Application of Machine Vision in UAVs for Autonomous Target Tracking

Joshua Patrick Effland
West Virginia University

Application of Machine Vision in UAVs for Autonomous Target Tracking

Joshua Patrick Effland

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University in
partial fulfillment of the requirements
for the degree of

Master of Science
in
Aerospace Engineering

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2008

Keywords: Machine Vision, Target Detection, Target Tracking, Circle Detection

Abstract

Application of Machine Vision in UAVs for Autonomous Target Tracking

Joshua Patrick Effland

This research presents experimental results for the application of Machine Vision (MV) techniques to address the problem of target detection and tracking. The main objective is the design of a prototype "UAV" surveillance environment to emulate real-life conditions. The model environment for this experiment consists of a target simulated by a small electric train system, located at "ground" level, and a MV camera mounted on a motion-based apparatus located directly above the model setup. This system is meant to be a non-flying mockup of an aerial robot retrofitted with a MV sensor. Therefore, the final design is a two degree-of-freedom gantry simulating aircraft motions above the "ground" level at a constant altitude. On the "ground" level, the design of the landscape is an attempt to achieve a realistic natural landscape within a laboratory setting. Therefore, the scenery consists of small scale trees, bushes, a mountain, and a tunnel system within a 914 mm by 1066 mm boundary. To detect and track the moving train, MV algorithms are implemented in a Matlab/Simulink® based simulation environment. Specifically, image pre-processing techniques and circle detection algorithms are implemented to detect and identify the chimney stack on the train engine. The circle detection algorithms analyzed in this research effort consists of a least squares based method and the Hough transform (HT) method for circle detection. The experimental results will show that the solution to the target detection problem could produce a positive detection rate of 90% during each simulation while utilizing only 56% of the input image. Tracking and timing data also shows that the least squares based target detection method performs substantially better then the HT method. This is evident from the result of using a 1-2 Hz frequency update rate for the Simulink® scheme which is acceptable, in some cases, for use in navigation for a UAV performing scouting and reconnaissance missions. The development of vision-based control strategies, similar to the approach presented in this research, allows UAVs to participate in complex missions involving autonomous target tracking.

Acknowledgements

Many people have helped me reach this milestone in my career. To begin, I would like to acknowledge and give thanks to my research advisor, Dr. Marcello Napolitano, who provided me with this project and available funding which allowed this thesis to be completed. I would like to say thanks to him for having confidence in me and giving me this great opportunity to work on this project. He has been a great teacher throughout my graduate and undergraduate studies and will never be forgotten. Also, I would like to thank my committee for their help and time, which included Dr. Brad Seanor, Dr. Yu Gu, and Dr. Xin Li.

I also want to give special thanks to my colleagues Dr. Brad Seanor, Dr. Yu Gu, Dr. Srikanth Gururajan, and Dr. Giampiero Campa for providing me with support and guidance throughout my graduate studies. I would also like to give recognition to many other graduate students who were always willing to provide me with help and support when I needed it. This includes Sergio Tamayo, Marco Mammarella, and Jason Jarrell. Through this process I have made some great friends that will never be forgotten. The memories through the good and bad times have all been worth while thanks to all of you.

Most importantly, I would like to give thanks to my family who has made this opportunity possible. My mother and grandmother have given me endless amounts of support which have helped make every goal of mine obtainable. None of this would have been possible without their help.

Table of Contents

List of Tables

List of Figures

Nomenclature

Symbol	Description
English	
(a,b)	Center coordinate for circle
(r,s)	Intensity transformation function input point
(x,y)	Pixel edge point locations
A	A matrix of the FFF algorithm
A(i,,j)	Accumulator matrix
c	Vector of coefficients
D	D vector of the FFF algorithm
E	Slope of contrast-stretching function
E	Error between edge point and center location of circle
F	Gradient magnitude matrix
f	Binary image resulting from threshold function
f	Relative aperture
f(x,y)	Intensity value of pixel in image
G	Intensity gradient of image
g	Arbitrary function representation
g(x,y)	Output pixel value of intensity transformation method
I	Gray level intensity value
i	X-component of accumulator matrix
j	Y-component of accumulator matrix
L	Number of input pixels
L	Luminance
M	Number of rows in image
m	Threshold pixel intensity for contrast-stretching function
N	Number of columns in image
n	Number of edge points
n	Total number of edge points
P(x,y)	Edge pixel locations on circle
$P_o(x_o,y_o)$	Center edge pixel locations on circle
r	Pixel intensities of input image
r	Circle radius
s	Pixel intensities of output image
T	Threshold intensity value
T	Intensity transformation method
t	Shutter time
t	Gray level intensity value within the upper and lower limits
v	Vector of coordinates

x	x-component of pixel location
X	Resulting vector of the FFF algorithm
y	y-component of pixel location
z	Difference between edge point distance and circle radius

<u>Subscript</u>

f	Upper bound of gray level range
i	Edge pixel reference number in x-direction
j	Edge pixel reference number in y-direction
l	Lower bound of gray level range
o	center location of circle
t	Gray level intensity value
x	Intensity gradient in x-direction
y	Intensity gradient in y-direction

<u>Acronym</u>

A	Amps
ATA	Advanced Technology Attachment
CCD	Charge Coupled Discharge
CPU	Central Processing Unit
CT	Computed Tomography
D	Diagonal
DC	Direct Current
DDR	Double Data Rate
DDR2	Double Data Rate 2
DG	Differential Gradient
DIO	Digital input/output
DOD	Department of Defense
DoF	Degree of Freedom
DSC	Digital Still Camera
EKF	Extended Kalman Filter
FAA	Federal Aviation Administration
FFF	Fast Finding and Fitting
FLIR	Forward-Looking Infrared
FOV	Field of View
fps	Frames per second
FSB	Front side bus
GB	Gigabyte
GHT	Generalized Hough Transform
GHz	Gigahertz
GPS	Global Positioning System

H	Height
HT	Hough transform
ID	Identification
IR	Infrared
mA	Milliamps
MB	Megabytes
MHz	Megahertz
mm	Millimeters
MRI	Magnetic Resonance Imaging
ms	Milliseconds
MSE	Mean square error
MSI	Micro-Star International
MV	Machine Vision
NASA	The National Aeronautics and Space Administration
oz	Ounces
PC	Personal Computer
PID	Proportional, Integral, and Derivative
R&D	Research and Development
RAM	Random access memory
RGB	Red, Green, Blue
SATA	Serial advanced technology attachment
SRRAM	Synchronous Dynamic Random Access Memory
TM	Template Matching
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
USF	University of South Florida
V	Volts
VDC	Volts Direct Current
W	Width
WVU	West Virginia University

Chapter 1. Introduction

1.1. Problem Definition

Unmanned Aerial Vehicles (UAVs) can generally be described as remotely piloted or autonomously piloted aircraft with sizes ranging from the small 5-pound SkySeer to the 2,250-pound Predator B. UAVs can carry different types of sensors, payloads, cameras, and communication equipment for an ever increasing range of applications. A large majority of the UAVs in use within the United States are military crafts[1]. For the military realm, they have played a large role in reconnaissance and intelligence gathering missions since the 1950s and more combat related mission are envisioned in the future[2]. The progression of UAVs is leading to the development of a more lethal platform capable of performing precisions air and ground strikes. This is evident from Figure 1.1 where the evolution of UAVs is displayed along with a useful reference to the Department of Defense's (DOD) major UAV programs.

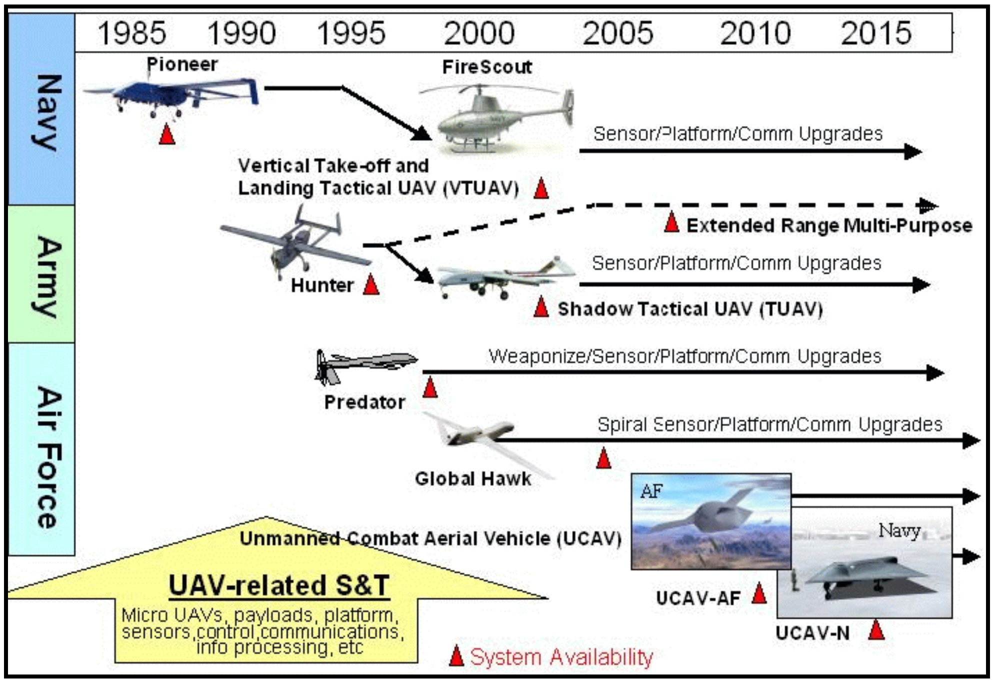

1985
1990
1995
2000
2005
2010
2015
Navy
Army
Air Force
Pioneer
FireScout
Sensor/Platform/Comm Upgrades
Vertical Take-off and Landing Tactical UAV (VTUAV)
Extended Range Multi-Purpose
Hunter
Sensor/Platform/Comm Upgrades
Shadow Tactical UAV (TUAV)
Weaponize/Sensor/Platform/Comm Upgrades
Predator
Spiral Sensor/Platform/Comm Upgrades
Global Hawk
AF
Navy
Unmanned Combat Aerial Vehicle (UCAV)
UAV-related S&T
Micro UAVs, payloads, platform, sensors, control, communications, info processing, etc
UCAV-AF
UCAV-N
System Availability

Figure 1.1: UAV Programs, 1985-2015[3]

It is obvious from this diagram that the military and government sponsored companies have provided a great deal of support for UAV Research and Development (R&D) which can be accredited to their flexibility and moderately low-cost[4]. Currently, Army UAV systems support land warfare operations for different mission structures. Typical missions include:

- intelligence;
- surveillance and reconnaissance;
- battle damage assessment;
- targeting;
- persistent stare for continued operations, and convoy protection[5].

For example, Northrop Grumman has developed a UAV with missile-firing capabilities where this system is used for the U.S. Army's extended-range multi-purpose UAV program and is currently used in Iraq for surveillance missions[6]. UAVs are also very critical for U.S. Air Force missions where the RQ-1 Predator is used for surveillance and reconnaissance operations. The mission structure for this UAV involves acquiring surveillance imagery obtained from video cameras and an infrared system which is distributed in real-time to the front-line soldier and operational commander. The Predator can then play a multi-role for armed reconnaissance and solder protection. Predator UAVs have been operation in Bosnia since 1995 as part of the Operation Enduring Freedom in Afghanistan and Operation Iraqi Freedom[7].

The operation of UAVs is tightly controlled in the U.S. where they can only be flown in restricted airspace cut off from manned aircraft. Before UAVs can be issued for widespread use, outside of the military realm, the Federal Aviation Administration (FAA) must be convinced that an unmanned aircraft is capable of the same tasks a manned aircraft is capable. For example, in a manned aircraft, human pilots are responsible for looking out the cockpit window to avoid head on collisions. Addressing this issue in UAVs, Northrop Grumman is currently conducting research that use video cameras and sophisticated image processing software to hunt for incoming aircraft[8]. In situations where another aircraft is detected, the UAV would then carry out evasive maneuvers. The increasing popularity of UAVs is a direct consequence of their ability to extend tasks normally performed by human beings, who are classified to have a high degree of failure resulting from exhaustion, distractions, and illness. This can be directly related to the extensive use of UAVs in Iraq where they account for 80 percent of all current flights and use digital cameras and sophisticated image processing software to detect incoming targets[9]. This capability is one example of the importance of UAVs in a military application, by eliminating the high risk posed to human pilots.

In recent years, the evolution of microelectronics has shown direct results associated with improved sensors, larger memory, and faster microcomputers, which has significantly improved the level of UAV technology. Utilizing new technology with low-cost platforms, engineers are able to design highly sophisticated UAVs that can be employed within a large mission structure. Furthermore, advances in digital camera technology have lead to the creation of powerful vision based sensors that are light weight and easily integrated with UAVs. The overlying task at hand then becomes the integration of the vision based sensors with sophisticated image processing software. This is necessary to provide UAVs with vital target information enabling them to perform specific missions such as reconnaissance and surveillance.

When considering the five senses (sight, hearing, smell, taste, and touch) many consider sight to be the primary source of data for humans. For this reason biological vision is studied and modeled in terms of psychological processes implemented by the human brain. Computer vision can be described as a compliment to biological vision through interdisciplinary exchange. This exchange has helped lay the path for vision to play a large role in the expansion of UAV capabilities. Manipulating computer vision into UAVs is defined as a form of Machine Vision (MV)[10]. Within the basic innate action of sight or vision by the human eye and extremely efficient information processing through the human brain, the process of scene analysis occurs very quickly and easily. The applications noted can be very difficult to implement/reproduce with the same outcome through MV when large amounts of data have to be interpreted. For this reason, many of the current MV algorithms are not applicable for real-time application. This introduces challenges for designing and implementing an on-board system for a UAV platform. However, advances in microelectronics and MV algorithms are leading to the capabilities of real-time video processing MV tasks on-board small UAVs.

Furthermore, the application of using MV based measurements for target detection and tracking is the current task at hand. This research effort is geared towards the development of a MV system which would utilize an on–board camera for interpreting an image to extract, identify, and track an object of interest via an aircraft platform. The purpose is to address the target detection and tracking problem and evaluate and implement proper hardware and software approaches necessary to complete this task. This includes the evaluation and acquisition of Commercial Off The Shelf (COTS) hardware that can be utilized for this application. The software portion of this work involves the evaluation and implementation of image processing software necessary to detect a specific feature of an object using captured video frames. Once the target is detected, it must then be tracked as it moves along its path, which will be made possible through the implementation of a control algorithm.

1.2. Target Detection and Tracking

Research efforts in the area of MV initially became a topic of interest at West Virginia University (WVU) based on previous work in the topics of autonomous aerial refueling, runway detection, and aerial surveillance by unmanned systems. Through these applications of MV, extensive algorithms should reliably detect an image feature, leading to the control of an aircraft platform based on a robust control system design. The correlation of this concept is thoroughly applied within this research effort by utilizing MV based techniques for measurement estimations of position and velocity of a moving target. The measurements will be an input into a position control system, which will generate a feasible trajectory for the “UAV,” so that the target will stay in the camera’s field of view (FOV).

To perform experimental testing of this research strategy, a mock environment shown in Figure 1.2 is setup to resemble a scaled model environment that simulates a UAV flying over a mountainous terrain.

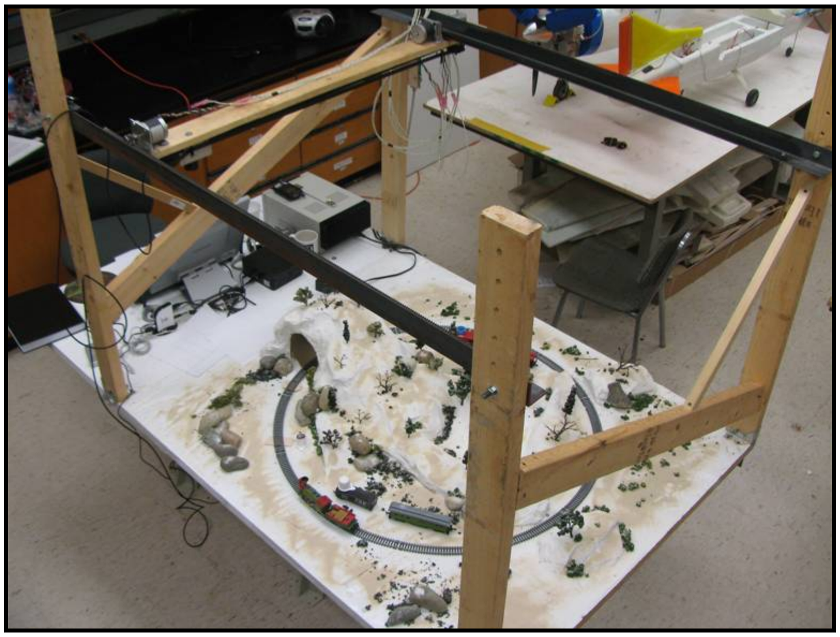


Figure 1.2: Model Environment Setup

This environment incorporates a digital camera for capturing images and continuously detects and tracks a single "ground" target. To accomplish this task, a mechanical positioning system is constructed to represent the UAV position. This system consists of stepper motors driving the camera on a rail type system, which allows movement within a two-dimensional grid at a predetermined height. At "ground" level, shown in Figure 1.3, there will be a target incorporated within the scaled model environment, which will be continuously tracked by the MV system once detected. A small electric train (model railroad setup) is utilized as the target vehicle for this application.

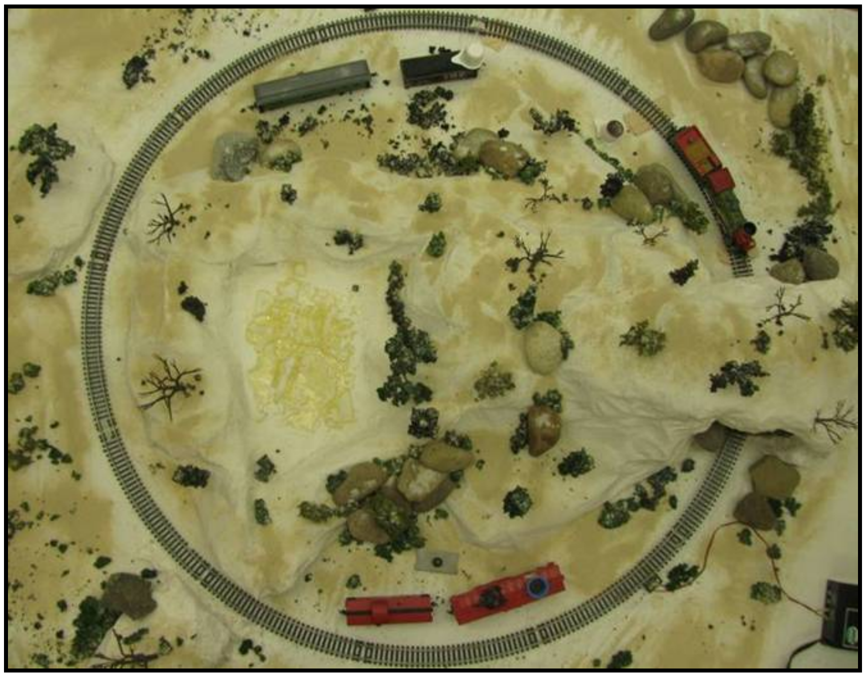


Figure 1.3: "Ground" Level of Model Environment

When selecting the correct techniques for addressing the target detection problem the first issue that must be addressed is the processing time for the algorithms. This is critical since the video stream that would be acquired on-board a UAV, in a real life scenario, must be processed in a timely fashion in order to perform tasks such as target detection while providing vital inputs to an on-board navigation system for tracking purposes and avoiding possible collisions with other obstacles. Although, for the purpose of this effort, the MV software will not be fully capable of real-time processing, the issue of computation time is still of concern. The MV algorithms must be computationally efficient to process each extracted video frame and allow input data to be placed into a position control system. These tasks must be performed in a timely manner in order to keep the target in the FOV. With this in mind, a concerning issue exists with a majority of the existing MV algorithms. In some cases these algorithms are not for use with mobile platforms, so the software environment and algorithms used must be chosen that produces the best results for this effort.

A common problem within MV, image processing, and computer vision is the task of determining if an image contains a specific object, target, or feature. Many existing algorithms used for this task are at best able to detect specifically defined features and shapes[11]. The general case of object detection for random shapes in random situations is far from satisfactory. There are a variety of image segmentation problems, which fall under the categories of object recognition, identification, and detection. In principle, the problem addressed in this document is a classic example of an image segmentation problem. Based on the defining features of the target, the image processing algorithms are used to clearly distinguish the target from the background. Supporting this concept, the pre-processing methods become a critical aspect in the design of the MV software. To further support this claim, it is a common trend that pre-processing algorithms usually require more computational time then the main image processing algorithm, therefore; it is important to make them as efficient as possible. However, it can also be debated that it is equally important to design a robust and efficient feature detection algorithm for the main-processing stage. This is important to differentiate a target from other features, which could produce false detections. This document will show that equal emphasis is taken for the design of the main algorithm and the selection of the pre-processing methods.

1.3. Research Objectives

This research is used to demonstrate, in principle, the design, implementation, and experimental validation of a prototype UAV surveillance system using MV for the purpose of target detection and tracking.

The envisioned final scenario features a MV camera system which will be used to perform surveillance missions within a defined area. Specific MV algorithms will be used to extract information from captured frames of a live video camera stream. With these capabilities, the MV algorithms will be used to detect moving targets and estimate a target's position and velocity. Once the target's information is extracted, the position control system will generate a feasible trajectory to keep the target within the camera's FOV. The tasks outlined to complete this effort are described below:

- Select and integrate MV hardware that can be used to test MV algorithms in a laboratory environment;
- Design and construction of a scaled model environment for testing various MV algorithms;

- Develop filtering methods using Matlab® and Simulink® programming tools for <u>diminishing image noise in digital images</u> captured from the MV camera. Experiments will be completed to compare various filtering techniques based on their timing profiles and statistical analysis. The filtering techniques that best satisfies the performance requirements for computation time and noise suppression will be utilized;
- Develop and test MV software using Matlab® and Simulink® programming tools that is <u>capable of detecting circular patterns</u>. This software is used to extract the target from the digital image based on its defining circular features. Experiments will be performed to asses the performance of the software when multiple objects are introduced in the background. These experiments are used to obtain the repeatability performance, timing profiles, and statistical analysis of the methods used to complete this task;
- Develop and integrate an active control scheme using Simulink® programming tools. This enables the camera system to continuously track the target along its path. Experiments will be performed to assess the performance of the control scheme.

1.4. Overview of Thesis

This thesis will be organized in the following manner:

- Chapter 2 will discuss the results of prior research on image segmentation, pattern recognition and related image processing techniques. The focus will be placed on applications and hardware setups utilized in the aerospace industry;
- Chapter 3 will discuss the theory behind the image processing algorithms used and the proposed solution to each problem. Specifically it will cover the methods used for circle detection and tracking targets;
- Chapter 4 will explain the setup of the experimental environment. This includes detailed descriptions of how each pre-processing method is utilized and the implementation of the main processing algorithms. The hardware and laboratory configurations will be explained in detail;

- Chapter 5 will show the simulation results obtained for the target recognition and tracking algorithms;
- Chapter 6 will contain the conclusive documentation of the results that are obtained from this effort and recommendations for continuing efforts leading to actual vehicle implementation.

Chapter 2. Literary Review

2.1. General Description

MV has been extensively used within ground based industrial and manufacturing processes. The absence of size and weight restrictions supports the wide spread use of MV in these processes. However, this is not the case for UAV-based MV applications, which require smaller and light weight products while preserving accuracy and reliability. As technology improves over time, MV techniques are being used for a broader range of applications requiring more highly versatile MV systems. Some examples of these more sophisticated MV applications are surveillance, automated inspection, vehicle guidance, traffic monitoring and control, and biometric measurements. As technology will continue to evolve, MV systems are capable of higher computational loads and are much more efficient.

The MV industry can be characterized as a continuously evolving industry where price and performance ratios constantly improve. An increase in processing speeds for vision-processing hardware allows for an increase in processing power reserve. This reserve can be used for more intelligent vision tools in order to simplify a system's programming[12]. These advances in semiconductor technology have also enabled vision processing systems to decrease in size. This substantial reduction in size will most likely result in an embedded hardware system that will encompass all of the capabilities of current MV systems. This special purpose system would then be designed to perform one or a few specific tasks, sometimes with real-time computing constraints. In contrast, the general PC is used to complete many different tasks. The small embedded system is more specifically task driven; allowing engineers to optimize the device to make it smaller and less expensive. This sensor is envisioned to be the adoption of network distributed computing techniques. In the future, any network capable device will be used to configure and monitor MV applications[13]. The vision sensor will continue to become smaller, faster, and easier to integrate.

The next section of this document, Section 2.2, will outline the commonly used methods of image segmentation and their applications within many different fields. A brief understanding of each method is provided prior to the literary review.

2.2. Image Segmentation Methods

Image segmentation is a processing method that has been used for a variety of applications. These applications apply to many different fields including:

- medical;
- aerospace;
- manufacturing
- agriculture.

The medical field accounts for a vast majority of these applications where image segmentation is used for:

- the detection of tumors and other pathologies;
- the measurement of tissue volumes;
- computer-guided surgery;
- diagnosis processes.

Image segmentation techniques are also becoming very popular within the aerospace field. Examples include aerial surveillance, runway detection, and guided landings for spacecraft. This is a direct result of the advances in MV technology which have sparked the development of improved image segmentation algorithms and allowed MV systems to become much lighter and smaller along with an increase in computational power and speed. These advanced systems are being more commonly used in aircraft and spacecraft; specifically for UAVs. Enhanced technologies have triggered the development of superior techniques for image segmentation applications which may not have been possible with "old" technology and image segmentation methods.

The task of image segmentation can be described as the partitioning of a digital image into multiple regions or sets of pixels[14]. This process is performed to transform an image into multiple sections that are easier to analyze and interpret. Segmentation algorithms are generally based on two basic properties of pixel intensity values: discontinuity and similarity. The first category is based on segmenting an image based on abrupt changes in pixel intensity, such as edges. Edge-based segmentation methods fall in this category which will be discussed in Section 2.2.1. The second category, similarity is based on segmenting an image into sections that are similar to specific criteria. This category includes pixel-based methods, region-based methods, and connectivity-preserving relaxation-based methods, which will be discussed in Sections 2.2.2, 2.2.3, and 2.2.4, respectively.

2.2.1. Edge-based Segmentation Method

Edge-based methods are an alternative path for image segmentation which allows features of an image to be extracted based upon the edges in the image. Two methods are utilized to detect the edges of images, template matching (TM) and the differential gradient (DG) approach. The basic concept shared by these two methods is determining the intensity gradient magnitude, otherwise known as the change of pixel intensity from one location to the next. An area of an image where the gradient magnitude is noticeably large is considered as an indicator for an edge. Edge-based methods usually work well when they are used in conjunction with other superior segmentations methods. The edge-based methods rely on edge detection operators, which are used to mark the points in a digital image where the luminous intensity changes abruptly[15]. These operators are defined as the Canny, Roberts, Sobel, and Prewitt operators. Edge detection operators combined with other higher order segmentation methods and image processing techniques are used to determine the presence of an edge in a digital image.

However, using edge detection as an image segmentation approach can lead to some inadequate results. These types of results are exhibited in images when detected edges have gaps in their boundary descriptions. This is obtained when transitions between the image regions may not be abrupt enough or poor lighting conditions exist. Also, in some cases, edges may be detected in an image that aren't apart of the region boundary, which is a result from the given image being noisy or poorly illuminated[16]. A common solution to this problem is the use of a threshold technique combined with an edge detection method to help eliminate noisy features in an image. For example, a pixel would be defined in an image as being an edge pixel if its intensity gradient magnitude is greater then a specific threshold value. If the boundary region is clearly represented in an image then this method of filtering will be sufficient, however the image contrast may vary widely. In this case there may be very little gained from using the thresholding technique.

For simple cases of horizontal and vertical edge detection, a graphical approach can be utilized to apply image segmentation. A method described by Kumar, Hart, and Ahuja performs edge based segmentation where the sum of pixel intensities along each directional peak is taken to be the boundary edge[17]. This algorithm is sufficient when the image is free of severely curved edges. Another method for edge detection is known as the Marr-Hildreth algorithm which is used to find the Laplacian and Gaussian-smoothed image and obtains a result as a set of zero-crossing lines. This specific method is attractive for the fact that it does not use any form of threshold in the detection process. However, this algorithm has the same limitations as the previously mentioned edge detection operators. These are other examples where complex images that contain large amounts of noise and poor illumination may require the implementation of additional algorithms to enhance the edge detection process.

The design of edge detection operators has increased to a much more advanced stage, such that edges are detected with high pixel accuracy oriented to fractions of a degree. Unfortunately some of these methods are very complex and result in very large computational loads. When the image is free of noise or is considered to be a “clean” image, edge based segmentation techniques become very efficient. Then, once noise is added into an image, difficulties arise when trying to detect true edges; some edges are detected where there is no true border and some edges are not detected where a border exists. Furthermore, low level edge segmentation algorithms might not be capable of extracting edges that have overlapping boundaries. There have been many attempts to resolve these issues and to attempt to eliminate noise interference. One solution is the implementation of higher level edge segmentation methods, such as the Hough transform (HT).

The HT is classified as a higher level edge segmentation method that provides a highly robust method for locating specific features in images. Historically, the HT has been the predominant means of detecting straight edges. This method was introduced by Paul Hough in 1962. It was developed and refined for the particular purpose of detecting straight edges. It was originally used for detection of lines, but has been modified for detection of arbitrary shapes such as ellipses, corners, and circles. The modified methodology of HT utilized today was invented by Richard Duda and Peter Hart in 1972 who called it the "generalized Hough transform" (GHT)[18]. The implementation of this method requires an analytical representation of the arbitrary shape or boundary, which is the only information required. This characteristic makes the HT particularly useful for computing a global description of a feature, in which the number of solution classes need not be known a prior. The motivating idea behind the development of the edge-based technique is to coordinate point or pixel locations indicating their contributions to a globally consistent solution. The main advantage of the HT process is that it is fairly tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise[19].

2.2.2. Pixel-based Segmentation Methods

A simple approximation to image segmentation can be achieved by an image processing technique previously mentioned as thresholding. This is classified as a low level pixel-based segmentation method that proves to be adequate in cases where the boundaries are highly distinguishable. This procedure involves separating the light and dark pixels of an image by scanning an entire image, pixel by pixel, and then classifying each pixel as foreground or background. Although this technique is very popular, it has a limited range of effectiveness. One problem that may occur in thresholding is determining an optimum threshold value. This is difficult when illumination is not uniform throughout the image. So, inadequate lighting is a problematic issue when using this technique. This can cause the boundary of an object in the foreground to become blurry, making it difficult to determine a threshold value to segment the foreground from the background. To solve this problem methods known as adaptive and local thresholding have been proposed as solutions[20]. In most cases, other techniques of image segmentation are performed to establish cases in which intensity thresholding can be applied. Since this method is a relatively simple pixel-based segmentation method it is commonly used in correlation with more advanced segmentation methods.

A more advanced pixel-based method of imaging segmentation is known as clustering which is performed in the work completed at Coventry University[21]. In this research clustering is used to search for distinct groups of pixels in the image space. These groups are expected to have distinct features and can be clearly differentiated.

Clustering methods are used to combine an observed image into groups of data that satisfy two main criteria: each group or cluster is homogeneous and each group or cluster should be different from other clusters[22]. These techniques have been applied to a wide range of research problems. Overall, the cluster analysis is a great utility that can be applied to any problem requiring the classification of a large image into manageable groups of data.

2.2.3. Region-based Segmentation Method

Unlike pixel-based, region-based methods are considered to be a more advanced image segmentation method. The objective of this method is to directly partition an image into regions rather then using the distribution of pixel properties. Regions are formed by grouping pixels or smaller regions based on certain criteria such as color, texture, or boundaries. The selection of the similarity criteria for each region is based on the available image data and the problem at hand; image descriptors are sometimes used for this classification. A disadvantage of this method is that the descriptors can lead to false results if connectivity or adjacent pixel information is not available for the region growing process. A control issue identified with this method is an inadequate stopping rule that determines when the region has reached its maximum number of pixels in the growth process[23].

An easy approach to region-based segmentation is to use a procedure that grows regions from a set of seed points. Regions will then grow based on the chosen criteria and the location of the seed points. This method requires strict criteria for the seed point selection which can result in large computation time when comparing the surrounding pixels to the seed points.

An alternative approach is used in the work completed at the University of Putra Malaysia[24] which subdivides an image initially into arbitrary regions then merges or splits the regions based on the chosen criteria. For this technique, two main algorithms are designed; the merging and the splitting algorithm. These work iteratively to satisfy the conditions. The merging algorithm compares different regions which are then merged together if they represent the same properties. The splitting algorithm is used to divide large regions, which may not be completely uniform, into smaller regions that share similar properties. This algorithm begins by representing the entire image as a region. It then uses the approach of subdividing the image into successively smaller and smaller regions so that, for any region, the chosen criteria is true. If the sub region does not satisfy the given criteria the region will continue to be divided until the criteria is satisfied. If only the splitting algorithm is used then the end result could contain adjacent regions with similar properties. This problem can be solved using the merging algorithm in conjunction with splitting algorithm. The adjacent regions in the portioned image that share similar properties can then be merged together.

After reviewing these methods it is obvious that they have positive aspects as well as drawbacks for image segmentation applications. One obvious drawback is the computation time. This can drastically increase based on the complexity of the image and necessity for strict criteria to perform the merging and splitting techniques.

2.2.4. Connectivity Preserving Relaxation Segmentation Method

A newer approach towards image segmentation is the connectivity preserving relaxation method or otherwise known as the active contour model. This method is utilized in research regarding active contours for detection and tracking of moving targets[25]. In this research an active contour, or 'snake', starts near an initial contour, or spline, then dynamically evolves. The spline is attracted towards edges of a boundary with an effort to minimize an energy cost function. The contour is initially set somewhere near the edge of interest. It then dynamically evolves and is drawn towards the edge of the object. Therefore, an active contour model is a segmentation method that doesn't depend on a specific shape which makes it applicable to a number of different segmentation problems. Spline curves, however, can represent very complex curves, which could add complexity to the shrink and expansion operations. Other problems that snakes present are their inability to move towards objects that are far away from the initial spline location. It is also difficult for them to move into boundary indentations. Due to the computational cost and disadvantages that can result from this method, it is probably not the best solution for a near real time application. In other applications where computational cost is not an issue, this new segmentation method has drastically increased in popularity.

2.3. Image Segmentation Applications

Many different applications of image segmentation have been investigated. The four methods previously described are the main focus of this research. All of the information that was found during this research stage is explained and documented in Sections 2.3.1 through 2.3.5. Each section is segmented into their respective research areas, which include a detailed description of the images segmentation methods used.

2.3.1. Aerospace Related Image Segmentation Applications

As mentioned earlier, MV has shown to be a key topic of research for applications within the aerospace industry. For these applications, MV is typically used to eliminate or assist the presence of a human operator. Currently, many different agencies and universities are conducting research involving MV applications. UAVs, such as the USAF Predator, use infrared and visible sensors to help detect camouflaged targets, while companies like Harsh Environment Applied Technologies Inc. utilize various IR sensors to create better night vision systems for soldiers, motorized vehicles, and ships. The principle at hand is to eliminate the necessity of human participation which will reduce the need to place a person or crew under harmful or extreme conditions. This is a key reason why research within the aerospace industry has consistently involved UAVs. MV applications will be used to further expand UAV capabilities and better approximate a manned aircraft as they continue to be an essential research topic within the aerospace industry.

Traditionally, UAVs are used for missions in military settings that pose high human risk and significance. The trend of reducing human involvement has now become appealing to civil and commercial users as well. For example, in 2005, researchers at National Oceanographic & Atmospheric Administration flew a UAV into the tropical storm Ophelia. Since the use of UAVs is virtually risk-free, it can be flown in very dangerous and extreme conditions where a manned aircraft could not perform as efficiently. The U.S. Forest Service is exploring UAV technology as well. In October of 2007, in conjunction with NASA, they flew a UAV over a 40,200 acre fire near Palm Springs, California. For 16 hours the unmanned vehicle circled the area while beaming down images that are used to determine the perimeter of the fire. Manned flights are much less efficient for the fact that they have to terminate flights when the pilot is fatigued, hungry, or for other reasons. With the UAV applications previously described, as well as with many other applications, it is essential that they have the ability to detect a specific target or object in the air or on the ground. This ability can be accomplished with MV, which will drastically increase the longevity of a UAV and its capabilities. After conducting an extensive literary review, many engineering applications related to this research topic are investigated. Each of the applications are explained in this section and are separated into their respective research areas.

Airborne surveillance has proven to be very important to a wide range of occupations, such as boarder patrol, crop control, wildfire detection, and resource investigation. A topic under investigation at the University of South Florida (USF) involves aerial surveillance, which represents a MV application that will interpret data acquired by a UAV onboard camera[26]. The objective of this research was to design a MV system that would be able to interpret data detected by a MV sensor. Once the data is interpreted, this processes a system decision to send an alarm while circling the target location. This research involves components of image processing such as noise reduction and feature extraction. Images acquired by the MV sensor are then subjected to gauss filtering for noise reduction purposes. The selection of the noise filter is based on its ability to remove noise without significant information loss. When using a Gaussian filter, the size of the filter is an important factor considering performance requirements. A larger filter (i.e.: 7x7) would offer a more accurate representation of the Gaussian function, but would include additional computational cost and, therefore, more processing time. The use of a smaller filter would reduce the number of computations, but would sacrifice accuracy of the Gaussian bell. Some other filters commonly used in image processing include mode and median filters.

The next stage of research at USF is a feature extraction process, which is involved in the surveillance process. For this process a region-based technique is used, which partitions the image into various regions. The size and mean intensity for each of these regions are selected as features. The features are then extracted from the image by using the pixel-based method known as thresholding. The process of thresholding converts the initial gray-scale image into a binary image in which objects appear as black figures on a white background or vise versa, according to the threshold value. The areas with a larger grayscale intensity value are then extracted for image processing.

