2011-01-01

2D Gaussian Object Motion Detection

Miguel Angel Chaidez
University of Texas at El Paso, machaidez@miners.utep.edu

2D GAUSSIAN OBJECT MOTION DETECTION

MIGUEL ANGEL CHAIDEZ

Department of Electrical & Computer Engineering

APPROVED:

John Moya, Ph.D., Chair

Virgilio Gonzalez, Ph.D.

Luis Trueba, Ph.D.

Benjamin Flores, Ph.D.
Acting Dean of the Graduate School

Dedication

I would like to dedicate this work to my loving family and wonderful fiancée Ana Maria Garcia whom have supported me through my endeavors to complete my Thesis. Without all of their help I would not be where I am today and for that I will always be thankful.

2D GAUSSIAN OBJECT MOTION DETECTION

by

MIGUEL ANGEL CHAIDEZ, BS in Electrical/Computer Engineering

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical & Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

August 2011

Acknowledgements

I would like to acknowledge the assistance and collaboration of Dr. John Moya, Ralph Loya, Robert Soto, Jorge Cardenas, Erica Gaytan, Robert Almendarez, Jacob Ontiveros, Lindsay Fredrick, Nathaniel Medrano, and Donna Alhakeim. They are the support group which allowed me to complete my thesis in one year.

Abstract

Dr. John Moya, and associated research assistants, have previously created an image-change recognition algorithm (JESSE) to mark changes within an image. The focus of this thesis is to present a physical application and modification of this algorithm in order to detect a surgeon's hand and verify chip placement on a printed circuit board.

There are different techniques in implementing visual recognition and motion detection with smart systems but the high cost and complicated calibration of these systems make them impractical. The goal was to create a system that is simple, inexpensive and applicable to multiple applications that will allow the user maximum flexibility.

The hardware necessary to provide input to the JESSE algorithm consists of a CMOS sensor camera, a light platform that serves as a stage and a laptop/pc with the necessary software (MATLAB and Visual Studio C# 2010). The Platform itself includes an adjustable stand for the CMOS camera, as well as an optional LCD, 16x16 LED Dot Matrix, pan and tilt servos, and a power supply. An algorithm that runs in the background as a user friendly GUI provides feedback on object targeting and was made in Visual Studio with cross-platform implementation to run on 32 and 64 bit architectures.

The JESSE algorithm itself samples two images, a reference and a test, via four Gaussian-weighted filters. Then using three of the Gaussians filters' results triangulates the position of change in the image. The location of this change may be marked in a new image, creating an output image with reference crosshairs.

The hardware as well as software platform is able to detect movement of a surgeon's hand or the absence of chip on a pc board, thus providing x and y change coordinates in a 2D space. The resulting coordinates x, y can be used for multiple applications such as a virtual mouse, running the pan and tilt on a second camera using two servos, or plotting to a dot matrix display, all of which have been implemented using the platform created from this thesis. There are also further possibilities for utilizing the resulting coordinates, as will be discussed at the end of this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The hardware designed in this thesis is a visual measurement tool /monitoring device to detect object change in real time by capturing images in a lossy system. The main focus was to improve the process currently used during surgery and PC Board quality detection.

Computer-aided surgery is performed where the position of an instrument in a person's body is adjusted via a technician controlling joysticks or complicated computer vision techniques and a camera. Computer Vision is also used in manufacturing to detect chip movement/absence. Although these techniques are effective, they carry high cost to the user and lack portability [1]. This project consisted of creating a very simple approach that avoids complications and creates a very affordable and user-friendly device.

The hardware aspects of the design consist of a web camera, a UART to USB converter, an additional microprocessor, a LED array to provide feedback, and a power supply. In the surgical application a gown spread out over a rolling cart serves as a background (reference picture) and multiple positions of a surgeon's hand are assessed over the background in different positions (test pictures). Through trial and error, a colored filter and direct florescent light was found to aid in giving the best results. The main goal was to use acquired pictures along with previously developed MATLAB algorithms to isolate the position of the surgeon's hand with a pinpointing set of crosshairs. The results are effective in allowing the creation of a monitoring device as compared to the techniques used currently for surgery. By isolating the surgeon's hand, one is also able to verify that the approach could be used for showing how the hand can be used or seen as a virtual mouse. Further, one similarly assess PC boards for the movement/absence of a chip.

The project focused on the idea that a surgeon when going into surgery must keep their hands sanitized as much as possible. A process is followed in order to keep a surgeon from introducing any bacteria. First, a surgeon puts on a surgeon's hair cap to prevent hair contamination, next their hands are washed up to the elbow, then a sanitization chamber is entered to remove any bacteria from their clothes, boots are then put on to cover their shoes, next their gown goes on, and lastly gloves are put on over the gown. In order to keep a surgeon's hands as sanitized as possible, techniques such as a

technician controlling an instrument by joysticks along with computer vision have been developed to avoid a surgeon physically touching anything and possibly contaminating their patient. Associated problems and complications have risen from these techniques and the presented design is built to avoid these complications.

Problems associated with a technician using joysticks to adjust an instrument's motions during surgery are that the equipment along with the added personnel being used is expensive. The joysticks are made with rubber boots, which have linings within them to aid gripping. These rubber boots are used surgery after surgery and it is difficult to keep them uncontaminated, which creates a problem for the hospital. Another problem is associated with the second technique, computer vision. Computer vision can be described as a technique that utilizes a computer to capture an image and evaluate it through image processing [2]. Computer vision approaches create complications because they rely on a surgeon's body to signal how to adjust an instrument. This creates problems since different surgeons do not have the same length of arms among other things. Computer vision is also commonly not user friendly, with high computer literacy required and difficulty in learning immediately.

1.1 Computer Vision and Image Processing

The human vision system is able to recognize objects or aspects of objects regardless of shape, color and/or distance. Computer Vision entails the use of algorithmic processing of images to extract information in a similar fashion. Image processing is a related area and refers to more general processing approaches, especially non real-time applications. The outcomes of these areas of research have led to applications such as counting of bottles in an assembly line, counting people passing through a door, etc. [3].

There are multiple techniques for computer vision and image processing, such as image segmentation, contouring, and database matching of static reference images to detected objects[4-9]. These methods are the basis for most current object recognition systems.

1.1.1 Image Segmentation

Image segmentation is the process of assigning pixels to groups to convert images into regions separated by curves and lines. This technique allows edge detection and grouping of pixels with similar color, texture and intensity and is commonly used to detect curves in images in order to build 3D models of objects. In medical applications, this method is may be used to detect tumors and skin cancer spots. Figure 1.1 shows how an image is outlined using segmentation. Further, the binary image provides a contour of the image, which can be used for object recognition [1, 4, 6, 10-13].

Figure 1.1: A silver bowl detected with the use of segmentation

In more complex implementations it is possible to stack segmented areas together in order to create 3D images. This latter process is computing intensive, further making this method a burden for low budget implementations [4, 11, 14].

Along with this conversion it is often necessary to use Gaussian smoothing and Morphological transformation in order to detect the shape of an object. This latter processing is complex and destructive, as the image is smoothed to fill-in blank spaces, which may or may not originally have been there [8, 15].

1.1.2 Image Contouring

Image contouring (or touring) as represented by Horry and Phy [7] is used to develop 3D touring systems from a 2D image. The process is best described in Figure 1.2 where the user input, the 3D polygon and the 3D structures, are used to create a representation of an object in a plane. This process requires calibration from the user as the position/distance of the camera from the object are essential for proper extraction of the object's coordinates [4, 8, 9, 14].

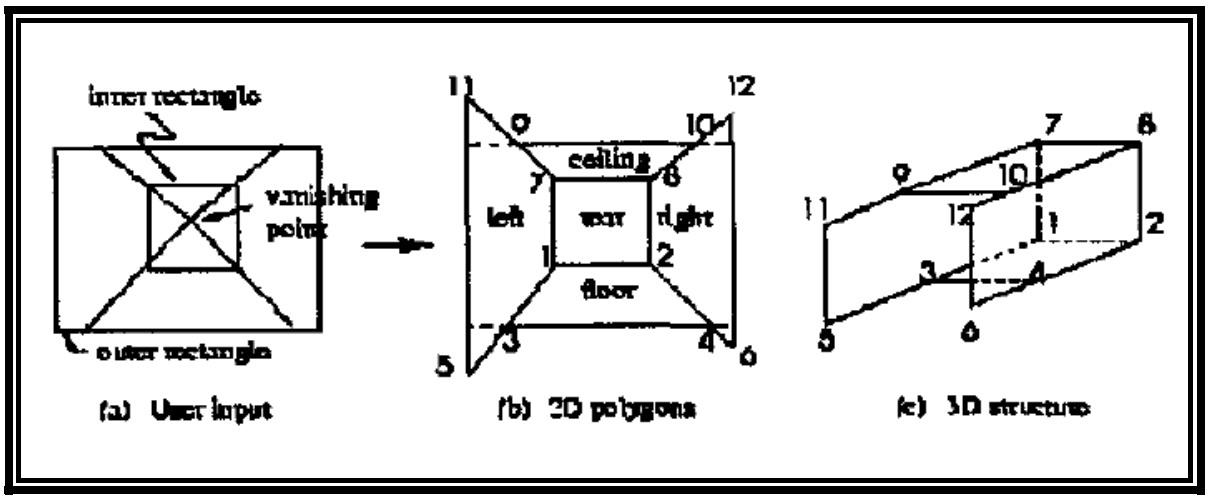


Figure 1.2: Model of the Horry and Phy Detection System [7]

This model has been the basis for some computer vision in the last decade, as other algorithms have stemmed from this process of mapping and reconstruction in different planes. The conversion of image pixels into numbers is very useful in representing the image [4, 8, 9].

