M.Sc. Thesis report
Masters in Electronics Design, 30hp

Vision Sensor
Distance & Angle

Prasanna Kumar P
Abstract

The objective of the thesis is to design a real time machine vision system using a smart camera to determine the distance and viewing angle from a target board. A machine vision algorithm is designed and, by calibrating that system to a target board, it is possible to measure the distance and viewing angle. The smart camera is a Matrox IrisGT running Windows XP Embedded. Image pre-processing, labelling, feature extraction and a triangulation technique are used in the algorithm in relation to the measurement. By using the Visual C++ library and Matrox imaging library this application is developed and deployed in the smart camera. At a later stage, a serial communication is established between the smart camera and a PC and the results of the distance and angle are shown using a distance scale and angular meter in LabVIEW. The application is analysed at different resolutions to check the frame rate and power consumption of the smart camera for the application, also the accuracy of position measurement and latency of the system is measured.

Keywords: Machine vision, smart camera, Matrox IrisGT, triangulation, LabVIEW.
Acknowledgements

I would like to express my sincere thanks to my supervisor Dr. Benny Thörnberg for his patience and tireless help throughout my thesis work. I would also like to thank Abdul Waheed Malik and my colleague Qaiser Anwar for their help and support during my work. Finally I would like to thank my family & friends for tolerating my intolerance and encouraging me in my life.
# Table of Contents

Abstract ............................................................................................................... ii  
Acknowledgements ........................................................................................ iii  
Terminology ..................................................................................................... vi  
Acronyms vi  

1 Introduction ............................................................................................ 1  
1.1 Background ............................................................................................. 1  
1.2 Overall aim .............................................................................................. 1  
1.3 Verifiable goals ....................................................................................... 2  
1.4 Contribution ............................................................................................ 2  
1.5 Outline ..................................................................................................... 2  

2 Related work ........................................................................................... 5  

3 Methodology ........................................................................................... 7  
3.1 Distance Measurement .......................................................................... 7  
3.2 Viewing Angle Measurement .............................................................. 9  
3.3 Experimental evaluation ..................................................................... 10  
3.3.1 Frame speed vs Resolution ................................................................. 11  
3.3.2 Power vs Resolution ............................................................................ 13  
3.3.3 Accuracy of position measurement ................................................... 13  
3.3.4 Latency of the system .......................................................................... 14  

4 Theory .................................................................................................... 15  
4.1 Develop an application and deploying in Iris GT........................... 16  
4.2 Powering Iris GT & connecting the I/O pins .................................... 17  

5 Design .................................................................................................... 19  
5.1 Algorithm .............................................................................................. 19  
5.2 Implementation .................................................................................... 20  
5.2.1 Continuous Image Acquisition .......................................................... 21  
5.2.2 Segmentation ........................................................................................ 21  
5.2.3 Morphological Post Processing .......................................................... 22  
5.2.4 Component Labeling ........................................................................... 23  
5.2.5 Feature Extraction ................................................................................ 24  
5.2.6 Calculation .......................................................................................... 24  
5.2.7 Displaying Results in Serial Monitor ................................................ 25  

6 Results .................................................................................................... 31  
6.1 Position of the camera ......................................................................... 31  
6.2 Evaluation results.................................................................................... 34
6.2.1 Frame speed vs Resolution ................................................................. 34
6.2.2 Power vs Resolution ............................................................................ 35
6.2.3 Accuracy of position of measurement .............................................. 36
6.2.4 Latency of the system .......................................................................... 37

7 Discussion .............................................................................................. 39

8 Conclusions ........................................................................................... 41

References ......................................................................................................... 43

Appendix A: Documentation of developed program code ......................... 45
Terminology

Acronyms

CCD           Charge Coupled Device
VGA           Video Graphics Array
USB           Universal Serial Bus
Digital I/O    Digital Input / Output
RS 232        Recommended Standard no 232
RGB           Red Green Blue
MS DOS         Microsoft Disk Operating System
FPS            Frames Per Second
LIDAR         Light Detection And Ranging
TOF           Time Of Flight
TDC           Time to Digital Converter
APD           Avalanche Photo Diode.
1 Introduction

1.1 Background

The real time machine vision system is used to acquire images, process them in order to retrieve information and use the information to design an application. The process of processing the image and retrieving information is known as image processing. An example of a real time machine vision system is a traffic camera where the number plates of vehicles can be recognized and in this case the images are captured and the information i.e., numbers are retrieved from the images. In our application images of a target board with known reference points are grabbed and these are processed in order to determine the distance between the camera and the target board and, in addition, the viewing angle from the target board.

The smart camera used is the Matrox Iris GT 1900c/x* series. These smart cameras are commercially available and they are mainly used in real time machine vision systems. The Iris GT is inbuilt with Windows XP Embedded. The commercial smart cameras are easy to use in relation to building an application, but require a development computer for creating the application. The application is created in Visual C++, by using the standard C++ library and the Matrox imaging library on the development computer.

1.2 Overall aim

The project’s overall aim is to determine the distance and viewing angle from a calibrated board to the camera. A real time video processing method is developed to some extent when the camera is moved towards the left side, right side and in front of the target board and thus enabling the distance and viewing angle of the camera to be calculated. At a later stage, these results are transferred serially to a PC and displayed in both a measuring scale and an angular scale using LabVIEW software. In addition, the processed real time video frames should be displayed in the Iris screen. As this completes the real time video processing, the next to important factors for consideration are the frame rate and the power dissipation. The video processing in this application involves applying image processing to every single frame captured by the camera. Since this process involves the image processing, the more illumination that there is to the board greater will be the reduction in the
number of processing steps. In addition, the external noise that is caused by the environment must be reduced so that the image processing steps do not become too complicated. The camera used in this project is the smart camera Matrox Iris GT, which is a commercial smart camera used for designing machine vision systems. A study is conducted of the Iris GT and Matrox Imaging software installation on the development PC as is developing and deploying an application using the camera.

### 1.3 Verifiable goals

The main goal of this project is to develop an optical measurement system capable of measuring the distance and angle to a reference pattern attached to a target board. A mathematical algorithm for calculating the angle and distance has been previously developed as has been a Matlab script for calculating the distance and angle for one single image. The developed algorithm and script have been thoroughly studied and, at a later stage, using the mathematical algorithm and Matlab script an application is developed in the smart camera. The application is developed in Visual C++ using the standard C++ library and the MIL Library provided by Matrox with the smart camera. Since it is a real-time system, the important factors which must be considered are the frame rate and power. The frame rate and power consumed by the camera for this application must be measured and the factors for improvement are required to be verified. In addition, the accuracy of the position calculated by using the mathematical algorithm must be manually verified. Thus an overall performance and latency of the system is studied.

### 1.4 Contribution

The triangulation method for finding the distance and angle was provided by the author’s supervisor Benny Thörnberg, who also provided a Matlab script for determining the distance and angle from an image. The Visual C++ script using the Matrox imaging library for a real-time system was designed by the author. Additionally a serial communication was established between the camera and a PC in order to view the results using LabVIEW software.

