+i];
        Blu = (double)BluTemp[CWidth*j+i]/(double)RedTemp[CWidth*j+i];

        if(Red>=1.2 && Blu<=0.9)
        {
           m00++;
           m10 += j;
           m01 += i;
        }
    }
  }

  if(m00>100)  // minimum no. of pixels detected b4 move enable
  {
    X = (double)m01/(double)m00;
    Y = (double)m10/(double)m00;
  }

  else
  {
    X = 0.5*(CHILD_END_X - CHILD_START_X);
    Y = 0.5*(CHILD_END_Y - CHILD_START_Y);
  }

  data->PixErr[0] = 0.4*data->PixErrN[0]+0.3*data->PixErrN1[0]+0.3*data->PixErrN2[0];
  data->PixErr[1] = 0.4*data->PixErrN[1]+0.3*data->PixErrN1[1]+0.3*data->PixErrN2[1];

  IPMoveMtrPulse(0, data->PixErr[0]*81.68*0.4);
  IPMoveMtrPulse(1, data->PixErr[1]*79.75*0.4);

  while(IPMoveNotComplete())
    Sleep(1);
  FrameCount++;
}
```

Figure 5.3: C-Code of Position-Based Look-and-Move Sampling Loop

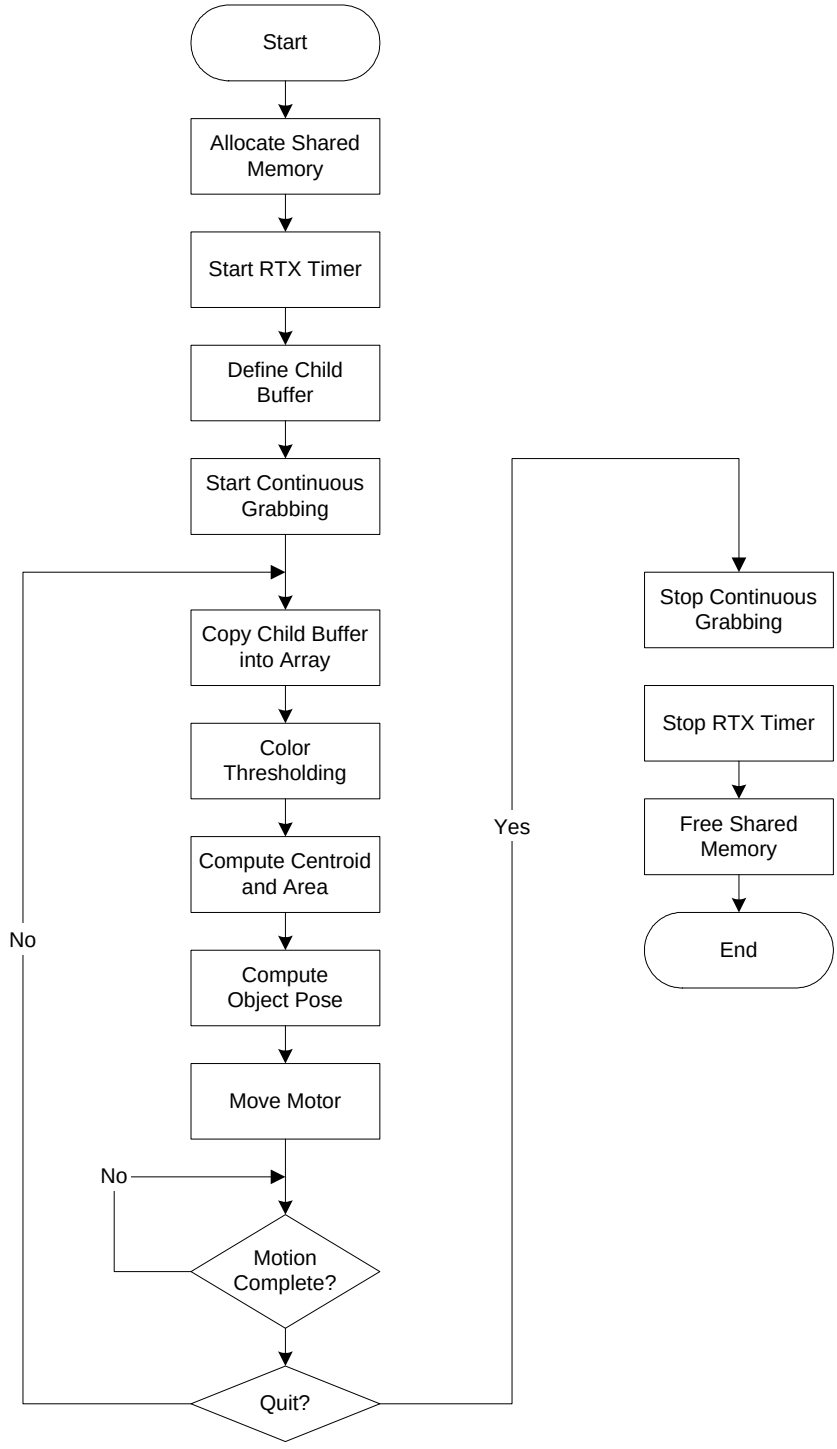

Figure 5.4: Flowchart of Dynamic Position-based Look-and-move Setup

5.1.3 System Characteristic

In this section, the system characteristic of the Position-based Look-and-Move structure is discussed. To study the system characteristic, a step input is given as the visual feedback. The red object is positioned in such a way that its centroid is away from the optical center. The magnitude of this step input is not critical to the observation as the interest is with the response of the system. The program is then executed while the pixel errors and motor counter readings are being recorded.

Step Response for different value of proportional gain is recorded and plotted in the figures below. It can be seen in Figure 5.3 that there is pronounce oscillations for the first ten seconds after the step input is applied to the system with a proportional gain of 1.0. The system is tested under other values of gain. The plots for gain value 0.4 are shown in Figures 5.7, 5.8, 5.9 and 5.10. The scales are maintained for easy comparison.

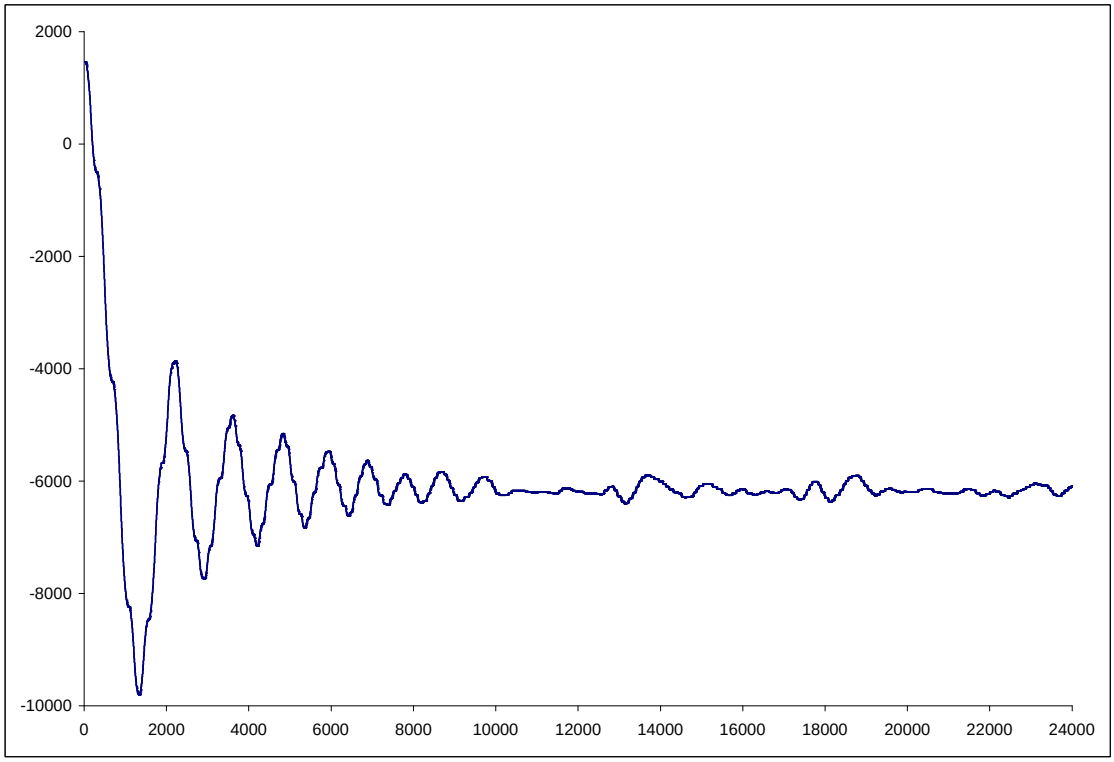

Figure 5.5: Plot of x against time(ms) for Motor 0 with gain K = 1.0

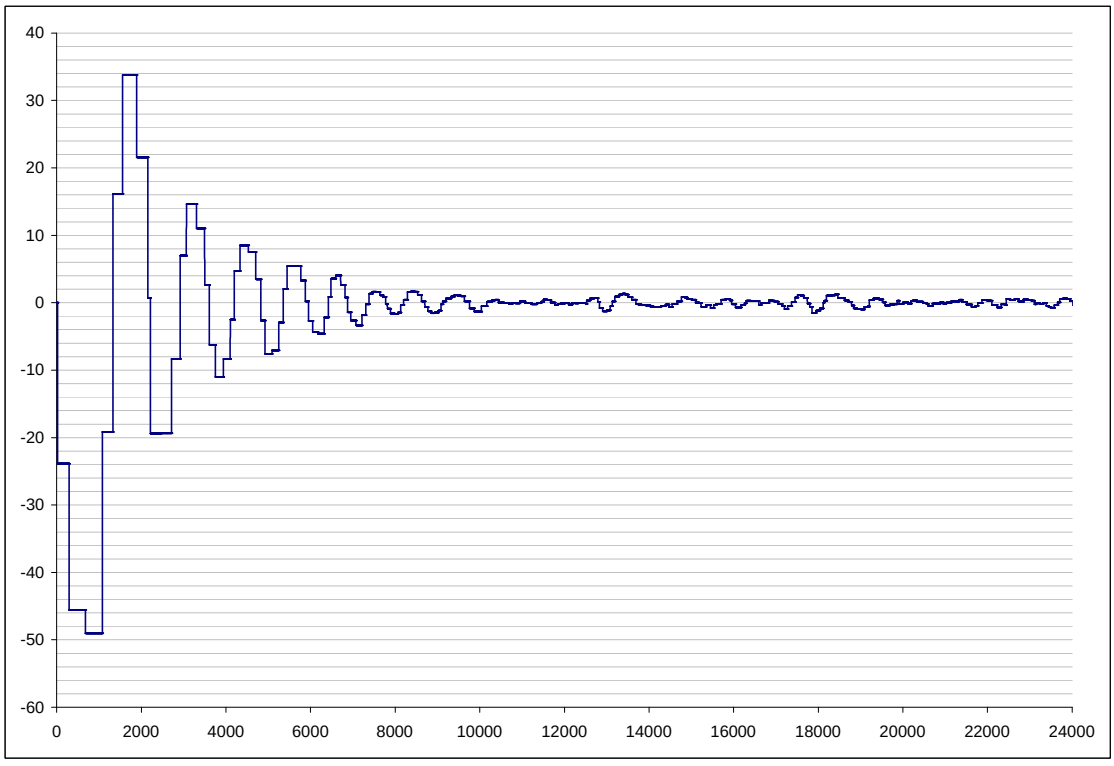

Figure 5.6: Plot of pixel error against time(ms) for Motor 0 with gain K = 1.0

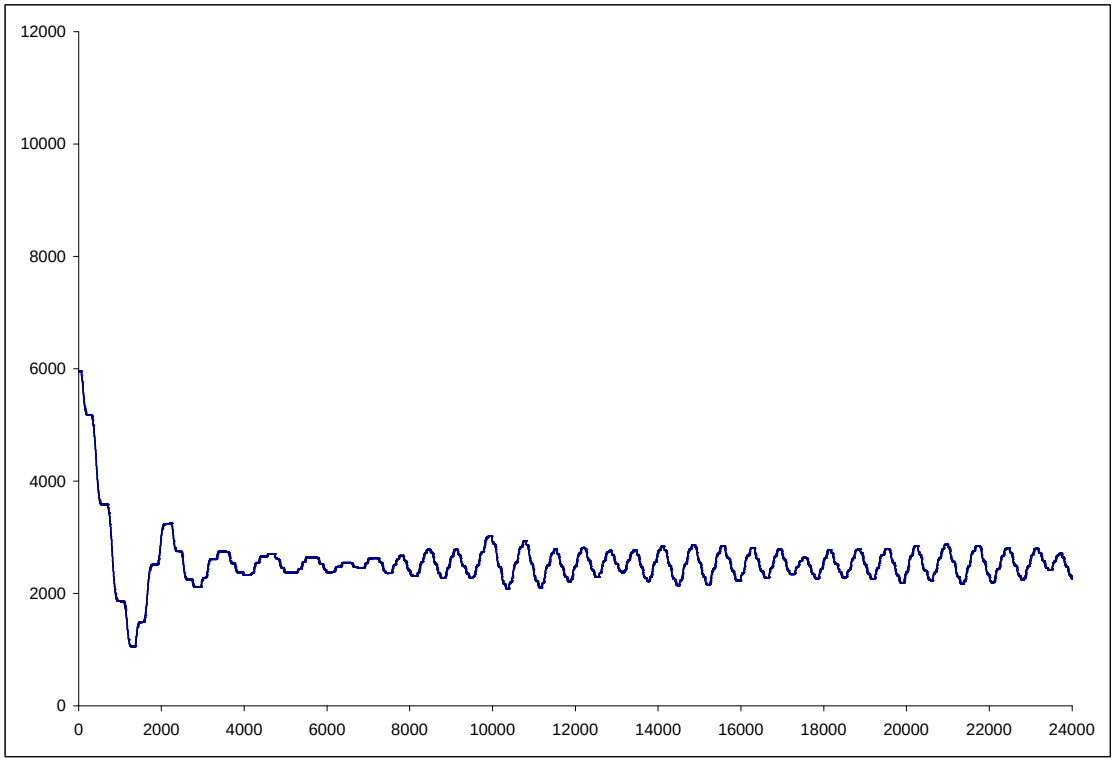

Figure 5.7: Plot of x against time(ms) for Motor 1 with gain K = 1.0

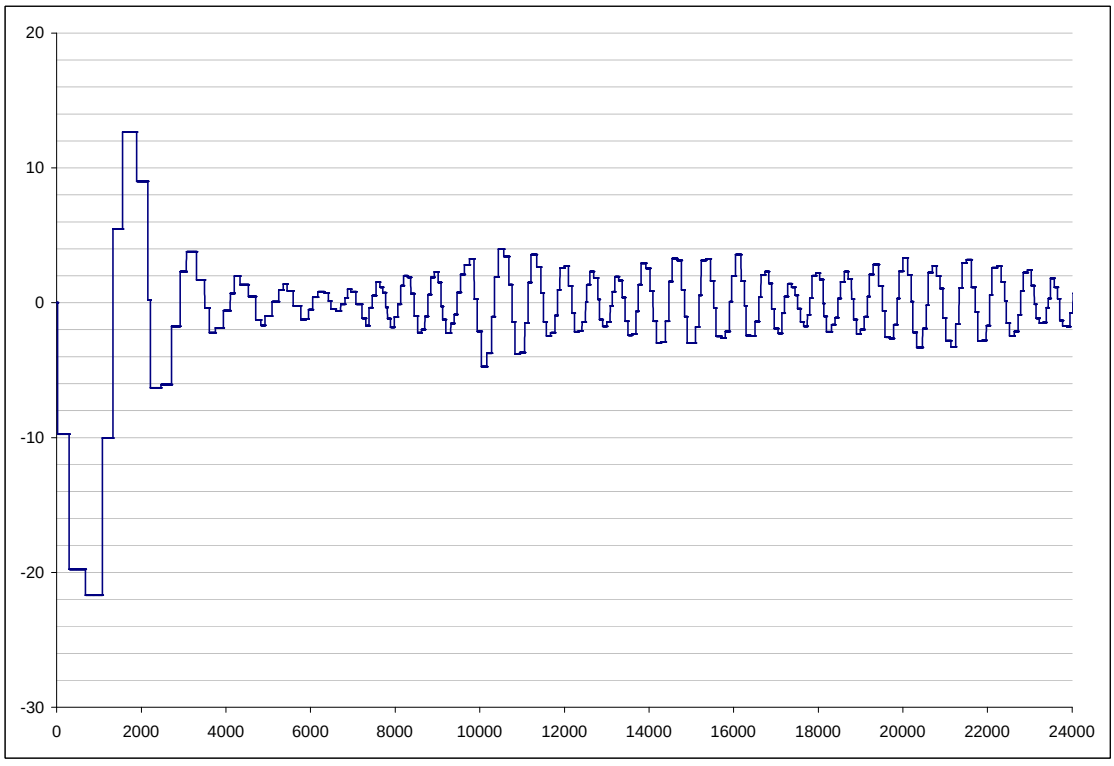

Figure 5.8: Plot of pixel error against time(ms) for Motor 1 with gain K = 1.0

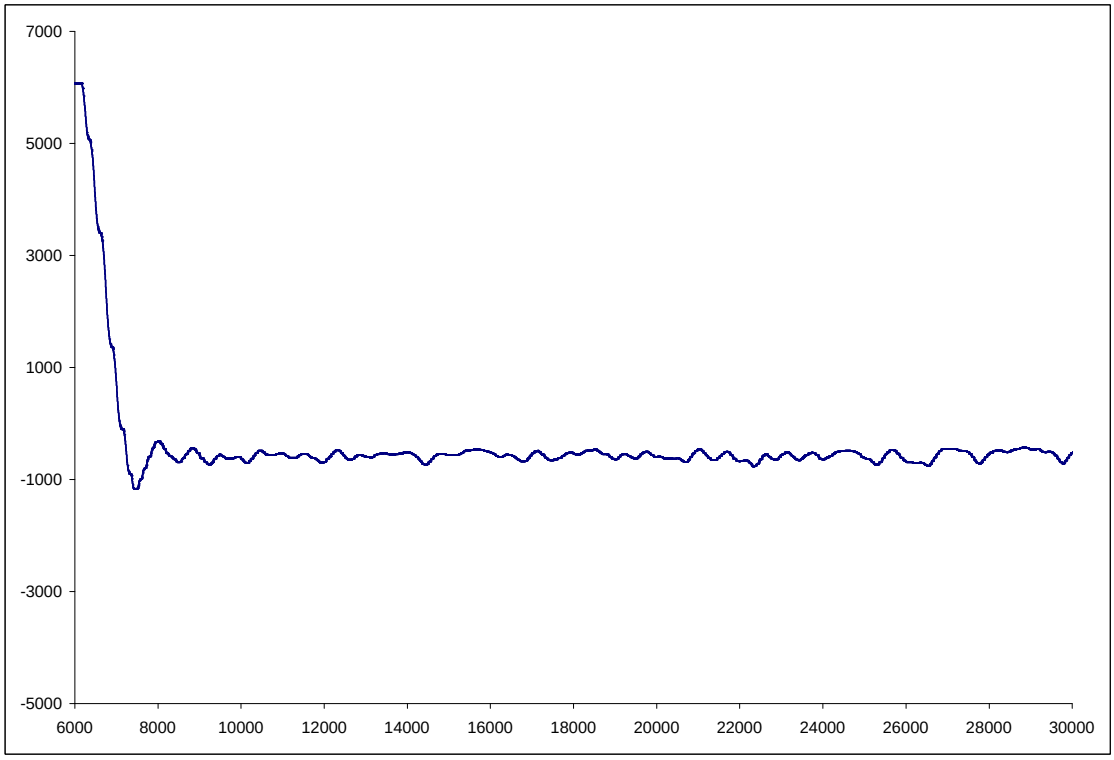

Figure 5.9: Plot of x against time(ms) for Motor 0 with gain K = 0.4