An extension of the use of this algorithm can be seen in the Eu-Mi Ji et. al's [7]hand shape recognition system where they try to replace the keyboard and mouse via detection of hand gestures. Their intent was to replace gloves or other such markers in order to reduce requirements for external devices to the user. In a similar fashion the present algorithm's implementation is meant to reduce the use of expensive cameras and other such devices to allow the user maximum flexibility with a reduced cost.

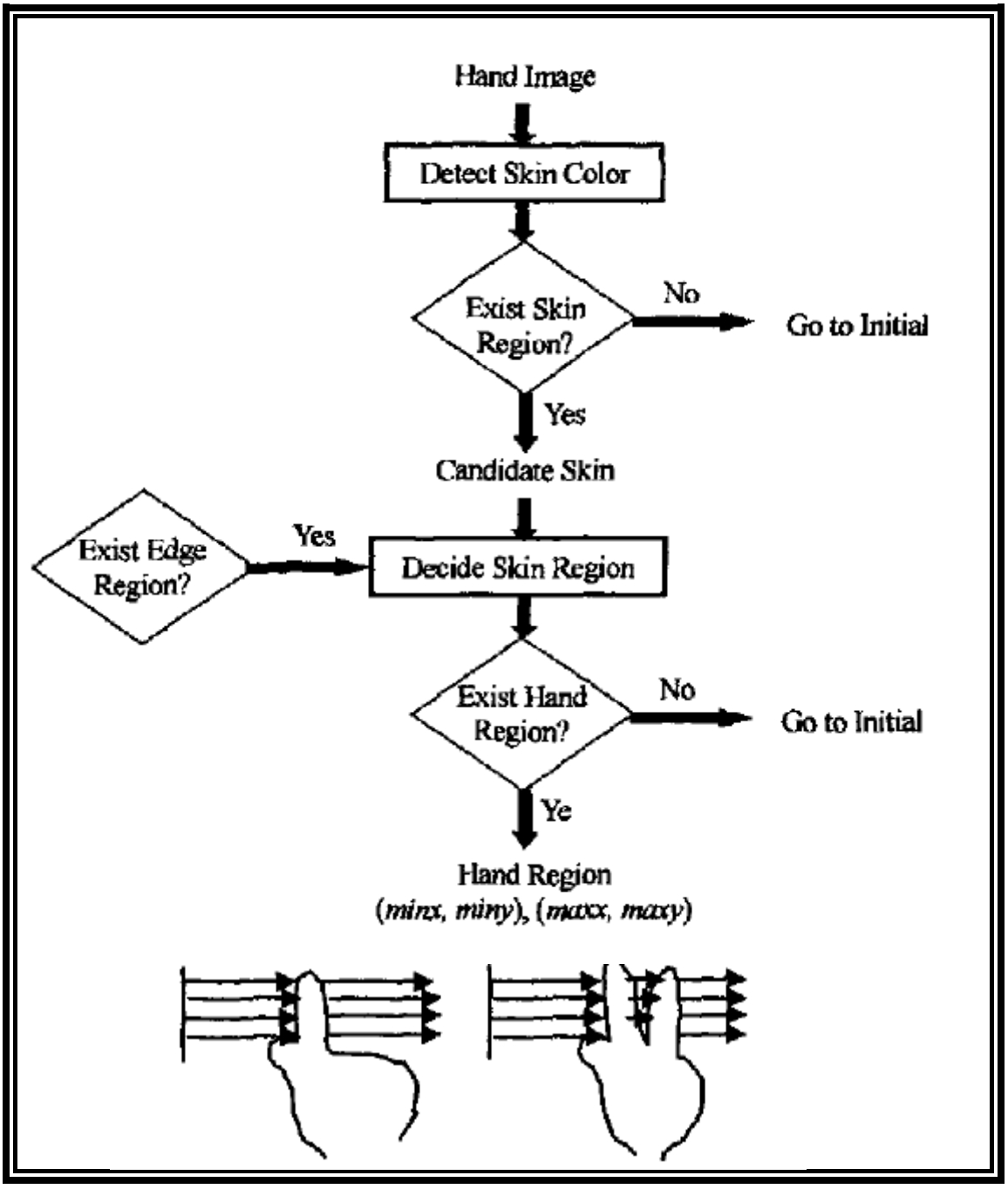


Figure 1.3: Skin Comparison Flow Chart and Hand Touring [8, 16].

1.1.3 Database Matching

The use of pixel/template matching (or skin matching techniques that are relevant to efforts similar to the one here) is the process of comparing a test pixel color (or the colors of groups of pixels)

to a database. Further, the combination of pixel matching with touring allows detection of the color and shape of an object like a hand. An example approach can be seen in Figure 1.3. This visual model by Eu Mi Ji et al. [1, 4, 8] shows a flowchart of skin detection and touring for detection of the hand region.

The process of skin comparison is effective for skin detection but it requires grouping similar pixels together then comparing those to a database in order to detect a hand. Various visual problems like occlusion and orientation can make the latter difficult. Adding touring removes shadows and the exact positioning of the hand [4, 8, 17].

The implementation of hand detection in the present case does not significantly depend on the color of the object. Further, the hardware aspects are designed to limit the effects shadowing has on the object itself. The only requirement is the need for a visual difference between the object being detected and the background to provide solid representation of the position.

1.1.4 RGB to HSV Conversion

The standard image is composed of red, green and blue colors blended together to form a picture. The conversion from RGB to HSV (Hue, Saturation, and Value) is a way to convert an image in order to represent color more compactly [1, 8, 15]. The method relies on a cylindrical concept where angular position indicates a class of color (hue), the radial distance from the center indicates the darkness or lightness of the color (saturation) and the vertical axis position indicates the intensity of the color. This conversion is common in computer graphics but is not as useful for pixel color matching due to its lack of color separation.

1.1.5 RGB to Grayscale Conversion

To reduce processing effort, an image may be converted into grayscale form (sometimes referred to as black and white) [1, 11, 18]. In certain cases it is possible to convert and image solely into black and white pixels (a binary image) in order to further reduce the computing cost over color analysis.

Shadowing in this process greatly distorts images, i.e. creating object pixels where there were none before. An example of conversion from an RGB image to grayscale is shown in Figure 1.4 [1, 12].

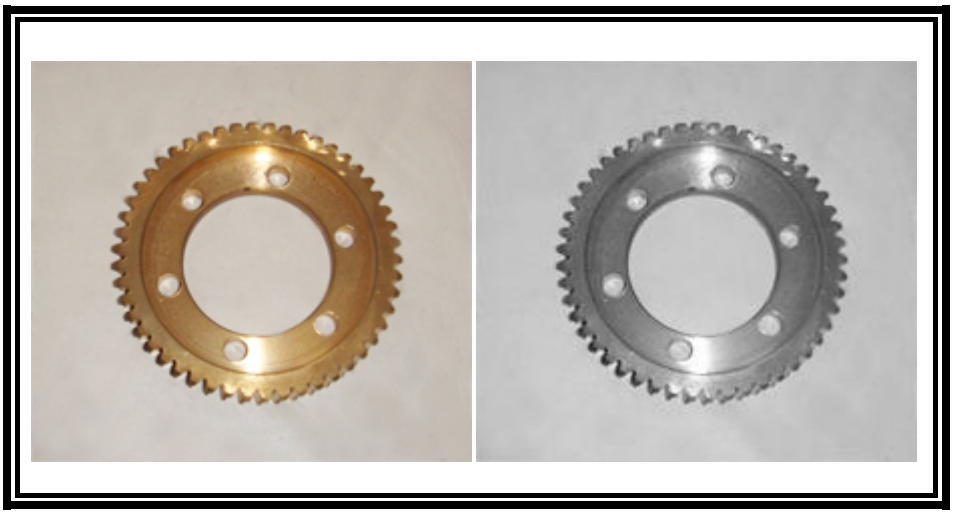


Figure 1.4: Conversion of an Image into B&W

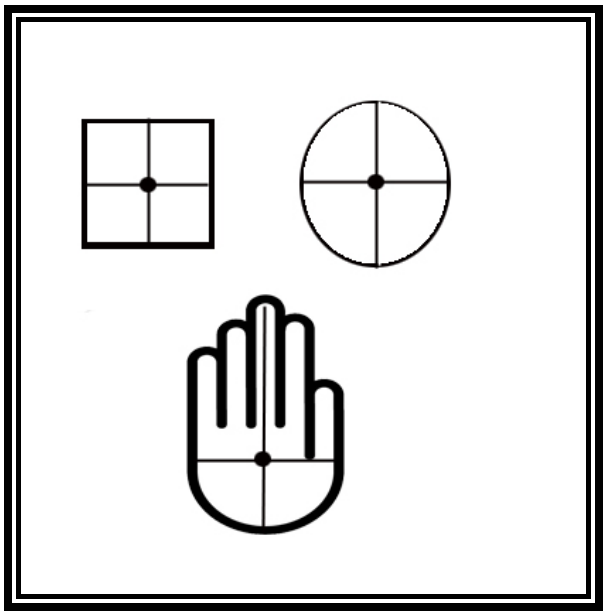


Figure 1.5: Centroid of Objects regardless of their shape or size[17]

1.1.6 Centroid Data Extraction

A centroid is the center of a figure is determined by its mean position. It is the averaged center of mass when discussing objects. The shape of the object does not matter; its overall plane area is summed and averaged to find the centroid. In a Gaussian curve the peak of the curve is the centroid and this can be used to summarize an area into a single value. Figure 1.5 and 1.6 show how the centroid of objects can be represented [1, 3].

