Visual Feedback Stabilisation of a Cart Inverted Pendulum

A Thesis Submitted to the University of Bradford for the Degree of
Doctor of Philosophy in the School of
Electrical Engineering & Computer Science

Stephen Donald INGRAM
BEng (Hons), MPhil, IEng, MIET
2016

Abstract

Keywords: Visual Control, Cart Inverted Pendulum, PID Control, Artificial Neural Networks, Intelligent Algorithms.

Vision-based object stabilisation is an exciting and challenging area of research, and is one that promises great technical advancements in the field of computer vision. As humans, we are capable of a tremendous array of skilful interactions, particularly when balancing unstable objects that have complex, non-linear dynamics. These complex dynamics impose a difficult control problem, since the object must be stabilised through collaboration between applied forces and vision-based feedback. To coordinate our actions and facilitate delivery of precise amounts of muscle torque, we primarily use our eyes to provide feedback in a closed-loop control scheme. This ability to control an inherently unstable object by vision-only feedback demonstrates an exceptionally high degree of voluntary motor skill. Despite the pervasiveness of vision-based stabilisation in humans and animals, relatively little is known about the neural strategies used to achieve this task.

In the last few decades, with advancements in technology, we have tried to impart the skill of vision-based object stabilisation to machines, with varying degrees of success. Within the context of this research, we continue this pursuit by employing the classic Cart Inverted Pendulum; an inherently unstable, non-linear system to investigate dynamic object balancing by vision-only feedback. The Inverted Pendulum is considered to be one of the most fundamental benchmark systems in control theory; as a platform, it provides us with a strong, well established test bed for this research.

We seek to discover what strategies are used to stabilise the Cart Inverted Pendulum, and to determine if these strategies can be deployed in Real-Time, using cost-effective solutions. The thesis confronts, and overcomes the problems imposed by low-bandwidth USB cameras; such as poor colour-balance, image noise and low frame rates etc., to successfully achieve vision-based stabilisation.

The thesis presents a comprehensive vision-based control system that is capable of balancing an inverted pendulum with a resting oscillation of approximately $\pm 1^{o}$. We employ a novel, segment-based location and tracking algorithm, which was found to have excellent noise immunity and enhanced robustness. We successfully demonstrate the resilience of the tracking and pose estimation algorithm against visual disturbances in Real-Time, and with minimal recovery delay. The algorithm was evaluated against peer reviewed research; in terms of processing time, amplitude of oscillation, measurement accuracy and resting oscillation. For each key performance indicator, our system was found to be superior in many cases to that found in the literature.

The thesis also delivers a complete test software environment, where vision-based algorithms can be evaluated. This environment includes a flexible tracking model generator to allow customisation of visual markers used by the system. We conclude by successfully performing off-line optimization of our method by means of Artificial Neural Networks, to achieve a significant improvement in angle measurement accuracy.

Acknowledgements

First and foremost, I would like to thank Almighty God, for giving me the opportunity and the strength to complete this research. Without his endless love, I would not have made it this far.

This thesis would not have been possible without the guidance and support of several individuals, who have extended their valuable time in the preparation and completion of this research.

I would like to express sincere gratitude to my supervisors, Prof. Rami Qahwaji and Prof. Marian Gheorghe, for their invaluable support, encouragement and guidance throughout the course of my PhD study. I would also like to thank my former advisors, Prof. Alamgir Hossain and Prof. Keshav Dahal, for their supervision, insightful comments and never-ending support. I wish to thank the research administration team at the University of Bradford, particularly Ms Rona Wilson and Ms Judy Cowley for their assistance over the years. I would also like to acknowledge the financial support provided by Goodrich Engine Control Systems (now Rolls Royce) and Balfour Beatty Rail Technologies, which made it possible for me to pursue my PhD study in the early years.

Finally, I would like to express heartfelt gratitude to my family for their continuous encouragement, moral support and unconditional love. Without them, this thesis would not have been possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

General Introduction

"*The heights by great men reached and kept were not attained by sudden flight, but they, while their companions slept, were toiling upwards in the night."*

Henry Wadsworth Longfellow, "The Ladder of St. Augustine" (1850)

Humans and animals are capable of a tremendous array of skilful interactions, all of which are the result of internal and external forces acting on the body's musculoskeletal structure. It has been long understood that the somatic nervous system generates internal, voluntary forces through stimulation of muscles, in a process called motor control (Ghez and Krakauer, 2000). External forces are outside influences that act on the body, such as gravity, acceleration and friction, etc.

To coordinate these interactions and ensure delivery of the correct muscle torque, we primarily use our eyes to provide vision-based feedback in a closed-loop control scheme. While performing these interactions, we often encounter highly unstable objects that have non-linear dynamics. These complex dynamics impose a difficult control problem, since the object must be stabilised through the interaction between applied forces and vision-based feedback. An example would be a wait-person balancing a tray of drinks; navigating around a busy room, while trying not to spill the contents. This seemingly trivial task requires precise, Real-Time control, and is one that demonstrates an extremely high degree of voluntary motor skill.

1.1 Motivation

Dynamic object balancing is one of the most complex physical interactions that we perform as humans; however, we know relatively little about the neural strategies used to achieve stabilisation using vision-only feedback (Burdet et al., 2013), (Milner et al., 2006). This knowledge gap has resulted in a number of on-going investigations within the fields of Control Engineering and Robotics research, where such a skill is highly desirable.

For decades, the problem of balancing an Inverted Pendulum has been widely used as a benchmark for testing the efficacy of various control schemes (Kurdekar and Borkar, 2013). The Inverted Pendulum represents the basis of many complex systems, and it is deemed to be one of the most important test beds in the fields of Control Engineering and Robotics research (Boubaker, 2014). Using the Inverted Pendulum as a test platform, we explore the following questions and use them to form the basis of our research:

- In a closed-loop control scheme, what strategies are used to balance an unstable object using vision-based feedback? And how are these strategies implemented in the real-world (i.e. outside of simulation)? Can these strategies be used to stabilise an Inverted Pendulum?

- Visual feedback stabilisation is state-of-the-art in the field of Control Engineering, and as a consequence, expensive control systems are widely accepted (Chavez-Romero, 2015). Can we develop a cost-effective system using off-the-shelf (OTS) components or 3D printable parts?

- The answer to the previous question will depend on the performance of the cost-effective solution. Our final questions are: How well do these strategies perform as compared to other visual feedback approaches? What methods are used to evaluate vision-based Inverted Pendulum systems and can we achieve improved performance with a cost-effective solution?

In the context of this research, we employ the classic Cart Inverted Pendulum to investigate vision-based dynamic object balancing, to address the aforementioned questions. Essentially, the Cart Inverted Pendulum comprises of an unrestricted cart and a weighted pole. The pole is fixed on a pivot, which can rotate in a single dimension. The cart is driven by force in such a way as to keep the pole in an upright position at all times (Ponce et al., 2014). Although this is a simple system from a construction point of view, it has three interesting properties that make it attractive to control engineers and robotics researchers:

1. It is highly unstable – The inverted position is the point of unstable equilibrium, as can be seen from pole-zero plots (Bernard, 1987).

2. It is highly non-linear – The dynamic equations consist of non-linear terms.

3. It is underactuated – The system has two degrees of freedom (2-DOF) but only one actuator. This property helps to make the system cost-effective, but the control problem becomes more challenging.

To achieve stabilisation, the cart position must be tightly regulated by applying frequent corrective forces via the motor, thus balancing the pole and maintaining its upright position. This task is known as the stabilisation problem, and is the most widely explored challenge posed by the Inverted Pendulum.

One significant aspect of the Inverted Pendulum is that it represents the basis of many complex systems; such as Motion Gait Modelling, Satellite Attitude Control, Rocket Stabilisation and Multi-Rotor Vehicle Dynamics, as described by Boubaker (2014), a prominent scholar in this field.

Some real-world examples of Inverted Pendulum systems are illustrated in Figure.1.1:

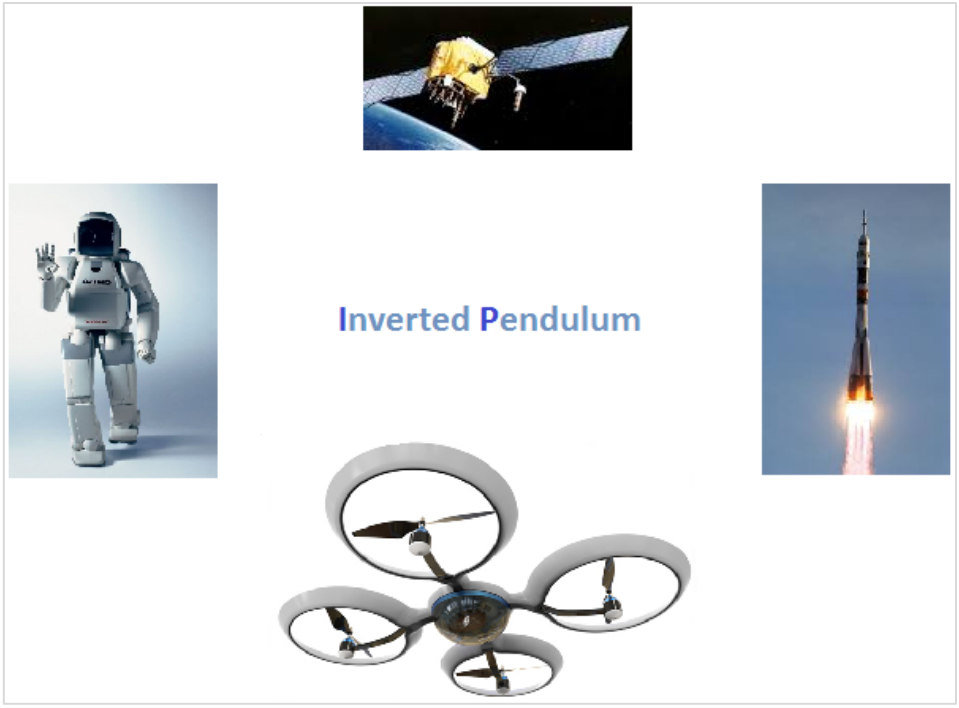


Figure.1.1 *– Real-world examples of Inverted Pendulum systems. Clockwise from the top: Satellite attitude control, rocket stabilisation, multi-rotor vehicle dynamics and robot walking.*

1.2 Aim and Objectives

The aim of this research is to build a comprehensive and effective method of dynamic object balancing by means of vision-only feedback, and to optimise the method by using Artificial Neural-Networks. The algorithms outlined in the proposed method must allow for Real-Time operation, even in the presence of visual disturbances. To achieve this aim, the following core objectives were pursued:

1. Gain a comprehensive understanding of the classic Cart Inverted Pendulum; its dynamics, and the traditional methods used to achieve stabilisation.

2. Review the state-of-the-art in vision-based Inverted Pendulum control; covering hybrid (sensor fusion), vision-only, and “intelligent” approaches.

3. Develop a high-performance; cost-effective test bed to allow algorithms to be applied to the Cart Inverted Pendulum stabilisation problem.

4. Perform data harvesting to establish a baseline to which our method can be evaluated.

5. Formulate a novel, vision-based control algorithm for pose (position and orientation) estimation and apply it to the Cart Inverted Pendulum stabilisation problem.

6. Explore the use of Artificial Neural Networks for optimisation and demonstrate their merit in our application.

The scope of this research is limited to the Cart Inverted Pendulum stabilisation problem only. The position control and swing-up problems (Durand, 2013), (Merakeb et al., 2013), (Lam, 2004) are not addressed in this work.

1.3 Main Challenges

In this research, we focus on the most challenging control task posed by the Cart Inverted Pendulum, which is known as the stabilisation problem (Durand, 2013). We compound this challenge by seeking to balance the Inverted Pendulum using vision-only feedback, which is both complex and computationally demanding. The inherent

instability of the system poses a significant challenge; a robust, Real-Time controller is required to achieve stabilisation. The main difficulties that were encountered during this research are summarised in the following subsections:

1.3.1 Observing Bandwidth Constraints

One of the main challenges encountered in this research was the strict bandwidth constraints of the system, which required negotiation in order to achieve Real-Time performance. In this context, bandwidth refers to the time available to conduct image acquisition, image processing and control signal output. In vision-based control, many tasks are required to be performed, including filtering, feature detection, correlation, etc. The fusion of complex vision processing tasks within closed-loop control dictates the complexity of the controller. A flexible algorithm was therefore required that allowed the compromise between speed and estimation error. Understanding this compromise was central to our research. High-speed image acquisition, efficient processing tasks and high-speed communications were all key bandwidth challenges.

1.3.2 Achieving Robust Vision-Based Pose Estimation

Achieving robust, vision-based, pose estimation is extremely challenging; it is, however, widely accepted that many classical vision processing tasks can be solved effectively, particularly in the automotive domain (Fu et al., 2015). In some cases, the solution requires ideal image sequences, high bandwidth and even a hybrid approach. Images acquired by low-cost USB cameras, in poor lighting conditions are less than ideal for the task. Inferior colour balance, sensor noise, and other artefacts render

classical image processing methods inadequate in this context. As a consequence, more robust methods for pose estimation needed to be developed.

1.3.3 Developing a Cost-Effective Test Bed

A high-performance test bed can be achieved by employing expensive hardware resources. We considered that an essential requirement should be a low barrier to entry, with regards to cost. This was mainly to encourage the number of researchers that could reproduce our work. The test bed should, therefore, make efficient use of OTS components. One of the main challenges of this research was achieving high-performance operation, while observing tight cost constraints. In this context we define high-performance as a level that is equivalent to or better than that achieved by expensive commercial solutions aimed at research.

1.4 Prototype System Overview

The prototype Cart Inverted Pendulum system that was developed during the course of this research is illustrated in Figure.1.2. A permanent magnet D.C. (PMDC) motor was used in conjunction with stainless steel rods and bushings to form the basis of the linear sliding cart mechanism. A third stainless steel rod was used to form the pendulum shaft. This rod was attached by a coupling arrangement to a linear potentiometer, which was used as a conventional means of measuring angular displacement and angular velocity during testing. The control interface was realised using a PIC32-Pinguino, a 32-Bit microcontroller based development board from OLIMEX Ltd (see Table.1.1 for the key specifications).

Figure.1.2 *– An Overview of the Closed-Loop Cart Inverted Pendulum System.*

Parameter	Value
Max speed	80Mhz
Flash memory	512KB
RAM	32KB
USB	OTG/2.0
A/D Channels/Resolution	16/10-Bits
A/D Sample Rate	1MSPS
PWM Channels/Resolution	5/8-Bits
I/O Pins	64

Table.1.1 *– Main PIC32-Pinguino Specifications.*

1.5 Approach

A continuous sequence of still images is captured at the maximum frame rate of 30fps by the USB 2.0 camera, which is trained at the Cart Inverted Pendulum apparatus. The positioning of the camera is such that the entire pendulum (including the Cart and Pole) is encompassed by the 75º Field Of View (FOV) of the camera. Although the USB 2.0 camera meets the low-cost requirement, it presents several limitations. The shutter speed and gain controls are two such examples, which are automatic and offer no software access. These limitations result in occasionally blurred and saturated images that required accommodation by the system.

The USB camera offers a wide angle, manual focus lens, which was ideal for capturing images with the greatest depth range possible. This type of lens is essential when trying to obtain pendulum pose using a single camera system (Hespanha, 2000). A typical image from the camera is shown in Figure.1.3; the poor colour balance and irregular lighting conditions should be noted.

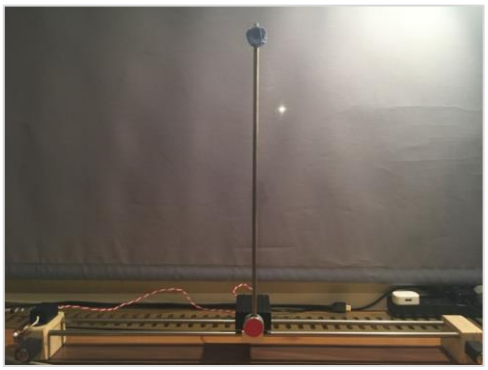


Figure.1.3 *– A typical image captured by the Inverted Pendulum camera – Note the poor colour balance and irregular lighting conditions.*

A Laptop computer was used to perform image acquisition and execution of the vision-based control algorithms. The main specifications of the Laptop are given in Table.1.2 (this machine was used throughout unless otherwise stated). The algorithms were implemented in Microsoft Visual Basic .NET (VB.NET) 2008 using the EmguCV library (EmguCV, 2010), which is a VB.NET wrapper of the popular OpenCV library (Uke, 2013), (Bradski and Kaehler, 2008).

The vision-based control algorithms were found to be computationally expensive and in order to achieve Real-Time performance, a multi-threaded architecture was used to share the load between several processors; this concept is discussed further in Chapter 4. The vision-based tracking algorithm presented in Chapter 5 generates a suitably robust estimate of the pendulum's pose. This estimate is pre-processed and passed to the motion control algorithm, which then computes the required motor speed and direction parameters (i.e. velocity). A bespoke communications protocol was developed to allow efficient transfer of data between the Laptop computer and the motor controller, via a USB 2.0 interface.

Parameter	Value
Model	Hewitt Packard (HP) Envy
Cores/Threads	2/4
Processor Type	Intel® Core™ i7-6500U
Processor Speed	2.6 GHz
Memory	12.0 GB
Graphics Card	Intel® HD Graphics 520
Operating System	Windows 10™ Home Edition

Table.1.2 *– Main Laptop Computer Specifications.*

1.6 Major Contributions

The main contributions of this research are as follows:

1. The development of a high-performance, cost-effective control system to facilitate research into vision-based control of the classic Cart Inverted Pendulum. As previously stated, in this context we define high-performance as a level which is equivalent to, or better than that achieved by commercial solutions aimed at research. This development was necessary as commercial Inverted Pendulum systems were prohibitively expensive.

2. Experimentation using the developed control system to establish a baseline of a classical feedback approach. This work provided us with the required numerical data to perform analysis.

3. The proposal of novel, Real-Time algorithms for pendulum localisation and pole angle estimation using vision-only feedback. These algorithms are an original contribution to knowledge.

4. Application of the proposed algorithms for stabilising a real-world Inverted Pendulum using vision-only feedback and the appraisal thereof. This work is an original contribution to knowledge.

5. A proposed enhancement to the vision-based control algorithm by means of Artificial Neural Networks. This work is an original contribution to knowledge.

1.7 Thesis Overview

The vision-based control theory presented in this thesis is applied to the task of stabilising the classic Cart Inverted Pendulum, an inherently unstable, non-linear system. The test bed summarised in Section 1.4 was used throughout to undertake the investigations.

Chapter 1 has provided a general introduction to the research area, including the motivation, aim and objectives, main challenges and significant contributions. Chapter 2 provides a theoretical foundation on which the solution is built; some of the more specific background knowledge (particularly on image processing) is presented within the relevant chapter. Chapter 3 delivers a literature review of research that is relevant to the work presented in this thesis. We adopt a broad approach to the literature survey in an effort to answer some of the fundamental questions posed in Section 1.1. Chapter 4 describes, in detail, the development of our test bed and provides technical justification for the main components used. Chapter 4 also presents a summary of the results gathered during the practical experimentation phase. These results were used to establish a baseline, to which we could compare our vision-based approach. Chapter 5 provides details of the Inverted Pendulum localisation algorithm, which uses our Multiple-Segments Moments Tracking (MSMT) method. Chapter 6 optimises the solution presented in Chapter 5, using Artificial Neural Networks to perform off-line pendulum angle estimation. Chapter 7 summarises the contributions of the thesis and suggests areas that require further investigation.

Chapter 2

Background

2.1 The Cart Inverted Pendulum

For decades, the problem of balancing an Inverted Pendulum has been widely used as a benchmark for testing the efficacy of various control schemes (Kurdekar and Borkar, 2013). The Inverted Pendulum represents the basis of many complex systems, and it is deemed to be one of the most important examples in the fields of control engineering and robotics research (Boubaker, 2014). Various Inverted Pendulum structures have been developed and controlled in the literature; however, owing to its physical suitability, we explore the classic Cart Inverted Pendulum.

The Cart Inverted Pendulum consists of a pole of length $\boldsymbol{l}$, weighted by a mass of $\boldsymbol{m}$, balancing on a mobile cart. The cart is driven by an external force $\boldsymbol{F}$, in such a way as to keep the pole in an upright position at all times. Typically, sensors are used to measure the angular displacement $\boldsymbol{\theta}$ and the angular velocity $\boldsymbol{\omega}$ of the pole. The input force $\boldsymbol{F}$ is a function of these parameters and is used to balance the pole by maintaining $\boldsymbol{\theta}$ at zero; this balancing act is known as the stabilisation problem. Prior to designing a suitable controller, it is important to derive and understand the equations of motion for the Cart Inverted Pendulum.

2.1.1 Cart Inverted Pendulum Dynamic Equations

In this section, the dynamic equations for the Cart Inverted Pendulum are derived from Newton's second law of motion, and also by Lagrangian mechanics. The system has Two Degrees of Freedom, the linear motion of the cart in the X-axis and the rotational motion of the pendulum in the X-Y plane; thus, there will be two dynamic equations. A schematic representation of a typical Cart Inverted Pendulum is shown in Figure.2.1. The parameters that are relevant to the derivation of the dynamic equations are given in Table.2.1.

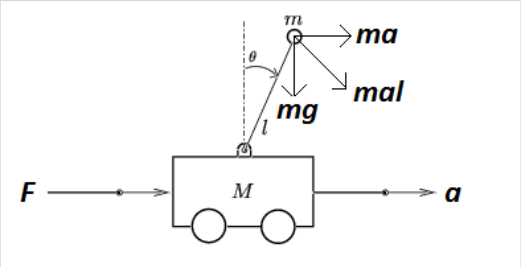


Figure.2.1 *– Schematic Representation of the Cart Inverted Pendulum.*

Symbol	Quantity
M	Mass of the pole
M	Mass of the cart
L	Length of the pole
G	Acceleration due to gravity
X	X coordinate or position of the cart
V	Translational velocity of the cart
A	Translational acceleration of the cart
N	Horizontal relation force between the cart and the pole
F	External force applied on the cart
Θ	Vertical angle of the pole
ω	Angular velocity of the pole
A	Angular acceleration of the cart

Table.2.1 *– Physical Parameters of the Cart Inverted Pendulum.*

A. Newtonian Method

The application of Newton's second law of motion is one of the most fundamental methods used to derive the dynamic equations of a system (Bernard, 1987). Mathematically, this is expressed as follows:

$$\sum F = Ma \tag{2.1}$$

By applying equation (2.1) to the individual parts of the system, namely the cart and the pole, we get the corresponding equations:

$$F + N = Ma \tag{2.2}$$

And:

$$N = m\omega^2 l sin\,\theta - ma - m\alpha l cos\,\theta \tag{2.3}$$

The horizontal relation force between the cart and the pole, ***N***, must be eliminated; therefore, by simplifying equation (2.3) we obtain:

$$F = (m + M)a - m\omega^2 l sin\theta + m\alpha l cos\theta \tag{2.4}$$

The total external torque acting on the mass (***m)*** about the pivot point (between the cart and the pole) is $mglsin\theta$, and the moment of inertia of ***m*** about the same point is ml^2. By applying $\sum \tau = I\alpha$ about this point of contact, we obtain the following equations:

$$mglsin\theta = ml^2(\alpha + acos\theta) \quad (2.5a)$$

$$mgsin\theta = m\alpha l + malcos\theta \quad (2.5b)$$

$$gsin\theta - \alpha l - acos\theta = 0 \quad (2.5c)$$

Equations (2.4) and (2.5c) are adequate to describe the dynamics of the Cart Inverted Pendulum, as only two equations are required to solve for the unknown parameters $\boldsymbol{a}$ and $\boldsymbol{\alpha}$. Also, the internal relation force, ***N***, has been eliminated.

B. Lagrangian Method

The Lagrangian approach is another popular method used to derive the dynamic equations of a system. This approach allows one to deal with scalar energy functions rather than vector forces and accelerations, as is the case with the Newtonian method; thereby reducing the opportunities for error. Also, in many complex cases, such as the Cart Inverted Pendulum, the Lagrangian approach is typically simpler than the Newtonian method (Bernard, 1987).

The physical parameters used in the Lagrangian approach are identical to those used in the Newtonian method (see Table.2.1); excluding the horizontal relation force, which is not required in this approach. The initial step in deriving the equations of motion is to derive the Lagrangian $\boldsymbol{L}$:

$$L = T - V \quad (2.6)$$

Where the parameters $\boldsymbol{T}$ and $\boldsymbol{V}$ are the total Kinetic and potential energies of the system, respectively. We start by determining the total kinetic energy of the system at

a given time. The Cartesian coordinates of ***m*** at any given time would be $(x + lsin\theta)$ and $(lcos\theta)$ respectively. The ***X*** and ***Y*** components of the velocity of ***m*** can be determined by means of differentiating the position coordinates of ***m***. Therefore, $V_x = (\dot{x} + l\dot{\theta}cos\theta)$ and $V_y = (-l\dot{\theta}sin\theta)$. The total kinetic energy of the system, ***T***, can therefore be written as:

$$T = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m[(\dot{x} + l\dot{\theta}cos\theta)^2 + (-l\dot{\theta}sin\theta)^2] \tag{2.7}$$

Using the cart as a ground level reference, and **m** to be the only mass above this reference point, then the total potential energy of the system, ***V***, can be written as:

$$V = mglcos\theta \tag{2.8}$$

Equations 2.7 and 2.8 can be used to determine the Lagrangian:

$$L = \frac{1}{2}(M + m)\dot{x}^2 + ml\dot{x}\dot{\theta}cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 - mglcos\theta \tag{2.9}$$

As previously mentioned, the Cart Inverted Pendulum has two degrees of freedom and the cart force, ***F***, is the only external force applied that can change ***x*** or $\dot{\boldsymbol{x}}$; no other external force or torque can affect $\boldsymbol{\theta}$ directly. The equations of motion for the Cart Inverted Pendulum can, therefore, be derived by simplifying the following:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F \quad \text{and} \quad \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0 \tag{2.10}$$

The first equation of motion can be found by simplifying the following:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = F \tag{2.11}$$

$$\frac{d}{dt}\left[(M+m)\dot{x} + ml\dot{\theta}cos\theta\right] = F \tag{2.12a}$$

$$(M+m)\ddot{x} + ml\ddot{\theta}cos\theta - ml\dot{\theta}^2 sin\theta = F \tag{2.12b}$$

$$F = (m+M)a - m\omega^2 lsin + m\alpha lcos\theta \tag{2.12c}$$

The second equation of motion can be found by simplifying the following:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0 \tag{2.13}$$

$$\frac{d}{dt}\left[ml\dot{x}cos\theta + ml^2\dot{\theta}\right] - \left[-ml\dot{x}\dot{\theta}sin\theta + mglsin\theta\right] = 0 \tag{2.14a}$$

$$ml\left(\ddot{x}cos\theta - \dot{x}\dot{\theta}sin\theta + l\ddot{\theta} + \dot{x}\dot{\theta}sin\theta - gsin\theta\right) = 0 \tag{2.14b}$$

$$gsin\theta - \alpha l - acos\theta = 0 \tag{2.14c}$$

Equations 2.4 and 2.5c are the same as 2.12c and 2.14c respectively, thereby verifying the equations of motion for the Cart Inverted Pendulum system. These equations are not explicitly defined in terms of $\boldsymbol{a}$ and $\boldsymbol{\alpha}$, and hence need to be solved simultaneously. By solving we obtain:

$$a = \frac{m\omega^2 lsin\theta - mgsin\theta cos\theta + F}{msin^2\theta + M} \tag{2.15}$$

$$\alpha = \frac{(m+M)gsin\theta - m\omega^2 lsin\theta cos\theta - Fcos\theta}{Ml + mlsin^2\theta} \tag{2.16}$$

2.2 Proportional Integral Derivative Control

In this section, we present the theoretical and analytical foundation for Proportional Integral Derivative (*PID*) control. *PID* control is undoubtedly the most commonly used method of applying feedback in natural and human-made systems (Åström, 2002). This type of control is used to regulate the output of a system to that of a given set-point. The theoretical basis for the operation of these controllers was first described by Maxwell (1868); however, it was not until the early twenties that *PID* controllers were first developed using theoretical analysis (Minorsky, 1922). Since then, many examples can be found in the literature, where they have been used to solve a wide range of control problems, including the Inverted Pendulum stabilisation problem.

2.2.1 Negative Feedback

PID control is an application of negative feedback. The concept of negative feedback is a simple but powerful tool within control engineering. Negative feedback can reduce the effects of perturbations and can render a system insensitive to external variations, i.e. it can achieve stability. A block diagram of a negative feedback control loop is illustrated in Figure.2.2. The system has two main components, the process (system) and the controller. The process has a single input, which is the manipulated variable, also known as the control variable. It is denoted by the symbol $\boldsymbol{u}$. The control variable influences the process by means of an actuator, which, in our case, is a D.C. motor.

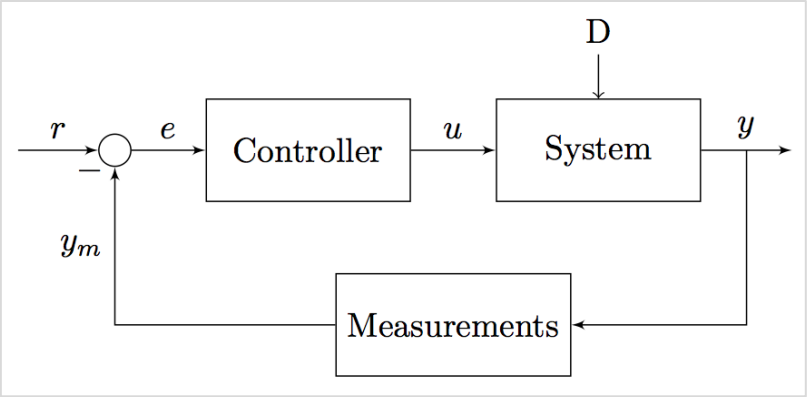


Figure.2.2 *– Block Representation of a Negative Feedback Control Loop.*

The process output, which is also known as the process variable, is denoted by the symbol ***y***. This variable is measured directly (or indirectly) by a sensor to produce $\boldsymbol{y_m}$. With regards to Figure.2.2, the actuator and the sensor are considered to be part of the block labelled "System". The desired value of the process variable is known as the set-point or the reference value and is denoted by **r**. The control error ***e*** is the difference between the set-point and the process variable, i.e. $e = r - y_m$ (Åström, 2002). Widespread application of the negative feedback principle has resulted in significant breakthroughs in the field of Control Engineering.