Other research related to reconnaissance, surveillance, and target acquisition using small UAVs integrated with MV has also been completed at Drexel[27]. The concept in this research emphasizes on the ability to fly autonomously in a near-Earth environment in scenarios where GPS data and communications are not available. This requires sensors that will allow the vehicle to compensate for the loss of signal. The researchers at Drexel address this issue by designing a test rig that can capture the performance characteristics of a non-flying mockup of a UAV shown in

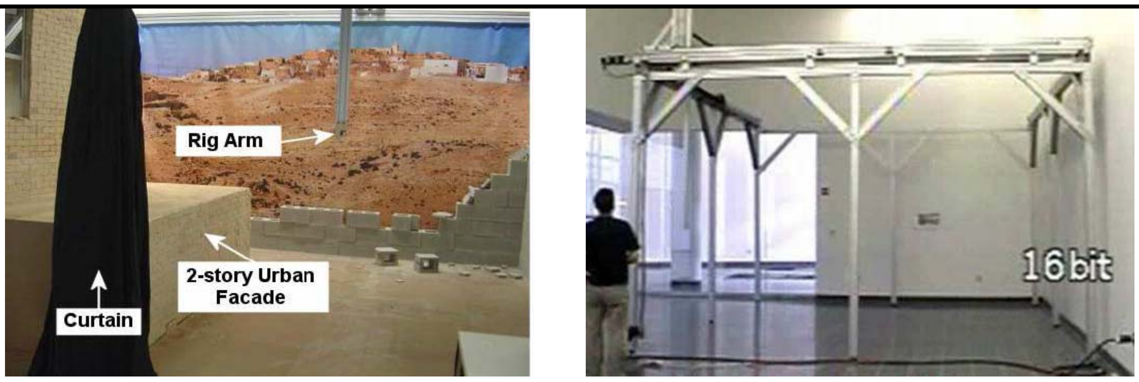


Figure 2.1: Experimental UAV Test Rig

The mockup UAV is retrofitted with collision avoidance sensors. In this set up the sensor data is inputted into a mathematical model of the aircraft. The aircraft is then moved by the 6 degrees of freedom gantry, using model reference adaptive control. The concept is to expose small UAV performance to small places like forests and buildings. The controller structure is used to control the gantry motors, which ultimately controls the position of the mockup UAV. State feedback and proportional integral and derivative (PID) controllers are successfully implemented to position the gantry arm; to control the gantry motor in the z-axis, integral control is necessary to implement gravity compensation. To demonstrate the hardware in the loop test, a collision avoidance algorithm is designed and tested moving the gantry around the small environment using inputs from the sensors. The controller gives a new target position until it is safe to move back to the original position. Within the small environment the mockup UAV is equipped with an infrared sensor (IR) to determine distances from objects. From the analyzed test results it is concluded that the IR sensor responds differently to colors and textures of objects. A problem with using just an IR sensor is the difficulty in distinguishing objects that are similar in shape and color. Using a similar collision avoidance algorithm, the full scale UAV should be equipped with more efficient sensors such as a MV camera.

There has also been a significant amount of research focused on autonomous formation flight which has been performed at WVU[28,29]. Additional formation flight research was also conducted at Georgia Institute of Technology[30,31,32]. Autonomous aircraft flight has proven to be a very important research topic for the aerospace industry. This research focuses on techniques to achieve robustness to specific areas of aerodynamic interference. One specific issue that is addressed in formation flight is the loss of communications. In other particular applications active radio communications should be avoided entirely. In these cases when there is a break of communications between autonomous aircraft, it is impossible for the follower aircraft to know the position of itself with respect to the leader aircraft. An alternative method for radio communications in formation flight would be an aircraft vision system that would be able to interpret the leader aircraft's position and attitude from measured data. The research conducted by WVU focuses on a MV system that is used to detect five markers that are placed on the leader aircraft[33]. In this research, the follower aircraft identifies and detects the markers on the leader aircraft using MV then, using pose estimation algorithms, the relative displacement between each aircraft is estimated. Once the position and attitude of the leader aircraft is determined with respect to the follower aircraft, formation control algorithms are used to allow the follower aircraft to emulate the flight of the leader aircraft.

Formulation flight researched at the Georgia Institute of Technology focuses on performing vision-based formation flight control of multiple aircraft when obstacles are present[30, 31]. This research also addresses the challenge of minimizing communication between the aircraft. The approach to this research is to observe the leaders state of motion through MV techniques. The vision information is then utilized through two different techniques for formation control. One method uses the Extended Kalman filter (EKF) to estimate the relative velocity and position for aircraft guidance. The second method is based on directly utilizing the vision measurements obtained through MV. This is done by regulating the image position to be in the center of the image plane and by regulating the size of the target image. This solution for maintaining formation flight cannot be separated from obstacle avoidance. The controller design for obstacle avoidance in this research is based on a reactive "steer towards silhouette edge" approach. In this research there are several approaches for vision-based formation control none of which required communication between aircraft.

Researchers at the University of California at Berkeley implement another example of vision-based control[34]. This research is based on the vision-based control of a small UAV following the path of a road. The UAV detects specific features of the road in order to provide the necessary inputs to the control algorithms. This allows the UAV to autonomously follow the road. Real time road detection is an essential part of this research, which is achieved by applying various MV techniques such as the HT. In this effort, various pre-processing techniques along with HT are used to detect the road. After the pre-processing stage is completed and a rough location of the road is detected, lane markings on the road are then searched for within the image. Once the lane markings are found, HT is used to test for multiple candidate lanes. The HT, along with a robust line fitting algorithm, is used to determine the center lane marking of the road. Once the road detection process is completed further research is conducted towards developing lateral control strategies which allow the UAV to autonomously follow the road. The first strategy of this vision-based control uses direct PID feedback between the lateral error and the commanded turning angle. The second and third strategies both use nonlinear controllers to aim the aircraft at a point along the road. The third strategy is also used to control the lateral motion of the aircraft with respect to the road. Keeping the velocity of the aircraft constant, the controller commands the aircraft to aim at a desired point along the road. The control signal is determined by creating a geometric relationship between the aircraft velocity and position. The vision-based control systems described in this research are intended for testing on a UAV test bed, but simulations are initially performed to complete tests of the vision system and control algorithms under ideal conditions. The road detection algorithm using HT is implemented and is able to detect ninety percent of the road images presented to the vision system. During the simulation, using the velocity ratio controller, the aircraft successfully tracks the road for 2.5 km.

This is an example of how HT can be useful in aerospace applications for feature detection.

Experimental MV tests have also focused on 'runway detection,' which is displayed from previous work completed at WVU. In this research MV is used to detect lines on a runway during aircraft landing and takeoff. To complete this task it is important to design an algorithm to perform in a real-time or near real-time manner. For this criteria, the software environment, algorithms used, and complexity of the filtering process are influenced. In conclusion to this research effort it is found that the pre-processing stage had the greatest effect on the processing time and must be taken into account when choosing the image filtering methods. A second problem developed when using a line detection algorithm. It is obvious that other items in an image may contain prominent lines, which will interfere with the results of the main line detection algorithm. To resolve these issues there is three essential processes performed. These image processing functions are labeled as edge detection, morphological opening, and HT. Two main types of edge detection methods studied include Gradient Based and Laplacian Based methods. These detection methods are used to detect the edges or straight lines on a runway. Edge detection has proven to be an important image processing tool, but yields inconsistent and unreliable results caused from background noise. To solve this problem, the line-detecting algorithm known as HT is then used in conjunction with an edge detection routine. Therefore, edge detection is used as a low level filtering technique and HT is used as the main line detection algorithm.

The next image processing tool used in this research is known as morphology, which is considered part of the pre-processing stage. Morphology includes eight image processing operations that filter an image based on structural elements. In this research only two of these operations are used, morphological dilation and morphological erosion. This combination is used for noise removal and to find certain shapes in the image that are defined by the structural element.

Morphological operations are further explored in research at Ghent University with grayscale image interpolation[35]. This research designed an algorithm to detect and eliminate jagged edges in magnified images. This task is accomplished utilizing mathematical morphological techniques such as morphological opening and the hit-miss transform. Morphological opening is an image processing technique based on two basic operators, dilation and erosion. Dilation is used to expand objects into the background and is able to eliminate "salt" noise within an object. Erosion is used to shrink objects and is able to eliminate "pepper" noise. When an image is magnified, the number of pixels the image covers is increased, allowing for finer detail, not seen in the original image. This magnification process also introduces jagged edges to the image boundaries, which make it harder to distinguish the true edges. The hit-miss transform is a morphological operator used in the algorithm along with morphological opening. The hit-miss transform can be used to search an image for particular characteristics of a shape or other particular image features. In this algorithm, the hit-miss transform is used to detect corners of a binary image. A problem arose when using the hit-miss transform on grayscale images. The classification of foreground and background pixels is not straightforward, which made the detection of corners more difficult. To account for this, the binarization method is completed only for the pixels that are covered by a specific part of an image. The pixels are classified according to a threshold value.

The algorithm used in the grayscale image, interpolation can be broken into four steps. The first step is pixel replication; the image is magnified by a certain degree, which resulted in jagged edges. The next step is corner detection, which uses the hit-miss transform to determine the positions of corner pixels, both real and false (due to jagged edges). Once the positions of the corners are determined, the next step is to validate the corners as either real or false. The corner pixels created from a jagged edge are labeled as false corners and the real corner pixels of the image are retained. The final step is labeled as pixel swapping, which swaps the binary pixel values, classified as false corners, from 0 to 1 or vise versa. This operation is repeated iteratively until the edge of the image is enhanced and smoothed creating a more refined edge.

A current application of MV being used by the military and other operators around the world is the forward-looking infrared radar (FLIR). The FLIR imaging systems are used to perform missions as diverse as search and rescue, border patrol, and other reconnaissance missions. These mission profiles require a high performance imaging system that can be operated by a variety of airborne platforms. One type of FLIR used by the U.S. military is the FLIR Star SARIRE^{TM} HD[36]. This system incorporated with MV technology, will allow UAVs to track, range find, and locate targets at high resolutions with a 25 kilometer range. This type of performance makes it extremely useful with airborne platforms. This technology offers wide-area surveillance and over the horizon detection and tracking that otherwise may go undetected by surface based sensors. This is a prime example of the capabilities of MV with airborne platforms, which will continue to increase in popularity as technology advances.

2.3.2. Electrical and Computer Engineering

In addition to aerospace engineering, other applications of MV are found in areas such as electrical and computer engineering. Examples in these areas focus on enhanced edge based segmentation methods to perform different image processing techniques.

The robustness of the HT and its usefulness in detecting circular features in an edge-enhanced image has, again, been proven. A technique at the department of electronic engineering at the City Polytechnic of Hong Kong is aimed at improving the efficiency of circle detection using HT and to reduce the size its accumulator array[37]. This approach focuses on approximating a circular image with a set of straight line segments. Based on this technique, only a two dimensional (2-D) accumulator array is necessary. This path is chosen because the HT is a robust technique mainly used for straight line detection within edge enhanced images; using the HT to detect circular images can be a time costly operation depending on the circumstances. The implementation of this method for circle detection is proven to have a shorter processing time, use less memory, and perform more efficiently. In this designed method a circular object is represented by four sets of vector pairs. The corresponding "vertex" position of the circle is calculated using the HT for line segment patterns and the results are accumulated in a 2-D array. The accumulator array is then used to estimate the presence and location of the local peaks, which will determine the location of the vertices of the corresponding circles. This research shows that the proposed algorithm for circle detection gives satisfactory results in noisy environments and determines the location of partial circles.

Another application of HT for circle detection is shown in the work completed at Hunan University[38]. This research proves how the GHT for circle detection can be enhanced to improve the detection time and detection rate. Their proposed method involves an adaptive randomized HT algorithm. In this algorithm a moving window is defined that moves over the entire image iteratively. Over each window HT is used to perform circle detection. Therefore, the original detection task for a large edge image is broken into a number of detection tasks for smaller sub-images. This method is introduced to suppress the effect of noise on the detection precision and increase the detection rate. Two issues to consider when performing this algorithm are the size of the moving window and the length of the moving step for the window. If these parameters are chosen correctly then the proposed algorithm is found to produce a faster running speed and a higher detection rate.

2.3.3. Manufacturing Industry

Current technology has limited MV to specific areas of manufacturing such as inspecting products on a conveyor belt. This entails the use of MV for a variety of visual inspection applications identified as:

- identification;
- recognition;
- gauging;
- flaw detection.

Industry has traditionally relied on manual control to manage these production processes, which require humans to visually inspect and control the entire manufacturing procedure. However, the manufacturing industry can be quite tedious for a human as it requires repetitious work for long periods of time. With this in mind, it is known that humans tend to be easily affected by distractions, boredom, and restlessness. These traits can affect the overall quality and reliability of human workers within the manufacturing industry. Therefore; MV is favored in this field of work because it offers several advantages over human operators. These advantages include: non contact operation, high speed inspection, long term continuous inspection, consistency and accuracy, and cost reduction. MV technology has begun to aide manufactures by performing a variety of important functions which have diminished the economic disadvantages associated with employing human inspectors.

Applications of MV are often used in the food industry. A typical application is displayed in the assessment of fruit quality[39]. Many inspection systems in the food industry utilize some type of camera system to categorize their product. In this case an industry has to deal with large quantities of citrus harvested every year, which has to be classified depending on their quality. To complete this task, the quality of the fruit is classified using MV systems to analyze their external features such as the color, size, and skin defects. This type of system will eliminate the necessity of using humans to classify the fruit manually, which will allow for a much faster inspection process. This MV system segments the colorful fruit from the background by using a threshold technique on the digital image. The next step consists of using a boundary extractor to detect the boundary of each piece of fruit in the scene. The performance of this system is not perfect due to bad positioning of fruit which can occur when a boundary in the image belongs to two or more fruit. For this reason, a specific algorithm is used to detect this case of bad positioning and separate the individual fruit by detecting the contact points between the fruit edge boundaries. Once the individual fruits are segmented, the fruit surface is then inspected, which segments the blemishes on the surface from the rest of the image. Fruit with large blemishes are then sorted from the batch.

In another manufacturing application the use of HT is very attractive when the detection of specific shapes or straight lines is necessary. In particular, this MV technique is utilized for quality inspection of rice seeds[40]. Combined with some image pre-processing techniques, HT accurately separates the rice seeds into three categories based on their distinguishing characteristics. The separation into their respective categories is based on the location and number of Hough peaks on each seed. This becomes a useful tool for external feature measurement and another reason why automatic inspection using MV can improve the quality of the product, reduce the necessity of human operation, and eliminate flawed manual inspections.

2.3.4. Medicine

As stated previously, MV is helping manufacturers maintain efficient productivity, but that's not the end of its relationship with inspection systems. MV is being used to track and inspect medical devices as well as improve the capabilities of digital imaging systems. Within the medical industry, MV is being used for three types of applications:

- automated product inspection;
- imaging;
- product tracking.

Imaging has proven to be an important diagnostic tool within the medical field and is used in many medical procedures. Medical imaging systems are among the largest application areas for MV. These systems consist of magnetic resonance imaging (MRI), computed tomography (CT), and digital x-rays. Advances in this area of medicine have proven to be very useful for improving medical procedures used to reveal, diagnose, or examine the human body. MV is not exclusively used for diagnostic procedures, but has also become very important for people with vision problems where specific MV techniques are being used to enhance their sight. It is evident that there is a great deal of research being performed, utilizing MV applications to improve the medical industry.

Using MV to aid in image recognition for the visually impaired is becoming a very popular topic of research[41]. In a recent paper a method is designed that utilizes MV techniques to read and interpret visual displays that would otherwise be difficult to interpret for visually impaired people. This research focuses on text detection using a threshold method to distinguish the text from the background, which allows the text to be converted to a binary form. While conducting this research effort some error is found when there is bad illumination in the original image causing some of the pixels to fall below the threshold value. This is solved by performing two iterations of erosion on the new image. Once the text is detected the next stage performs character recognition. In this process the extracted image is compared to a template of the character to be matched. The basic approach of this method can lead to false interpretations because of the range of variability between characters. To account for this, it is defined what characters are allowed in each region of the image; the recognition module is provided with a set of templates to account for the variability. This is yet another example of thresholding being used as a pre-processing method to a more complex processing technique.

In nuclear medicine brain scanning is one of the most widely used procedures. This procedure is useful for detecting brain abnormities such as tumors, lesions, or head trauma. Research has been completed that utilizes MV techniques to perform these kinds of procedures[42]. When using a MV system for brain scanning the initial task is to segment the brain from the image. In this research this is completed using the technique of variable thresholding and region classification. Thresholding is used to filter out the brain from the background. Once this is completed a more advanced region based technique is used to classify the areas within the brain. Once the brain regions are identified the next step is to identify any abnormities within the brain. This procedure is very similar as the one used to identify the brain. Therefore; with some modifications, similar techniques are used to design an algorithm to locate the abnormities within the brain tissue. A problem occurs when detecting tumors that lie along the boundary of the brain, which tend to result in a false boundary extraction using only a thresholding technique. To solve this problem a pattern recognition technique is used that compares the boundary segments of normal brain images to those of patients with tumors. In this research thresholding is used as a pre-processing technique allowing more advanced techniques to be utilized to process the image.

The importance of using MV in medical imaging is also displayed in research completed at St. Mary's University. In this research effort MV is utilized by segmenting a moving organ in a CT scan[43]. A number of image segmentation techniques are used and compared to properly segment features within the scan. It is determined that segmenting images from a CT scan can be difficult due to the fact that the organs do not have uniform density and there can be large amounts of noise caused from the moving organs. The techniques used in this research for image segmentation are classified as: thresholding, pixel classification, and edge-based approaches. It is made apparent that each segmentation method has its advantage over the other. The pixel classification method known as the K-Means Clustering algorithm is proven to be more effective in speed and in detecting small variations in pixel intensity values within the different regions of an organ.

Similar to other applications of MV, HT also plays an important role within medical imaging; this is yet another example which displays its versatility. Research completed at Texas Tech University uses an approached based on the GHT to develop a robust segmentation technique to locate the cervical vertebrae within x-ray images[44]. To achieve good success with this segmentation approach it depends on three parameters: accumulator votes, gradient information, and template representation. Since x-ray images typically do not represent clear edges within the image it is necessary to perform Gaussian filtering before edge detection if performed on the image. Once this is completed the next step is to choose a template representation of the target image. When using HT this is a critical step to obtain the necessary number of votes in the accumulator matrix. The task of segmenting the cervical vertebrae with GHT is completed using a number of templates with different vertebra orientations. This makes the process very robust for the fact that bone growth can vary drastically from person to person. The HT approach is very useful for detecting features with unique characteristics. Using MV for vertebra inspection is likely to reduce error resulting from a physician’s diagnosis process.

Another example of the HT being used in the medical field is shown from the research completed in China which uses HT for iris segmentation[45]. For iris recognition both the inner and outer boundary of the iris can be classified as circles, therefore; the circular HT is utilized to detect these boundaries. It is determined that when the quantity of data to analyze is large, the HT method is difficult to use in a near real-time manner. To reduce the amount of data, a thresholding technique is used to eliminate unwanted features in the image. This will reduce the computation time for the HT algorithm by reducing the amount of edge points in the image. This is yet another application of HT that shows that this technique has high accuracy and is a good method to detect object edges.

2.3.5. Agriculture Field

The agriculture industry is another field that utilizes MV technology in many applications. One particular application is shown by research completed at the University of Illinois. In this work the K-Means Clustering algorithm is used to detect weeds in a soybean field[46]. To accomplish this task a MV system is employed on an herbicide spraying machine. This machine is then used to scan the soil rows between the soybean rows for specific weeds. The spraying machine applies MV algorithms to the captured images from the camera to distinguish particular plants. In each captured image a K-means cluster analysis is used to associate individual pixels in the image with specific classes based on their characteristics. Each weed that is detected will then be sprayed with the herbicide and its location will be stored for later evaluation. Once the plant is classified as a weed and sprayed by the machine, it can later be evaluated to determine if it is killed or not. This process eliminates the necessity of the farmer to manually evaluate the soybean fields saving time, money, and energy. The results have proven that MV can be utilized to increase time efficiency and cost effectiveness when large farm lands and crops need to be evaluated.

Chapter 3. Theoretical Background

3.1. Overview of Theoretical Background

The theoretical approach to this research is broken down into two stages. The first stage involves the image processing methods used in this research, which include two distinctive layers. These layers are described as the pre-processing stage and the higher level processing methods. The pre-processing stage consists of low to medium level image processing functions which are used to help reach the overall goal of target recognition and tracking. Without the exploitation of these functions it would make it impossible to obtain results from the higher level methods. The significance and theory behind each of these functions will be explained in detail in Section 3.2. The second layer of the first stage is categorized as the higher level functions used to complete the main task of image recognition. The main algorithms and methods used in this layer will be explained in Section 3.3. The second stage of this research involves the target tracking problem. This stage utilizes the results obtained from the first stage to produce the necessary input to achieve continuously target tracking. The methods and techniques necessary to complete this task will be discussed in Section 3.4.

3.2. Image Processing Functions

As discussed above, this section will explain the pre-processing methods used in the first layer of the image processing stage, which include the image acquisition process and enhancements methods. To obtain a full understanding of the methods in the first layer, it is necessary to provide an explanation of the simplest and most appealing areas of digital image processing: the representation and definition of a digital image. This is discussed in Sections 3.2.1 and 3.2.2, which cover the coordinate conventions used throughout this research and the definition of a digital image, respectively. Section 3.2.3 defines the methods used for digital image enhancement. This process is useful for enhancing specific image features in the target detection process. Finally, Section 3.2.4 will discuss the theory and importance of thresholding within digital image processing, which is used in the edge and target detection methods.

3.3. Coordinate Convention

The overall objective of a MV system is to analyze digital images measured from a vision sensor. To accomplish this goal it is necessary to convert the sensed data into a digital form. This conversion process results in a quantized matrix consisting of real numbers. This data matrix, or digital image, will be represented with the same coordinate convention throughout this research. To explain the conversion process, it is first assumed that a digital image, *f*, is sampled and is represented using the nomenclature, *f*(x,y). The resulting digital image is the a matrix with *M* rows and *N* columns where integer values represent each spatial (plane) coordinate, (x,y). For example, the value of the coordinate located at the origin is (1,1), and (2,1) is used to represent the second coordinate along the first row. Figure 3.1 displays the coordinate convention used throughout this research.

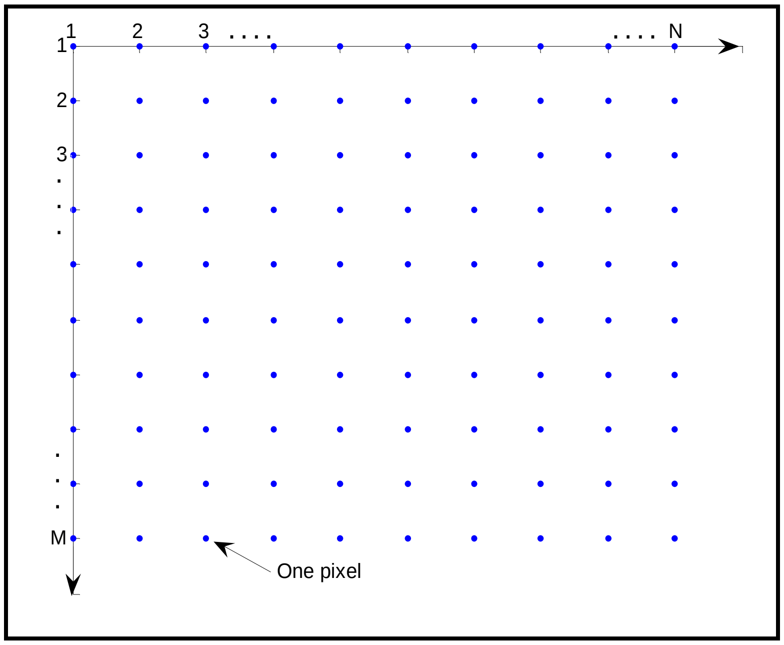


Figure 3.1: Coordinate Convention for Image Data Matrix

The coordinate system in Figure 3.1 will result in the following representation for a digital image:

$$f(x,y)=\begin{bmatrix} f(1,1) & f(1,2) & \ldots & f(1,N) \\ f(2,1) & f(2,2) & \ldots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \ldots & f(M,N) \end{bmatrix} \tag{3.1}$$

The right side of the equation represents the digital image, where *x* and *y* are discrete integers that fall between the ranges: x = [1,M] and y = [1,N]. Each spatial coordinate in the digital image then represents a pixel or image element that is categorized as either color or intensity level.

3.3.1. Image Definition

The image obtained from the MV camera is a color image, which is represented by three components corresponding to the red, green, and blue components; for this reason it is referred to as an RGB color image. For this reason, an RGB color image has a quantized representation of an *M*x*N*x3 array of color pixels, where each pixel value is a triplet at a specific location in the digital image. To better understand this representation, this type of digital image can be compared to a combination of three gray scale images stacked on top of each other, that when fed into the red, green, and blue inputs of a color display result in a color image displayed on the screen. The range of values that represent each pixel value is determined by the data class of the image.

Converting between data classes is a frequent operation when using image processing operations, which is completed several times in the MV software. If an RGB image is of class double then the range of values is [0,1]. For RGB images of class uint8 (8-bit) or uint16 (16-bit) the range of values is [0, 255] and [0, 65535], respectively. When using the standard 8-bit case to represent a RGB color image it has 16.8 million possible colors. Other possible representations include 32-bit and 64-bit, which result in a larger color range for the image representation. The data classes used throughout this research are double and 8-bit. An example of an RGB image of class double is displayed in Figure 3.2. This type of image definition is used in Matlab® software environment.

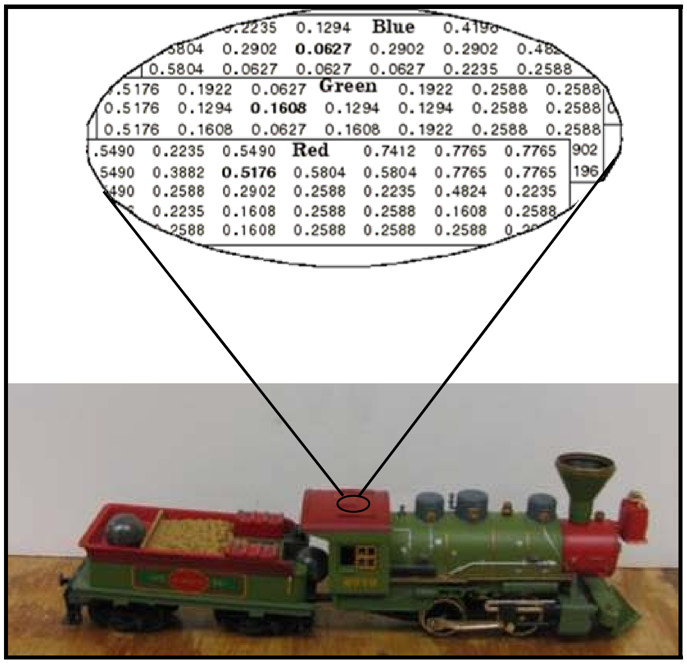


Figure 3.2: RGB Color Image

An RGB color image can sometimes be referred to as an indexed image, which consists of a data matrix of integers as well as a colormap matrix. The colormap matrix is an *m*x3 array of class double where *m* is the number of defining colors. Each row of the colormap specifies the red, green, and blue components used in a specific pixel color in the image. This representation of an RGB color image is usually not displayed unless a colormap is defined. Regardless if the colormap is defined or not, this image representation always occurs. This method is useful when it is necessary to represent different variations of a certain color. For example, an infinite number of shade variations can be used to represent one color, which can then be used to construct an image. The colormap concept is shown in Figure 3.3.

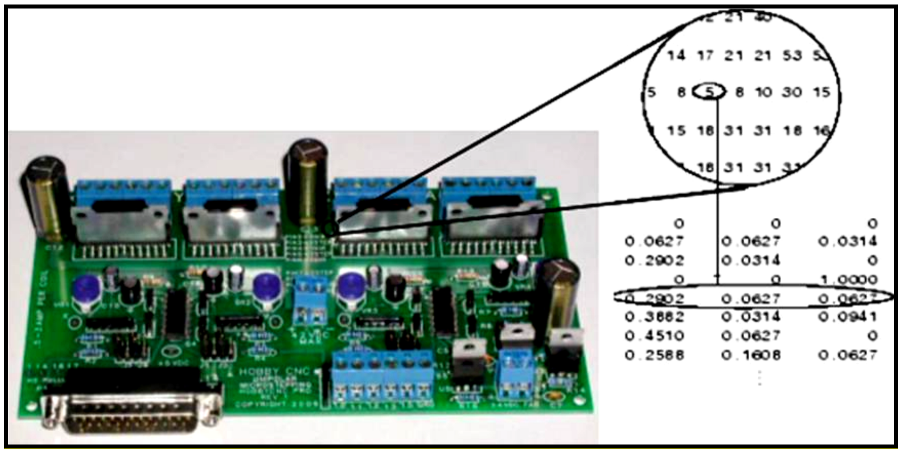


Figure 3.3: Colormap Representation for RGB Image

The input image type used in the image processing software is referred to as a grayscale image, which is a data matrix that represents brightness intensities within a range of pixel values according to the data class. In other words, for an image of class double the value of *f* at any spatial coordinate will have values ranging between 0 and 1 where an intensity of 0 represents black and an intensity value of 1 represents white. When comparing grayscale images to RGB color images, a grayscale image is not represented by a colormap. This is true because it is represented by a single data matrix where each element of the matrix corresponds to one pixel value, unlike a color image where each element corresponds to three pixel values. A grayscale image can be compared to a single color component that makes up a color image, with the exception that the image is not represented by color but a brightness level from black to white. When discussing similarities between the two types of images, the discrete representation of the images has no limits except those set by the image processing software. An example of a grayscale image is shown in Figure 3.4.

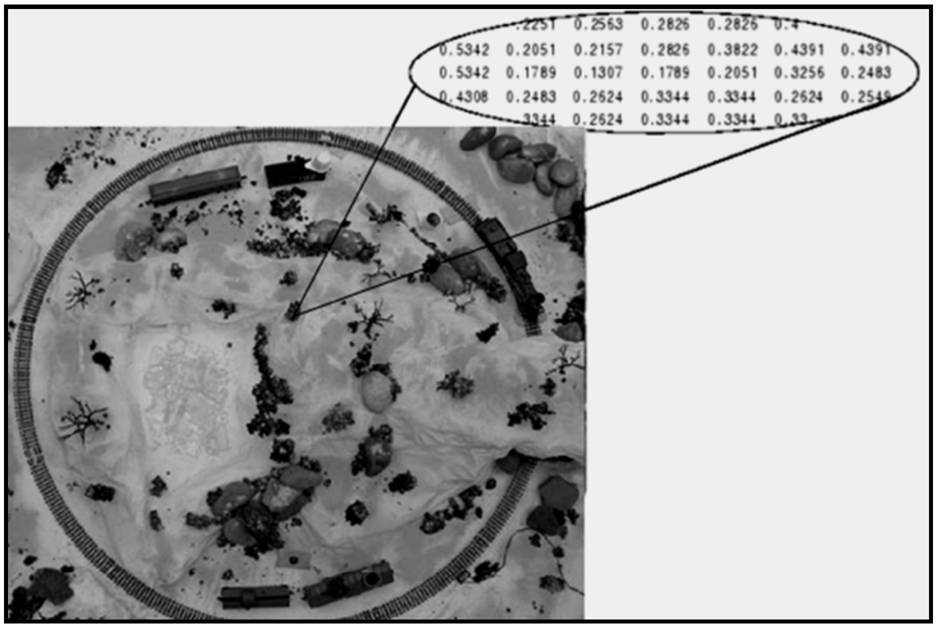


Figure 3.4: Intensity Image Construction

3.3.2. Intensity Transformation

Intensity transformation methods are used to directly manipulate the pixels in the spatial domain. The spatial domain techniques discussed in this section are represented by Equation (3.2) [47]:

$$g(x,y) = T[f(x,y)] \tag{3.2}$$

Where *f(x,y)* is the input image, *T* is the operator that is performed on the input image, and *g(x,y)* is the processed output image. Several gray-level intensity transformation methods are analyzed during this research. These methods include:

- logarithmic transformation;
- contrast-stretching transformation;
- negative transformation;
- gamma correction.

For this particular research effort, the best results are obtained when contrast-stretching transformation is applied to the digital images. This operation is useful in this research to help illuminate specific features on the target.

Contrast-stretching transformation is a piecewise linear function used for dynamic range manipulation for each pixel value[47]. This transformation method is commonly used for low-contrast images resulting from poor illumination, an inadequate dynamic range in the sensor, or an incorrect lens setting. In this research, this method is utilized to manipulate the range of gray level pixel values in the digital image to illuminate a specific feature on the target. The typical transformation used for contrast stretching is shown in Figure 3.5. The locations of the points *(r_1,s_1)* and *(r_2,s_2)* control the output of the transformation function.

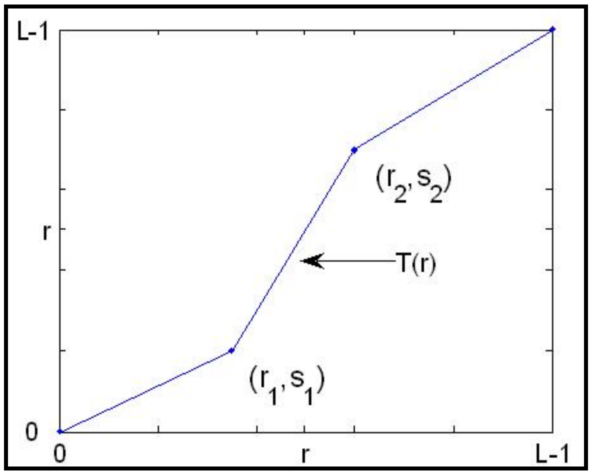


Figure 3.5: Contrast-Stretching Transformation Function

The function shown in the figure above is represented using the notation expressed in Equation (3.3)[48]. This representation is expressed in equation shown below:

$$s = T(r) = \frac{1}{1 + \left(m/r \right)^{E}} \tag{3.3}$$

where *L* represents the number of input pixels, *r* represents the intensities of the input image, *E* controls the slope of the function, and *s* is the intensity value of the output image. This equation is implemented on each pixel in the digital image; it compresses the input values lower than *m* into a narrow range of dark levels and compresses the input values higher than *m* into a narrow range of bright levels in the output image. The illustration shown in Figure 3.6 shows an example of an input and output image when Equation (3.3) is applied to the digital image. The input image shown in Figure 3.6a is a typical intensity image. The result of applying contrast-stretching to the digital image is then shown in Figure 3.6b.

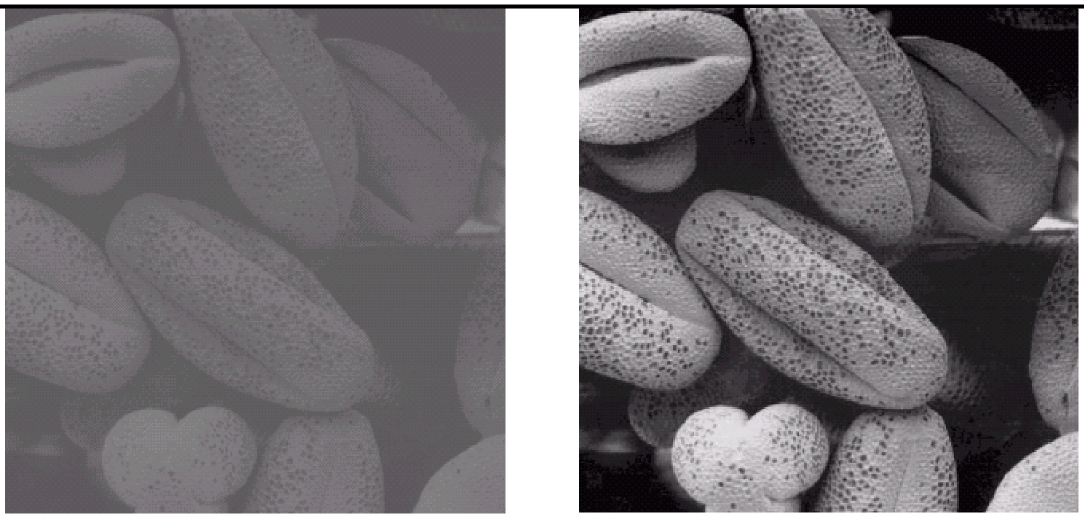

(a) Original Low Contrast Image (b) Result of Contrast-Stretching

Figure 3.6: Example of Contrast-Stretching Transformation

3.3.3. Color Level Conversion

Color level conversion is described as the process of converting an RGB image into an intensity image. This process results in an MxNx3 data matrix transformed into an MxNx1 data matrix, which has its obvious advantages. The first noticeable advantage is the reduced complexity of an image. The size of the data matrix is drastically reduced as a result of this conversion process; the representation of an intensity image is a third of the amount of data used to represent an RGB image.

This image conversion process is completed using the Matlab® software environment by the function, *rgbtogray*, which converts RGB colors to grayscale values by forming a weighted sum of the R, G, and B components of the image. This process is represented by Equation (3.4)[49] below:

$$Intensity = \begin{bmatrix} 0.2989 & 0.5870 & 0.1140 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.4}$$

where the R, G, and B components are the color level values for each respective pixel in the original color image. The class of the output intensity image will match that of the input RGB image. So, the pixel values of the output image will be represented by the same range of values used in the input image. An example of this process is shown in

Figure 3.7. The image shown in Figure 3.7a is the input image represented by a three-dimensional data matrix, one for each color component. The image shown in Figure 3.7b is the output image which is now represented by a one-dimensional data matrix.