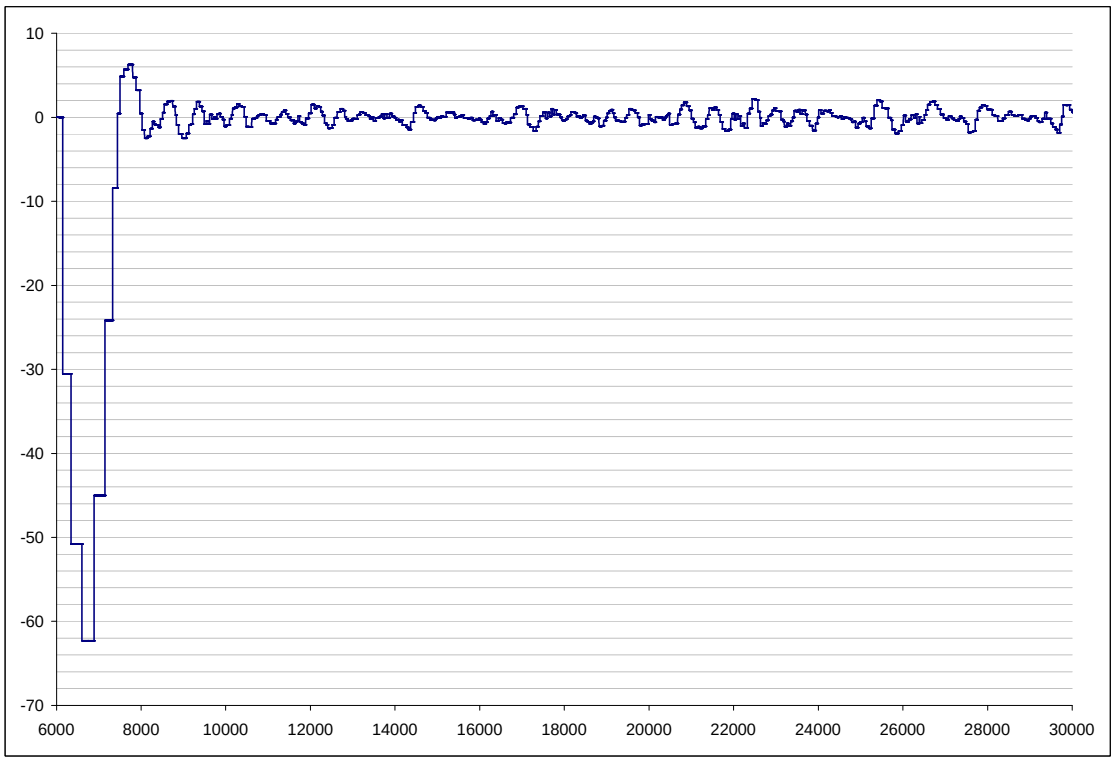

Figure 5.10: Plot of pixel error against time(ms) for Motor 0 with gain K = 0.4

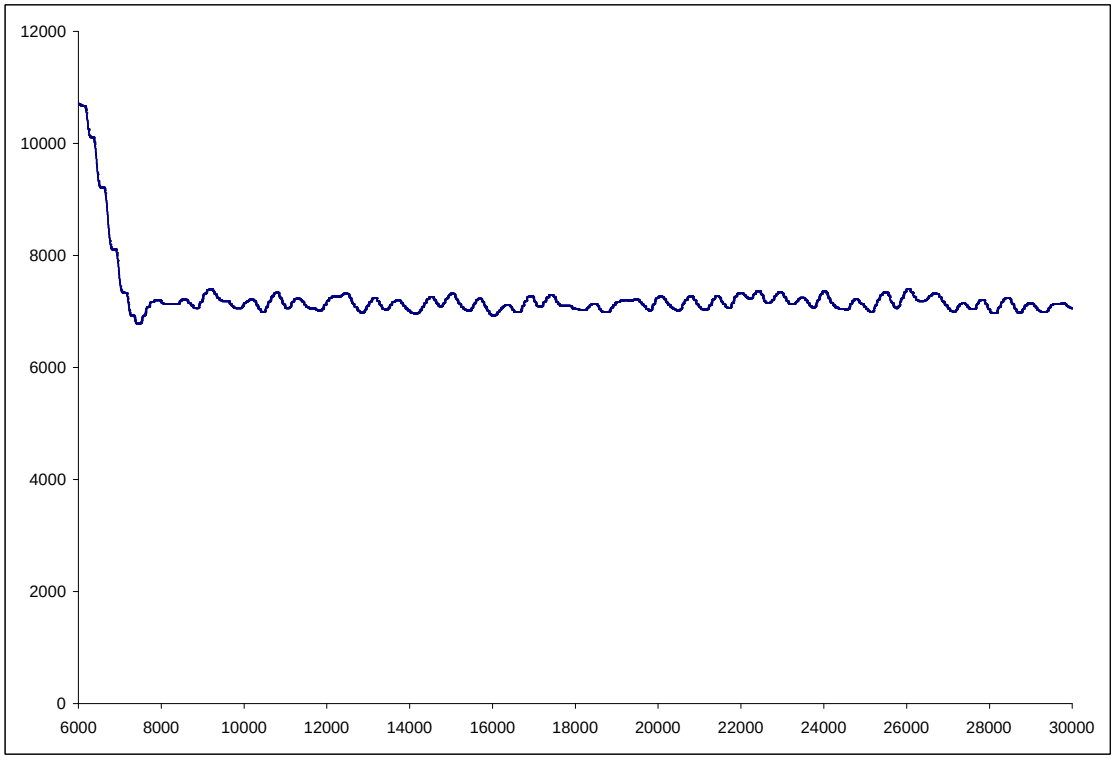

Figure 5.11: Plot of x against time(ms) for Motor 1 with gain K = 0.4

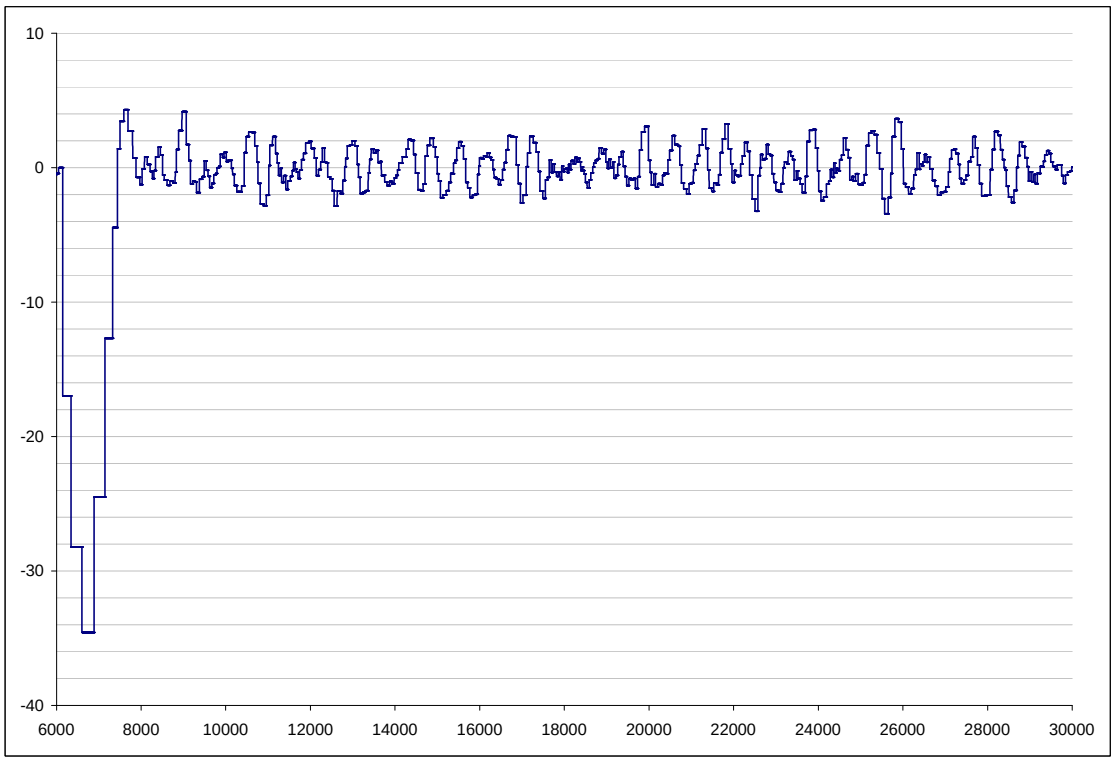

Figure 5.12: Plot of pixel error against time(ms) for Motor 1 with gain K = 0.4

When a gain of 0.4 is used, it is observed that the settling time is reduced from about 10 seconds to about 5 seconds for both the pan and tilt axes. The overshoots are caused by errors in calibration. Image distortion at the edges is a possible cause of this error in calibration.

It can be seen from the pixel error against time plots that when a gain of 1.0 is used, more time is needed to complete the point-to-point movement hence reducing the vision sampling rate. The width of the horizontal step is the time needed to complete each point-to-point move. When gain is reduced to 0.4, the path becomes shorter, hence increasing the sampling rate which affects the tracking performance.

The robot motion is observed to be jerky when tracking a moving object. An image frame is grabbed only when the robot comes to a complete stop. After an image is grabbed and the object's relative pose computed, each axis will accelerate linearly to a maximum velocity then decelerate linearly to a stop. When this motion is completed, then the next image frame is grabbed. This explains why the motion tends to be jerky.

This implementation of the position-based look-and-move structure more suited for tracking stationary objects within the camera view instead of a moving object. For better performance in tracking moving target, the Image-based Visual Servo structure is implemented and discussed in Section 5.2.

5.2 Image-Based Visual Servo Structure

5.2.1 Calibration

Calibration involves relating the X and Y pixel spacing in the image plane to a length in the real world with a factor. For position-based visual servoing techniques, pose determination is required for the computation of inverse kinematics for the control of robot axes. Hence calibration is particularly important when implementing the Position-Based techniques. In Image-Based techniques, pose determination is not required as the control of robot axes are all relative to the feature detected.

5.2.2 Implementation