### 1.5 Outline

In Chapter 2 the related works to this project are explained.
Chapter 3 states the triangulation technique used to determine the distance and angle and, in addition, the experimental evaluation.

In chapter 4 the theory in relation to the smart camera and Matrox breakout board will be illustrated.

In chapter 5 the design of the entire system and means of implementation in Visual C++ and LabVIEW are discussed.

In chapter 6 the system results and experimental evaluation are presented.

In chapter 7 there are various discussions concerning the experimental evaluation.

Chapter 8 the conclusion relating to the thesis work is stated.
2 Related work

In recent years, machine vision applications have increased at a phenomenal rate. Applications such as traffic security system, speed control system, navigation system, crop monitoring and many more have had quite an impact. General machine vision approaches and their limitations are discussed in [15].

Shang Wenqin describes navigation systems for the blind, which involves extracting the road region and detecting obstacles on a road, and measuring the distance between the blind person and the obstacles [13]. It is important to detect an object and to calculate the centre of gravity of the object in an image and Abdul Waheed Malik in [12] presents a method using a work flow of the software system to do just this.

The position of the object is calculated using the camera model in a navigation system used in aircrafts [14]. The camera model could be an alternative to the triangulation method for calculating a position but, designing the camera model is more complex than the triangulation used in this project.

Min Gu Lee and his team developed a 3D LIDAR sensor named KIDAR-B25, which is a 3D Light detection and ranging sensor, to measure 3D image information with high range accuracy, high speed and a compact size. For measuring the distance to the target object they developed a range measurement unit which is implemented by TOF using a TDC chip, Si APD receiver and a pulse laser transmitter as an illumination source [16].

For robots, a stereo matched feature that is located in the coordinate system is used as a landmark for navigation. Jorgen, Fredrick and Lars Asplund developed a method to calculate the location of a landmark by using stereo matched features [17].

Junichi Ido developed an indoor navigation for a humanoid robot using a view sequence and for navigation they deploy a laser range sensor mounted over the head and obtain 3D information. They place a landmark in a known position and use the template matching method [18].
3 Methodology

The distance and angle can be calculated with the assistance of known reference points, the focal length of the camera and the distance in the image plane. The target board is selected such that it contains six objects (as shown in figure 3.1). The reference points are placed in the target board in such a way that they are able to achieve an orthogonal projection (horizontal & vertical distance between each of the pairs are the same).

Fig 3.1 – Six circular objects in the black target board

3.1 Distance Measurement

The target board is placed in front of the camera, as shown in fig 3.2, and before attempting a distance measurement, the objects in the image should be sorted as shown in the ‘Sort Matrix’ below, as this is more convenient for measurements. Then the distances between object 1 & object 2, object3 & object4, object5 & object6 are calculated and these are named as the vertical distances 1, 2, 3 respectively.

Sort = [object 1  object 3  object5

         object 2  object4  object6]
In fig 3.2, the point 'q' (object 1&2) and 'w' (object 3&4) are four objects in the board i.e., on the world coordinate system. In this case, 'a' is the distance in the image plane i.e., on the image coordinate system. The line 'xw' is the optic axis of the camera or the z coordinate of the camera to the object 3&4. The line 'qy' is another projection of object 1&2 to the image plane. The triangle pqw is formed by the projections beyond the target board.

\[
\begin{align*}
\text{where,} \\
R & = \text{distance between the camera & target board} \\
A & = \text{distance between objects in the target board} \\
a & = \text{distance in the image plane} \\
f & = \text{focal length of the lens} \\
r_{d} & = \text{distance between the focal points z & q} \\
d_{A} & = \text{distance between focal points z & p} \\
s & = \text{distance between p&q} \\
\text{angle } zxy & = \text{angle wpq} = 90\text{degree} \\
\text{angle pqw} & = \Phi = \text{viewing angle} \\
\text{angle pqw} & = \beta \\
\end{align*}
\]

\[\beta = 90 - \Phi\]
\[ \cos \beta = \cos(90 - \Phi) = \sin \Phi \quad ------ 1 \]

\[ \cos \beta = \frac{s}{A} = \sin \Phi \quad ------ \]

\[ s = A \sin \Phi \quad \text{-------- 2} \]

\[ wp = A \sin \beta \]

\[ wp = A \sin(90 - \Phi) = -A \sin(\Phi - 90) = A \cos \Phi \]

\[ wp = A \cos \Phi \]

where the length \( dA = R + wp = R + A \cos \Phi \)

\[ dA = R + A \cos \Phi \quad \text{-------- 3} \]

\[ \frac{\alpha}{dA} = \frac{a}{f} \quad \text{-------- 4} \]

Substituting equations 2 & 3 in 4

\[ \frac{A \sin \Phi}{R + A \cos \Phi} = \frac{a}{f} \quad \text{-------- 5} \]

At \( \Phi = 90 \) equation 5 becomes

\[ \frac{A}{R} = \frac{a}{f} \]

\[ R = \frac{fA}{a} \]

where \( R \) is the distance between the camera and the target board. By using the above equation, the distance between the target board and the camera can be calculated. This can be generally stated as the equation shown below.

\[ \text{Distance} = \frac{\text{Focal length} \times \left( \frac{\text{Reference point in real world}}{\text{Euclidean distance in image plane}} \right)}{a} \]

3.2 Viewing Angle Measurement

In figure-3.2 \( \Phi \) is the viewing angle of the camera to the board. From equation 5,

\[ \frac{A \sin \Phi}{R + A \cos \Phi} = \frac{a}{f} \]
\[ Af \sin \theta = aR + aA \cos \theta \]

\[ aR = Af \sin \theta - aA \cos \theta \]

\[ aR = -A(f \sin \theta + a \cos \theta) \quad \text{------- 6} \]

In equation 6 by using trigonometric identities [8],

\[ f \sin \theta + a \cos \theta = \sqrt{a^2 + f^2} \cos \left( \theta + \tan^{-1} \left( \frac{a}{f} \right) \right) \]

So,

\[ f \sin \theta + a \cos \theta = \sqrt{a^2 + f^2} \cos \left( \theta + \tan^{-1} \left( \frac{a}{f} \right) \right) \]

\[ = \left( \sqrt{a^2 + f^2} \right) \cos \left( \theta + \tan^{-1} \left( \frac{a}{f} \right) \right) \]

\[ f \sin \theta + a \cos \theta = -\sin \theta \left( \sqrt{a^2 + f^2} \right) \quad \text{------- 7} \]

Substitute equation 7 in 6

\[ aR = -A(-\sin \theta \left( \sqrt{a^2 + f^2} \right)) \]

\[ aR = A(\sin \theta \left( \sqrt{a^2 + f^2} \right)) \]

\[ \theta = \sin^{-1} \left( \frac{aR}{A\sqrt{a^2 + f^2}} \right) \]

\( \theta \) is the viewing angle from the camera to the target board. The viewing angle is calculated by using additional information regarding the distance between the camera and target board. This can be generally stated as the following equation.

\[ \text{Angle} = \sin^{-1} \left( \frac{\text{Euclidean distance in image} \times \text{Distance}}{\text{Reference point in real world} \times \sqrt{\text{Euclidean distance}^2 + \text{focal length}^2}} \right) \]

### 3.3 Experimental evaluation

The system should be evaluated by performing four main evaluation techniques, which are

- Frame speed vs Resolution
• Power vs Resolution
• Accuracy of position measurement.
• Latency of the system

The terms for measuring frame rate and the terms used in latency measurement such as frame count, frame missed and timer function are discussed in the following section.