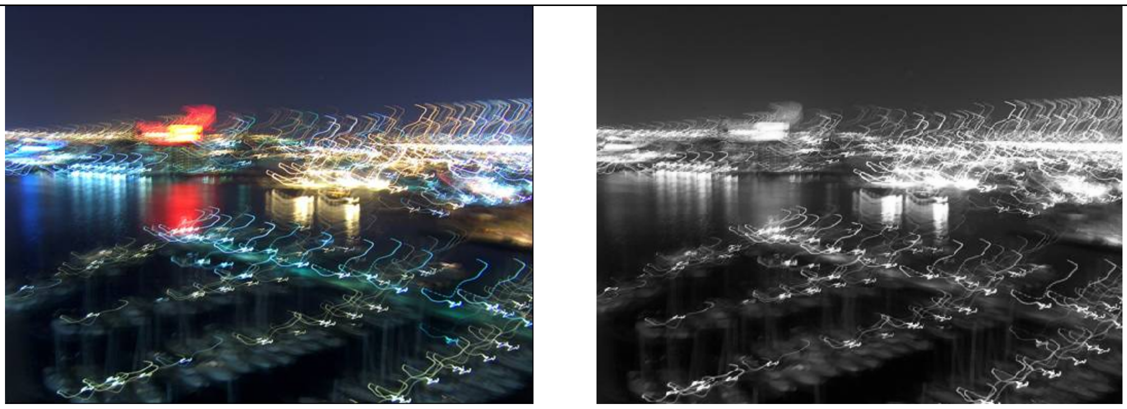

(a) RGB Color Image (b) Intensity Image

Figure 3.7: Example of Color Level Conversion

3.3.4. Thresholding

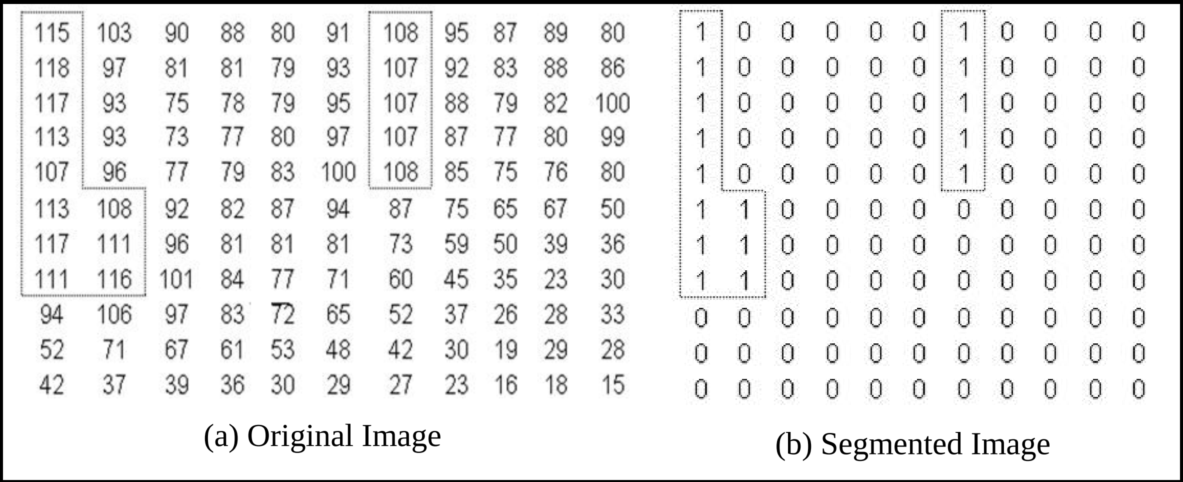
Thresholding is an image segmentation technique that is very useful for filtering images from homogenous features. This technique is an important feature for many different image processing methods and algorithms. The most common application of thresholding is its utilization when extracting objects from the background of an image where distinctive modes separate the foreground from the background. In this case a threshold value, T, can be defined where any pixel location, (x,y), for which $f(x,y) > T$ is called a foreground point; otherwise, the pixel is labeled as a background point. Thresholding also plays an important role in edge detection methods which will be explained later in this chapter.

For grayscale images the threshold operation is effective in extracting objects which have a distinct ranges or patterns of grey level values. Another important application of thresholding, which is utilized in this research, is its use to reduce the amount of noise in the image by eliminating undesired image features, which enhances the signal to noise ratio. Reducing the noise in an image from this simple technique drastically decreases the computation time for the higher level image processing methods.

The mathematical representation of thresholding is the best way to obtain a full understanding of this segmentation process. Once the mathematical representation is understood, the operation can easily be applied to any image. For example, if a given image, *f*, has the gray level range [I_1,I_k], and *t* is any gray level value between I_1 and I_k, the result of thresholding the image at *t* is the binary image represented by f_t defined by Equation (3.5).

$$f_t(x,y) = \begin{cases} 1 & if \quad f(x,y) \geq t \\ 0 & if \quad f(x,y) < t \end{cases} \tag{3.5}$$

This procedure is used to eliminate the noise in the image that is represented by gray level values below the threshold value, *t*, which will obtain a new pixel value of 0. The image pixels that are of interest will be represented by a pixel value of 1, producing a binary image. A visual representation of this process is shown in Figure 3.8. Figure 3.8a represents an 8-bit grayscale image with corresponding pixel intensity values. The threshold, *t*, for this particular example is chosen to be the pixel intensity value of 106. Therefore, following the mathematical representation in Equation (3.5), the resulting segmented image is shown in Figure 3.8b.



115	103	90	88	80	91	108	95	87	89	80
118	97	81	81	79	93	107	92	83	88	86
117	93	75	78	79	95	107	88	79	82	100
113	93	73	77	80	97	107	87	77	80	99
107	96	77	79	83	100	108	85	75	76	80
113	108	92	82	87	94	87	75	65	67	50
117	111	96	81	81	81	73	59	50	39	36
111	116	101	84	77	71	60	45	35	23	30
94	106	97	83	72	65	52	37	26	28	33
52	71	67	61	53	48	42	30	19	29	28
42	37	39	36	30	29	27	23	16	18	15

(a) Original Image

1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

(b) Segmented Image

Figure 3.8: Example of Thresholding Process of Intensity Values

3.3.5. Sobel Edge Detector

The Sobel edge detection method is utilized in this research effort which has proven to be a common method for detecting discontinuities in gray level pixel values. This segmentation method is formally introduced in Section 2.2.1. The focus of this section is based on the theory behind this particular method. Some concepts previously introduced in the earlier section will be briefly stated again to be able to establish connection.

In many cases edge detection operators are used in conjunction with other image processing techniques, which is why they are so popular in image processing applications. There are several ways to perform edge detection, but the majority of the methods are classified into two categories known as the gradient method and Laplacian method. Gradient methods can further be broken down into specific edge detection operators known as the Roberts, Prewitt, and Sobel. Each of these gradient methods are used to approximate the pixel intensity gradient level of an image, *f(x,y)*, at each pixel location. The difference recognized with each operator is evident in their respective convolution masks, which will be explained later in this section. The implementation of each operator is used to determine where edges exist in images, which are areas with strong intensity contrasts.

The Laplacian method, unlike the gradient method, is represented by one operator only, known as the Canny edge detection operator. This method for edge detection is generally not used in its original form. As a method that calculates the second-order derivative of the image, which yields a double-edge image, it is usually very sensitive to noise and it is unable to calculate edge direction. This method is commonly combined with Gaussian smoothing to reduce the amount of noise, which is why it is sometimes referred to as the Laplacian of a Gaussian. The Canny edge detection operator is then utilized to determine the location of the edges via zero-crossing of the second derivative of the Gaussian smoothed image.

As previously stated, the Sobel operator is classified as a gradient method for edge detection. A gradient based method is used to determine the first-order derivative of a digital image, which can be approximated using the Sobel operator. The gradient of a pixel located in a digital image will have the representation shown in Equation (3.6).

$$\Delta f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\delta f}{\delta x} \\ \dfrac{\delta f}{\delta y} \end{bmatrix} \tag{3.6}$$

The important value used in the edge detection process is the magnitude of Equation (3.6), which is represented by Equation (3.7) below.

$$\Delta F = mag(\Delta f) = \left[G_x^2 + G_y^2\right]^{1/2} \tag{3.7}$$

The operation described by Equation (3.7) should be implemented for each pixel in the image to determine the maximum rate of increase of *f(x,y)* per unit distance in the direction of Δf. To re-state this direction, this operation is used to return edges at the points where the gradient of the image is a maximum. For example, an edge in a digital image is represented by a group of pixels whose intensity level decrease and increase as the line is approached. The edge pixels are determined by the pixels that have the peak gradient magnitude within the group of pixels. To begin this process, the partial derivatives shown in Equation (3.6) can be approximated for an entire image by applying the convolution mask representing the Sobel operator displayed by Equation (3.8)[47] and Equation (3.9). Each of these masks is used to calculate the component of the gradient in the x-direction, G_x, and component of the gradient in the y-direction, G_y, respectively.

$$G_x = \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & -2 \\ -1 & 0 & -1 \end{bmatrix} \tag{3.8}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.9}$$

This procedure is displayed using a sample 3x3 pixel group shown in Equation (3.10) which represents an area of gray level pixels within a digital image.

$$I = \begin{bmatrix} i_1 & i_2 & i_3 \\ i_4 & i_5 & i_6 \\ i_7 & i_8 & i_9 \end{bmatrix} \tag{3.10}$$

Implementing each of the masks previously explained on the neighborhood of pixels, the resulting gradients are calculated using Equation (3.11) and Equation (3.12).

$$G_x = (i_7 + 2i_8 + i_9) - (i_1 + 2i_2 + i_3) \tag{3.11}$$

$$G_x = (i_3 + 2i_6 + i_9) - (i_1 + 2i_4 + i_7) \tag{3.12}$$

This procedure is followed for each edge operator; the only difference lies in their respective matrix representations.

As previously stated, the masks in Equations (3.8) and (3.9), are used to obtain the gradient components in the *x* and *y* directions. Computation of the magnitude gradient requires that the two components be combined in the manner displayed by Equation

(3.7). However, both masks can be applied to an image to compute the gradient components separately. This is useful in cases where either vertical or horizontal lines are the edges of interest. The procedure for this method is shown below using a gray scale picture of a circuit board.

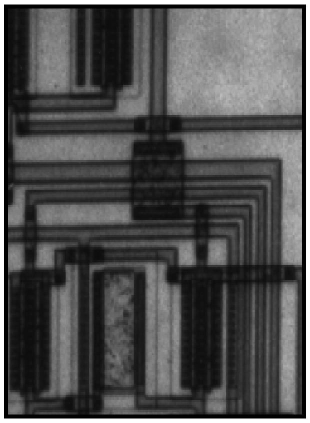
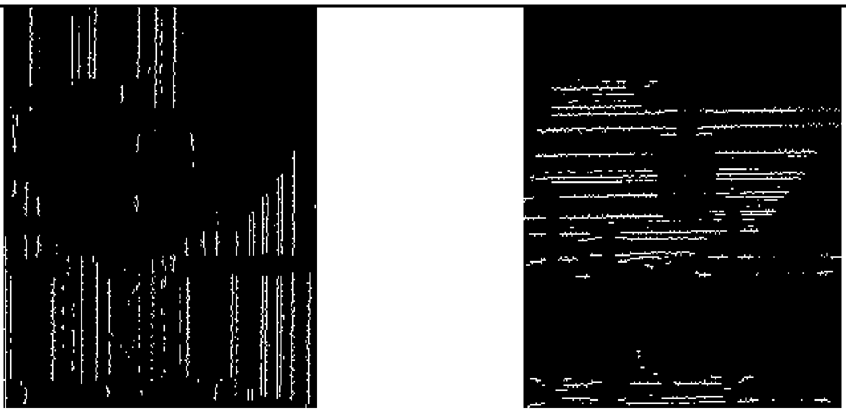


Figure 3.9: Intensity Image of a Circuit Board

From this point the Sobel operator is applied to the image to detect the horizontal and vertical edges. Therefore the masks shown in Equations (3.8) and (3.9) are applied to the image and the results are shown in Figure 3.10a And Figure 3.10b, respectively.



(a) Horizontal Edge Detection of Board (b) Vertical Edge Detection of Board

Figure 3.10: Sobel Edge Detection Process

These results can then be combined together to determine the gradient magnitude matrix following Equation (3.7). The resulting image is shown in Figure 3.11 where each edge is detected in the circuit board. This is the exact procedure that Matlab® follows when the edge detection algorithms are utilized within the software. A binary image is produced by applying a threshold algorithm to filter out insignificant edges. From this point forward the background pixels are filtered out and labeled as the black pixels and the edges are then defined by the white pixels. The sensitivity threshold value is user defined where all edges weaker than the threshold value will be ignored.

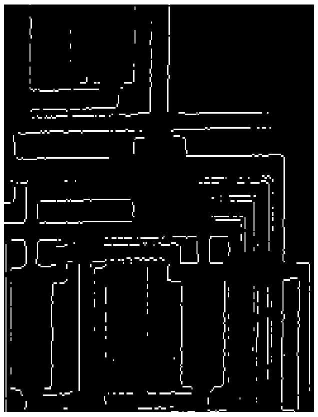


Figure 3.11: Result of Sobel Edge Detection Process

Edge detection is an essential image pre-processing technique utilized in the target recognition portion of this research. In order for this pre-processing technique to be effective the outline of the feature must be obvious.

3.4. Circle Detection

The detection of circular objects is very important in this research effort, acknowledging the fact that circular patterns suggest the existence of important features on the target. To accomplish this task, the method that immediately presents itself is a low level edge detection method that involves boundary tracking. Unfortunately, these types of algorithms are insufficiently robust and tend to produce inadequate results. Therefore, it is necessary to utilize edge detection algorithms that are able to overcome many of these problems. Two methods that can be effective in this research application are the HT method and an estimation technique known as the fast, finding and fitting (FFF) algorithm for circle detection[50]. These methods are found to be particularly robust and efficient. Both of these techniques are investigated for their achievability of accurate circle detection.

The HT is considered to be a higher-level image processing technique for edge detection. In most applications, this method is used for line detection but can be used to detect arbitrary shapes. The application of the HT for circle detection appears to be the most straightforward use of this technique. Using this technique in combination with a low level edge detection method proves to be a sufficient method for circle detection. To accomplish this task, HT uses a mathematical transformation in combination with a global search method to search for peaks in the output matrix which correspond to the center location of circular objects. Since edges have a nonzero pixel width, noise will always affect the process of peak location.

Although HT proves to be an adequate method for circle detection, its speed and accuracy can be a concerning issue. Depending on the amount of edge pixels in the input image, significant amounts of time can be spent transforming the image into parameter space and locating the global maximum in the data matrix. When considering these issues it is important to have a secondary method for circle detection that is less computationally costly. Based on the research for circle detection methods, the FFF algorithm is chosen as the second method for circle detection. This method uses the circle's geometric symmetry and an estimation theory to estimate locations of circular features. Based on the estimation theory, least-square estimators[51] are used to estimate the origin and radius of circles within a digital image. This process along with the HT approach will be taken and the results of each solution will be compared.

3.4.1. Hough Transform

The HT application is typically used as a line detection method through mathematical transformation. More common line detection methods, such as Sobel edge detection, typically identify pixels lying only on edges. However, the resulting edge pixels seldom fully characterize a line because of noise and other reasons that cause the intensity levels of an edge feature to be altered. Therefore, the HT is used as a higher level edge detection algorithm to determine meaningful line segments in an image. Although the focus of the HT is typically on straight lines it is also be applied to any function of the form $g(v,c)=0$, where v is a vector of coordinates and c is a vector of coefficients. This description is very appealing for circle detection. For circle detection, the principle of this method is to recognize that there are an infinite number of circles that pass through a given point. The HT is used to determine which of these circles pass through the greatest number of points.

To determine if a number of points lie on the same circle it is necessary to define a representation of a circle that enables the comparison between the points, which is the procedure that the HT follows. Considering an edge point, *(x_i,y_i)*, the HT utilizes the general representation of a circle shown in Equation (3.13).

$$(x-a)^2+(y-b)^2=r^2 \tag{3.13}$$

Infinitely, many circles pass through *(x_i,y_i)* while satisfying the equation of a circle for varying values of *a*, *b*, and *r*. Equation (3.13) can be oriented to the form shown in Equation (3.14).

$$a=x\pm\sqrt{r^2-(y-b)^2} \tag{3.14}$$

where the ab-plane, or parameter space, is now considered. Using this orientation the value of *a* is calculated for every possible value of *b* for a fixed radius, *r*. This means the shape of a single circle can be represented for a fixed pair of coordinates, *(x_i,y_i)*. A second point, *(x_j,y_j)*, also represents its own distinctive circle in the parameter space. If these points lie on the same circle in the image space, then the circle in the parameter space intercepts the circle associated with the point *(x_i,y_i)* at *(a',b')*, where *a'* and *b'* represent the center coordinates of the circle in the image space. All *(x,y)* points that make up a circle in the image space each represent different circles in the parameter space which all intersect at (a',b'), in theory. This concept is displayed in Figure 3.12 where three points taken from the geometric space represent three circles in the parameter space shown by the green dashed lines. This is performed by inputting each *(x,y)* coordinate into Equation (3.14). The intersection of each circle in Figure 3.12 is shown by the red dot, which gives the center coordinates of the circle shown in the geometric space. This is the procedure that the HT follows for circle detection.

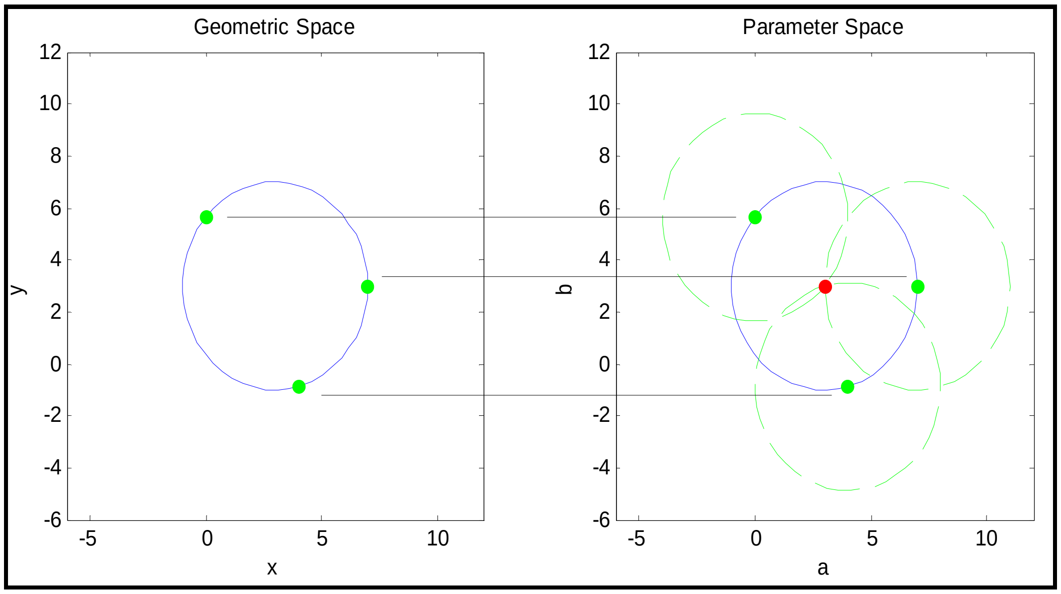


Figure 3.12: Hough Transform Process for Circle Detection

The computational attractiveness of the HT arises from subdividing the parameter space into accumulator cells. The cells combined then represent the accumulator matrix, *A(i,j,),* which is initially set to 0; the size of the matrix depends on the image space expected for the location of the center coordinates. Respectively, for every edge point in the *(x,y)* plane, the parameter *a* is calculated based on Equation (3.14) in which the parameter *b* is equal to each of the allowed subdivision values on the b-axis. Each calculated value of *a* is then rounded off to the nearest value on the a-axis. If a value for b_i results in a solution for a_j then $A(i,j) = A(i,j) + 1$ to account for the spatial coordinate system. When this procedure is completed for every edge point, then the value in each accumulator cell in the accumulator matrix represents the number of points that make up the circle in the *x-y* plane with the defined radius. The local maxima of the accumulator array corresponds to the center location of the detected circle in the image. This algorithm is much easier to implement when the radius of the circle is known prior to the detection process. When this isn't the case, the accumulator matrix would be represented by a three dimensional matrix rather then two dimensions as explained above. The computation time would increase drastically in this situation.

Once the list of *a* and *b* values for a particular edge point is determined, then the circle is plotted in the ab-plane. This process is continued until the *a* values calculated, from the range of *b* values, are plotted for each edge pixel in the input image. Once this task is completed then a threshold function determines the location of peak values in the parameter space. The location of peaks in the parameter space corresponds to possible locations of circular origins. Each circular origin point is represented by its corresponding *a* and *b* value, which can be directly related to the center point of a circle in the image space. This process for circle detection is better displayed by the flowchart representation shown in Figure 3.13.

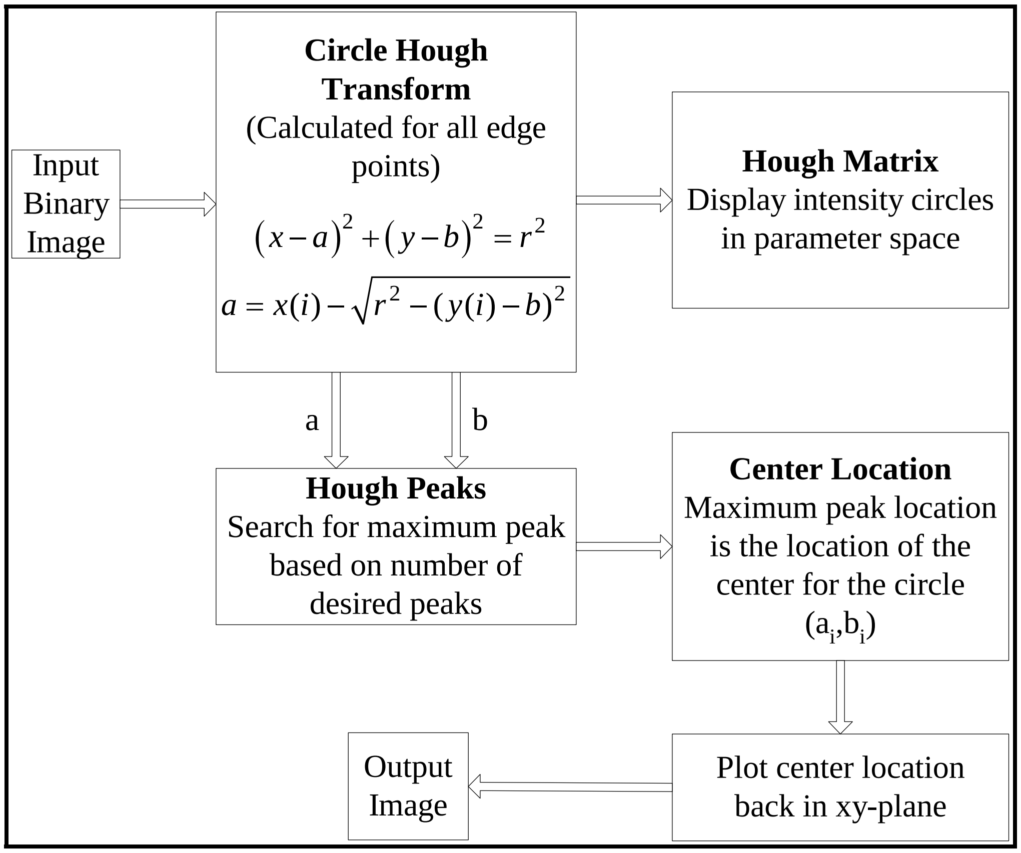

Input Binary Image
Circle Hough Transform
(Calculated for all edge points)
$(x-a)^2+(y-b)^2=r^2$
$a = x(i) - \sqrt{r^2 - (y(i)-b)^2}$
Hough Matrix
Display intensity circles in parameter space
a
b
Hough Peaks
Search for maximum peak based on number of desired peaks
Center Location
Maximum peak location is the location of the center for the circle
(a_i,b_i)
Plot center location back in xy-plane
Output Image

Figure 3.13: Hough Transform Operations for Circle Detection

3.4.2. Fast Finding and Fitting Algorithm

The FFF algorithm is based on an estimation theory which estimates the center location of a circle and its radius. The estimation theory is based off of the least square estimator. This method utilizes a linear regression technique to estimate parameters of a circle without leading to a nonlinear system of equations unlike most statistical methods. The input image to this algorithm is a binary image consisting of groups of detected edge pixels found by implementing low level image processing techniques. Each group of edge pixels is then inputted into the detection algorithm separately. Each edge pixel within its respective group is then used to estimate a possible center location and radius. To determine the location of a circle, the algorithm minimizes the sums of the residual, or error, between the predicted and estimated values. A detailed procedure is described below.

The representation of a circle is shown in Figure 3.14. The center of the circle is located at $P_o(x_o,y_o)$ and the circle has a radius, r, and n number of edge points. The edge points are represented as $P_i(x_i,y_i)$.

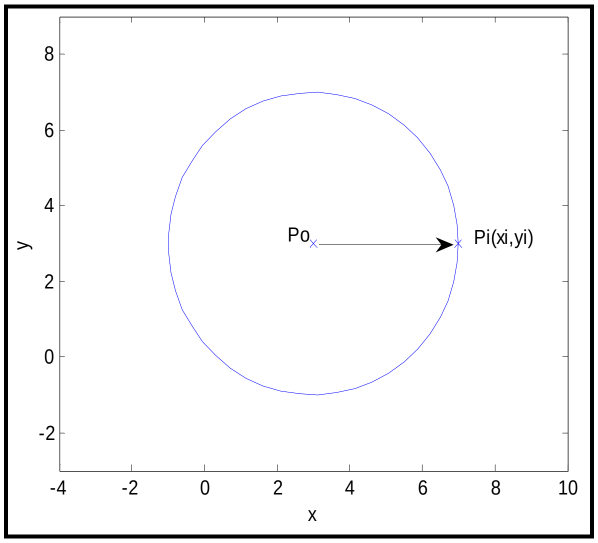


Figure 3.14: Representation of a Circle

The circle in Figure 3.14 has the following mathematical representation:

$$(x_i - x_o)^2 + (y_i - y_o)^2 = r^2 \tag{3.15}$$

or

$$(x_i - x_o)^2 + (y_i - y_o)^2 - r^2 = 0 \tag{3.16}$$

However, since the edge pixels are determined from pre-processing techniques, in most cases the resulting edges are not an exact representation as the one displayed in Figure 3.14. This causes the right hand side of Equation (3.16) to not be exactly equal to zero. Therefore, using the estimation theory an error function can be represented as follows:

$$E = \sum_{i=1}^{n} \left[(x_i - x_o)^2 + (y_i - y_o)^2 - r^2 \right]^2 \tag{3.17}$$

where *n* represents the total number of edge points. To represent the sum of the errors of all edge points inputted into the algorithm, the following representation is made:

$$E = \sum_{i=1}^{n} \left[x_i^2 - 2x_i x_o + y_i^2 - 2y_i y_o + z^2 \right]^2 \tag{3.18}$$

where *z* is defined below:

$$z = x_o^2 + y_o^2 - r^2 \tag{3.19}$$

The optimum estimation is determined from the group of edge pixels whose error function is a minimum. Thus, the partial derivative of the error function, *E*, with respect to x_o, y_o, and *z* have to satisfy the condition shown in Equation (3.20).

$$\left\{ \begin{array}{c} \dfrac{\delta E}{\delta x_o} = 2\sum_{i=1}^{n} \left(x_i^2 - 2x_i x_o - 2y_i y_o + z \right)\left(-2x_i \right) = 0 \\ \dfrac{\delta E}{\delta y_o} = 2\sum_{i=1}^{n} \left(x_i^2 - 2x_i x_o - 2y_i y_o + z \right)\left(-2y_i \right) = 0 \\ \dfrac{\delta E}{\delta z} = 2\sum_{i=1}^{n} \left(x_i^2 - 2x_i x_o - 2y_i y_o + z \right) = 0 \end{array} \right\} \tag{3.20}$$

The condition shown in Equation (3.20) can be rewritten as a matrix equation: *AX=D*. As *X* is a full rank matrix, the solution to *X* can be determined by calculating the inverse of the A matrix and rewriting the equation as follows: $X=A^{-1}D$. Using Equation (3.20) the following representations for the *A*, *D* and *X* matrices are shown below:

$$A = \begin{bmatrix} 2\sum_{i=1}^{n} x_i^2 & 2\sum_{i=1}^{n} x_i y_i & -\sum_{i=1}^{n} x_i \\ 2\sum_{i=1}^{n} x_i y_i & 2\sum_{i=1}^{n} y_i^2 & -\sum_{i=1}^{n} y_i \\ 2\sum_{i=1}^{n} x_i & 2\sum_{i=1}^{n} y_i & -n \end{bmatrix} \tag{3.21}$$

$$D = \begin{bmatrix} \sum_{i=1}^{n} \left(x_i^3 + x_i y_i^2 \right) \\ \sum_{i=1}^{n} \left(x_i y_i^2 + y_i^3 \right) \\ \sum_{i=1}^{n} \left(x_i^2 + y_i^2 \right) \end{bmatrix} \tag{3.22}$$

$$X = \begin{bmatrix} x_o \\ y_o \\ z \end{bmatrix} \tag{3.23}$$

Using this representation and solving for the *X* matrix, the resulting center location of the proposed circle is estimated at *(x_o,y_o)*. Furthermore, the radius can be estimated using Equation (3.24) below:

$$r = \sqrt{x_o^2 + y_o^2 - z} \tag{3.24}$$

Now that a center location and radius are estimated, the mean squared error[52] (MSE) of the estimator, or circle radius, is determined. This is determined by finding the difference in the estimator and the quantity to be estimated. For this to occur it is necessary to know the size of the circle to be detected prior to implementing the algorithm. If this is known, the following procedure can be completed to determine the MSE of the estimated circle:

$$Error = \sum_{i=1}^{n} \left[(x_i - x_0)^2 + (y_i - y_o)^2 - r^2 \right]^2 \tag{3.25}$$

$$\text{MSE}=\frac{Error}{n} \tag{3.26}$$

Following the detailed procedure explained above, the FFF algorithm consists of the following steps implemented in Matlab® software environment.

1. Scan the binary image from left to right and top to bottom. Create a matrix, *d*, containing labels for the connected groups of pixels in the image
2. For each labeled group of pixels find the *(x,y)* pixel location for each pixel within the group where *n* is the total number of edge points in the corresponding group.
3. Compute each component of the *A* matrix shown in Equation (3.21).
4. Compute each component of the D matrix shown in Equation (3.22).
5. Solve for X, the estimated value of the center location:

   $$X=A^{-1}D$$
6. Compute the estimated radius of the circle using Equation (3.24).
7. Sum the errors for all edge points in the corresponding pixel group.
8. Calculate the MSE for the corresponding pixel group.
9. Determine if the MSE and estimated radius values satisfy the threshold values.

10. Repeat steps 2 through 9 for each labeled group.
11. Determine which group of pixels that fall within the requirements has the correct radius.

The FFF algorithm is determined to be a feasible approach towards circular feature detection. Once this method is decided upon, further research is completed to test its feasibility for vision control problems. Computational efficiency of the algorithm is directly related to the amount of edge pixels in the image. As the number of labeled groups of pixels increase, the amount of computation time increases because the above calculations must be implemented for each edge pixel within each group. For circle detection purposes, the algorithm is very efficient when utilizing the MSE calculation. Circular objects are easily detected within the image. To decrease the computational time for this algorithm efficient pre-processing techniques must be utilized to eliminate the amount of noise in the image, but at the same time enhance the important features of the target.

3.5. Motor Control Operations

To resemble the characteristics of a UAV in flight, the experimental setup must allow the MV camera to be continuously and controllably positioned over the target once it is sensed be the camera. In this experimental setup, three stepper motors are used to control the position of the camera. This positioning system is a two degree of freedom (DoF) system; one motor moves the camera along the y-axis while two separate motors are used to move the camera along the x-axis. To properly control the MV camera in the experimental setup, the necessary motor inputs must be obtained. To acquire these inputs, it is necessary to convert the sensed data from the MV camera into usable information. The information obtained from the MV camera is the pixel locations of the detected target along the *x* and *y*-axis of the image. Once the spatial coordinate location of the target is known in the image frame, the distance along each axis is calculated between the center of the image, at pixel location (320,240), to the pixel location of the target within the image. This procedure results in two pixel distances, one distance along the x-axis and one along the y-axis of the image. Figure 3.15 represents the target position situation for calculating the pixel distances.

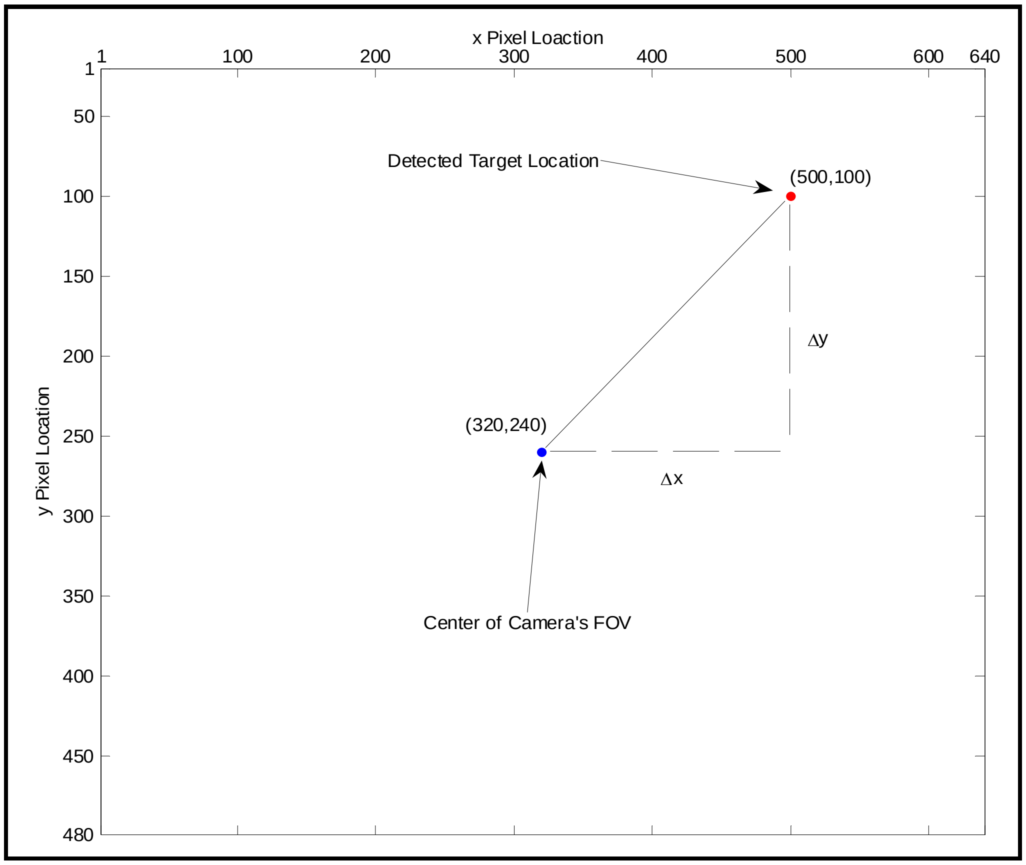


Figure 3.15: Target Position Situation for Calculating Pixel Differences

Depending upon the location of the target with respect to the center of the image, the values may be positive or negative. The sign convention of the image is shown in Figure 3.16.

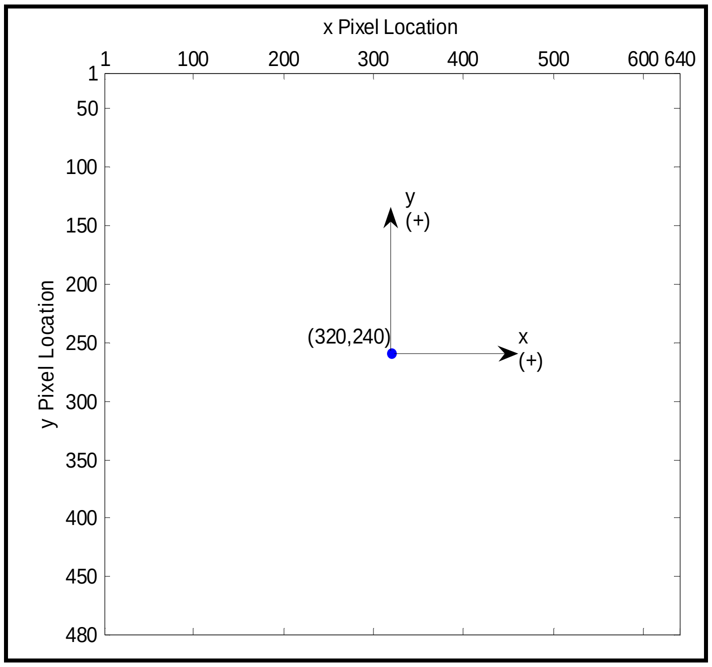


Figure 3.16: Image Sign Convention

The blue dot in the middle of the image represents the center of the MV camera's FOV. If the target is detected to the left or below the blue dot, the pixel distance will have a negative sign convention and vise versa. This orientation is used to determine the direction the MV camera must move.

To convert each pixel distance into a usable form of measurement, such as meters, it's necessary to form a relationship between the two forms of measurement. This relationship is obtained from the camera specifications. It's already known that the camera's FOV at its height above the "ground" level is 0.635 meters wide by 0.486 meters high. From this information a linear relationship can be defined which relates the pixel dimensions of the image to the dimensions, in meters, of the camera's FOV; the ratio of meters to pixels is approximately 1:1000. Using this relationship, the pixel distance sensed from the MV camera, represented by Δx and Δy in Figure 3.15, is converted into the distance, in meters, that the camera must travel to position the detected target in the center of the camera's FOV.