As with the Position-based Look-and-Move structure, a real-time position control using PID is implemented. However, joint controllers for trajectory planning are not required in the Image-based Visual Servoing structure. The information obtained from the feature is used directly as a control signal for the system.

Figure 5.13 shows the block diagram of the visual servoing structure that is adopted for the setup.

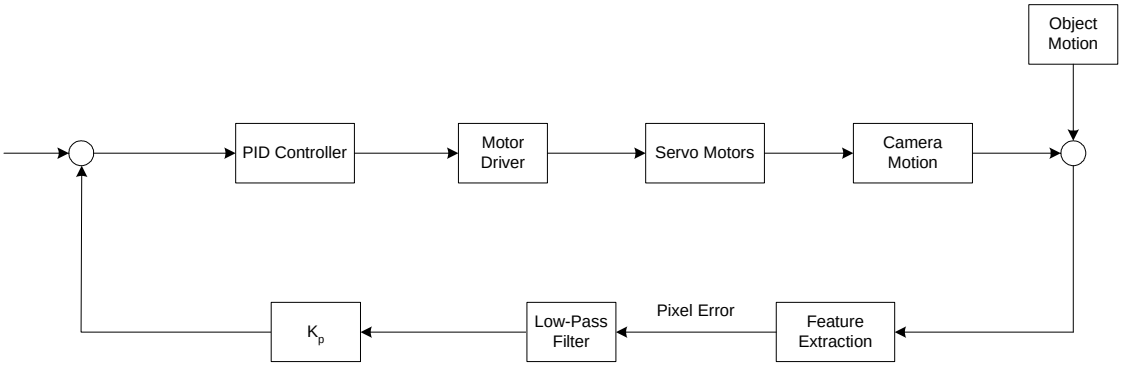

Figure 5.13: Image-Based Visual Servo Structure

The internal PID control loop runs at 1 KHz real-time while the vision feedback loop runs at approximately 139 MHz.

A Proportional Gain value is needed before the Pixel Error can be use as a feedback to the control loop because of the difference in the units. The selection of this gain value will be discussed more in Section 5.2.3 of this chapter.

The C-codes for the real-time PID control with visual feedback is presented in Figure 5.14. Pixel Error is being passed from the visual feedback loop into the PID control loop through the shared memory which is accessible to both loops. The flowchart of the Image-based Visual Servo structure is shown in Figure 5.15.

```
while(1) // real-time loop
{
  for(axis = 0; axis < NAXIS; axis++)
  {
    data->X[axis] = RltReadCntr(axis);       // read counter

    data->XRef[axis]= data->XRef[axis] + 0.4*data->PixErrN[axis];  // visual servoing

    XErrN[axis] = data->XRef[axis] - (double)data->X[axis];

    data->control[axis] = data->controlOld[axis]
                        + data->K1[axis] * XErrN[axis]
                        + data->K2[axis] * data->XErrN1[axis]
                        + data->K3[axis] * data->XErrN2[axis];

    DACout = (USHORT)(-data->control[axis]) + 0x1000;
    addpus=(PUSHORT)(STG_A + DAC0 + 2 * axis);
    RtWritePortUshort(addpus, DACout);       // writes to port

    data->controlOld[axis] = data->control[axis];
    data->XErrN2[axis] = data->XErrN1[axis];
    data->XErrN1[axis] = XErrN[axis];
  }
}
```

Figure 5.14: C-Code of Real-time PID Control with Visual Feedback

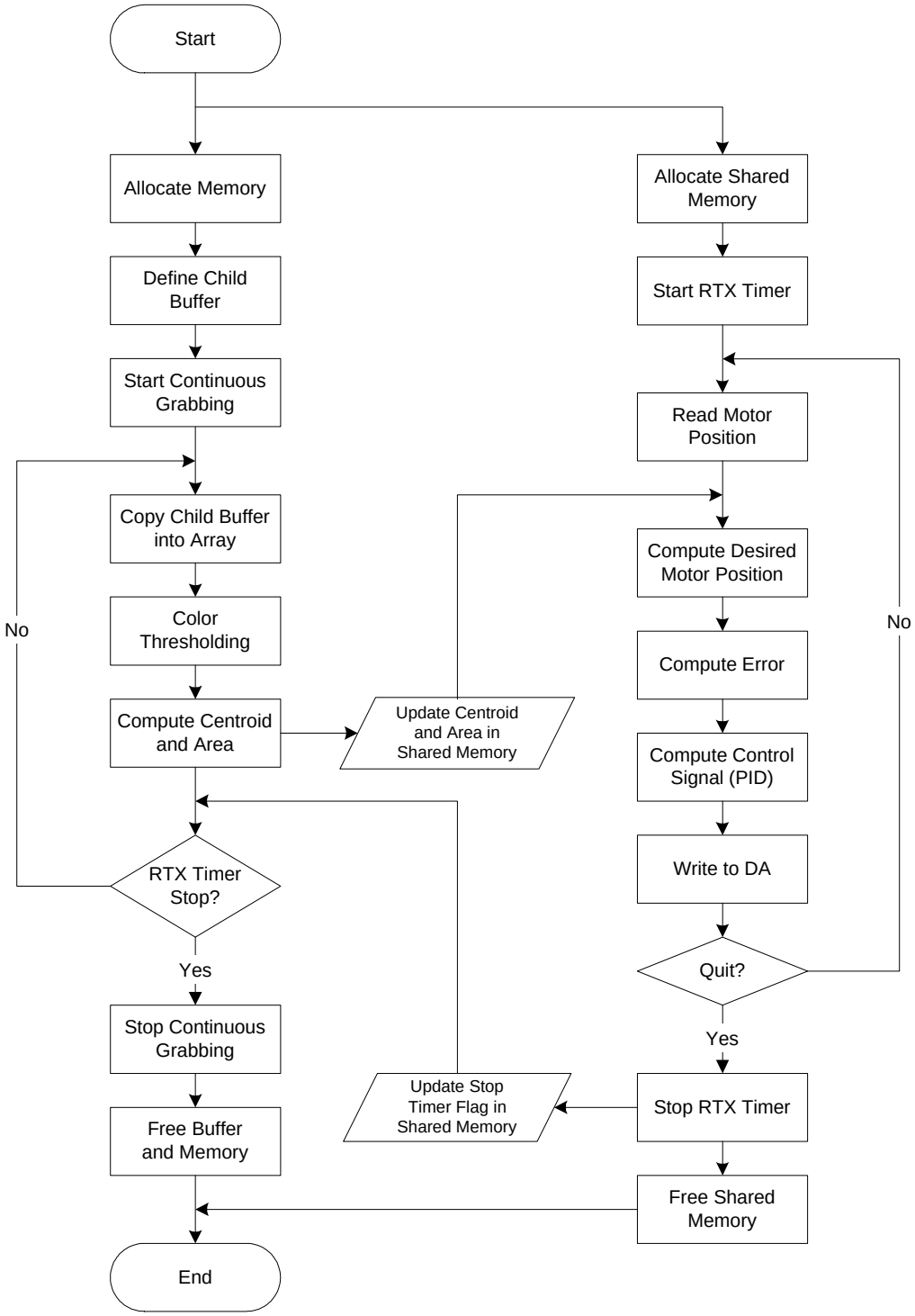

Figure 5.15: Flowchart of Image-based Visual Servoing Setup

5.2.3 System Characteristic

In this section, the system characteristic of the Image-based Visual Servo setup is discussed. Similar to that for the Position-based Look-and-Move structure, a step input is given as the visual feedback. The red object is positioned in such a way that its centroid is away from the optical center. The program is then executed while the pixel errors and motor counter readings are being recorded.