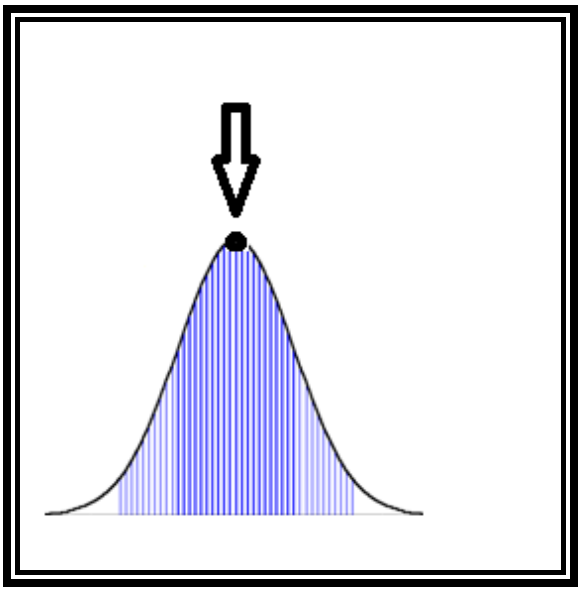


Figure 1.6: Centroid in respect to a Gaussian [3]

1.1.7 Blob/Particle Filtering

Blob or Particle Filtering is the process of comparing groups based on similar properties. The purpose of a blob filter is to ensure reliable understanding of a region within a noisy background. This process removes unwanted groups, leaving a black background with areas of interest. This process distorts the original image, but leaves blobs to be used for analysis. Using probability it is possible to reduce the number of blobs in the region of interesting by making regions of possibility [4, 9, 12].

The downfall of this process is that if the background and the object of interest are of similar color then the object will be removed thus decreasing the amount of usable data for object detection. Figure 1.7 shows how groups with wanted data will be affected by blob filtering.

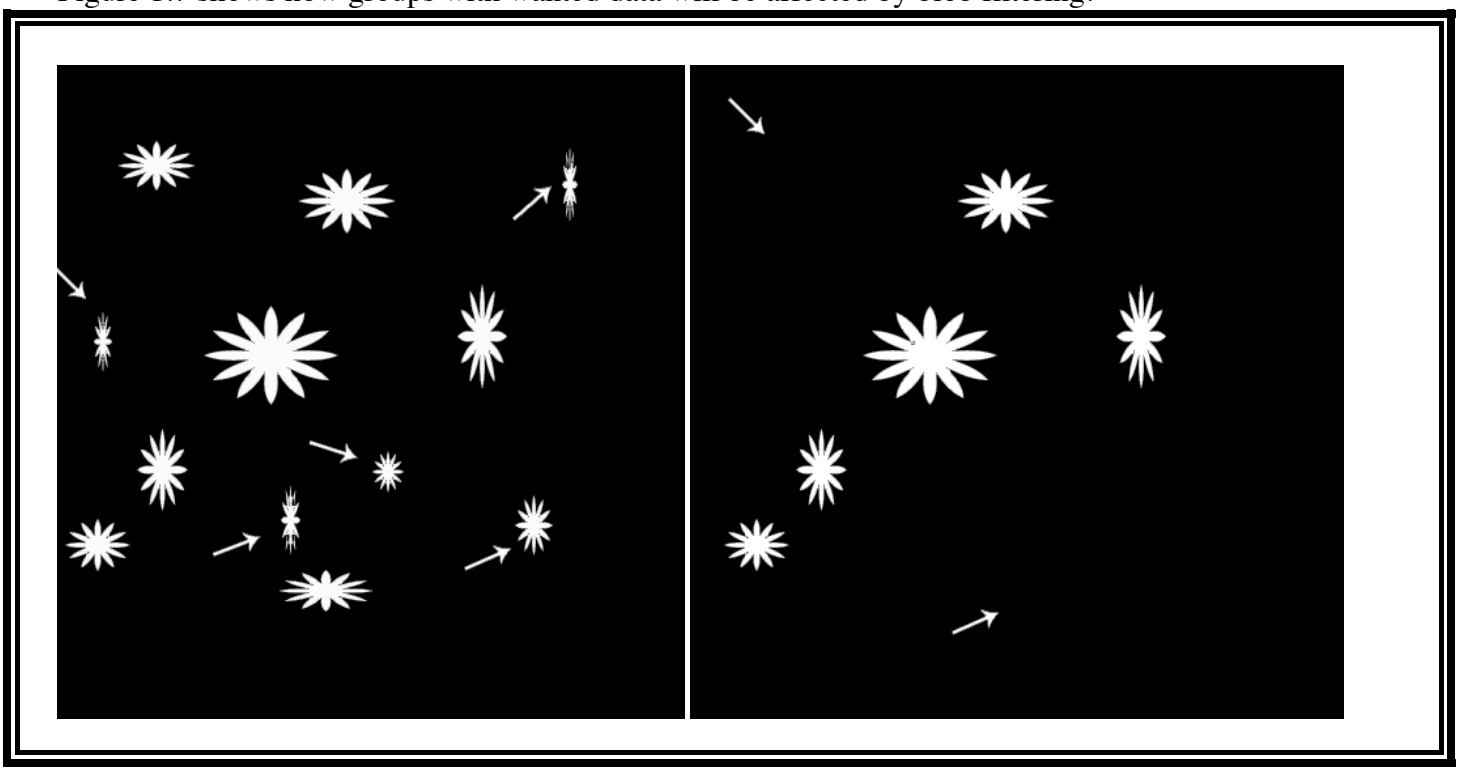


Figure 1.7: Blob Filtering. At the left in image blobs which are noise are pointed to by arrows. The right reveals the removed noise but also the loss of areas of interest (where the arrows point to missing blobs) [5, 9].

1.1.8 Bayesian Network and Hidden Markov Model (HMM)

Bayesian Networks indicate the level of relationship between similar events, and as such can predict future events [19]. This method can be used for a single sequence of events. In the case presented here, the path of a single object can determine from previous samples. This method is limited by the amount of objects within a background and can give false positives by predicting the wrong path [4, 20, 21].

Using a Two-layered Bayesian Network it is possible to predict upper and lower probabilities of the nature objects, further decreasing the error in object recognition. This method may remove clutter by dividing the respective sample into two parts and determining a path as a consequence of intercommunication two networks [1, 4, 5, 13, 19].

A Hidden Markov Model can be defined as a probabilistic model of states based on a Dynamic Bayesian Network. This implies the user's actions can be predicted. Thus areas of interest can be isolated.

There are two types of HMMs, the Coupled Hidden Markov Model (CHMM) and the Primitive-based Coupled Hidden Markov Model (PCHMM) [20]. The second type is an extension of the CHMM via dynamically allocating different regions of interest to determine multiple movements [19, 20].

An example of the states of a PCHMM's can be seen in Figure 1.8. The use of mapping and probability of for instance the movement of persons in a classroom may allow for interpretation of present or future images with respect to previous frames. However, although this method can be used in complex scenes, it requires training by the user [4, 19, 20]. For instance, the shape of each person is different. Thus a database is necessary in order to predict movements and positioning in the scene. Both HMM methods are a subset of pattern recognition and thus need a wide range of input in order to properly train a system to recognize movements[4, 19].

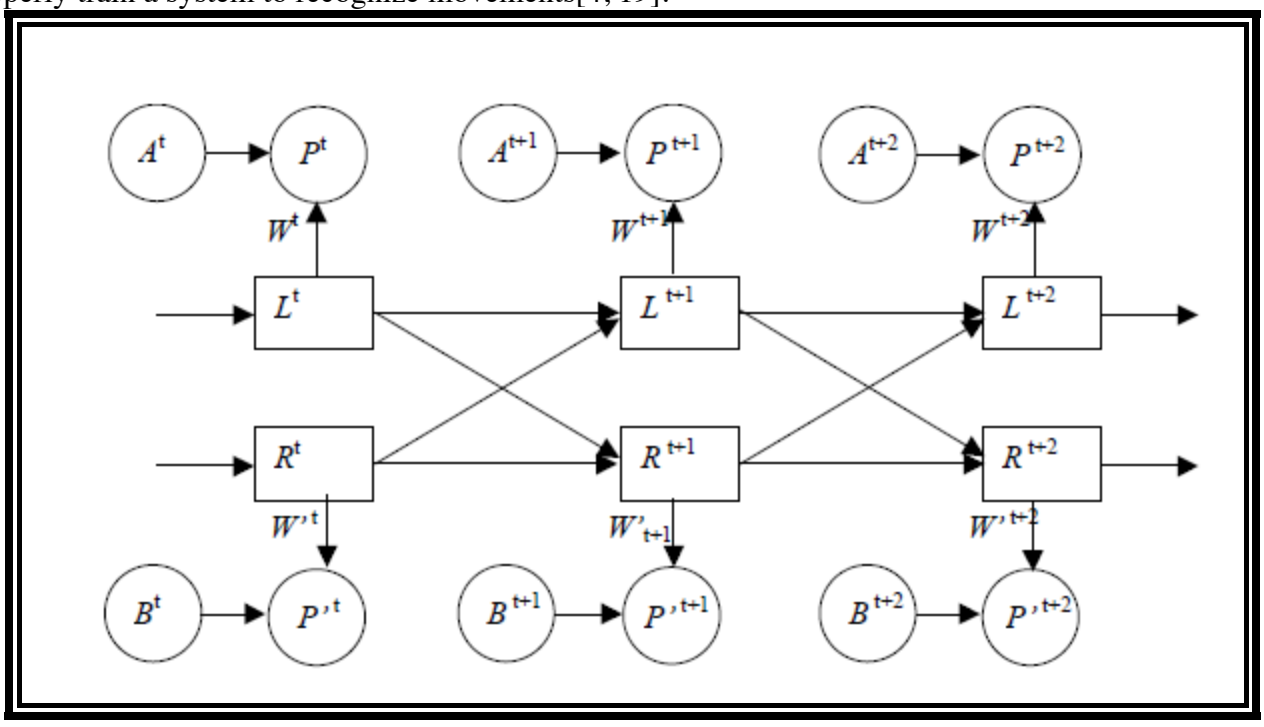


Figure 1.8: Primitive-based Human Markov Model State Representation [20]

1.2 JESSE ALGORITHM DESCRIPTION

This thesis utilizes research performed by previous researchers [22]. In general the previous effort developed an algorithm (JESSE) that generates the location of image change via a set of image powers collected by the Gaussian filtering of a reference and a test image. The procedure is captured in Figure 1.9.