Once this distance is calculated the next step is to convert the linear distance into a rotational distance. This is necessary to determine the number of rotations each motor must turn to position the MV camera in the correct location. This procedure is completed using Equation (3.27).

$$n_{total} = \frac{\left(n_{motor} \big/ revolution \right)}{d_p} * D \qquad (3.27)$$

where n_{total} is the total number of steps for the stepper motor, d_p is the pitch diameter of the gear attached to each respective motor, D is the distance the MV camera must travel along the x and y-axis of the model environment, and n_{motor} is the number of steps per motor revolution. The stepper motor specifications are explained later in Section 4.1.4; the value for the gear pitch diameter is different for the motor system along each direction.

Once the motor inputs, in respect to number of revolutions, are determined the speed of each motor is then controlled. This is determined based on the distance the detected target is from the center of the camera's FOV. The target position situation for determining the motor speeds is shown in Figure 3.17.

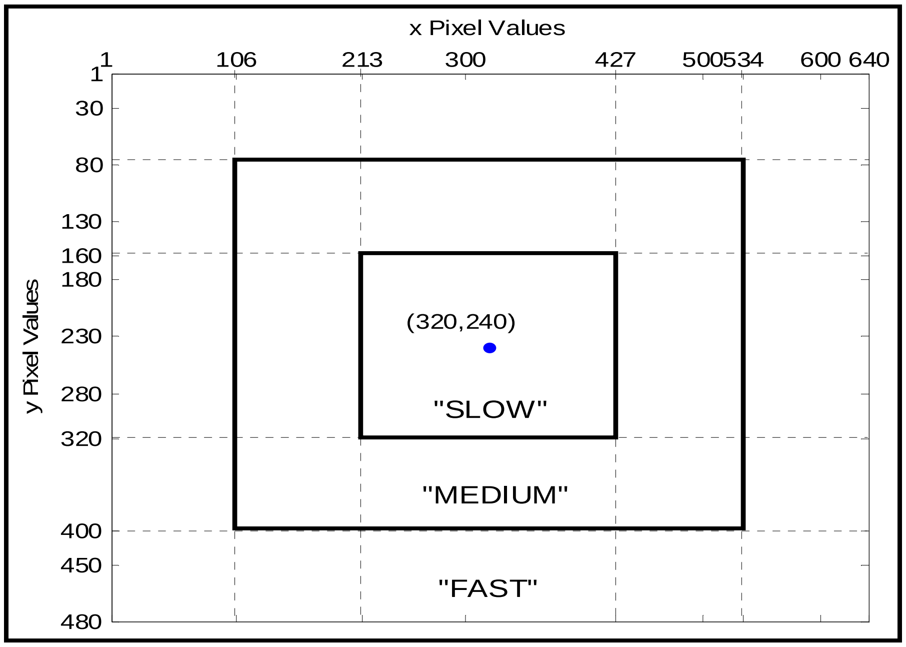


Figure 3.17: Image Situation for Speed Control Decisions

The velocity controller has three settings. The "fast" speed is experimentally determined through a number of simulations. This is determined as the speed which allows the fastest movement of the camera up until the point before the stepper motors begin to slip. The "medium" and "slow" speeds are then reduced by a certain percentage of the "fast" speed. The speed is selected based on the position of the target within the image frame. If the target is located within the center boundary then the motor speed is set to "slow." If the target is located outside of the center boundary and within the second largest boundary then the motor speed is set to “medium.” Finally, if the target is located outside the second largest boundary then the motor speed is set to “fast.” This velocity controller allocates a smooth motor performance while implementing the tracking algorithm. The estimated values for each speed setting are shown in Table 3.1.

Table 3.1: Estimated Speed Settings for Motor Control

Speed Setting	Estimated Speed (m/s)
“Slow”	0.105
“Medium	0.185
“Fast”	0.218

Chapter 4. Experimental Procedures

The experimental procedure for this research topic requires the design and application of software algorithms utilizing specific hardware in a laboratory setting. The ability to implement software in an actual experimental setting enhances the level of evaluation that can be obtained. Utilizing real images with real hardware allows an increased number of issues to be addressed rather than depending on prerecorded video or virtual images for testing the software. The experimental procedures for target detection and tracking are explained in three sections. The first section, Section 4.2, includes a detailed description of the experimental setup, hardware descriptions, and hardware setup. Sections 4.3 and 4.4 contain detailed descriptions of the software used for the target detection problem and target tracking problem, respectively.

4.1. Experimental Environment Setup Procedure

The experimental setup consists of the configuration and selection of specific hardware that is used to develop potential solutions to this research problem. The selection of the hardware determines what information can be obtained and affects the overall software design. This section provides a detailed description of the experimental environment and a physical and functional description of each piece of MV hardware. With this information, the functionality and importance of each piece of hardware will be made evident, and its direct relationship to the work effort.

4.1.1. Description of Machine Vision Research Computer

The MV research computer used in this research was purchased in separate pieces and assembled with the specific task of video and hardware experimentation. The performance of this computer proved to be adequate for MV research regarding aerial refueling and runway detection, lab experiments carried out by a former WVU graduate student. The selection process for each piece of the computer was based off of the fastest possible components for the budget available at that time. High speed computer components are critical relating to the amount of processing power MV applications usually require. Although computer technology has drastically improved since the original construction of this MV computer, this machine proved to be satisfactory for this MV experimentation.

Utilizing Intel technology, the main processor of the computer was selected as the Intel Pentium 4, 3.2 gigahertz (GHz) Prescott processor. The processor is placed on a Micro-Star International (MSI) brand 915G motherboard. The combination of the processor and motherboard allow the front side bus (FSB) to run at 800 megahertz (MHz). The FSB was very critical in the design of this computer. The FSB is used to carry the signals between the central processing unit (CPU) and the random access memory (RAM); therefore, the speed of the FSB is directly related to the speed of the computer[53]. The computer is also using 256 megabytes (MB) of Double Data Rate 2 (DDR2) Synchronous Dynamic Random Access Memory (SDRAM), which runs at a speed of 2700 MHz. The DDR2-SDRAM was considered to be an upgrade over the existing Double Data Rate (DDR) memory available at the time. This type of memory introduces features that allow the computer to obtain higher speeds and memory usage while using less power. The selection of this device is based off of the speed of the FSB, which should typically run at speeds twice as fast[54]. The hard drive speed is another critical performance specification that must be chosen. The hard drive chosen for this machine is a special edition Western Digital 80 gigabyte (GB) hard drive, which runs on a serial advanced technology attachment (SATA) bus. A SATA bus type hard drive is designed to replace parallel ATA technology and its development also supports overcoming the constraints that are increasing the difficulty of speed enhancements for the classic bus type hard drives[55]. Therefore, this bus type hard drive allows for a faster data transfer, reducing the time it takes for data to be received and stored. A photograph of the MV computer is shown in Figure 4.1, which includes a 17” flat screen monitor.

Figure 4.1: MV Computer

4.1.2. Description of the Lumenera USB Camera

In this research, the camera utilized for the laboratory experiment is a Lumenera brand, model Lu075 camera. The Lu075 camera is a high-speed USB 2.0 lightweight color camera. The camera is equipped with a high-quality 1/3" charge coupled discharge (CCD) sensor which makes it ideal for capturing objects in motion. At a maximum useable frame rate of 60 frames per second (fps) the MV camera can produce an obtainable resolution of 640 by 480 pixels. The camera is normally powered via the USB cable with a 5 Volt (V) supply. In cases where the USB cable does not supply power, a power adapter can also be used to power the camera, which must have a regulated 6 V direct current (DC) and a minimum current rating of 1000 milliamps (mA). The camera specifications are listed in Table 4.1.

Table 4.1: Lumenera Camera Specifications

Image Sensor	1/3" format, 5.8mm x 4.9 mm array
Effective Pixels	640 x 480
Frame Rate	7.5 fps, 15 fps, 60 fps
Sensitivity	High
Exposure	Auto/Manual
White Balance	Auto/Manual
Gamma Correction	1 or correction
Dimensions (W x H x D)	2.25 x 3.85 x 1.56 inches
Power Consumption	2.5 Watts
Lens Mount	C-Mount (CS-Mount optional)
Mass	300 grams
Power Supply	USB 2.0 bus supply or external 6 VDC, 500mA

This camera is designed to be used within a large variety of industrial applications. Many of its features make it very appealing for MV applications. Although most of the camera capabilities are not necessary for this research effort, they are determining factors for the utilization of this specific camera. These camera features include: gamma correction, white balance, shutter speed, size and weight, and the image sensor. Based on the image processing methods used in this effort, the gamma correction function is not necessary; therefore, it is set to the default value of 1. The exposure is also an adjustable parameter, which is set at its default value of 15.6 milliseconds (ms). This setting is adequate for the given conditions in the laboratory environment. The contrast and brightness are also parameters that can be adjusted manually through the Lumenera software. However, rather then using this approach it is much faster and easier to adjust the brightness through the camera lens. The contrast of the image is adjusted using the Matlab® software environment. The Lumenera camera is shown in Figure 4.2.

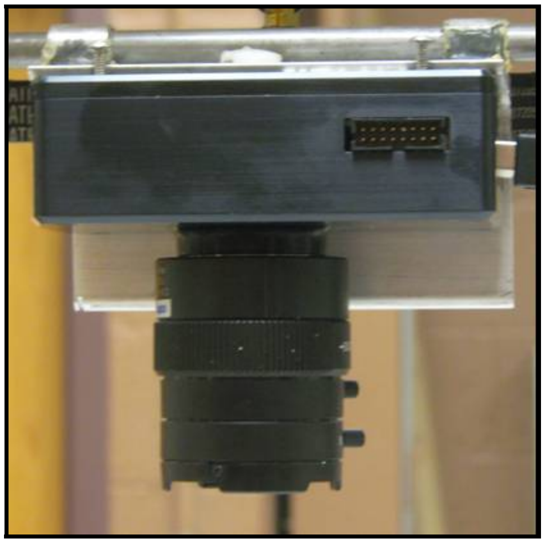


Figure 4.2: Lumenera Camera

The lens used with this camera is a Lumenera brand, model Lu935105CS lens. This lens is utilized because it provides a sufficient FOV for simulation studies. The lens has a focal length between the range of 3.5~10.5 mm with manual iris and focus. A detailed specification list for the Lumenera camera lens is shown in Table 4.2.

Table 4.2: Lumenera Camera Lens Specification List

Mount Type		CS-Mount	
Application		1/3” CCD Sensor	
Focal Length		3.5 ~ 10.5 mm	
Iris Control		Manual	
Focus Control		Manual	
Weight		64 grams	
FOV at 1 meter		~883 mm x ~617 mm	
Angle of View	Diagonal	1/3”	108.4 ~ 34.23^o
	Horizontal		81.6 ~ 27.2^o
	Vertical		59.2 ~ 20.4^o

4.1.3. Miniature Model Environment Setup

In this laboratory experiment, the target object chosen for the detection and tracking problem is a small locomotive train model. This type of target object is chosen because of its importance and utilization within society. Due to its various benefits, rail transportation is a major form of public transportation in many countries and is a major component of logistics. With the functionality of locomotive transportation systems, there are many reasons why a UAV would be given the task to detect and track a moving train. The specific model train used in this experiment is shown in Figure 4.3, which is classified as a HO scale model train. Figure 4.3b displays a top view of the train; to get a true perspective of the train size a dime is placed beside the train in the figure. For HO scale models 3.5 millimeters (mm) represents one real foot, which works out to a scale ratio of 1:87. The train is powered by a 0 to 18V DC variable power supply, which allows the speed of the train to be changed by regulating the supply voltage.

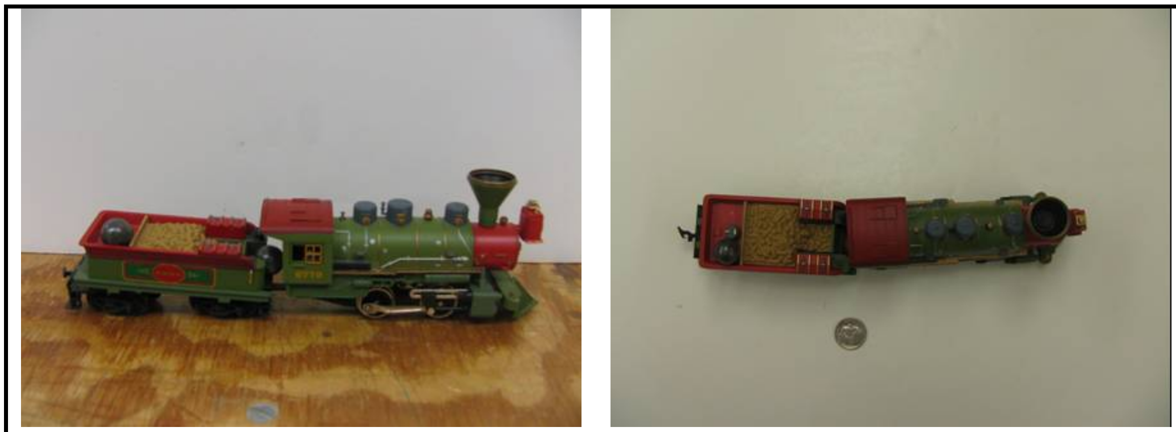

(a) Side View of Model Train (b) Top View of Model Train

Figure 4.3: HO Scale Model Train

The lab environment for this experiment consists of a train system and its surrounding scenery. The design of the landscape is an attempt to achieve as much detail and to be as close to reality as possible within a laboratory setting. Therefore, the scenery consists of small scale trees, bushes, and a mountain and a tunnel system to hide the train from the FOV. The train is set up to travel in a circular path within the small simulated environment. The diameter of the circular track is approximately 1 meter. Incorporating all of these features in the setup allows for a more realistic implementation of the MV software. This design allows issues to be addressed that could generate issues for the MV system, such as background interference. The model environment which includes the train system is shown in Figure 1.3.

The MV camera is mounted on a positioning apparatus located directly above the model environment. The apparatus can be manually adjusted to a desired height ranging from 0.90 to 1.5 meters. Once the apparatus is placed at a desired height it is necessary to design a mechanical positioning system that would allow the camera to change position in the plane at that height. To detect the moving train the camera must be able to move to any position in the plane over the "ground" level, which has a 914 mm (width) by 1066 mm (length) boundary. This system is a mockup of an aerial robot that is retrofitted with a MV sensor. Therefore, the final design is a two degree-of-freedom (DoF) gantry that is able to move in the xy plane. Figure 4.4 is a concept drawing of the test rig displaying the orientation of each axis and its different components (note: drawing is not to scale). In this figure, the y-axis represents the width of the "ground" level and the x-axis represents the length. Figure 1.2 represents the model environment after the construction stage.

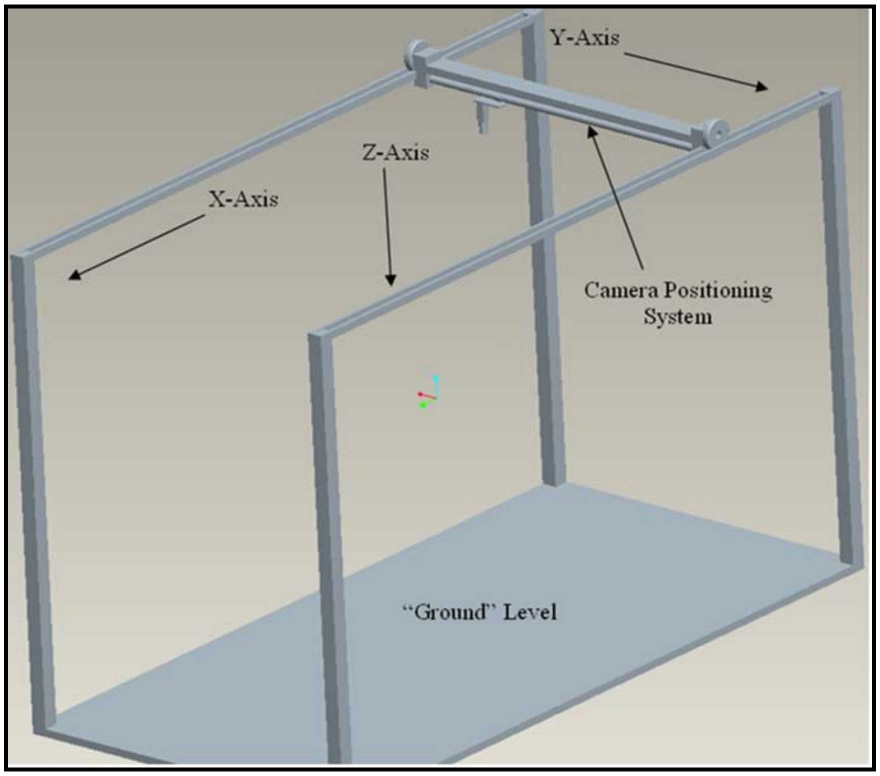


Figure 4.4: Concept Drawing of Model Environment

Three motors are used to position the camera at a position in the plane above "ground" level. These motors are incorporated into a mechanical system that converts the rotational motion of the motors into linear motion of the camera. The camera platform is shown in Figure 4.5.

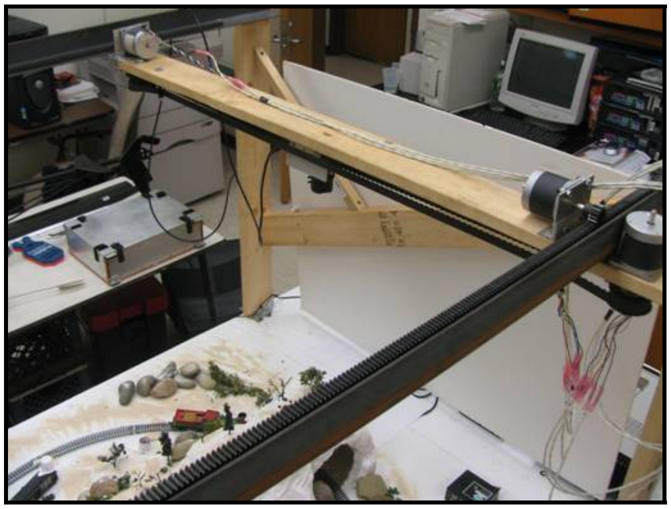


Figure 4.5: Camera Positioning Apparatus

Two motors are used to position the camera along the x-axis which is driven utilizing a rack and pinion design. It is necessary to use two rack and pinion systems with two motors to produce a sufficient amount of torque to overcome the total weight of the camera apparatus. The motor that positions the camera along the *y*-axis is driven by a belt system with the camera attached to the belt and a slider bar. The specifications for each system are shown in Table 4.3.

Table 4.3: Specifications for Gear Systems

Pinion Pitch Diameter	22.23 mm
Pinion Number of Teeth	14 Teeth
Belt Pulley Pitch Diameter	48.51 mm
Belt Pulley Number of Teeth	16 Teeth
Belt Pitch Length	1143 mm

The selection of each piece of hardware in Table 4.3 is based on the motor constraints and the minimum velocity of the train. Each motor system must produce a sufficient load inertia-to-rotor inertia ratio to ensure that the motors will be able to run at high speeds given the loads they must overcome. The linear traveling distance per rotational revolution is another important factor which is affected by the pitch diameter of the gear and pulley; a larger pitch diameter allows for a larger linear traveling distance per revolution. Out of the given options from a variety of manufacturers, the specifications shown in Table 4.3 are the best choices for this research application.

4.1.4. Stepper Motors and Stepper Motor Controller

To position the camera at a location within the model environment it is necessary to select and design a motion control system. Some available motion control systems are categorized as: stepper motors, DC motors, Brushless motors, linear stepper motors, and servo motors. For this application stepper motors are used for the motion control system. A key advantage for using a stepper motor is the ease of determining the position of the camera system using the number of input steps applied to each motor. Another advantage is the motor's position can be precisely controlled without the use of a feedback control system.

A typical stepper motor system consists of three elements, which are usually interfaced with a computer. These elements include: the indexer, driver, and stepper motor[56]. The indexer is, in most instances, a microprocessor capable of creating step pulses and directional signals for the driver. The driver then converts the pulses and signals from the indexer into power necessary to drive the motor windings. The stepper motor driver that was purchased and built for this application is the 4-axis HCNCPRO Driver Board Kit manufactured by Hobby CNC. This kit allows 3 to 4 axis unipolar stepper motor chopper control. The specifications for the driver board are shown below:

- 3 or 4 axis unipolar chopper control;
- Individual or simultaneous control of 2/4 Phase Stepper Motors;
- Accepts 5, 6, or 8 wire stepper motors only, 4 wire types are not usable;
- 42VDC maximum input voltage, 12VDC minimum input voltage. 24VDC minimum recommended voltage;
- 3.0 Amps Maximum per Phase, 500ma (.5A) minimum. Each axis is adjustable throughout this range;
- 1/1, 1/2, 1/4, 1/8, and 1/16 micro-stepping;
- Step and direction control;

- Idle current reduction to 50% when idle for 10 seconds. The time delay can be changed;
- Built-in protection circuit to help protect against blown chips on stepper motor short or open connections;
- 3.7" by 6.8" double sided with top silkscreen and thru plated holes and lead free solder PCB;
- Power on reset.

This particular board is chosen because of its low cost and ability to control up to four stepper motors simultaneously. Other key advantages are its ability to perform micro-stepping and that it has a built in protection circuit. Figure 4.6 displays the driver after the assembly stage.

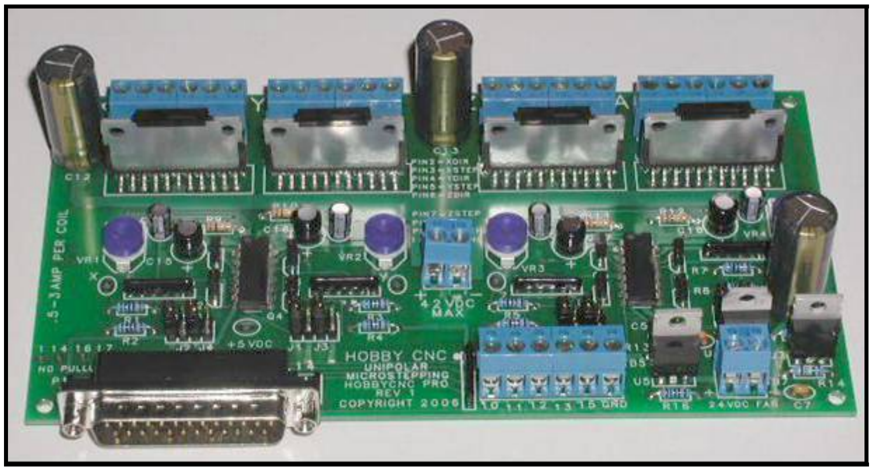


Figure 4.6: Stepper Motor Driver

The stepper motor driver is interfaced with the MV computer through the parallel port using the Matlab® software environment. Utilizing Matlab's® data acquisition toolbox[57], this software provides digital input/output (DIO) capabilities. The parallel port consists of eight data lines, four control lines, five status lines and eight ground lines. The data acquisition toolbox uses the parallel port to input and output digital values similar to typical DIO systems. To access the parallel port lines, the MV CPU comes equipped with a 25-pin female connector. The port identifications (IDs) and the associated pin numbers are shown in Table 4.4.

Table 4.4: Parallel Port IDs and Associated Pin Numbers

Port ID	Pins	Description
0	2-9	Eight I/O lines, with pin 9 being the most significant bit.
1	10-13, and 15	Five input lines used for status
2	1, 14, 16, and 17	Four I/O lines used for control

To provide inputs to the stepper motor driver, pins two through nine will be utilized. Table 4.5 explains the connections the driver board makes with the computer's parallel port.

Table 4.5: Parallel Port Pin Function

Pin	Function
2	X direction
3	X step
4	Y direction
5	Y step
6	Z direction
7	Z step
8	A direction
9	A step

Through this pin selection, four stepper motors can be controlled: motor "X," motor "Y," motor "Z," and motor "A." Within the stepper motor system, the Matlab® software environment can be classified as the indexer as it supplies the step pulses and directional signals to each respective motor.

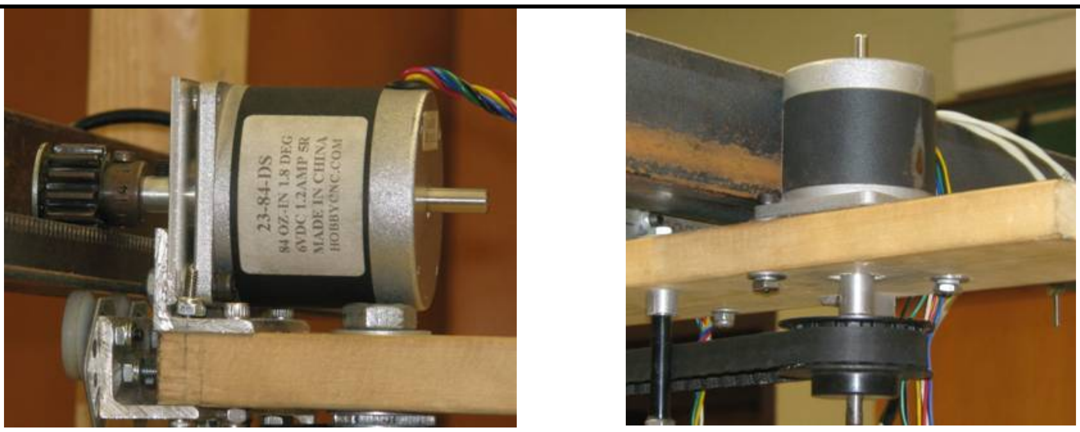
This stepper motor driver will accept 5, 6 and 8 wire stepper motors rated from 500 milliamps (mA) to 3 amps (A) per coil. To select the proper stepper motor for this application the first step is to determine the type of stepper motor that should be utilized. Either a unipolar or bipolar would be required. This characteristic defines the winding arrangement for the electromagnetic coils in a two phase stepper motor. In most cases a stepper motor driver is used to activate the drive transistors in the correct order for a unipolar stepper motor. The ease of this operation makes this type of motor very popular. When looking at the other motor type, the driving circuit for bipolar motors is more complicated because of their winding arrangement. For this reason a unipolar stepper motor is chosen for this research effort.

The next step is to select the proper unipolar stepper motor that could perform well under the conditions of the laboratory setup. The motor must supply enough torque to overcome any inertial loads while operating at speeds similar to the model train. Since the speed of the train is adjustable, it can be set relatively low, which eliminates the necessity of a high speed motor. The next determining factor is the maximum torque output of the motor. The stepper motors controlling the apparatus in the direction along the x-axis, shown in Figure 4.4, must produce enough torque to overcome 4.128 kilograms (kg), which is the weight of the camera positioning apparatus. Accounting for these considerations during the selection process, the stepper motors used in this experiment are Hobby CNC brand, model #23-84-DS stepper motors. The specifications for these motors are shown in Table 4.6.

Table 4.6: Stepper Motor Specifications

Maximum Holding Torque	59.32 N-cm
Rotor Inertia	2.47 g-m^2
Motor Type	2 phase
Shaft Type	Dual Shaft (3/8” diameter)
Step Angle	1.8^o
Current Per Phase	1.2A

The stepper motors are powered by an external variable power supply. The power supply is connected directly to the stepper motor driver, which supplies 15 V and typically draws 2.25 amps during the experimental procedures. Figure 4.7 shows a picture of each respective stepper motor system used to control the camera positioning apparatus.



(a) Stepper Motor for Rack and Pinion System (b) Stepper Motor for Belt System

Figure 4.7: Stepper Motor Systems

4.1.5. Luminance Meter

To assess the performance of the software under different lighting conditions it is necessary to measure the amount of luminance at each condition. The luminance measurement is then used to quantify the amount of visible light reflected from the "ground" surface. To obtain luminance measurements, a proposed method from research completed in Germany uses a digital still camera (DSC)[58]; this method is favored since the equipment is easily obtainable and quite accurate. When using this method to obtain luminance measurements it should be known that a digital camera is not an exact measuring instrument. However, for this application the luminance results of a digital camera will be sufficient since it is not necessary to have the absolute luminance value .

In this research effort, the camera utilized for the luminance meter is a Canon brand, model EOS 400D. It is determined that, to calculate the luminance, the exposure value (EV) should be obtained where EV is defined in ISO 2721[59]. Each EV value combines shutter time, *t*, and the relative aperture, *f*, combinations that result in the same exposure. This is shown by the representation in Equation (4.1):

$$2^{EV} = \frac{f^2}{t} \tag{4.1}$$

For a given ISO speed, the EV can then be used as a measure of luminance. In this research the luminance will be expressed in EV for ISO 100 speed. The relationship between EV at ISO 100 and luminance is shown in Equation (4.2):

$$L = 2^{EV-3} \tag{4.2}$$

In order to determine the EV measurements at different lighting conditions, the Canon DSC is used to obtain the measurements for the shutter time and relative aperture at each lighting condition for ISO 100 speed. Equation (4.1) is then used to calculate the EV, which is then used to calculate the luminance, following Equation (4.2).

4.2. Limitations of the Experimental Setup

Before the experimental procedures are discussed further, the limitations to the experimental setup should be identified. Since this experiment is performed in a controlled environment there are many issues that are not fully explored that are likely to occur in a natural, non-controlled situation. Limitations to the setup are noted as:

1. *Lighting conditions are controlled* – The lighting conditions are fully controlled. A possible solution to this is to perform the experiment in a location where the lighting conditions can be altered to account for many different conditions. In this research the lighting could only be changed to obtain two different lighting conditions. If an extensive lighting test could be conducted, this would produce a more realistic effect for the experiment.
2. *Distance of the camera from the train* – The distance of the camera to the train is limited. This constraint allowed the train to appear larger then it might appear in a normal situation. Because of this limitation the software might be able to detect the train easier due to the increased size of the train within the image frame. A possible solution to this limitation is to use a smaller camera or raise the height of the existing camera.
3. *Camera had limited motion* – The positioning system for the camera is only a two DoF system, which allows movement in the x and y axis. This limitation allowed for the design of a more simplistic positioning control system. A more realistic situation would allow the camera to pitch, roll,

or yaw about its axis. This would introduce other DoFs that would better simulate a UAV in flight.

These limitations had a direct effect on the performance of the MV algorithms used to solve this problem. It is important to address these limitations before the main algorithms are discussed. A solution for each limitation should be reached before these algorithms can be implemented in a real life scenario.

4.3. Software Used for Target Detection and Tracking

The software used to perform target detection consists of two methods, the Simulink® based method using HT and the Simulink® based method using the FFF algorithm. To apply both of these methods for target detection, many tasks must be performed. These tasks include the data acquisition stage, the image filtering stage, the main feature detection algorithm, and an error identification scheme. Finally, the final stage consists of the tracking software. This stage is incorporated with the target detection software which allows position control of the camera. Each stage is represented by their respective colored block in the main Simulink® scheme shown on the following page in Figure 4.8. Each block will be fully explained in the proceeding sections.

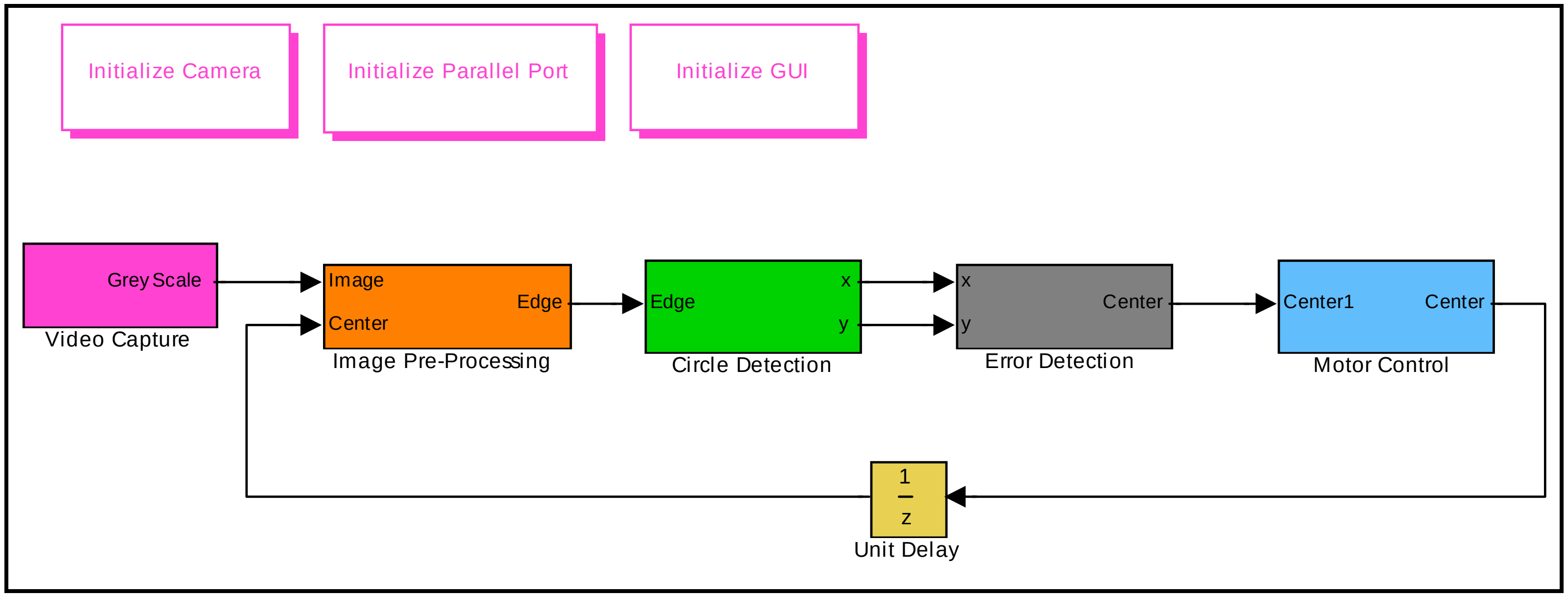


Figure 4.8: Main Simulink® Scheme for Target Detection and Tracking

The implementation of this software requires a detailed explanation for each of the main concepts used to solve this research problem. The image acquisition block shown in Figure 4.8 is represented by the magenta block. This block will be explained in Section 4.4.1. The image pre-processing and edge detection routines are represented in Figure 4.8 as the orange block. This block will be discussed in Section 4.4.2. The main target detection algorithm is represented in Figure 4.8 as the green block. This block is interchangeable between the HT algorithm and the FFF algorithm. Each of these subsystems will be discussed in Section 4.4.3. The error correction block is a subsystem that contains a routine that is capable of determining the necessary inputs when the camera is unable to detect the target. This subsystem is represented by the gray block in Figure 4.8, which will be explained in Section 4.4.4. Finally the position control routine, represented by the red block in Figure 4.8, supplies the necessary inputs to the stepper motors, which is based on the location of the train in the captured image frame. This block will be explained in Section 4.4.5.

4.3.1. Image Acquisition Stage

The purpose of this Simulink® scheme is to perform target detection and tracking for a near real-time application. Utilizing a small model environment to perform this research effort eliminated the necessity of using a pre-recorded video to test the designed software. As a result of this, the image acquisition stage in the Simulink® scheme is used to capture frames from a live video feed. Evaluating images from a live video produces slightly different results during each test. This evaluation method better simulates the flight and video characteristics of a flying UAV allowing for a more realistic evaluation of this scheme.

The Image Capture subsystem, shown in Figure 4.9, contains all of the necessary procedures to capture a frame from the input video stream and convert the image frame from RGB to intensity.

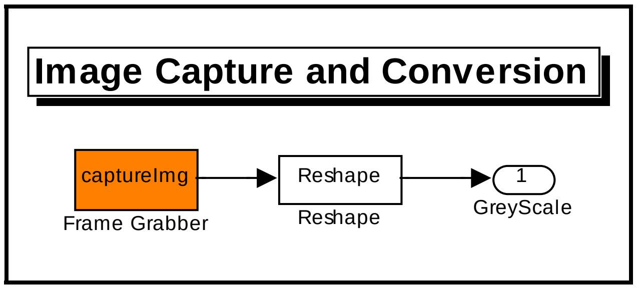


Figure 4.9: Image Capture Subsystem

The blocks used in this subsystem are a level-2 S-function and a Reshape block, both of which are included in the standard blocks within the Standard Simulink® Blockset[60]. The Capture Image S-function block is shown in Figure 4.9 as the orange block. Using an S-function provides the ability to build a general purpose block that can be used many times in a simulation which allows the variation of specific parameters in each simulation. The parameters of this S-function include: sampling time, dimensions of the captured image, and the video source. The video input source must be initialized before the scheme can be implemented. To accomplish this, Matlab's® image acquisition toolbox[61] is used to connect and configure the MV camera. Once the video source is initialized the S-function is used to perform color space conversion on a captured image frame. The reshape block is the used to obtain the correct dimensions of the image data matrix after the color conversion process. Therefore, the captured image is converted from a 640x480x3 RGB image to a 640x480x1 grayscale intensity image of class double. This conversion operation is explained in detail in Section 3.2.4. A typical result of the Image Capture subsystem is shown in Figure 4.10.