If the system to be controlled exhibits an adequate output response with Proportional (*P*), Proportional-Integral (*PI*) or Proportional-Derivative (*PD*) only control, then there is little point implementing the full *PID* algorithm. When using *P*, *PI* or *PD* only control, the implementation of the algorithm will be kept to a minimum. We shall, therefore, discus the characteristics of each term individually.

2.2.2 Proportional Term

The proportional term produces a system control variable that is directly proportionate to the error. Using the *P* term in isolation is the simplest form of intermediate value control; however, it gives rise to a fixed offset error, except when the system control input is at zero, and the system process value equals the desired set-point (Atmel, 2016). As depicted in Figure.2.3, a stationary error arises in the system process value directly after a change in the desired value.

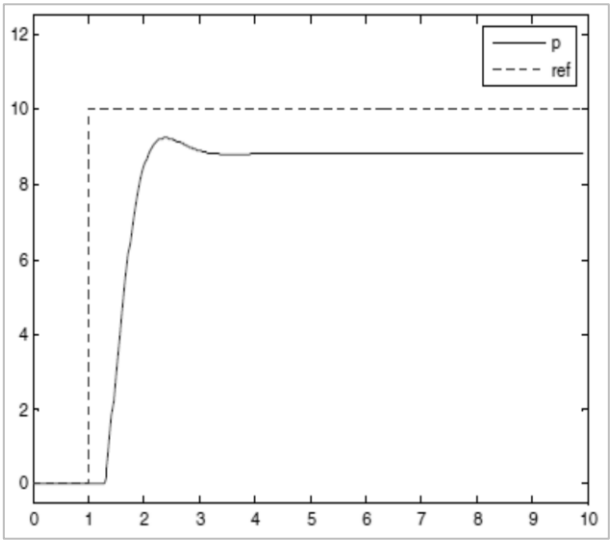


Figure.2.3 *– Step Response of the P-only Controller.*

Care should be taken when employing *P*-only control. Since the control signal is directly proportional to the error, an excessively large gain (K_p) gives rise to an unstable system. This can be seen in the following equation:

$$u = K_p(y_{sp} - y) = Ke \tag{2.17}$$

2.2.3 Integral Term

As previously mentioned, proportional only control has the disadvantage that the process variable often deviates from the set-point by a constant offset. The integral term contributes an additional amount to the system control input from the sum of the previous errors. The accumulation of the error will continue until the system process value equals the desired set-point, and therefore results in zero offset error. The most common use of the *I* term is in conjunction with the *P* term, to form a *PI* controller. Using the *I* term in isolation simply forms a low-pass filter, which gives rise to a slow-to-respond and often unstable system. Figure.2.4 shows the step responses of the *I* term and *PI* controller.

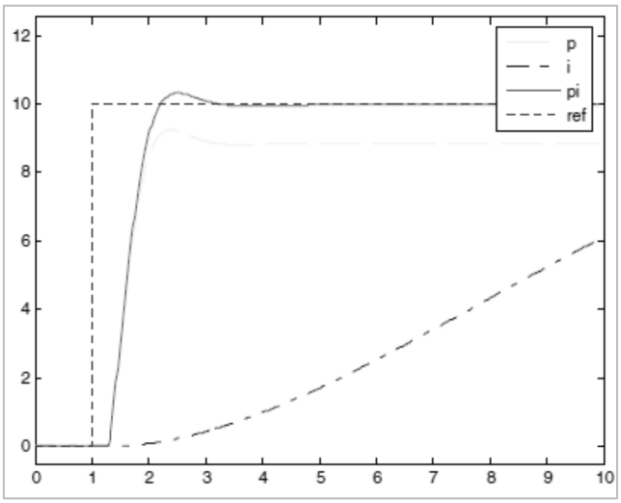


Figure.2.4 *– Step Response of the* P *term, I term and PI Controller.*

As depicted in Figure.2.4, the *PI* controller response has no fixed error and the *I* term response is extremely slow. The *I* term can be represented mathematically as follows:

$$u(t) = K_i \int_0^t e(\tau)d\tau \qquad (2.18)$$

Where K_i is the integral gain. Assuming that there is a steady state, with a constant error e_0 and a constant control signal u_0; then it follows from the above equation that:

$$u_0 = k_i e_0 t \tag{2.19}$$

Since u_0 is a constant, it follows that e_0 must be zero. This is also true for the *PI* controller:

$$u(t) = Ke(t) + K_i \int_0^t e(\tau) d\tau \tag{2.20}$$

This type of controller is extremely common and very capable.

2.2.4 Derivative Term

The derivative term contributes an additional amount from the rate of change of the error in the system. A rapid shift in error results in a significant contribution to the control input. This characteristic improves the response to a sudden change in the system state or reference value. The *D* term is typically used with *P* or *PI* to form a *PD* or *PID* controller respectively. An excessively large *D* term usually gives rise to an unstable system. Figure.2.5 depicts the *P*-only, *PI* and *PID* controller responses.

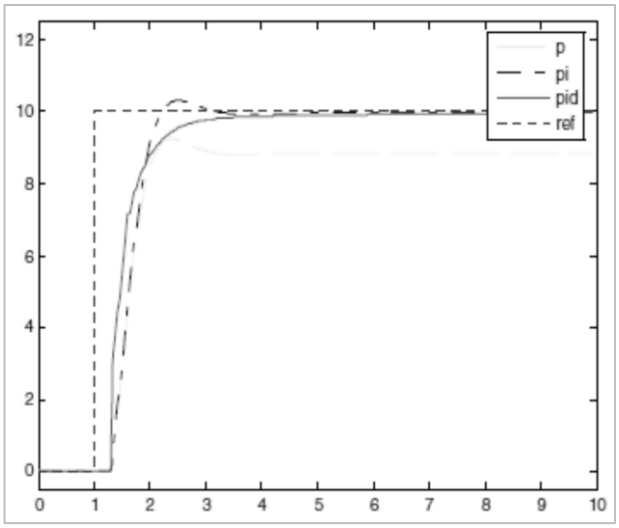


Figure.2.5 *– Step Response of the P only, PI and PID Controller.*

The response of the *PID* controller is superior as compared to *P*, *PI* or *PD* only control. The PID controller can be expressed by the following equation:

$$u(t) = Kp(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}) \quad (2.21)$$

Where the parameter ***u*** is the control signal and ***e*** is the system error. The control action is thus a sum of three terms, representing the past (by the integral), the present (by the proportional), and the future (by the derivative). This temporal relationship is illustrated in Figure.2.6.

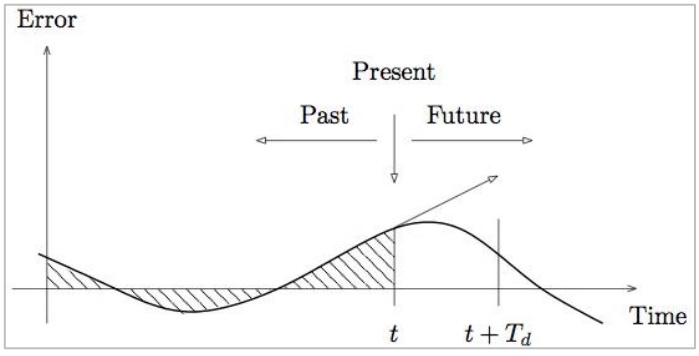


Figure.2.6 *– Temporal Relationship between PID Terms.*

The main parameters of the controller are the proportional gain (*Kp)*, integral time (*Ti)*, and derivative time (*Td)* (Åström, 2002). As we have seen in Figure.2.5, the *PID* controller offers the best system response and is therefore explored in our research.

2.2.5 Integrator Windup

Within a closed-loop control scheme, there are usually nonlinear phenomena that must be accommodated. These are typically limitations in the actuators; for example, a motor has limited rotational speed, and a valve can only be opened or closed, etc. For a system that operates over a wide range of conditions, it may be the case that the controller output reaches the actuator limits. When this occurs, the feedback loop is broken, and the system operates in open-loop mode; as the actuator will remain at its limit independent of the process output for as long as it remains saturated. The integral term will continue to accumulate since the error is typically zero. Both the integral term and the *PID* controller output could become very large. The control signal will inevitably remain saturated, even when the error changes. It may, therefore, take a considerable amount of time for the integrator and the controller output to return inside the saturation range. We refer to this situation as integrator windup.

2.2.6 Controller Tuning

The aim of controller tuning is to obtain the optimum parameters necessary to achieve a fast response time and good system stability. These two requirements are, however,

mutually exclusive; i.e. a quick response time yields poorer stability and vice versa. This concept is illustrated in Figure.2.7.

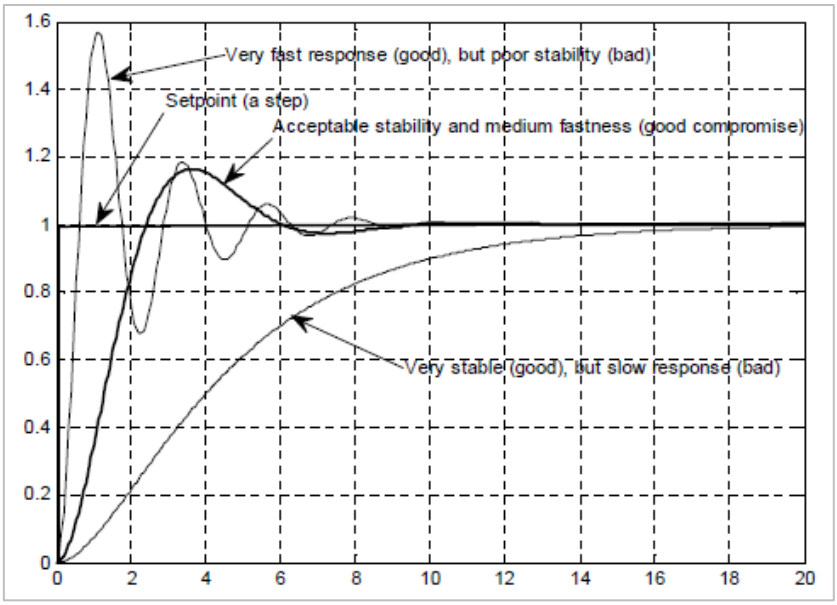


Figure.2.7 *– Mutually Exclusive PID Requirements.*

Ziegler and Nichols published a paper (Ziegler and Nicholas, 1942) describing techniques for *PID* controller tuning. Their concept was to tune a controller based on the following idea: perform an experiment, extract some simple features of process dynamics from the experimental data, and determine the controller parameters from them. The integral gain and derivative gain are both set to their lowest values, and the proportional gain is increased until the system begins to oscillate. The value of K_p at the point of instability is called K_{MAX} and the frequency of oscillation is f_O. The proportional gain is then reduced by a predetermined amount and the integral and derivative gains are set as a function of f_O. A flow chart of this procedure is given in Figure.2.8. We employ this well-established method to tune our PID controller.

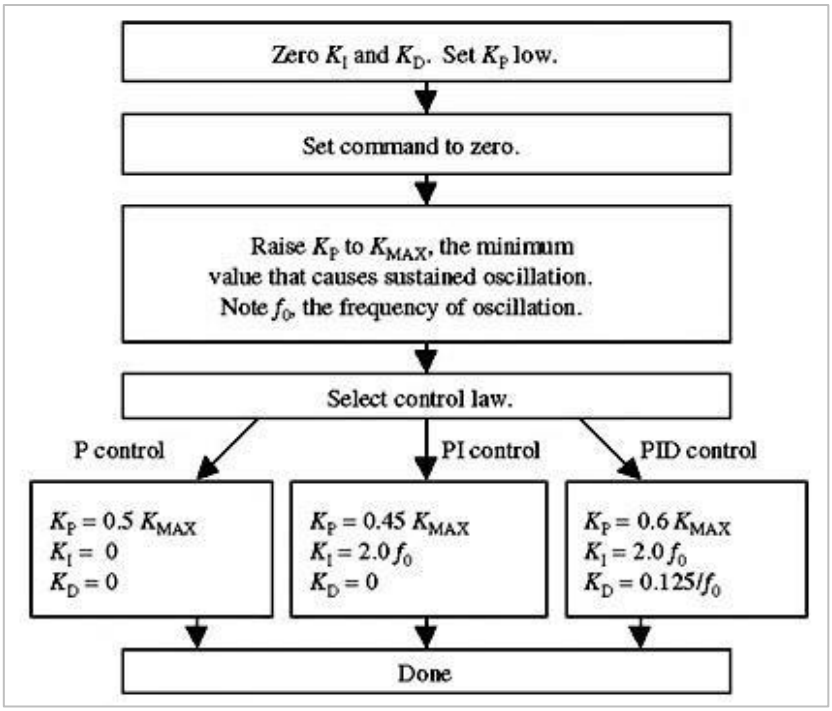


***Figure.2.8** – Ziegler-Nicholas Tuning Procedure.*

2.3 Artificial Neural-Networks

Artificial Neural-Networks (*ANNs*) (Rosenblatt, 1958) and more specifically multi-layer perceptrons (*MLPs*) (Rumelhart et al., 1986) have been studied for decades. They are designed to model biological networks such as the brain. Like the brain, these models can solve complex tasks such as classification, function approximation and optimisation. These models are composed of simple Processing Elements (*PEs*), which can exhibit extremely complex global behaviour, which is determined by the connections between PEs and other network parameters. These simple PEs and interconnecting weighted links (which are analogous to biological neurones and synapses respectively) have been shown to solve complex, non-linear problems. Each *PE* operates independently in parallel, summing all of its weighted inputs and

applying the result to an activation function. Mathematically a PE can be expressed as follows (Hassoun, 1995):

$$y(x) = g(\sum_{i=0}^{n} W_i X_i) \tag{2.22}$$

Where ***x*** is a neurone with ***n*** inputs ($\boldsymbol{x}_0 \ldots \boldsymbol{x}_n$), a single output (***y(x)***) and ***n*** link weights ($\mathbf{w}_0 \ldots \mathbf{w}_n$), which determine the level of gain or attenuation to be applied to a particular input. **g** is an activation function that determines how strong the output should be, based on the sum of the weighted inputs. For many applications, the activation function takes the form of a sigmoid curve, defined as:

$$g(x) = \frac{1}{1+e^{-s(x+t)}} \tag{2.23}$$

Where **s** is the steepness factor of the curve and **t** is a value which causes a shift away from zero. This type of function allows the output of the neurone to change continuously for any value between 0 and 1.

2.3.1 Artificial Neural-Network Cost Function

When executing an ANN, there will inevitably be some cost value associated with that execution. It is clear that equation (2.22) must be resolved for each neurone in the network, i.e. a multiplication and an addition per connection; also, we must resolve the activation function for each neurone. This gives the following time cost:

$$T = cA + (n - n_i)G \tag{2.24}$$

Where $\boldsymbol{c}$ is the number of interconnections, $\boldsymbol{n_i}$ is the number of inputs, $\boldsymbol{n}$ is the number of neurones, $\boldsymbol{A}$ is the cost of calculating the sum of the weighted input, $\boldsymbol{G}$ is the cost of the activation function, and $\boldsymbol{T}$ is the overall cost. Single layer networks are not very useful; PEs are usually arranged to form multi-layered networks, as shown in Figure.2.9.

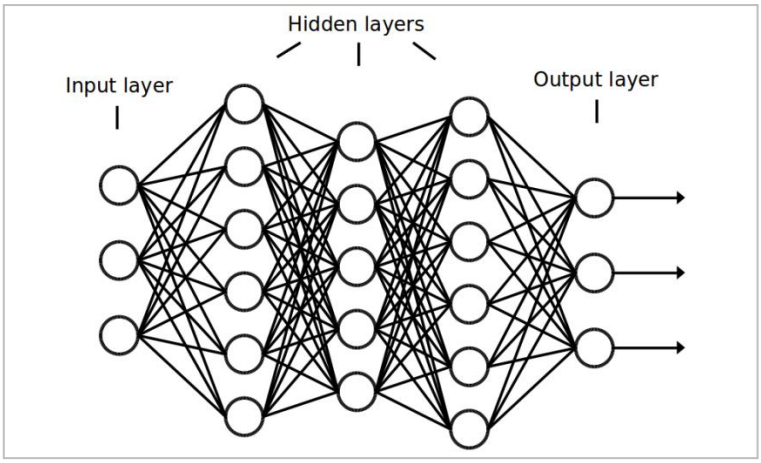


***Figure.2.9**- Multi-Layer Perceptron (MLP) Model.*

When considering a fully interconnected, multi-layered network, equation (2.24) can be re-written as:

$$T = (l-1)(n_i^2 + n_i)A + (l-1)n_l G \qquad (2.25)$$

Where $\boldsymbol{l}$ is the number of layers and $\boldsymbol{n_l}$ is the number of neurones in each layer. This computational cost equation can be used to make quantitative performance comparisons between ANNs and traditional computational methods.

2.3.2 Artificial Neural-Network Learning

Before an ANN can be successfully applied, it must first learn to solve a given problem. There are several learning paradigms; each corresponds to a particular abstract learning task.

- **Supervised learning**: where the algorithm generates a function that maps inputs to desired outputs.

- **Unsupervised learning**: where the algorithm generates a set of input data since examples are not available.

- **Reinforcement learning:** where the algorithm learns how to act given an observation of the real-world. Every action has some impact on the environment, and the environment provides feedback that guides the learning algorithm.

A suitable training method for the applications proposed in this research is supervised learning since example training data will be available for the network. One of the most popular supervised learning algorithms used is the back-propagation algorithm. Essentially, an error is calculated and propagated back through the network while the weights are adjusted to reduce the mean square error (MSE). Firstly, the input is propagated through the network using equation (2.22). The error $\mathbf{e_k}$ of a single output neurone **k** can be defined as:

$$e_k = d_k - y_k \tag{2.26}$$

Where the parameter $\mathbf{y_k}$ is the calculated output and $\mathbf{d_k}$ is the desired output of the neurone **k**. This error value is used to calculate $\boldsymbol{\delta_k}$, which is used for adjusting the weights. $\boldsymbol{\delta_k}$ is defined as:

$$\delta_k = e_k g'(y_k)$$

(2.27)

Where the parameter **g'** is derived from the activation function **g(x)**. δ_j of previous layers can be defined as:

$$\delta_j = \eta g'(y_j) \sum_{k=0}^{K} \delta_k w_{jk} \tag{2.28}$$

Where the parameter **K** is the number of neurones in the layer and **η** is the learning rate, which determines how much the weight should be altered. ANNs are truly powerful tools for classification, function approximation and optimisation. In our research, we considered ANNs for control optimisation of our vision-based method.

Chapter 3

Related Work

In this chapter we present a survey of previous work relating to visual stabilisation of the Cart Inverted Pendulum, an application of Vision-based Control. Section 3.1 opens with a historical overview of vision-based control, and outlines several key concepts within the field. As discussed in Chapter 1, low-cost USB cameras are subject to image imperfections, and any vision-based measurements utilising this technology may, as a result, contain significant errors. In this section, a method of achieving greater robustness is therefore discussed. In Section 3.2, we survey the strategies used to stabilise the Cart Inverted Pendulum using traditional feedback approaches. Finally, Section 3.3 concludes by exploring a range of methods used for vision-based stabilisation. This review goes some way to address some of the fundamental questions posed in Chapter 1.

3.1 Vision-Based Control

Vision-based control, also known as Visual Servoing (VS) (Hutchinson et al., 1996) is the use of visual information in the feedback loop of a control system (Zhang, 2013). One of the earliest examples of VS can be found in the works of Shirai and Inoue (1973); where a control system was developed that could calculate the difference between the desired position of an object and its actual current position, by means of visual processing. The term “Visual Servoing” was formally introduced by

Hill and Park (1979) in their paper titled “Real Time Control of a Robot with a Mobile Camera”. The authors described VS with a “Unimate1”, an early industrial robot. Sanderson and Weiss (1980) made a distinction between the different types of VS, namely direct and indirect approaches. Essentially, with direct VS, a control signal is derived from the visual information and sent directly to the robotic actuator, while the indirect VS approach only generates a reference signal for control; a separate controller executes the motion based on the generated information. Hutchinson et al. (1996) and Chaumette (2014) provide a comprehensive study of VS by reviewing a broad range of applications. Chesi and Hashimoto (2010) and Azizian et al. (2014) provide thorough reviews of the developments in VS, some of which are still considered to be state-of-the-art.

3.1.1 Position-Based Visual Servoing

Position-Based Visual Servoing, also known as PBVS employs an estimated object pose as the main control objective (Weiss et al., 1987). When applied in a single camera environment, full knowledge of the intrinsic parameters of the camera (i.e. focal length, principal point, etc.) is essential, since PBVS operates in 3D Cartesian space. Visual processing errors have long been attributed to complex object form (DeMenthon and Davis, 1995) and oversimplification of the object model (Lowe, 1992). PBVS offers several different solutions to address these errors. An early contemporary approach is to decouple the translational and rotational motions, which results in a straight line trajectory for the camera in Cartesian space (Chaumette and Hutchinson, 2006). This solution allows the camera to follow a deterministic

trajectory, which is the shortest path in Cartesian space. The principle limitation of this approach is the inability to control image features directly. It is possible that an object may leave the FOV, which will compromise the stability of the system. Much research has been directed at solving this problem.

Historically, Wilson et al. (1996) used an Extended Kalman Filter (EKF) (Welsch & Bishop, 1995), (Chui and Chen, 2017) to estimate the state of the target system (i.e. pose). The filter was able to cope with the loss of image features. In their proposed method, features were represented by a covariance matrix, which was altered according to the presence and location of an object in the FOV. Essentially, the effect of a missing feature is removed from the pose estimation. An alternate approach was proposed in the work undertaken by Thuilot et al. (2002), who proposed a PBVS method to keep the object in the FOV by tracking an iteratively computed trajectory.

3.1.2 Image-Based Visual Servoing

Image-based visual Servoing, also known as IBVS, employs image measurements directly as a control objective (Weiss et al., 1987), (Thomas et al., 2014). The task is expressed as an error-function, which should be minimised, by applying a suitable control law. The primary advantage of this approach is that a 3D model is not required, and consequently, the image processing task is simplified. Conventional IBVS uses the error between corresponding image features that lie on a 2D Cartesian plane. In addition to 2D image features, other types of features have been studied and applied as visual measurements for feedback purposes.

The work presented by Thomas et al. (2014), Xie et al. (2016), Schramm et al. (2004) and Cervera et al. (2003) demonstrates IBVS with 3D image features. The authors factored 2D depth of image points into account while controlling the error. In their paper, Andreff et al. (2000) used 3D lines for Servoing. A control law was derived for multiple lines, which required the depth to be observed. The work presented by Tahri and Chaumette (2004) studied Moments as visual features, which derived the analytical form of the interaction matrix (i.e. the matrix that corresponds to camera and feature velocities). An extension to this non-parallel configuration was later proposed by the same authors (Tahri and Chaumette, 2005). Hajiloo et al. (2016) presented an IBVS controller for a 6-degrees-of freedom (DOF) robotic system based on the robust model predictive control (RMPC) method. The controller was designed to consider the visual Servoing system's input and output constraints, such as physical limitations and visibility constraints. The proposed IBVS controller avoided the inverse of the image Jacobian matrix and hence could solve the intractable problems for the classical IBVS controller, such as large displacements between the initial and the desired positions of the camera. To verify the effectiveness of the proposed algorithm, the authors simulated Real-Time experimental results on a 6-DOF robot manipulator with eye-in-hand configuration, which they presented and discussed. Bora1 et al. (2015) present a robust analysis of constrained IBVS based on Nonlinear Model Predictive Control (NMPC). They argued that real applications such an aerial or a fast underwater robotic system suffer from the presence of external disturbances. These kinds of disturbances, they conclude, are inevitable in physical systems; it is therefore of great interest to employ robust controllers. The authors

suggest that rigorous robustness analysis should be conducted. In this, the IBVS system under the MPC framework is proven to be Input-to-State Stable (ISS) and permissible upper bounds of the disturbances are provided.

3.1.3 Applications of Visual Servoing

Visual feedback control loops have been introduced to expand the flexibility and accuracy of robotic systems. The aim of the VS approach is to control a mechanical platform by employing the information provided by a vision system. Applications of robotic equipment are both diverse and widespread. Such equipment is often used to supplant human operators, as improvements in speed, accuracy, and reliability make then ideal for repetitive manufacturing tasks. Robots can also be deployed in locations and situations where it is unsafe or unfeasible for humans to work, such as deep-sea diving or space exploration, etc.

It is often necessary for robotic equipment to locate itself relative to surrounding objects. For example, in a production environment, components usually arrive in an uncontrolled manner, and their orientation must be determined accurately before further operations can proceed (Giordani et al., 2013), (Berger et al., 2000). While it is true that other technologies exist for determining component orientation, the simplicity and speed of a vision-based system make it an attractive choice. In addition to applications for fixed robots, the work undertaken in this research could cross over into the domain of mobile robotics. For example, the inspection of critical parts inside of a nuclear reactor could be safely accomplished by utilising a mobile robot

equipped with a camera (Gargade et al., 2013), (Kita et al., 1999). While such robots are currently guided by humans, a system that could control and automatically stabilise itself would significantly assist with the avoidance of human error.

3.1.4 Vision-Based Object Tracking

Object tracking is a significant area of research in its own right. It is concerned with tracking items that move in front of a camera (Wu et al., 2013), (Yilmaz et al., 2006), and is a vital component of vision-based control. Prior to being manipulating by robotic equipment using a vision-based system, an object must first be identified then tracked. Usually, tracking is performed by locating distinctive image features, which are easily identifiable and invariant to position, orientation and lighting changes. Distinctive features include histograms of pixel intensity or colours (Comaniciu et al., 2000) and exemplar contours (Buyssens et al., 2015), (Toyama and Blake, 2001). One disadvantage of this kind of feature tracking is the higher computational cost associated, as detection of unique features has a tendency to be quite complex. This disadvantage affects the Real-Time performance that can be achieved. A more efficient approach is to use unique image markers within the scene, which significantly improves the robustness of the detection algorithm, and aids pose estimation. This unique marker approach is employed by our method, as it supports our requirement for robust operation; we discuss this further in Chapter 5.

Existing tracking methods can be split into two distinct groups: those that rely on predetermined models of the tracked scene, and those that learn the surroundings as the scene develops. Significant progress has been made in the latter method since the work of Se et al. (2001) and Davison (2003). Systems that have the ability to learn and create models automatically are progressive, as manual creation of models is time-consuming and restrictive. Methods such as these have an advantage when the scene is variable; however, static scenes allow the exploitation of prior knowledge to assist with tracking performance.

3.1.5 Edge Detection

Edge detection is a key component of vision-based object tracking and is the method used by the majority of tracking systems reviewed in the literature. It encompasses a variety of mathematical approaches aimed at identifying points within an image where the brightness changes suddenly; these points are usually organised into a set of curved line segments called edges.

The work presented by Smith (1997) reviews many traditional methods used for edge detection within an image. Canny (1986) offers a classic paper on edge detection; in his work, object edges are modelled as intensity changes with additive white Gaussian noise (AWGN). Canny derived an optimal filter that maximised the product of a detection probability criterion and a localisation accuracy criterion; he demonstrated that his optimal filter could be approximated by the differential of a Gaussian, which in turn was approximated by the difference of two further Gaussians,

with a good degree of accuracy. Canny observed, however, that his filter was not applicable for locating boundaries between textures. The research presented by Rong et al. (2014) builds on the work of Canny by reducing the sensitivity to noise and thereby improving edge detection. Two adaptive threshold selection methods based on the mean of an image gradient magnitude and standard deviation were put forward for two kinds of (typical) images (one has reduced edge information, and the other has rich edge information) respectively. The improved Canny algorithm is much simpler and easy to realize. Experimental results show that the algorithm can preserve more useful edge information and is more robust to noise.

3.1.6 Pose Estimation Filtering

The Kalman filter (Kalman, 1960) is an algorithm that employs a series of measurements observed over time, which contain statistical noise and other inaccuracies to produce estimates of unknown variables (Brahim et al. 2015). These estimates are typically more precise than those based on a single measurements alone. Welch & Bishop (1995) describe the optimal method for parameter estimation using a Kalman Filter for linear equation models. It was shown to be effective for implementing a model, which assumed that camera velocity was constant. For examples where the equations are non-linear, the EKF linearizes the equations about the current state. EKF has been widely applied in visual tracking as a method of smoothing the output result and thereby providing robust pose estimations (Assa et al., 2014), (Armstrong & Zisserman, 1995).

An alternative to EKF is the Unscented Kalman Filter (UKF) (Gyorgy et al., 2014), (Haykin, 2001), which uses linearizing approximations that are usually greater in accuracy than the direct method used in EKF. Both approaches are limited, however, to applications where the process noise is Gaussian. It is possible to use a Gaussian mixture model to represent the distribution in a multi-modal system more accurately. This approach was used to overcome ambiguities in the tracking of humanoids in work done by Cham & Rehg (1999). Rather than implementing computationally expensive Kalman filtering techniques, we explore the use of efficient, image based filters to achieve greater robustness; this is discussed in Chapter 4.

3.2 Stabilising the Cart Inverted Pendulum

The Linear Quadratic Regulator (LQR) (Terra et al., 2014) is an effective velocity control technique. In work presented by Chatterjee et al. (2002), stabilisation of the Cart Inverted Pendulum system was successfully carried out by linearization of the state model, and designing an LQR to control cart velocity. The effectiveness of Inverted Pendulum stabilisation under linear state feedback control has also been analysed by Henders and Soudack (1996). In their work, the authors concluded that the dynamic equations indicate the existence of stability regions in 4D state-space, and an algorithm was developed that transformed the 4D state-space to 3D space. Bettayeb et al. (2014) stabilised the Inverted Pendulum using fractional *PI*-state feedback, which allowed them to decompose the polynomial into a first order fractional polynomial and integer polynomial; this lead to excellent results in terms of stability, accuracy and robustness. In work presented by Dunnigan (2001), a tutorial

was delivered, which introduced the concept of digital control system design by pole placement using state estimation. The work covered both deterministic and stochastic state estimation. A realisation of intermittent linear-quadratic pole-placement control was empirically demonstrated in the work of Gawthrop and Wang (2006); the authors reported that their approach achieved excellent performance when controlling an Inverted Pendulum. Das and Paul (2011) developed a successful, robust periodic controller with zero pole placement capability for an Inverted Pendulum control system. The authors demonstrated the superiority of their solution over linear time-invariant (LTI) approaches. Dynamic H-Infinity compensation has been developed and applied to the Inverted Pendulum problem in work undertaken by Van der Linden and Lambrechts (1993). The authors considered dry friction and concluded that the controller did not perform satisfactory if it was not taken into account. A controller for stabilising the Inverted Pendulum and minimising the cart position was discussed by Lozano and Fantoni (2000). Their control strategy was based on manipulating the cart and pendulum system energy.