JESSE obtains five image power values for each image, one power that simply sums the intensities of the image pixels and four Gaussian filtered powers which are generated from summing the pixel intensities that result from convolving non-concentric Gaussian filters with each image. Then via a process of inverse filtering and triangulation, these powers are used to locate scene changes. The outcome of this process is an x and y coordinate relative to the positions on the image. These procedure can be used for multiple applications such as a virtual mouse or PC Board chip detection, which are further discussed in detail later in this thesis.

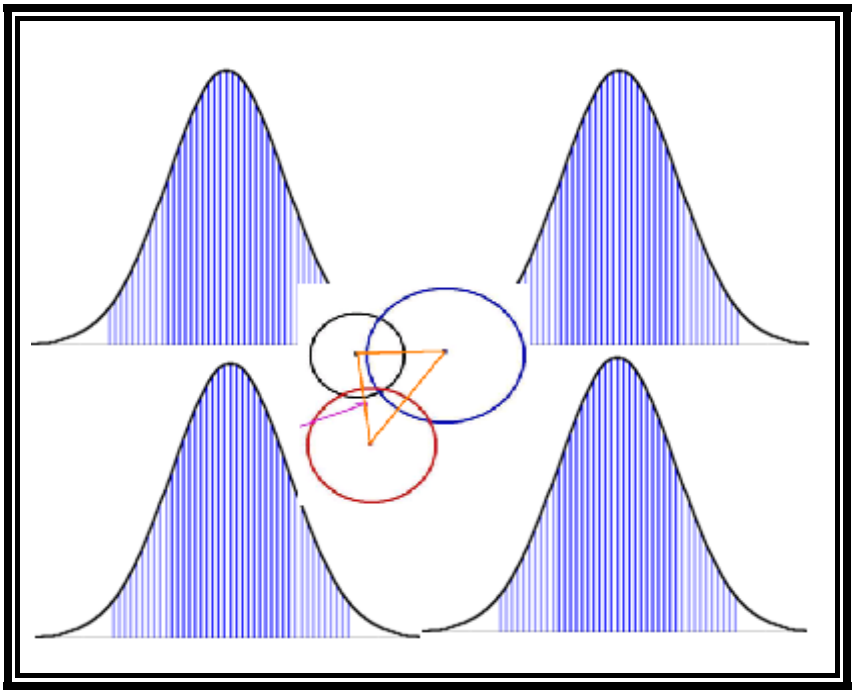


Figure 1.9: Generalized JESSE Algorithm

Chapter 2: Testing Platform and Research Development

The platform design took into consideration the need for an efficient and replicable product. The platform had certain requirements such as mobility, versatile of lighting positioning and user accessibility.

2.1 PROCESSING COMPONENTS

The platform consists of three main processing components a C328 UART camera, an FTDI Serial Converter, and a laptop with the needed software to process the algorithm. The C328 camera was chosen because of its complete control during the image capturing process, giving the developer full access to the internal programming of the camera.

The C328-7640 is a VGA camera module that performs as a compressed JPEG still camera and can be attached to a wireless or PDA host. Users can transmit a snapshot command from the host in order to capture a full resolution single-frame still picture. The picture is then compressed by the JPEG engine and transferred to the host thru a serial port.

The Features of the camera are as follows:

- Small in size, 20x28mm
- VGA resolution, down sample to QVGA or CIF
- 3.3V operation
- Low power consumption 60mA
- User friendly commands to control the module
- UART interface of up to 115.2Kbps
- Auto detect baud rate and make connection to the host
- Power saving mode
- Various lens option

The C328 has the following physical properties as shown in Figure 2.1.

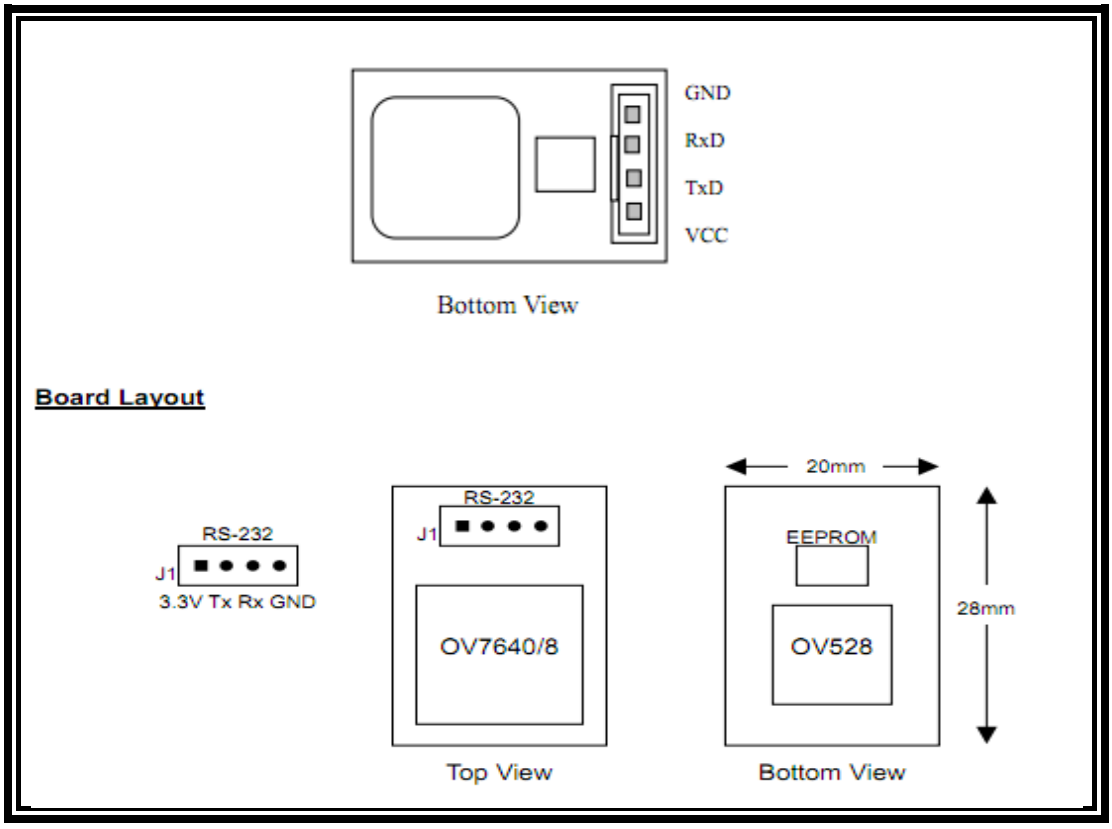


Figure 2.1: C328 UART Camera Dimensions and Connections

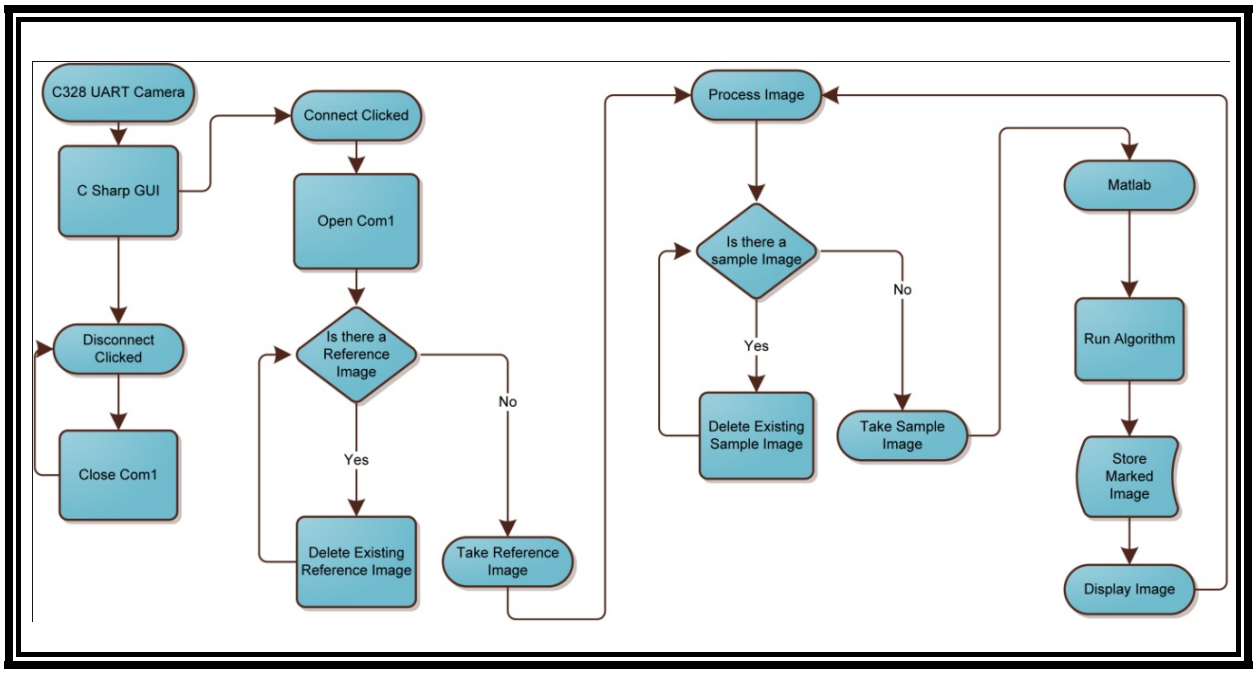


Figure 2.2: GUI Complete Flowchart

In order to communicate with the C328 the process depicted in Figure 2.2 was performed utilizing Visual Studio 2010. The GUI with which the user can directly control the entire process is seen in Figures 2.3 and 2.4. Figure 2.4 shows an image processed to display the position of a hand against the reference blue background.