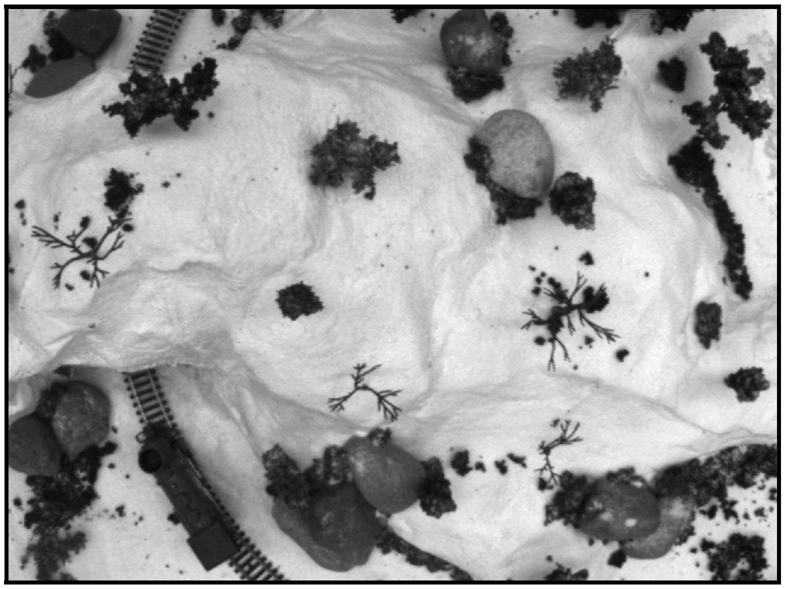


Figure 4.10: Arbitrary Frame Captured from the Image Capture Subsystem

4.3.2. Image Preparation, Edge Detection, and Filtering

The Image Pre-Processing subsystem is shown below in Figure 4.11. This subsystem contains all the functions necessary to convert the 640x480x1 intensity image to a 640x480x1 binary image, which is defined by a matrix of 0's and 1's for pixel values.

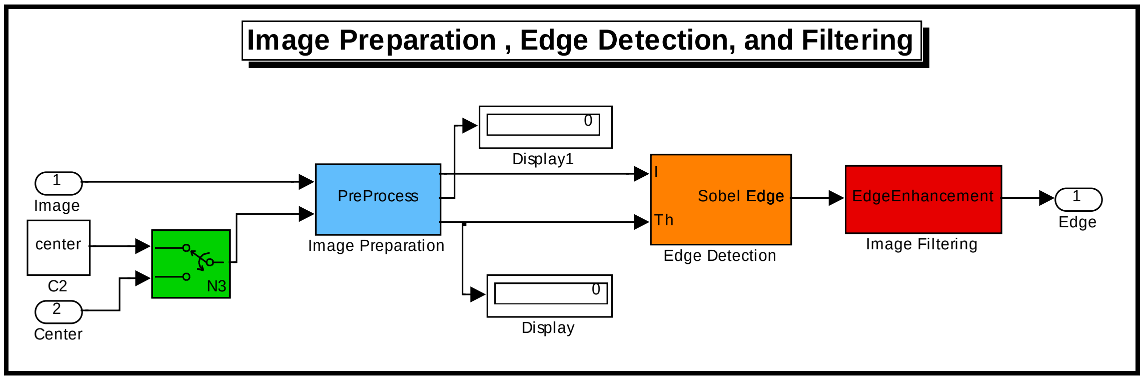


Figure 4.11: Image Pre-Processing Subsystem

This subsystem performs image preparation, edge detection, and filters out most of the noise left in the binary image. The input to this subsystem is the intensity image captured from the image acquisition block and the coordinates of the train's previous location obtained through a feedback routine. The output of the subsystem is then a filtered, binary image. The blocks used in this subsystem are standard blocks within the Standard Simulink® Blockset. The blue and red blocks, shown in Figure 4.11, represent the image preparation and image filtering routines, respectively. These blocks are both level-2 S-functions designed using the Matlab® software environment. The orange block in Figure 4.11 is a standard Simulink® block used to perform Sobel edge detection on the image outputted from the Pre-Process S-function.

The list of parameters for the Pre-Process S-function include: image dimensions, sampling time, threshold value for the edge detection routine, and the pixel intensity transformation inputs. This S-function performs the intensity transformation method known as contrast-stretching, which is previously explained in Section 3.2.3. This function also performs an image cropping routine on the input image to help decrease computation time during the simulation. The cropping routine is performed by taking the full resolution image, which in this case is 640x480, and confines the image along each axis based on the previous location of the target. The result is the same input image, but it's cropped along its sides resulting in a 480x360 image of pixel data. This allows approximately 44% of the image to be ignored while still allowing the target to be detected. The portion of the image that is confined is based on the previous location of the target within one of the four equally divided quadrants. If the train is detected in one of these quadrants then the cropping algorithm crops the input image around the specified quadrant with an added buffer zone. The portion of the image outside the cropped image is set to the pixel intensity value of 1, or white. The buffer zone ensures that the train will be detected when it is near the edge of the quadrants, this area can be adjusted depending on the execution speed of the software. If the system has a relatively fast processing speed then the buffer zone can be reduced as long as the train isn't fast enough to exit the buffer zone before the next frame is grabbed. Figure 4.12 displays the image broken into its four equal quadrants. Figure 4.13 displays the result of the cropping function when the train is detected in the forth quadrant. Two other important points are considered when the train is not detected in the previous frame and when the train is detected in the exact center of the image at pixel location (320,240). When these situations occur then the cropping routine is skipped and the entire image is inputted into the Simulink® scheme.

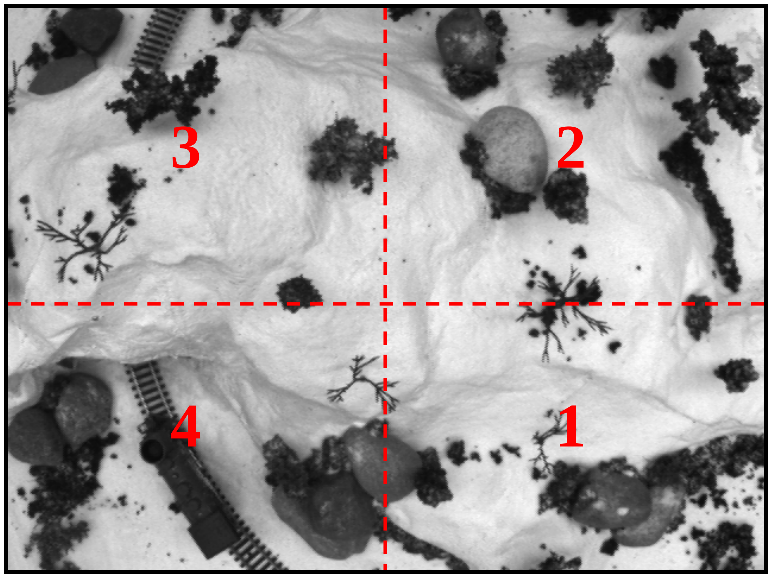


Figure 4.12: Arbitrary Image Frame Broken Into Four Quadrants

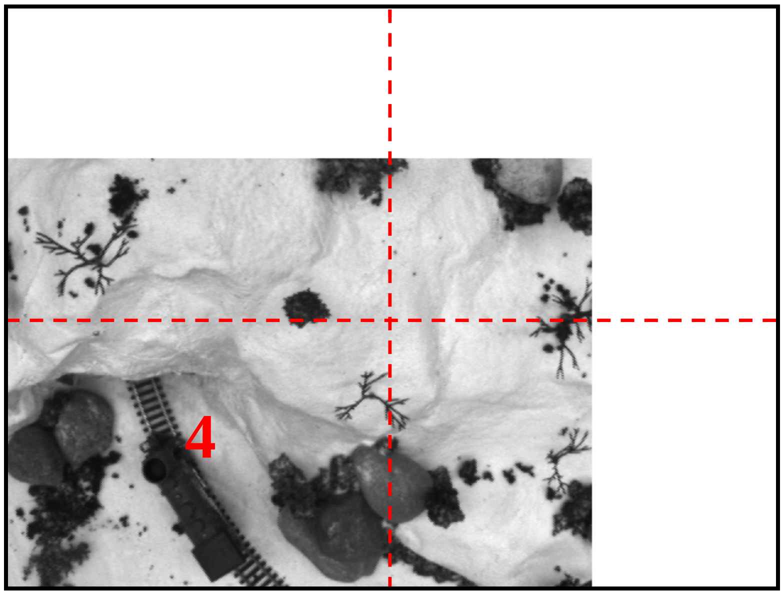


Figure 4.13: Result of Cropping Routine on Image Frame

In the first three steps, no cropping alterations are performed on the input image. This three step buffer allows the circle detection routine to become established and to find a good set of target coordinates. This three step buffer is chosen based off of previous simulation results. It is found that less than three time steps could result in false detections at the beginning of the simulation. If more than three time steps are used no notable benefits are obtained. Therefore, it is determined there is no reason to postpone the cropping routine for more than three time steps. This principle is controlled by the green block shown in Figure 4.11. This block is called an 'N-Sample Switch' and its job is to change state after the specified number of samples has occurred. Once the desired number of samples occurs the switch will flip and the last detected train image coordinates will be input into the Pre-Process block.

After the cropping routine is completed, the contrast-stretching transformation method is performed on the new image. This method is previously discussed in Section 3.3.2. This operation is performed on the image to produce an overexposed image, which is necessary to illuminate the chimney stack on the train for the edge detection process. The resulting image is shown in Figure 4.14.

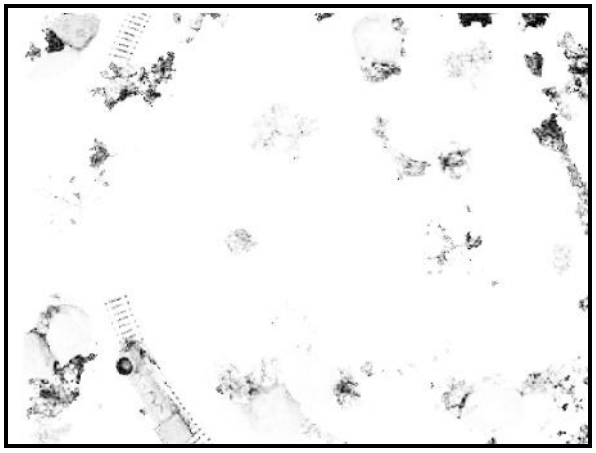


Figure 4.14: Result of Contrast-Stretching Transformation

Once this is complete the new image enters the Sobel Edge Detection block. This routine is previously discussed in Section 3.3.5. The output binary edge image from the Sobel edge detection routine is then sent to the Edge Enhancement block. The list of parameters for this S-function include: image dimensions, sampling time, and the filtering function input. This S-function utilizes a function found within Matlab's® image processing toolbox[62] known as the *bwareaopen* function. This function removes, from a binary image, all connected pixel components that have less than a specified number of pixels, producing a filtered binary image. This filtering function reduces the amount of noise within the image. The pixel threshold value is based off of previous simulation results. The highest possible threshold value is used to eliminate unimportant pixels, but ultimately leaves the pixels defining the train unaltered. The output of this subsystem is then a filtered binary edge image.

4.3.3. Circle Detection

The subsystems shown in Figure 4.15 and Figure 4.16 both contain only one high-level image processing operation. In Figure 4.15 this block is represented by the Circle Detection level-2 S-function block which is displayed as the purple block. In Figure 4.16 the Hough Transform level-2 S-function block, displayed as the orange block, represents the high-level image processing operation. These blocks are standard blocks included in the Standard Simulink® Blockset and are designed using the Matlab® software environment. The HT and FFF algorithms and their respective applications are previously discussed in Sections 3.4.1 and 3.4.2, respectively. In this section the inputs and outputs for these applications, within this Simulink® scheme, are discussed. To switch between the two different methods for the target detection routines, the respective subsystem must be represented as the green block in Figure 4.8.

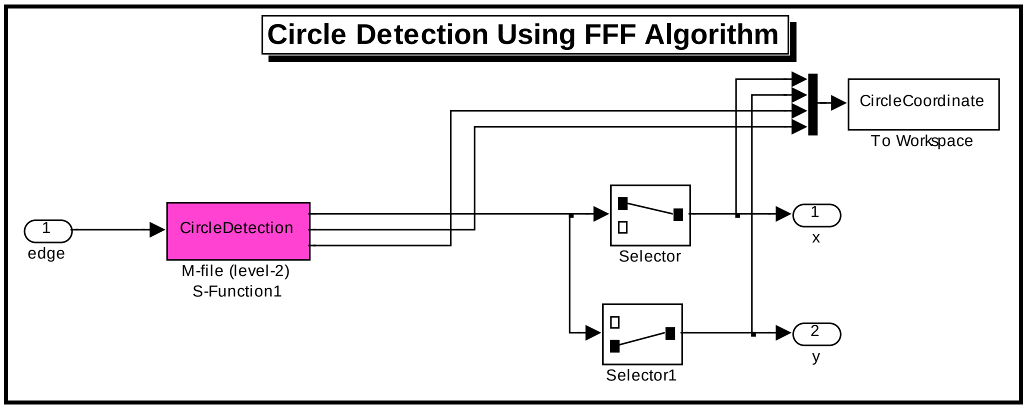


Figure 4.15: FFF Circle Detection Subsystem

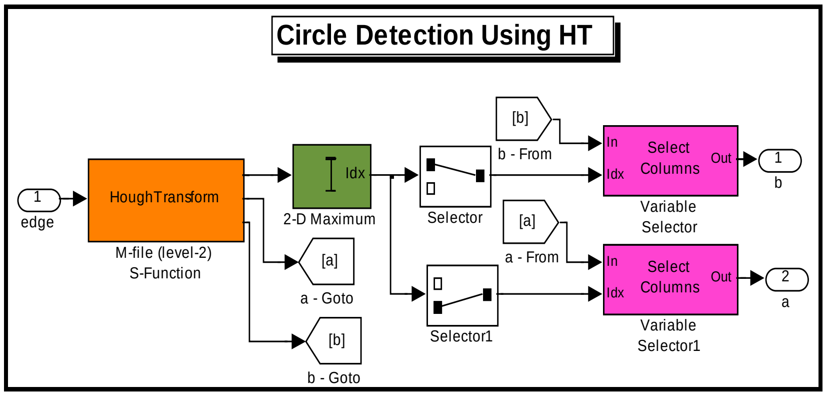


Figure 4.16: Hough Transform Circle Detection Subsystem

4.3.3.1. HT Operations

The list of parameters for the HT level-2 S-function include: image dimensions, x-axis and y-axis resolution, and the sampling time. The HT subsystem uses the binary edge image obtained from the Pre-Processing subsystem, discussed in Section 4.3.2, as the input. In particular, the HT S-function takes this input directly. This block is found in Figure 4.16 as the orange colored block. The outputs of the HT S-function are the *a* and *b* values that correspond to the location of the maximum peak value in the accumulator matrix. These values represent the center coordinates of the detected circle. Figure 4.18 and Figure 4.19 illustrate the typical plots obtained when performing a simulation using the HT method for circle detection when Figure 4.17 is represented as the input image after the edge filtering process.

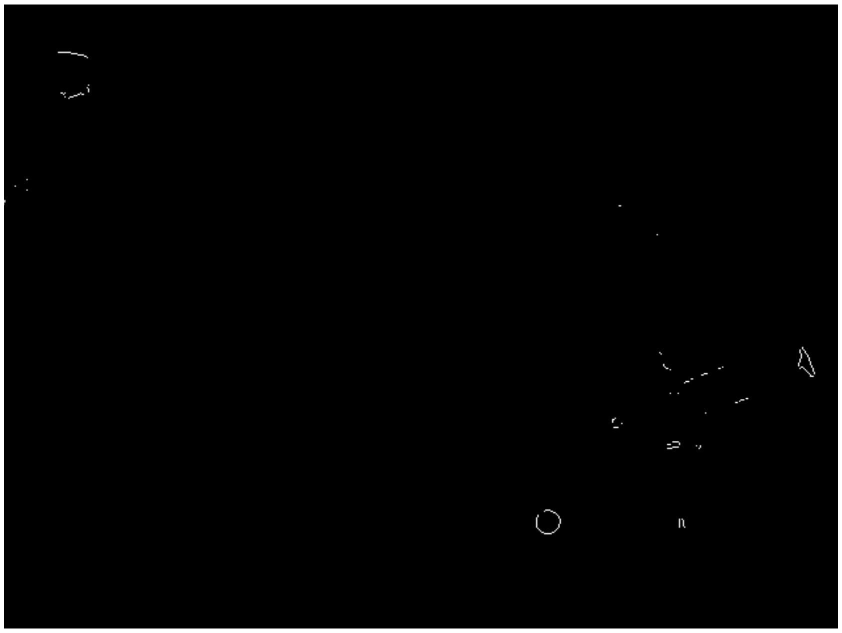


Figure 4.17: Input Binary Image to HT Circle Detection Algorithm

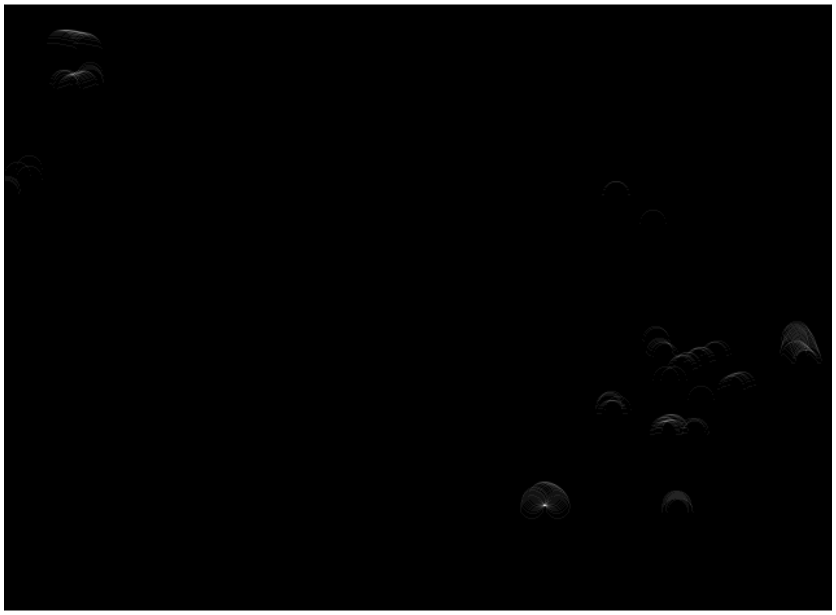


Figure 4.18: Output Hough Transform Matrix

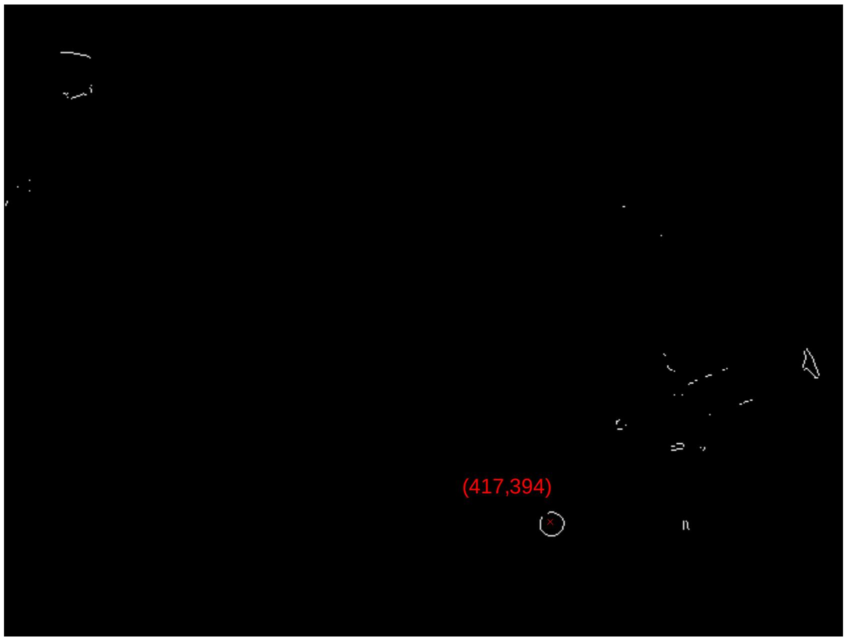


Figure 4.19: Output Binary Image from Hough Transform Algorithm

4.3.3.2. FFF Algorithm Operations

The list of parameters for the FFF S-function include: the maximum MSE value, the circle radius, image dimensions, bias, and sampling time. Similar to the HT subsystem, the FFF subsystem uses the black and white edge image obtained from the Pre-Processing subsystem where the FFF S-function takes the input directly. This block is displayed in Figure 4.15 as the magenta colored block. The outputs of the FFF S-function are the *x* and *y* coordinates of the center location for the detected circle. These values correspond to the circle that best satisfies the user defined MSE and circle radius criteria. A typical input image to the FFF algorithm is shown in Figure 4.20. Figure 4.21 illustrates the result obtained when performing a simulation using the FFF method.

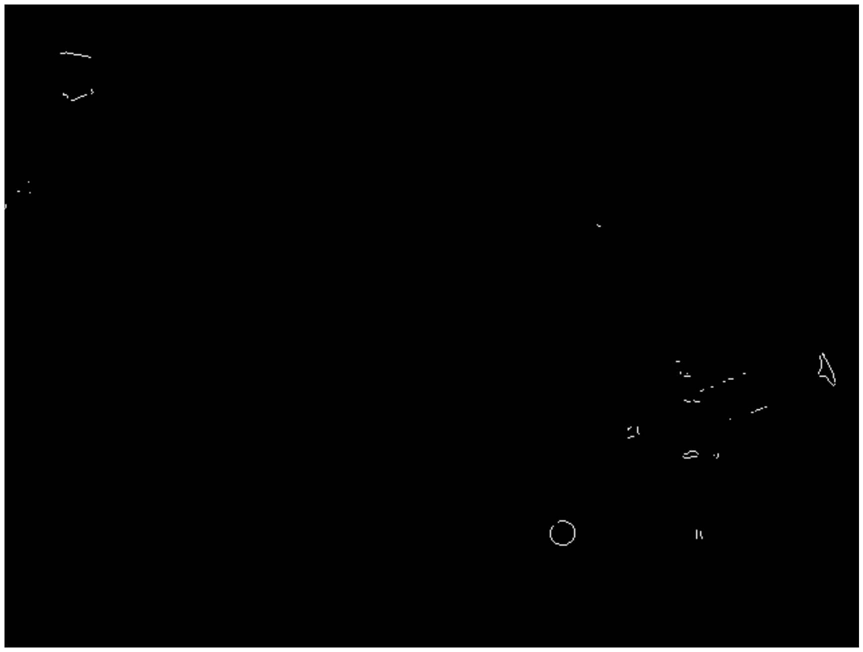


Figure 4.20: Input Binary Image to FFF Algorithm

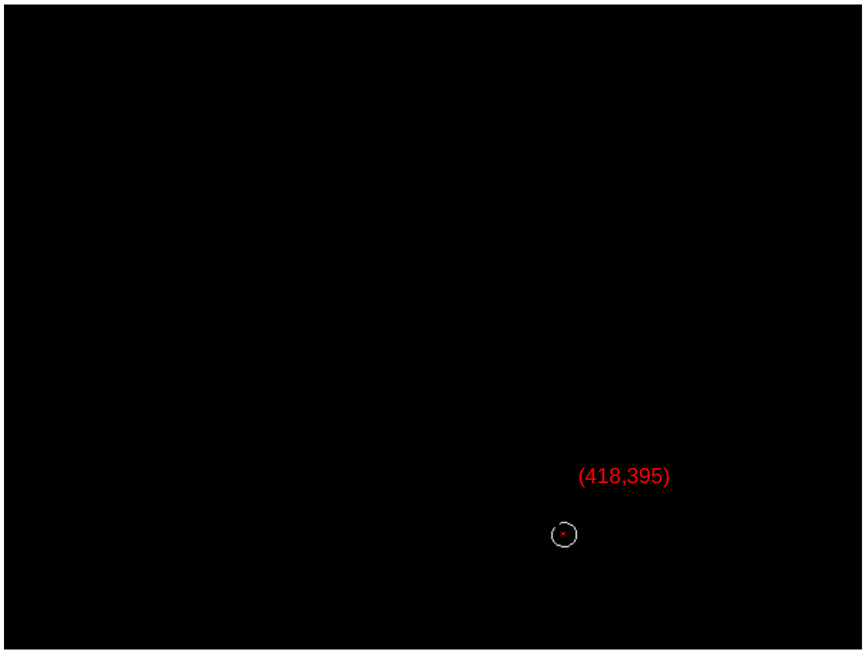


Figure 4.21: Output Binary Image of FFF Algorithm

4.3.4. Error Detection Operations

The Error Detection subsystem is shown previously in Figure 4.8 as the gray block. All of the blocks in this subsystem are standard blocks found in the Standard Simulink® Blockset. This subsystem takes the output of the Circle Detection subsystem and validates that the train is detected. In the case of zero detection, the spatial coordinates, (*x*,*y*), will be equal to (0,0). If this is the case, then the subsystem discards the current information and uses the values from the last time step for either the new x-coordinate or the y-coordinate values. This is performed by using two time step delays and a threshold value to compare both the *x* and *y* coordinates. The coordinate that is chosen as the new input is determined by the Flag subsystem.. The Error Detection subsystem is shown in Figure 4.22 on the following page.

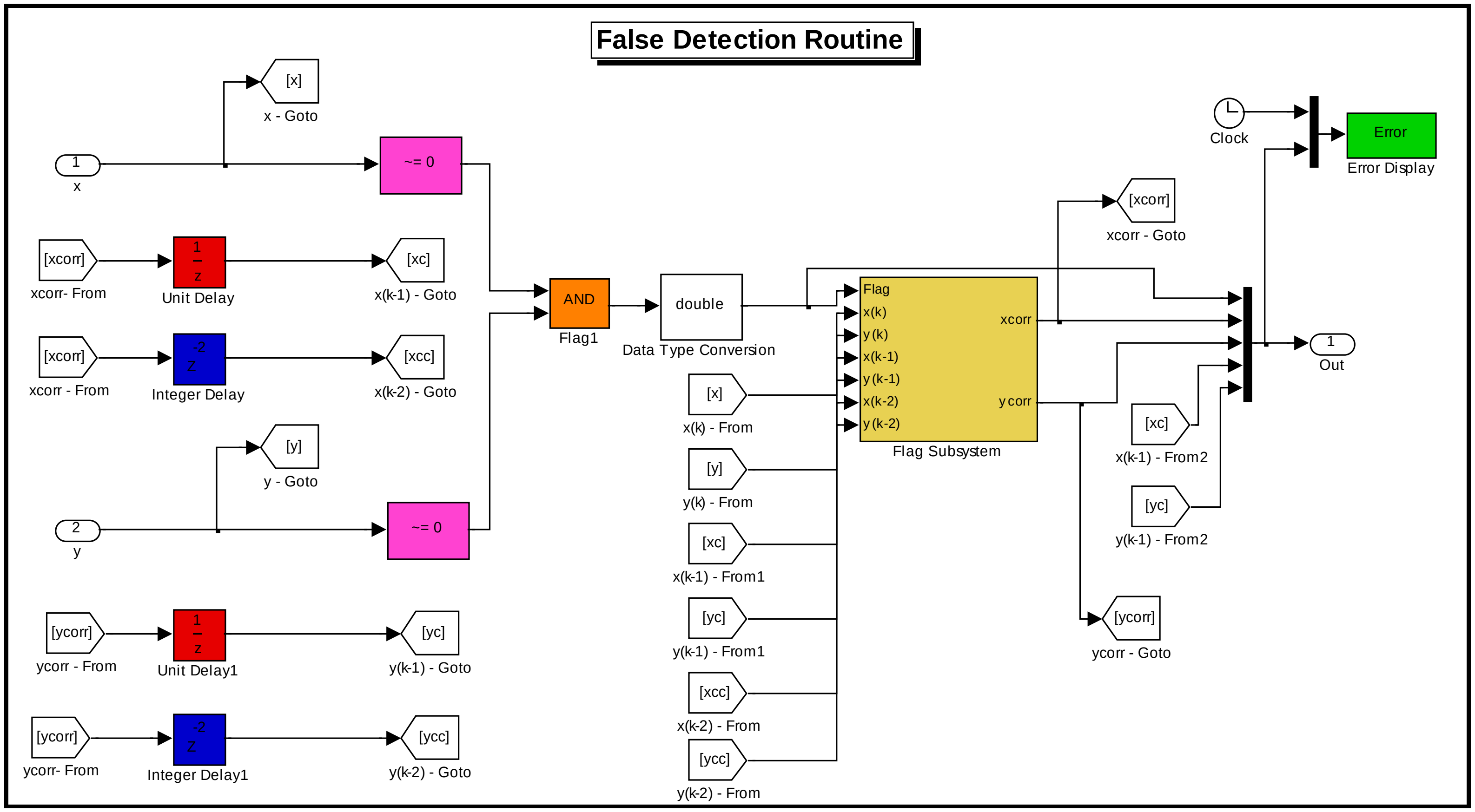


Figure 4.22: Error Detection Subsystem

In the Error Detection subsystem, each coordinate value enters the magenta blocks found in Figure 4.22. These blocks essentially compare the current outputs of the circle detection algorithm with a threshold value of 0. If both coordinates are greater then 0, representing a positive detection, then the block will output a value of 1, if not 0 is the output. The outputs of these blocks allow the user to determine which time steps the detection algorithm failed to detect the target. The flags are then fed into a ‘AND’ operator displayed by the orange block in Figure 4.22. The ‘AND’ block will output a true signal, or value of 1, only if both input signals are greater then the threshold, 0. The ‘AND’ block allows the system to check both coordinate values simultaneously.

The process at this point shows a flag value indicating if the train has been detected in the image frame. If there is an error in the detection software then there should be some attempt to guess the location of the train in the next time step. This is completed using the Flag subsystem. This subsystem determines what the outputs of the Error subsystem should be if the target detection software fails. The Flag subsystem is shown in Figure 4.23.

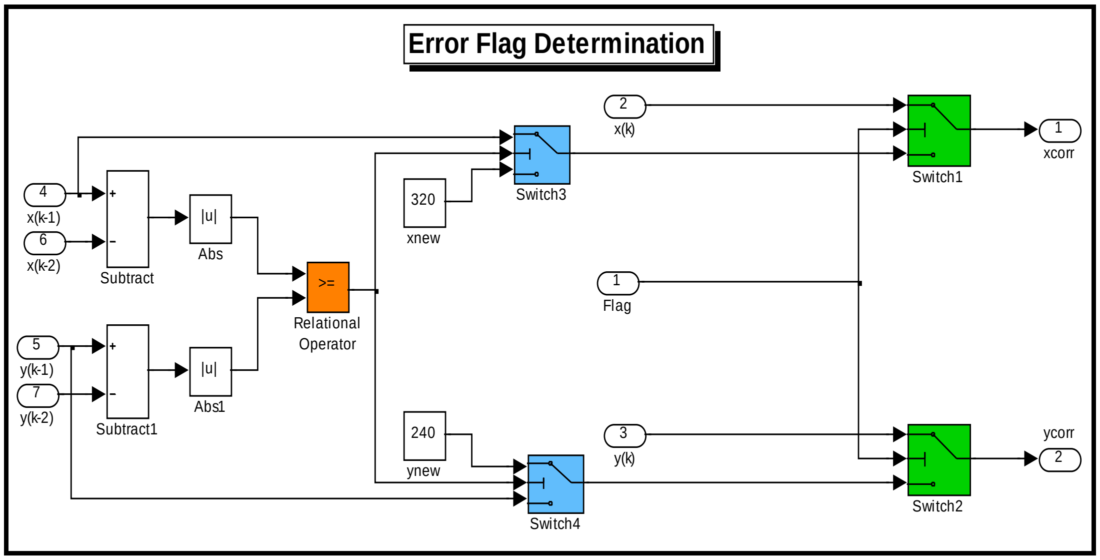


Figure 4.23: Flag Subsystem

The error flag is determined using both the *x* and *y* coordinate values together, meaning that if one value is wrong, then both values will be “corrected.” This is shown by the set of green switches, in Figure 4.23, which identifies which values exit the subsystem, the current values or the corrected values. The corrected values are determined by examining the difference between the spatial coordinate values of the last two time steps in the simulation. The absolute difference is calculated between the coordinates at each time step. The differences are then compared by the relational operator block, which is represented by the orange block in Figure 4.23. This block will output a true signal, or 1, if the first input is greater then or equal to the second input. If not, then the output is a false signal, or 0. There is now a flag indicating which direction in the image, along the x-axis or along the y-axis, has the greatest pixel difference between the last two time steps. If the pixel difference along the *x*-axis is larger, the *x*-coordinate from the previous time step is used as the new input and the *y*-coordinate is set to 320. If the pixel difference along the *y*-axis is larger, then the *y*-coordinate from the previous time step is used as the new input and the *x*-coordinate is set to 240. Based off the error flag, only if the detection software failed, the output of the subsystem will either be ($x_{(k-1)}$,320) or (240,$y_{(k-1)}$). The "corrected" inputs are represented in a way to move the camera in only one direction, the direction of the new input. This process is shown by the set of blue switches shown in Figure 4.23.

The overall goal of this subsystem is an attempt to estimate the next location of the train if the detection subsystem fails to detect it in the current time step. Only one direction is estimated due to the fact that it is highly unlikely that the next position of the train will be located using both of the coordinates from the previous time step. This is determined by a number of simulation tests using different error correction schemes. Another important feature of this subsystem is that it only attempts to estimate the new position of the train once. In other words, if the Circle Detection subsystem fails to locate the target in two sequential time steps, then the "UAV," or camera system, will be given the necessary input to "loiter," or hold its position until the target enters the camera's FOV once again. This feature is important to eliminate the "UAV" or camera system from moving randomly within the model environment. The flag values are also stored for every time step to help determine where the error occurred in the detection software.

4.3.5. Motor Control

The Motor Control subsystem is shown previously in Figure 4.8 as the light blue colored block. It should be noted that all of the blocks in this subsystem are standard blocks found within the Standard Simulink® Blockset. This block takes the *x* and *y* center coordinates of the circle, detected from the Circle Detection subsystem, and calculates the number of input steps for each motor to place the target in the center of the cameras FOV. The coordinates are based off the coordinate system shown in Figure 3.1. This subsystem also records the position and velocity of the camera and train at each time sample. This is performed by using a feedback loop with a one time step delay and an 'N-Sample Switch.' Two feedback loops are required to obtain the positions for both the camera and the train. The initial position of the camera is a constant specified by the user in the graphical user interface (GUI), which will be discussed in the next section. Within the GUI the user has the option to choose from five locations within the model environment for the initial condition. These locations are within the boundary of the model environment. It should be noted that the position and velocity of the train is based off the data obtained through the MV camera, therefore; if a false detection occurs then the measured data of the train will be incorrect. The Motor Control subsystem is shown in Figure 4.24.

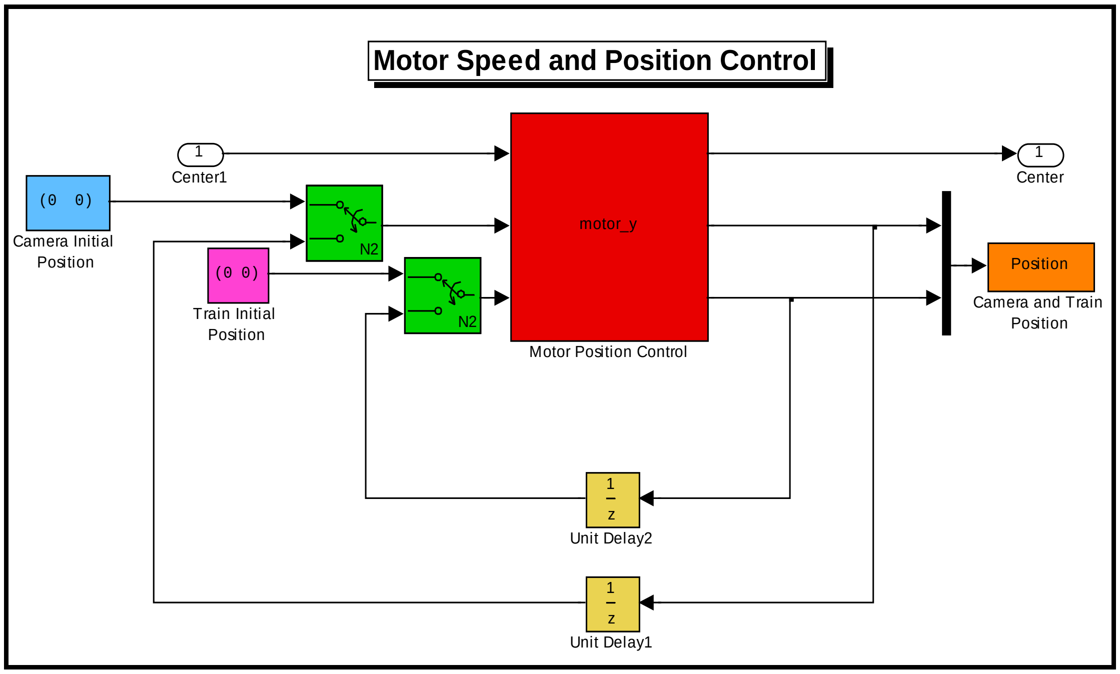


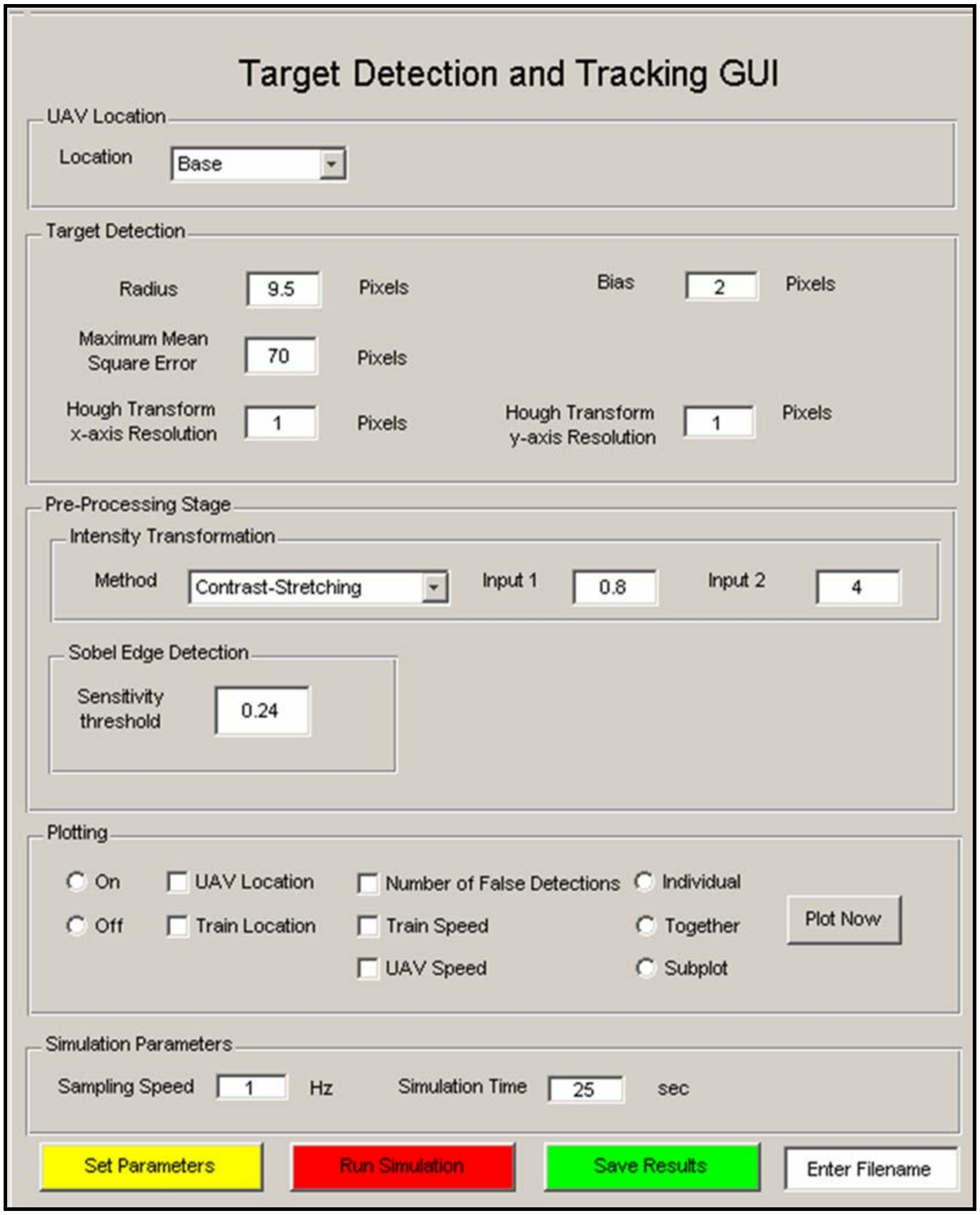
Figure 4.24: Stepper Motor Control Subsystem

The red block shown in Figure 4.24 is a level-2 S-function designed using Matlab® software environment. The inputs to this block are the initial positions of the camera and train and the center spatial coordinates of the detected circle within the image frame. This block is the heart of the Motor Control subsystem, that is, this level-2 S-function performs all of the necessary conversions and calculations necessary to control the stepper motor position and speed. The overall goal of this system is to minimize the distance between the location of the target and the center of the camera's FOV.