**Frame Count**
The Mil library function M_PROCESS_FRAME_COUNT is used for this process and it returns the number of frames grabbed in sequence.

**Frame Rate**
The function M_PROCESS_FRAME_RATE returns the average of camera’s rate and the total amount of time to process in frames per second.

**Frame Missed**
The Mil library function M_PROCESS_FRAME_MISSED returns number of frames missed while computing the distance and angle function. This value is actually calculated as a difference between the frame rate and the number of grabbed frames.

**Application Timer**
The function MAppTimer is used for benchmarking operations, M_TIMER_RESET basically resets the timer or clock to zero. M_TIMER_READ reads the time in seconds since last reset.

### 3.3.1 Frame speed vs Resolution
The Iris GT has an effective pixel resolution of 1600 x 1200 and the maximum supported frame rate at this resolution is 15 frames per second. The distance and angle technique is introduced as a function and this function is applied to every frame of the Iris GT that has been grabbed. At this point, the frame rate at the output video, after calculating the distance and angle, will not be the same as the supported 15fps. An improved better real time video processing is achieved by having higher frame rates; the two techniques are implemented in order to analyse the improved frame rate. The first one is grabbing (cropping) a part of the image from the whole effective resolution which has been grabbed by the camera. The next involves resizing the whole resolution
to the desired size and applying the algorithm. This analysis is performed at different resolutions such as 640 x 480, 768 x 576 and 1600 x 1200.

**Cropping**

The camera will send the whole frame to its frame grabber, but by using Mil functions only part of the image is grabbed and processed. For example, if a 640 x 480 resolution is used, then it crops the first 640 x 480 pixels from the whole 1600 x 1200.

![Fig 3.3 – Cropping method](image)

This cropping method is performed at three different resolutions, namely 640 x 480, 768 x 576 and in 1600 x 1200. After applying the image processing algorithm to all the cropped frames, the frame rate can be observed. The most important term in cropping the image is the pixel size for the image, which at the different resolutions will never change. The pixel size remains the same at 4.4µm x 4.4µm for all the cases.

**Resizing**

The camera sends the whole frame to its frame grabber at an effective resolution and later, this whole resolution is resized into different resolutions. The nearest neighbour interpolation technique is used for resizing, where each position take the value of the nearest neighbour pixel. When resizing the image, the pixel size for the different resolutions will change. To calculate the pixel size the following formula is used
The image size (diagonal) in the sensor is 8.923mm in all cases, but the horizontal and vertical size change based on the selected resolution. The frame rate is observed for the resized frames.

In fig 3.4 the resizing method is described, when the effective resolution is resized into 640 x 480 and the pixel spacing (pixel distance) is changed from 4.4µm x 4.4 µm to 11.1 µm x 11.1 µm and the pixel spacing is calculated by means of the pixel size formula.

3.3.2 Power vs Resolution
The power consumed by Iris GT for running the distance and angle application is measured. The power is measured across the pin J15 & J17 in the breakout board using a digital multimeter, the details regarding the pin will be explained in the following chapter. The power is measured in three different resolutions and for both the cases involving cropping and resizing.

3.3.3 Accuracy of position measurement
The calculated distance and angle must be verified manually, regardless of cost, since the accuracy of the measurement is very important. Thus, the system is firstly placed at 90 degrees to the target board by following the right angled triangle rule and the distance is measured at various positions both manually and theoretically in LabVIEW. In addition, the accuracy of the angle is measured manually and by using the Iris GT.

\[
\text{Pixel size} = \frac{\text{image size (diagonal)}}{\sqrt{\text{horizontal size}^2 + \text{vertical size}^2}} \quad \text{eqn 2.3.1}
\]
3.3.4 Latency of the system

The latency is the measure of response time of the system for running an application or it is the time between the activity initiation and the viewing of results. In Iris GT the latency due to grabbing frame and latency due to computation of the distance and the angle function are measured. The latency of the computation is measured by introducing the timer or clock at the start of hook function and it waits until an interrupt is given by pressing the keyboard, once the interrupt is initiated another timer is introduced at that point. These timers start its clock since the last reset, so at last the difference between the end timer and start timer is calculated and using the number of frames processed the computation latency is measured. The other latency is due to grabbing frame and it is measured using the Mil library function M_PROCESS_FRAME_RATE which calculates the average of camera’s rate and total processing time. At a later stage the time taken for one frame is calculated. Also the total time for entire application is measured by introducing the clock at start of the application i.e., before the camera starts grabbing and storing in buffer and also at the end of the application. The processed frame and missed frames are calculated by using the Mil library function.

Hook function = distance & angle algorithm

Computation latency (CL) = Time taken for hook function/processed frames

Latency due to grabbing frame (GL) = 1/frame rate per second

Total Latency - (CL + GL)

The processed frames, missed frames, frame rate, time on hook function, synchronization time and total time is calculated at various occasion and conditions. At a later stage the average of total latency is calculated.
4 Theory

The Matrox Iris GT is a smart camera, which has Windows XP Embedded which is able to act as a PC based machine vision system as shown in fig 4.1. Iris GT has a Sony ICX274AQ progressive scan CCD image sensor with square pixels of diagonal size 8.923mm (Type 1/1.8). It has 2.01M effective pixels with an effective resolution at 1600 x 1200. The pixel size at the effective resolution is 4.4µm x 4.4µm. The Iris GT has a 1.6GHz Intel Atom processor with 512 MB of volatile memory and 2GB of non-volatile memory.

Fig 4.1 – Matrox Iris GT

The Iris GT has three main interface connectors, namely the digital I/O and power connector, the VGA/USB connector and the 100/1G Base T connector as shown in fig 4.2. The digital I/O and power connector is a M12 17 pin male connector that is used to both transmit and receive a digital I/O signal and an RS-232 signal, and also to provide power to the Iris GT. The VGA/USB connector is a M12 12 pin female connector that is used to transmit and receive a standard RGB analog video signal, it also transmits and receives USB signals. The 100/1G Base T connector M12 8 pin female connector establishes communication at 10Mbit/sec, 100Mbit/sec, and 1Gbit/sec.
Iris GT has a C-mount 12 mm lens. The software environment used in the development system in order to create an application is the Microsoft Visual Studio 2008 and the Matrox imaging library. The communication is established between the development system and the Iris GT through a 1Gbit ethernet interface and can be remotely debugged using a Visual Studio remote debugger.