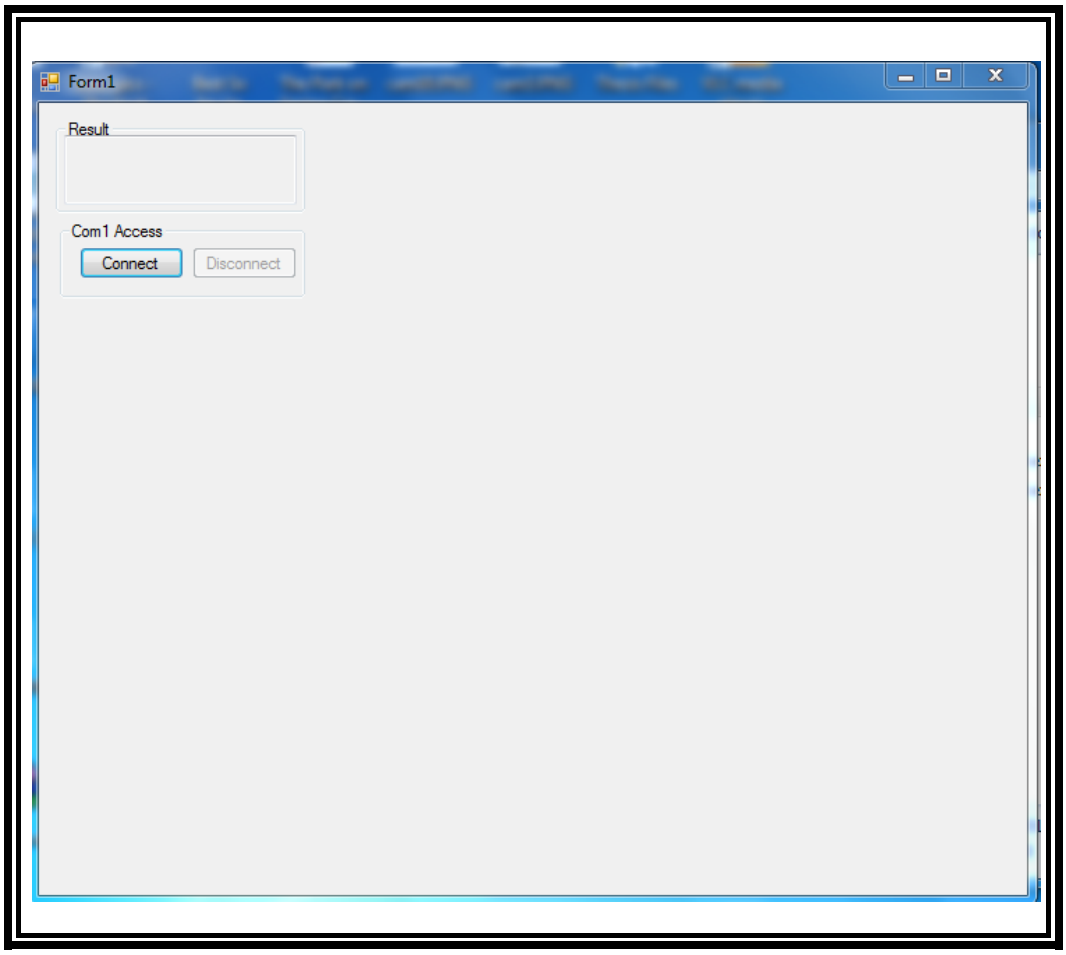


Figure 2.3: GUI used to provide feedback to users

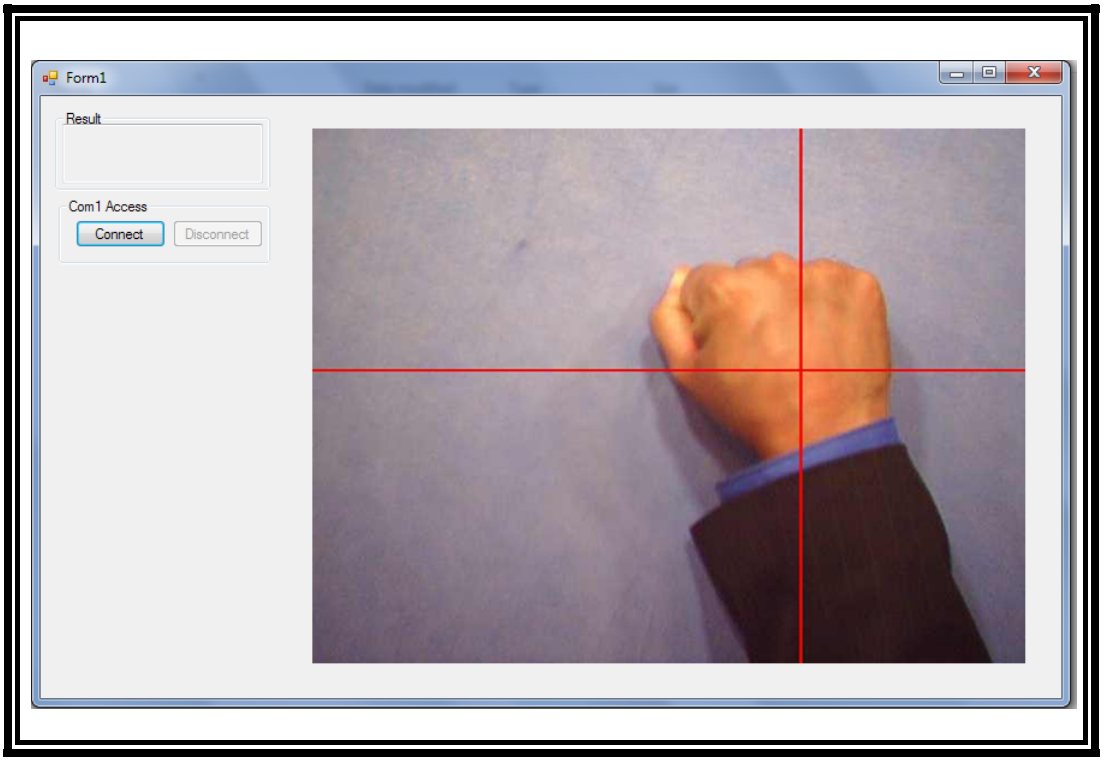


Figure 2.4: Output as it is displayed in the GUI

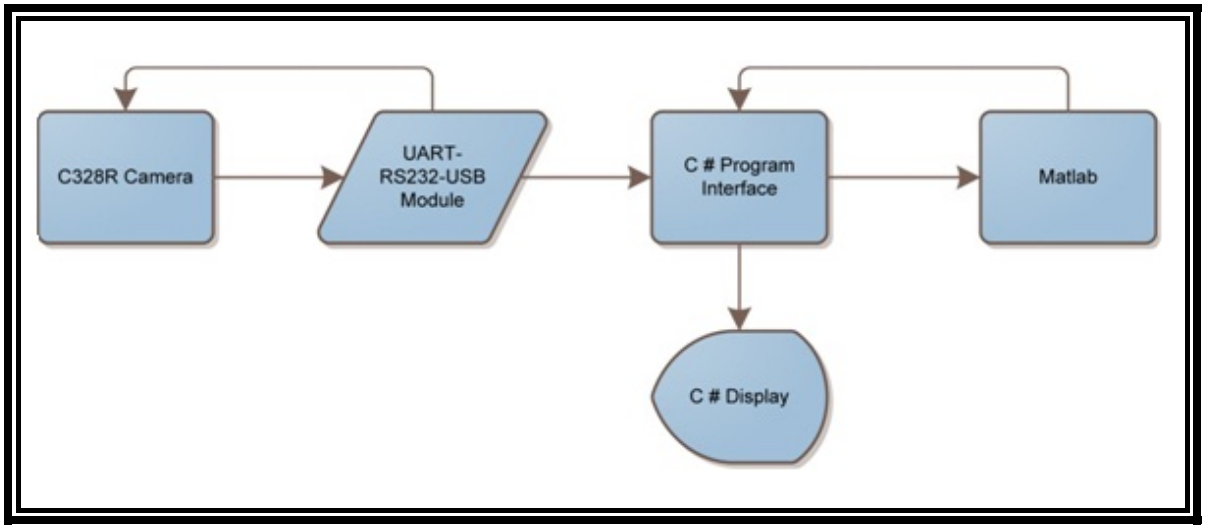


Figure 2.5: Overview of Platform Processing

The Image capturing alone would not be enough process the images thus the program also invokes MATLAB 2009 in order to run the JESSE Algorithm. A simplified overview of the complete

process can be seen in Figure 2.5. Figure 2.6 displays the hardware components utilized, i.e. the C328 Camera sends images through the FTDI UART to USB communication port on the laptop. The image data is processed via the laptop. These coordinates may be sent to external devices.