In order to begin the detailed discussion of the Motor Position subsystem the list of parameters for the S-function must be stated. These parameters include the sampling time and the initial state of the parallel port pins, two through seven, which are represented by the vector: [0 0 0 0 0 0]. The parallel port representation is explained earlier in Section 4.1.4. The first operation that is performed is the calculation of the pixel distance between the circle location and the center of the camera's FOV. The pixel distance is calculated along the *x* and *y* axis of the image. Each distance is then converted into the number of steps and direction that each stepper motor must move to minimize each pixel distance. The conversion process for this process is explained previously in Section 3.5. Once the number of steps and direction for each motor is determined the speed of the motor is determined. This parameter is determined by the process explained previously in Section 3.5. The speed of each motor is controlled by a delay factor set within the S-function. The output of this subsystem includes the new position of the train and camera system and the spatial coordinates of the detected circle.

4.3.6. Description of Graphical User Interface

This section will explain the features and functions of the GUI used in this research for the target detection and tracking scheme. The GUI has many features useful for keeping track of the simulation results and inputs as well as a user friendly plotting interface. The GUI is created using the Matlab® software environment. To explain the GUI in detail, its features will be broken down into separate sections. The simulation inputs will be explained in Section 4.3.6.1. The plotting section will be explained in Section 4.3.6.2, which allows for any of the selected plots to be analyzed after the simulation is complete. The final section of the GUI to be covered is the video analysis portion. This stage will allow image frames at different stages throughout the scheme to be analyzed and help the user to adjust different parameters according to the simulation results or to simply view the output. The video analysis stage will be explained in Section 4.3.6.3. The GUI used for this research is shown in Figure 4.25.


Target Detection and Tracking GUI

UAV Location
Location: Base

Target Detection
Radius: 9.5 Pixels
Bias: 2 Pixels
Maximum Mean Square Error: 70 Pixels
Hough Transform x-axis Resolution: 1 Pixels
Hough Transform y-axis Resolution: 1 Pixels

Pre-Processing Stage
Intensity Transformation
Method: Contrast-Stretching
Input 1: 0.8
Input 2: 4
Sobel Edge Detection
Sensitivity threshold: 0.24

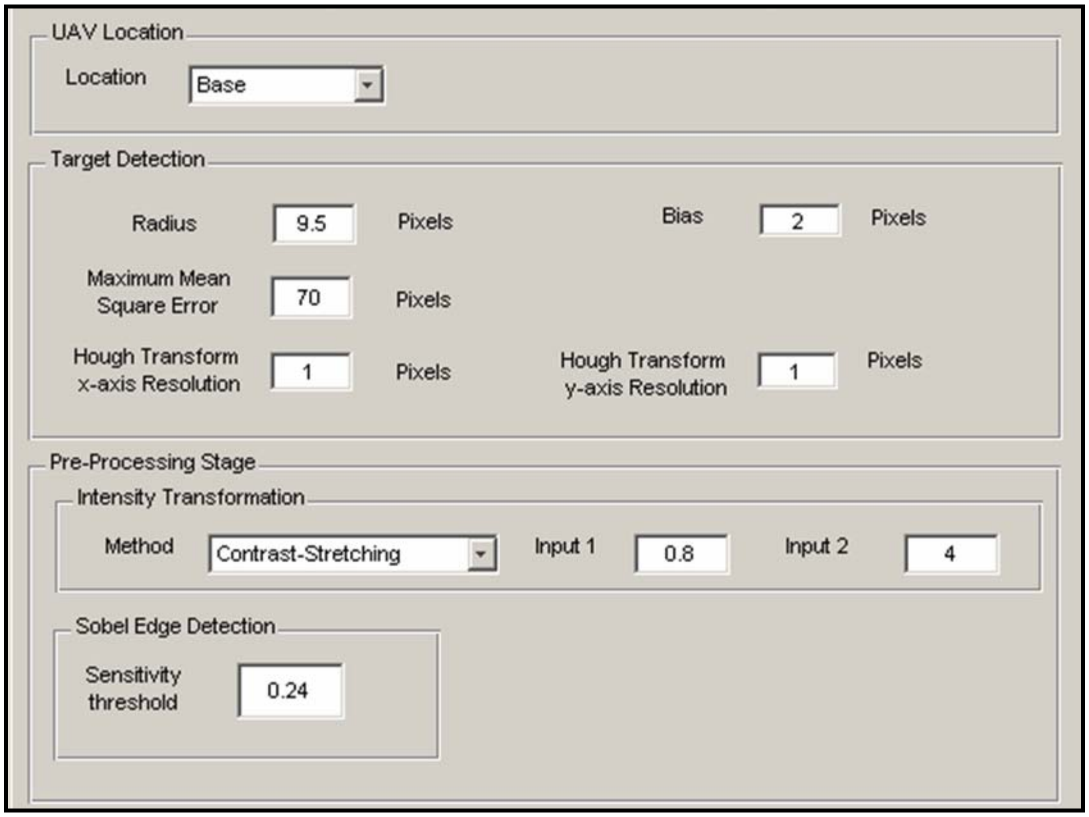
Plotting
○ On
○ Off
☐ UAV Location
☐ Train Location
☐ Number of False Detections
☐ Train Speed
☐ UAV Speed
○ Individual
○ Together
○ Subplot

Simulation Parameters
Sampling Speed: 1 Hz
Simulation Time: 25 sec

Figure 4.25: Target Detection and Tracking GUI

4.3.6.1. Simulation Inputs

The simulation input portion of the GUI is self explanatory. When the GUI is opened each input will be represented with a default value, which is displayed in their respective edit box. The simulation inputs are broken up into three sections: the "UAV," or camera, starting position, the pre-processing inputs, and the target detection inputs. The simulation inputs section of the GUI is shown in Figure 4.26 and a description of each input is provided below.


UAV Location

Location: Base

Target Detection

Radius: 9.5 Pixels

Bias: 2 Pixels

Maximum Mean Square Error: 70 Pixels

Hough Transform x-axis Resolution: 1 Pixels

Hough Transform y-axis Resolution: 1 Pixels

Pre-Processing Stage

Intensity Transformation

Method: Contrast-Stretching

Input 1: 0.8

Input 2: 4

Sobel Edge Detection

Sensitivity threshold: 0.24

Figure 4.26: Simulation Inputs

The following breakdown explains each input parameter:

1. *Location* – The "UAV," or camera, starting position allows the user to choose where the camera system should start at the beginning of the simulation. There are five options to choose from. The initial position of the camera is represented as the "base," which is represented by the coordinate (0,0) in the grid layout of the model environment. The other four locations are shown in Table 4.7. The distances are measured with respect to the "base" location in the grid layout. The "UAV," or camera

system, will then move to the starting position before the simulation is started. The camera starting position input is used for the initial camera position block in Figure 4.24.

Table 4.7: Initial Camera Starting Location Options

Location	x (mm)	y (mm)
Sector 4	171.45	525.15
Sector 3	984.25	525.15
Sector 2	984.25	101.60
Sector 1	171.45	101.60

2. *Method* – This allows the user to test four different intensity transformation methods. The method chosen for this research is the contrast-stretching operation. The Pre-Processing S-function block shown in Figure 4.11 requires this input as a parameter to select the correct intensity transformation method.
3. *Input 1* – Since the contrast-stretching operation is used in this research, this input will only be discussed in regards to that particular method. This input is used to choose the threshold pixel intensity value for the contrast-stretching function. The Pre-Processing S-function block shown in Figure 4.11 requires this input as a parameter.
4. *Input 2* – Since the contrast-stretching operation is used in this research this input will only be discussed in regards to that particular method. This input is used to control the slope of the contrast-stretching function. The Pre-Processing S-function block shown in Figure 4.11, requires this input as a parameter.
5. *Sensitivity Threshold* – This represents the intensity gradient threshold value used in the Sobel edge detection operation. Any gradient value below this threshold is disregarded as an edge pixel in the resulting binary image. This value is used in the Edge Detection block shown in Figure 4.11.
6. *Radius* – The circle radius input is used for both circle detection algorithms. The radius value is a parameter used in the Circle Detection

S-function block shown in Figure 4.15. The circle detection algorithms then search the image frame for circles with the user defined radius. This input value sets the upper cut off threshold for the range of radii in the FFF circle detection algorithm.

7. *Bias* – This is used to determine the cutoff radius threshold value for the FFF circle detection algorithm. Essentially, the FFF algorithm is designed to detect circles with a radius within a certain range. This allows a margin of error to be accounted for in the detection process. The bias input quantifies the margin of error by setting the lower limit of the range. This value is a parameter used in the Circle Detection S-function block shown in Figure 4.15.
8. *Maximum Mean Square Error* – This input is used to set the maximum MSE threshold value for the FFF circle detection algorithm. Any image feature that has a mean square error above this threshold value is disregarded as the target feature. This input is used in the Circle Detection S-function shown in Figure 4.15.
9. *HT x-axis Resolution* – Resolution for the descritization of the x values used in the HT Circle Detection S-function block shown in Figure 4.16. This number determines the interval of the *a* values in the parameter space.
10. *HT y-axis Resolution* – Resolution for the descritization of the y values used in the HT Circle Detection S-function block shown in Figure 4.16. This number determines the interval between the *b* values in the parameter space.

4.3.6.2. Trend Plotting Section

The trend plotting section of the GUI is very useful for fine tuning the input parameters or for simply viewing the results. This portion of the GUI enables the user to obtain a very in depth data analysis routine by using the check boxes to select specific data to plot. The plotting routine can then be executed by the plot now button and controlled by the on/off buttons. If the plot button is turned off then no data will be plotted until the plot now button is pressed. If the plot button is turned on then the plots will automatically be displayed after the simulation has been completed. After each simulation, the information obtained from the MV camera is collected along with the error values. This data can then be plotted to analyze the results. The values of the inputs along with the data obtained from the simulation can also be saved if the user chooses to do so. This is useful to obtain certain trends in the data as the input parameters are altered throughout each simulation. The user now has the ability to determine how a particular change altered the performance of the Simulink® scheme by comparing the results of different experiments.

There are also options that allow the user to choose the type of plots. The options on this GUI allow the user to select a single plot per figure type, multiple plots per figure type, or a subplot figure type. This additionally allows the user to analyze the results in a fashion such that results can be compared on a single plot or across multiple plots. This way an easy comparison can be made of the results when the input parameters are adjusted. The trend plotting section of the GUI is displayed in Figure 4.27.

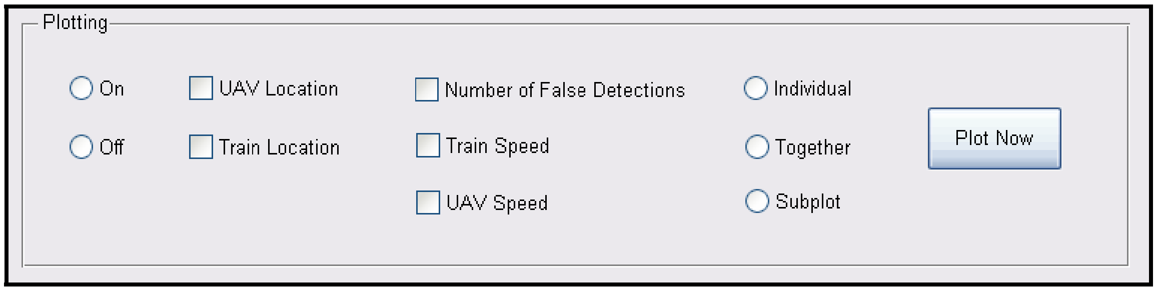


Figure 4.27: Trend Plotting Section of the GUI

4.3.6.3. Video Analysis

Another approach utilized to analyze the target detection and tracking scheme can be accomplished through visual interpretation. This is possible by examining several video output windows located at different stages in the Simulink® model. Through this analysis the user will be able to see the results of the pre-preprocessing stage, which includes the intensity transformed image, the image cropping routine results, and the edge detection and edge filtering results. The user can also determine how well the circle detection software is working.

During the operation of the simulation, five video windows appear. The first window is the unaltered input image captured from the image capture routine. The second window shows the result of the intensity transformation and cropping routines, together. This is useful to analyze the characteristics of the train around the edges of the quadrants within the image frame, which will help determine the necessary size of the buffer zone. The user has the ability to watch the train location as it approaches the edges of the cropped image. If the train passes through the buffer zone before it is detected, this will result in a false detection. This video analysis window will allow the user to watch for these types of events and, if necessary, make any corrections accordingly.

The third video window that is available is the result of the edge detection routine performed on the image after pre-processing has occurred; the result is a binary image. At this point the user will begin to visualize the presence of the desired image features. A clear representation should be apparent; if this is not the case then the intensity transformation input values should be changed within the Pre-Process S-function. There may also be significant amounts of noise in the binary image. If this is true it may be necessary to make an adjustment to the sensitivity threshold for the edge detection routine. If there is a problem with the circle detection algorithm, then this is the window where the root of the problem is displayed.

The forth window is the output of the Edge Enhancement S-function in the Pre-Processing subsystem. The result is a filtered binary image of the resulting image obtained from the edge detection routine. Here, the result of altering the filtering threshold value is displayed. The purpose of the threshold is to help reduce the image noise, which affects the computation time of the circle detection algorithm. To find the appropriate threshold value, the user can visually analyze the results during the simulation and determine if the threshold is too high or too low. If the threshold value is too high then this could potentially eliminate parts of the desired image feature, which would reduce the detection rate of the circle detection algorithm. In the other case, too much noise will be present in the image, which can affect the computation workload for the software.

The fifth window is the result of the Circle Detection subsystem. This display shows the result of the detected feature that satisfies the conditions established by the user in the GUI. The center location of the circular feature will be plotted on the binary input image. This video window is another place where the user can possibly determine what is causing the scheme to not perform correctly. This will allow the user to determine if the center location of the detected feature actually belongs to the desired feature. If the wrong feature is detected, it may be necessary to alter the detection criteria to help eliminate false detections. A large range for the radius criteria and a large maximum mean square error value will result in the detection of undesired features in the image. In the other case where a small range of radii is set and small maximum mean square error value is chosen, it will make it difficult to detect any features in the image. A complete analysis of the scheme can then be performed by looking at each video window and analyzing the affects of changing the input parameters in the GUI. A screenshot shown in Figure 4.28 and Figure 4.29 includes all the video analysis windows available to the user.

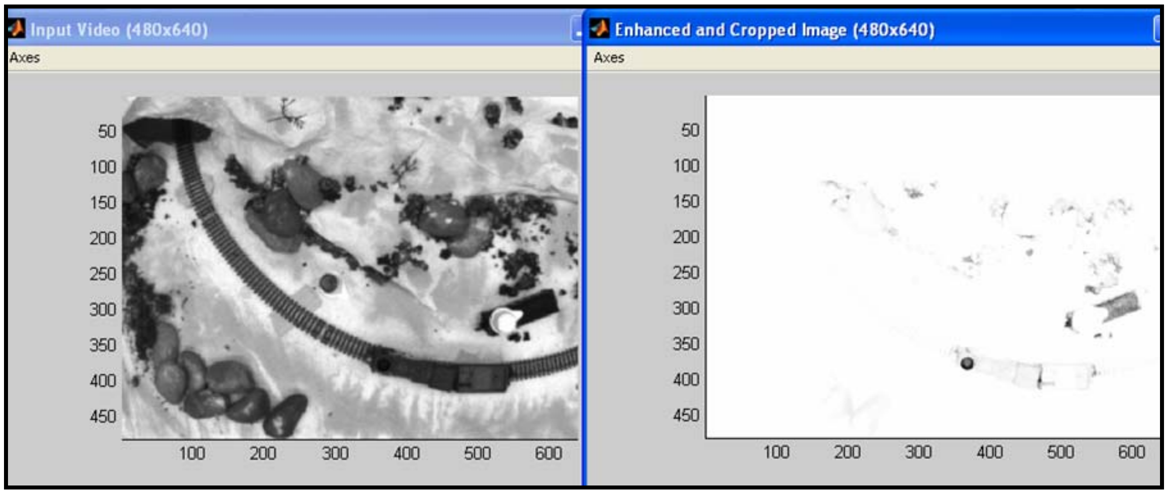


Figure 4.28: Output Window 1 and 2 of Simulink® Scheme

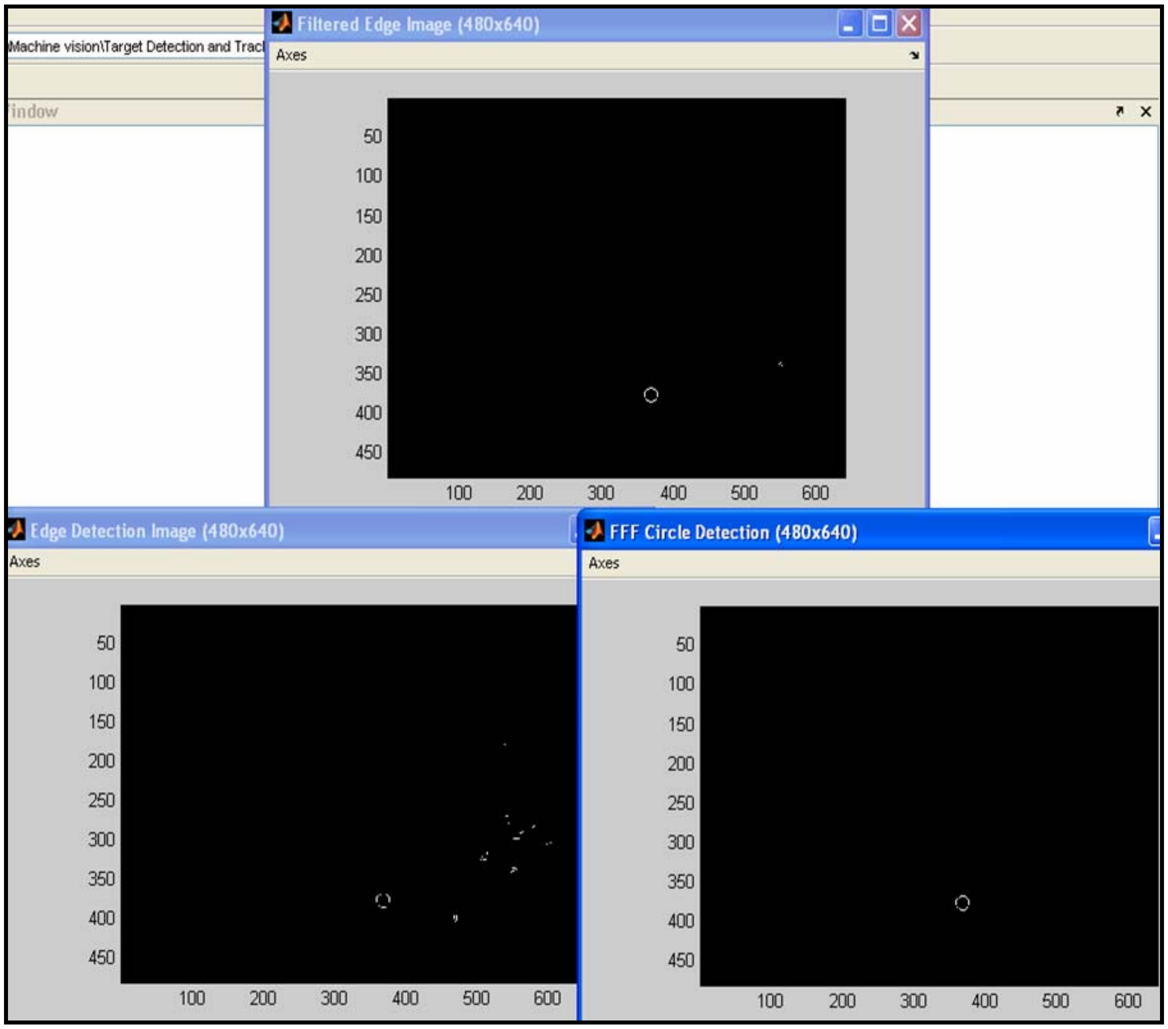

Machine vision\Target Detection and Tracl
Filtered Edge Image (480x640)
Axes
Edge Detection Image (480x640)
Axes
FFF Circle Detection (480x640)
Axes

Figure 4.29: Output Window 3, 4, and 5 of Simulink® Scheme

Chapter 5. Simulation Results and Evaluation

5.1. Target Detection and Tracking Results Using the FFF Algorithm

The solution to the target detection and tracking problem can be evaluated several ways. For this research effort, the detection and tracking scheme is put through a number of tests to display its overall performance in terms of computational workload, robustness, and efficiency. These tests include a statistical evaluation and a visual means of evaluation of captured frames from a live video feed. The use of a live video feed, however, makes it difficult to obtain similar results through each test due to the randomness of image characteristics in the video. These characteristics are largely influenced by the motion of the camera, where a captured frame may have more or less motion introduced than other captured frames. Another key note is the computational workload, which is increased by introducing the use of a frame grabber procedure within the detection and tracking procedure. The use of a frame grabber procedure with Matlab® or Simulink® also introduces a time delay due to the software is accessing the camera hardware through the Windows operating system, which can cause the results of the computational workload analysis to be slightly skewed. It is possible that this time delay could be reduced if another operating system other then Windows is utilized. The results of each experimental test conducted will be discussed in detail in the following sections. Section 5.1.1 explains the computational workload evaluation of the enhanced and original target detection and tracking software. This section analyzes the computational workload of the software when the amount of image data becomes significantly increased. The next section, Section 5.1.2, is used to analyze the error estimation for the estimated train position. Section 5.1.3 entails the visual means of evaluation for the train velocity and acceleration estimations, which is comprised of visually evaluating the output of the scheme and determining if it is working properly. Finally, Section, 5.1.4, evaluates the affect of changing the luminance or lighting conditions. In this section, the position estimation results will then be compared at each test condition. The following tables represent the simulation tests, which are used to evaluate the target detection and tracking algorithms in Sections 5.1.1, 5.1.2, 5.1.3, respectively. Table 5.1 shows the experimental breakdown for the computational workload evaluation of the software. In this table each test is a 25 second simulation using an update frequency of 1 Hz. Four tests were performed at two different threshold values for the edge detection method, which was then used to simulate S/N variations.

Table 5.1: Computational Workload Test Breakdown Used for Evaluation

Computational Workload Evaluation Via Matlab® Profiler	
Test 1: Threshold Condition #1 for Sobel Edge Detection	Test 1: 25 sec simulation at 1 Hz.
	Test 2: 25 sec simulation at 1 Hz.
	Test 3: 25 sec simulation at 1 Hz.
	Test 4: 25 sec simulation at 1 Hz.
Test 2: Threshold Condition #2 for Sobel Edge Detection	Test 1: 25 sec simulation at 1 Hz.
	Test 2: 25 sec simulation at 1 Hz.
	Test 3: 25 sec simulation at 1 Hz.
	Test 4: 25 sec simulation at 1 Hz.

Table 5.2 shows the experimental breakdown for the error evaluation of the software.

Table 5.2: Error Estimation Trial Breakdown Used for Evaluation

Position Estimation Error	
"Actual" vs. Estimated Target Position	**RMS Error**
Trial 1: 25 sec simulation at 1 Hz; train travels CW.	Trial 1
Trial 2: 25 sec simulation at 1 Hz; train travels CCW.	Trial 2
Trial 3: 25 sec simulation at 1 Hz; train travels CW and stops.	Trial 3
Trial 4: 25 sec simulation at 1 Hz; train travels CW and then CCW.	Trial 4
Trial 5: 50 sec simulation at 1 Hz; train travels CW.	N/A

In this table each trial represents a different path taken by the target, which is explained later in Section 5.1.2. The first four trials are used to show the error evaluation for a 25 second simulation while using a 1 Hz update frequency. The fifth trial is used to show the result from a 50 second simulation at 1 Hz. The RMS error is then calculated between the "actual" and estimated target position for Trials 1 through 4.

Table 5.3 shows the experimental test breakdown for the tests performed at the second lighting condition. Each test is used to display the target position estimation results obtained from a 25 second simulation with a 1 Hz rate.

Table 5.3: Robustness to Change in Lighting Test Breakdown Used for Evaluation

Robustness to a Change in Lighting
Test 1: 25 sec simulation at 1 Hz; train travels CW.
Test 2: 25 sec simulation at 1 Hz; train travels CW.
Test 3: 25 sec simulation at 1 Hz; train travels CW.

5.1.1. Computational Workload Analysis

To evaluate the efficiency of each method utilized for the target detection and tracking algorithm, a comparison is made using the Matlab® Profiler. The profiler method allows the computational workload of each function within the scheme to be analyzed independently. This includes the total time each function took to execute as well as the number of times each function is executed. Also included in this analysis, is the code breakdown for each function, which includes the number of times each line in the code is executed and the total execution time. The profiler data is used to obtain the computational workload of each function used within the scheme. This breakdown allowed each section to be compared within the software scheme. Each algorithm is broken down into five sections, which are explained in the following list:

1. *Capturing the Image Frame* – The computer creates a video input object and acquires a single image frame from the video object;
2. *Pre-Processing* – This consists of each pre-processing method used in the scheme;
3. *Circle Detection* – The method used to detect the distinguishable feature on the train. The FFF algorithm and HT method are used for this section;
4. *Motor Control* – This is utilized to control the stepper motor inputs for position and speed;
5. *Other Lines and Overhead* – This is used for all other lines within the scheme, such as matrix reshaping and image class conversions.

The computational workload analysis is completed for each version of the target detection and tracking software for two different conditions. For each condition, the Sobel edge detection threshold value is altered to display the affect this specific value has on the computation time for other sections. The first step in this procedure is to determine the threshold value that will obtain the best overall result. This is completed by performing an edge detection analysis at four different locations within the experimental environment, as shown in Table 5.4.

Table 5.4: Sobel Edge Detection Analysis

Target Detection and Tracking: *Sobel Edge Detection Analysis*				
Location	**Maximum Gradient Value**	**Minimum Gradient Value**	**Number of Edge Points**	**Average Gradient Value**
Sector 1	0.4	0.0082	48	0.21
Sector 2	0.45	0.0183	57	0.25
Sector 3	0.41	0.0019	54	0.25
Sector 4	0.4	0.0098	56	0.24
Average Value	0.42	0.0096	54	0.24

The results in this table are obtained from images captured at the sector locations listed in Table 4.7. In Table 5.4, the gradient values referred to are the edge pixels of the detected target found using the FFF circle detection algorithm. Once the circle is detected, the edge pixel locations are obtained and then used to obtain the gradient values within the gradient magnitude matrix approximated with the Sobel edge operator. Using this procedure, the maximum and minimum pixel gradients values are obtained at each pixel location. This table also identifies the total number of edge points that are detected with the FFF algorithm.

The results in Table 5.4 above are used to determine the best overall threshold value for the Sobel edge detection method. After analyzing the edge detection results the first analysis for the computational workload is completed using a gradient threshold value of 0.20. This value is set below the average value to account for blurriness obtained when the image is captured during the simulation. When the image is blurred the edges will result in weaker gradient magnitudes. The second test is conducted using a gradient threshold value of 0.10. This value is chosen to be fifty percent smaller than the first condition, which allows for weaker edges to be detected within the captured image.

This will display the timing characteristics of the pre-processing stage and circle detection algorithm when the number of edge pixels is drastically increased.

Table 5.5 lists the time spent on each section of the software previously described using the enhanced and original version for the image representation method. Using the unmodified version for image representation, the entire image is used as an input for the target detection and tracking scheme throughout the simulation tests. For the enhanced version, the image cropping routine is then executed to reduce the size of the input image processed in the scheme.

Table 5.5: Timing Comparison Between Target Detection and Tracking Methods

Threshold Value for Sobel Edge Detection	Target Detection/Tracking Speed Comparison using *Matlab® Profiler*	Detection and Tracking Algorithm		Modified Detection and Tracking Algorithm	
0.2	**Machine Vision Process**	**Time (s)**	**Executions**	**Time (s)**	**Executions**
	Capturing the Image Frame	2.31	26	2.35	26
	Pre-processing	7.12	26	5.21	26
	Circle Detection	1.41	26	1.30	26
	Motor Control	11.07	26	10.29	26
	Other Lines and Overhead	3.76	N/A	4.07	N/A
	Total Time	25.67	N/A	23.22	N/A
0.1	**Machine Vision Process**	**Time (s)**	**Executions**	**Time (s)**	**Executions**
	Capturing the Image Frame	2.10	26	2.23	26
	Pre-processing	7.40	26	5.69	26
	Circle Detection	10.27	26	8.22	26
	Motor Control	11.28	26	11.28	26
	Other Lines and Overhead	4.08	N/A	4.03	N/A
	Total Time	35.13	N/A	31.46	N/A

Results in the previous table are obtained from simulations performed for 25 seconds with a 1 second update. After analyzing the results, several comparisons can be made between each version of the target detection and tracking software and the results obtained for each threshold value. The first comparison will be used to evaluate the difference between each software version since the characteristics are similar throughout the timing comparison. The first result is the time required to capture the image frames during the experiment. This procedure accounts for approximately 2.3 seconds of the total simulation time. Technically, these values should be equal because this process is exactly the same for each experiment; however, there are slight differences in the times.

This can be credited to the different processes the Windows software is running in the background. It is also believed that the timing result for this process could be reduced if operating software, other than Windows, is used.

The largest time variation is found in the pre-processing stage. In the unmodified software version the entire image is processed; this accounts for 307,200 pixels in each captured image. On the other hand, the enhanced version performs the same operations on a smaller portion of the image, accounting for only 172,800 pixels in each captured image. This is a 44% decrease in the amount of image pixels used for the pre-processing stage; the effect of this difference is evident after comparing the timing difference between the two methods. The fact that the image scanning area is drastically reduced in size directly correlates to the increased efficiently of the software. For example, the pre-processing stage for the unmodified version takes 7.12 seconds, in comparison to the 5.21 seconds it takes for the pre-processing section for the modified version. This difference accounts for a 26.83% decrease in time spent on this section.

The circle detection section of the software is also affected by the difference in each software version. Using the results obtained from the first threshold test, the unmodified target detection and tracking software spends 1.41 seconds scanning the entire image for the target while the modified version takes 1.30 seconds, utilizing only a portion of the image in the scanning process. This difference accounts for a 7.8% decrease in time spent on this process. A combination of a high threshold value for the edge detection method along with the image filtering routine used in the pre-processing stage eliminates a large amount of the noise in the image before the circle detection algorithm is performed. It may appear that the decrease in input image size between each software version may not be beneficial for this section because the computational time has been minimally impacted. However, the computational workload for this section is better displayed from the results obtained for the second threshold test condition, using the lower threshold value. The circle detection algorithm for the unmodified software version takes 10.27 seconds and 8.22 seconds for the modified version. This difference accounts for a 23.10% decrease in computation time, a much more noticeable affect.

One last trend that should be noted within the comparison between the software versions is the time spent on the motor control section. This section relates to the tracking portion of the software and accounts for almost half of the time necessary for the total simulation time. This section is strictly used to convert the information obtained in image processing portion of the software to motor inputs referring to position and velocity. There is some difference between the two methods related to the computational time for this section. This difference can be directly correlated to the fact that the total time for this section is accounted for in the time it takes the camera to move from one position to the next; therefore, the time varies slightly depending on the distance the camera must travel.

Finally, after reviewing the computational time for each section, the total simulation time taken by each algorithm is shown. Using these results, it is apparent that the modified version for the target detection and tracking software is more efficient resulting in a decreased computational workload. An important characteristic of the computational time is that the software has real time capabilities. This is only capable using the modified software version where the average computation time per frame is 0.92 seconds, which is equivalent to an update rate of about 1 Hertz (Hz).

The second comparison used to evaluate the efficiency of the software algorithms is accomplished by displaying the effect of changing the threshold value for the Sobel edge detection method. This is performed to identify the influence of adding more image noise to the target detection portion of the software. To quantify the amount of noise added to the image, the amount of image pixels is compared between each threshold condition. In arbitrarily captured binary images there are 1,460 and 204 pixels using a threshold value of 0.10 and 0.20, respectively. This accounts for an 86.03% decrease in the amount of image pixels after the edge detection routine has been completed. After reviewing the results, the change in threshold value has the greatest affect on the computation time for the circle detection stage. This stage, for the first threshold condition, has a computation time of 1.30 seconds and a computation time of 8.22 seconds for the second test condition. The difference in time accounts for an 85% decrease. This alteration has little to no affect on the other stages of the software. Therefore, it is determined that the amount of image pixels resulting from the edge detection routine has the greatest influence on the circle detection stage, drastically increasing the computation time.

5.1.2. Position Error Estimation

Error analysis is an important aspect for the performance evaluation of this research effort. This evaluation tool is important for occurrences where the software is unable to or falsely detects the target. However, since the software utilized during this research is very efficient, other means of error estimation must be formulated. There are two methods that are implemented to perform this evaluation. The first method compares the detected train position to the train track position, which is obtained by measuring its location within the model environment. It was initially thought that this would be an accurate evaluation method. However, after performing this method, it is concluded that it is an inadequate procedure for error estimation; a false representation of the position error is introduced. Therefore, a second method for the error evaluation is formulated, which compares the estimated train position to the "actual" train position. The "actual" train position is determined by measuring the location of the train on the train track within each input image. To measure the location of the train, reference points on the train track are pre-recorded within the 2-D grid and the train position is then extrapolated between the reference points in each captured image. There are a total of 21 reference points on the train track. In an ideal situation there would be a sensor, such as GPS, to obtain the exact location of the train. This is not the case in this experiment; therefore, the "actual" train position may have slight error induced from the extrapolation process. However, for this experiment, the method used to obtain the "actual" train position is sufficient enough to display the error estimation.

For each error evaluation method there are five trial simulations performed, in each trial the train travels a different path. This is necessary to show the overall effectiveness of the target detection and tracking software. Each train path is represented by a trial number, where each trial is explained below:

1. The train travels clockwise around the track;
2. The train travels counterclockwise around the track;
3. The train travels clockwise around the track. At two locations the train stops for a couple seconds and then continues the same path;
4. The train initially travels clockwise around the track, then stops and changes direction;

5. The train travels counterclockwise around the track for an extended simulation time.

Trial 5 is used to evaluate the response of the software when the train enters the tunnel system.

Although the first error evaluation method is determined to be an inaccurate procedure, the results are provided to obtain a better understanding of the process. The results of both methods are then compared to display the superiority of the second error evaluation procedure. The procedure of the first method is shown in Figure 5.1, which displays the estimated train position and the train track position.