An Energy-speed-gradient approach has been proposed and analysed by Shiriaev et al. (2001), where global stability is guaranteed. A feedback control law was presented for near global stabilisation of the Inverted Pendulum in the work of Angeli (2001), this also ensured asymptotic stability. The work delivered by Garcia et al. (2005) detailed the development of continuous time, sliding mode control and a discrete time, sliding mode controller for an Inverted Pendulum. The authors also claimed near global stabilisation. A hybrid controller for both swing-up and stabilisation has

been attempted in work presented by Srinivasan et al. (2009); the authors successfully applied input-output linearization, energy control and singular perturbation theory to achieve stabilisation. A combined controller was also designed by Zhao and Spong (2001), which again ensured global stabilisation. This approach employed a linear controller for stabilisation, a linear controller for cart position and a combination of several bang-bang controllers for a minimum swing-up time.

There is an ever increasing trend towards the use of intelligent algorithms based on Fuzzy Logic, Genetic Algorithms and Artificial Neural Networks for Inverted Pendulum control and optimisation (Ponce et al., 2014). Shiung and Chen (1998) presented a fuzzy logic based adaptive sliding mode controller, which was capable of automatically correcting for plant non-linearity. Tang et al. (2016) described an approach to realise the stability control of the planar inverted pendulum system, which is a typical multi-variable and strong coupling system. A new fuzzy-evidential controller based on fuzzy inference and evidential reasoning is proposed. Firstly, for each axis, a fuzzy nine-point controller for the rod and a fuzzy nine-point controller for the cart are designed. Subsequently, in order to coordinate two controllers of each axis, a fuzzy-evidential coordinator is proposed. In this new fuzzy-evidential controller, the empirical knowledge for stabilization of the planar inverted pendulum system is expressed by fuzzy rules, while the coordinator of different control variables in each axis is built, incorporated with the dynamic basic probability assignment (BPA) in the frame of fuzzy inference. The fuzzy-evidential coordinator makes the output of the control variable smoother, and the control effect of the new

controller is improved compared with other work. The experiment in MATLAB showed the effectiveness and merit of the proposed method. An adaptive Lyapunov-based Fuzzy Logic controller was described in the works of Wong et al. (1998); the design was verified for the Cart Inverted Pendulum by means of simulation. A Self-organizing Fuzzy controller was designed in work undertaken by Young-Moon et al. (1995). This was also verified for an Inverted Pendulum system. Stability analysis for a Fuzzy Logic, model-based, non-linear controller using Genetic Algorithms was presented in work done by Lam et al. (2003); the authors used an arithmetic crossover and non-uniform mutation, based on Lyapunov's stability theorem with a smaller number of Lyapunov conditions. They successfully applied this to the Inverted Pendulum stabilisation problem. The work undertaken by Tao et al. (2008) produced a controller that was able to ensure global stabilisation. The authors successfully balanced the Inverted Pendulum in zero gravity condition. In their paper, a two controller approach was suggested; a Fuzzy swing-up controller and a sliding position controller.

The choice of parameter gain setting in the design of Fuzzy logic and *PID* controllers significantly affects their performance. The paper presented by Sung-Kwun et al. (2004) gives several methods for estimating gain parameters for an Inverted Pendulum using artificial intelligence. Actuator saturation and integrator windup (where applicable) is also of great importance in the design of Inverted Pendulum control systems; this problem has been addressed with the assistance of a Tagaki-Sugeno (T-S) type, Fuzzy Logic based, gain scheduling algorithms presented by

Yong-Yan and Zongli (2003). The modelling ambiguity is considered as norm bounded uncertainty; the problem of defining the region of attraction for T-S Fuzzy Systems based on normal state feedback is defined with the help of Linear Matrix Inequalities (LMIs). The work undertaken by Lam et al. (2003) demonstrates a Fuzzy Logic controller based on Single Input Rule Modules (SIRMs) with dynamically connected inference modules. The SIRMs were switched dynamically between the modules, one for pendulum angle and the other for the position of the cart; the controller switching was biased towards the angular position. The work undertaken by Koo (2001) presents a Model Adaptive Reference Fuzzy Controller (MARFC). Where the Fuzzy Logic information base is modified according to the error generated from the reference model and the actual pendulum.

In work undertaken by Chakraborty et al. (2013) the authors demonstrate the capability of genetic algorithms to solve complex and constrained optimisation problems by optimising the gain parameters of a *PID* controller. An Inverted Pendulum was used as a means to verify the reliability and robustness of their method. Genetic Algorithms were also applied to the Inverted Pendulum problem by Chen and Wong (2002), in a scheme to generate Fuzzy Logic controllers automatically. Each parameter of the Fuzzy Logic controller is tuned, with the assistance of a fitness function to guide the searching algorithm.

In each example found in the literature, the application of intelligent algorithms for traditional controller optimisation reported positive results. This positive report was

inspirational in the course of our research and we decided to employ ANNs to optimise our vision-based method.

3.3 Vision-Based Stabilisation of the Cart Inverted Pendulum

As demonstrated in Section 3.2, various techniques have been proposed for controlling the Cart Inverted Pendulum using non-vision-based feedback approaches. A limited number of methods, however, have been reported that use visual information for pendulum control. Starting with one of the earliest pieces of work, Magana and Holzapfez (1998) successfully controlled the pendulum angle with a vision based Fuzzy Logic controller. They concluded that oscillation in the pendulum angle at rest was approximately ±2.7º. Quesada and Velasco (2006) studied vision-based control of an Inverted Pendulum and applied full state feedback using a Digital Signal Processor (DSP). The total oscillation at rest was reported to be ±10º, which was an achievement given the restricted processing platform. A similar solution is provided by Tu and Ho (2010). They controlled a rotating pendulum using a system based on a Field Programmable Gate Array (FPGA) and a DSP, with visual feedback. The pendulum angle was successfully measured by attaching visual markers to detect the pole. Based on the results provided in their study, the pendulum angle was successfully controlled within ±1.5º. Wang et al. (2008) used a camera to measure the pendulum angle and employed an incremental encoder to monitor the cart position. Under stabilisation control, the pendulum angle and the cart position were reported to be oscillating at approximately ±10º and ±0.04m respectively.

Wang (2010) presented a hybrid approach, fusing a Charge Coupled Device (CCD) camera and two encoders. He demonstrated a new modelling and trajectory tracking control system for a 2-DOF Inverted Pendulum, employing contactless feedback. His proposed modelling method transformed the 2-DOF problem into a 1-DOF one by choosing a new balancing plane for the pendulum at each camera sampling instant. For each plane, two feedback loops were considered; the first was an observation loop, processing the delayed and sampled angle information delivered by the vision system. The second, being a stabilisation loop, realised the Lyapunov function based control, to stabilise the pendulum system. The author used the results gained by simulation to demonstrate the performance of the proposed method. Performance issues were highlighted in the method, which was illustrated by experimental figures and videos. Results showed significant oscillations in the stabilisation control, which were attributed to inaccurate viscous friction identification between the cart and the pole. Stuflesser and Brandner (2008) proposed a purely vision-based control method using cascaded particle filters. Their approach was able to swing up and balance the pendulum, while avoiding the use of visual markers on any part of the system. Low-cost components were used to realise the single camera tracking system; experimental results demonstrated that it was able to robustly control the pendulum in Real-Time using a standard desktop PC. Wenzel et al. (2000) presented a system that makes use of dedicated markers, which are detected in a sequence of camera images using a pattern matching algorithm. Martinez and Becerra (2006) employed optical flow as the primary source of information to train and consequently apply an adaptive

controller to the Inverted Pendulum problem; their work was based on simulations in a virtual reality environment.

Kizir et al. (2012) successfully stabilised the Inverted Pendulum using visual feedback, without any markers. The authors reported that the pendulum angle and cart position were controlled with minimal oscillation ±0.2° and ±0.002m respectively. Their work was very impressive and set the standard for vision-based control of an Inverted Pendulum. Hladik (2016) examined the possibility of achieving fast, on-the-fly image processing and control of an unstable system (Inverted Pendulum) using a dedicated embedded smart camera, with an on-board Digital Signal processor. The author used a convolution based algorithm to detect the position of the pendulum and a Linear Quadratic (LQ) regulator to control its position. The author concluded that to successfully control the Inverted Pendulum using a camera based system, a high frame rate was required.

3.4 Conclusion

In this chapter we have explored previous work relating to visual stabilisation of the Cart Inverted Pendulum; an application of Vision-based Control. A broad review of historical research surrounding visual stabilisation outlined several key concepts within this field.

Early examples of VS were introduced, notably the work of Shirai and Inoue (1973); which calculated the difference between the desired position of an object and its actual current position.

An evaluation of the strategies used to stabilise the Cart Inverted Pendulum were explored using traditional feedback approaches. A continuation into examining methods used for Vision-based stabilisation, addressed many of the questions posed in Chapter 1.

A review of Position-Based Visual Servoing looked at the estimated object pose as the main control objective. This highlighted that when applied in a single camera environment, full knowledge of the intrinsic parameters of the camera (i.e. focal length, principal point, etc.) were essential. A consideration of contemporary approaches empathized the need to decouple the translational and rotational motions; which resulted in a straight line trajectory for the camera in Cartesian space.

Applications of Visual Servoing were further reviewed, which examined visual feedback control loops; a method introduced to expand the flexibility and accuracy within the field of robotics. Vision-Based Object Tracking was identified as a significant area of research. This area, which is concerned with the tracking of objects moving in front of a camera, was found to be vital in Vision-based control. It was concluded, however, that higher computational costs are associated with detection of unique features. Markers were identified as a solution that can ease the computational burden. This was deemed to be the preferred method employed by the majority of tracking systems reviewed in the literature.

Pose Estimation Filtering was an important aspect to review. This proposed an algorithm that employed a series of measurements observed over time. This method contained statistical noise and other inaccuracies to produce estimates of unknown variables; which established that these estimates were typically more precise than those based on a single measurement alone.

In examining an effective velocity control technique, a consideration of The Linear Quadratic Regulator (LQR) was observed. This highlighted the works of Chatterjee et al. (2002), who discussed how stabilisation of the Cart Inverted Pendulum system was successfully carried out by linearization of the state model, and designing an LQR to control cart velocity. Following a comprehensive review of Vision-based stabilisation of the Inverted Pendulum, various methods used for solving the problem were identified. The majority of solutions found in the literature were verified by means of simulation. We believe that further exploration is certainly possible in this field, particularly outside of simulation, by using a real-world Inverted Pendulum system.

Chapter 4

Test System Development and Experimentation

In this chapter, we discuss the development of our vision-based Cart Inverted Pendulum system, and the experimentation phase that followed. We begin with an overview of the main system components and provide some technical justification for their use. We discuss the design of the *PID* controller for the stabilisation task, starting with the key specifications. We build on equation (2.21) presented in Chapter 2, by expressing it in discrete form to allow it to be executed in the digital domain. We present our *PID* control algorithm, which is able to stabilise the Inverted Pendulum; and we tune it using the Zeigler-Nicholas method, as discussed in Chapter 2. We then discuss the test software development, starting with the main specifications; we also give an overview of the Graphical User Interface. We conclude the chapter with a discussion of the experimentation phase and present a summary of the results.

4.1 Main System Components

A. Low-Cost USB Camera

The Sony PlayStation® 3 (PS3) Eye was selected as the low-cost USB camera in our system. The camera has a maximum resolution of 640 x 480 pixels and can capture up to 60 Frames per second (FPS) at this resolution. It has a colour CMOS image sensor that enhances low-light operation. The camera has a two-setting, fixed focus

lens, which offers either 56º or 75º FOV, for different framing applications. The camera was placed at a fixed distance from the Inverted Pendulum to allow the FOV to cover the cart and pole with maximum resolution. This camera was chosen primarily because of its high frame rate ability, low-light performance and low purchase price. The PS3 Eye camera is shown in Figure.4.1.

Figure.4.1 – *Sony PlayStation® 3 Eye USB Camera.*

B. PIC32-Pinguino Development Board

During the specification phase of the Inverted Pendulum control unit, several microcontroller development boards were considered. The PIC32-Pinguino board (see Figure.3.2) from OLIMEX Ltd was ultimately chosen, as it presented several key advantages over the other options:

- It is a high performance, industrial grade, electronic hardware development platform with open-source architecture. These characteristics ensured a robust

solution and allow unrestricted access to the schematics and mechanical board files (See Appendix.1).

- It is compatible with many well-established PIC microcontroller development environments such as MPLAB and MPLABX. The firmware could also be written in the 'C' programming language, which allowed rapid development. The Microchip PIC C32 compiler was employed, as it was readily available and was free for non-commercial use.

- It is an extremely low-cost platform considering the rich feature set offered. The development board included all of the necessary ADC channels, PWM modules, USB port, and digital IO required for interfacing with the Cart Inverted Pendulum.

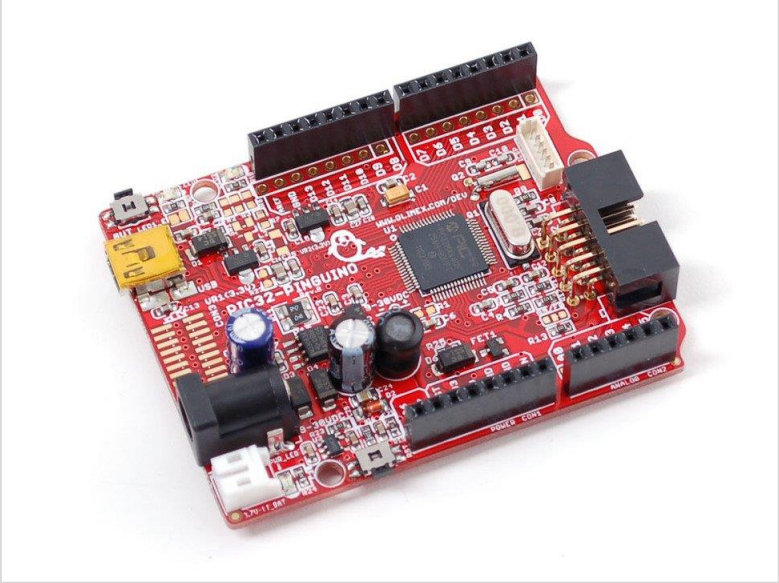


Figure.4.2 – *PIC32-Pinguino Development Board from OLIMEX Ltd.*

The pin allocation for the PIC32-Pingunio board is shown in Table.4.1. This table is referenced in the following subsections. For full schematic and mechanical diagrams, the reader is referred to Appendix.1.

PIN	CON1	CON2	CON4	CON5
1	RST	A0	D0	D8
2	3.3V	A1	D1	D9
3	5.0V	A2	D2	D10
4	GND	A3	D3	D11
5	GND	A4	D4	D12
6	VIN	A5	D5	D13
7	-	-	D6	GND
8	-	-	D7	REF

Table.4.1 *– PIC32-Pinguino Pin Allocation Table.*

C. Angular Displacement Sensor

To measure the angular displacement **θ** and angular velocity ***ω*** of the pole a linear, circular potentiometer was used. A circular potentiometer is a resistor whose resistance changes with the movement of the rotating shaft. Mathematically, this relationship can be expressed as follows:

$$R_p(\theta)=k_p\theta \tag{4.1}$$

Where $\boldsymbol{R_p}$ is the potentiometer's variable output resistance, $\boldsymbol{\theta}$ is the angular position of the pole in degrees (or radians), and $\boldsymbol{k_p}$ is the proportionality constant in ohms/degree. Figure.4.3 shows a typical potentiometer and associated schematic representation, as used in our Cart Inverted Pendulum platform.

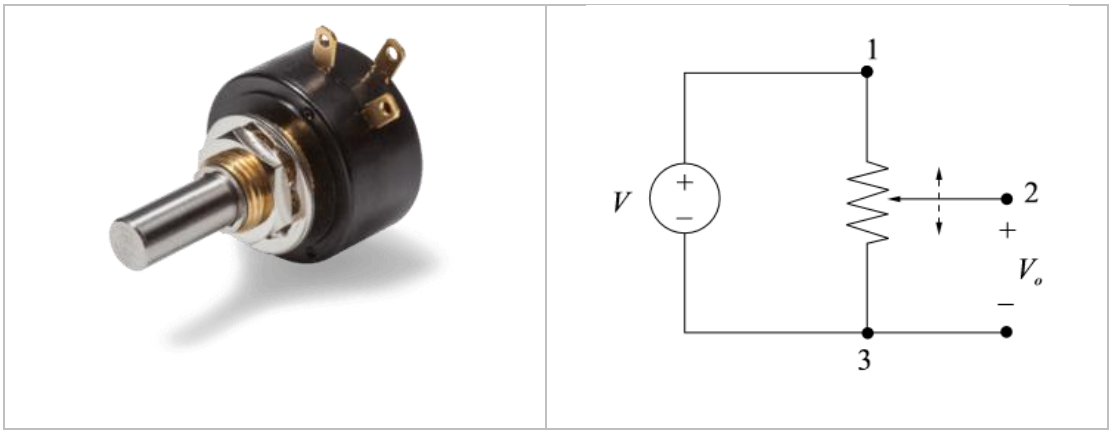


Figure.4.3 *– Typical Potentiometer and Schematic Representation.*

From equation (4.1):

$$V_0 = \frac{R_{23}}{R_{13}} V = \frac{R_p}{R_{13}} V = \frac{K_p \theta}{R_{13}} V = \bar{K}_p \theta, \qquad \bar{K}_p \triangleq \frac{K_p}{R_{13}} V \tag{4.2}$$

Where $\boldsymbol{R_{23}}$ is the resistance between terminals 2 and 3 (the potentiometer's output resistance **Rp**) and $\boldsymbol{R_{13}}$ is the resistance between terminals 1 and 3 (the potentiometer's constant, permanent resistance), and **Kp** is the proportionality constant in volts/degree. From (4.2) it can be seen that if we know **Kp**, then we can determine the angular displacement by measuring the output voltage **Vo**.

The potentiometer's output voltage (**Vo)** was applied to pin 2 (A1) of the PIC32-Pinguino development board, to allow the ADC to make digital measurements of $\boldsymbol{\theta}$**.** With a reference voltage of 3.3V and an ADC resolution of 10-bits, this gave 3mV per bit of resolution. A ratiometric arrangement was used with the potentiometer and the ADC; this gave 9mV per degree of rotation. We therefore achieved an angular resolution of 0.33º per bit, which was sufficient for our application. This angular resolution could be significantly improved by using an external ADC with a higher converter resolution.

D. Direct Current Motor

The direct current (D.C.) motor has become an important drive configuration for many application areas. A D.C. motor is used to develop torque on the pendulum cart via a 1:1 ratio belt drive mechanism. The motor is of the type "Permanent Magnet", and the behaviour of this kind of machine is caused by the interaction between electrical and magnetic forces. The motor chosen for the project is 12V D.C. motor, which is similar to the one shown in Figure.4.4.

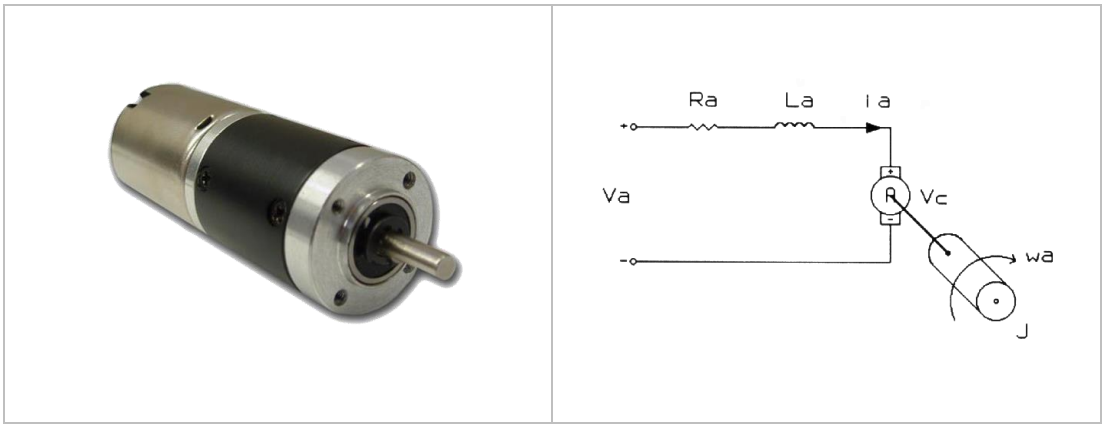


Figure.4.4 *– Typical D.C. Motor and Schematic Representation.*

Mathematically, the D.C. motor can be expressed as a differential equation by applying Kirchhoff's voltage law:

$$V_a - V_{Ra} - V_{La} - V_C = 0 \tag{4.3}$$

Applying Ohm's law, the voltage across the resistor can be expressed as:

$$V_{Ra} = iR_a \tag{4.4}$$

Where iR_a is the armature current. The voltage across the inductor is proportional to the change of current through the coil with respect to time and this can be expressed as:

$$V_{La} = L_a \frac{d}{dt} i_a \tag{4.5}$$

Where L_a is the inductance of the armature coil. Finally, the reverse electromotive force (EMF) can be express as:

$$V_C = K_v \omega_a \tag{4.6}$$

Where K_v is the velocity constant determined by the flux density of the permanent magnets, the reluctance of the iron core of the armature, and the number of turns on the armature winding. ω_a is the rotational velocity of the armature. By substituting equations 4.4, 4.5, and 4.6 into equation 4.3 we arrive at the following differential equation:

$$V_a - i_a R_a - L_a \frac{d}{dt} i_a - K_v \omega_a = 0 \quad (4.7)$$

From equation 4.7 it can be seen that the rotational speed is directly proportional to the applied input voltage. Therefore, to control the speed of the cart along the pendulum track, we must modulate the voltage applied to the D.C. motor.

E. Motor Driver and Pulse Width Modulation

The coils of the D.C. motor are connected to the applied voltage source via an H-Bridge driver. The H-Bridge is an electronic circuit that is used in control of relatively high current devices, particularly where the device polarity is required to be reversed, e.g. cart direction control. The H-Bridge driver that was used in this research is the L298N, which is a monolithic Integrated Circuit (IC). This modular part was chosen since it reduced the number of external components required, as shown in Figure.4.5.

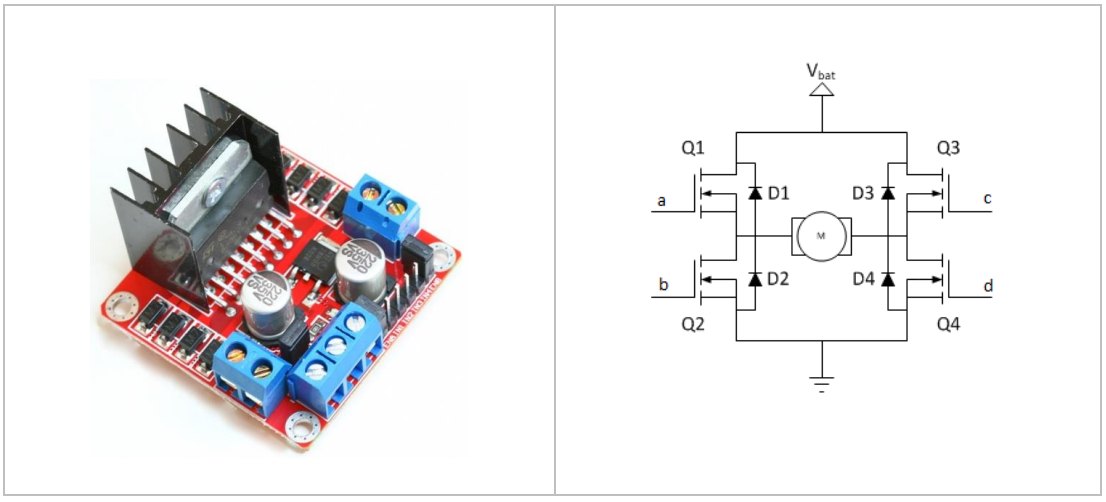


Figure.4.5 *– Structure of an H-Bridge Driver Module.*

With reference to the schematic diagram shown in Figure.4.5, to apply the voltage from Vbat directly across the motor, a pair of diagonally opposed transistors is required to be switched on. Depending on which pair of transistors is activated, the motor will rotate clockwise (CW) or counter-clockwise (CCW). For example, when transistors 'Q1' and 'Q4' are enabled (and 'Q3' and 'Q2' are disabled) a positive voltage is applied to the motor. Conversely, when transistors 'Q1' and 'Q4' are disabled (and 'Q3' and 'Q2' are activated), the voltage is reversed to the motor, reversing the direction of movement. Transistors 'Q1' and 'Q2' (and 'Q3' and 'Q4') should never be activated simultaneously; this would lead to a short circuit condition, resulting in permanent damage to the driver. Diodes D1 to D4 offer back EMF protection for the transistors during switching of the motor due to its inductance. The following table (Table.4.2) summarises the operation:

Q1	Q2	Q3	Q4	Operation
1	0	0	1	CW
0	1	1	0	CCW
0	0	0	0	Freewheel
0	1	0	1	Break

Table.4.2– *Summary of H-Bridge Driver Operation.*

The L298N driver is fitted with a sufficiently large heat-sink to dissipate the power generated by the driver IC. The greatest source dissipation inside the driver IC is the power dissipated in the FET ON resistance ($R_{DS(ON)}$). The power dissipated in each transistor, which is attributed to $R_{DS(ON)}$ is given by the following expression:

$$PRDS = (HS\_RDS(ON)\ x\ (IOUT)) + (LS\_RDS(ON)\ x\ (IOUT)) \quad (4.8)$$

Where PRDS is the power dissipated in the output FETs, HS_RDS(ON) is the resistance of the high-side FET, LS_RDS(ON) is the resistance of the low side FET, and IOUT is the output current being applied to the motor.

As shown in equation (4.6), the simplest method of controlling the rotation speed of the D.C. motor is to control the driving voltage; the greater the voltage applied, the greater the speed of the motor. Pulse Width Modulation (PWM) is a well-established method for controlling an output voltage with a variable equivalent signal in the digital domain. In PWM, the ratio of "on" time to "off" time (duty cycle) of a carrier signal determines the average voltage, and therefore the speed of the motor (see Figure.4.6).

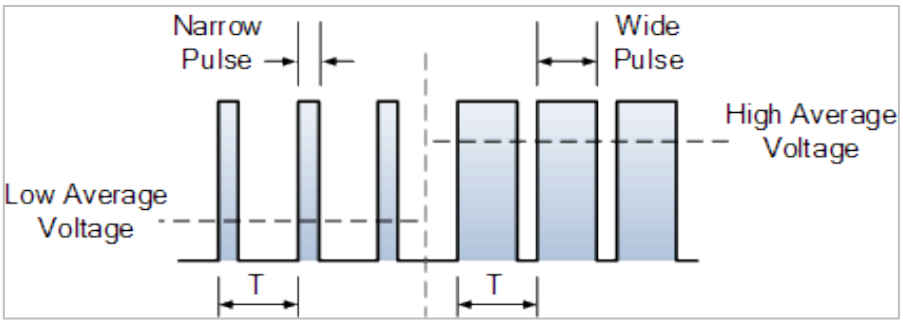


Figure.4.6 *– PWM Voltage Control Waveform.*

The carrier frequency must be considerably higher than that which would affect the load; i.e. the waveform perceived by the load must be as smooth as possible. The main advantage of PWM is that power loss in the switching devices is kept to a minimum. When the switch is disabled, there is essentially zero current flow; and

when it is enabled, there is virtually no voltage drop across the switch. Power loss, being the product of voltage and current, is thus close to zero in both cases. Mathematically, PWM output can be expressed as follows:

$$\delta(t) = sign(V_r(t) - V_c(t)) \quad (4.9)$$

Where V_r is the ramp waveform generator output, V_c is the compensator output and sign is a sign function, which gives the binary output depending on the difference between V_r and V_C. The PIC32-Pinguno development board was connected to the motor driver via two digital IO pins (D0 and D1); one pin was used to control the motor direction and the other pin for outputting the PWM signal, which controlled the motor speed. This action, in turn, controlled the direction and speed of the cart along the linear slide.

F. Linear Slide Mechanism

A linear slide is a bearing mechanism that is designed to provide motion in one or two dimensions. The Cart Inverted Pendulum is a linear slide that comprises of a 380mm x 90mm chassis, which supports a mobile plastic cart on a bearing. Attached to the cart is a steel bar, which is connected to the angular displacement sensor via a coupling arrangement. This bar forms the actual pendulum pole. An element of friction was present in the mechanism due to the handcrafting of a number of mechanical supports. The cart bearing and the support rods were lubricated with Teflon based lubricant to ensure that the friction was kept to a minimum. The main components of the linear slide are shown in Figure.4.7.

Figure.4.7 – *Linear Slide Rod and Bearings.*

4.2 Custom PID Controller

To stabilise the Cart Inverted Pendulum using *PID* control, required a suitable controller that was programmed with the appropriate control law. We developed our controller using the Microchip MPLAB 8 IDE, using the principles discussed in Chapter 2.

4.2.1 Controller Specification

The first phase of the *PID* controller development was to explicitly define a set of requirements for the behaviour of the system. These requirements are summarised as follows:

A. Controller Input

Along with the internal reference signal r(t), the *PID* controller shall accept a single input y(t), which is the angular displacement **θ** of the pendulum. This input shall be in the range of 0-3.3V, corresponding to minimum and maximum pendulum angles of 0-360° respectively. As discussed in Chapter 2, the difference between the reference value and the set-point shall denote the error signal e(t) i.e. e(t) = r(t) – y(t).