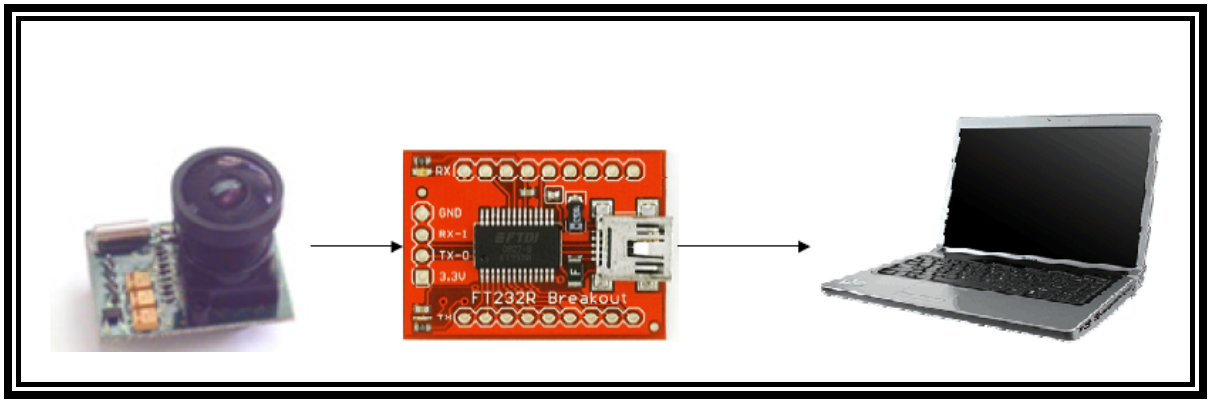


Figure 2.6: Hardware Module Interfacing

2.2 PLATFORM STAGES OF DEVELOPMENT

The design was performed in the following stages:

- Research stage: Versatility as well as portability requirements of the platform were assessed. The use of a sturdy but moveable stage with a beam assembly to hold the camera would allow for maximum portability and a sufficiently steady structure for acquiring constant images. Appropriate lighting, as is a requirement in all vision-based studies, is crucial to gathering usable data. Thus, a flexible lighting configuration that allowed for various light bulb types as well as positions of the lights and using the same beam assembly as the camera was conceptualized.
- Design stage: A heavy projector cart was selected to provide a relocatable but yet camera-stable platform. A wooden imaging surface covered with a surgical gown or similar was proposed in order to reduce ambient background image noise. Both LED and incandescent lighting with clamps were proposed as lighting sources. These along with the camera and any communications interfaces would be hung off of beams.

- Construction stage: The construction stage implemented the concept generated in the design stage. An image of the platform is visible in Figure 2.7. The beam assembly was built from 1/8th inch metal parts easily purchased at a hardware store. This frame has pre-cut notches and allows straightforward add-ons such as different placements of the camera and lights, and multiple implementation of the setup.
- Final Testing/Calibration stage: Due to imaging requirements assessed upon use, the platform was altered several times (discussed in detail later). These changes included camera lens filtering and lighting changes to produce improved images. The filtering was performed with colored transparencies. These filters reduce both shadowing and noise issues. With improved lighting, the filtering was eventually removed and viable image samples were produced.

Figure 2.7: Developed Platform with Lighting, Camera and Stage

2.2.1 Lighting and Camera

Illumination in computer vision must take into account ambient noise and shadowing considerations. There were three types of illumination tested throughout the construction of the platform. The first was standard cabinet lights (see Figure 2.8), lighting commonly used in many buildings to provide indoor illumination. A resulting pair of background images along with analysis of their difference can be seen in Figure 2.9. This analysis reveals that the images captured the frequency at which the lights oscillate. Obviously, this can be a source of distortion in the image's processing.

The second approach at illumination was the use of LED lighting (see Figure 2.10). Examples of the resulting images versus background fluorescent lighting along with the analysis of their difference can be seen in Figure 2.11. This analysis shows that some areas are exposed more brightly than others; thus also potentially creating erroneous data, especially when object movements are more pronounced.