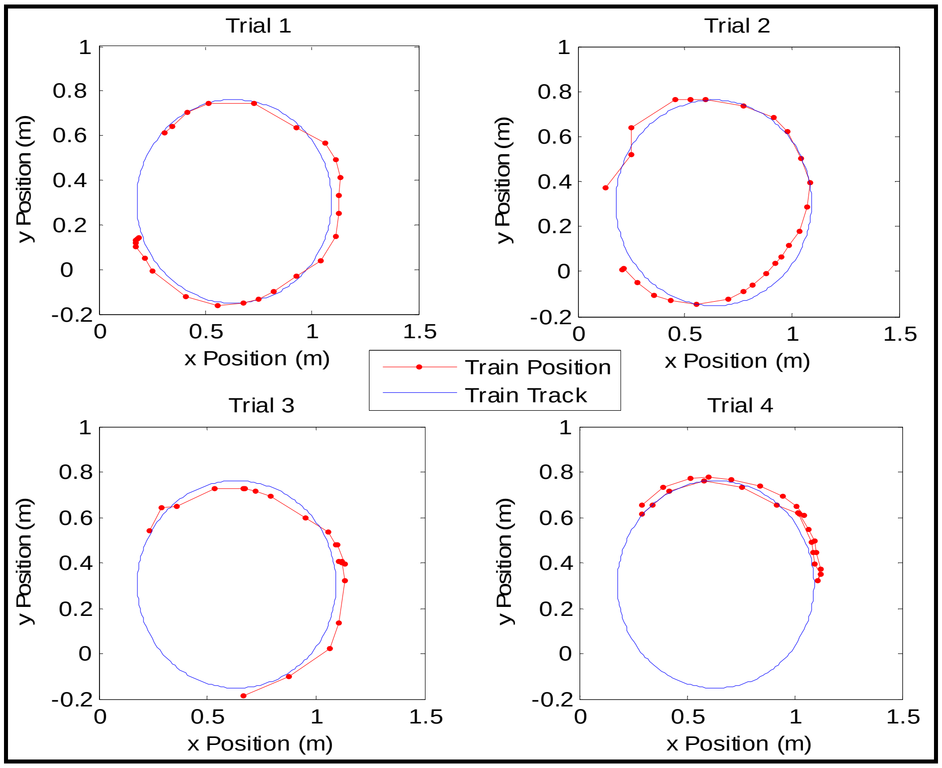


Figure 5.1: Train Track Position vs. Estimated Train Location for All Trials

The train track position is shown as the blue line and the detected train locations found from the MV camera are shown by the red data points. The measurements are in terms of *x* and *y* locations on the “ground” level of the experimental setup. As shown in the plots, some error is introduced between the detected location of the train and the train track. A major cause of this error is due to the movement in the camera as it changes position. In some cases the camera lens is not completely parallel to the “ground” surface in the model environment. This causes the camera measurements to be offset by some value, introducing error, as the results show in Figure 5.1.

A better representation of the estimated error is shown in Figure 5.2. Similar to the previous figure, the estimated position is shown in red and the track position is shown in blue. This plot is captured from the data obtained in the Trial 1 simulation shown in Figure 5.1. In this figure it is much easier to see how the estimated position overshoots the train track. At this time of the simulation the camera is slightly tilted causing the measurements to be offset, which is a major reason why this error evaluation method is inaccurate.

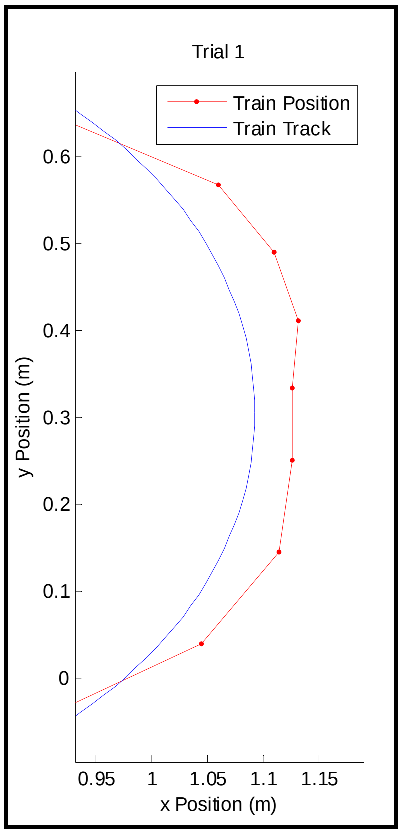


Figure 5.2 Train Track Position vs. Estimated Train Location for Trial 1

Another example of the error estimation is shown in Figure 5.3 Using the first method, the Euclidean distance is calculated from the measured position of the train to the train track.

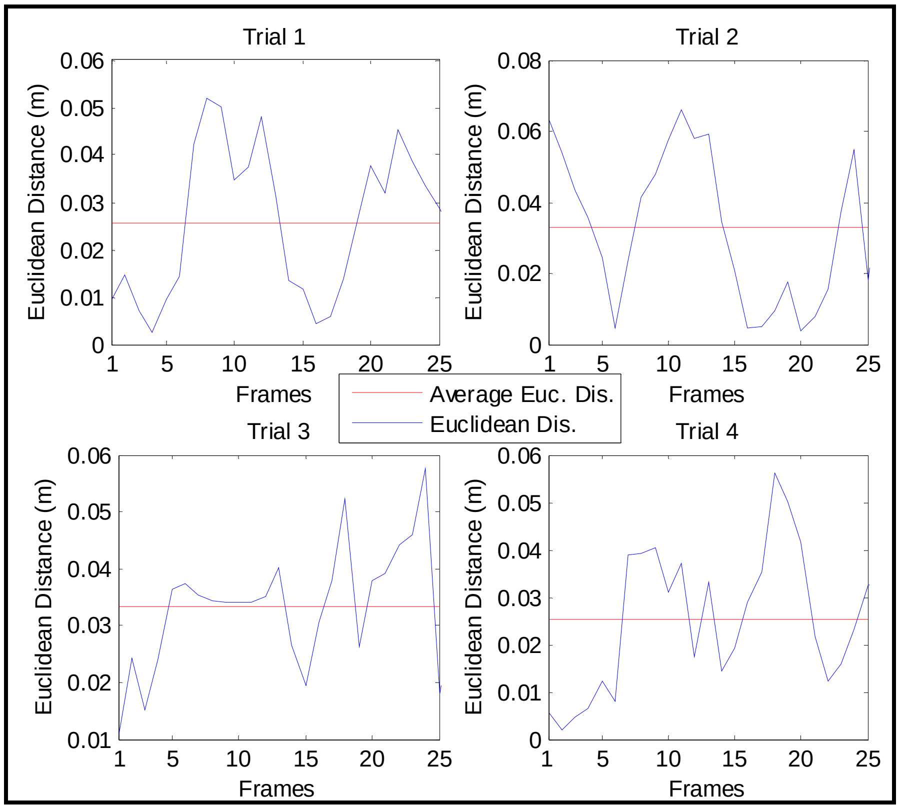


Figure 5.3: Euclidean Distance between Target and Train Track Position

The shortest Euclidean distance from the train track to the detected train position, shown in Figure 5.1, is considered to be the Euclidean distance for each estimated position. It is shown in Figure 5.3 that the error estimation is below 0.07 meters for all trials. In Trials 1, 3, and 4 the error is never above 0.06 meters. As stated previously, a significant portion of error is a result of the camera movement while changing locations in the experimental environment. To represent the overall error in each trial, the root mean squared error (RMSE) is calculated for the position error estimation. The RMSE of the Euclidean distance is shown in Table 5.6. The minimum, maximum, and mean Euclidean distances are also displayed for each trial.

Table 5.6: RMS Error between Estimated Train Location and Train Track

Trial	Max. Euclidean Distance (m)	Min. Euclidean Distance (m)	Mean Euclidean Distance (m)	RMSE (m)
1	0.0563	0.0026	0.0257	0.0299
2	0.066	0.0039	0.0331	0.0391
3	0.0577	0.011	0.0333	0.0349
4	0.0563	0.0021	0.0254	0.0293

After examining Table 5.6, the RMSE values are relatively similar for each trial. The results from Trial 2 produced the largest RMSE, which is due mainly to the error shown in Figure 5.2.

During each trial run there is no indication as to where the error exactly occurs. This is evident after examining each trial in Figure 5.1, where the error occurs at different locations throughout each trial. If the camera had the same orientation and path throughout each trial the results would better display the accuracy of the target detection and tracking software. Therefore, the error is caused from mechanical issues rather then a software design issue, which would require a redesign of the positioning apparatus. It is also evident from these results that the time required to process each image is sufficient enough to allow the camera to keep the train in its FOV as it follows the train along its path. If this is not the case then the train would eventually move out of the camera's FOV.

The error estimation for the fifth trial is shown in Figure 5.4. Similar to the previous figures, the red line represents the estimated train location and the blue line represents the train track location. The simulation in Trial 5 differs from the other trials shown in Figure 5.1 for the fact that this simulation is performed for 50 seconds rather then 25 seconds. This trial is used to analyze the error correction portion of the software, or display the actions of the camera positioning system when the train enters the tunnel. In Figure 5.4 the starting and ending points for the trial are labeled and the box represents the tunnel system in the model environment. The error plot in Figure 5.4 represents the flag output during the trial test, where an output of 1 represents positive target detection and an output of 0 represents zero target detection. After analyzing the results from Trial 5, the train exits the camera's FOV when it enters the tunnel. This situation occurs at two instances throughout the test, at frame 11 and 37. The reactions of the camera positioning system are different in each case. In the first case, at frame 11, the necessary inputs are given to the positioning system to allow the camera to detect the train as it exits the tunnel system, which is evident from the trial test results. This is similar for the second case, however once the train enters the tunnel it stops. Therefore, the train never reenters the camera's FOV after entering the tunnel. At this time, the camera is given the command to hold its position until the train enters its FOV again. In a real situation the UAV would be given the command to loiter until the train exits the tunnel system.

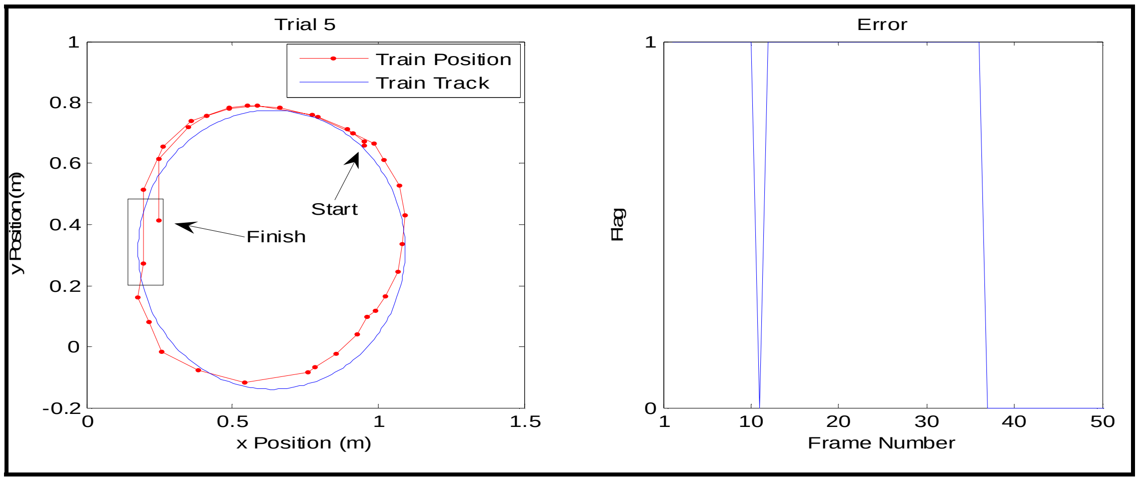


Figure 5.4: Results of Error Correction Scheme for Trial 5

The second method for the error evaluation is used to compare the estimated location of the train to the "actual" position of the train. Using this method a comparison is also made between the estimated values with and without added GPS error. This approach is taken because in a real situation, a UAV is usually equipped with a GPS receiver to determine its location and speed in time. With this in mind, there are several external sources that introduce error into the GPS measurements. To simulate this effect, an error of ±5 meters is randomly added to the estimated train position. The RMS error is then calculated and compared between each case of this error evaluation method. Figure 5.5 displays the error evaluation without GPS error for simulation Trials 1 though 4.

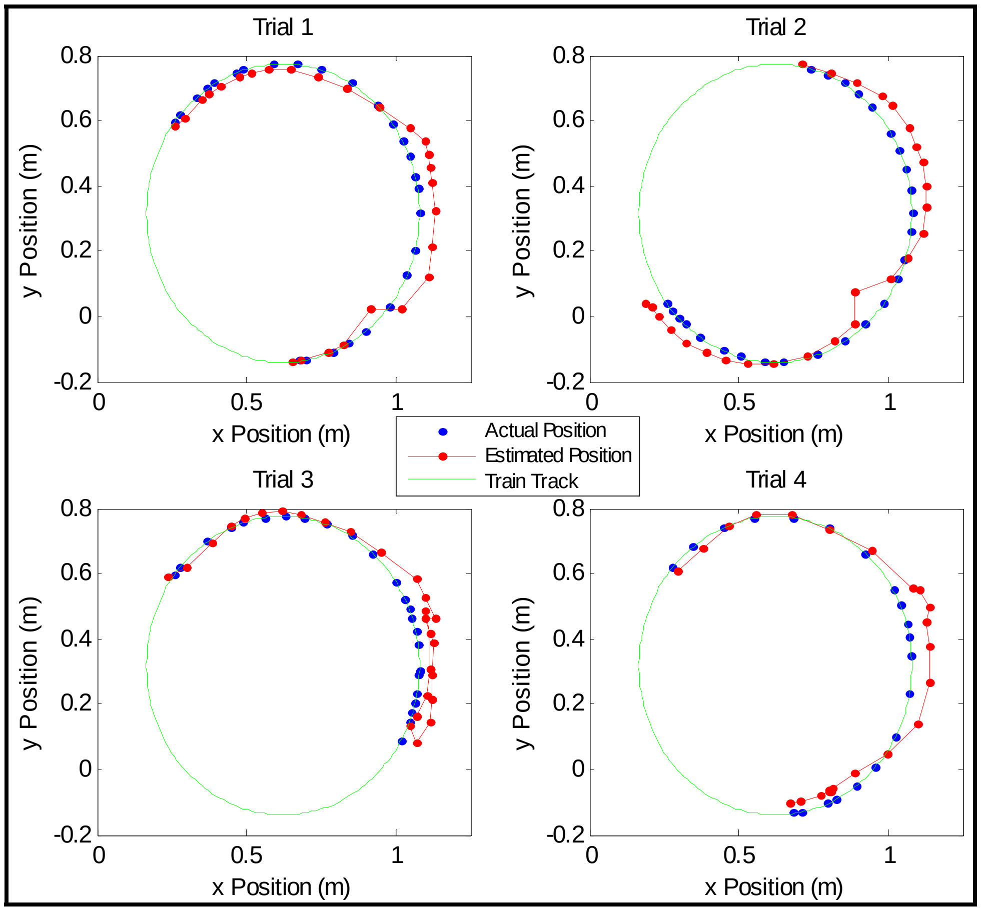


Figure 5.5: "Actual" vs. Estimated Location for All Trials without GPS Error

In Figure 5.5 the "actual" train position is shown by the blue data points, the estimated train locations are shown by the red data points, and the green line represents the train track. The measurements are in terms of *x* and *y* locations on the "ground" level of the experimental setup. A better representation of the error is shown in Figure 5.6, which displays the results of Trial 2, shown in Figure 5.5.

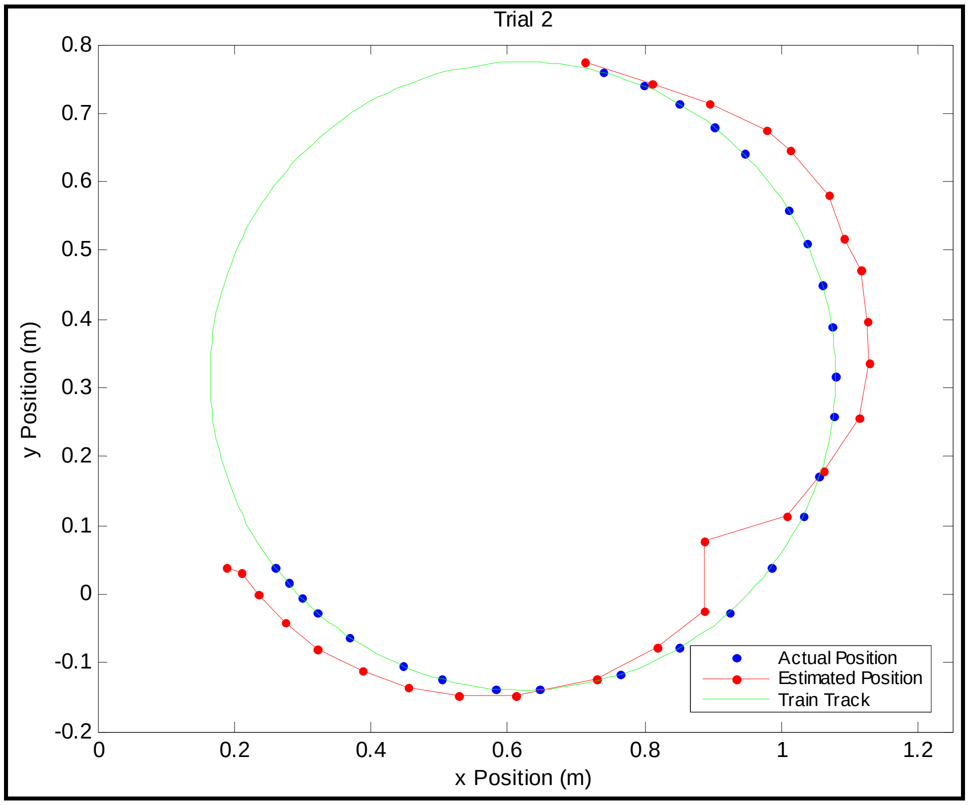


Figure 5.6: "Actual" vs. Estimated Train Location for Trial 2 without GPS Error

Before the results of this method are further discussed, a comparison is made between both error evaluation methods to fully understand each procedure. A portion of Figure 5.6 is used to compare both methods. Similar to the first method, the Euclidean distance is calculated, but following a different procedure. To visually display the calculated Euclidean distance from each method the comparison is shown in Figure 5.7.

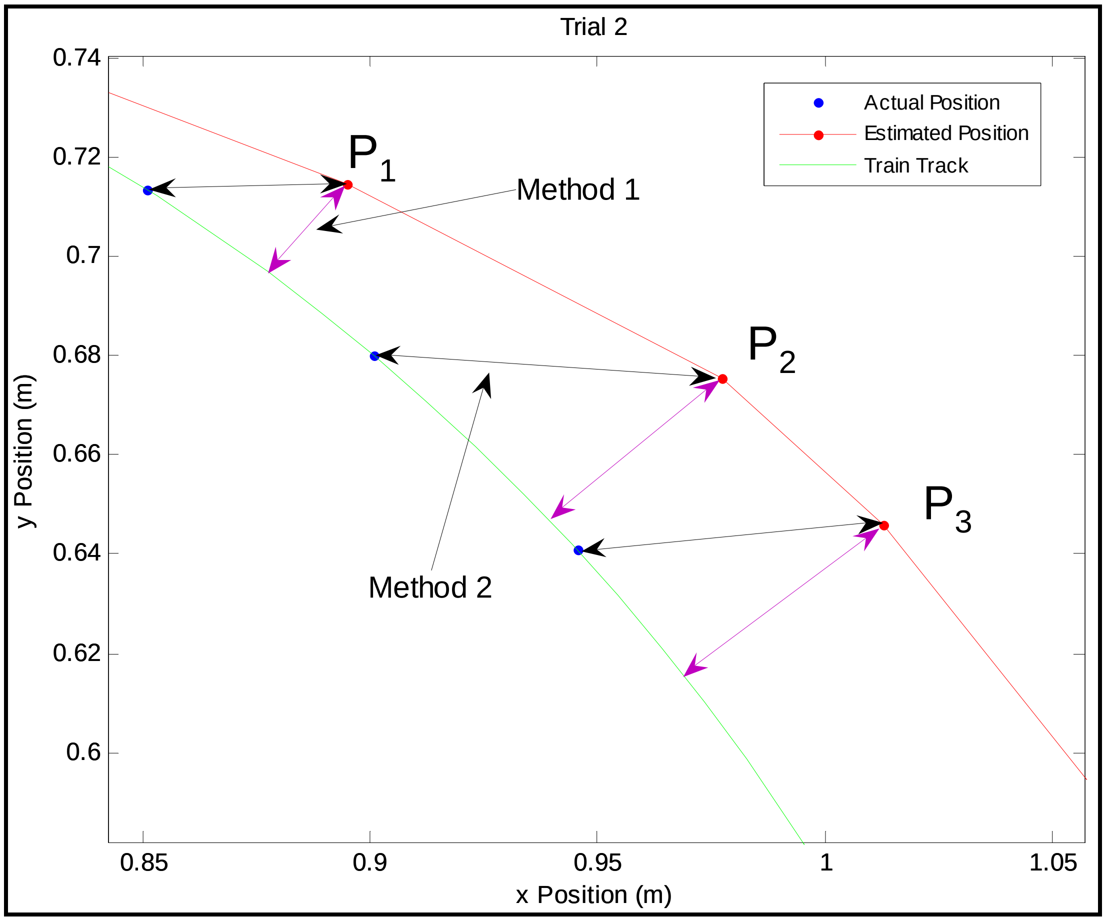


Figure 5.7: Euclidean Distance Calculation for Trial 2 without GPS Error

In the figure above, the purple and black arrows represent the Euclidean distance calculations using the first and second method, respectively. It is evident from the length of each arrow that the Euclidean distance can be quite larger when the estimated target position is further away from the "actual" position. This is the determining factor why the first method is much less accurate. To quantify the difference between the measurements, the Euclidean distance for each estimated target position shown in the above figure is shown in Table 5.7, which also includes the percent difference between each method.

Table 5.7: Comparison of the Euclidean Distance Calculation Methods for Trial 2

Point (P)	Euclidean Distance Using Method 1 (m)	Euclidean Distance Using Method 2 (m)	% Difference
1	.0249	.0439	43.28
2	.0467	.0767	39.11
3	.0532	.0675	21.19

It is evident that the second method displays more error in the position estimation results. However, it is more accurate for this evaluation procedure. As stated previously, this error is accounted for from the orientation of the camera changing as each image is captured. If the camera lens is exactly perpendicular to the "ground" level when each image is captured the error would be much less.

Using the second evaluation procedure, the error with the added GPS error is then displayed in Figure 5.8. In this figure the "actual" train position is shown by the blue data points, the estimated train locations with GPS error are shown by the red data points, and the green line represents the train track. The measurements are in terms of *x* and *y* locations on the "ground" level of the experimental setup.

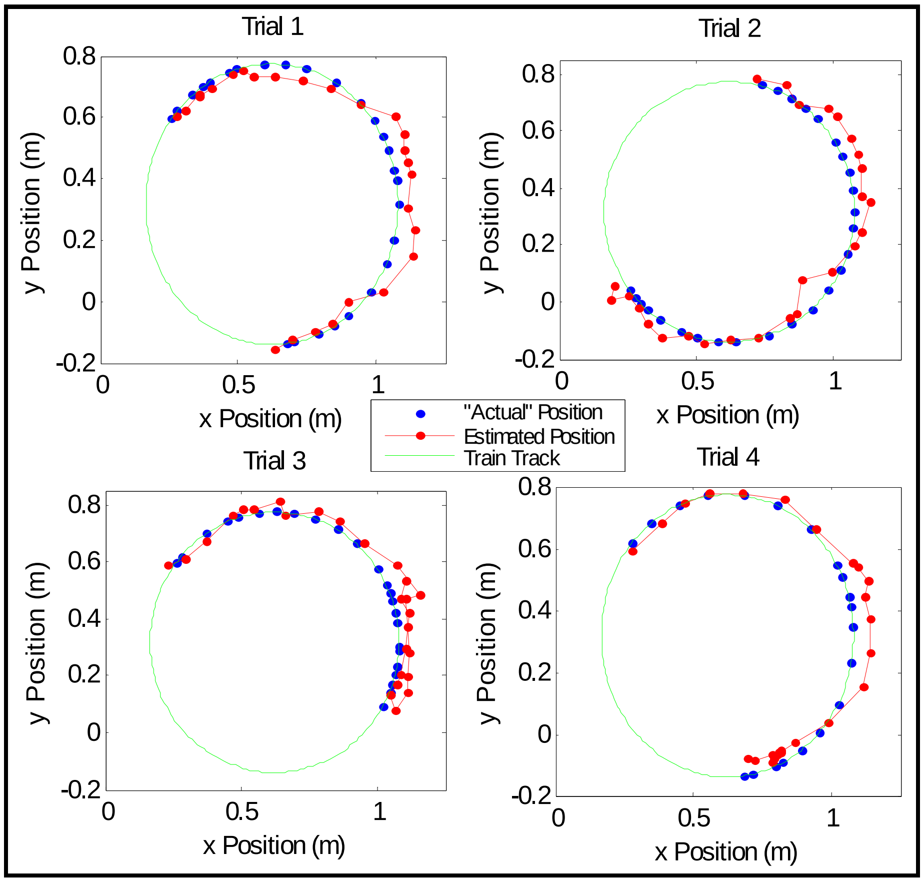


Figure 5.8: "Actual" vs. Estimated Location for All Trials with GPS Error

It is evident that the added GPS error introduces random error in the estimation of the target location. To quantify and compare the error between the two different cases, with and without GPS error, the RMS error is determined using the Euclidean distance calculations. The results are shown in Table 5.8.

Table 5.8: RMS Error between Estimated and "Actual" Train Location

	Trial	Max Euclidean Distance (m)	Min Euclidean Distance (m)	Mean Euclidean Distance (m)	RMSE (m)
Without GPS Error	1	0.0754	0.0072	0.0354	0.0413
	2	0.1053	0.0114	0.0502	0.0540
	3	0.0799	0.0053	0.0357	0.0417
	4	0.0930	0.0033	0.0424	0.0488
With GPS Error	1	0.0991	0.0091	0.0382	0.0457
	2	0.1180	0.0065	0.0505	0.0559
	3	0.0800	0.0075	0.0407	0.0460
	4	0.1157	0.0124	0.0486	0.0559

After examining Table 5.8 it is evident that the position estimation results with random GPS error produce slightly larger RMS error. Trial 2 produces the largest RMS error in each case. Although the RMS error is larger, it is apparent that the added GPS error does not have a significant affect on the results. It is also believed, in most cases, the GPS measurements would be more accurate then ±5 meters. There are only some situations which would cause the error to reach ±5 meters. For example, this could occur when several of the satellite signals is lost.

A comparison can now be made to represent the difference between the RMS errors for each error evaluation method. The RMS error values are shown in Table 5.9.

Table 5.9: Comparison between Error Estimation Methods

Trial	RMSE Using Method 1	RMSE Using Method 2 (w/o GPS Error)	RMSE Using Method 2 (w/ GPS Error)	% Difference of Method 1 & 2 (w/o GPS Error)
1	0.0299	0.0413	0.0457	27.60
2	0.0391	0.0540	0.0559	27.78
3	0.0349	0.0417	0.0460	16.31
4	0.0293	0.0488	0.0550	39.96

After examining Table 5.9, it is evident that the RMS error results found using the second method show larger RMS error. This further displays the inaccuracy of the first error evaluation procedure. For this reason the second error evaluation procedure is chosen as the more accurate evaluation method other then using a sensor to measure the exact location of the train.

Using the scale of the model environment, the estimated error can then be related to a real-life situation. Therefore, this experiment is comparable to a UAV flying approximately 68 meters above ground level while detecting and tracking a ground target. The error results at this altitude are shown in Table 5.10.

Table 5.10: RMSE at an Altitude of 68 Meters

Trial	RMSE Using Method 2 w/o GPS Error (m)	RMSE Using Method 2 w/ GPS Error (m)
1	3.60	3.98
2	4.70	4.87
3	3.63	4.00
4	4.25	4.79

To reduce the error it is necessary to enhance the design of the MV camera positioning apparatus. The camera should be perpendicular to the “ground” level at all instances throughout the simulation. Due to time and cost restrictions, this task was not feasible during this particular research effort, but should be noted for future work using this laboratory setup.

5.1.3. Velocity and Acceleration Estimations

Similar to the position estimation results, there is no measured data to compare the estimated values of the train velocity and acceleration results. For this reason, it is very difficult to analyze the accuracy of the results. In any case, the method used to estimate the velocity of the train is by utilizing the “*tictoc*” command in Matlab®. This command is used to output the measured time it takes to process each frame of the live video feed. This data along with the distance the train traveled between each sequential frame is used to estimate the velocity and in turn, estimate the acceleration of the train. The time output after each sample is different due to the fact that this includes the time it takes the camera to move to each location. The time it takes to capture each frame is approximately the same; variations with these times are due to the fact that Windows is running other programs in the background. The computation time for the pre-processing stage and circle detection algorithm are relatively similar, as well, between each frame. Variation in the computation time is then accounted for by the motor control section. Additional time is also required if the image falls under one of the cases that require the entire image to be processed. The computation time will be greater in this situation than in the other scenario that uses the cropped image as the input.

The results of the train velocity and acceleration estimates are displayed in Figure 5.9 and Figure 5.10, respectively. The velocity and acceleration estimations are only shown for the first four trials for the position estimation results shown in Figure 5.1. By close examination of these plots it is evident that the train is not moving at a constant velocity throughout each simulation. This is obvious due to the fact that it is difficult to keep the train moving at a constant velocity unless the maximum velocity setting is chosen. In this experiment, this would not be possible due to speed limitation of the stepper motors. On the other hand, an attempt to obtain slow train speeds would cause the train to stop on the track due to weak contact points between the two surfaces. Therefore, to keep the train constantly moving along the track it is necessary to manually adjust its power supply at instances when the train appeared as if it is going to stop. As a result, the velocity of the train constantly changed during the course of each experiment. However, the average velocity for each trial is relatively precise as seen from Figure 5.9 where this value is approximately 0.11 m/s.

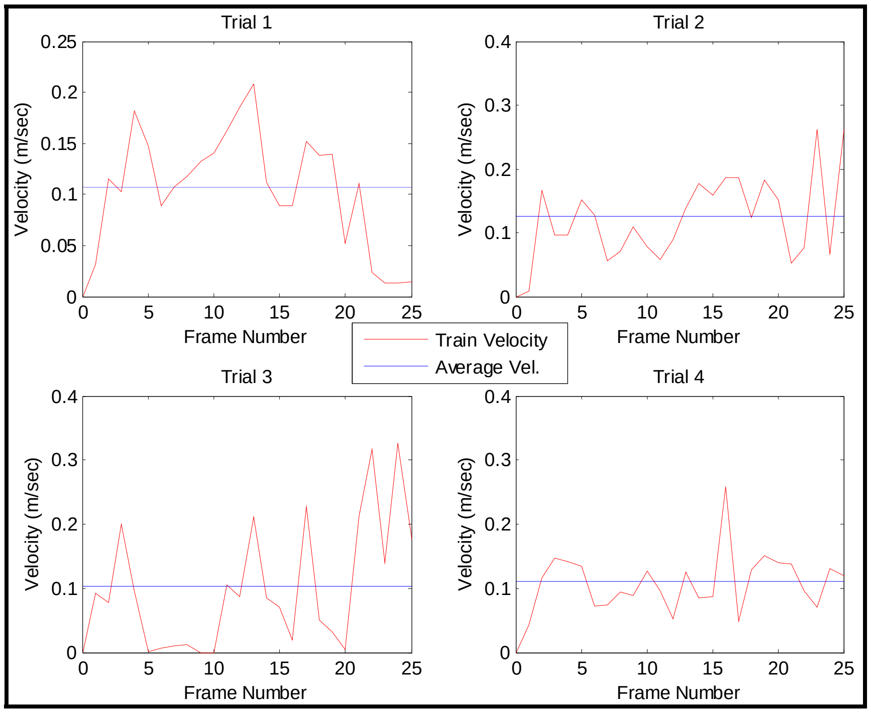


Figure 5.9: Velocity Estimation for each Trial

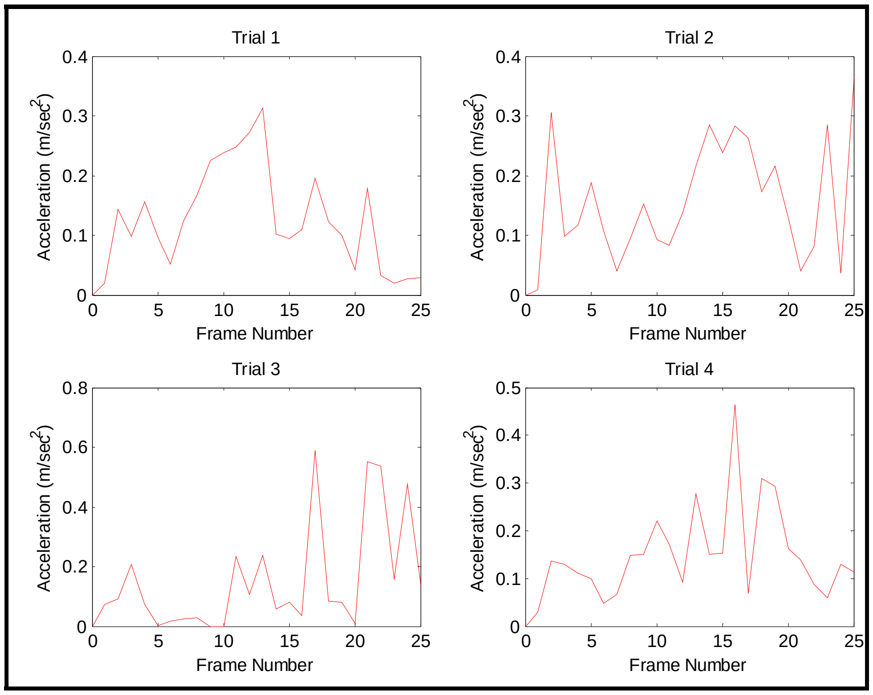


Figure 5.10: Acceleration Estimation for each Trial

After examining the figures above, it is evident that the velocity estimation procedure works well. In order to quantify this indication, the results from the position and velocity estimations should be compared for Trial 3. As previously stated, in this trial the train stopped at several instances during the simulation to display the actions of the positioning control system. It may not be clearly evident in Figure 5.1 at what locations the train stopped, but the evidence is shown in the velocity estimation plot for Trial 3. After close examination, the results from this plot show that the train stops between frames 9 and 10 and frame 20. Comparing this visual evidence with the data obtained from Trial 3, this is a true deduction. Other evidence is apparent when analyzing the distance between each data point in Figure 5.1. When the distance between two data points is large, the velocity at this instance is represented by a larger value and vise versa. This trend proves that the velocity controller is working properly.

5.1.4. Robustness to Different Lighting Conditions

A performance analysis is conducted to assess the performance of the software at two different lighting conditions. This limited range of tests can not be considered a complete and extensive study of the effect of different lighting conditions, but it presents the basic outcome of the performance characteristics of the software. This type of performance analysis is chosen because it is fairly easy to manipulate and it can be measured and quantified with the available lab equipment. The effect of changing lighting conditions is a very important design parameter. This is a very practical disturbance that may be encountered in MV applications on UAVs.

As a result of using a controlled laboratory setting for this research effort there are very few lighting conditions available. Therefore, only two conditions are chosen for the performance assessment. The first lighting condition is the brightest obtainable lighting condition, which is created by utilizing a 100% of the florescent lighting. The second lighting condition is obtained by reducing the florescent lighting by 50% and draping a net over the model environment to further reduce the light. Once the proper lighting conditions are determined for each condition, the EV value is recorded and calculations are made to quantify the different luminance values at each condition, which is shown in Table 5.11. In this table, the luminance value for the second test condition displays a 70% decrease when compared to the first test condition. The luminance value is also measured for a cloudy day to give a better idea of the amount of light that each experiment is subjected to. This value shows that natural light conditions are accounted for by a much larger luminance measurement; approximately 100% larger then the first test condition. It is believed that this increase in luminance would make it easier to detect the target by making it more distinguishable from the background. Finally, once the luminance values are recorded a detailed performance analysis is performed to assess the results obtained at each condition. This analysis includes a visual assessment of the software output for the position estimation performed for three tests.

Table 5.11: Luminance Measurements Obtained From the Canon EOS 400D

Luminance Measurements				
Lighting Test Condition	f-number	k	EV_{100}	L (cd/m^2)
#1	4	1/50	9.64	99.73
#2	4	1/13	7.70	29.86
Cloudy day at 11:35 A.M. on 11/18/2007	4	1/100	10.64	200.00

To visually interpret the difference between the two lighting conditions, Figure 5.11 displays two arbitrary images captured under each lighting condition with their respective luminance histogram diagram. Each histogram diagram displays the pixel intensity distribution throughout each image. After analyzing the diagrams, it is evident that the amount of luminance is reduced shown from the pixel intensity distributions. The distribution for the image captured under the first test condition is represented by an overexposed histogram plot where a large portion of the pixel intensity values are focused near the bright side of the contrast spectrum. In the other situation, the image captured under the second test condition represents a pixel intensity distribution that is focused near the middle of the contrast spectrum. The result of the second test condition will make it more difficult to detect the train in each simulation.