Step Response for different value of proportional gain is recorded and plotted in the figures below. The time scale is maintained for easy comparison. Data for a sampling time of 7 seconds are plotted.

When a gain value of 0.1 is used, it can be observed from Figures 5.17 and 5.19 that it takes more than 3 seconds for the pixel error of both axes to reach within 2-pixel error. As the gain value is increased from 0.1 to 0.4, it can be observed from the plots that time taken to reach within 2-pixel error decreases. When the gain value is 0.4, it takes less than a second for the pixel error to reach within 2 pixels.

It can also be seen from the graph that the pixel error tends to oscillate within the 2-pixel error range. This range whereby the pixel error oscillates can be used as a measure of the system accuracy. It is also observed that as the gain increases, the frequency of this oscillation increases and the motion of the robot tends to be stiffer. Gain value of 0.4 is a good compromise between response time and smooth robot motion.

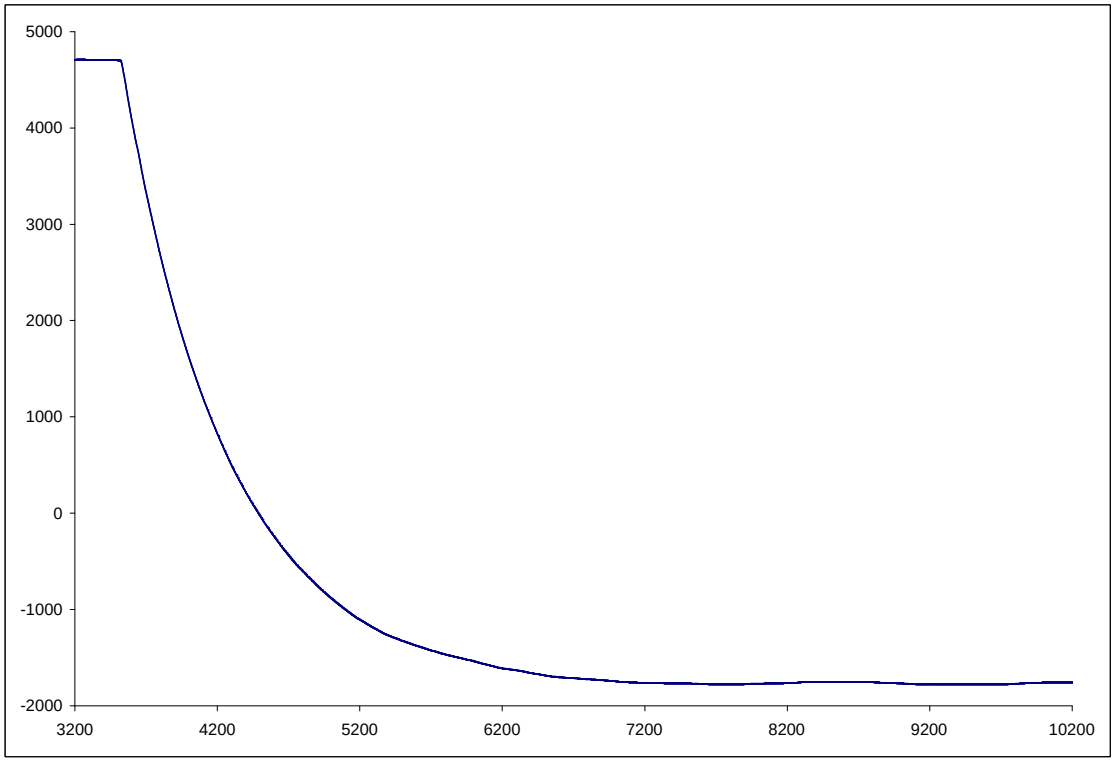

Figure 5.16: Plot of x against time(millisecond) for Motor 0 with K = 0.1

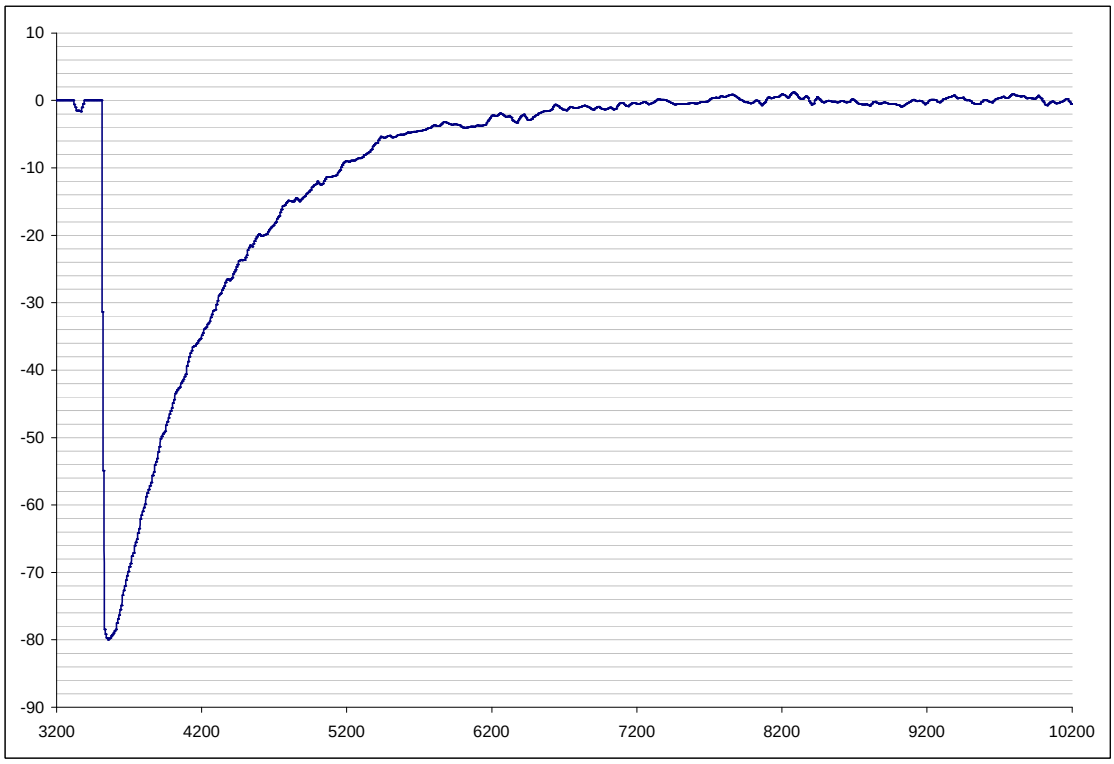

Figure 5.17: Plot of Pixel Error against Time(millisecond) for Motor 0 with K = 0.1

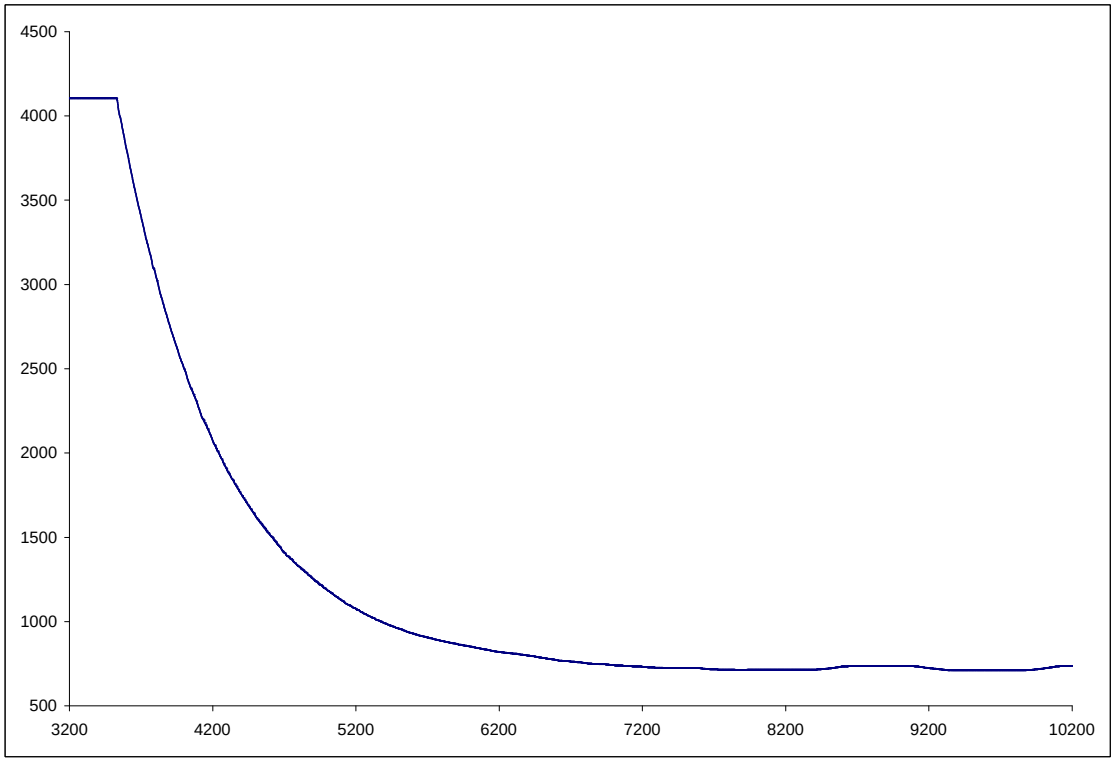

Figure 5.18: Plot of x against time(millisecond) for Motor 1 with K = 0.1

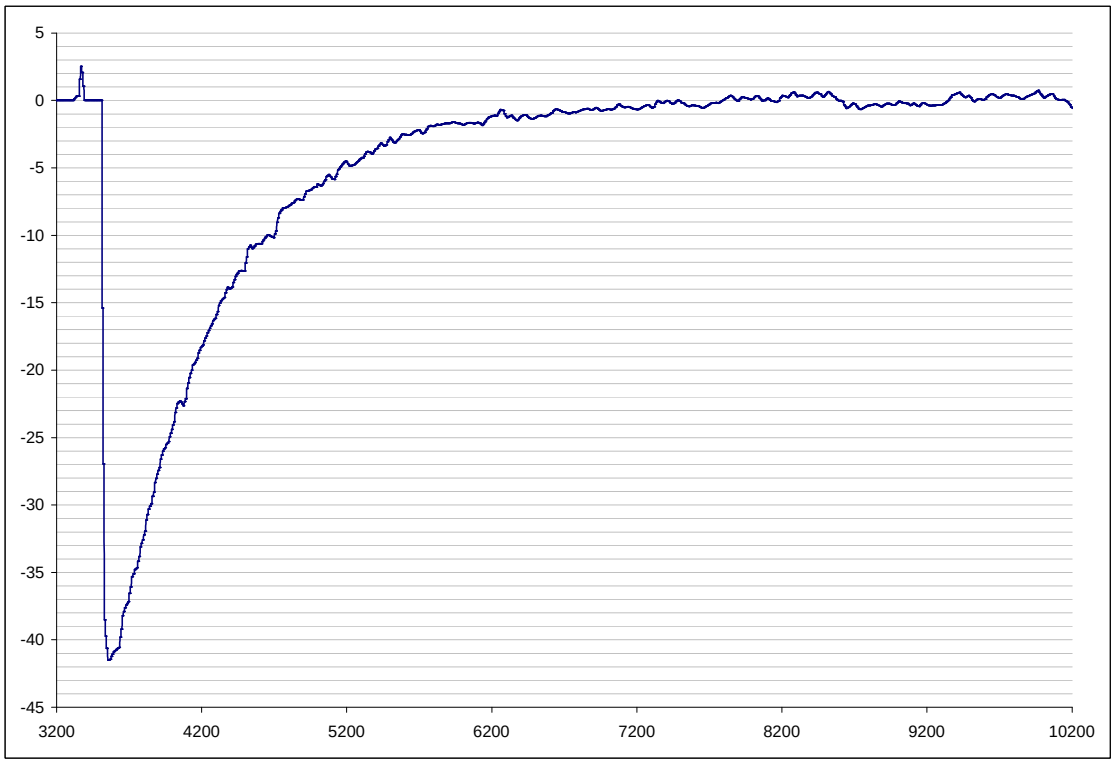

Figure 5.19: Plot of Pixel Error against Time(millisecond) for Motor 1 with K = 0.1

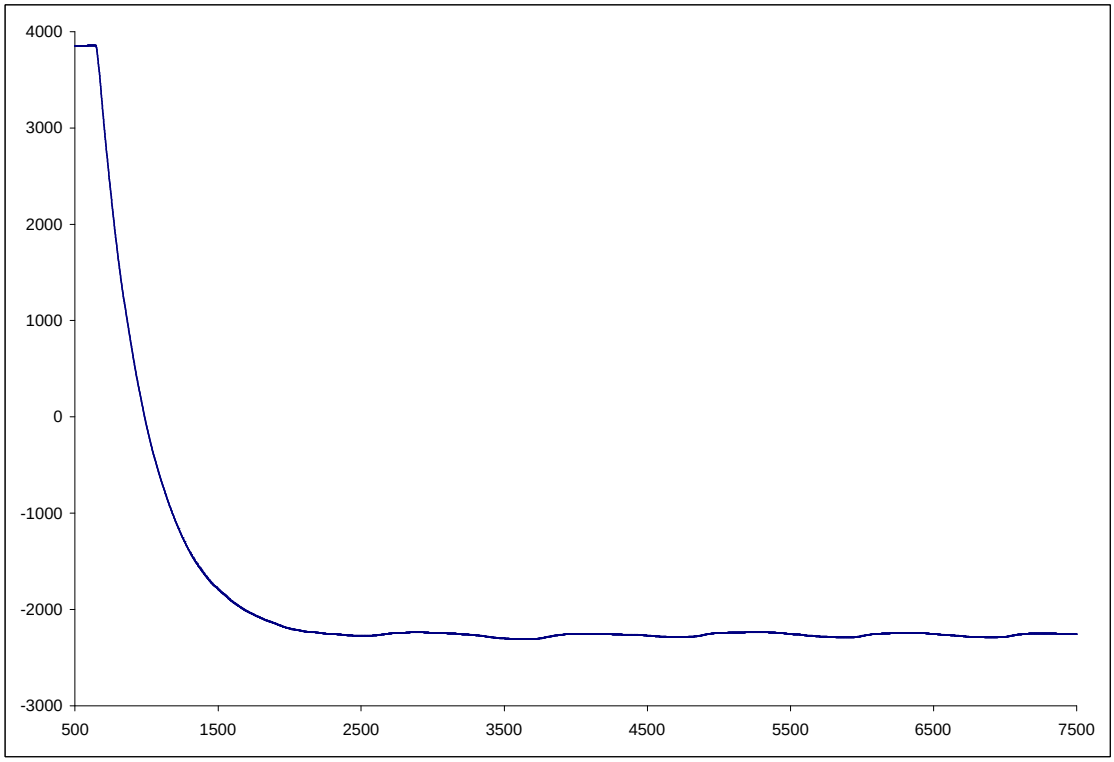

Figure 5.20 Plot of x against time(millisecond) for Motor 0 with K = 0.2

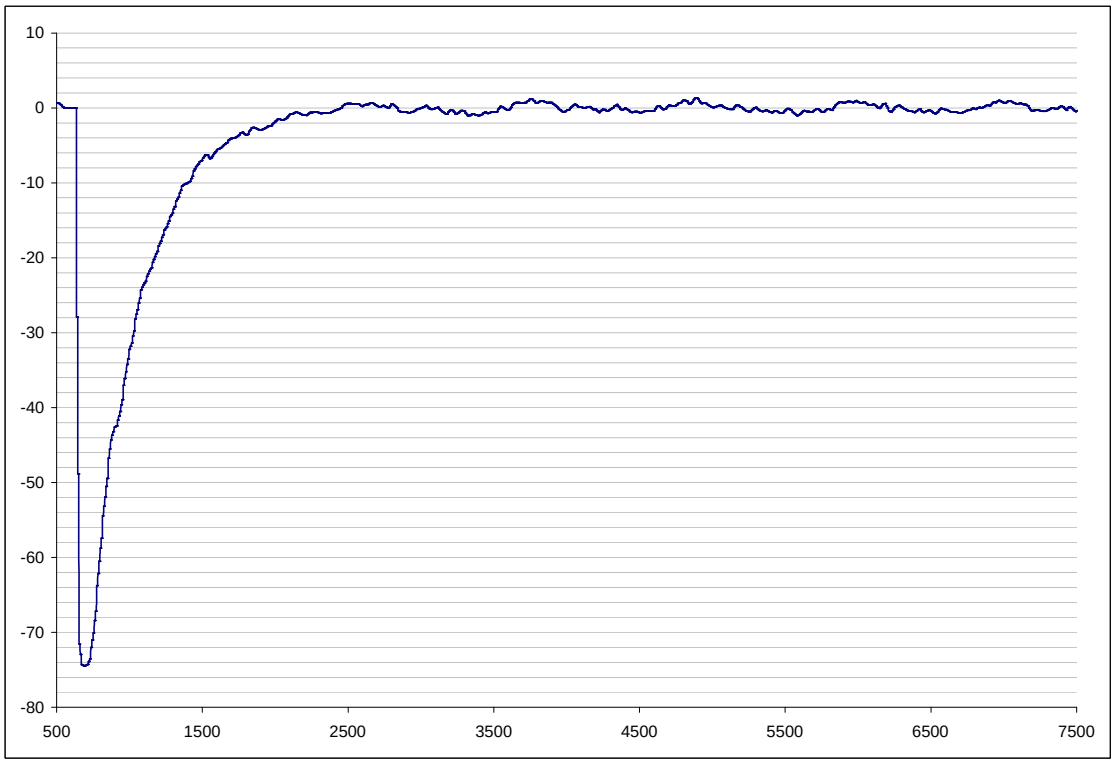

Figure 5.21: Plot of Pixel Error against Time(millisecond) for Motor 0 with K = 0.2