Figure 2.8: Cabinet Lights used as first illumination test

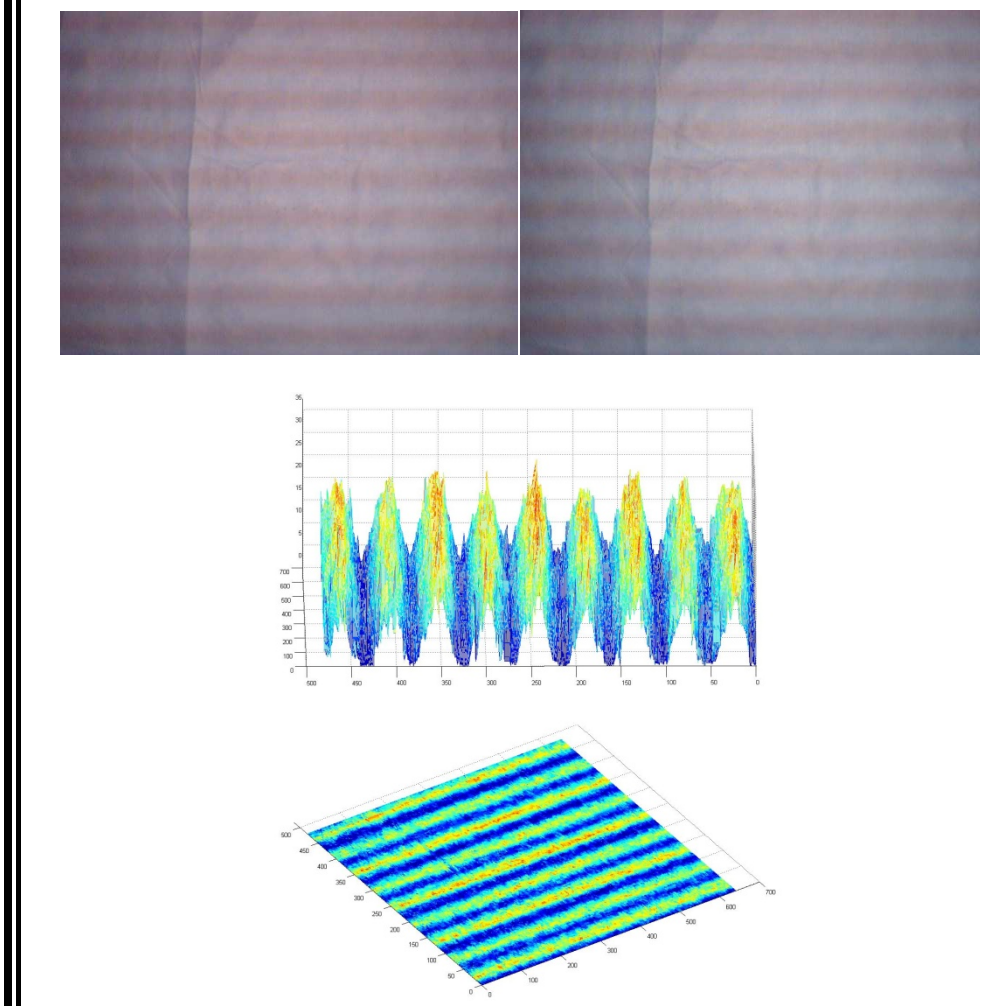


Figure 2.9: Images captured using Cabinet Lights where distortion is visible

Figure 2.10: LED lighting used to remove shadowing effect

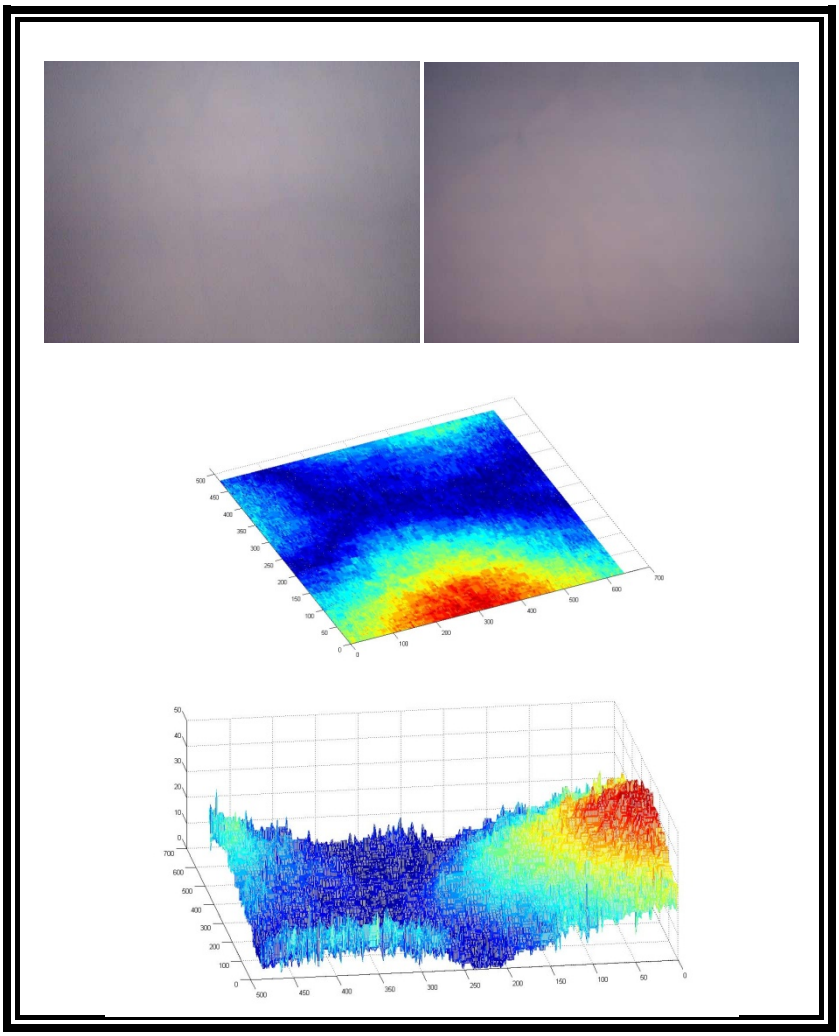


Figure 2.11: Resulting Images from using only LED lighting

The final approach at illumination was to use incandescent light bulbs along with LED lighting (see Figure 2.12). As revealed in the analysis in Figure 2.13, as compared to ambient background light (and the above lighting approaches), this may result in more even lighting throughout the background.

Different lighting configurations (depending on the object being tested) were setup in order to provide overall best lighting and to reduce shadows cast onto the background. The lighting consisted of four to six lights, depending on the application. The lights are held in place by clamps allowing the user to alter the lighting placement.

For the surgeon's hand application, six lights in a row at the back of the table (see Figure 2.12) provided the best overall effect and removal of shadows. The PC Board Lighting setup consisted of 4 LED lights at the corners of the platform (see Figure 2.14). In the latter since there is more limited movement of the test objects, lighting hot spots were not an issue. However, this was not the case for the former.

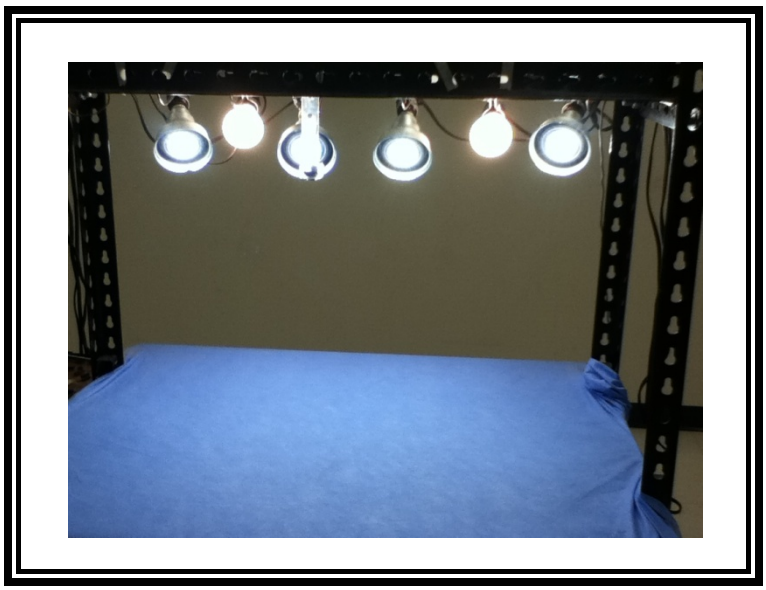


Figure 2.12: LED and incandescent lighting combination

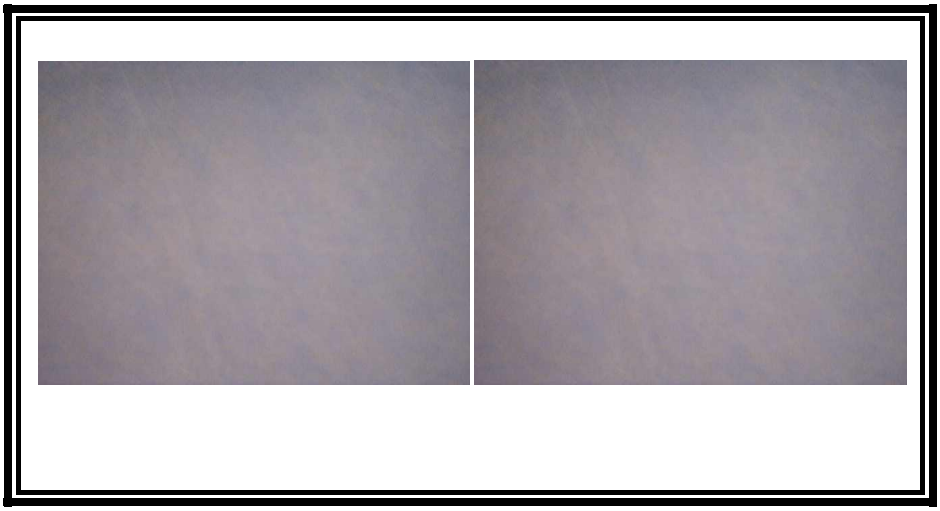


Figure 2.13: LED and incandescent lighting Images and Analysis

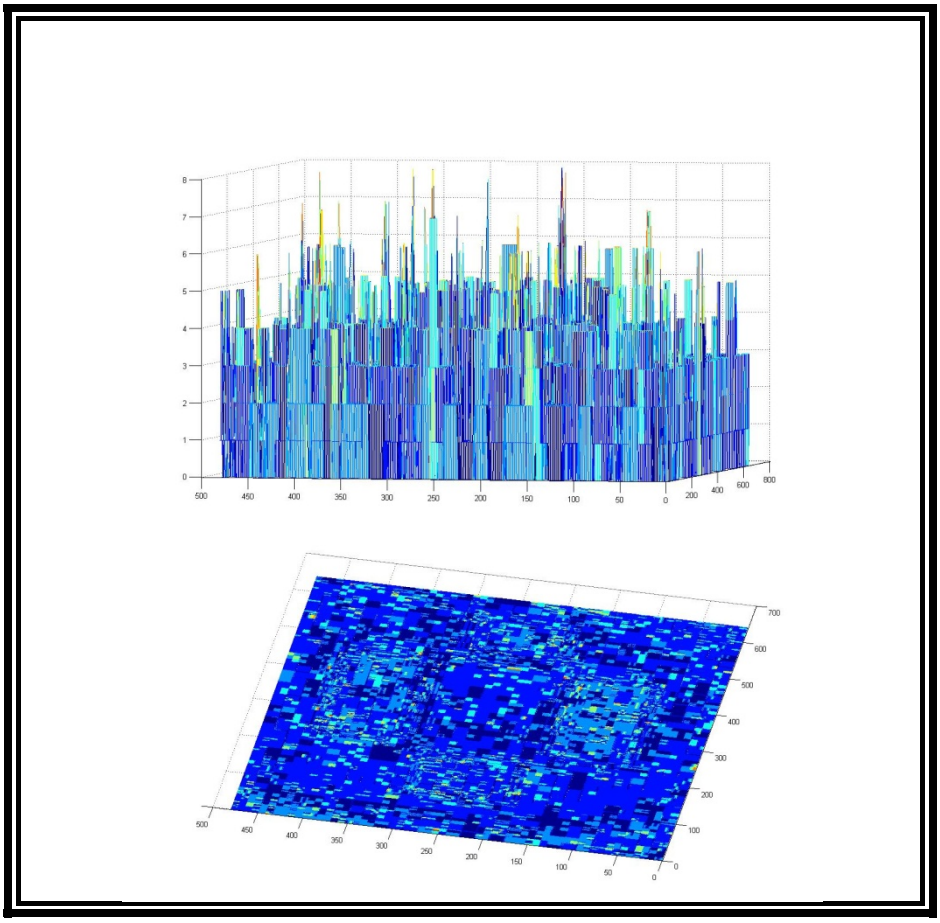


Figure 2.13(cont): LED and incandescent lighting Images and Analysis

Different lighting configurations (depending on the object being tested) were setup in order to reduce shadows cast onto the background. The lighting consisted of four to six lights, depending on the application. The lights are held in place by clamps allowing the user to alter the lighting placement. For the surgeon's hand application, six lights in a row at the back of the table (see Figure 2.12) provided the best removal of shadows. The PC Board Lighting setup consisted of 4 LED lights at the corners of the platform. These pushed shadows away from the center. The latter lighting setup is shown in Figure 2.14.