B. Controller Output

The PID controller shall provide a single PWM output u(t) that is capable of driving a D.C. motor, which is attached to the cart. This output shall be in the range of 0-12V, where 0V corresponds to a stationary state and 12V corresponds to full torque. The direction of the motor shall be controlled by a single digital output, where '0' denotes CCW and '1' denotes CW.

C. Sampling Rate

To achieve the necessary impulse response, the controller input signal y(t) shall be sampled at a frequency (F_s) of at least 1KHz. The input signal y(t) shall be filtered with an anti-aliasing filter prior to being sampled to avoid distortion. From the Nyquist-Shannon sampling theorem, the cut-off frequency of this filter shall be $F_s/2$ or 500Hz. A simple first order RC filter shall be used to achieve this.

D. PID Parameter Gains

The units of P, I and D gains shall be given in integer notation, and shall be in the range of 0 and 1023. A value of 0 shall denote the lowest gain, and 1023 shall denote the highest value.

4.2.2 Discrete PID Control Implementation

The realisation of analogue *PID* control on a microcontroller-based platform is not feasible and dictates the move from a continuous time format of equation (2.22) (see Chapter 2) to a discrete time approximation. There are three parameters relating to the

error function, e(t), that must be approximated to implement the full *PID* control algorithm. Figure.4.8 shows the *PID* controller model, where K_P, K_I, and K_D denote the gains of the proportional, integral, and derivative terms respectively.

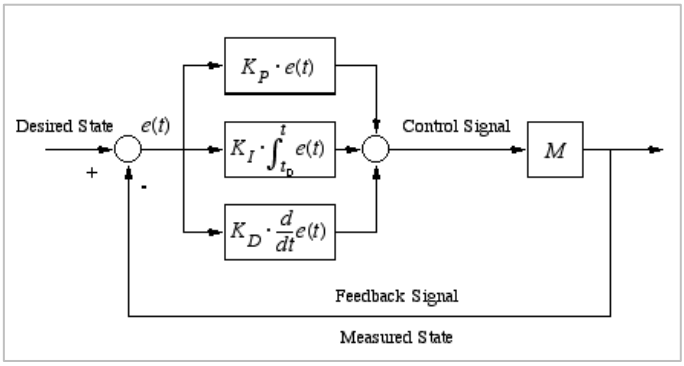


Figure.4.8 *- PID Control System Model.*

The transfer function of the system shown in Figure.4.8 is given by:

$$\frac{u}{e}(s) = H(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{4.10}$$

This gives *u* with respect to *e* in the time domain:

$$u(t) = Kp(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}) \tag{4.11}$$

Approximating the integral and the derivative terms gives the discrete form:

$$u(n) = K_p e(n) + K_i \sum_{k=0}^{n} e(k) + K_d(e(n) - e(n-1)) \tag{4.12}$$

Where:

$$K_i = \frac{K_p T}{T_i} \qquad K_d = \frac{K_p T_d}{T} \tag{4.13}$$

The controller was found to be improved by basing the derivative term on the process value only:

$$u(n) = K_p e(n) + K_i \sum_{k=0}^{n} e(k) + K_d (y(n) - y(n-1)) \tag{4.14}$$

The equation given in (4.14) was implemented on our PIC32-Pinguino microcontroller development board. Our approach was to configure an 8-Bit timer counter to continuously overflow, which caused an interrupt service routine (ISR) to be called periodically. Each occurrence of the ISR forced a read of the pendulum angle and a ready flag to be set, which signalled the main routine to perform a PID update. The main routine executes equation (4.14) and outputs the control variable to the motor driver, thereby stabilising the Cart Inverted Pendulum. A flow-chart of our discrete PID control algorithm is shown in Figure.4.9. The reader is referred to Appendix.2 for the actual ‘C’ code listing.

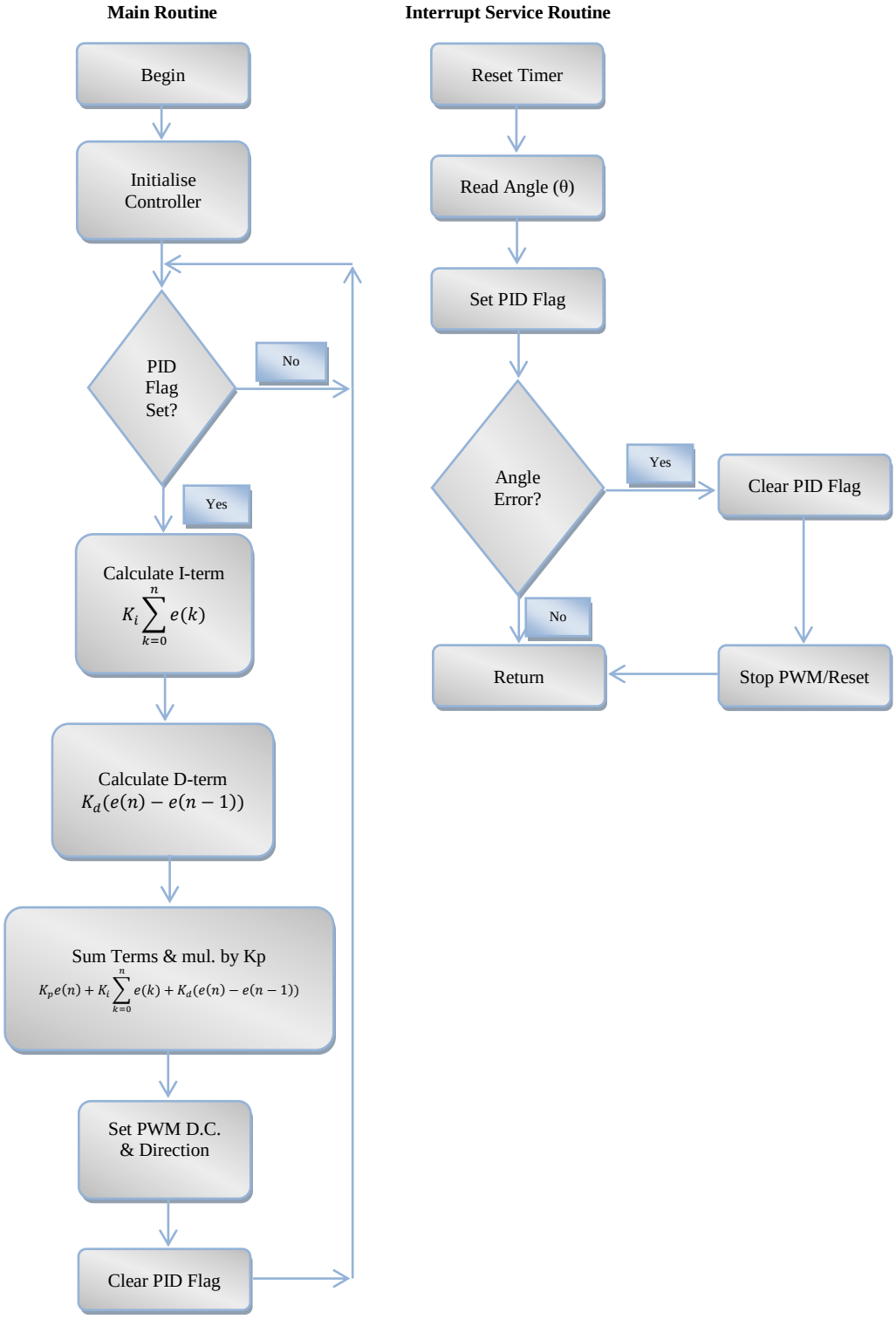


***Figure.4.9** – Our Discrete PID Control Algorithm.*

4.2.3 Controller Tuning

The constant gain values of the discrete *PID* controller play a vital role in the system output response and stability. Since a detailed mathematical description of the exact system is unavailable, experimental tuning of the *PID* parameters was therefore necessary. To find the constant gain values of our controller, the Zeigler-Nicholas tuning procedure was followed, as discussed in Chapter 2. This procedure was performed and the pendulum was tuned accordingly, prior to the data-harvesting phase.

4.3 Test Software Development

To facilitate testing of our vision-based control algorithms, a custom software environment was developed, using Microsoft VB .NET 2008 and the EmguCV image processing library. A Graphical User Interface (GUI) was designed to show Real-Time system information, and to provide a framework for algorithm testing.

4.3.1 Software Specifications

The first step of the software development phase was to explicitly define a set of requirements for the behaviour of the tool. These requirements are summarised as follows:

A. The software shall capture images from the USB camera and make them available for processing by our algorithm. The images shall be available in Blue-Green-Red (BGR) and Hue-Saturated-Values (HSV) formats. The images shall be displayed for monitoring purposes.

B. The software shall provide two customisable image filters, which shall be used for marker identification and tracking. These filters shall allow the HSV values to be set individually. The individually filtered images shall be displayed for monitoring purposes.

C. The software shall provide morphological filtering options such as erosion and dilation, and shall offer a minimum segment size option.

D. The software shall provide a range of information charts, which shall include the parameters of pendulum angle, controller error and driver output. These parameters shall also be available in the form of a live on-screen-display (OSD).

4.3.2 Graphical User Interface

The Microsoft VB .NET environment allowed us to design our GUI to have of two main sections. The left-hand section showed image related windows and the right-hand section showed pendulum related charts (see Figure.4.10). The GUI allowed us to dynamically switch between traditional PID control and vision-based control. The reader is referred to Appendix.3 for Form layouts and VB .NET code listings.

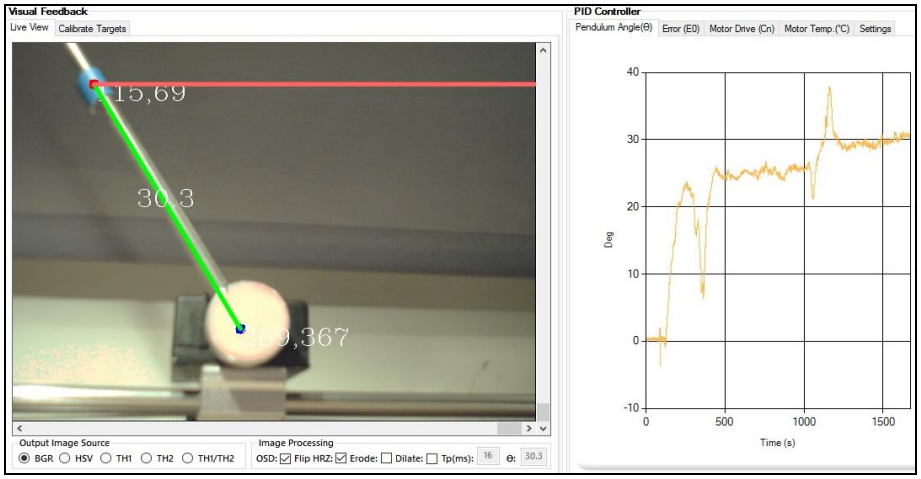


Figure.4.10 *– Test Software Graphical User Interface – Main Window View.*

Regarding Figure.4.10, the main window shown on the left displays live images, which have been captured by the USB camera in BGR or HSV format. The user is given the opportunity to view the image filters, which are used for marker identification. The two image filters can be customised to create a model of the cart and pole markers, by using a segmentation technique (discussed in Chapter 5). The customisation panel is accessed via a separate tab, as shown in Figure.4.11.

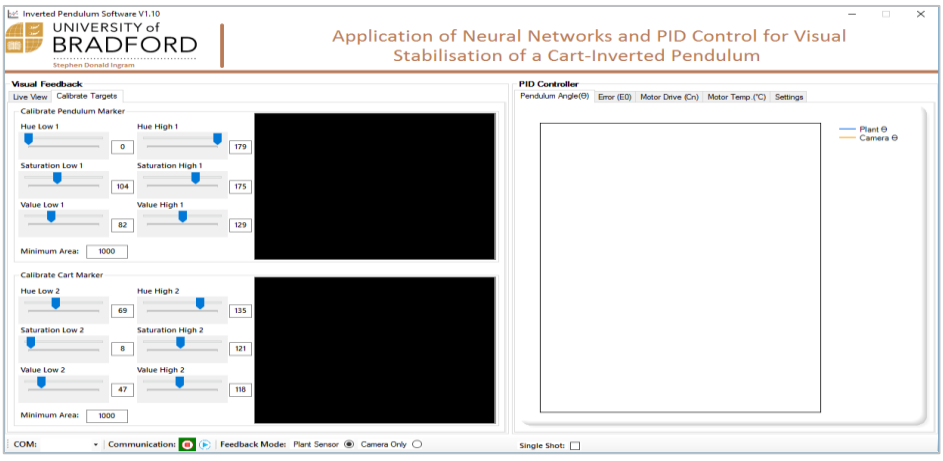


Figure.4.11 *– Test Software Graphical User Interface – Marker Customisation View.*

With regards to Figure.4.11, on the left-hand section of the GUI, each individual value of the HSV filter is customisable between 0-255. This allows maximum flexibility when creating a model of the cart and pole markers. The HSV colour space is discussed in Chapter 5. On the right-hand section of the GUI, a range of pendulum related parameter charts are provided. These charts are generated by the software to allow monitoring of the pole angle, system error and other physical parameters during testing. The image processing algorithm is executed on a separate thread to the main GUI. This method of concurrent programming is used to assist with the Real-Time performance of the system.

4.3.3 OpenCV Image Library

To perform the complex image processing tasks required by our vision-based control method, we used the Open Source Computer Vision Library (OpenCV) as the basis for our image processing code. OpenCV is a library of programming functions, which are designed for Real-Time computer vision applications; it offers in excess of 2000 optimised algorithms (see Figure.4.12). It has a BSD license (free for commercial or research use), which is primarily why it was chosen for this research. The main advantages of OpenCV are:

A. It has been optimised for Real-Time applications.
B. It is independent of operating system and hardware platform.
C. It offers both low and high-level APIs for image functions.

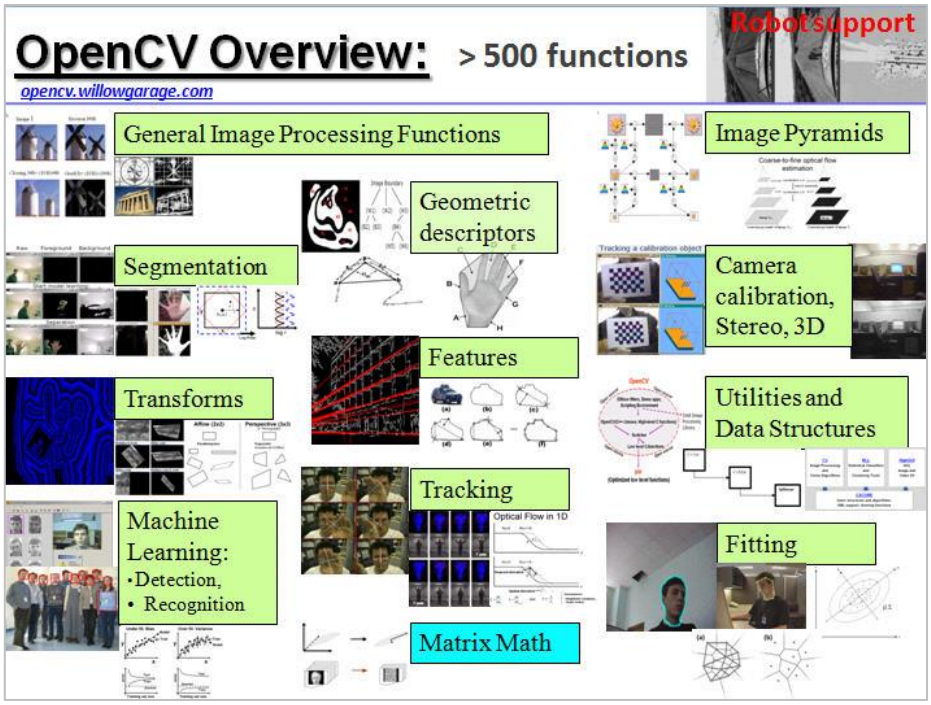


Figure.4.12 *– Overview of OpenCV Functions (OpenCV Wiki, 2016).*

OpenCV was originally written in the 'C' programming language and has been developed to provide a complete C++ interface. Our chosen language suite, however, was VB .NET. A compatible encapsulation of the OpenCV library was therefore required.

4.3.4 EmguCV Library

As previously mentioned, OpenCV is a C/C++ library, which does not easily integrate into the .NET platform without extensive supplementary code. We therefore used the EmguCV cross-platform, .NET wrapper for OpenCV in our application. EmguCV has been written in C# and can be compiled in Mono. The main advantage is that it is able to run on any platform that is Mono compatible, including Linux, Solaris and Mac OS. An overview of the EmguCV library is given in Figure.4.13.

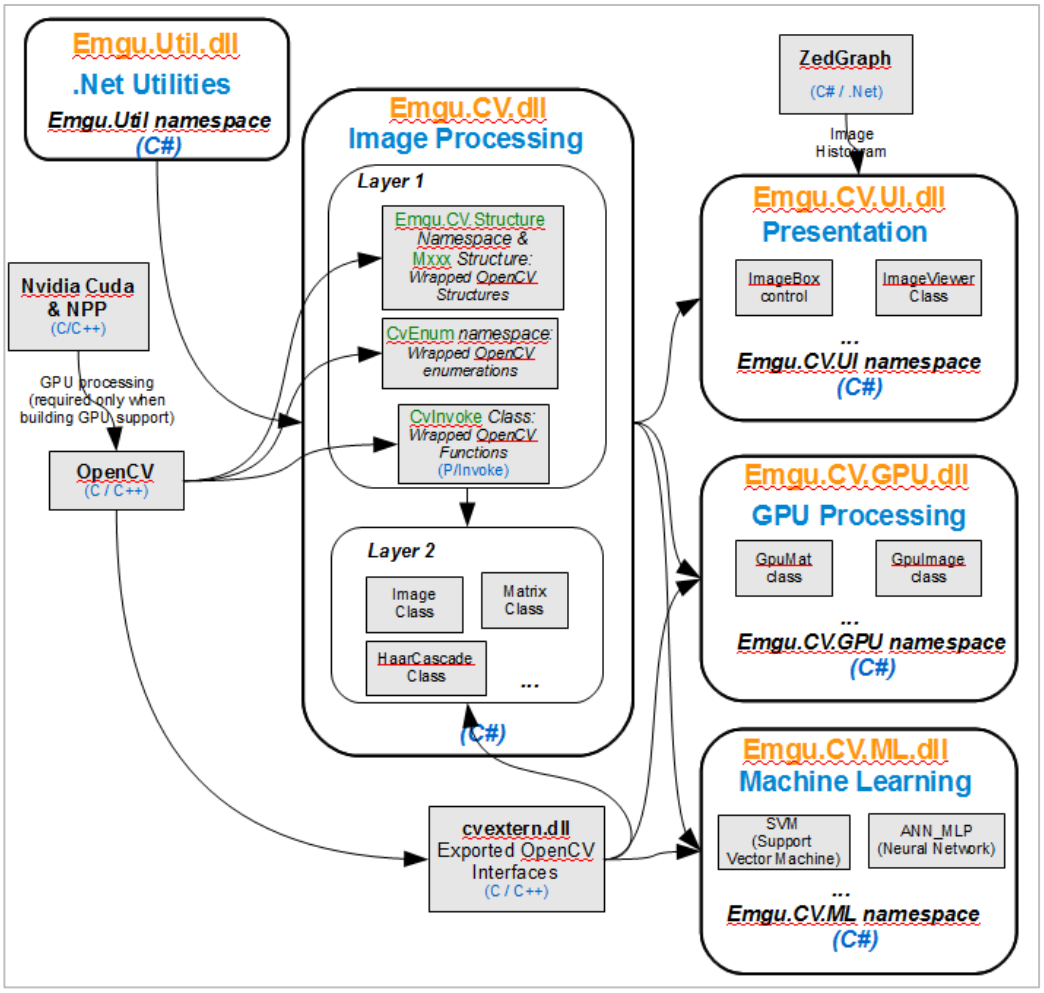


Figure.4.13 *– Overview of EmguCV Library (Emgu.com, 2016).*

4.3.5 Morphological Image Restoration

Morphological image restoration provides a means to reconstruct damaged regions of an image. Image morphology, in its most fundamental form, is constructed by operations on pixel sets. Dilation and Erosion are the two most significant morphological operations in our research. The final state of any pixel in the output image is defined by applying a rule to the corresponding pixels and its neighbours in the input image. The segments are expanded in Dilation, which means that small

voids are filled, and disjointed objects are connected. Conversely, Erosion diminishes the segments by etching away their boundaries. The ability to select the proper structuring element and the required morphological operation is offered by the test software.

4.4 Experimentation and Summary of Results

This section presents a summary of the results obtained by experimenting with our Cart Inverted Pendulum system (shown in Figure.4.14) using traditional PID control. Only a summary of the results is provided, a detailed evaluation of the effects of each parameter would far exceed the scope of this thesis.

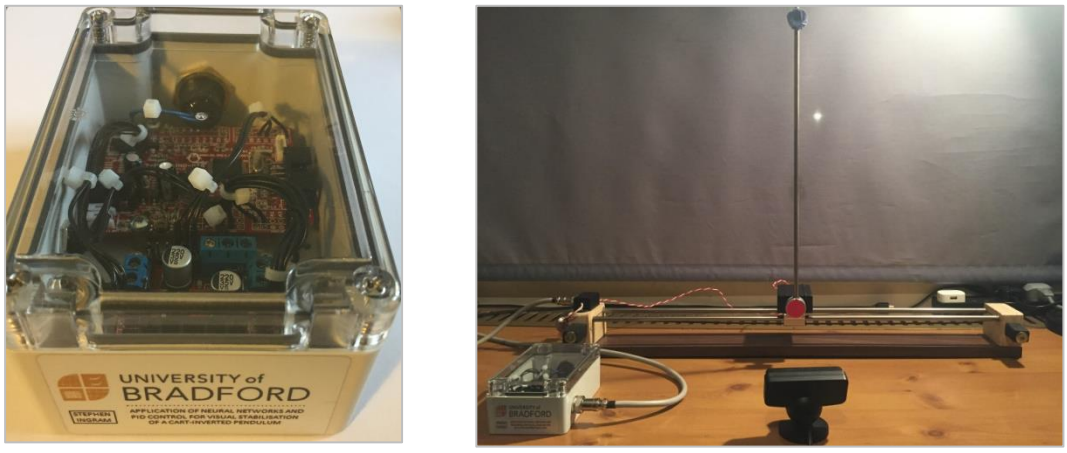


Figure.4.14 *– Bespoke Cart Inverted Pendulum System and Controller Unit.*

4.4.1 Test Setup and Approach

All of the experiments were undertaken using the HP Envy Laptop discussed in Chapter 1. In order to evaluate the performance of the system, several experiments were conducted. The accuracy of the angle measurement was determined and the

resting oscillation was measured. Once the measurement accuracy and resting oscillation was established, we applied several external disturbances to the pole, as the pendulum was stabilised at its equilibrium. We observed the recovery time and the PID controller response by using the test software to plot the output of the angular sensor.

4.4.2 Results Summary

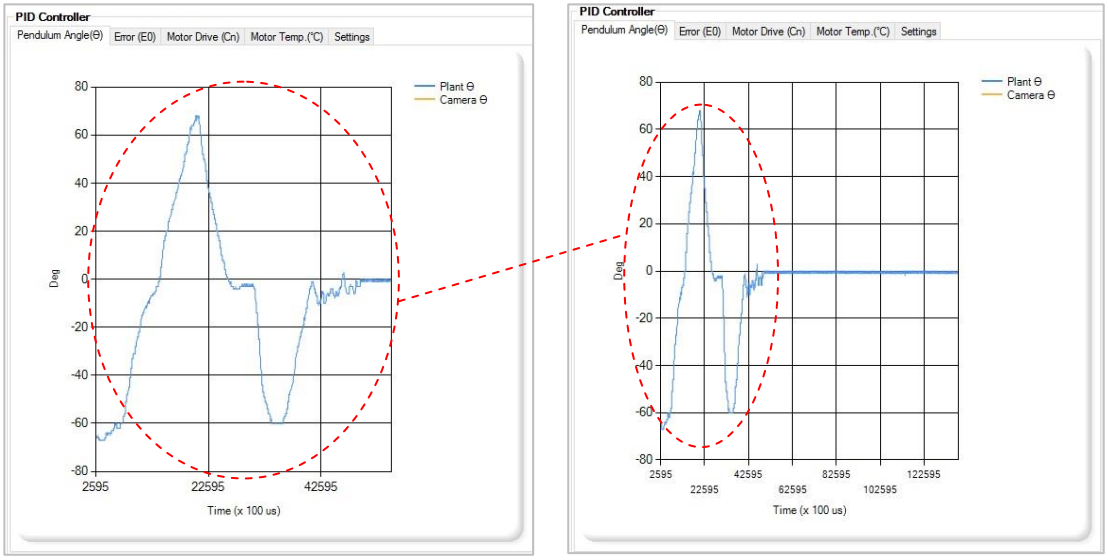


Figure.4.15 *– Cart Inverted Pendulum Angle Measuremnt Tests.*

In the first phase of testing, we slowly increased the pendulum angle from zero (fully vertical) to ±60º, over a period of approximately 30s. The pendulum angle was measured with an inclinometer, and in each instance the angle was found to be correctly measured. The pendulum angle was measured equally, both sides of the vertical point, indicating that there was no asymmetry in the measurement system. The plots that were captured by the test software were monotonic and without

distortion or non-linearity (see Figure.4.15). The Inverted Pendulum angle was successfully measured with an accuracy of approximately 0.3º.

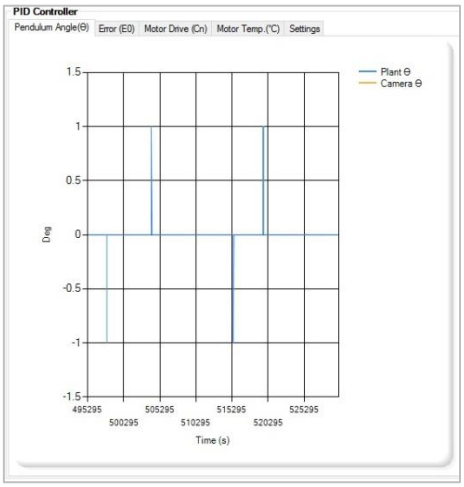


Figure.4.16 *– Cart Inverted Pendulum Resting Oscillation.*

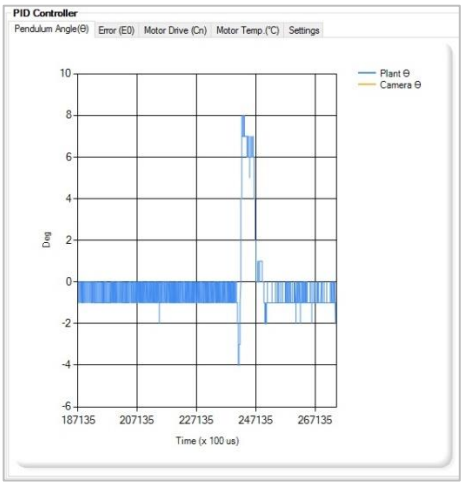


Figure.4.17 *– Cart Inverted Pendulum Disturbance Response.*

In the second phase of testing, we measured the amplitude of the resting oscillations by allowing the pendulum to idle at the unstable equilibrium. Under PID control, we observed resting oscillations of up to ±1º (see Figure.4.16). The average resting oscillation was determined to be approximately 0.3º (which was measured over a period of 60s).

In the third phase of testing, we applied external disturbances to the pendulum pole while it was at its unstable equilibrium. The recovery time and controller response were monitored via the test software. The pendulum was seen to recover from disturbance angles of up to 25º. Figure.4.17 demonstrates the recovery profile of the pendulum when subjected to an 8º disturbance. The pendulum fully recovered within 500ms to the upright position. The results showed that we successfully stabilised the Cart Inverted Pendulum using our bespoke controller unit.

Chapter 5

Cart and Pole Localisation by Multiple-Segments Moments Tracking

In this chapter, we present the theoretical foundation and mathematical representation of our Multiple-Segments Moment Tracking (MSMT) algorithm for locating the Cart and Pole markers within an image. We begin by discussing the theory of moments and define them in discrete notation, which forms the basis of our detection and tracking method. We present a list of moment invariants, some of which are used in the practical implementation of the algorithm. We then briefly discuss the HSV colour space and present the mathematical representation of the chromatic components. We then discuss image thresholding and morphological filtering, both of which are vital to the robust detection of the Cart and Pole markers. We conclude the chapter by presenting our novel Cart and Pole tracking algorithm.

5.1 Theory of Moments

Moments describe numerical quantities at a known distance from a specified reference point. They are frequently used in statistics to describe the scattering of random variables; likewise, they are employed in mechanics to describe bodies by their distribution of mass. In its most elementary form, a moment is the product of the distance to a point, raised to a power, multiplied by a physical quantity, at that point. Usually, in applications such as the one presented in this research, the quantity is not

concentrated solely at a single point; therefore the moment is the integral of that quantity's density over space.

5.1.1 Two-Dimensional Cartesian Moment

As previously mentioned, the quantity is not typically concentrated solely at a single point and the moment is the integral of the density over space. Our research focuses on images that operate in 2D Cartesian space (i.e. that have x and y-axis). The 2D Cartesian moment, m_{pq}, of order p + q, of a density distribution function, f(x, y), is defined as (Hu, 1962):

$$m_{pq} \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^p f(x,y) dx\, dy \tag{5.1}$$

The 2D moment for an (I by J) scalar (greyscale) image, g(x, y), is given by:

$$m_{pq} \equiv \sum_{y=0}^{J-1} \sum_{x=0}^{I-1} x^p y^q g(x,y) \tag{5.2}$$

Since our approach deals with binary image objects the moment is given by:

$$m_{pq} \equiv \sum_{A} x^p y^q \tag{5.3}$$

Where the summation extends over all of the elements in A, i.e. all the "active" pixels within the image array.

A complete set of moments of the order n consists of all moments, m_{pq} , such that p + q ≤ n and contains (n+1)(n+2) / 2 elements.