4.1 Develop an application and deploying in Iris GT

Fig 4.2 – Interface connectors of Iris GT

Fig 4.3 – Development PC & Iris GT
A development PC is necessary in order to write a Visual C++ script using the Microsoft visual studio and, at a later stage, using the remote debugger, the developed application can be deployed in the Iris GT. At first, the local area connection should be established between the development PC and the Iris GT using the 1Gbit Ethernet interface. The Matrox Iris GT and the development PC are configured using a static IP address as given in the data sheet of the Iris GT. After the connection has been established between the Iris GT and the PC, the remote debugger should be installed on both sides so that the application can be deployed on the Iris GT. Once the remote debugger has been installed, the application can be deployed in the Iris GT and an executable file is created in the Iris GT. Later, the executable file can be executed solely in the Iris GT without connecting to the development PC and the Iris GT then works as a standalone device to run the application.

### 4.2 Powering Iris GT & connecting the I/O pins

A breakout board is used to power the Iris GT and to connect the I/O cable, re-route signals, generate triggers and send/receive information through the serial cable.
In figure 4.4 the block A shows the smart camera connector, in which J15, J16, and J17 are the power connectors and the ground. J13 & J14 are the RS232 transmitter and receiver pin and are used to send the distance and angle values to the LabVIEW. The 24V DC power is provided to the breakout board in the block B, which can be seen in fig 4.4. In the figure shown below, a power source to the breakout board in block B, digital I/O pins in block A and a female RS232 cable at block E can be seen. The remainder of the unused digital I/O pins are wrapped together using insulating tape.

Fig 4.5 – Breakout Board
5 Design

5.1 Algorithm

The overview of the design is shown in fig 5.1, where it is divided into three main steps, namely the continuous image acquisition, image processing algorithm and displaying results in a serial monitor.

The image processing algorithm is shown in fig 5.2 where the captured image is processed by the following steps, segmentation, morphological post processing, component labelling, feature extraction and calculation. The results of this algorithm provide the calculated distance and viewing angle from the board, and, at a later stage, these results are transferred serially using the rs232 cable and displayed in the LabVIEW window.
5.2 Implementation

The three main steps of the overall design are the continuous image acquisition, image processing algorithm and displaying the results in a serial monitor. The continuous image acquisition is conducted by using the MIL function [4] and is stored in the buffer. Following this, the images stored in the buffer are processed sequentially by using the Image processing algorithm and the results are stored in another temporary buffer. Later, these values in the buffer are transferred using the rs232 serial communication to another PC and displayed in the LabVIEW window. Additionally, the processed video frames are displayed in an MIL window on the camera side.
5.2.1 Continuous Image Acquisition

The continuous image grab is conducted using the MIL function MdigGrabContinuous [4]. This function by default is asynchronous as it grabs one image into the buffer while processing the previously grabbed images. An improved real time grab is achieved by having a higher frame rate. When the memory buffer is allocated by the user before grabbing the images, a higher frame rate is thus achieved. The memory buffer and the maximum buffering size are initialized using the MIL and standard Visual C++ library, later, by using the grab function the images are captured and stored in the allocated buffer. The images in the buffer then undergo various steps in the image processing algorithm in order to yield the distance and angle.

![Multiple images captured using Iris GT](image)

- Memory Buffer

5.2.2 Segmentation

In image segmentation, the object and the background are separated by converting the standard rgb image to the binary image. In the binary image, the object represents the logical value ‘1’ and the background represents the logical value ‘0’. In a binary image, the objects, i.e. the six circular spots, can be easily differentiated from the background, which is basically the segmentation process. In fig 5.3 the six spots can be viewed for which the spots represent the logical ‘1’ and the remainder is the logical ‘0’.
5.2.3 Morphological Post Processing

Due to illumination and some external environmental disturbances, noise is introduced in the image as some tiny objects. In order to remove of the tiny objects, the area of the circular light spots can be defined in the binary image or, the tiny disturbance can be merely removed by giving the threshold value as the number of pixels which are less than the light spots, to be removed from the binary image. By conducting this morphological post processing, the object i.e., the circular light spots, can be well defined and easily selected for the component labelling. In fig 5.4, near the bottom right corner, a small white object is introduced which may be due to noise in the image.
5.2.4 Component Labeling

After segmentation and morphological post processing, each of the light spots should be labelled as a separate component or object in the image. While labelling the object, either ‘4’ connectivity or ‘8’ connectivity can be used and the connectivity defines how the objects are connected in relation to the pixel.

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

4 connectivity

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

8 connectivity

In the above 3x3 window, the pixel at position (2,2) is the centre pixel. In ‘4’ connectivity, the neighbouring pixels are the ‘4’ nearest pixels (one in black color). In ‘8’ connectivity, the neighbouring pixels are the ‘4’ connected pixels and the pixels that are diagonally connected (one in green color). In this algorithm the object in the image is labelled by using the ‘4’ connectivity.
5.2.5 Feature Extraction

The light spots have been labelled as each separate object in the image and the information from the labelled object is obtained. The six light spots are considered as six blobs in the image. Now, the centre of gravity of each blob is calculated and the pixel’s x & y position of the centre of gravity in the image is observed. The pixel’s x & y positions are used in determining the vertical and horizontal spacing between the six blobs as shown in the methodology. In the centre of gravity of each blob the red color mark can be seen in figure 5.5.

![Fig 5.5 – Centre of gravity of each light spot](image)

5.2.6 Calculation

Using the position of the centre of gravity of the six blobs, they can be sorted or rearranged in the manner as shown in the table below. After sorting the blobs, the vertical and horizontal distances are calculated as shown below.

<table>
<thead>
<tr>
<th>Blob 1</th>
<th>Blob 3</th>
<th>Blob 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blob 2</td>
<td>Blob 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Blob 6</td>
</tr>
</tbody>
</table>

Vertical distance 1 = distance between blob 2 and blob 1
Vertical distance 2 = distance between blob 4 and blob 3
Vertical distance 3 = distance between blob 5 and blob 6
Horizontal distance 1 = dist. between blob 1 & 3 + dist. between blob 3 & 5

Horizontal distance 2 = dist. between blob 2 & 4 + dist. between blob 4 & 6

The equation,

\[ R = \frac{f_A}{a} \]

is used and here, ‘R’ is the distance from the board.

By using the equation,

\[ \Phi = \sin^{-1} \left( \frac{aR}{A \left( \sqrt{a^2 + f^2} \right)} \right) \]

for which ‘\( \Phi \)’ is the viewing angle from the target board. The camera is placed in different positions such as in the right side, centre, and over the left side facing the board; the distance and angle from the board is then calculated.