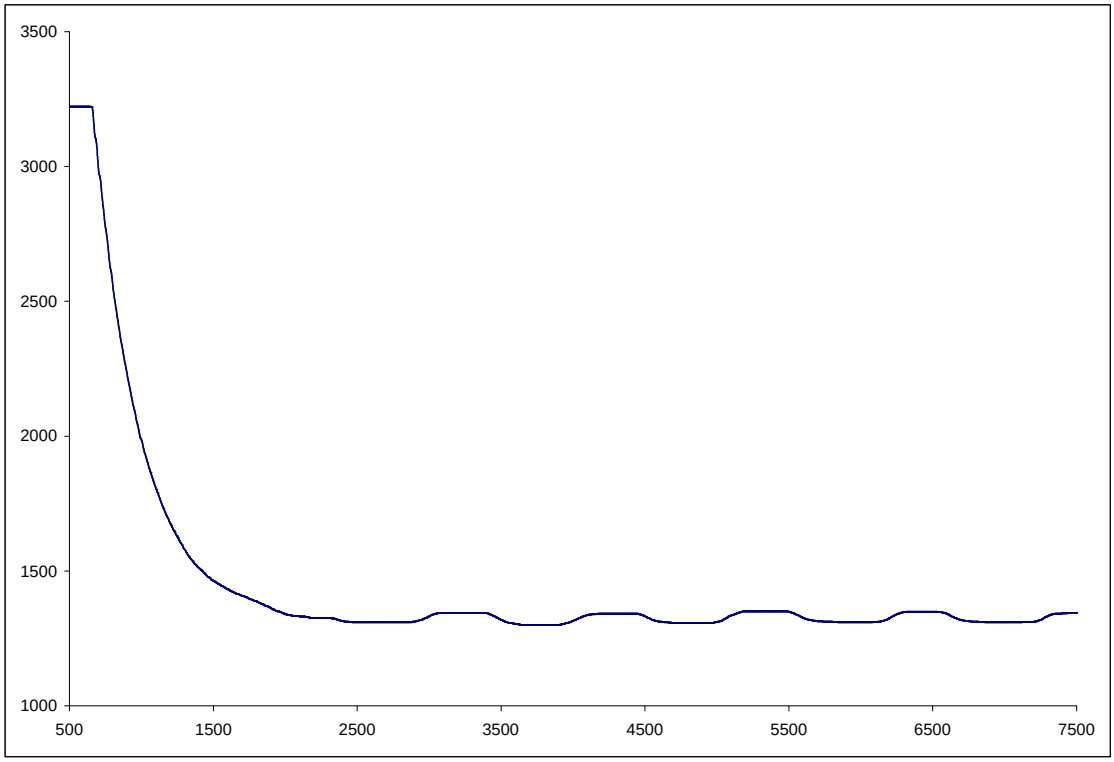

Figure 5.22: Plot of x against time(millisecond) for Motor 1 with K = 0.2

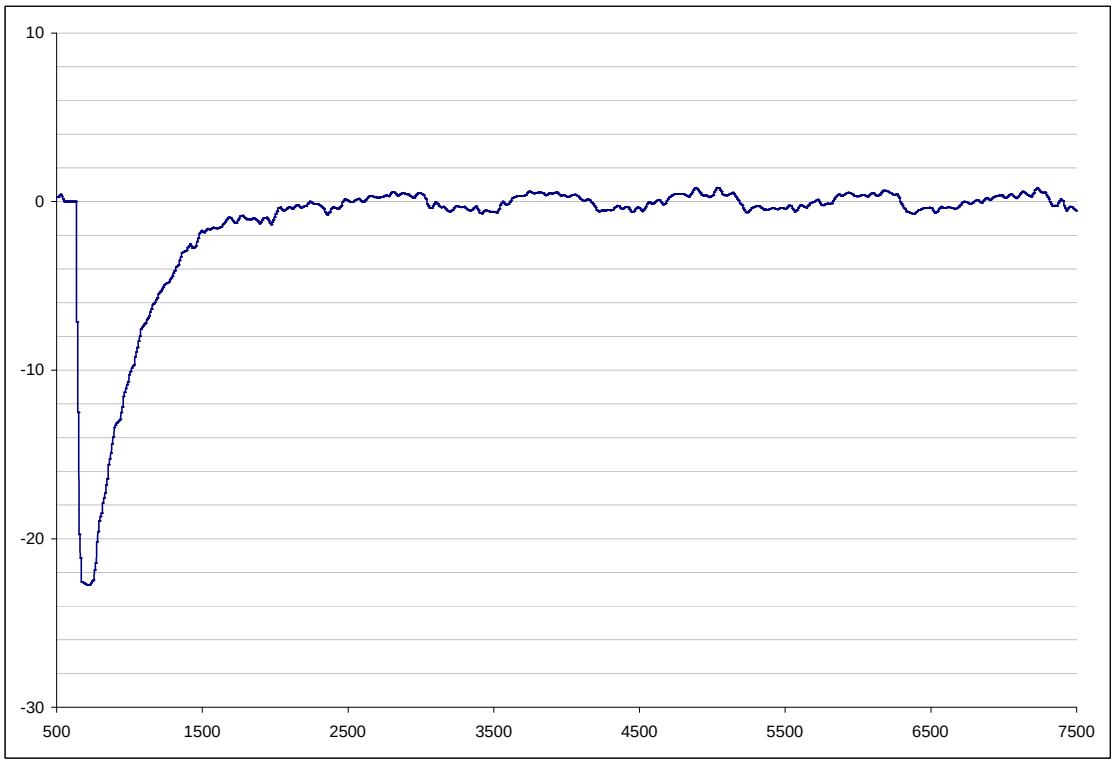

Figure 5.23: Plot of Pixel Error against Time(millisecond) for Motor 1 with K = 0.2

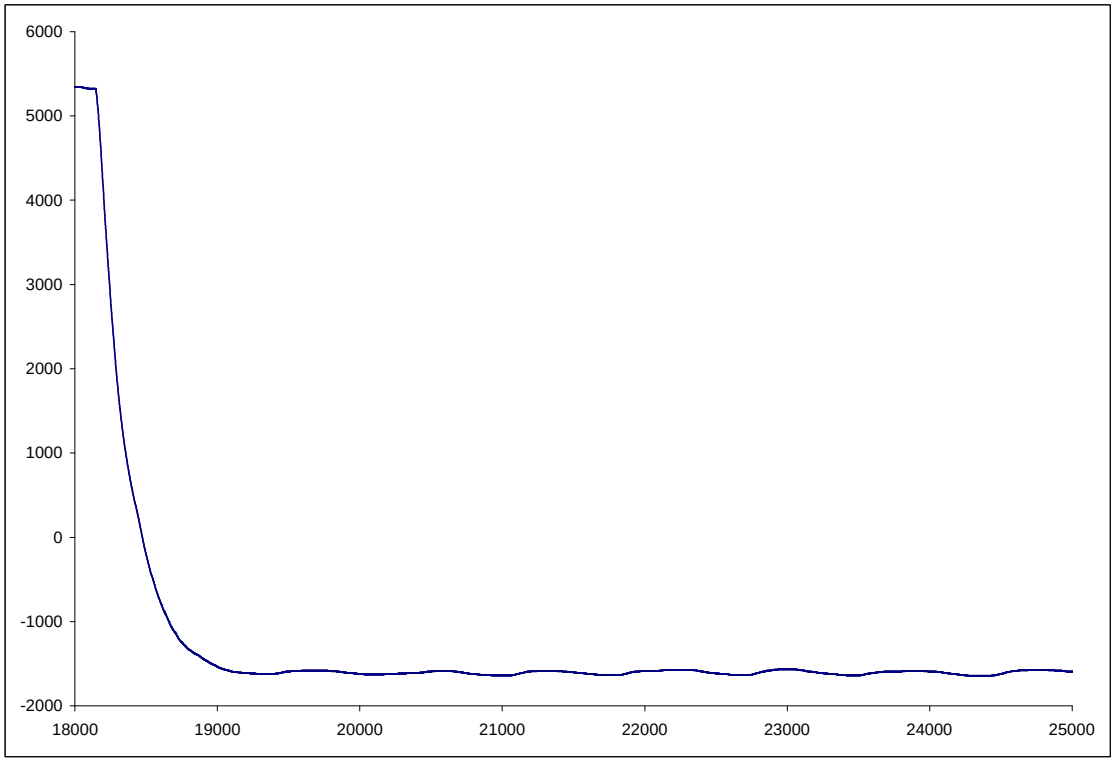

Figure 5.24: Plot of x against time(millisecond) for Motor 0 with K = 0.3

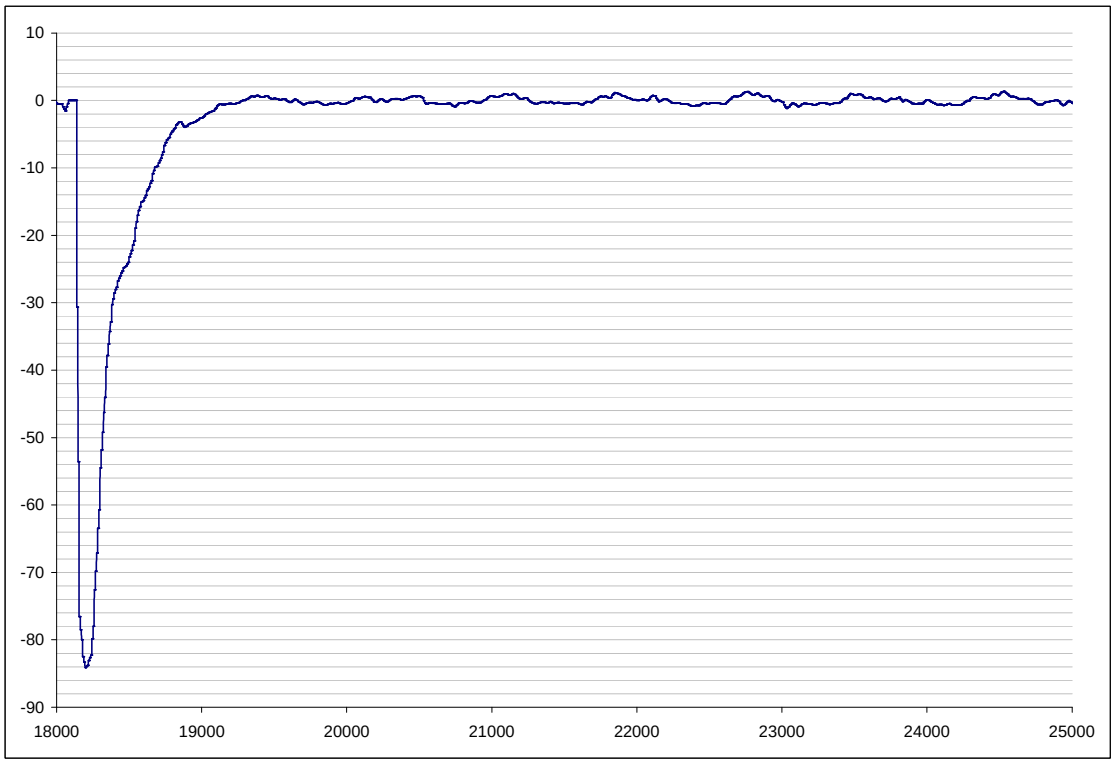

Figure 5.25: Plot of Pixel Error against Time(millisecond) for Motor 0 with K = 0.3

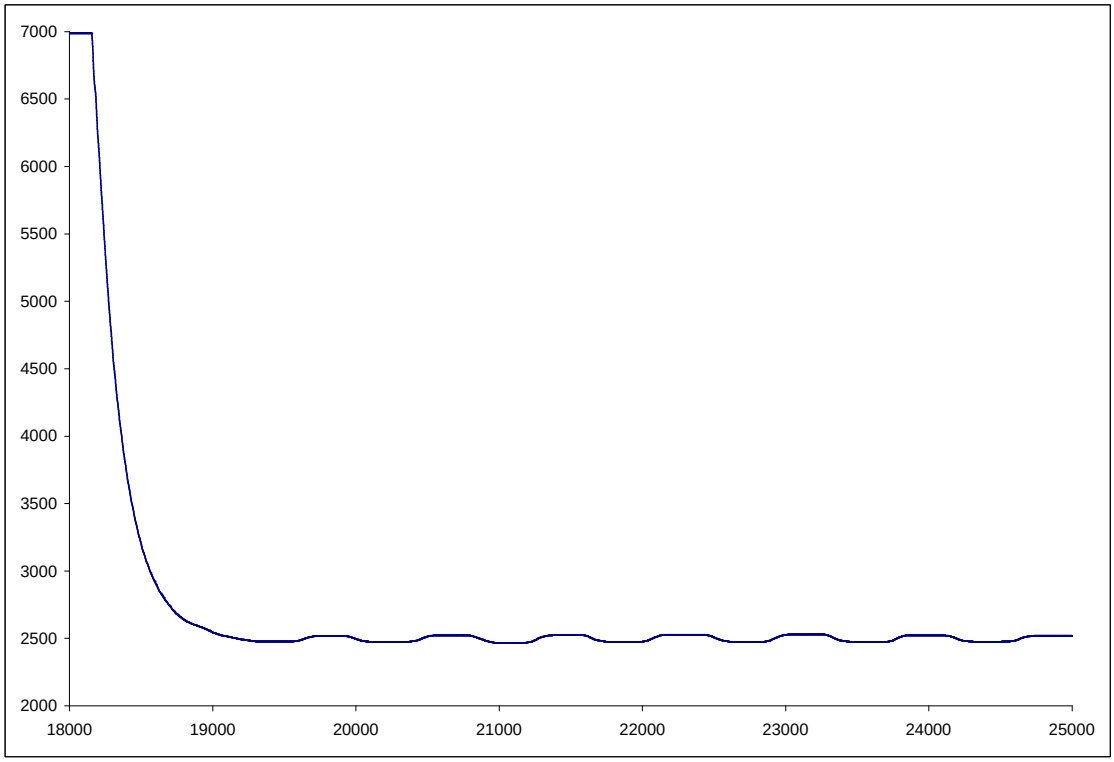

Figure 5.26: Plot of x against time(millisecond) for Motor 1 with K = 0.3

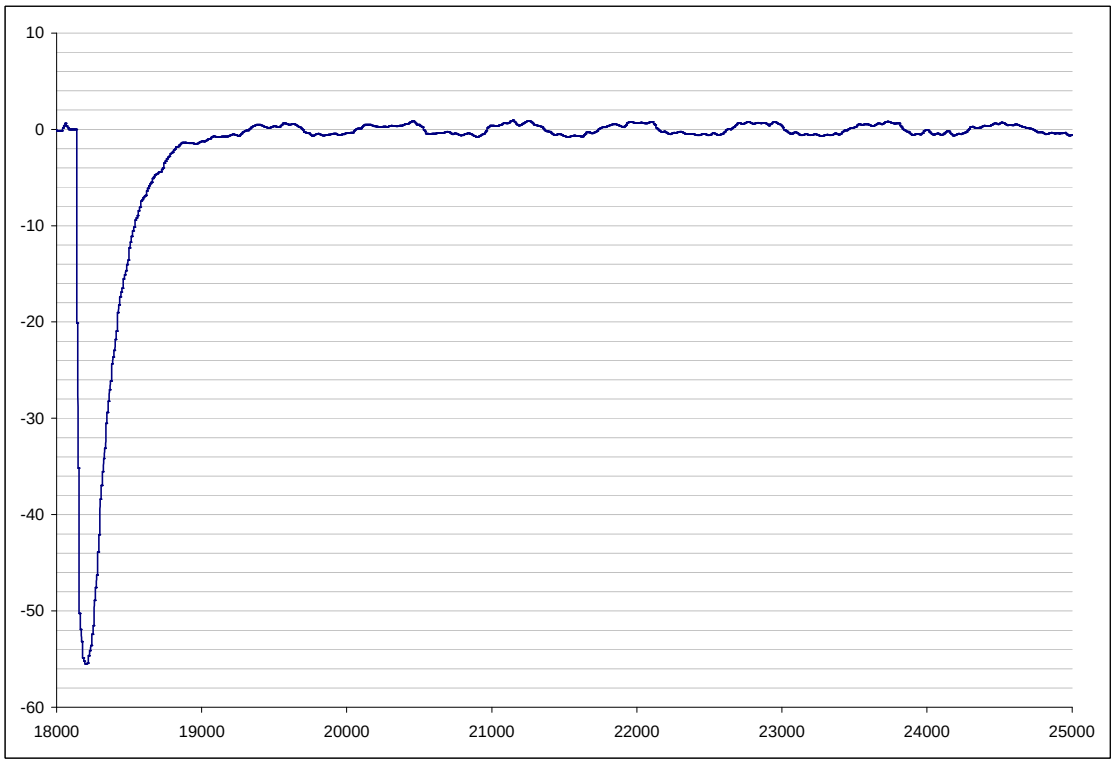

Figure 5.27: Plot of Pixel Error against Time(millisecond) for Motor 1 with K = 0.3

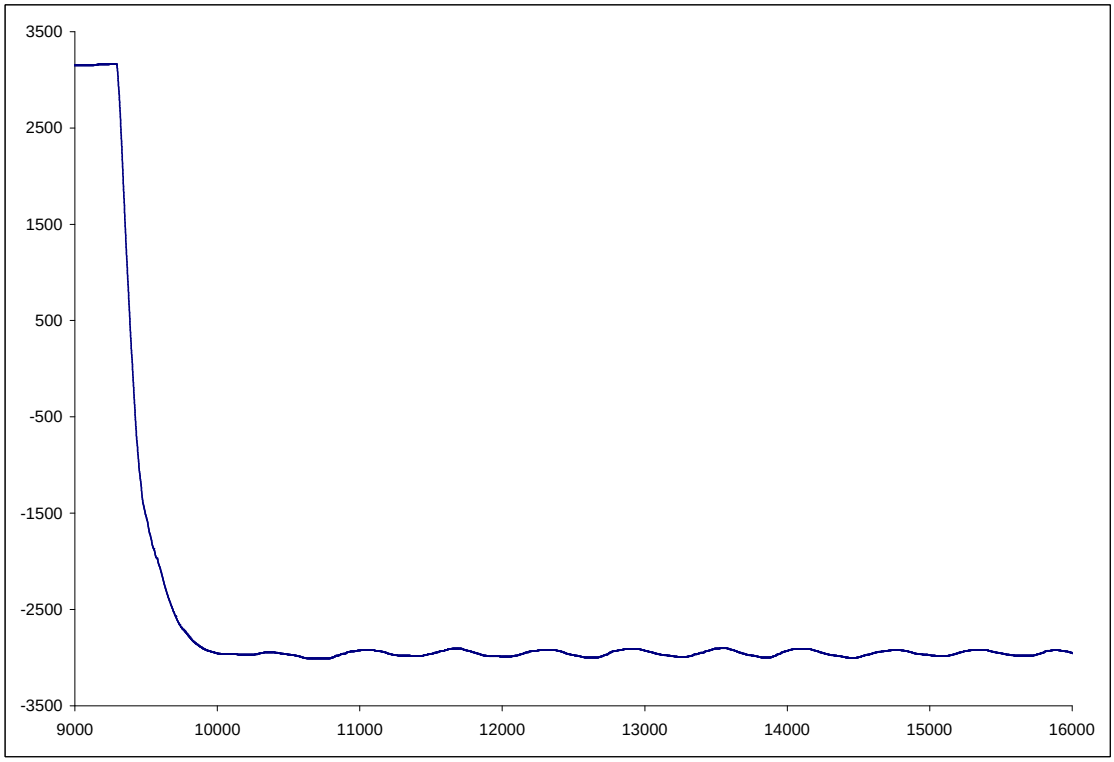

Figure 5.28: Plot of x against time(millisecond) for Motor 0 with K = 0.4

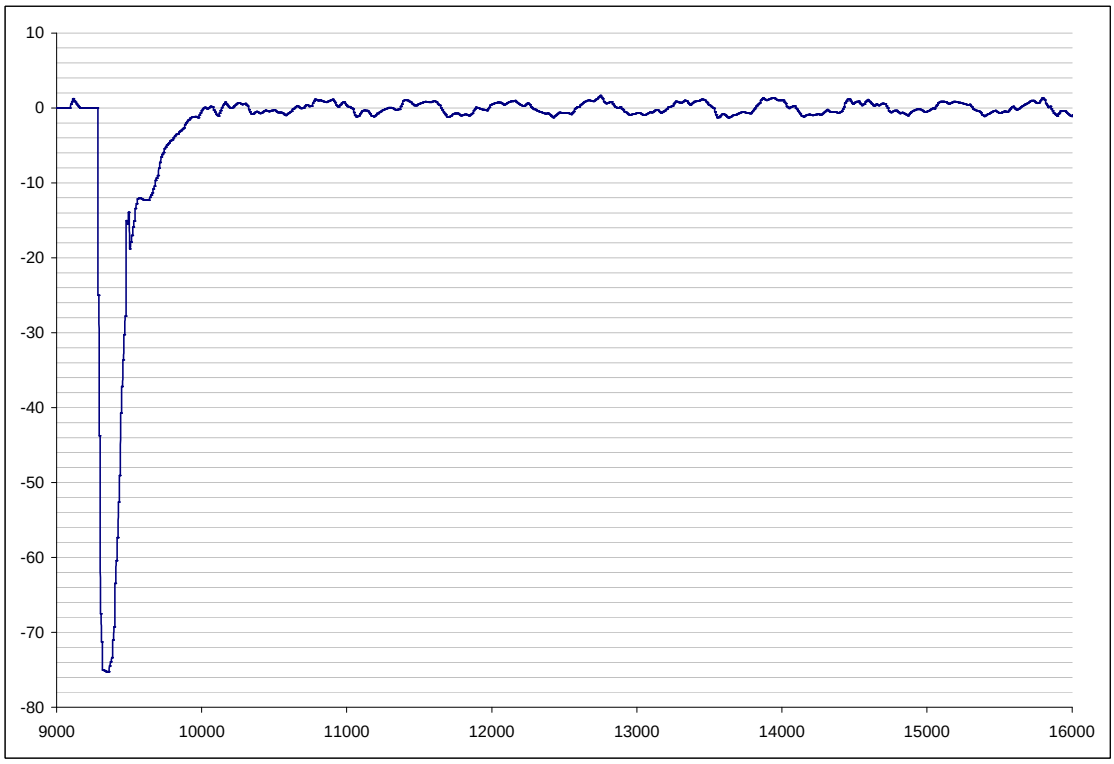