5.1.2 Hu Moments for Marker Identification and Tracking

Our approach is based on image moments for object identification and tracking. Image moments and moment invariants play a crucial role in object recognition and shape analysis. The use of moments for image analysis and pattern recognition was inspired by Hu (Hu, 1962). His Uniqueness Theorem presented, stated that if f(x, y) is piecewise continuous and has non-zero values in the finite region of the (x, y) plane, then the moments of all orders exist. It can, therefore, be shown that the moment set $\{m_{pq}\}$ is uniquely determined by f(x, y) and conversely, f(x, y) is uniquely determined by $\{m_{pq}\}$. We propose to simultaneously segment the two uniquely coloured markers; the one on the Cart and the other on the Pole. This allows us to uniquely identify the Cart and Pole in every frame. Each image segment has a finite area and is piecewise continuous; therefore, moments of all orders exists and a moment set can be computed that will describe the information contained within the image segment. To describe all of the information continued within an image segment would require a potentially infinite number of moment values. Our approach is to select a meaningful subset of moment values that contain sufficient information to describe the markers within a given image. By maintaining the position of the unique markers in each frame, we achieve tracking of the Cart and Pole.

5.1.3 Fundamental Lower-Order Moments

The lower-order moments represent several well-known (and for our research, essential) geometric properties of a distribution. To highlight these properties and demonstrate the applicability to object identification and tracking, one can regard the moment values of a function that is binary and continuous, i.e. our segmented Cart and Pole markers. The moment values for this distribution can be explained in terms of shape characteristics of the markers.

A. Zero-Order Moments: Area

The definition of the zero-order moment, $\{m_{00}\}$, of f(x, y) represents the mass of the given distribution function or image. When computed for our segmented marker, the zero moment represents the total object area. m_{00} is given by:

$$m_{00} \equiv \iint_{-\infty}^{\infty} f(x,y)dx\,dy \tag{5.4}$$

B. First-Order Moments: Centre of Mass

The two first-order moments, $\{m_{10}, m_{01}\}$, are used to locate the centre of mass (CM) of the Cart and Pole markers. The coordinates of the CM, ($\overline{x}$, $\overline{y}$), is the intersection of the lines, x = $\overline{x}$, and y = $\overline{y}$, parallel to the x and y-axis respectively, where the first-order moments are zero. Alternatively, x = $\overline{x}$ and y = $\overline{y}$ represent areas where all the mass may be concentrated, without change to the first-order moments about the x and

y-axes respectively. Regarding moment values, the coordinates of the CM are given by:

$$\overline{x} = \frac{m_{10}}{m_{00}} \quad \overline{y} = \frac{m_{01}}{m_{00}} \tag{5.5}$$

The CM describes a unique position that may be used as a reference point to describe the position of the markers within the image. If the marker segments are positioned such that the CM is coincident with the origin of the FOV, i.e. ($\overline{x}$ = 0) and $\overline{y}$ = 0), then the moments computed for that marker is referred to as the central moment and is designated by μ_{pq} (Note that $\mu_{10} = \mu_{01} = 0$). Hu derived combinations of moments that are invariant with respect to scale, position and orientation, which are based on the theories of invariant algebra. Seven moment invariants were defined, which were calculated from the central moment. These are given by:

$$\mu_{00} = M_{00} \tag{5.6a}$$

$$\mu_{01} = 0 \tag{5.6b}$$

$$\mu_{10} = 0 \tag{5.6c}$$

$$\mu_{11} = M_{11} - \overline{x}M_{01} = M_{11} - \overline{y}M_{10} \tag{5.6d}$$

$$\mu_{20} = M_{20} - \overline{x}M_{10} \tag{5.6e}$$

$$\mu_{02} = M_{02} - \overline{y}M_{01} \tag{5.6f}$$

$$\mu_{21} = M_{21} - 2\overline{x}M_{11} - \overline{y}M_{20} + 2\overline{x}^2M_{01} \tag{5.6g}$$

$$\mu_{12} = M_{12} - 2\overline{y}M_{11} - \overline{x}M_{02} + 2\overline{y}^2M_{10} \tag{5.6h}$$

$$\mu_{30} = M_{30} - 3\overline{x}M_{20} + 2\overline{x}^2M_{10} \tag{5.6i}$$

$$\mu_{03} = M_{03} - 3\overline{y}M_{02} + 2\overline{y}^2M_{01} \tag{5.6j}$$

It is on this principle that we build our localisation algorithms. The concept of MSMT is one in which the segmented marker moments are superimposed onto into one image, containing multiple moments, from which vital information about the Cart and Pole can be determined. For further information the reader is referred to Hu's paper on Image moments (Hu, 1962).

5.2 HSV Colour Space

The Colour space is an abstract mathematical model, which describes the range of available colours as ordered lists. Due to some unique advantages within the field of computer vision and computer graphics (Ibraheem et. Al., 2012), the Hue, Saturation and Value (HSV) colour space was chosen for our image processing. HSV colour space is related to the group of perception-based colour spaces and is fundamentally based on the human perception of colour.

Figure.5.1 illustrates the standard representation of the HSV colour space. Different shades and colours are defined and represented while moving anti clockwise from 0 to 360 degrees, at which points the colour red can be found. The Saturation (S) value defines the number of pixels available to represent a given colour. S increases towards the edge of a circular cross section of the cone and decreases towards the center. The highest and lowest saturation of a colour is represented by 100% and 0% respectively. The Value (V) contains the information related to the brightness or darkness of a pixel. V increases when moving towards pixels with higher intensity

values and vice versa. The maximum and minimum intensity values of a colour are represented by 100% and 0%, respectively.

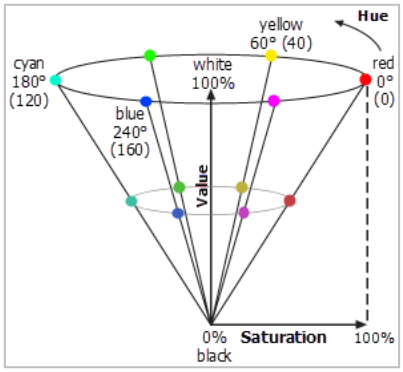


Figure.5.1 *– HSV Colour Space Representation.*

An image represented in the BGR colour space (as natively provide by our USB camera) can be converted to a HSV colour space image by applying the following equations:

$$H = \cos^{-1}\left(\frac{0.5(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{5.7}$$

$$S = 1 - \left(\frac{3}{R+G+B}\right)\min(R,G,B) \tag{5.8}$$

$$V = \max(R,G,B) \tag{5.9}$$

There are intrinsic dependencies between the three HSV components. The H component will have no significance when the S or V values are represented by the lowest value, i.e. 0%. The colour will be shown as black if the V component is represented by the lowest value. A pure white colour is obtained when the V

component is represented by the highest value, i.e. 100%; and S component is represented by the lowest value, i.e. 0%. For further information on the HSV colour space the reader is referred to the work of Ibraheem et al. (2002).

5.3 Segmentation by Thresholding

As discussed in section 4.1, segmentation is required to isolate the Cart and Pole markers within an image. The term segmentation, when used in the context of image processing means: separating the image into background and object of interest. The object of interest, in this case, is the Cart and Pole markers.

Image thresholding is one of the most effective and simplest segmentation techniques to implement (Sahoo et al., 1988). In this method, structures within the image (i.e. markers) are segmented by comparing their intensity value to one or more intensity thresholds. An image can be segmented into two regions by using only one threshold value; however, it is more common to segment an image into multiple regions using multiple thresholds (Sahoo et al., 1988). This multiple thresholding concept is adopted to generate the compound marker segments in our method. Figure.5.2 illustrates the concept of thresholding and how it is determined by the histogram shape of an image.

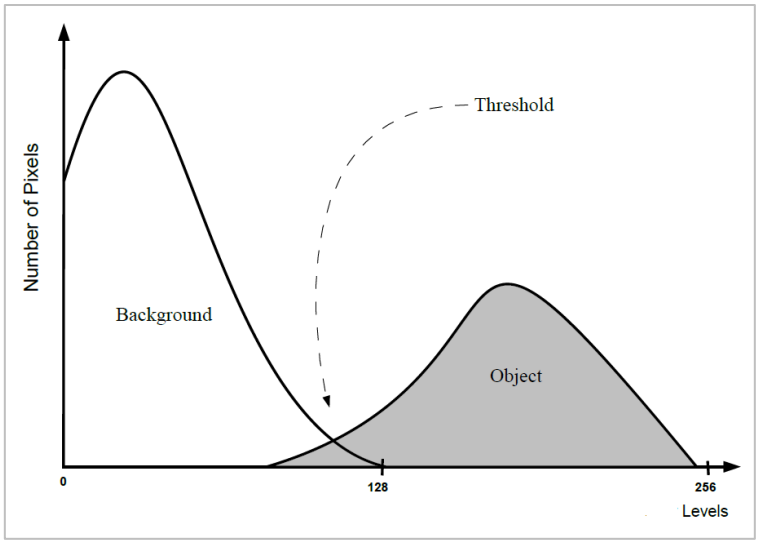


Figure.5.2 – *Thresholding Example.*

We apply a simple thresholding approach, where the image is searched pixel by pixel and the following mathematical rule is applied:

If function f(x, y) > T then function f(x, y) = 0 else function f(x, y) = 255.

This provides a binary image to which the theory of moments can be applied to determine information about the markers. Often, the image may contain unwanted or incomplete segments; it is, therefore, necessary to perform some morphological image restoration.

5.4 Morphological Filtering

Morphological operations are non-linear functions, which are used to describe the shape and structure of objects within an image (Meyer and Beucher, 1990). Cart and Pole marker segmentation is challenging due to the complex shape and structure of

the objects produced due to lighting changes. Morphological operations can be used in this case to regulate the shape of the marker objects. Some of the most commonly used morphological operations are; erosion, dilation, opening and closing operations, which are used to regulate the marker object. Pixel connectivity is determined based on neighbourhoods of several pixels known structure set. This function is applied to remove small particles from within the images. This procedure is required to prevent noise objects being considered as cart or pole markers.

5.4.1 Erosion and Dilation

Erosion is an operation which takes an average image A, and structuring element, S, to construct a new image B. The structuring element is simply a small image, typically a 3×3 set of pixels. The erosion, A $\ominus$ S, then, is the act of placing the centre of the structuring element on each pixel of the input image and determining if this local image is identical to the structure element. If so, a pixel is placed in that position of the output image. Mathematically, this can be expressed as follows:

$$A \ominus S = \{(x,y) \mid S(x,y) \subset A\} \tag{5.10}$$

where $S(x,y)=\{(x+i,y+j) \mid (i,j) \in S\}$ is just the translation of S to position (x,y).

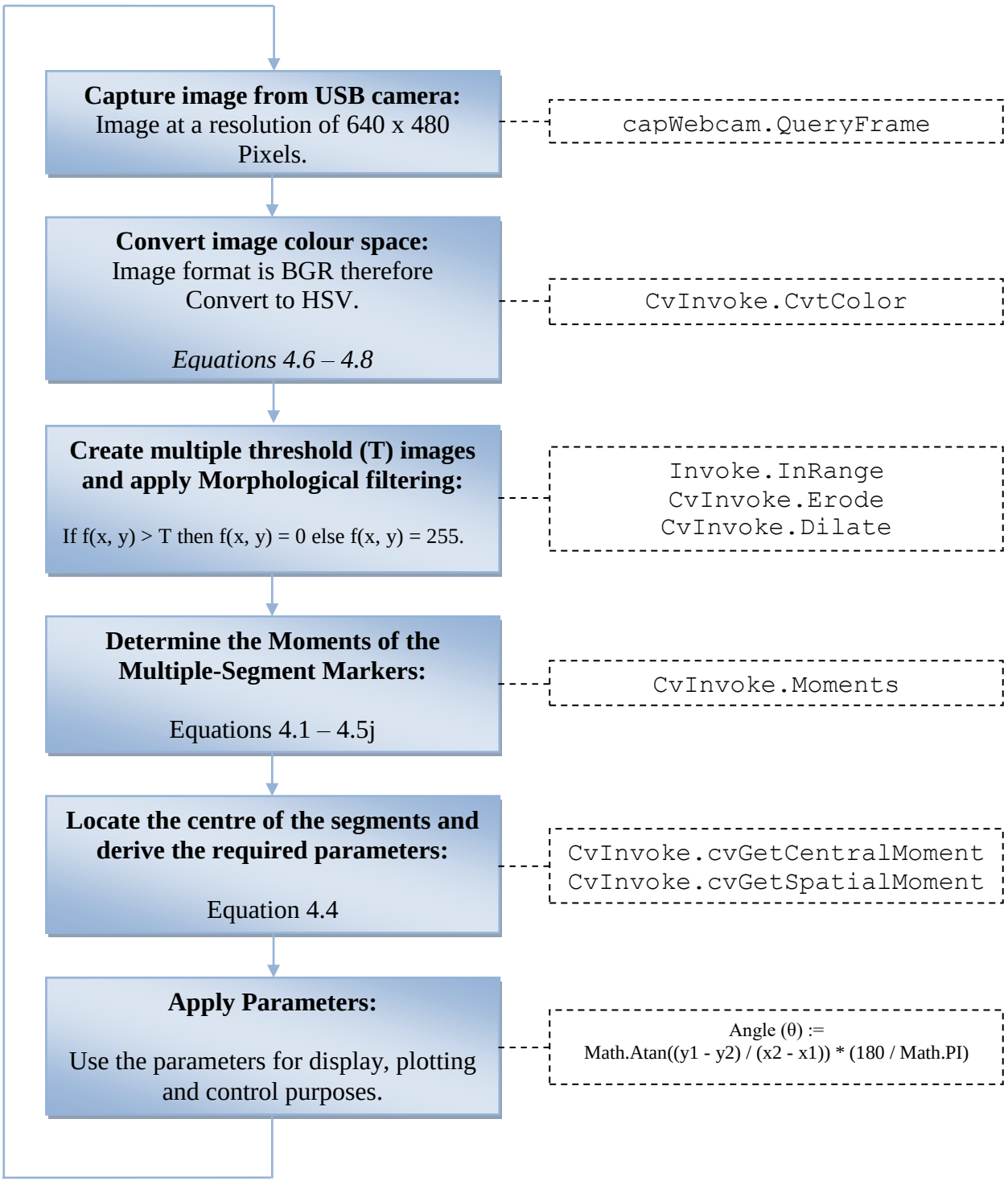
Dilation is simply the opposite of erosion and is defined as follows:

$$A \oplus S = (Ac \ominus Sc) \tag{5.11}$$

As the name implies, this operation expands the object in the image, according to a structuring element.

5.5 Localisation and Tracking Algorithm

Our novel identification and tracking algorithm is shown below, alongside EmguCV realisation of the key parts:



5.6 Implementation and Results

The Cart and Pole were fitted with two uniquely coloured markers. The marker colours were chosen by experimentation, based on two colours that were found to be individually identifiable with the least amount of image noise. In our experiments we chose blue and red, as they gave good results during testing. The software tool discussed in Chapter 4 was used to implement the algorithm presented in Section 4.4. The results are illustrated in the following subsections.

5.6.1 Capture Image and Convert Colour Space

Figure.5.3 shows an image captured by the USB camera and converted from BGR to HSV colour space. This was achieved by invoking the "capWebcam.QueryFrame" and "CvInvoke.CvtColor" functions in EmguCV.

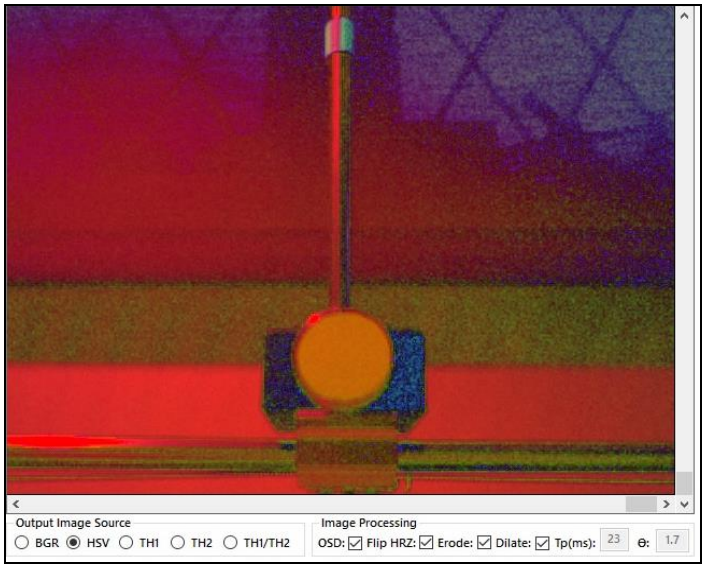


Figure.5.3 *– Capture Image and Convert Colour Space (HSV Image).*

5.6.2 Create Multiple-Segments and Apply Filters

Figure.5.4 and 5.5 illustrates the result of applying thresholding to the image based on the unique markers. The result is two separate images, which contain a segmented marker; one for the Cart (Figure.5.4) and one for the Pole (Figure.5.5).

Figure.5.4 – Segmented Image of the Cart Marker.

Figure.5.5 – Segmented Image of the Pole Marker.

Morphological Dilation and Erosion was applied to both images and the original images were overwritten, with new filtered copies. This improved the reliability of the segmentation and ensured that false segments were removed. The Morphological structuring element for both Dilation and Erosion were determined experimentally. The individual images were then combined to give a composite, Multiple-Segments image as shown in Figure.5.6. This image was then used for further processing.

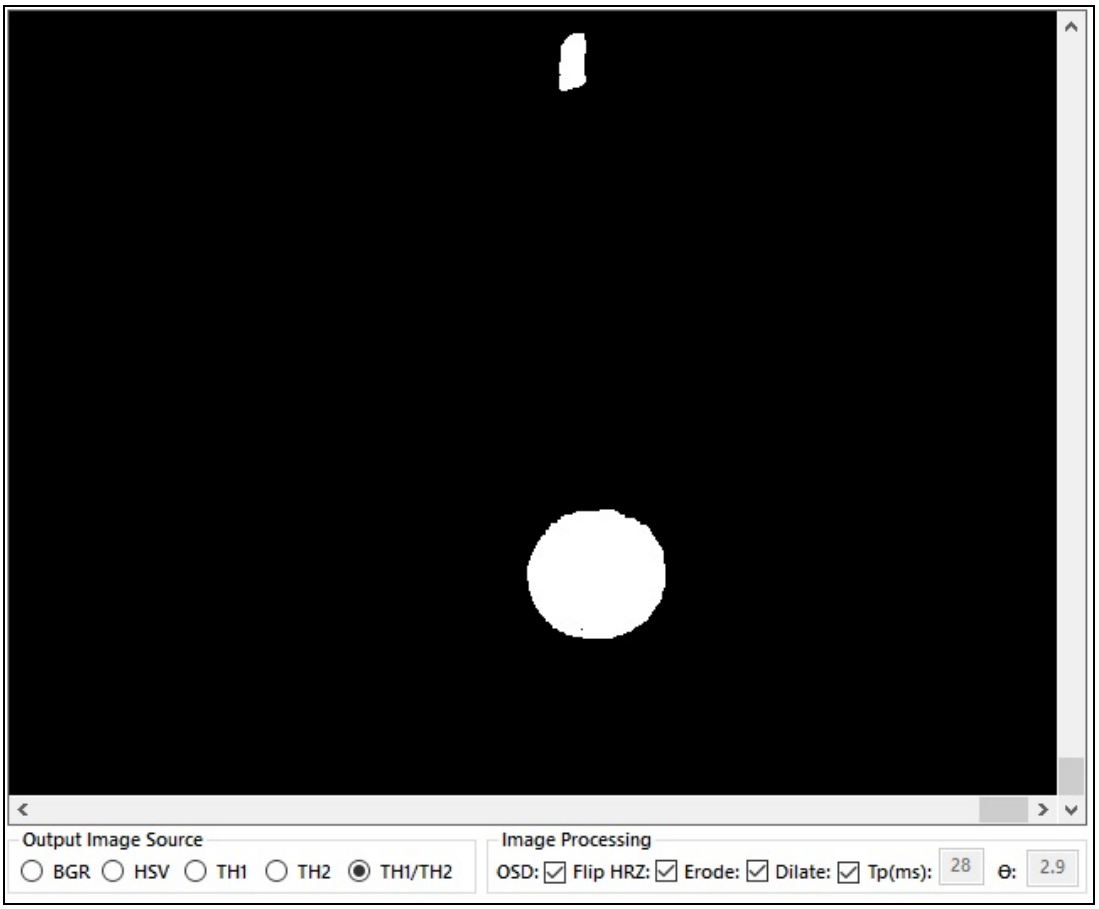


Figure.5.6 *– Composite Multiple-Segments Image.*

5.6.3 Robustness to Visual Disturbances

To demonstrate the robustness of the algorithm to visual disturbances, the pendulum was allowed to idle about the unstable equilibrium and different coloured objects were used to apply a disturbance to the Pole. The performance was excellent; the marker tracking algorithm rejected all non-marker coloured objects in the FOV of the camera (see Figure.5.7).

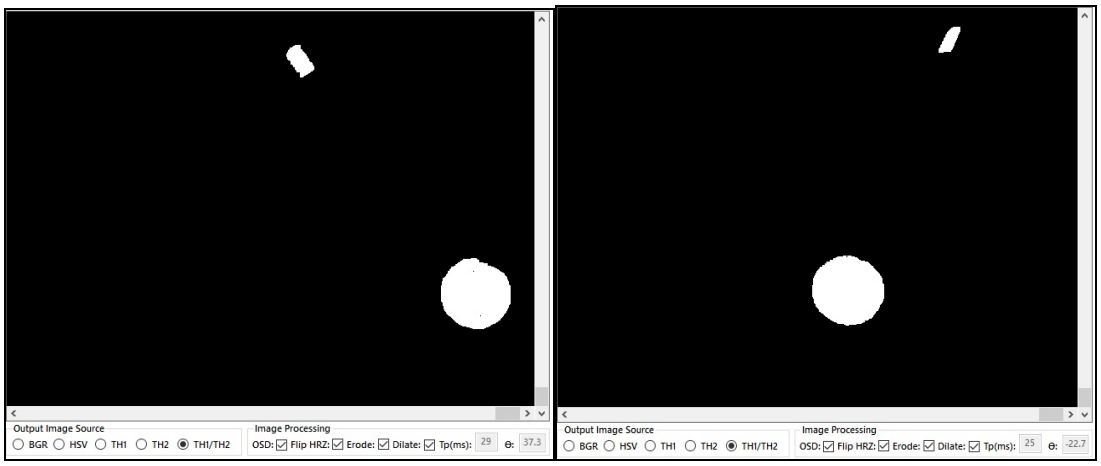


Figure.5.7 *– Application of Visual Disturbances.*

The disadvantage, however, was that if a marker coloured object was introduced within the FOV of the camera, then confusion would arise and the tracking system would fail. We called this the “Marker Coloured Disturbance” problem.

5.6.4 Vision-Based Measurement and Control

In a similar approach to the PID controller testing that was performed in Chapter 4, the accuracy of the visual angle measurement system was established and the resting oscillation under vision based control was determined. In the first phase of testing, we slowly increased the pendulum angle from zero (fully vertical) to ±45º, over a period of approximately 30s. The pendulum angle was compared to the potentiometer measured angle; in each case the angles were found to measuring within 0.8º of each other (see Figure.5.8 to Figure.5.12).

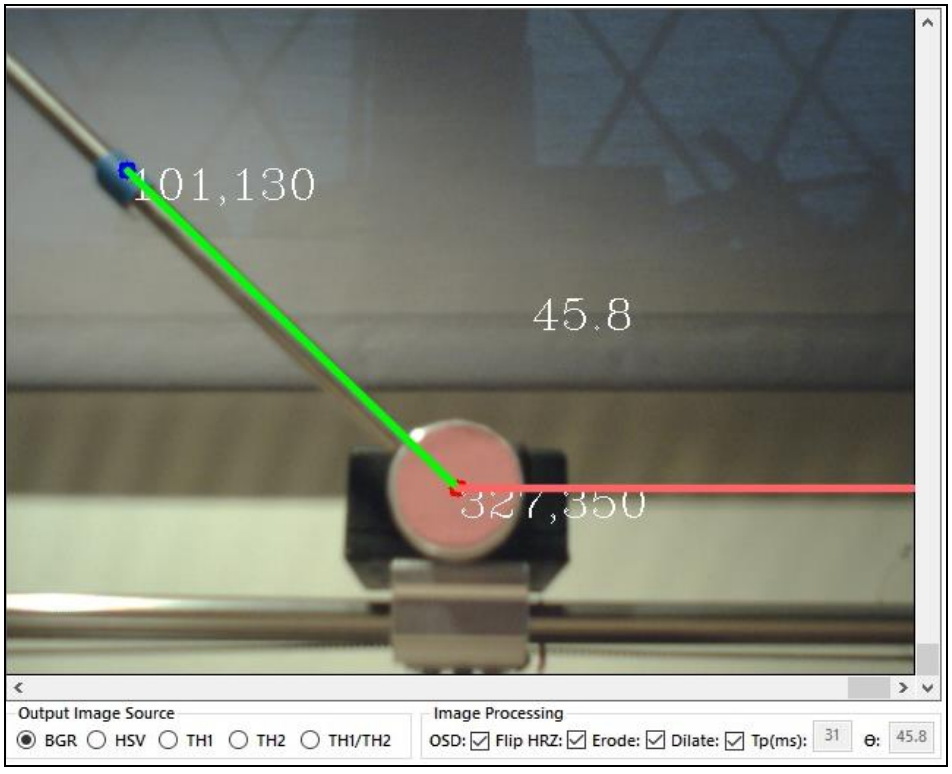


Figure.5.8 *– Visual angle measurement test – Pendulum angle of 45º measured visually as 45.8º, 1.77% Error.*

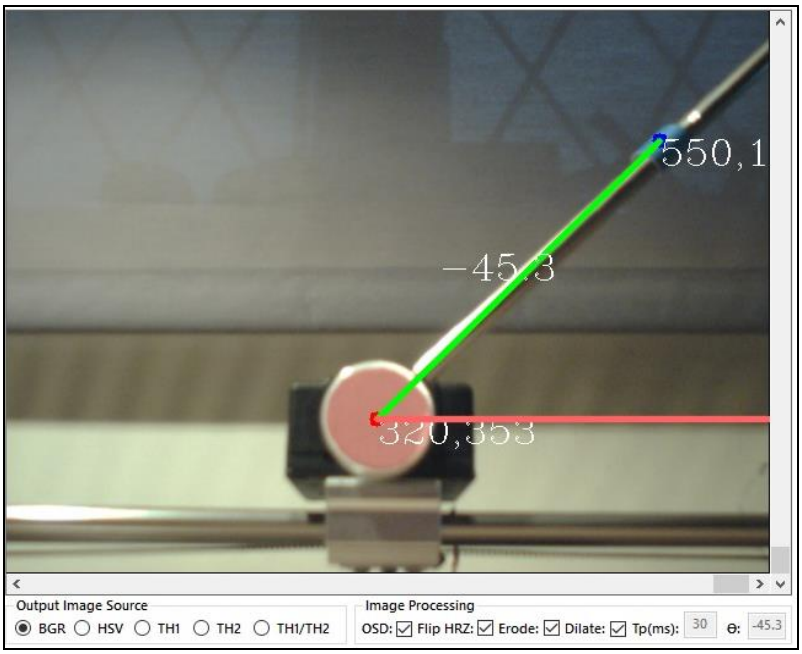


Figure.5.9 *– Visual angle measurement test – Pendulum angle of -45º measured visually as -45.3, 0.67% Error.*

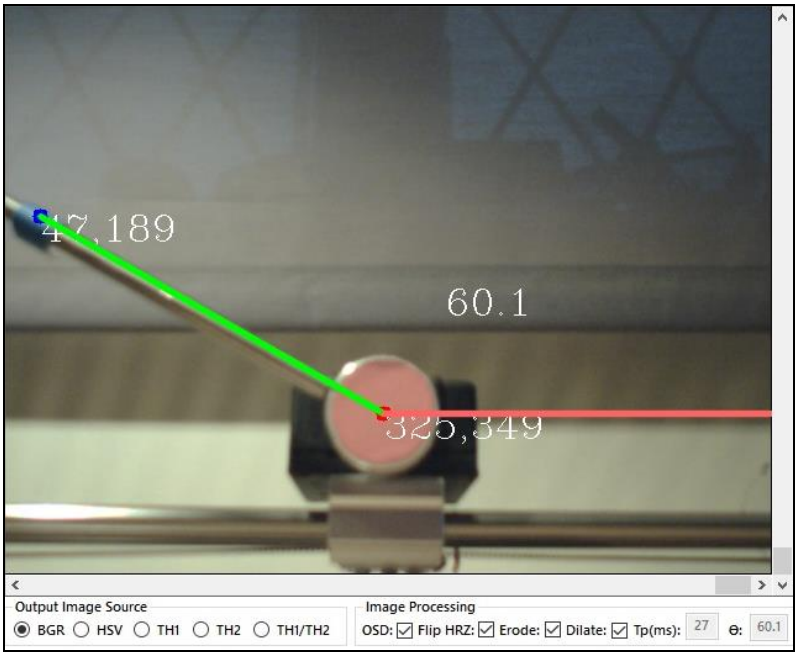


Figure.5.10 *– Visual angle measurement test – Pendulum angle of 60º measured visually as 60.1º, 0.17% Error.*

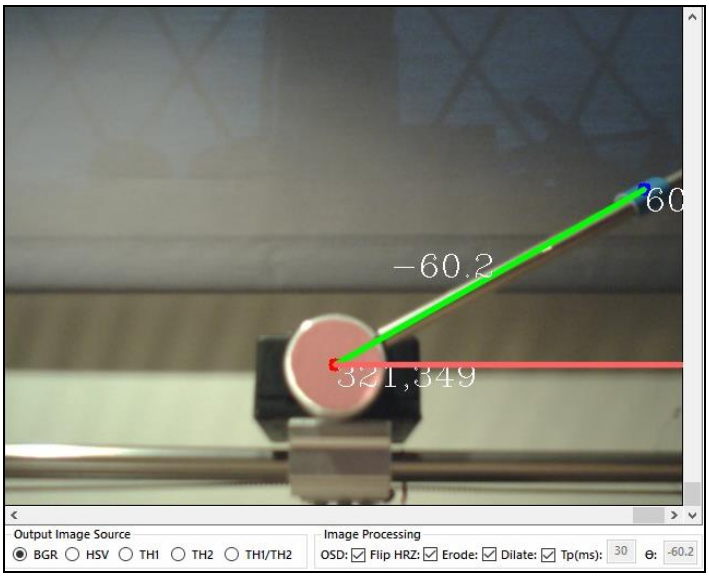


Figure.5.11 *– Visual angle measurement test – Pendulum angle of -60º measured visually as -60.2º, 0.33% Error.*

In the second phase of testing, we measured the amplitude of the resting oscillations under vision-based control by allowing the pendulum to idle at the unstable equilibrium. We observed resting oscillations between 2.0º and 5.8º (see Figure.5.12). The pendulum was able to recover from small disturbances that were limited to ±10º.