5.2.7 Displaying Results in Serial Monitor

The calculated values for the distance and angle are converted from numbers to strings using Visual C++ library [10]. These strings are then transferred to a PC using an RS232 serial cable. In the PC, the LabVIEW software is installed and the strings are converted back to numeric values, and then displayed in the distance scale and in an angular meter. Several LabVIEW functions are used to open the serial port, receive string values and these strings are later split into appropriate distance and angle values and, finally, the strings are converted into numeric values and displayed. The following flowchart explains the workflow of the serial communication and displaying of the results.
Numbers to strings in IRIS GT

The calculated values for the distance and angle are converted into strings using CString in Visual C++ library [10][10]. The distance and angle for a frame are combined and sent as a single string which consumes 11 bytes. In this case, 5 bytes are used for distance, 5 bytes for angle and a single byte for a constant string in order to separate them in LabVIEW. If the distance is 1.254 and angle is 60.63 the string has the appearance “1.254B60.63”.

Fig 5.6 – Workflow of serial communication & LabVIEW
Transferring data through RS232

The RS232 serial connection is established between the Iris GT and a PC. Initially, the baud rate, byte size and stop bits are set on both sides. A baud rate of 115200, 8 bits per byte and one stop bits were set. In LabView there are specific blocks to perform this operation, which can be discussed in the following.

Open serial port in LabVIEW

![VISA Configure Serial Port](image)

**Fig 5.7 – Configure Serial port**

The visa configure serial port consists of several input options such as, the resource name, in which the specific com port number is given. The baud rate is set to 115200 and 8 bits per byte is also given. The basic input options are shown over the left side of the block in figure 5.7 and the output of this block consists of the incoming signals from the specific com port.

Reading and splitting the incoming values

![Read the values](image)

**Fig 5.8 – Read the values**

The visa read block reads the incoming signals from the com port as a string and displays it as a string. As stated previously, it displays the distance and the angle as “1.254B60.63”. Now the distance, which is the first 5 bytes and angle, which is the last 5 bytes, are split using the block search and split string which is shown in figure below.
The results of the search/split string blocks will be the exact distance and angle values, but, in string format.

**Convert string to number and displaying results**

The incoming distance and angle are both in string format; these strings are converted into numbers by using the fract/exp strings to the numbers block as shown in the figure below.

The resulting distance and angle values are then shown in the measuring scale and angular meter. The results keep on updating for each and every frame in the Iris GT, and the resulting distances and angles are transmitted serially and displayed in LabVIEW. The final block of LabVIEW is shown in figure 5.11.
In the above figure, the distance and angular meters can be seen, which will be keep on updating for every frame taken by the Iris GT.
6 Results

In this chapter the distances and viewing angles at different positions for the Iris GT in real time video processing can be seen. In addition, the evaluation of the different experiments carried out is shown.

6.1 Position of the camera

At first, the Iris GT is placed at three different positions i.e., at left side, right side and in front of the target board. Then, a Visual C++ script is developed for grabbing a single frame and it calculates the distance and angle from the board and later sends it to a PC via a serial cable and this is shown in LabVIEW. In figure 6.1 the camera is placed in front of the board and the distance and viewing angle from the board is displayed in the LabVIEW window. The distance between the camera and target board is found to be 1.288 meters and the viewing angle is about 90 degrees.

![Figure 6.1 – Front view & results](image)
In figure 6.2, the camera is placed over the right side of the board and the distance and angle are calculated for the right side view. The distance between the camera and target board is found to be 0.928 metres, and the viewing angle is about 69.56 degrees.

Figure 6.2 – Right side view
In figure 6.3, the camera is placed over the left side of the target board. The distance between the Iris GT and the target board is found to be 0.899 meters, and the viewing angle from the board is found to be 117.1 degrees.

The distance and angle algorithm is then applied in relation to real time video processing and a separate Visual C++ script is developed and the results of the distance and angle are printed in the command prompt and in the LabVIEW window sequentially as the individual frames are captured and processed. In figure 6.4 it can be noted that the distance and angle are printed continuously.
6.2 Evaluation results

The four main experimental evaluations are carried out as discussed previously in chapter 3 and the results were discussed in the following section.

6.2.1 Frame speed vs Resolution

The Iris GT supports up to 15 fps at an effective resolution. After applying the image processing algorithm, the frame speed is verified for different resolutions. The resolution is varied by using image resizing and also an image cropping technique.

The table below shows the frame speed at three different resolutions and for which the pixel size remains constant.
<table>
<thead>
<tr>
<th>Image Resolution</th>
<th>Frame rate per second</th>
<th>Pixel size</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 x 480</td>
<td>33.9</td>
<td>4.4µm x 4.4µm</td>
</tr>
<tr>
<td>768 x 576</td>
<td>29.3</td>
<td>4.4µm x 4.4µm</td>
</tr>
<tr>
<td>1600 x 1200</td>
<td>14.7</td>
<td>4.4µm x 4.4µm</td>
</tr>
</tbody>
</table>

Table 6.1 – Analysed results for cropping

The results for the image resizing can be viewed in Table 6.2 where the pixel spacing changes for different resolutions and is calculated using equation 2.3.1. The image size (diagonal) for the sensor is 8.923mm in the equation and the horizontal and vertical sizes change depending on the selected resolution.

<table>
<thead>
<tr>
<th>Image Resolution</th>
<th>Frame speed per second</th>
<th>Pixel size in µm</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 x 480</td>
<td>15</td>
<td>11.1 x 11.1</td>
</tr>
<tr>
<td>768 x 576</td>
<td>14.7</td>
<td>9.2 x 9.2</td>
</tr>
<tr>
<td>1600 x 1200</td>
<td>14.7</td>
<td>4.4 x 4.4</td>
</tr>
</tbody>
</table>

Table 6.2 – Analysed result for resizing

From both the above shown methods, it can be seen that the image cropping has more influence on the frame rate. The frame rate significantly increases as the resolution decreases.

### 6.2.2 Power vs Resolution

The power consumed by the Iris GT is tested at different resolutions in relation to performing the distance and angle function over every frame. The table shown displays the power consumed for both cropping and resizing.
<table>
<thead>
<tr>
<th>Image Resolution</th>
<th>Power consumption (cropping)</th>
<th>Power consumption (resizing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 x 480</td>
<td>5.28 – 5.52 W</td>
<td>5.28 W</td>
</tr>
<tr>
<td>768 x 576</td>
<td>5.29W</td>
<td>5.28 – 5.30 W</td>
</tr>
<tr>
<td>1600 x 1200</td>
<td>5.28W</td>
<td>5.28 – 5.30 W</td>
</tr>
</tbody>
</table>

Table 6.3 – Power vs Resolution

Generally the power consumption is constant while varying the resolutions and it is approximately in the range of 5.28 – 5.52 Watts.

### 6.2.3 Accuracy of position of measurement

The accuracy of the position of measurement is verified manually and in LabVIEW. Initially, the camera remains in front of the target board i.e., 90 degrees and the distances are measured at various positions both manually and in LabVIEW.