Figure 5.29: Plot of Pixel Error against Time(millisecond) for Motor 0 with K = 0.4

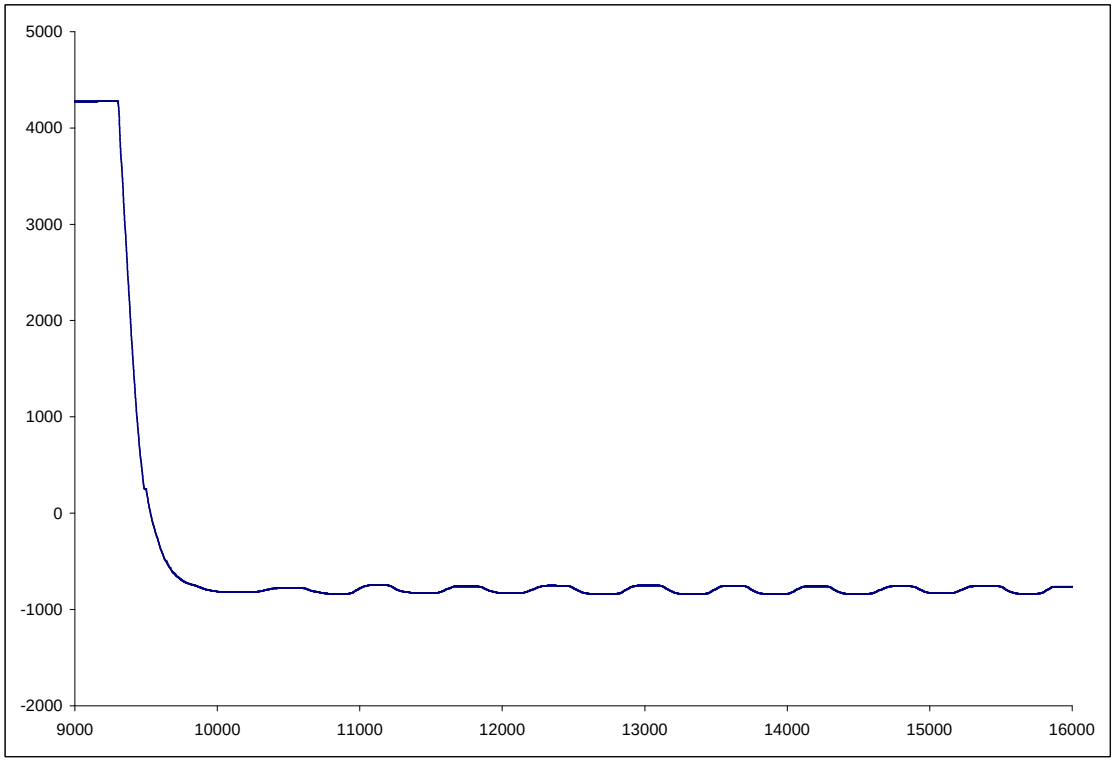

Figure 5.30: Plot of x against time(millisecond) for Motor 1 with K = 0.4

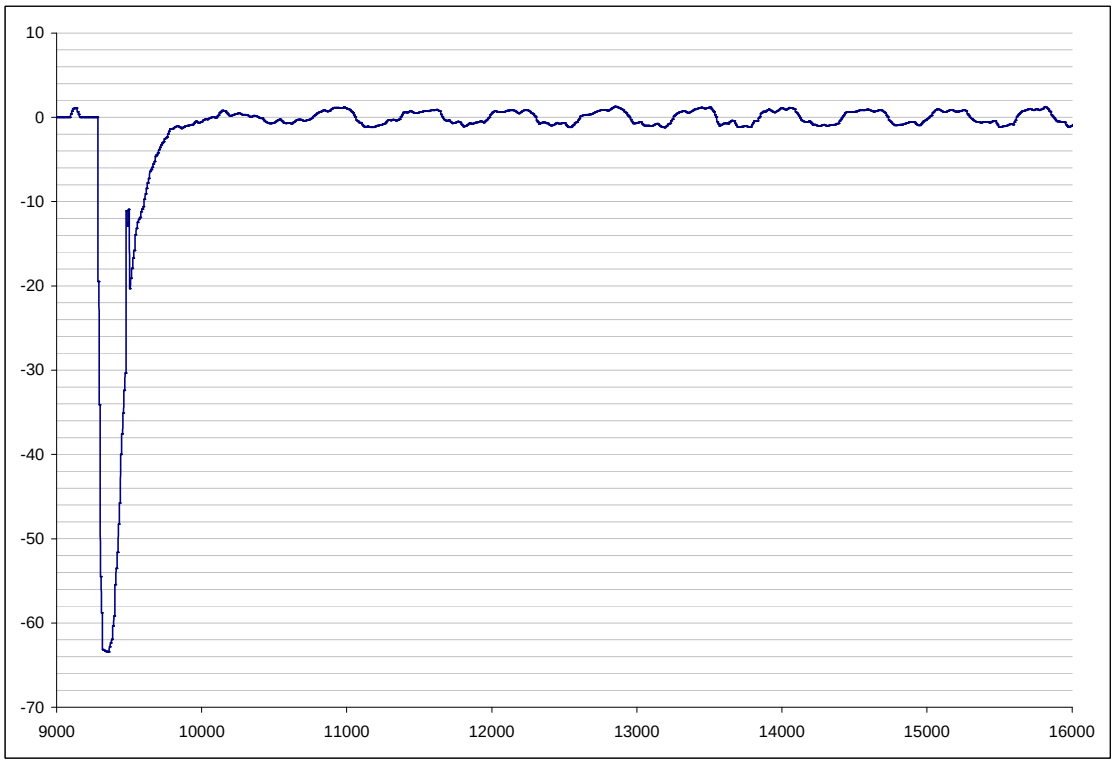

Figure 5.31: Plot of Pixel Error against Time(millisecond) for Motor 1 with K = 0.4

5.2.4 Tracking Performance

As with all tracking device, the performance measurement is the speed at which the robot can track the object. In the case of rotary robots, the tracking performance is measured in terms of angular speed.

In order to get this performance measurement, the target is accelerated from rest within the camera view until the robot fails to track its motion. This is done a few times both vertically and horizontally. Motor position data are collected and plotted in Figure 5.32 and Figure 5.33 for each axis respectively.