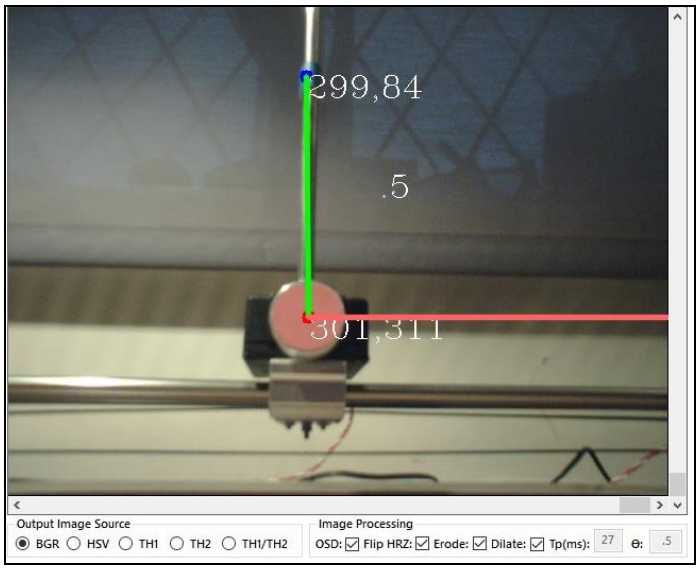


Figure.5.12 *– Visual Control: Resting Oscillation Test.*

Chapter 6

Pendulum Angle Estimation by Artificial Neural-Networks

In the previous chapter, we demonstrated a computational method of pendulum angle estimation by using our MSMT method. In this chapter, we present an alternate technique for measuring the pendulum angle by means of Artificial Neural-Networks. We start by building on the foundation presented in Chapter 3, focusing on the Multi-Layer Perceptron as a function approximation tool. We then design our Neural-Network for pendulum angle estimation and train it using harvested data. We conclude the chapter by evaluating the performance of the Neural-Network solution and compare it to the MSMT method presented in the previous Chapter.

6.1 Function Approximation

Learning the unique mapping between the input and output space from a set of data is the primary concern in many real-world applications. Such a problem usually occurs when it is costly, either in terms of time or complexity to compute the true function, or when this function is unknown. In place of an explicit formula to denote the function f(x), only pairs of input-output data in the form of (x, f(x)) are available. If we let:

$$x_i \in R^m, i = 1, 2, \dots, N \tag{6.1}$$

and

$$d_i \in R^1, i = 1, 2, \dots, N \tag{6.2}$$

where N is the number of input vectors with dimension m, and N real number outputs respectively. The goal is to determine the unknown function f(x):Rm→R1 that satisfies the interpolation where:

$$f(x_i) = d_i, \ i = 1, 2, \dots, N \tag{6.3}$$

The accuracy of the fitness of d_i by the function f(x) is given by an error function. A commonly used error function is defined by:

$$E(f) = \frac{1}{2}\sum_{i=1}^{N}(d_i - y_i)^2 \tag{6.4}$$

Where y_i is the actual response (f(x)). The main objective is to minimise the error function E(f) to enhance the accuracy of the estimation. This is the main objective of function approximation.

6.2 Artificial Neural-Networks for Function Approximation

As we briefly discussed in Chapter 2, ANNs have been shown to be particularly good at number of different tasks. They have been widely applied to solve many problems, including non-linear function approximation. An ANN is characterized by its architecture, learning algorithms and activation functions. The architecture describes the interconnections between the neurons. They typically consist of an input layer, an output layer and one or more hidden layers. Hornik (Hornik, 1989) and Cybenko (Cybenko, 1989) have shown that a three layer ANN is capable of approximating any

arbitrary, non-linear function f(x) with any desired degree of accuracy. Consequently, ANNs have been applied in various applications, especially in applications related to parameter estimation; this is primarily due to its ability to find the association between input-output data without the need for predetermined models. The most common ANN structure is the MLP (Rumelhart et al., 1986). Figure.6.1 illustrates the structure of a MLP.

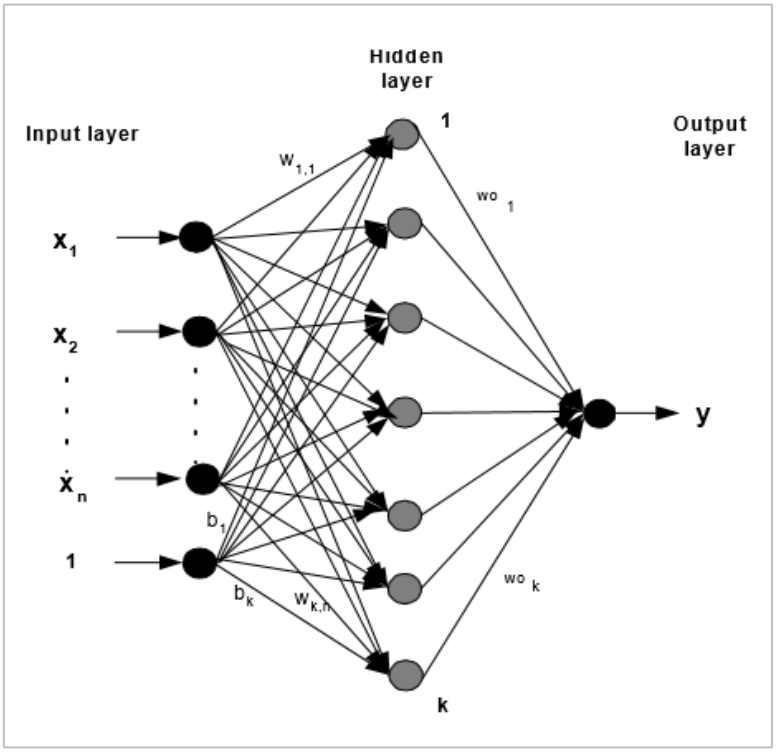


Figure.6.1 *- Typical Multilayer Feed-Forward ANN.*

The output node, j, in the hidden layer is given by equation (2.22) in Chapter 2 and the output of the network is given by:

$$y = \sum_{i=1}^{k}(wo_i h_i) \tag{6.5}$$

where w_{ji} are the weights connecting the input values to node j in the hidden layer, and $wo_i$are the weights from the hidden layer to the output layer. Depending on the learning algorithm applied, the ANN can be categorized as one of the following:

1. Fixed Weight ANNs: These do not need any form of learning.

2. Unsupervised ANNs: These networks are trained (weights are adjusted) based on input data only. The networks learn to adapt using experience gained from previous inputs.

3. Supervised ANNs: These are the most commonly applied ANNs. In these networks, the system employs both input and output data. The weights and biases are updated for every set of input/output data pair.

Our MLP falls into the supervised learning category.

The activation function relates the output of a neuron to its input, based on the neuron's input activity level. Some of the commonly used activation functions include: the threshold, piece-wise linear, sigmoid, tangent hyperbolic, and the Gaussian function. The learning process of the MLP network involves using the input-output data to determine the weights and biases. One of the techniques used to obtain these parameters is the back-propagation algorithm (see Chapter 2). In this method, the weights and biases are adjusted iteratively to achieve a minimum MSE between the network output and target value. The ANN development process that we followed is shown in Figure.6.2.

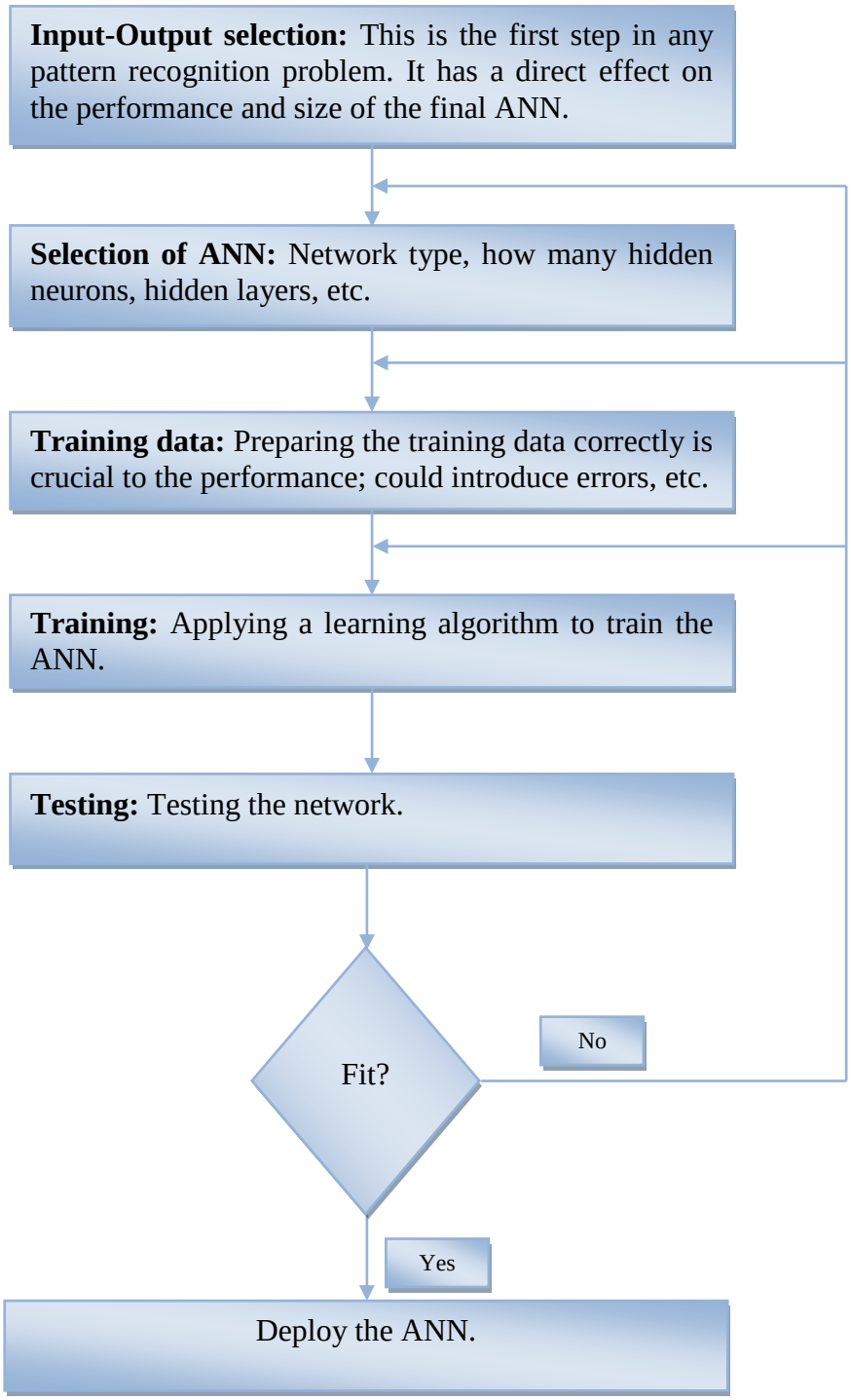


Figure.6.2 – *Neural-Network Development Process.*

6.3 Development of the ANN for Pendulum Angle Estimation

The purpose of our ANN is to estimate the angle of the pendulum by using the Cartesian Coordinates of the Cart and Pole marker segments (see Chapter 5). A three-layer MLP was used for the purpose of pendulum angle estimation. We followed the development procedure outlined in Figure.6.2.

1. Input-Output and Hidden Layer Selection

The inputs that are required by our network are the four Cartesian Coordinates of the Cart and Pole markers, as described in Chapter 5. The output (θ) of the neural network model consists of a single neuron, which represents the pendulum angle for the specific Cartesian Coordinates. That is:

$$\theta = f((x1, y1), (x2, y2)) \tag{6.6}$$

In the input layer, the number of neurons is equal to the number of coordinates, i.e. four (X1, Y1; X2, Y2). The number of neurons in the hidden layer was determined experimentally, by studying the network behaviour during the training process; taking into consideration some factors like convergence rate, error criteria, etc. Different configurations were tested and the best configuration was selected based on the accuracy level required. The activation function used in this layer was the Sigmoid due to its efficiency in nonlinear approximation tasks. The output layer has only one neuron, which represents the pendulum angle.

2. Network Training

The ANN was trained offline by using data that was harvested during the experimentation phase (Chapter 4). X1, Y1 and X2, Y2 coordinates were collated into training files, along with the actual pendulum angle measured by the potentiometer. These input-output groups were used to train the ANN using the Fast Artificial Neural Network (FANN) library (see Figure.6.3). Various training algorithms were tested and the RPROP (Riedmiller and Braun, 1993) was ultimately chosen as it gave the fastest convergence and lowest MSE. The initial weights were randomised with values between 0 and 1. The Cartesian inputs were normalised to have values between 0 and 1 (this sped up the convergence). The ANN was trained until the MSE was determined to be low enough for the network to be applied to solving the problem.

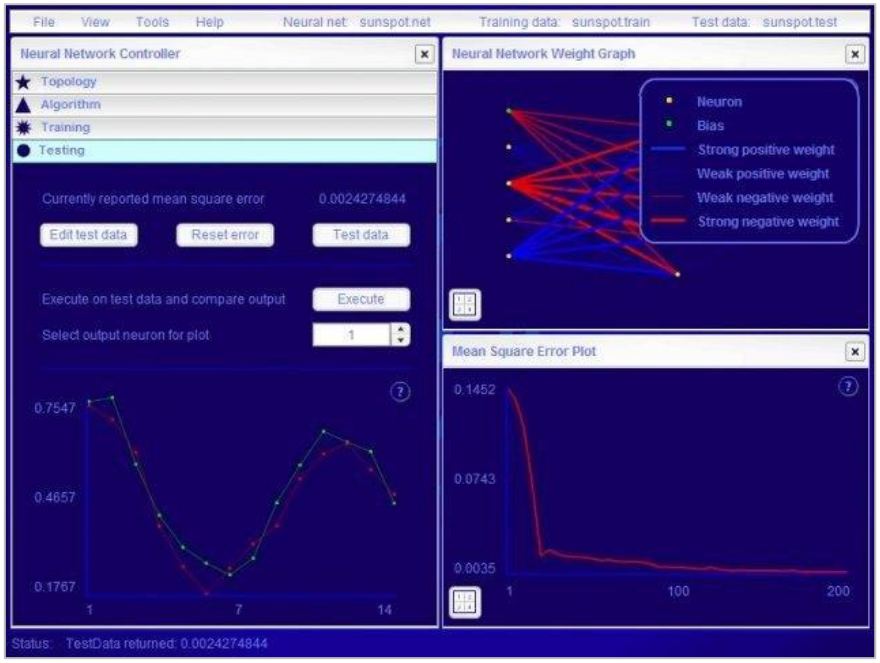


Figure.6.3 – *FANN Toolbox: Example Window.*

3. Network Testing

The ANN was initially tested by generating an artificial sinusoidal signal, which represented the smooth movement of the pendulum from left to right, about the vertical axis. The simulated Coordinates were fed to the network and the output angle was compared to the ideal input, and quantified for accuracy. A second test was performed, which applied previously "un-seen" segment coordinates to the network inputs and comparing the output result with the actual pendulum angles stored in the harvested data.

6.4 Results

During initial testing using the simulated coordinate data, the accuracy was determined and the ANN was found to perform extremely well. The results showed that the network was able to estimate the pendulum angle to within 0.4º of the actual angle (see Figure.6.4). This was significantly better than the MSMT method presented in Chapter 5. In the subsequent tests, using previously "un-seen" coordinates from the harvested data, the ANN was found to accurately estimate the pendulum angle with a similar degree of accuracy.

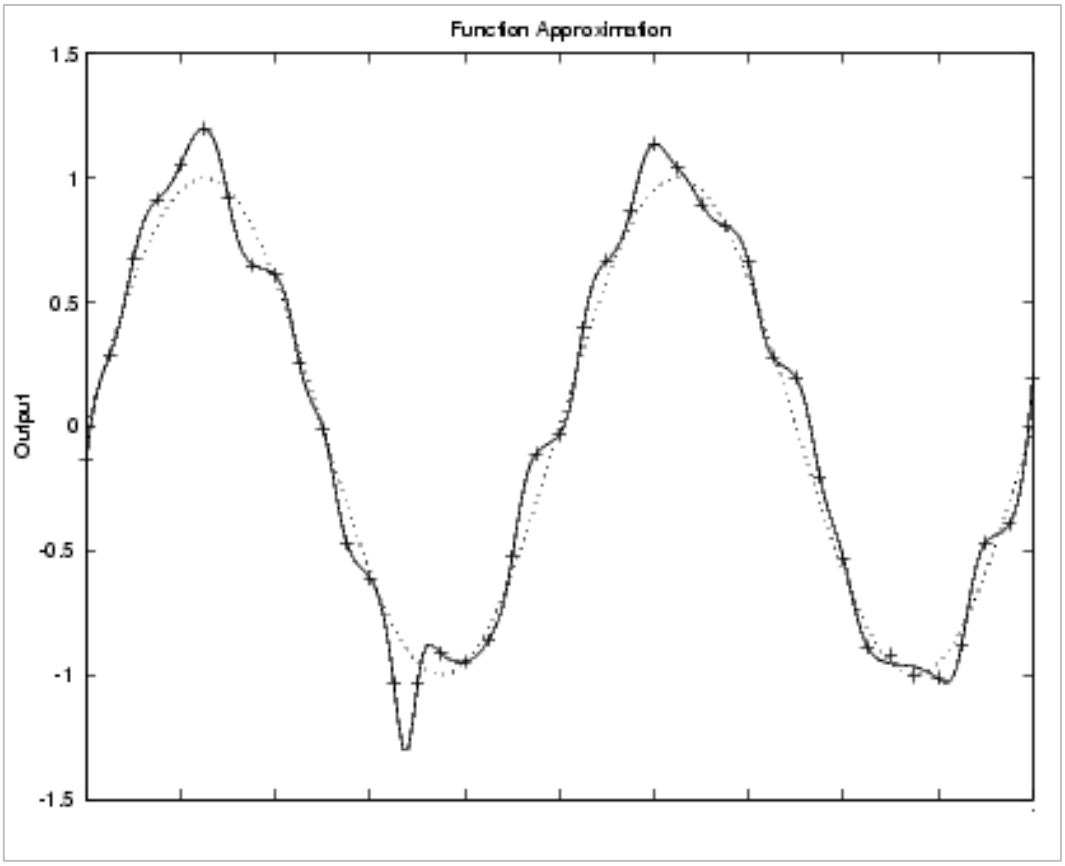


Figure.6.4 – Function Approximation Results.

The use of the Artificial Neural-Networks to estimate pendulum angle from Cart and Pole segment coordinates is presented in this chapter. Results obtained with the help of a training data is given and compared to those obtained using the MSMT method. The results indicate that the proposed ANN approach significantly out performs the MSMT approach.

Chapter 7

Conclusion and Further Work

7.1 Summary

Dynamic object balancing by means of visual feedback control is a complex and challenging task. Within the context of this research, we employed the classic Cart Inverted Pendulum, an inherently unstable, non-linear system to investigate vision-based control. During our research, we encountered many solutions applied to solving the Inverted Pendulum stabilisation problem using traditional feedback techniques; however, due to the unique challenges presented, relatively few approaches have been reported that use visual feedback for control. Additionally, due to the high cost of commercial Inverted Pendulum systems, many of the solutions that were reported in the literature were validated by means of simulation.

Our research confronted the problem of vision-based Inverted Pendulum control, which was made even more challenging by real-world implementation of a low-cost test system. Our test system employed a standard USB camera to capture images; this was particularly problematic as the camera was found to have poor colour-balance and significant pixel noise in low light conditions. Furthermore, the inherent instability of the Inverted Pendulum meant that external disturbances had significant, unpredictable, effects on its motion. This suggested the use of a segment-based location and tracking method, since image segments are particularly immune to noise.

We successfully addressed these challenges by developing novel techniques for pose estimation; and by applying them efficiently, we ensured Real-Time operation. The resulting system was able to achieve Real-Time visual stabilization of the Cart Inverted Pendulum, even in the presence of visual disturbances.

7.2 Contributions

The thesis has made the following contributions:

i. **A comprehensive test platform**: One of our research questions was *can we implement a real-world (i.e. outside of simulation), vision-based control system that is capable of balancing an unstable object*? During the early stages, we discovered that the cost of commercial Inverted Pendulum systems was prohibitively high. A custom made test system was therefore developed in Chapter 4, which allowed us to evaluate our novel, vision-based control algorithms. The test system was made even more challenging by the use of low-cost components, to promote reproduction of our work and to stimulate further research. We successfully developed a low-cost system that was able to balance the Inverted Pendulum with a resting oscillation of ±1º.

ii. **Complete test software environment:** We developed a complete graphical software environment using Microsoft Visual Basic .NET, in which our algorithms could be tested. This software allowed us to switch between traditional feedback control for data harvesting and vision-based control for algorithm testing. The software provided a multi-threaded architecture to assist with the Real-Time

requirements of the Inverted Pendulum system. A flexible tracking model generator was implemented to allow customization of the visual markers used by the system. This improved the robustness of the pose estimation.

iii. **Novel tracking and pose estimation algorithm:** We proposed a novel tracking and pose estimation algorithm, which was able to locate the cart and the pole from an image stream using visual marker tracking. The robustness of the tracking and pose estimation against visual disturbances was successfully demonstrated. The algorithm was able to detect the position and angle of the pendulum in Real-Time, with minimum amount of error.

iv. **Novel application of our pose estimation algorithm:** We successfully applied our pose estimation algorithm to the Inverted Pendulum stabilisation problem. We achieved pendulum oscillations of between 2° and 5°, which sits in the median of that reported in the literature using other methods.

v. **Novel ANN optimization method:** We successfully optimized our method by means of Artificial Neural-Networks to achieve a significant improvement in the angle measurement accuracy.

7.3 Further Work

Although the system presented in this thesis was shown to work reasonably well, there are some areas that require further investigation and improvement:

i. The thesis has only addressed the Cart Inverted Pendulum stabilisation problem. The position control and swing-up problems have not been solved visually in this research. Solutions to these problems have been reported in the literature; however, they have been found to be lagging behind the stabilisation problem, in terms of attention. Researchers such as Wang et al. (2008) and Stuflesser and Brandner (2008) are reporting positive results.

ii. Although our solution was shown to be robust to visual disturbances, due to the Real-Time requirement of the system, the amount of filtering that could be performed was limited. Further research is therefore possible into a more effective filtering to achieve greater robustness. The Kalman filter could be explored further and applied to smoothing the pose estimation (pose filtering).

iii. The system presented in this research employed visual markers to aid with the identification and tracking of the cart and pole. Visual markers are a limitation and eliminating them would improve the robustness of the method. Also, this would eliminate the "marker colored disturbances" problem discussed in Chapter 4. Approaches along the lines of Stuflesser and Brandner (2008) are very promising.

iv. The Cart Inverted Pendulum system developed during this research is limited by mechanical friction, due to hand crafted parts. Further work is certainly possible to reduce this. A standard set of mechanical parts could be captured using Computer Aided Design (CAD), which could be made available for 3D printing. This would facilitate further research into the Inverted Pendulum and vision-based stabilisation.

References

Andreff, N., Espiau, B. and Horaud, R., 2002. Visual servoing from lines. *The International Journal of Robotics Research*, *21*(8), pp.679-699.

Angeli, D., 2001. Almost global stabilization of the inverted pendulum via continuous state feedback. *Automatica*, *37*(7), pp.1103-1108.

Åström, K.J., 2002. Control system design lecture notes for ME 155a. *Department of Mechanical and Environmental Engineering University of California Santa Barbara*, p.333.

M. Armstrong & A. Zisserman, 1995. Robust Object Tracking. In Proc. Asian Conference on Computer Vision, vol. I.

Atmel, 2016. Discrete PID Controller on tinyAVR and megaAVR devices. www.Atmel.com. Accessed May, 2016.

Berger, M., Auer, T., Bachler, G., Scherer, S. and Pinz, A., 2000. 3D model based pose determination in real-time: strategies, convergence, accuracy. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000* (Vol. 4, pp. 567-570). IEEE.

Olfa Boubaker, 2014. The inverted Pendulum: A Fundamental Benchmark in Control Theory and Robotics. National Institute of Applied Sciences and Technology, INSAT. 2014.

Bradski, G., and Kaehler, A., 2008. Learning OpenCV: Computer Vision with the OpenCV. ISBN-13: 978-0596516130. 2008.

J. Canny, 1986, A Computational Approach to Edge Detection. IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), 679–698. 1986.

Cervera, E., Del Pobil, A.P., Berry, F. and Martinet, P., 2003. Improving image-based visual servoing with three-dimensional features. *The International Journal of Robotics Research*, *22*(10-11), pp.821-839.

Francois Chaumette, 2014. Visual Servoing. K. Ikeuchi. Computer Vision: A Reference Guide, Springer, pp.869-874, 2014.

Chaumette, F. and Marchand, E., 2004. Recent results in visual servoing for robotics applications. In *8th ESA Workshop on Advanced Space Technologies for Robotics and Automation, ASTRA 2004.* (pp. 471-478).

Chaumette, F. and Hutchinson, S., 2006. Visual servo control. I. Basic approaches. *IEEE Robotics & Automation Magazine*, *13*(4), pp.82-90.

Cham, T.J. and Rehg, J.M., 1999. A multiple hypothesis approach to figure tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)* (Vol. 2, pp. 239-244). IEEE.

Chatterjee, D., Patra, A. and Joglekar, H.K., 2002. Swing-up and stabilization of a cart–pendulum system under restricted cart track length. *Systems & control letters*, *47*(4), pp.355-364.

Chen, C.C. and Wong, C.C., 2002. Self-generating rule-mapping fuzzy controller design using a genetic algorithm. *IEE Proceedings-Control Theory and Applications*, *149*(2), pp.143-148.

Comaniciu, D., Ramesh, V. and Meer, P., 2000. Real-time tracking of non-rigid objects using mean shift. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)* (Vol. 2, pp. 142-149). IEEE.

Das, S.K. and Paul, K.K., 2011. Robust compensation of a Cart–Inverted Pendulum system using a periodic controller: Experimental results. *Automatica*, *47*(11), pp.2543-2547.

Davison, A.J., 2003, October. Real-time simultaneous localisation and mapping with a single camera. In *Iccv* (Vol. 3, pp. 1403-1410).

Dementhon, D.F. and Davis, L.S., 1995. Model-based object pose in 25 lines of code. *International journal of computer vision*, *15*(1-2), pp.123-141.

Dunnigan, M.W., 2001. Enhancing state-space control teaching with a computer-based assignment. *IEEE Transactions on Education*, *44*(2), pp.129-136.

Milner, T.E., Franklin, D.W., Imamizu, H. and Kawato, M., 2006. Central representation of dynamics when manipulating handheld objects. *Journal of neurophysiology*, *95*(2), pp.893-901.

Burdet, E., Franklin, D.W. and Milner, T.E., 2013. *Human robotics: neuromechanics and motor control*. MIT press.

Emgu CV, 2010. Emgu CV: OpenCV in .NET (C#, VB, C++ and More). http://www.emgu.com/wiki/index.php/Main_Page. Accessed April, 2015.

Espinoza-Quesada, E.S. and Ramos-Velasco, L.E., 2006, August. Visual servoing for an inverted pendulum using a digital signal processor. In *2006 IEEE International Symposium on Signal Processing and Information Technology* (pp. 76-80). IEEE.

Friedland, Bernard, 1987, Control System Design. New York: McGraw-Hill, pp 30-52.

Fu, C.Y., Wang, C.K. and Park, E., 2015. A Survey of Computer Vision Research for Automotive Systems, Proceeding of University of North Carolina at Chapel Hill, May 2015.

Garcia, J.P.F., Ribeiro, J.M.S., Silva, J.J.F. and Martins, E.S., 2005. Continuous-time and discrete-time sliding mode control accomplished using a computer. *IEE Proceedings-Control Theory and Applications*, *152*(2), pp.220-228.

Gawthrop, P.J. and Wang, L., 2006. Intermittent predictive control of an inverted pendulum. *Control Engineering Practice*, *14*(11), pp.1347-1356.

Ghez, C. and Krakauer, J., 2000. The organization of movement. *Principles of neural science*, *4*, pp.653-73.

Chesi, G. and Hashimoto, K. eds., 2010. *Visual servoing via advanced numerical methods* (Vol. 401). Berlin: Springer.

Hassoun, M.H., 1995. *Fundamentals of artificial neural networks*. MIT press.

Haykin, S.S. and Haykin, S.S. eds., 2001. *Kalman filtering and neural networks* (pp. 221-269). New York: Wiley.

Henders, M.G. and Soudack, A.C., 1996. Dynamics and stability state-space of a controlled inverted pendulum. *International journal of non-linear mechanics*, *31*(2), pp.215-227.

Martinez, G. and Becerra, V.M., 2004, October. Preliminary results from a real time control system using optical information. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)* (Vol. 6, pp. 5923-5928). IEEE.

Hespanha, J.P., 2000. Single-camera visual servoing. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)* (Vol. 3, pp. 2533-2538). IEEE.

Hill, J., 1979. Real time control of a robot with a mobile camera. In *9th Int. Symp. on Industrial Robots, 1979* (pp. 233-246).

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.

Hung, C.C. and Fernandez, R.B., 1993, June. Comparative analysis of control design techniques for a cart-inverted-pendulum in real-time implementation. In *1993 American Control Conference* (pp. 1870-1874). IEEE.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, *82*(1), pp.35-45.

Kita, N., Davison, A.J., Kuniyoshi, Y., Hara, I., Matsui, T., Rougeaux, S., Hori, T. and Hirai, S., 1998. Mobile sensing robots for nuclear power plant inspection. *Advanced Robotics*, *13*(3), pp.355-356.

Johnny Lam, 2004. Control of an Inverted Pendulum. Proceeding of University of California, Santa Barbara, June 2004.