<table>
<thead>
<tr>
<th>Distance in LabVIEW</th>
<th>Distance Manual</th>
<th>Angle in Degrees</th>
<th>Error in metres</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.628m</td>
<td>0.60m</td>
<td>90</td>
<td>0.028</td>
</tr>
<tr>
<td>1.264m</td>
<td>1.23m</td>
<td>90</td>
<td>0.034</td>
</tr>
<tr>
<td>1.498m</td>
<td>1.47m</td>
<td>90</td>
<td>0.028</td>
</tr>
</tbody>
</table>

Table 6.4 – Error for Distance

The difference between the distance in LabVIEW and from the manual measurement can be seen in the above table. The error is very low and there is a possibility of there being a minor error in the manual measurement.

The angle is measured using the equation,
\[ \theta = \sin^{-1}\left( \frac{aR}{A\sqrt{a^2 + f^2}} \right) \]

Sometimes at angles from 91 to 98 degrees the angle becomes \( \theta = \sin^{-1}(>1) \), so it yields a complex numbers. In order to avoid this, a condition is given while programming, which could be \( \sin^{-1}(>1) = \sin^{-1}(1) \), which then forces the result to be 90 degrees. In relation to this condition, the system fails to provide an accurate angle.

### 6.2.4 Latency of the system

The latency of the system is measured by placing the camera at various locations and in different time interval. The results of processed frame, missed frame, time for hook function (computation time for distance and angle algorithm), synchronization time, frame rate and total time are presented in the table 6.5.

<table>
<thead>
<tr>
<th>S N o</th>
<th>Processed frame</th>
<th>Missed frame</th>
<th>Sync time in sec</th>
<th>Computation time in sec</th>
<th>Frame rate per second</th>
<th>Total time in sec</th>
<th>Computation latency</th>
<th>Latency due to grab</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2014</td>
<td>280</td>
<td>0.177</td>
<td>68.311</td>
<td>30.181</td>
<td>68.509</td>
<td>33.9ms</td>
<td>33.1ms</td>
</tr>
<tr>
<td>2</td>
<td>1613</td>
<td>211</td>
<td>0.176</td>
<td>54.472</td>
<td>29.431</td>
<td>54.668</td>
<td>33.7ms</td>
<td>33.9ms</td>
</tr>
<tr>
<td>3</td>
<td>4718</td>
<td>749</td>
<td>0.177</td>
<td>161.75</td>
<td>29.4</td>
<td>161.95</td>
<td>34.2ms</td>
<td>34ms</td>
</tr>
<tr>
<td>4</td>
<td>2043</td>
<td>314</td>
<td>0.176</td>
<td>70.164</td>
<td>28.531</td>
<td>70.361</td>
<td>34.34ms</td>
<td>35.04ms</td>
</tr>
<tr>
<td>5</td>
<td>1562</td>
<td>236</td>
<td>0.176</td>
<td>53.72</td>
<td>30.384</td>
<td>53.917</td>
<td>34.39ms</td>
<td>32.91ms</td>
</tr>
<tr>
<td>6</td>
<td>3586</td>
<td>562</td>
<td>0.176</td>
<td>122.89</td>
<td>27.199</td>
<td>123.09</td>
<td>34.26ms</td>
<td>36.76ms</td>
</tr>
<tr>
<td>7</td>
<td>2502</td>
<td>390</td>
<td>0.176</td>
<td>85.92</td>
<td>28.65</td>
<td>86.116</td>
<td>34.34ms</td>
<td>34.9ms</td>
</tr>
<tr>
<td>8</td>
<td>5954</td>
<td>943</td>
<td>0.176</td>
<td>203.84</td>
<td>27.802</td>
<td>204.04</td>
<td>34.23ms</td>
<td>35.96ms</td>
</tr>
<tr>
<td>9</td>
<td>7904</td>
<td>1081</td>
<td>0.177</td>
<td>265.32</td>
<td>29.968</td>
<td>265.52</td>
<td>33.56ms</td>
<td>33.36ms</td>
</tr>
<tr>
<td>10</td>
<td>2547</td>
<td>397</td>
<td>0.176</td>
<td>87.436</td>
<td>29.107</td>
<td>87.633</td>
<td>34.32ms</td>
<td>34.35ms</td>
</tr>
<tr>
<td>11</td>
<td>4713</td>
<td>634</td>
<td>0.177</td>
<td>158.191</td>
<td>33.928</td>
<td>158.38</td>
<td>33.5ms</td>
<td>29.47ms</td>
</tr>
<tr>
<td>12</td>
<td>6014</td>
<td>962</td>
<td>0.176</td>
<td>206.18</td>
<td>29.714</td>
<td>206.37</td>
<td>34.28ms</td>
<td>33.65ms</td>
</tr>
</tbody>
</table>

Table 6.5 Latency of the system

Computation latency (CL) = Time taken for hook function/processed frames

Latency due to grabbing frame (GL) = 1/frame rate per second

\[ \text{Total Latency} = (\text{CL} + \text{GL}) \]
<table>
<thead>
<tr>
<th>Computation Latency (CL) in ms</th>
<th>Latency due to grab (GL) in ms</th>
<th>Total latency CL+GL in ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.9</td>
<td>33.1</td>
<td>67</td>
</tr>
<tr>
<td>33.7</td>
<td>33.9</td>
<td>67.6</td>
</tr>
<tr>
<td>34.2</td>
<td>34</td>
<td>68.2</td>
</tr>
<tr>
<td>34.34</td>
<td>35.04</td>
<td>69.38</td>
</tr>
<tr>
<td>34.39</td>
<td>32.91</td>
<td>67.3</td>
</tr>
<tr>
<td>34.26</td>
<td>36.76</td>
<td>71.02</td>
</tr>
<tr>
<td>34.34</td>
<td>34.9</td>
<td>69.24</td>
</tr>
<tr>
<td>34.23</td>
<td>35.96</td>
<td>70.14</td>
</tr>
<tr>
<td>33.56</td>
<td>33.36</td>
<td>66.92</td>
</tr>
<tr>
<td>34.32</td>
<td>34.35</td>
<td>68.67</td>
</tr>
<tr>
<td>33.5</td>
<td>29.47</td>
<td>62.97</td>
</tr>
<tr>
<td>34.28</td>
<td>33.65</td>
<td>67.93</td>
</tr>
</tbody>
</table>

Table 6.6 Total Latency

The average of computation latency = 34.08ms  
Average of latency due to grab = 33.95ms  
Average of total latency = 68.03ms  
These above calculated results are obtained at 640 x 480 resolution.
7 Discussion

Based on the experimental results, various aspects can be concluded such as an improvement in the frame rate and, for this, two methods were followed. It appears that image cropping is more effective than image resizing. In image cropping, as the resolution decreases, the frame rate increases, but, in image resizing, it is almost constant, since the frame grabber (camera) grabs at the whole resolution and the image resizing is part of the processing operation. The figure shown below emphasizes the variation between the two processing methods.