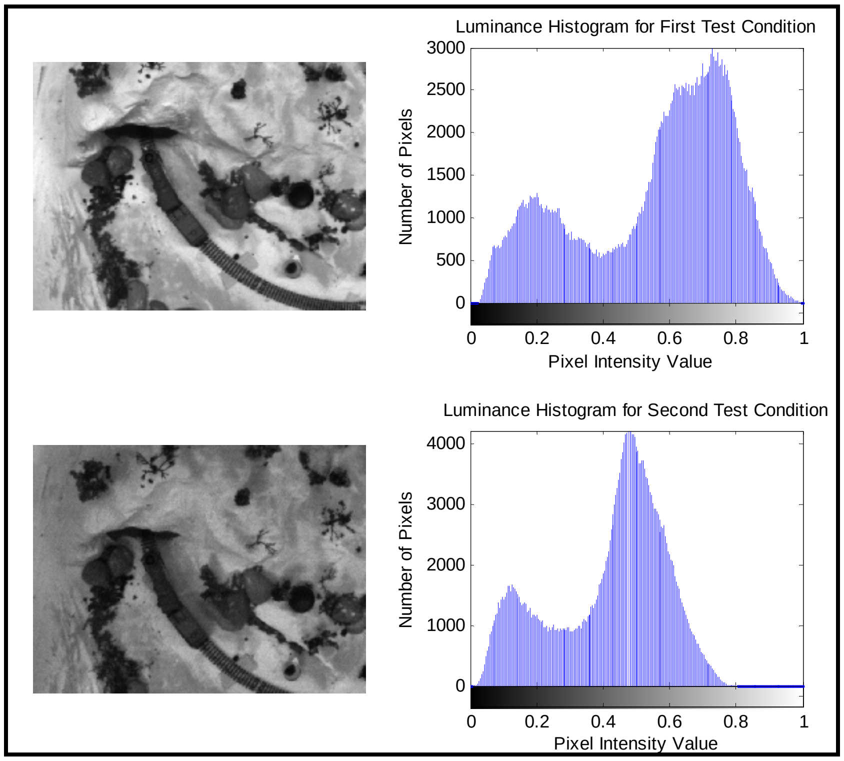


Figure 5.11: Luminance Histograms for Lighting Test Conditions

After performing several experiments under the second test condition there are a few parameter adjustments that are essential in order to produce acceptable results. It is necessary to adjust the contrast-stretching function that is used during the first lighting condition. Each function is shown in Figure 5.12, in which the blue line represents the contrast-stretching function used in the simulations under the first lighting condition and the red line represents the function used under the second lighting condition. The function representations are obtained by altering the intensity threshold value, *m*. An understanding of the procedures for this method is provided in Section 3.3.2. To produce acceptable results under the second lighting condition, it is necessary to alter the contrast-stretching function and increase the threshold value for Sobel edge detection.

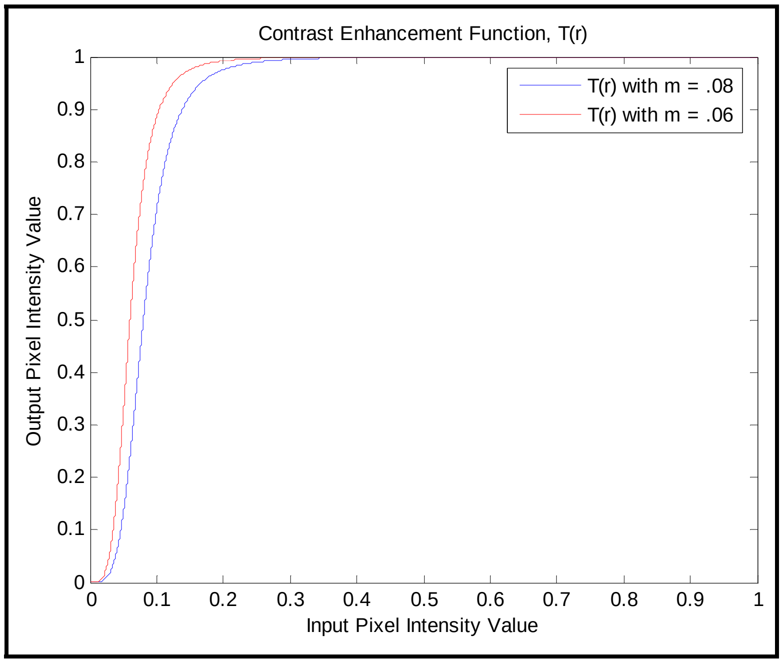


Figure 5.12: Contrast-Stretching Functions Used in Each Condition

The data used from the software to analyze its performance under different lighting conditions is based solely on the train position estimation results. The results shown in this section are obtained under the second test condition. It is not necessary to perform any other tests under the first lighting condition since these results are previously displayed in Section 5.1.2. The tests performed in this section consist of a 25 second simulation where a frame is captured once per second; this is performed for three tests. The results of each of these tests are shown in Figure 5.13, Figure 5.14, and Figure 5.15, respectively. The red line in each plot indicates the estimated position of the train detected from the software and the blue represents the “actual” position of the train. The first two figures include the error results during each test experiment. This is not necessary for the third test since it detected a target at every frame location.

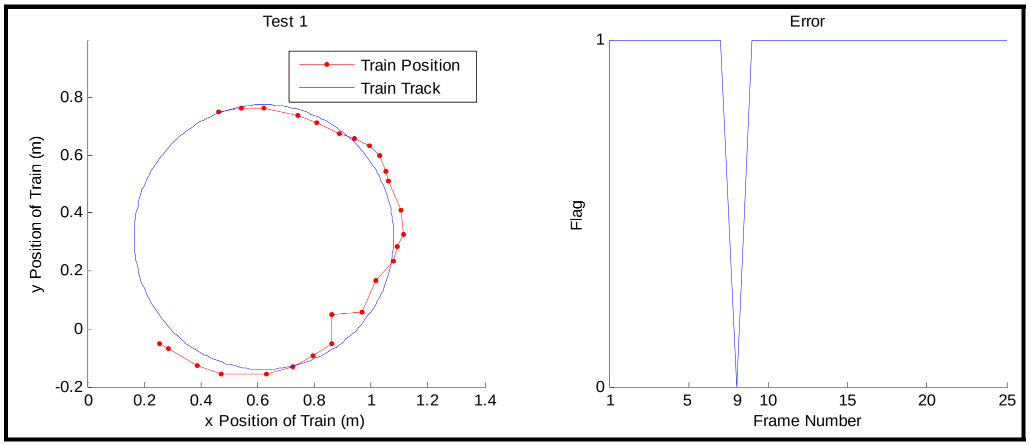


Figure 5.13: Position Estimation Results for Test 1 for Lighting Condition #2

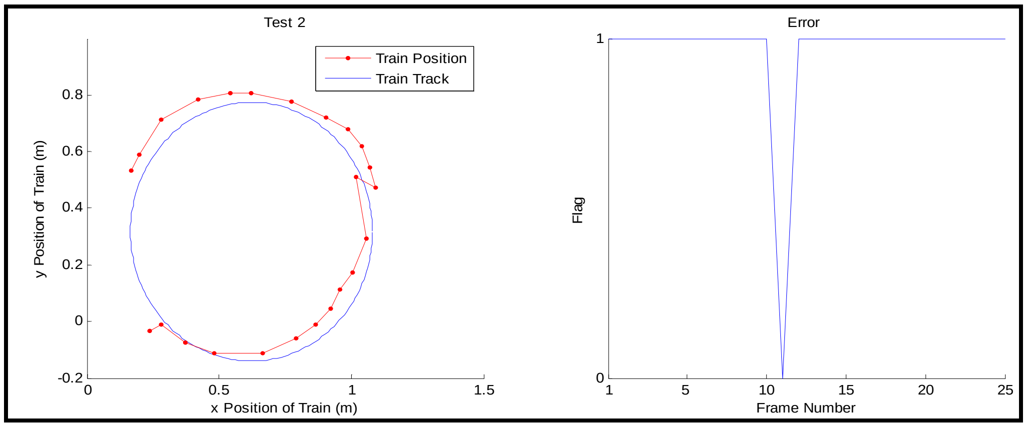


Figure 5.14: Position Estimation Results for Test 2 for Lighting Condition #2

As seen from the figures above, the positions estimation error did increase at arbitrary locations during each simulation. The result of this is evident from the error plots in each figure. Apparently the detection software is unable to detect the train at the frame number shown in each figure when the flag is equal to 0. However, this is not a huge issue due to the fact that the train is re-detected in the next frame. A combination of camera vibrations and reduced lighting is the leading cause of this error, making it harder to detect the edges of the train in each image. With the exception of this error, the data appears fairly consistent to the data obtained under the first lighting condition.

One last characteristic is determined while analyzing the results obtained from the third test, which is shown in Figure 5.15. After assessing the data obtained from the software the last captured frame in Test 3 resulted in a false detection, which is evident by the point offset from the train track. At this instance in the simulation the train entered the tunnel system and left the camera's FOV. In a normal situation the MV software should not detect and label any other features in the image as the target. However, since it is necessary to enlarge the threshold to account for the different lighting condition, it increased the chances of false detections.

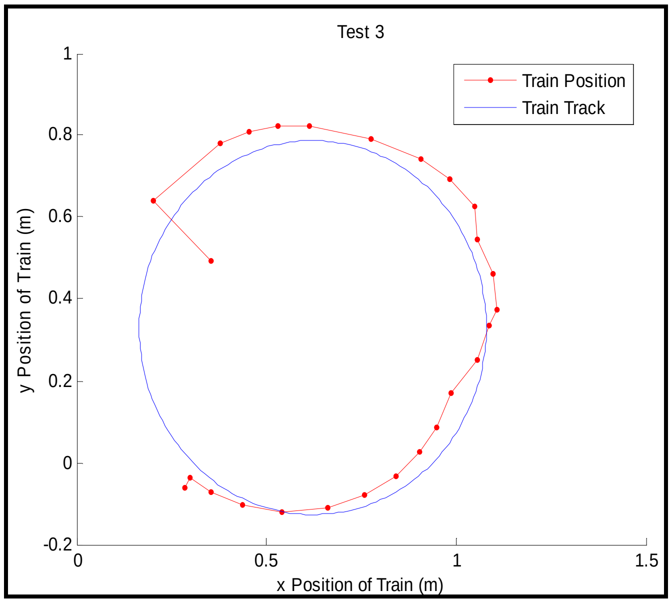


Figure 5.15: Position Estimation Result for Test 3 for Lighting Condition #2

Although more error is introduced when the lighting conditions are altered, it is not a significant amount and the detection software is still able to produce satisfactory results. If harsher lighting conditions are encountered it may be required to further enhance the detection software. To accomplish this, it may be necessary to add another constraint to the feature detection software, such as the Euclidean distance. In this case, a constraint would be placed on the distance between the estimated train position and the track. This would require more accurate position estimation to produce satisfactory results.

5.2. Target Detection and Tracking Using the HT Algorithm

To assess the performance of the HT for the circle detection procedure, normally there would be data obtained through a sensor, such as GPS, to compare the estimated position of the train against. However, in this research, the availability of such data is not obtainable. As a result, the analysis of this method is limited to the scope of tests that can be performed. This does not exclude a visual assessment of the performance of this software method, although it may be somewhat subjective. In the visual means of evaluation the output of the target detection and tracking software will be analyzed to determine if the software is working to the best of its capability. This form of evaluation will be covered in Section 5.2.1. One other aspect of this explored software method is its execution speed. This is always an important form of evaluation when dealing with MV applications. Therefore, a complete computational workload analysis is performed and discussed in Section 5.2.2. Table 5.12 displays the experimental breakdown for the computational workload of this software. Each test a 25 second simulation using a 1 Hz update frequency.

Table 5.12: Computational Workload Test Breakdown Used for Evaluation

Computational Workload Evaluation Via Matlab® Profiler
Test 1: 25 second simulation at 1 Hz.
Test 2: 25 second simulation at 1 Hz.
Test 3: 25 second simulation at 1 Hz.

Table 5.13 represents the experimental breakdown for the position estimation analysis. The first three tests are the results of a 25 second simulation while Test 4 is performed using a 50 second simulation.

Table 5.13: Position Estimation Analysis Breakdown Using the HT Method

Visual Analysis of Position Estimation
Test 1: 25 second simulation at 1 Hz; train travels CW.
Test 2: 25 second simulation at 1 Hz.; train travels CCW.
Test 3: 25 second simulation at 1 Hz; train travels CW then CCW
Test 4: 50 second simulation at 1 Hz; train travels CCW

5.2.1. Computational Workload Analysis

To evaluate the computational workload for this software method the Matlab® Profiler tool is utilized. The profiler is used to break down the time spent on each section of the software scheme. The section breakdown and their respective computation times are shown in Table 5.14. The description of each section is previously explained in Section 5.1.1.

Table 5.14: Timing Profile While Using the HT Method

Target Detection/Tracking Speed Comparison using *Matlab® Profiler*	Modified Detection and Tracking Algorithm	
Machine Vision Process	Time (s)	Executions
Capturing the Image Frame	2.30	26
Pre-processing	5.34	26
Circle Detection Method	15.05	26
Motor Control	11.77	26
Other Lines and Overhead	4.31	N/A
Total Time	38.78	N/A

The target detection and tracking simulations are performed for 25 seconds with a 1 Hz sampling rate, which is why the software is only executed 26 times. These tests are performed with the same sampling rate as the profiler test previously explained in Table 5.5. This is necessary to properly compare the two methods for circle detection in the MV software design.

The timing profile is broken down into the major subsystems of the Simulink® model. It is easy to see that the HT subsystem, taking 15.05 seconds, uses the most computation time compared to the other subsystems. This accounts for 38.8% of the total time required to perform the simulation. This characteristic is not surprising considering the number of calculations the HT method performs for a given simulation. It should be noted that this timing test is performed using the modified detection and tracking algorithm. If this test is performed using the enhanced software version the computation time for the HT subsystem would drastically increase since it is significantly affected by the number of edge pixels in the input image. Therefore, it is crucial to design an efficient pre-preprocessing stage to obtain the best possible signal to noise ratio. Comparing the results of this profiler test with the previous test, the other software sections are not affected by using the HT method. Computation time is approximately the same for these profiler tests. Therefore, since the rest of the subsystems have already been analyzed in the previous profiler test it is not necessary to further discuss their performance characteristics. Considering the results obtained in this profiler test, the frame rate is approximately 0.5 Hz. This frame rate would not be adequate enough to be utilized in a UAV for a real time application.

5.2.2. Performance Analysis

The estimated data obtained from the MV software is compared to the location of the train track within the model environment, which is obtained by measuring the location of the track. The following figures display the output of the target detection and tracking scheme using the HT method for circle detection. In each figure the red line indicates the estimated position measured through the MV process and the blue line represents the train track or the "actual" position of the train during the simulation. It is obvious that the software is working well as it detects and tracks the train at each location around the track during the simulation. This trend is displayed through each of the 4 trials shown below. It should be noted that the train is set at its lowest velocity setting to account for the increased processing time while using the HT method. Figure 5.16 represents the results obtained through three 25 second simulations while Figure 5.17 represents the result obtained through a 50 second simulation. The description of each test is provided below:

1. The train travels clockwise around the track;
2. The train travels counterclockwise around the track;
3. The train initially travels clockwise around the track, then stops and changes direction;
4. The train travels counterclockwise around the track.

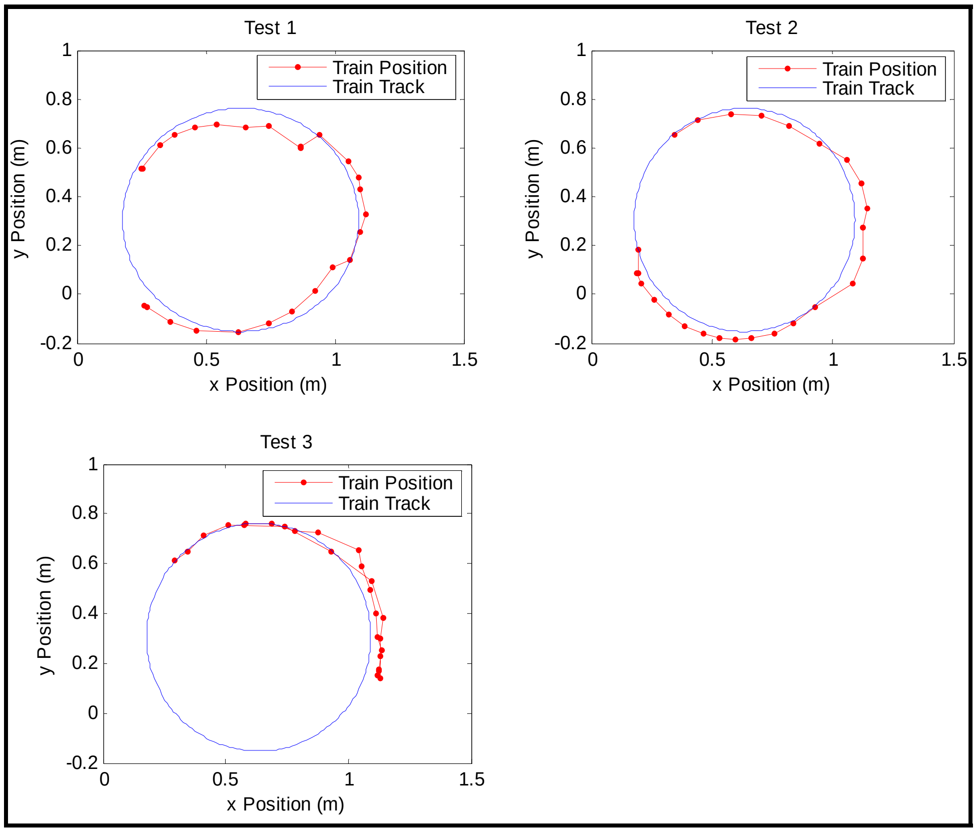


Figure 5.16: Position Estimation Result Using HT Method for 25 Second Simulation

The final representation of the position estimation is displayed in Figure 5.17. Similar to the previous representation, the red line represents the estimated train location and the blue line represents the "actual" train location. The simulation in Trial 4 differs from the other trials shown in Figure 5.16 for the fact that this simulation is performed for 50 seconds. This trial is used to analyze the error correction portion of the software, or display the actions of the camera positioning system when the train exits the cameras FOV, while using the HT method. In Figure 5.17 the starting and ending points for the simulation are labeled and the box represents the tunnel system in the model environment. As shown in the figure, the HT method successfully detects the train as it enters and exits the tunnel system.

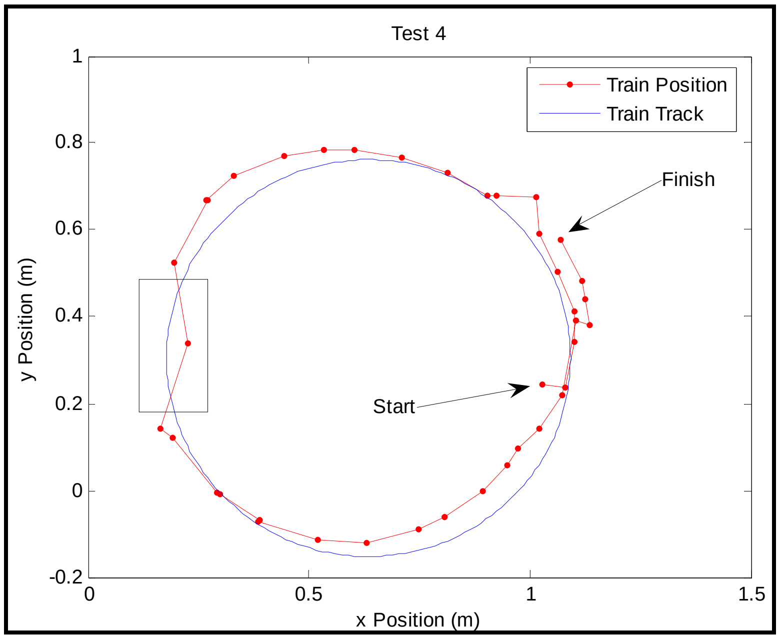


Figure 5.17: Position Estimation Result Using HT Method for 50 Second Simulation

These results obtained through this evaluation show a near flawless performance in detecting and tracking the train throughout each simulation. There are only three instances where the HT detection method falsely detects a "target," as shown in the results for Trial 1 and 4. With the results obtained using the HT method, ultimately, a UAV equipped with a control system would have no problem detecting the train, but the scheme would have to execute fast enough to properly track the target along its path. It is evident that when using this software method, accuracy would probably not be the cause of error. The source of error lies within its execution speed. Therefore, methods to increase the efficiency of the HT must be further researched before this MV technique can be utilized within a real time application for UAVs.

Chapter 6. Conclusion and Recommendations

6.1. Conclusion

The capability to shield a human pilot from extremely long or dangerous missions has significantly increased with the widespread deployment of UAVs. Traditionally, these platforms are used for intelligence, surveillance, and reconnaissance missions in military settings that pose high human risk. The trend of limiting human intervention within these applications has now become appealing to civil and commercial users as well. The popularity of these unmanned systems will continue to grow for a multitude of reasons as they not only eliminate the risk to humans, but reduce the error of direct human intervention, have the ability to cooperate with other UAVs, and are more cost effective. With these key concepts in mind, the goal of this research effort is to investigate the application of MV on UAV platforms. The application in this research consisted of the design, development, and the experimental validation of a prototype "UAV" surveillance system with MV capabilities for the purpose of target detection and tracking. At the completion of this research, the MV problems presented are satisfied. The mock "UAV" environment designed for this research, with its ability to test different MV algorithms, is consequential in meeting the need to show that this MV application is feasible and this type of technology is important to future UAV missions.

At the completion of performing a wide range of experiments within the prototype surveillance system, under different test conditions, the target detection and tracking software is shown to work very well. This is evident after analyzing the experimental results which show that the algorithms used for target detection exhibited a satisfactory behavior, having a 90% detection rate. However, after testing the two different approaches for circle detection, the FFF circle detection algorithm clearly outperforms the HT method as it repeatedly yields a much faster computation time. The RMS error is slightly better as well. The computation time of the software can be further reduced by performing a cropping routine on the input image to reduce the image size by 44%. Therefore, if further research is performed dealing with this application of MV it is suggested to utilize the FFF algorithm based on its speed and robustness.

During the tests performed in this research the target detection and tracking software managed to successfully detect and track the target in every case. The results of the position and velocity estimations are proven to be fairly accurate with the estimated RMS error being less then 0.0559 meters in the worst case scenario. The detection rate and accuracy of this software suggests that it can be reliable in much more diverse situations. An attempt to simulate these situations is made by assessing the software performance in two different lighting conditions. The lighting conditions are obtained by altering the amount of florescent lighting in the laboratory. The first lighting condition can be characterized by a luminance measurement of 99.73 cd/m^2 and the second condition recorded a measurement of 29.86 cd/m^2. This accounts for a luminance difference of approximately 70%. In both test conditions, the target is continuously detected and tracked, without significantly sacrificing its detection sensitivity at the lower luminance measurement. This is proven by analyzing the position estimation results where the detection rate is reduced from approximately 95% to 90%. The effect of altering the lighting condition is an indicator of the robustness of the software. This is important for the fact that this type of disturbance is almost guaranteed in a real life situation.

The target detection and tracking scheme exhibited satisfactory results at an update rate of [1-2] Hz. This value proved to be adequate within a laboratory environment and is currently believed to be satisfactory for operational effectiveness in UAV platforms. However, the update rate is based on the performance characteristics of the target and UAV; higher operational speeds require higher update rates to ensure the target stays within the camera's FOV. In these situations where higher update frequencies are necessary, further evaluation is required to decrease the computation time of the current software. Nevertheless, with increasing popularity of this MV application, powerful embedded systems are likely to be available in the near future. This could offer the ability to implement the methods presented in this research for target detection and tracking, at higher update rates.

6.2. Recommendations

The future of the prototype “UAV” surveillance system is certainly appealing for the analysis and implementation of different MV applications. If further research is envisioned regarding the topic of target detection and tracking, it is recommended to design a six DoF positioning system to better emulate the motions of a real aircraft. If this is accomplished, the first step would be to design a lateral controller to control the “UAVs” airspeed and turning rates, which is directly related to the aircraft’s orientation. Lateral control of the “UAV” could be achieved by commanding the vehicle’s turning rate to minimize the error between the desired heading angle and the current heading angle. The desired heading angle is directly related to the location of the target within the captured image frame. It is recommended to design a PID controller as a first attempt to accomplish this task.

In regards to the current software design, additional efforts could be made to increase the overall robustness of the software to make it more suitable for real life situations. Currently it is necessary to manually adjust the software parameters to obtain satisfactory results at different lighting conditions. Additional experiments could be performed to obtain a relationship between the measured luminance and the input parameters for the contrast-stretching function. Using this relationship, a mathematical model could be used to automatically determine the parameter inputs at each measured lighting condition. In addition, software could be written to enhance the error detection portion to better address the situation when the target leaves the camera’s FOV or when there is a false detection. In regards to this problem, it is recommended to form a relationship between the estimated target velocity and position to predict the location of the target within the environment. This prediction can be compared to the estimated values acquired through the MV camera. If there is a large amount of error between the two values then the estimated location could be the result of a false detection. This would further enhance the overall effectiveness of the target detection and tracking software.

Another issue that should be addressed, within the software design, is the situation when the target’s orientation is changed during the identification process. This can occur when either the train changes elevation or MV camera changes altitude. The current software design does not address this problem; the circle detection algorithms only search for features with a specific size and representation when the MV is positioned at a specific altitude. To resolve the problem, when the MV camera changes altitude, it is necessary to measure the amount of change through a sensor. With this information known the target representation can be scaled for the detection process. Therefore, if the MV camera is elevated further above the “ground” level then the software training data must be decreased in size. When the MV camera is lowered towards the “ground” level then the training data must be scaled larger. The other situation that can occur is when the target changes elevation. This is a more difficult problem to address because in a real situation it is highly unlikely that the actual position of the target will be known. Therefore, one solution would be to acquire the geographic elevation changes prior to the target detection and tracking process. If this information is known then the elevation of the target can be estimated based on the known location of the “UAV,” or MV camera, within the model environment. If the elevation of the target can be estimated then the training data for the target can be scaled accordingly.

Another problem that could occur is when the shape of the train’s chimney stack is not represented by a circle. This could happen when the MV camera is not directly over the target, which could cause the chimney stack to be represented by an ellipse. The current software does an adequate job of detecting the chimney stack when it is not directly over the target. However, the orientation of the train is not significantly altered during the detection process and the software allows for some error to be accounted for. If the orientation of the chimney stack is significantly altered, a possible solution to this problem would be to use an ellipse detection algorithm rather then a circle detection algorithm. This could then be used to detect the chimney stack when it is represented by an ellipse or a circle where a circle is a special case of an ellipse. Additional research should be completed for different ellipse detection methods if this approach is necessary.

Real-time target detection and tracking is an essential part of the vision based control of UAVs. The current software design is capable of operating at approximately 1 Hz, which is achieved using a high powered desktop computer. This is not sufficient enough to implement on-board a real UAV. In order to perform vision processing on-board a UAV, real-time capabilities must be achieved. To obtain this capability, the first recommendation would be to implement the software design in a different programming language. It is recommended to use a higher level programming language such as C or C++ for real-time system development. Another concerning issue involves the current operating software. For real-time computing, it is recommended to use a different operating system; Microsoft Windows operating system is not suitable for this task. For example, many delays in the software computation time are accounted for by accessing the MV camera through Windows. A possible solution to this is to use operating software that is more capable of real-time computing, such as the Linux operating environment. Real-time capabilities can lead to future experimental tests of this vision-based tracking system onboard an actual UAV.

References

1. Pike, John. "Unmanned Aerial Vehicles." FAS Intelligent Resource Program." http://www.fas.org/irp/program/collect/uav.htm. 2007 October 12.

2. Krock, Lexi. "Time Line of UAVs." Nova Science Programming on Air and Online. http://www.pbs.org/wgbh/nova/spiesfly/uavs.html. November 2002.

3. Bone, Elizabeth, Bolkcom, Christopher. "Unmanned Aerial Vehicles: Background and Issues for Congress." Report for Congress: RL318772. 2003, April 25.

4. Zuech, Nello. "Machine Vision Concepts" Applying Machine Vision (1988).

5. "Unmanned Aerial Vehicles (UAVs)." Army. Oct 2004. FindArticles.com. 04 Dec. 2007. http://findarticles.com/p/articles/mi_qa3723/is_200410?pnum=3 &opg= n9446288.

6. Fulghum, David A., Wall, Robert. "Israel Starts Reexamining Military Missions and Technology." Aviations Week. http://www.aviationweek.com/aw/generic/ story_generic.jsp?channel=awst&id=news/aw082106p2.xml. August 20, 2006.

7. "Predator RQ-1/MQ-1/MQ-9 Reaper – Unmanned Aerial Vehicle (UAV), USA." U.S. Airforce. http://www.airforce-technology.com/projects/predator/. 2007.

8. Wise, J. "Unmanned Planes, Already Critical to the Military" Popular Mechanics. April 2007.

9. Wise, J. "Unmanned Planes, Already Critical to the Military" Popular Mechanics. April 2007.

10. Davies, E.R. "Edge Detection" Machine Vision Theory, Algorithms, Practicalities. San Francisco: 2005.

11. Parker, J. R. Algorithms for Image Processing and Computer Vision. New York 1997.

12. Agapakis, J.E. "The Future of Machine Vision" *Quality Digest* http://www.qualitydigest.com/oct98/html/machfutr.html 1998.

13. McGarry, E.J. "An Outlook for Machine Vision" Machine Vision Online. 8 April 2007, 22:33 http://www.machinevisiononline.org/public/articles/ articlesdetails.cfm?id=1134.

14 . “Segmentation (image processing)” Wikipedia The Free Encyclopedia. 12 March 2007, 5:00 http://en.wikipedia.org/wiki/Segmentation_(image_processing).

15. “Edge Detection” Wikipedia The Free Encyclopedia. 30 March 2007, 15:41 http://en.wikipedia.org/wiki/Edge_detection.

16. Rosenfeld, A., Avinash C. Kak. “Edge Detection” Digital Picture Processing. New York: 1976. 275-294.

17. Kumar, A., Narendra Ahuja, John Hart “A Vision System for Monitoring Intermodal Freight Trains.” Applications of Computer Vision, 2007. WACV ’07. IEEE Workshop. Feb 2007.

18. “Hough Transform” Wikipedia The Free Encyclopedia. 13 April 2007, 14:05 http://en.wikipedia.org/wiki/Hough_transform.

19. Fisher, R., S. Perkins, A. Walker and E. Wolfart. Hough Transform. 2003. http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm.

20. Kak, Avinash C., Rosenfeld, Azriel. Digital Picture Processing. New York: 1976.

21. Camapum, Juliana F., Fisher, Mark H. “Segmentation using Spatial-Feature Clustering from Image Sequences.” Image Processing, 1998. ICIP 98. Proceedings 1998. Coventry UK: 1998.

22. Ali, Raza, Usman Ghani , Aasim Saeed. “Data Clustering and its Applications” 14 April 2007 http://dms.irb.hr/tutorial/tut_clustering_short.php.

23. Jin, Hailin, Yessi, Anthony J., Soatto, Stefano. “Region-based Segmentation on Evolving Surfaces with Application to 3D Reconstruction of Shape and Piecewise Contrast Radiance.” Computer Vision, ECCV 2004: 8th European Conference on Computer Vision.” 2004.

24. Salih, Qussay A., Ramli, Abdul Rahman. “Redion Based Segmentation Technique and Algorithms for 3D Image.” International Symposium on Signal Processing and its Applications (ISSPA) Kuala Lumpur, Malaysia: 13-16 August, 2001.

25. Paragois, Nikos. Deriche, Rachid. “Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects.” IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 22, No. 3. March 2000.

26. Kontitsis, M., Kimon P. Valavanis, N. Tsourveloudis. “A UAV Vision System for Airborne Surveillance.” Proceedings of the 2004 IEEE International Conference on Robotics & Automation. New Orleans, LA April: 2004.

27. Narli, Vefa, Paul Y. Oh. “A Hardware-in-the-Loop Test Rig for Designing Near-Earth Aerial Robotics.” Proceedings of the 2006 IEEE International Conference on Robotics and Automation. Orlando, Florida: May 2006.

28. Seanor, Brad, Giampiero Campa, Yu Gu, Marcello Napolitano, Larry Rowe, Mario Perhinschi. “Formation Flight Test Results for UAV Research Aircraft Models.” AIAA 1st Intelligent Systems Technical Conference. Chicago, Illinois: 20-22 September 2004.

29. Pollini, Lorenzo, Roberto Mati, Mario Innocenti, Giampiero Campa, Marcello Napolitano. “A Synthetic Environment for Simulation of Vision-Based Formation Flight.” AIAA Modeling and Simulation Technologies Conference and Exhibit. Austin, Texas: 11-14 August 2003.

30. Johnson, Eric N., Anthony J. Calise, Ramachandra Sattigeri, Yoko Watanabe, Venkatesh Madyastha. “Approaches to Vision-Based Formation Control.” 43rd IEEE Conference on Decision and Control. Atlantis, Paradise Island, Bahamas: 14-17 December 2004.

31. Sattigeri, Ramachandra, Anthony J. Calise, Johnny H. Evers. “An Adaptive Approach to Vision-Based Formation Control.” American Institute of Aeronautics and Astronautics.

32 Proctor, Alison A., Eric N. Johnson. “Vision-Only Aircraft Flight Control Methods and Test Results.” American Institute of Aeronautics and Astronautics. 2004.

33. Rowe II, Larry W. Machine Vision Applications in UAVs for Autonomous Aerial Refueling and Runway Detection. 2006.

34. Frew, Eric, Time McGee, ZuWhan Kim, Xiao Xiao, Stephen Jackson, Michael Morimoto, Sivakumar Rathinam, Jose Padial, Raja Senjupta. “Vision-Based Road-Following Using a Small Autonomous Aircraft.” 2004 IEEE Aerospace Conference Proceedings. 5 January 2004.

35. Ledda, Alessandro, Hiep Q. Loung, Wilfried Philips, Valerie De Witte, and Etienne E. Kerre. Advanced Concepts for Intelligent Vision System. “Grayscale Image Interpolation Using Mathematical Morphology.” Belgium: September 2006. 78-90.

36. "Star SAFIRE^{TM} HD" *FLIR Systems*. www.cvs.flir.com/lawenforcement/products/starSAFIREHD.cfm.

37. Yip, Raymond K. K., Dennis N. K. Leung, Stephen O. Harrold. "Line Segment Patterns Hough Transform for Circles Detection Using a 2-Dimensional Array."

38. Guo, Si-Yu, Xu-Fang Zhang, Fan Zhang. "Adaptive Randomized Hough Transform for Circle Detection Using Moving Window." Proceedings of the Fifth International Conference on Machine Learning and Cybernetics. Dalian: 13-16 August 2006.

39. Aleixos, N., J. Blasco, E. Molto, F. Navarron. "Assessment of Citrus Fruit Quality Using a Real-time Vision System." Pattern Recognition, 2000. Proceedings 15th International Conference. 3-7 September 2000.

40. Fang, Cheng, Ying Yi-bin. "Machine Vision Inspection of Rice Seed Based on Hough Transform." Journal of Zhejiang University Science. China: 3 December 2003.

41. Morris, Tim, P. Blenkhorn, L. Crossey, Q. Ngo, M. Ross, D. Werner, C. Wong. "Clearspeech: A Display Reader for the Visually Handicapped." IEEE Transactions on Neural Systems and Rehabilitation Engineering. December 2006.

42. Harlow, C. A., S. J. Dwyer III, G. Lodwick. "On Radiographic Image Analysis." Digital Picture Analysis. Vol 11. Germany: 1976.

43. Godbout, Andrew. "Segmentation Methods Applicable to Segmenting the Moving Organ in a CT Scan." Saint Mary's University. 2003.

44. Tezmol, Abraham, Hamed Sari-Sarraf, Sunanda Mitra, Rodney Long, Arunkumar Gururanjan. "Customized Hough Transform for Robust Segmentation of Cervical Vertebrae from X-Ray Images." Fifth IEEE Southwest Symposium on Image Analysis and Interpretation. 2002.

45. Tian, Qi-Chuan, Q. Pan, Y.M. Cheng, Q.X. Gao, "Fast Algorithm and Application of Hough Transform in Iris Segmentation." Proceedings of the Third International Conference on Machine Learning and Cybernetics. Shanghai. 26-29 August 2004.

46. Steward, B.L., L.F. Tian. "Real-Time Machine Vision Weed-Sensing." ASME Paper No. 98-7006. St. Joseph, MI: ASME.

47. Gonzales, Rafael C., Richard E. Woods. Digital Image Processing. 2002.

48. Gonzalez, Rafael C., Richard E. Woods, Steven L. Eddins. Digital Image Processing Using Matlab. 2002. 83.

49. "Video and Image Processing Blockset." The Mathworks http://www.mathworks.com/access/helpdesk/help/toolbox/images/index.html?/access/helpdesk/help/toolbox/images/f1413543.html&http://www.google.com/search?hl=en&q=rgb+to+intensity%2C+matlab.

50. Yi, Wei. "A Fast Fitting and Finding Algorithm to Detect Cirlces." Geoscience and Remote Sensing Symposium Proceedgins, 1998. IGARSS '98. 1998 IEEE International. Chnia: 1998.

51. Ayyub, Bilal M., McCuen, Richard H. Probability, Statistics, and Reliability for Engineers and Scientists. 2nd Edition. Florida: 2003.

52. Moore, David S., McCabe, George P. Introduction to the Practice of Statistics. 5th Edition. New York: 2004.

53. "Front Side Bus." Wikipedia The Free Encyclopedia. 10 November 2007, 9:03 http://en.wikipedia.org/wiki/Front_side_bus.

54. Ali, Razak M. "DDR2 SDRAM Interfaces For Next-Gen Systems." Electronic Engineering Times. http://www.eetasia.com/ARTICLES/2006OCT/PDF/EEOL_2006OCT16_ INTD_STOR_TA.pdf. 2006 October 16.

55. "Serial ATA: High Speed Serialized AT Attachment Revision 1.0." Serial ATA Working Group.

56. "Stepper Motor Basics." AMS Advanced Motor Systems, Inc. www.ams2000.com. 2000 July 7.

57. "Data Acquisition Toolbox." The Mathworks. http://www.mathworks.com/products/daq/.

58. Gabele, Helke, Wuller, Dietmar. "The Usage of Digital Cameras as Luminance Meters." SPIE-IS&T/ Vol. 6502 65020U-11. Germany: 2007.

59. ISO 2721: 1982, Photography – Cameras – Automatic controls of exposure.

60. "Standard Simulink Blockset." The Mathworks. www.mathworks.com/access/helpdesk/help/toolbox/simulink/.

61. "Image Acquisition Toolbox." The Mathworks. www.mathworks.com/products/imaq/description1.html.

62. "Image Processing Toolbox." The Mathworks. http://www.mathworks.com/products/image/.