Lam, H.K., Leung, F.F. and Tam, P.K.S., 2002. Design and stability analysis of fuzzy model based nonlinear controller for nonlinear systems using genetic algorithm. In *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291)* (Vol. 1, pp. 232-237). IEEE.

Van der Linden, G.W. and Lambrechts, P.F., 1993. H/sub∞/control of an experimental inverted pendulum with dry friction. *IEEE Control Systems Magazine*, *13*(4), pp.44-50.

Lowe, D.G., 1987. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, *31*(3), pp.355-395.

Lozano, R., Fantoni, I. and Block, D.J., 2000. Stabilization of the inverted pendulum around its homoclinic orbit. *Systems & control letters*, *40*(3), pp.197-204.

Magana, M.E. and Holzapfel, F., 1998. Fuzzy-logic control of an inverted pendulum with vision feedback. *IEEE Transactions on Education*, *41*(2), pp.165-170.

Minorsky, N., 1922. Directional stability of automatically steered bodies. *Journal of the American Society for Naval Engineers*, *34*(2), pp.280-309.

Park, Y.M., Moon, U.C. and Lee, K.Y., 1995. A self-organizing fuzzy logic controller for dynamic systems using a fuzzy auto-regressive moving average (FARMA) model. *IEEE Transactions on Fuzzy systems*, *3*(1), pp.75-82.

Ponce, P., Molina, A. and Alvarez, E., 2014. A review of intelligent control systems applied to the inverted-pendulum problem. *American Journal of Engineering and Applied Sciences*, *7*(2), pp.194-240.

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), p.386.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1985. *Learning internal representations by error propagation* (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.

Sanderson, A.C., 1980. Image based visual servo control using relational graph error signal. In *Proc. Int. Conf. Cybernetics and Society, Cambridge, 1980*.

Schramm, F., Morel, G. and Lottin, A., 2004. Image Based Visual Servoing from Groups of 3D points.

Se, S., Lowe, D. and Little, J., 2001. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)* (Vol. 2, pp. 2051-2058). IEEE.

Shirai, Y. and Inoue, H., 1973. Guiding a robot by visual feedback in assembling tasks. *Pattern recognition*, *5*(2), pp.99-108.

Chen, C.S. and Chen, W.L., 1998. Robust adaptive sliding-mode control using fuzzy modeling for an inverted-pendulum system. *IEEE Transactions on Industrial Electronics*, *45*(2), pp.297-306.

S. Smith, 1997. Reviews of Optic Flow, Motion Segmentation, Edge Finding and Corner Finding. Tech. rep., Oxford Centre for Functional Magnetic Resonance Imaging of the Brain.

Stuflesser, M. and Brandner, M., 2008, May. Vision-based control of an inverted pendulum using cascaded particle filters. In *2008 IEEE Instrumentation and Measurement Technology Conference* (pp. 2097-2102). IEEE.

Srinivasan, B., Huguenin, P. and Bonvin, D., 2009. Global stabilization of an inverted pendulum–Control strategy and experimental verification. *Automatica*, *45*(1), pp.265-269.

Oh, S.K., Pedrycz, W., Rho, S.B. and Ahn, T.C., 2004. Parameter estimation of fuzzy controller and its application to inverted pendulum. *Engineering Applications of Artificial Intelligence*, *17*(1), pp.37-60.

Tahri, O. and Chaumette, F., 2004, April. Image moments: Generic descriptors for decoupled image-based visual servo. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004* (Vol. 2, pp. 1185-1190). IEEE.

Tahri, O. and Chaumette, F., 2005. Point-based and region-based image moments for visual servoing of planar objects. *IEEE Transactions on Robotics*, *21*(6), pp.1116-1127.

Tao, C.W., Taur, J.S., Wang, C.M. and Chen, U.S., 2008. Fuzzy hierarchical swing-up and sliding position controller for the inverted pendulum–cart system. *Fuzzy Sets and Systems*, *159*(20), pp.2763-2784.

Thuilot, B., Martinet, P., Cordesses, L. and Gallice, J., 2002. Position based visual servoing: keeping the object in the field of vision. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)* (Vol. 2, pp. 1624-1629). IEEE.

Toyama, K. and Blake, A., 2001. Probabilistic tracking in a metric space. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* (Vol. 2, pp. 50-57). IEEE.

Tu, Y.W. and Ho, M.T., 2010, September. Design and implementation of DSP and FPGA-based robust visual servoing control of an inverted pendulum. In *2010 IEEE International Conference on Control Applications* (pp. 83-88). IEEE.

Varsek, A., Urbancic, T. and Filipic, B., 1993. Genetic algorithms in controller design and tuning. *IEEE transactions on Systems, Man, and Cybernetics*, *23*(5), pp.1330-1339.

Kurdekar, V. and Borkar, S., 2013. Inverted pendulum control: A brief overview. *International Journal of Modern Engineering Research*, *3*(5), pp.2924-2927.

Wang, H., Chamroo, A., Vasseur, C. and Koncar, V., 2008, June. Hybrid control for vision based cart-inverted pendulum system. In *2008 American Control Conference* (pp. 3845-3850). IEEE.

Wang, H., Vasseur, C., Koncar, V., Chamroo, A. and Christov, N., 2010. Modeling and trajectory tracking control of a 2-DOF vision based inverted pendulum. *Journal of Control Engineering and Applied Informatics*, *12*(3), pp.59-66.

Weiss, L., Sanderson, A. & Neuman, C. 1987. Dynamic visual servo control of robots: An adaptive image-based approach, IEEE Journal on Robotics and Automation 3(5), 404–417. 1987.

Welsch G., Bishop G. 1995. An Introduction to the Kalman Filter, UNC-CH Computer Science Technical Report 95-041, 1995.

Wenzel, L., Vazquez, N., Nair, D. and Jamal, R., 2000. Computer vision based inverted pendulum. In *Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference [Cat. No. 00CH37066]* (Vol. 3, pp. 1319-1323). IEEE.

Wilson, W.J., Hulls, C.W. and Bell, G.S., 1996. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, *12*(5), pp.684-696.

Wong, C.C., Huang, B.C. and Chen, J.Y., 1998. Rule regulation of indirect adaptive fuzzy controller design. *IEE Proceedings-Control Theory and Applications*, *145*(6), pp.513-518.

Yilmaz, A., Javed, O. and Shah, M., 2006. Object tracking: A survey. *Acm computing surveys (CSUR)*, *38*(4), p.13.

C. Yong-Yan, L. Zongli. 2003. Robust stability analysis and fuzzy-scheduling control for nonlinear systems subject to actuator saturation, IEEE Transactions on Fuzzy Systems, vol.11, no.1, pp. 57- 67, Feb 2003.

J.G. Ziegler and N.B. Nichols. 1942. Optimum settings for automatic controllers, Trans. ASME, vol. 64, pp. 759-768, 1942. 1942.

Merakeb, A., Achemine, F. and Messine, F., 2013, June. Optimal time control to swing-up the inverted pendulum-cart in open-loop form. In *2013 IEEE 11th International Workshop of Electronics, Control, Measurement, Signals and their application to Mechatronics* (pp. 1-4). IEEE.

Durand, S., Castellanos, J.F.G., Marchand, N. and Sanchez, W.F.G., 2013. Event-based control of the inverted pendulum: Swing up and stabilization. *Journal of Control Engineering and Applied Informatics*, *15*(3), pp.96-104.

Uke, N. and Thool, R., 2013. Moving vehicle detection for measuring traffic count using opencv. *Journal of Automation and Control Engineering*, *1*(4)

Zhang, X., Fang, Y. and Sun, N., 2015. Visual servoing of mobile robots for posture stabilization: from theory to experiments. *International journal of robust and nonlinear control*, *25*(1), pp.1-15.

Azizian, M., Khoshnam, M., Najmaei, N. and Patel, R.V., 2014. Visual servoing in medical robotics: a survey. Part I: endoscopic and direct vision imaging–techniques and applications. *The international journal of medical robotics and computer assisted surgery*, *10*(3), pp.263-274.

Chui, C.K. and Chen, G., 2017. *Kalman filtering* (pp. 19-26). Springer International Publishing.

Thomas, J., Loianno, G., Sreenath, K. and Kumar, V., 2014. Toward image based visual servoing for aerial grasping and perching. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2113-2118). IEEE.

Xie, H., Lynch, A.F. and Jagersand, M., 2016. Dynamic IBVS of a rotary wing UAV using line features. *Robotica*, *34*(9), pp.2009-2026.

Giordani, S., Lujak, M. and Martinelli, F., 2013. A distributed multi-agent production planning and scheduling framework for mobile robots. *Computers & Industrial Engineering*, *64*(1), pp.19-30.

Gargade, A., Tambuskar, D. and Thokal, G., 2013. Modelling and analysis of pipe inspection robot. *International Journal of Emerging Technology and Advanced Engineering*, *3*(5), pp.120-126.

Wu, Y., Lim, J. and Yang, M.H., 2013. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2411-2418).

Buyssens, P., Daisy, M., Tschumperlé, D. and Lézoray, O., 2015. Exemplar-based inpainting: Technical review and new heuristics for better geometric reconstructions. *IEEE transactions on image processing*, *24*(6), pp.1809-1824.

Rong, W., Li, Z., Zhang, W. and Sun, L., 2014. An improved CANNY edge detection algorithm. In *2014 IEEE International Conference on Mechatronics and Automation* (pp. 577-582). IEEE.

Brahim, Z., Aziz, N.A., Aziz, N.A.A., Razali, S., Shapiai, M.I., Nawawi, S.W. and Mohamad, M.S., 2015. A Kalman filter approach for solving unimodal optimization problems. *ICIC Express Lett*, *9*(12), pp.3415-3422.

Assa, A. and Janabi-Sharifi, F., 2014. A robust vision-based sensor fusion approach for real-time pose estimation. *IEEE transactions on cybernetics*, *44*(2), pp.217-227.

György, K., Kelemen, A. and Dávid, L., 2014. Unscented Kalman filters and Particle Filter methods for nonlinear state estimation. *Procedia Technology*, *12*, pp.65-74.

Terra, M.H., Cerri, J.P. and Ishihara, J.Y., 2014. Optimal robust linear quadratic regulator for systems subject to uncertainties. *IEEE Transactions on Automatic Control*, *59*(9), pp.2586-2591.

Bettayeb, M., Boussalem, C., Mansouri, R. and Al-Saggaf, U.M., 2014. Stabilization of an inverted pendulum-cart system by fractional PI-state feedback. *ISA transactions*, *53*(2), pp.508-516.

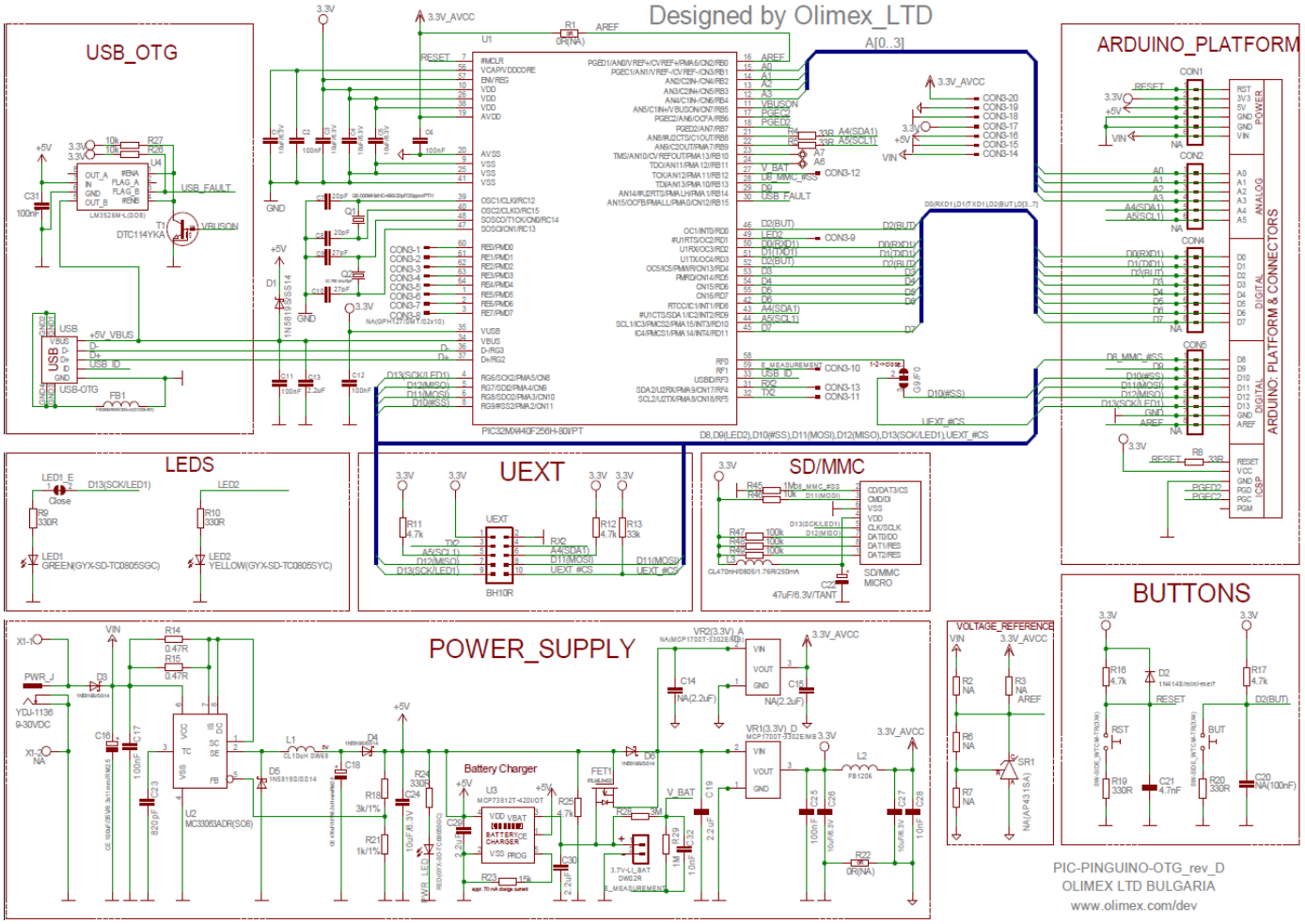
Chavez-Romero, R., Cardenas, A., Rendon-Mancha, J.M., Vernaza, K.M. and Piovesan, D., 2015. Inexpensive vision-based system for the direct measurement of ankle stiffness during quiet standing. *Journal of Medical Devices*, *9*(4), p.041011.

Chakraborty, K., Mukherjee, R.R. and Mukherjee, S., 2013. Tuning of PID controller of inverted pendulum using genetic algorithm. *Int. J. Soft Comput. Eng.*, *3*(1), pp.21 24.