![Frame speed vs Resolution](image)

Fig 7.1 – Plot for frame rate Vs resolution

In figure 7.1 the red line indicates the results for the image cropping and the blue line indicates the results for the image resizing and it can be clearly seen that the, cropping is more effective than resizing. The resolution in the y axis of the graph is only the horizontal pixel size. Also the interpolation method followed in this evaluation is nearest neighbour pixel interpolation, since nearest neighbour is the simplest
and fastest of image scaling. There are also interpolation techniques such as bilinear and bicubic can be used in Iris GT for resizing operation. The frame rate varies depending upon the computation load of the distance and angle algorithm and the viewing accuracy of the camera to the target board, so the presented frame rate varies from time to time and not a constant value. The maximum achievable frame rate is presented in the result chapter.

It can be stated that there is only minor disturbance to the power while changing the resolution and the average power found from both the cropping and resizing methods is approximately 5 watts.

In the distance measurement the error is low and it can be rectified by improving the triangulation method. The error in the angle can only be rectified when complex numbers are solved to determine the angle, which can be complicated and it makes programming much more complex. Since it introduces complexity to the programming, this could have a possible effect on the system performance in relation to achieving a higher frame rate.

The latency of the system measured at 640x480 resolution and the average latency is about 68.03ms. Together the computation latency and latency due to grabbing frame forms the total latency. These two latencies are measured using the frame rate, frame count, frame missed, computation time and total time.
8 Conclusions

The goal of the project has been achieved by designing a machine vision system using the Iris GT as a smart camera to determine the distance and angle to a calibrated target board. A visual C++ script has been developed for real time video processing in order to implement the distance and angle algorithm and send the data through the serial cable to a print in a PC. The input video undergoes various image processing steps including image segmentation, morphological post processing, component labelling, feature extraction and a calculation in the distance angle algorithm. These calculated results are later displayed in the LabVIEW window, which is transferred through the rs232 to a separate PC. After implementing the distance angle algorithm in real time video processing, the frame rate and power dissipation were calculated by using two methods, namely, image cropping and image resizing. The results were observed in both the cases and analyzed. At a later stage the accuracy of system and latency of the system is measured and analyzed. This machine vision system can be used in a navigation system to determine the distance and angle by placing a calibrated board or other object in a room. This navigation system can be installed for a blind person who stays in a hospital or special care unit and it can be used for navigating within the room or throughout an entire building. This application can be built in any cameras and for use with any platforms. The power dissipation and frame rate are the only factors that are required to be considered while designing this application in other cameras and platforms. In the future, a different algorithm could be developed for determining the distance and angle instead of the triangulation method or, it might be possible to improve the triangulation method.
References


Vision Sensor-
Distance & Angle
Prasanna Kumar P

References

2013-03-01


Appendix A: Documentation of developed program code

/******************This program grab continuously******************
*****calculates Distance & Angle from board for 1600x1200 Resolution***
/******************and displays output results in labview************/

//Local Defines
#include <mil.h>
#include <complex>
#include "atlstr.h"
#include "windows.h"
#include <iostream>

using namespace std;

//Threshold for morphological post processing
#define BWAREAOPEN 5L

//Maximum Buffering size
#define BUFFERING_SIZE_MAX 5

//To send the values through serial cable & configure serial port
bool serialport (CString portnumber, CString variable1, CString variable2, CString variable3)
{

    DCB serial;
    DWORD write;
    HANDLE hPort = CreateFile (portnumber,GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL);
    if (!GetCommState(hPort,&serial))
        return false;
    serial.BaudRate = CBR_115200; //115200 Baud
    serial.ByteSize = 8; //8 data bits
    serial.Parity = NOPARITY; //no parity
    serial.StopBits = ONESTOPBIT; //1 stop
    if (!SetCommState(hPort,&serial))
        return false;
    WriteFile(hPort,variable1,5,&write,NULL);
    WriteFile(hPort,variable2,1,&write,NULL);
    WriteFile(hPort,variable3,5,&write,NULL);
    CloseHandle(hPort);

    return 0;
}

//Initialize the structure for hook function
typedef struct
{
    MIL_ID MilSystem,
    MilDigitizer,
    MilImageDisp;

MIL_INT ProcessedImageCount;
    double Angle;
    double Distance;
} HookDataStruct;

//Distance & Angle hook function
MIL_INT MFTYPE DistanceAngle(MIL_INT HookType, MIL_ID HookId,
                                      void *MPTYPE
                                      HookDataPtr)
    {
        HookDataStruct *UserHookDataPtr = (HookDataStruct *)HookDataPtr;
        MIL_ID MilBlobResult,
        ModifiedBufferId,
        MilBlobFeatureList,
        MilBinImage;
        MIL_INT TotalBlobs,
        n;
        MIL_DOUBLE *CogX,
                *temp,
                *CogY;
        const double pi = 3.141592654;

        //Pixel Distance
        const double pixd = 4.4E-6;

        //Focal Length
        const double f = 0.012;

        //Horizontal & vertical distance in board
        const double D = 0.082;
        const double K = 0.127;
        double vdist[3];
        double rd[3];
        double hdist[2];
        double rad;
        double sinB;
        CString cs;
        CString fs;

        //Getting the ID of the grabbed buffer
        MdigGetHookInfo(HookId, M_MODIFIED_BUFFER+M_BUFFER_ID,
                        &ModifiedBufferId);

        //Allocating buffer
        MbufAlloc2d(UserHookDataPtr->MilSystem,
                        MbufInquire(UserHookDataPtr->MilImageDisp, M_SIZE_X, M_NULL),
                        MbufInquire(UserHookDataPtr->MilImageDisp, M_SIZE_Y, M_NULL),
                        1+M_UNSIGNED, M_IMAGE+M_PROC, &MilBinImage);

        //Converting into Binary Image
        MimBinarize(UserHookDataPtr->MilImageDisp,
                        MilBinImage,M_GREATER_OR_EQUAL, M_DEFAULT, M_NULL);

        //Morphological post Processing
        MimOpen(MilBinImage,MilBinImage,BWAREAOPEN,M_BINARY);

        //Calculating COG
        MblobAllocFeatureList(UserHookDataPtr->MilSystem, &MilBlobFeatureList);
        MblobSelectFeature(MilBlobFeatureList, M_CENTER_OF_GRAVITY);
MblobAllocResult(UserHookDataPtr->MilSystem, &MilBlobResult);
MblobCalculate(MilBinImage, M_NULL, MilBlobFeatureList, MilBlobResult);
MblobGetNumber(MilBlobResult, &TotalBlobs);
CogX = (MIL_DOUBLE *)malloc(TotalBlobs*sizeof(MIL_DOUBLE));
CogY = (MIL_DOUBLE *)malloc(TotalBlobs*sizeof(MIL_DOUBLE));
temp = (MIL_DOUBLE *)malloc(TotalBlobs*sizeof(MIL_DOUBLE));
MblobGetResult(MilBlobResult, M_CENTER_OF_GRAVITY_X, CogX);
MblobGetResult(MilBlobResult, M_CENTER_OF_GRAVITY_Y, CogY);