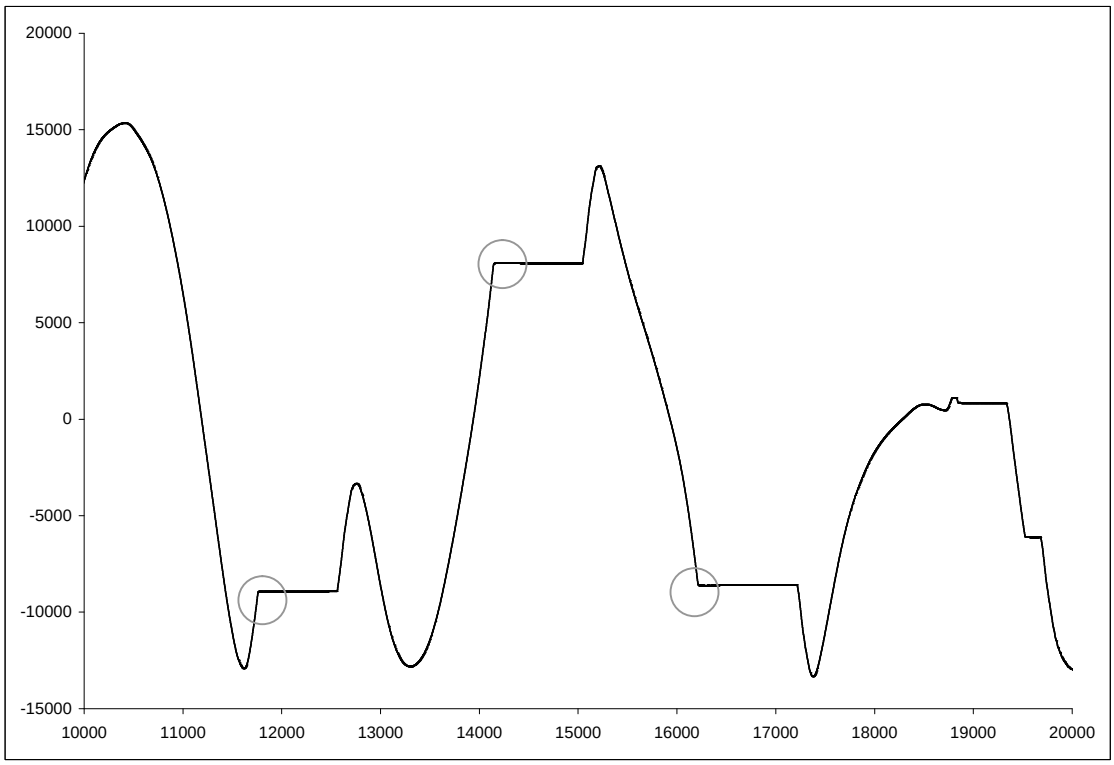

Figure 5.32: Plot of x against time(ms) for Axis 0 (Pan)

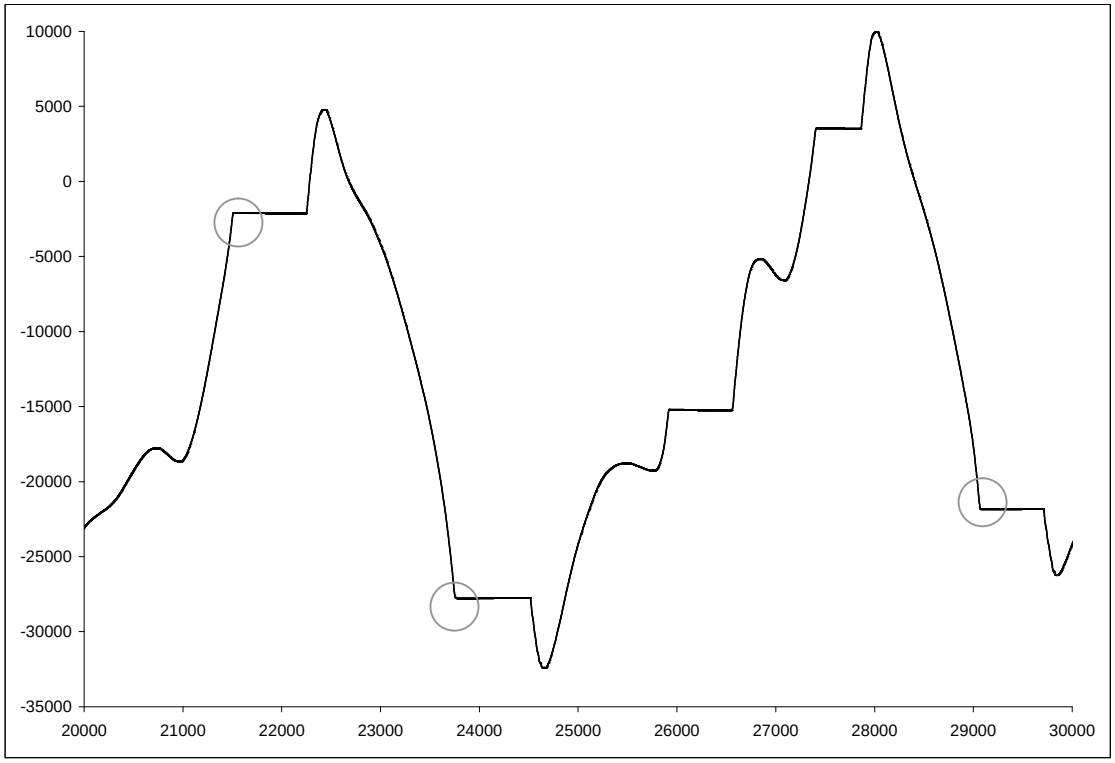

Figure 5.33: Plot of x against time(ms) for Axis 1 (Tilt)

The slope just before the zero gradient is the point just before the visual servoing system fails to track the object. The gradient at this point will be the maximum tracking speed of the system. Using the data generated during the tracking to compute for the maximum gradient, we get the following results in Table 5.2.

Axis	**Maximum Gradient (Pulse/ms)**	**Maximum Angular Speed (degree/s)**
0	53	38.16
1	92	66.24

Table 5.2: Maximum Angular Tracking Speed of Visual Servoing System

The maximum angular tracking speed for the axis 0 (pan) is 38.16 degrees per second while that of axis 1 (tilt) is 66.24 degrees per second. The pan axis has a lower tracking speed. This is expected as it is carrying the mass of the tilting mechanism.

5.3 3-Dimensional Tracking

The visual servoing system developed is capable of tracking moving objects within its pan and tilt range. The system however has no sense of how far away the object is from the robot. A laser-based distance measurement device is mounted on to the robot to add distance sensing capability to the system.

The laser device used is the Disto Pro by Leica. It is mounted with a certain known offset from the intersection of the orthogonal axes of rotation. Control of the Disto Pro is by the RS232 PC interface. The laser beam from the Disto Pro is turned on before the servoing begins. A red laser spot is projected on the target to be track. With a simple thresholding of the red band in the image, the laser spot is extracted and centroid computed using the same method mention in Section 4.2.5. This centroid value is computed for every image frame and used in place of the image optical center for object tracking. The computation of the object centroid remains the same. The laser spot will be servoed till both the laser centroid and the object centroid coinsides. At this point, the distance measurement is taken. With the known mounting offsets of the Disto Pro, the joint angles and the distance measurement, the 3-D Cartesian coordinate of the object with respect to the robot frame can be computed. With the Disto Pro, a virtual translational joint is added to the pan-tilt robot for 3-Dimensional Tracking.

Chapter 6 Conclusion and Recommendation

This chapter provides a summary of the entire dissertation. It summarizes the work done in the project and the results obtained. Section 6.2 proposes several recommendations for future work.

6.1 Conclusion

A modular 2-DOF pan-tilt robot has been designed and built successfully. The design of the modular rotary table can serve as a building block for other multi-axis robot in the future.

A suite of functions has been compiled into a static software library. Future development work based on the STG motion control board can benefit from this software library.

Two robotic visual servoing systems have been developed using the 2-DOF pan-tilt robot. They are namely the Dynamic Position-based Look-and-Move structure and the Image-based Visual Servo structure.

The Dynamic Position-based Look-and-Move system requires calibration in order to determine the object pose with respect to the robot. With the object pose known, the robot is controlled, using trapezoidal point-to-point movement, to its desired position. Motion of the position-based system is not smooth when tracking moving object. This

is inherent in the look-and-move structure because the robot has to come to a complete stop before grabbing a new image buffer and computing the object pose. The system's step response is studied and its look-and-move characteristic is observed from the data collected.

The Image-based Visual Servo system developed does not need calibration. The object pose is not required. The pixel error of the centroid is used directly as the desired position after multiplying by a proportional gain factor. The system's step response is studied with different value of this gain value. A gain value of 0.4 gives a smooth robot motion with reasonably fast response. The tracking performance of the system is measured. It has a maximum angular tracking speed of 38.16 degrees per second and 66.24 degrees per second for the pan and tilt axis respectively. Robot motion when tracking moving object is smooth with no jerky motion as opposed to the look-and-move system.

The Visual Servo structure is the preferred method when visual tracking is concerned. The Visual Servo structure can only be applied when the control signal can be fed to the motor amplifier directly. Unfortunately, most commercial robots have closed architecture with inbuilt joint controllers which disallow this to be done. In such a case, only Look-and-Move structures can be implemented.

Image-based structure makes use of feature information directly as a feedback to the system without the need to compute pose and inverse kinematics. It saves computational time but may not be so much of a concern when computing power is readily available nowadays.

6.2 Recommendation

As the cost of Windows-based real-time software RTX is quite substantial, RTLinux, a free Linux-based real-time software should be evaluated. Future work may include porting of the STG software library from the Windows/RTX Platform to the Linux/RTLinux platform.

More research can also be done in the area of color constancy. Study can be done on how to achieve better color constancy in the chromaticity space and color constancy using neural network.

To further enhance the performance of the Image-based Visual Servoing system, more work could be done in the prediction of the object motion.

The pan-tilt robot can be further developed into a 3D coordinate measurement system. Anti-backlash gears or direct drive can be explored in replacement of the standard worm gears used in the rotary table. Additional work in the calibration of the laser mounted 3D coordinate system would increase the system accuracy in measuring 3D coordinates.

LIST OF REFERENCES

[1] J. Hill et W. T. Park, “Real time control of a robot with a mobile camera.”, Dans *Proceedings of the 9th ISIR*, pages 233-246, 1979.

[2] P. K. Allen, “Real-time motion tracking using spatio-temporal filters”, *Proc. DARPA Image Understanding Workshop,* 1989, pp. 695-701.

[3] J. S. C. Yuan, “A general photogrammetric method for determining object position and orientation”, *IEEE Trans. Robotics and Automation,* vol. 5, no. 2, Apr. 1989, pp. 129-142.

[4] P. K. Allen, B. Yoshimi, and A. Timecenko, “Real-time visual servoing”, *Proc. IEEE Int. Conf. Robotics Automat.,* Apr. 1991, pp. 851-856.

[5] J. T. Feddema, “Real-time visual feedback control for hand-eye coordinated robotic systems “, *thesis submitted to Purdue University,* Aug. 1989.

[6] P. I. Corke, “High performance visual closed-loop robot control”, *thesis submitted to University of 'Melbourne,* Jul. 1994.

[7] A. C. Sanderson and L. E. Weiss, “Image-based visual servo control using relational graph error signals”, Proc. *IEEE, pp.* 1074-1077, 1980.

[8] B. Bhanu et al., “Qualitative target motion detection and tracking”, *Proc. DARPA Image Understanding Workshop,* 1989, pp. 370-398.

[9] P. I. Corke and R. P. Paul, “Video-rate visual servoing for robots”, *Grasp Lab, Dept. ofCompu. And Info. Sci., Univ. of Pennsylvania, Tech. Rep. MS-CIS-89-18,* Feb. 1989.

[10] E. D. Dickmanns and A. Zapp, “Autonomous high speed road vehicle guidance by computer vision”, *Proc. W^{h} IFAC World Congr.,* Jul. 1987.

[11] E. D. Dickmanns, “Computer vision for flight vehicles “, Z *Flugwiss, Weltraumforsch,* vol. 12, pp. 71-79, 1988.

[12] J. T. Feddema and C. S. G. Lee, “Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera”, *IEEE Trans. Syst. Man Cybern.,* vol. 20, no, 5, pp. 1172-1183,1990.

[13] J. T. Feddema, C. S. G. Lee and O. R. Mitchell, “Weighted selection of image features for resolved rate visual feedback control”, *IEEE Trans. Robotics Automat., vol. 7, no.* 1, pp. 31-47, 1991.

[14] J. T. Feddema and O. R. Mitchell, “Vision guided servoing with feature-based trajectory generation”, *IEEE Trans. Robotics Automat.,* vol. 5, no. 5, pp. 691 - 699, 1989.

[15] R. Goldenberg, W. C. Lau, A. She and A. M. Waxman, “Progress on the prototype PIPE”, *Proc. IE*

[16] D. J. Heeger, “Depth and flow from motion energy”, *Science,* vol. 86, pp. 657-663,1986.

[17] B. K. P. Horn and B. G. Schunck, “Dtermining optical flow”, *Artificial Intel!.,* vol. 17pp. 185-204, 1981.

[18] A. E. Hunt and A. C. Sanderson, “Vision-based predictive tracking of a moving target”, *Carnegie Mellon Univ., The Robotic Inst., Tech. Rep. CMU-RI-TR-82-15,* Jan. 1982.

[19] A. J. Koivo and N. Houshangi, “Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller”, *IEEE Trans. Syst. Man Cybern.,* vol. 21, no. 1, pp. 134-142, 1991.

[20] S. W. Lee and K. Wohn, “Tracking moving object by a mobile camera”, *Dept. of Conput. And Info. Sci., Univ. of Pennsylvania, Tech. Rep. MS-CIS-88-97,NQv.* 1988.

[21] R. C. Luo, R. E. Mullen Jr. and D. E. Wessel, “Anadaptive robotic tracking system using optical flow”, *Proc. IEEE Int. Conf. Robotics Automat.,* 1988, pp. 568-573.

[22] D. Tsakiris, “Visual tracking strategies”, *Master's thesis. Dept. of Electrical Eng.,* Univ. of Maryland, College Park, 1988.

[23] L. E. Weiss, A. C. Sanderson and C. P. Neuman, “Dynamic sensor-based control of robots with visual feedback”, *IEEEJ. Robotics Automat.,* vol. RA-3,no. 5, pp. 404-417, Oct. 1987.

[24] D. B. Westmore and W. J. Wilson, “Direct dynamic control of a robot using an end-point mounted camera and Kalman filter position estimation”, *Proc. IEEE. Int. Conf. Robotics Automat.,* 1991, pp. 2376-2384. *EE Int. Conf. Robotics Automat.,* 1987, pp. 1267-1274.