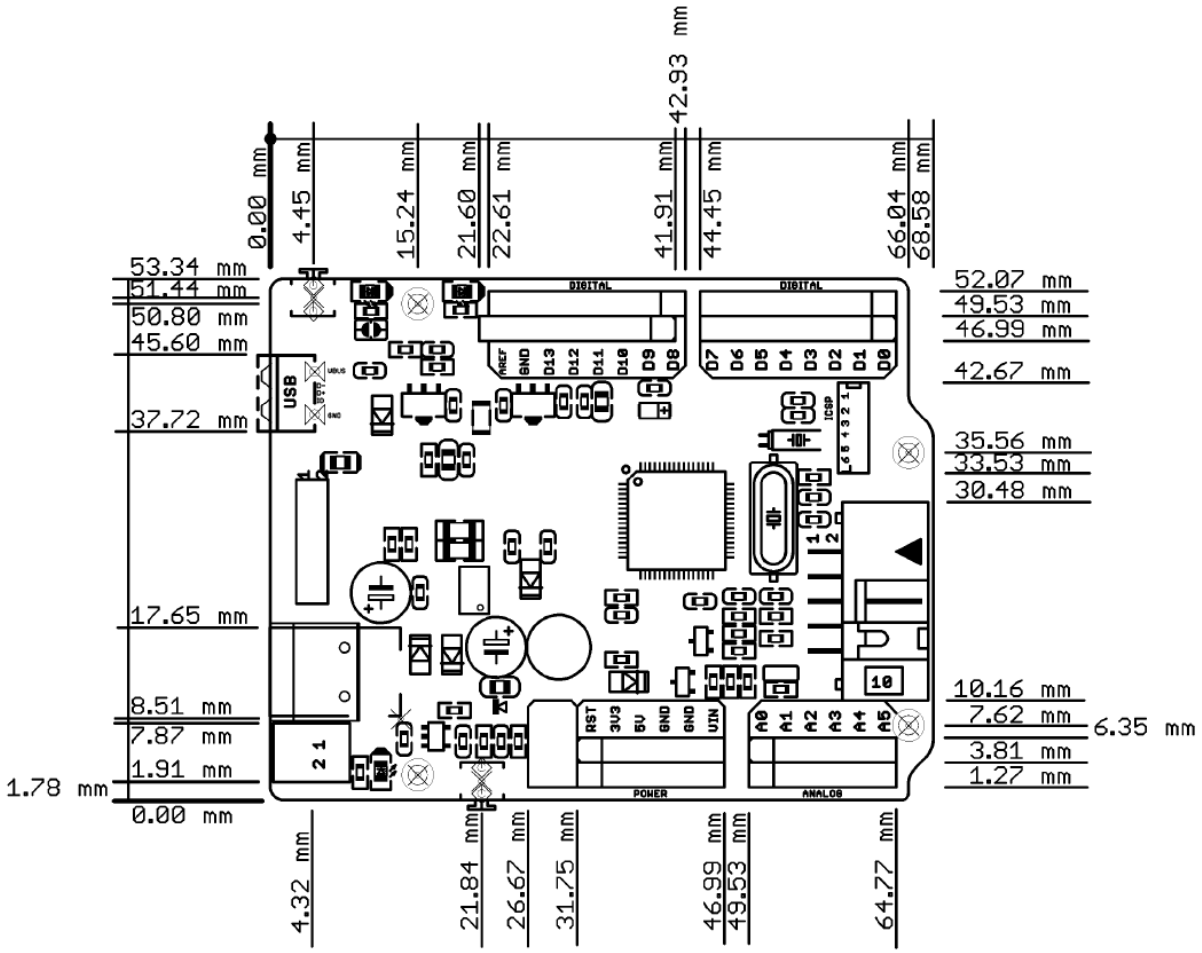
Appendix 1

PIC32-Pinguino Development Board

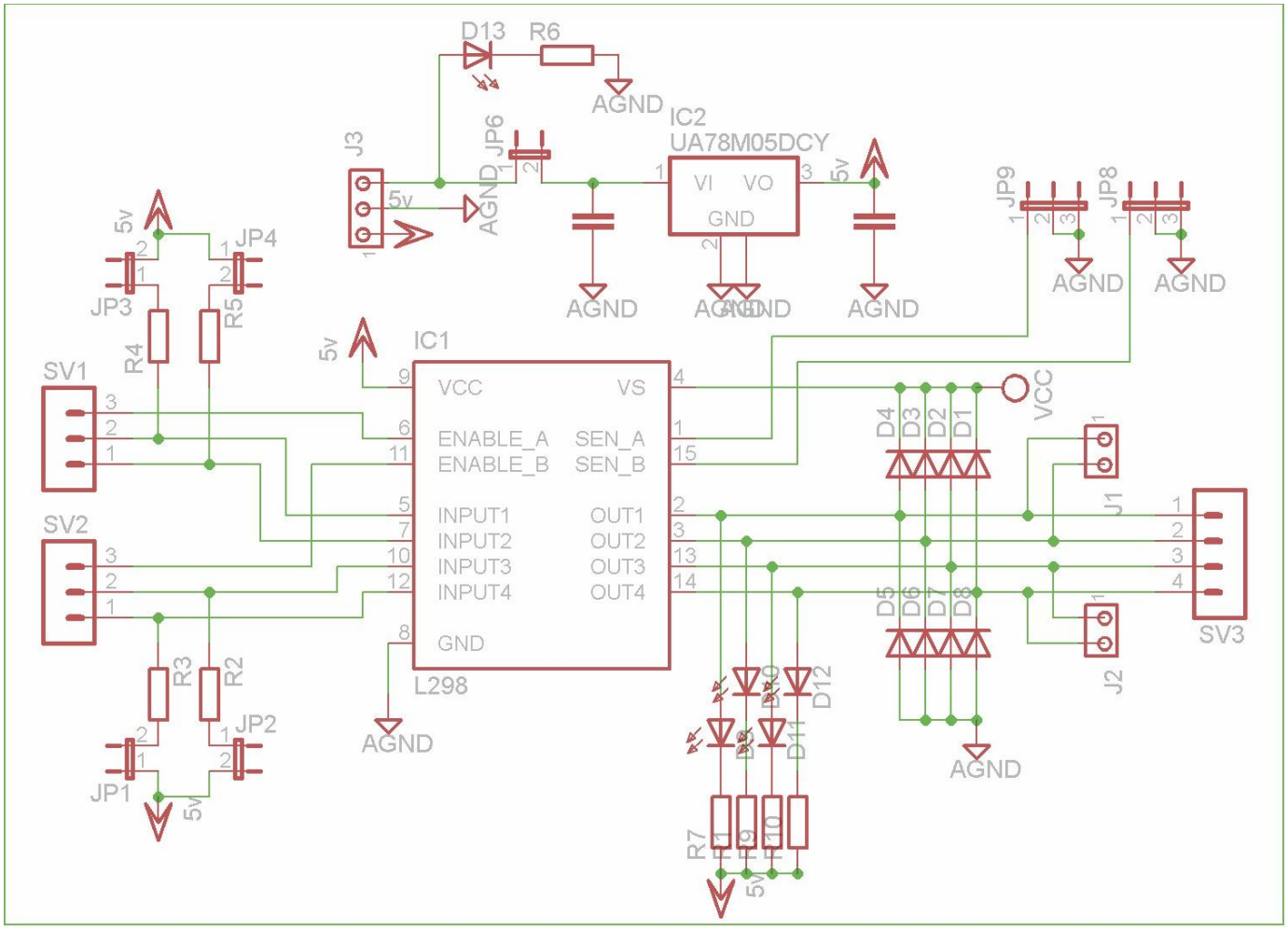
Schematics and Board Outline



Appendix.1.1 *– PIC32-Pinguino Schematic Diagram.*



Appendix.1.2 *– PIC32-Pinguino Schematic Diagram.*



Appendix.1.3 *– H-Bridge Driver Module Schematic Diagram.*

Appendix 2

Cart Inverted Pendulum

Controller Firmware

```
/*********************************************************************
 *
 *              Pendulum Firmware Version 1.0
 *
 *********************************************************************
 * FileName:        main.h
 * Dependencies: None
 * Processor:       PIC32MX
 * Compiler:        C32
 * Company:         Stephen Ingram - Ph.D Research
 *
 * Author           Date        Comment
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * Stephen Ingram                                   Created.
 ********************************************************************/
#define tris_self_power     TRISAbits.TRISA2
#define self_power          1

/** I N C L U D E S **************************************************/
#include <plib.h>
#include "./USB/usb.h"
#include "./USB/usb_function_cdc.h"

/** M A C R O S ******************************************************/
#define GetInstructionClock() 80000000

/** G E N E R A L  D E F I N I T I O N S ****************************/
#define TRUE                                    0x01
#define FALSE                                   0x00

#define IN                              TRUE
#define OUT                                     FALSE
#define PI                              3.14159265359
#define ANGLE_PER_BIT                   0.3515625
#define CRITICAL_ANGLE                  30.0
#define START_ANGLE                             5.0

#define VREF                                    3.3
#define RESOLUTION                              1024

#define DEFAULT_KP                              400//350//70
#define DEFAULT_KI                              25//25//7
#define DEFAULT_KD                              1//1//5//1

#define PID_LOOP_FREQ                   200.0 // Hz

#define MOTOR_MAX                               30.0  //175.0
#define MOTOR_MIN                               -30.0

#define DIR_OFF                         0
#define DIR_LEFT                        1
#define DIR_RIGHT                       2

#define INT_TIME_200US                  0x1F40
#define INT_TIME_500US                  0x4E20
#define INT_TIME_1MS                            0x9C40

/** I/O  D E F I N I T I O N S ***************************************/
#define LED1            LATFbits.LATF0
#define LED2            LATDbits.LATD1
#define DRV_EN          LATDbits.LATD2
#define DRV_IN1         LATDbits.LATD5
#define DRV_IN2         LATDbits.LATD6
#define BTN1            PORTDbits.RD0

/** P R O T O T Y P E S **********************************************/
double CosineInterpolate(double y1,double y2,double mu);
double LinearInterpolate(double y1,double y2, double mu);
unsigned int Read_ADC(unsigned int PIN);
```

```
void Set_Direction(unsigned char DIR);
void Set_Power(unsigned char PWR);
void Init_Controller(void);
void Set_Constants(void);
void Init_System(void);
void MSG_Handler(void);
void Send_Data(void);
void Init_ADC(void);
int main(void);
void PID(void);

/** G L O B A L  V A R I A B L E S************************************/
unsigned char USB_In_Buffer[64];
unsigned char USB_Out_Buffer[64];
unsigned char strData[64];
unsigned char strExt[512];

/** G L O B A L  V A R I A B L E S  P I D*****************************/
volatile signed int SumE_Min, SumE_Max, SumE, integral_term, derivative_term;
volatile signed short int en0, en1, en2, en3, off_set;
volatile unsigned char Feedback_Mode = FALSE;
volatile short int Camera_Angle;
unsigned short int ki, kd, kp;
volatile unsigned char do_PID;
volatile short int Motor_Temp;
volatile short int Set_Point;
volatile signed short int Cn;
volatile short int temp_int;
unsigned char Sample_Rate;
unsigned int Sample;
unsigned int Ts;


/********************************************************************
*                    Pendulum firmware V1.10
********************************************************************
* FileName:       Main.c
* Dependencies:   See INCLUDES section below
* Processor:      PIC32MX
* Compiler:       C32 1.12+
* Company:        Stephen Ingram - Ph.D Research
*
* Author          Date        Comment
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Stephen Ingram              Original.
********************************************************************/

/** C O N F I G U R A T I O N  B I T S ******************************/
#pragma config UPLLIDIV = DIV_2         // USB PLL Input Divider
#pragma config FPLLIDIV = DIV_2         // PLL Input Divider
#pragma config UPLLEN   = ON            // USB PLL Enabled
#pragma config FPLLODIV = DIV_1         // PLL Output Divider
#pragma config FPLLMUL  = MUL_20        // PLL Multiplier
#pragma config FWDTEN   = ON            // Watchdog Timer
#pragma config WDTPS = PS4096           // Watchdog timer postscaler
#pragma config POSCMOD  = HS            // Primary Oscillator
#pragma config FSOSCEN  = OFF           // Secondary Oscillator Enable
#pragma config FNOSC    = PRIPLL        // Oscillator Selection
#pragma config CP       = ON            // Code Protect - Read protect the code
#pragma config BWP      = OFF           // Boot Flash Write Protect - Disabled
#pragma config PWP      = OFF           // Program Flash WP Disabled
#pragma config ICESEL   = ICS_PGx2      // ICE/ICD Comm Channel Select
#pragma config FPBDIV   = DIV_8         // Peripheral Clock divisor

/** I N C L U D E S  ************************************************/
#include "Main.h"
#include <math.h>

/** G L O B A L  V A R I A B L E S***********************************/
volatile signed int last_real_angle = 0;
```

```
volatile unsigned char last_dir = 0;
volatile unsigned char current = 0;
volatile signed int last_angle = 0;
volatile signed int diff = 0;
volatile double step = 0.0;
unsigned char DTR_PRESENT;

/** I N T E R R U P T  S E R V I C E  R O U T I N E S **********/

/********************* TMR1 Interrupt ********************************/
void __attribute((interrupt(ipl1), vector(_TIMER_2_VECTOR), nomips16)) _T2Interrupt(void)
{
                        // Clear the flag
                        IFS0CLR = _IFS0_T2IF_MASK;

                        // Read motor temperature
                        Motor_Temp = 100 * (Read_ADC(2) * (VREF / RESOLUTION));

                        /* Select vision only feedback or potentiometer
                        if(Feedback_Mode == FALSE)
                        {
                                temp_int = (Read_ADC(1) - Set_Point);//(Read_ADC(1) - Set_Point);
                                Motor_Temp = (0.571231223 + (Cn * 0.94583474));
                        }
                        else
                        {
                                temp_int = (Camera_Angle-512);
                        }
                        */

                        /* Experiment with delays
                        if(Sample == 0)
                        {
                                // Toggle LED
                                LED2 ^= TRUE;
                                temp_int = (Read_ADC(1) - Set_Point);
                                step = (double)(temp_int - last_real_angle) / 10.0;
                                last_real_angle = temp_int;
                        }
                        else
                        {
                                // Interpolate the angle
                                temp_int = temp_int + (short int)(step + 0.5);
                        }

                        Sample++;
                        if(Sample == 10)
                        {
                                Sample = 0;
                        }
                        */

                        temp_int = (Read_ADC(1) - Set_Point);
                        en0 = temp_int + off_set;           // Store to error function asuming no over-flow

                        // Check pendulum within limits
                        if(flgRest == TRUE)
                        {
                                //if(temp_int == abs((short int)(START_ANGLE / ANGLE_PER_BIT))) do_PID =
                                TRUE; // Allowed to do PID function
                                if(temp_int > (short int)(START_ANGLE / ANGLE_PER_BIT)) //Check if error is
                                too large to start (positive)
                                {
                                        do_PID = TRUE; // Allowed to do PID function
                                }
                                if(temp_int < -(short int)(START_ANGLE / ANGLE_PER_BIT)) //Check if error
                                is too large to start (negative)
                                {
                                        do_PID = TRUE; // Allowed to do PID function
                                }
```

```
                        }

                        //do_PID = TRUE; // Allowed to do PID function

                        // Check for error in angle - pendulm limits
                        if(temp_int > (short int)(CRITICAL_ANGLE / ANGLE_PER_BIT))
                        {
                                Set_Power(0);                           // Stop PWM
                                Set_Direction(FALSE);

                                en0 = en1 = en2 = en3 = off_set = FALSE;        // Clear all PID constants
                                Cn = integral_term = derivative_term = SumE = FALSE;
                                do_PID = FALSE;                         // Stop doing PID
                                Ts = 0;
                        }
                        if(temp_int < -(short int)(CRITICAL_ANGLE / ANGLE_PER_BIT))             {


                                Set_Power(0);                           // Stop PWM
                                Set_Direction(FALSE);

                                en0 = en1 = en2 = en3 = off_set = FALSE;        // Clear all PID constants
                                Cn = integral_term = derivative_term = SumE = FALSE;
                                do_PID = FALSE;                         // Stop doing PID
                                Ts = 0;
                        }

                // Perform CDC service
        CDCTxService();
}

/******************** General exception ****************************/
void _general_exception_handler(unsigned cause, unsigned status)
{
                Nop();
                Nop();
}

/** I N I T I A L I S A T I O N  *********************************/
/*****************************************************************************
 * Function:        void Init_Controller(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Initialises the microcontroller IO pins.
 *
 * Note:            None
 *****************************************************************************/
void Init_Controller(void)
{
                // General TRIS
                TRISFbits.TRISF0 = OUT;         // LED1
                TRISDbits.TRISD1 = OUT;         // LED2
                TRISDbits.TRISD2 = OUT; // DRV EN
                TRISDbits.TRISD5 = OUT; // INP1
                TRISDbits.TRISD6 = OUT; // INP2
                TRISDbits.TRISD0 = IN;          // BTN1

                // All pins digital except AN1 - Potentiometer
                AD1PCFG = 0xFFFD;
}
```

```
/*********************************************************************
 * Function:        Init_ADC()
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Read ADC
 *
 * Note:            None
 ********************************************************************/
void Init_ADC(void)
{
   AD1CON1CLR = 0x8000;    // disable ADC before configuration

   AD1CON1 = 0x00E0;       // internal counter ends sampling and starts conversion (auto-convert), manual sample
   AD1CON2 = 0;            // AD1CON2<15:13> set voltage reference to pins AVSS/AVDD
   AD1CON3 = 0x0f01;       // TAD = 4*TPB, acquisition time = 15*TAD
            AD1CON1SET = 0x8000;           // turn on the ADC
}

/*********************************************************************
 * Function:        void Init_System(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Initialise the peripherals and system components
 *
 *
 * Note:            None
 ********************************************************************/
void Init_System(void)
{
            // Enable multi-vectored interrupts
            INTEnableSystemMultiVectoredInt();

            // Enable all interrupts
            INTEnableInterrupts();

            // Enable optimal performance
            SYSTEMConfigPerformance(80000000L);

            // Set PBUS clock divider (set to give 40MHz)
            mOSCSetPBDIV(OSC_PB_DIV_2);

            // Init OC3 module
            OpenOC3(OC_ON | OC_TIMER3_SRC | OC_PWM_FAULT_PIN_DISABLE, 0, 0);

            // Init Timer3 mode and period (PR3) (frequency of 1220Hz)
            OpenTimer3(T3_ON | T3_PS_1_1 | T3_SOURCE_INT, 0x10);//0xFFFF);

            // Setup ADC
            Init_ADC();

            // Disable JTAG port, but first wait 50ms so if we want to reprogram the part with
            // JTAG, we still have a tiny window before JTAG expires.
            //DelayMs(50);                                  // Provide a 50ms delay
            DDPCONbits.JTAGEN = FALSE;              // Disable JTAG port

            // Reset DTR present flag
```

```
	DTR_PRESENT = FALSE;

	USBDeviceInit();	//usb_device.c.  Initializes USB module SFRs and firmware

	// Enable the WDT
	WDTCONbits.ON = TRUE;

	// Init PID
	en0 = en1 = en2 = en3 = FALSE;
	ki = kd = FALSE;
	kp = off_set = FALSE;
	Set_Point = 511;
	Sample_Rate = 1;
	temp_int = integral_term = derivative_term = FALSE;
	SumE_Max = 30000;
	SumE_Min = 1 - SumE_Max;
	do_PID = TRUE;                                          // Allowed to do PID function
	Ts = 0;
	Sample = 0;

	// Disable postscaler
	T2CONbits.TCKPS = 7;

	// Disable timer
	TMR2 = FALSE;

	// Set default interrupt time - Equation is: Time_Required / (256 * 0.025 x 10-6)) because PS = 1:256.
	Was Time_Required / 0.025 x10-6 for PS = 1
	switch(Sample_Rate)
	{
		case 0:	PR2 = INT_TIME_1MS;
			break;
		case 1: PR2 = INT_TIME_500US;
			break;
		case 2: PR2 = INT_TIME_200US;
			break;
		default: PR2 = INT_TIME_1MS;
			break;
	}

	//PR2 = 0x30D;  // 200Hz
	//PR2 = 0xC80;  // 50Hz
	//PR2 = 0x3D09;	// 10 Hz
	PR2 = ((1.0 / PID_LOOP_FREQ) /  (256 * 0.000000025));

	// Clear interrupt flags and set priority
	IFS0bits.T2IF = FALSE;
	IEC0bits.T2IE = FALSE;
	IPC2bits.T2IP = 1;
	IPC2bits.T2IS = 1;

	// Enable timer
	T2CONbits.ON = TRUE;
	IEC0bits.T2IE = TRUE;
	LED2 = FALSE;
}
```

```
/** M A I N  F U N C T I O N ************************************/
/*********************************************************************
 * Function:        void main(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          int
 *
 * Side Effects:    None
 *
 * Overview:        Main program entry point.
 *
 * Note:            None
 ********************************************************************/
int main(void)
{
   // Initialise the microcontroller
            Init_Controller();

            // Initialise the overall system
   Init_System();

            //Get PID coefficients ki, kp and kd
            Set_Constants();

            // Main program loop
            while(TRUE)
   {
                        // Stop motor!!
                        if(!BTN1)
                        {
                                    Set_Power(0);
                                    Set_Direction(FALSE);
                                    do_PID = 0;
                        }

                  // Clear the WDT
                        ClrWdt();

                        // USB attach function
                        #if defined(USB_INTERRUPT)
                        if(USBGetDeviceState() == DETACHED_STATE)
                        {
                                    USBDeviceAttach();
                        }
                        #endif

                        // PID routine
                        if(do_PID)
                        {
                                    PID();
                                    Send_Data();
                        }

                        // Handle USB messages
                        MSG_Handler();
            }
}
```

```
/*********************************************************************
 * Function:        void Send_Data(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Sends inverted pendulum parameters to PC software
 *
 * Note:            None
 ********************************************************************/
void Send_Data(void)
{
        sprintf(strData,"%06d,%04d,%04d,%04d,%06d,%f,",Ts, temp_int, en0, Cn, Motor_Temp, step);
        putsUSBUSART(strData);
        Ts += 5;
}

/*********************************************************************
 * Function:        void PID(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Main PID function. The from of the PID is:
 *                                                  C(n) = K(E(n) + (Ts/Ti)SumE + (Td/Ts)[E(n) - E(n-1)])
 *
 * Note:            None
 ********************************************************************/
void PID(void)
{
        unsigned int tmp_sample_rate = FALSE;

        switch(Sample_Rate)
        {
                case 0:      tmp_sample_rate = 1000;
                             break;
                case 1: tmp_sample_rate = 2000;
                             break;
                case 2: tmp_sample_rate = 5000;
                             break;
                default: tmp_sample_rate = 1000;
                             break;
        }

        // Clear i and d terms
        integral_term = derivative_term = FALSE;

        // Calculate the integral term
        SumE = SumE + en0;                              // SumE is the summation of the error terms
        if(SumE > SumE_Max){                            // Test if the summation is too big
                SumE = SumE_Max;
        }
        if(SumE < SumE_Min){                            // Test if the summation is too small
                SumE = SumE_Min;
        }
        // Integral term is (Ts/Ti)*SumE where Ti is Kp/Ki

        // and Ts is the sampling period

        // Actual equation used to calculate the integral term is
```

```
		// Ki*SumE/(Kp*Fs*X) where X is an unknown scaling factor

		// and Fs is the sampling frequency
		integral_term = SumE / PID_LOOP_FREQ;		// Divide by the sampling frequency
		integral_term = integral_term * ki;		// Multiply Ki
		integral_term = integral_term / 16;		// combination of scaling factor and Kp

		// Calculate the derivative term
		derivative_term = en0 - en3;
		if(derivative_term > 120){						// Test if too large
			derivative_term = 120;
		}
		if(derivative_term < -120){						// test if too small
			derivative_term = -120;
		}
		// Calculate derivative term using (Td/Ts)[E(n) - E(n-1)]

		// Where Td is Kd/Kp

		// Actual equation used is Kd(en0-en3)/(Kp*X*3*Ts)
		derivative_term = derivative_term * kd;	// Where X is an unknown scaling factor
		derivative_term = derivative_term / 32;	// divide by 32 precalculated Kp*X*3*Ts

		if(derivative_term > 120){
			derivative_term = 120;
		}
		if(derivative_term < -120){
			derivative_term = -120;
		}

		// C(n) = K(E(n) + (Ts/Ti)SumE + (Td/Ts)[E(n) - E(n-1)])
		Cn = en0 + integral_term + derivative_term; // Sum the terms
		Cn = (Cn * kp) / 1024;					// multiply by Kp then scale

		if(Cn >= MOTOR_MAX)//1000
// Used to limit duty cycle not to have punch through
		{
			Cn = MOTOR_MAX; //1000
		}
  if(Cn <= MOTOR_MIN)//-1000
		{
			Cn = MOTOR_MIN;//-1000
		}
		if(Cn == 0)
		{						// Set the speed of the PWM
			Set_Power(0);				// Stop PWM
			Set_Direction(FALSE);
			off_set =0;
		}

		if(Cn > 0)
		{
			Set_Direction(DIR_RIGHT);	// Motor should go forward and set the duty cycle to Cn
			Set_Power(abs(Cn));		// Used to stop the pendulum from continually going
		}
		else
		{
			Set_Direction(DIR_LEFT);	// Motor should go backwards and set the duty cycle to Cn

			Set_Power(abs(Cn));	// Used to stop the pendulum from continually going around in a circle
		}

		en3 = en2;		// Shift error signals
		en2 = en1;
		en1 = en0;
		en0 = 0;
		do_PID = FALSE;	// Done
}
```

```
/*********************************************************************
 * Function:        Set_Constants(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Sets the PID constants
 *
 * Note:            None
 ********************************************************************/
void Set_Constants(void)
{
        kp = DEFAULT_KP; // Set Proportional Constant
        ki = DEFAULT_KI;   // Set Integral Constant
        kd = DEFAULT_KD; // Set Differential Constant
}

/*********************************************************************
 * Function:        Set_Direction(unsigned char DIR)
 *
 * PreCondition:    None
 *
 * Input:           unsigned char DIR: LEFT, RIGHT, OFF
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Sets the motor direction
 *
 * Note:            None
 ********************************************************************/
void Set_Direction(unsigned char DIR)
{
        // Set the motor direction
        if(DIR == DIR_LEFT)
        {
                // Set left direction
                DRV_IN1 = TRUE;
                DRV_IN2 = FALSE;
        }
        else if(DIR == DIR_RIGHT)
        {
                // Set righgt direction
                DRV_IN1 = FALSE;
                DRV_IN2 = TRUE;
        }
        else
        {
                // Set direction off
                DRV_IN1 = FALSE;
                DRV_IN2 = FALSE;
        }
}
```

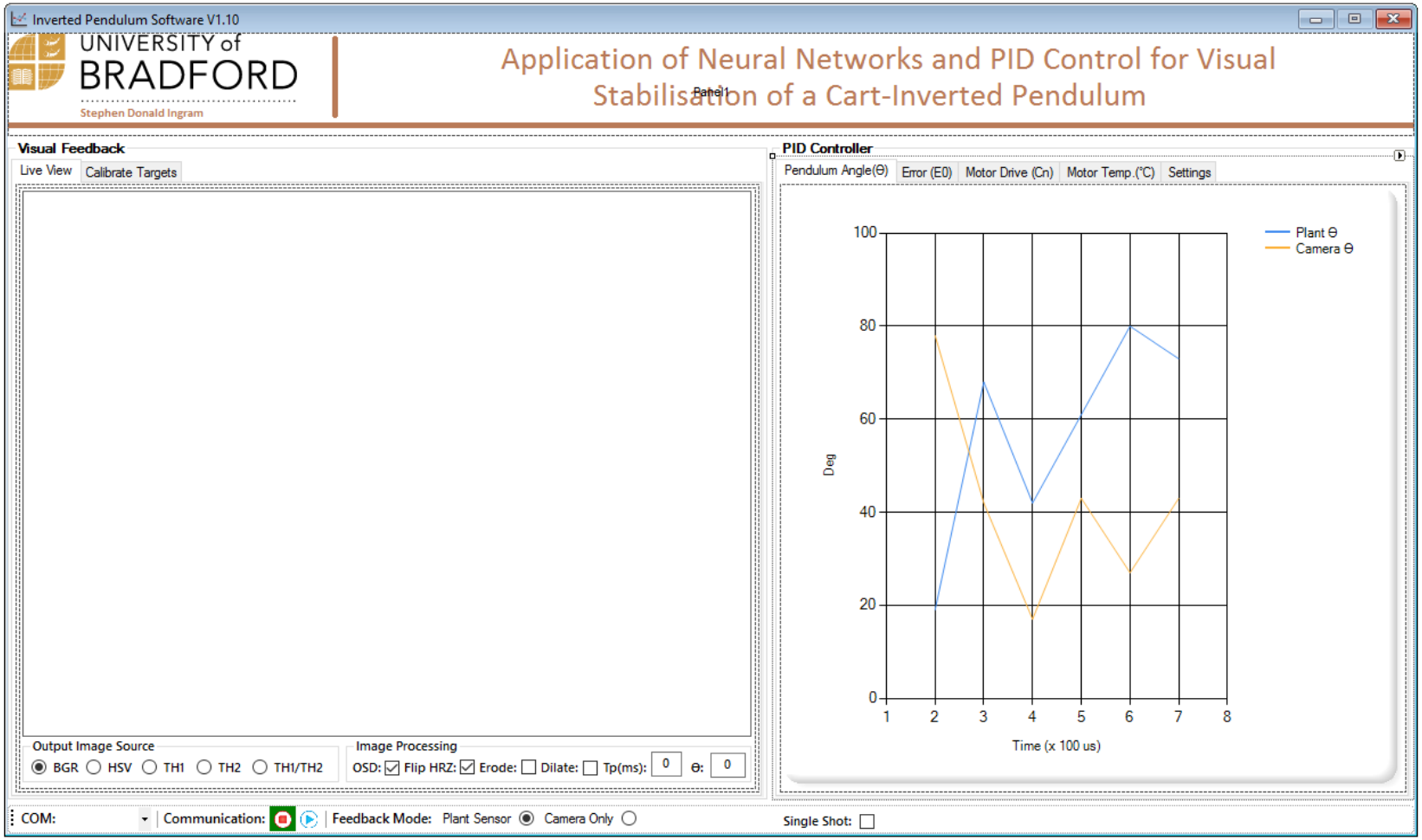
```
/*********************************************************************
 * Function:        Set_Power(unsigned char PWR)
 *
 * PreCondition:    None
 *
 * Input:           unsigned char PWR:  +/-127, 0 = off
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Sets the motor power
 *
 * Note:            None
 ********************************************************************/
void Set_Power(unsigned char PWR)
{
        unsigned int tmp;

        tmp = PR3;

        tmp = (unsigned int) (tmp * (PWR / MOTOR_MAX));
        // Set the motor power
        SetDCOC3PWM(tmp);
}

/*********************************************************************
 * Function:        unsigned int Read_ADC(unsigned int PIN);
 *
 * PreCondition:    adcConfigureMaual() has been called
 *
 * Input:           unsigned int PIN:
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Read ADC
 *
 * Note:            None
 ********************************************************************/
unsigned int Read_ADC(unsigned int PIN)
{
   AD1CHS = PIN << 16;                      // AD1CHS<16:19> controls which analog pin goes to the ADC

   AD1CON1bits.SAMP = 1;          // Begin sampling
   while( AD1CON1bits.SAMP );     // wait until acquisition is done
   while( ! AD1CON1bits.DONE );   // wait until conversion done

   return ADC1BUF0;               // result stored in ADC1BUF0
}
```

```
/********************************************************************
 * Function:        void MSG_Handler(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        Handles USB messages - for the console interface
 *
 * Note:            None
 *******************************************************************/
void MSG_Handler(void)
{
        static unsigned char  flgWelcomeMsg = TRUE;
        unsigned char x = 0, BytesRead = 0;
        short temp_setting = 0;

    // Exit if USB not configured or suspended
    if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl == TRUE)) return;

    // If we are ready to send data
    if(USBUSARTIsTxTrfReady())
    {
            // If port is open for the first time
            if((flgWelcomeMsg == TRUE) && (DTR_PRESENT == TRUE))
            {
                    // Send welcome message
                            putrsUSBUSART("\r\n-------------------------------------\r\nStephen Ingram -
Pendulum Controller \r\n\nConsole Interface\r\n\nFirmware Ver 1.10\r\n-------------------------------------\r\n");

                            // Clear flag so we only send welcome message once
                            flgWelcomeMsg = FALSE;

                            // Initialise counters
                            x = FALSE;
                    }
                    else if(DTR_PRESENT == FALSE)
                    {
                    flgWelcomeMsg = TRUE; // Set flag so we send start-up message when COM port opens
                    }
            }

    // Get number of bytes read (if any)
    BytesRead = getsUSBUSART(USB_Out_Buffer,32);

    // If one or more bytes have been received
    if(BytesRead != FALSE)
    {
            for(x = 0; ((x < BytesRead) && (x < 32)); x++)
            {
                    // If carriage return received
                    if(USB_Out_Buffer[x] == 0x0D)
                    {
                                    // Get test value and use Motor_Temp variable for debugging purposes
                                    //Motor_Temp = (((USB_Out_Buffer[0] - '0') * 1000) +
((USB_Out_Buffer[1] - '0') * 100) + ((USB_Out_Buffer[2] - '0') * 10) + (USB_Out_Buffer[3] - '0'));
                            }
                    }
            }

            // Perform CDC service
    CDCTxService();
}
```

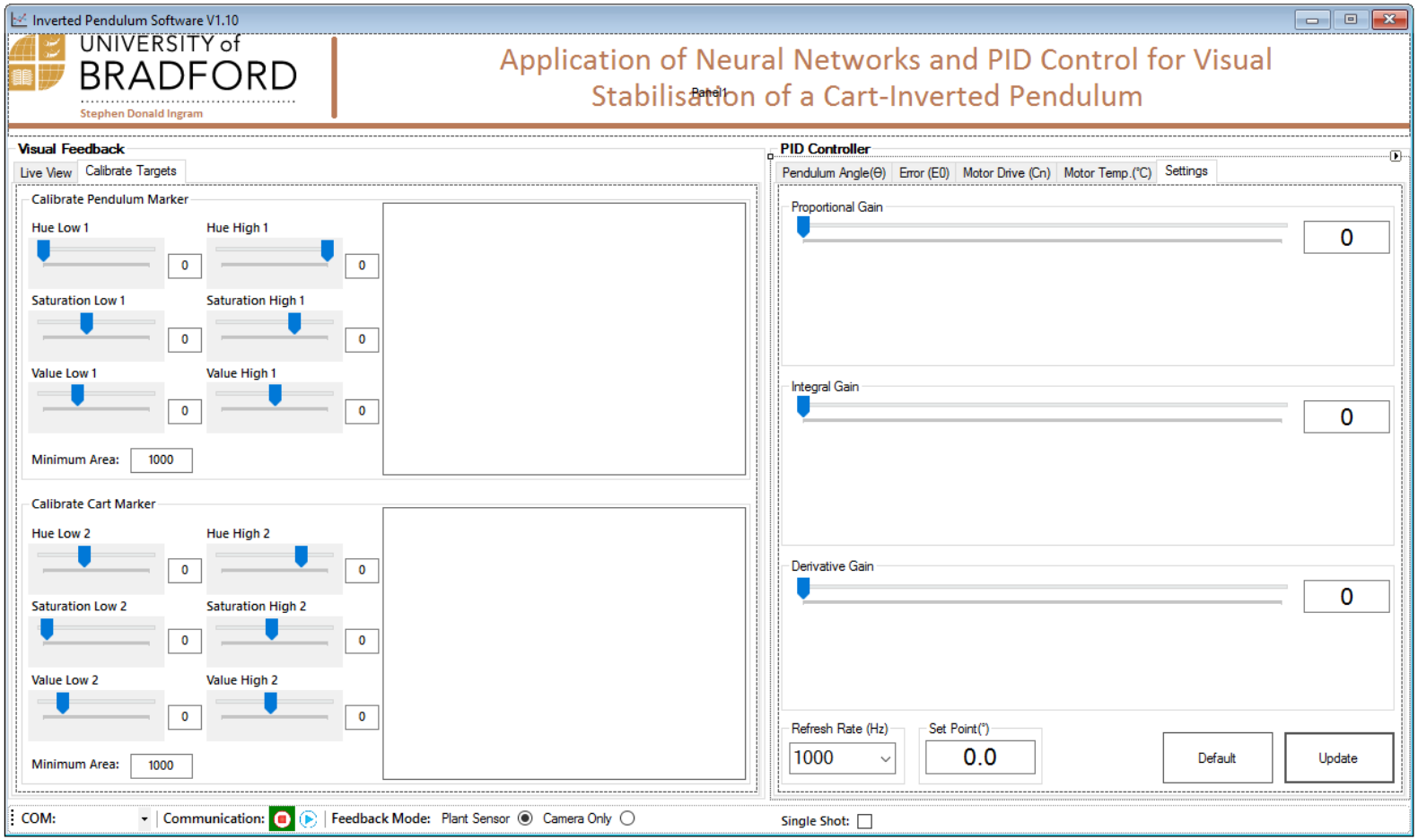
Appendix 3

Cart Inverted Pendulum

Test Application



Appendix.3.1 *- Main GUI Form.*



Appendix.3.2 *- Image Filter View.*

```
' Main image processing thread
Sub ProcessFrameAndUpdateGUI(ByVal sender As Object, ByVal arg As EventArgs)

    ' Debug!
    Dim sw As New Stopwatch

    Try
        ' Start stopwatch
        sw.Start()

        ' Capture the image - Takes a lot of time!!!
        imgOriginal = capWebcam.QueryFrame

        ' Flip image about the horizontal axis if option selected
        If CheckBoxFHRZ.Checked = True Then CvInvoke.Flip(imgOriginal, imgOriginal, FlipType.Horizontal)

        ' Create images
        Dim hsv_img As New Mat(imgOriginal.Size, DepthType.Cv8U, 3)
        Dim threshold_img1 As New Mat(imgOriginal.Size, DepthType.Cv8U, 1)
        Dim threshold_img1a As New Mat(imgOriginal.Size, DepthType.Cv8U, 1)
        Dim threshold_img2 As New Mat(imgOriginal.Size, DepthType.Cv8U, 1)

        ' Convert the image to HSV
        CvInvoke.CvtColor(imgOriginal, hsv_img, CvEnum.ColorConversion.Bgr2Hsv)

        ' Threshold the image to isolate the two marker colors
        CvInvoke.InRange(hsv_img, New ScalarArray(New MCvScalar(TrackLowH1.Value, TrackLowS1.Value,
        TrackLowV1.Value)), New ScalarArray(New MCvScalar(TrackHighH1.Value, TrackHighS1.Value,
        TrackHighV1.Value)), threshold_img1)

        CvInvoke.InRange(hsv_img, New ScalarArray(New MCvScalar(TrackLowH2.Value, TrackLowS2.Value,
        TrackLowV2.Value)), New ScalarArray(New MCvScalar(TrackHighH2.Value, TrackHighS2.Value,
        TrackHighV2.Value)), threshold_img2)

        ' Filter the threshold images if option selected
        If CheckFilterErode.Checked = True Then CvInvoke.Erode(threshold_img1, threshold_img1, structuringElement, New
        Point(-1, -1), 1, BorderType.Default, New MCvScalar(0, 0, 0))

        If CheckFilterErode.Checked = True Then CvInvoke.Erode(threshold_img2, threshold_img2, structuringElement, New
        Point(-1, -1), 1, BorderType.Default, New MCvScalar(0, 0, 0))

        If CheckFilterDilate.Checked = True Then CvInvoke.Dilate(threshold_img1, threshold_img1, structuringElement, New
        Point(-1, -1), 1, BorderType.Default, New MCvScalar(0, 0, 0))

        If CheckFilterDilate.Checked = True Then CvInvoke.Dilate(threshold_img2, threshold_img2, structuringElement, New
        Point(-1, -1), 1, BorderType.Default, New MCvScalar(0, 0, 0))

        ' Determine the moments of the two marker objects
        moments1 = CvInvoke.Moments(threshold_img1, 0)
        moments2 = CvInvoke.Moments(threshold_img2, 0)

        ' Calculate the area of the two marker objects
        area1 = CvInvoke.cvGetCentralMoment(moments1, 0, 0)
        area2 = CvInvoke.cvGetCentralMoment(moments2, 0, 0)

        ' Check that area1 is above the minimum threshold
        If (area1 > Convert.ToInt32(TextMinArea1.Text)) Then
            'x and y coordinates of the center of the object is found by dividing the 1,0 and 0,1 moments by the area
            x1 = Int(CvInvoke.cvGetSpatialMoment(moments1, 1, 0) / area1)
            y1 = Int(CvInvoke.cvGetSpatialMoment(moments1, 0, 1) / area1)

            'draw circle
            If CheckBoxOSD.Checked = True Then CvInvoke.Circle(imgOriginal, New Point(x1, y1), CInt(1), New
            MCvScalar(0, 0, 255), 10)

            'write x and y position
            If CheckBoxOSD.Checked = True Then CvInvoke.PutText(imgOriginal, x1.ToString + "," + y1.ToString, New
            System.Drawing.Point(x1, (y1 + 20)), FontFace.HersheyComplex, 1, New MCvScalar(255, 255, 255))
        End If
```

```
        ' Check that area2 is above the minimum threshold
        If (area2 > Convert.ToInt32(TextMinArea2.Text)) Then

       'x and y coordinates of the center of the object is found by dividing the 1,0 and 0,1 moments by the area
       x2 = Int(CvInvoke.cvGetSpatialMoment(moments2, 1, 0) / area2)
       y2 = Int(CvInvoke.cvGetSpatialMoment(moments2, 0, 1) / area2)

       'draw circle
        If CheckBoxOSD.Checked = True Then CvInvoke.Circle(imgOriginal, New Point(CInt(x2), CInt(y2)), CInt(1), New
        MCvScalar(255, 0, 0), 10)

       'write x and y position
         If CheckBoxOSD.Checked = True Then CvInvoke.PutText(imgOriginal, x2.ToString + "," + y2.ToString, New
        System.Drawing.Point(x2, (y2 + 20)), FontFace.HersheyComplex, 1, New MCvScalar(255, 255, 255))

        If CheckBoxOSD.Checked = True Then CvInvoke.Line(imgOriginal, New System.Drawing.Point(x1, y1), New
        System.Drawing.Point(x2, y2), New MCvScalar(0, 255, 0), 4, LineType.AntiAlias, 0)

       'draw reference line
        If CheckBoxOSD.Checked = True Then CvInvoke.Line(imgOriginal, New System.Drawing.Point(x1, y1), New
        System.Drawing.Point(640, y1), New MCvScalar(100, 100, 1000, 100), 4, LineType.AntiAlias, 0)

       ' Convert to angle and write to image
       angle = Math.Atan((y1 - y2) / (x2 - x1)) * (180 / Math.PI)

       ' Translate angle to be same as rig
       If angle > 0 Then
          angle = (90 - angle)
       Else
          angle = (-90) - angle
       End If

      ' Invert displayed angle
      angle = -1 * angle

      ' Format the dispalyed angle
      TextAngle.Text = angle.ToString("###.0").PadLeft(3)

      ' Plot chart - Camera
      'chtAngle.Series(1).Points.AddXY(varTime, Convert.ToDouble(TextAngle.Text))

      ' Plot other data
      chtAngle.Series(0).Points.AddXY(varTime, Convert.ToDouble(ang))

      'chtError.Series(0).Points.AddXY(varTime, (Convert.ToInt16(strData(1)) * 0.352))
      'chtDrive.Series(0).Points.AddXY(varTime, ((Convert.ToInt16(strData(2)) / 175.0) * 100))
      'chtTemp.Series(0).Points.AddXY(varTime, Convert.ToInt16(strData(3)))

        If CheckBoxOSD.Checked = True Then CvInvoke.PutText(imgOriginal, angle.ToString("###.0").PadLeft(3), New
        System.Drawing.Point((x1 + 50), ((y2 + y1) / 2)), FontFace.HersheyComplex, 1, New MCvScalar(255, 255, 255))

      ' Determine feedback sensor
      If RadioButtonCameraOnly.Checked Then
         FeedBackSensor = True
         Digital_Angle = Math.Round(((Convert.ToDouble(angle) / 0.3515625) + 512)).ToString.PadLeft(4, "0"c).ToString

         ' Update the pendulum angle
         sendSerialCommandFast("set_angle " & FeedBackSensor.ToString & Digital_Angle)
      Else
         FeedBackSensor = 0
      End If
   End If

   ' Stop stopwatch
   sw.Stop()

   " Display the processing time
   TextProcessingTime.Text = sw.ElapsedMilliseconds.ToString
```

```
        ' Display and format appropriate image
        If TabVision.SelectedTab.Name = "TabPageCalibrateTargets" Then
            ' Show threshold images as we are on calibrate tab
            ImageBoxTh1.Image = threshold_img1
            ImageBoxTh2.Image = threshold_img2
        Else
            ' Show other images as we are on the live view tab
            If RadioButtonBGR.Checked = True Then ibOriginal.Image = imgOriginal ' BGR format
            If RadioButtonHSV.Checked = True Then ibOriginal.Image = hsv_img ' HSV format
            If RadioButtonTH1.Checked = True Then ibOriginal.Image = threshold_img1 ' TH1 format
            If RadioButtonTH2.Checked = True Then ibOriginal.Image = threshold_img2 ' TH2 format
            If RadioButtonTH1TH2.Checked = True Then
                CvInvoke.Add(threshold_img1, threshold_img2, imgOriginal)  ' Combine two threshold images
                ibOriginal.Image = imgOriginal ' Display combined format
            End If
        End If
    Catch ex As Exception
        ' An exception occurred
    End Try
End Sub
```