//Sorting the position of the six objects
{  
  bool found = true;
  while (found)
  {
    found = false;
    for ( n = TotalBlobs;n >-1; n--)
    {
      if(CogX[n]<CogX[n+1])
      {
        temp[n] = CogX[n];
        CogX[n] = CogX[n+1];
        CogX[n+1] = temp[n];
        temp[n] = CogY[n];
        CogY[n] = CogY[n+1];
        CogY[n+1] = temp[n];
        found = true;
      }
    }
  }
}

//Calculating vertical distance in image
vdist[0] = pixd*sqrt(pow(CogX[4]-CogX[5],2)+pow(CogY[4]-CogY[5],2));
vdist[1] = pixd*sqrt(pow(CogX[2]-CogX[3],2)+pow(CogY[2]-CogY[3],2));
vdist[2] = pixd*sqrt(pow(CogX[0]-CogX[1],2)+pow(CogY[0]-CogY[1],2));
rd[0] = (((f*D)/vdist[0]));
rd[1] = (((f*D)/vdist[1]));
rd[2] = (((f*D)/vdist[2]));

//Distance from the target board
UserHookDataPtr->Distance = rd[1];

//Calculating horizontal distance in image
hdist[0] = 0.5*pixd*(sqrt(pow(CogX[3]-CogX[5],2)+pow(CogY[3]-CogY[5],2)))+sqrt(pow(CogX[2]-CogX[4],2)+(CogY[2]-CogY[4],2)));
hdist[1] = 0.5*pixd*(sqrt(pow(CogX[1]-CogX[3],2)+pow(CogY[1]-CogY[3],2)))+sqrt(pow(CogX[0]-CogX[2],2)+(CogY[0]-CogY[2],2)));

//Calculating angle if the camera is placed over the //right side, left side and infront of the board
if( rd[0] > rd[2])
{
  sinB = (rd[0] * hdist[0])/(K*sqrt(pow(hdist[0],2)+pow(f,2)));
  if (sinB > 1)
    sinB = 1;
  rad = (asin(sinB));
  UserHookDataPtr->Angle = (180*(rad))/pi;
}
else if (rd[2]>rd[0])

\[
\begin{align*}
\sin B &= (r_d[2] \cdot h_{dist}[1])/(K\sqrt{(h_{dist}[1], 2)+pow(f, 2)}) ; \\
if (\sin B > 1) \\
\sin B &= 1; \\
\text{rad} &= (\text{asin} (\sin B)); \\
\text{UserHookDataPtr->Angle} &= (180-180*(abs(\text{rad})/\pi)); \\
\end{align*}
\]

// Processed frames are updated and displayed
MimArith(ModifiedBufferId, M_NULL, UserHookDataPtr->MilImageDisp, M_ABS);

// Printing the Results of Distance & Angle for every frame
MosPrintf(MIL_TEXT("Distance=%f, Angle=%f
"), UserHookDataPtr->Distance,UserHookDataPtr->Angle);

cs.Format(_T("%f"),UserHookDataPtr->Distance);
fs.Format(_T("%f"),UserHookDataPtr->Angle);

// send the results through serial port
serialport("COM4",(PCTSTR)cs,"B", (PCTSTR)fs);
system("cls");

// free the buffers
MblobFree(MilBlobResult);
MblobFree(MilBlobFeatureList);
MbufFree(MilBinImage);

return 0;

// Main Function
int MosMain(void)
{
    
    MIL_ID MilApplication;
    MIL_ID MilSystem;
    MIL_ID MilDigitizer;
    MIL_ID MilDisplay;
    MIL_ID MilImageDisp;
    MIL_ID MilGrabBufferList[MAXBUFFERING_SIZE] = { 0 };
    MIL_INT MilGrabBufferListSize;
    MIL_INT n=0;

    HookDataStruct UserHookData;

    // Allocating defaults for application
    MappAllocDefault(M_SETUP, &MilApplication, &MilSystem, &MilDisplay, &MilDigitizer, &MilImageDisp);
    MappControl(M_ERROR, M_PRINT_DISABLE);

    // Allocating buffers for display
    for (MilGrabBufferListSize = 0; MilGrabBufferListSize<MAXBUFFERING_SIZE; MilGrabBufferListSize++)
    {
        MbufAlloc2d(MilSystem,
                    MdigInquire(MilDigitizer, M_SIZE_X, M_NULL),
                    MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL),
                    M_DEF_IMAGE_TYPE,
                    M_IMAGE+M_GRAB+M_PROC,
                    &MilGrabBufferList[MilGrabBufferListSize]);
    }
}
if (MilGrabBufferList[MilGrabBufferSize])
{
    MbufClear(MilGrabBufferList[MilGrabBufferSize], 0xFF);
}
else
    break;
MappControl(M_ERROR, M_PRINT_ENABLE);

//Clearing buffers
for (n=0; n<2 && MilGrabBufferSize; n++)
{
    MilGrabBufferSize--;
    MbufFree(MilGrabBufferList[MilGrabBufferSize]);
}
MosPrintf(MIL_TEXT("\nVision Sensor - Distance & Angle.\n"));
MosPrintf(MIL_TEXT("-----------------------------\n"));
MosPrintf(MIL_TEXT("Press <Enter> to start.\n"));

//Grabbing continuously and display
MdigGrabContinuous(MilDigitizer, MilImageDisp);
MosGetch();

//Halt the grab
MdigHalt(MilDigitizer);

//Initialize the data structure for hook function
UserHookData.MilImageDisp = MilImageDisp;
UserHookData.MilSystem = MilSystem;
UserHookData.Angle = 0;
UserHookData.Distance = 0;
UserHookData.MilDigitizer = MilDigitizer;
UserHookData.ProcessedImageCount = 0;

//The processing function is started and called for every frame
MdigProcess(MilDigitizer, MilGrabBufferList, MilGrabBufferSize,
            M_START, M_DEFAULT, DistanceAngle, &UserHookData);
MosPrintf(MIL_TEXT("Press <Enter> to stop.\n"));
MosGetch();

//The processing function is stopped
MdigProcess(MilDigitizer, MilGrabBufferList, MilGrabBufferSize,
            M_STOP, M_DEFAULT, DistanceAngle, &UserHookData);
MosPrintf(MIL_TEXT("Press <Enter> to end.\n"));
MosGetch();

//Free the Buffers
while(MilGrabBufferSize > 0)
    MbufFree(MilGrabBufferList[--MilGrabBufferSize]);
MappFreeDefault(MilApplication, MilSystem, MilDisplay,
                MilDigitizer, MilImageDisp);
    return 0;
}