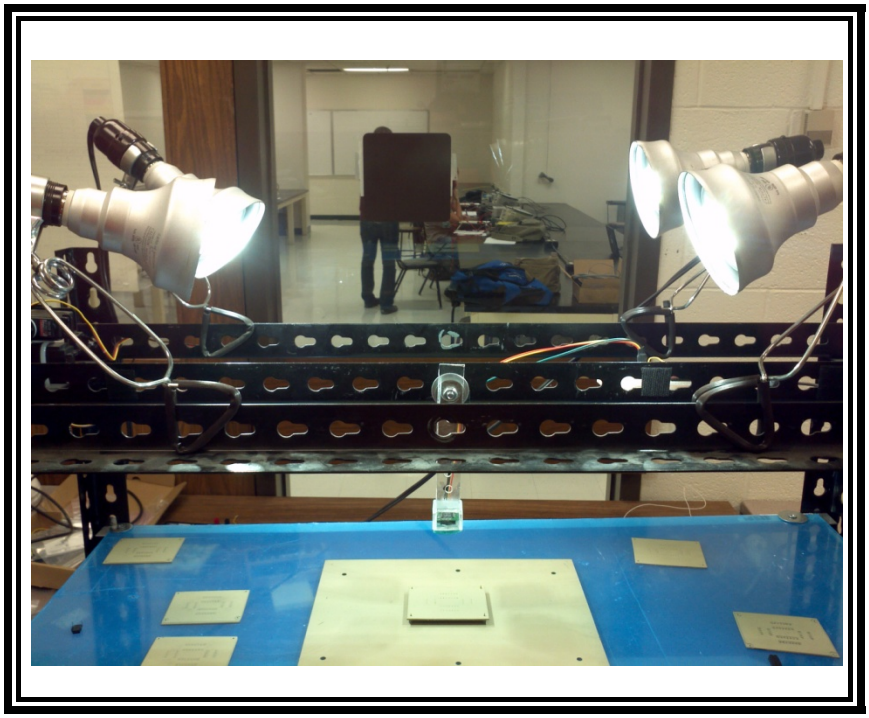


Figure 2.14: Platform setup for the PC Board Implementation

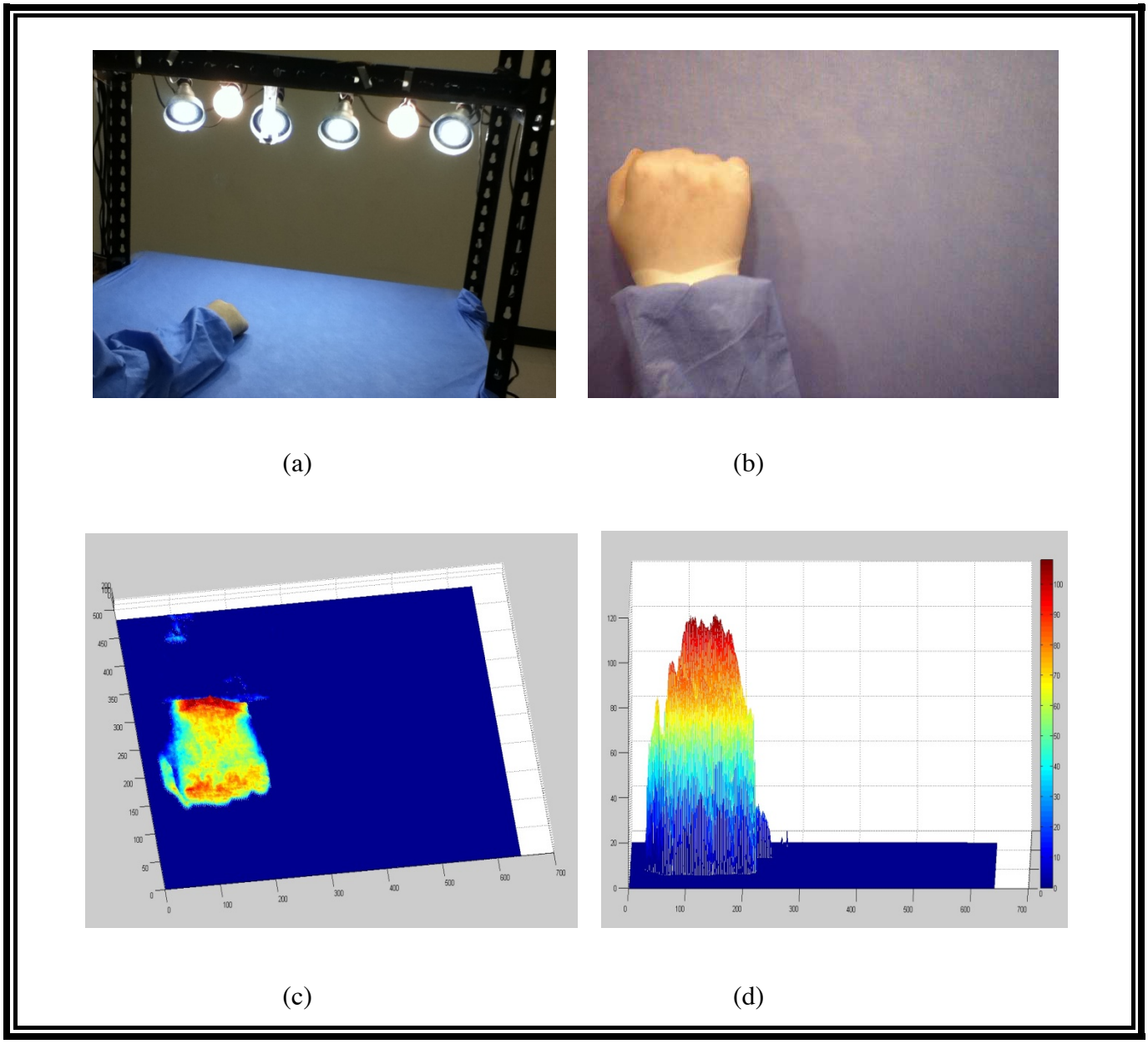


Figure 2.15: Appropriate lighting and its benefit. (a) Example of surgeon's arm placement. (b)Image of the surgeon's gloved hand and arm over a background gown, (c) Resulting analysis of the hand (described latter), and (d) intensity view of the surgeon's hand in (c).

2.3 Light Filtering and Hardware Noise Reduction

Light filtering can be done both in software and in hardware but in order to reduce processing overhead it is beneficial to use minimal software filtering. Thus, filtering was conducted using different color transparencies placed in front of the camera lens. These filters removed both shadowing and minor

lighting differences and Table 2.1 gives a description of the testing results with the C328 UART camera using different transparency material.

The difference between background noise intensity and object intensity is an indication of how well an object stands out from the background noise. This latter contrast information can be gathered by subtracting a background image from one that possesses an object of interest. Further, the larger the intensity difference, the better the contrast for object detection. Such contrast data for the present design where two sheets of green transparencies were used can be seen in Figure 2.16. Results for various combinations of transparencies can be seen in Table 2.1. The best results were produced via the use of two green transparency sheets. Examples of images showing the effects of the three color filter types on the camera can be seen in Figures 2.16 to 2.18.

Table 2.1: Image Filtering Results

Filter Material	Color Spectrum Analysis					
	Red		Green		Blue	
	Noise Level	Object Intensity	Noise Level	Object Intensity	Noise Level	Object Intensity
Single Red	24	35	25	70	30	30
Double Red	25	40	25	36	30	30
Triple Red	45	45	50	50	52	52
Single Green	40	60	40	40	30	30
Double Green	15	75	6	35	15	15
Triple Green	15	50	6	18	15	15
Triple Blue	30	80	30	40	35	35
Single Red Single Green	15	60	20	25	25	25

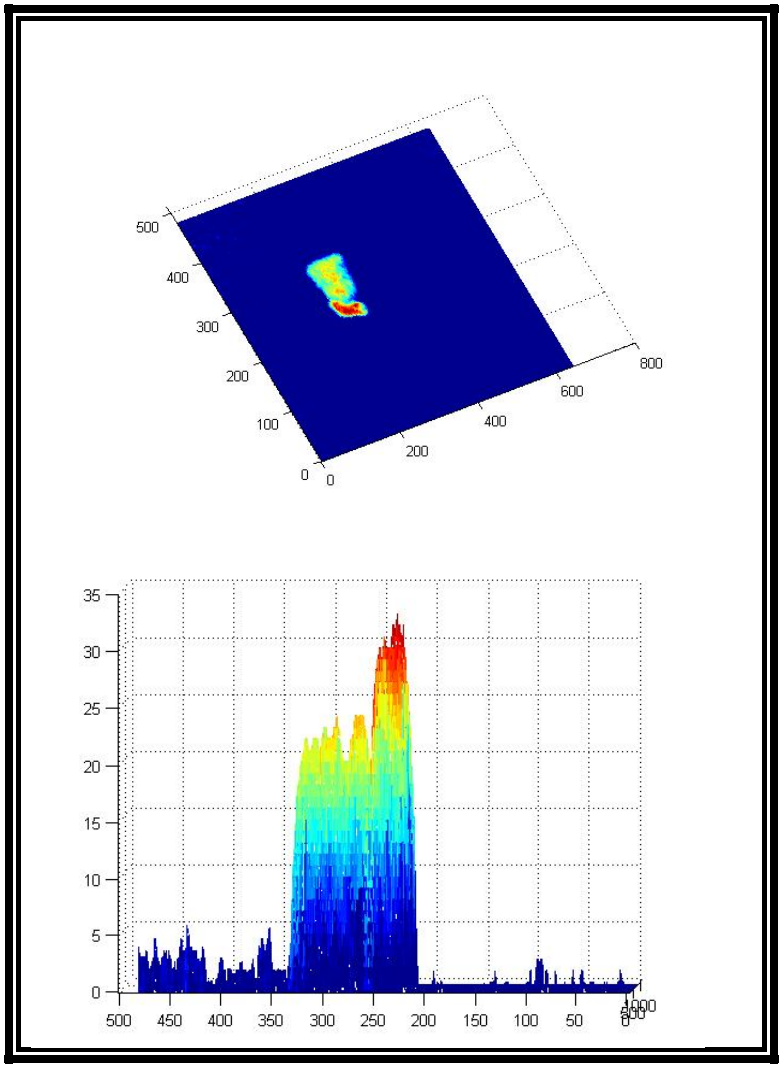


Figure 2.15: MATLAB Analysis using 2 Green Transparencies

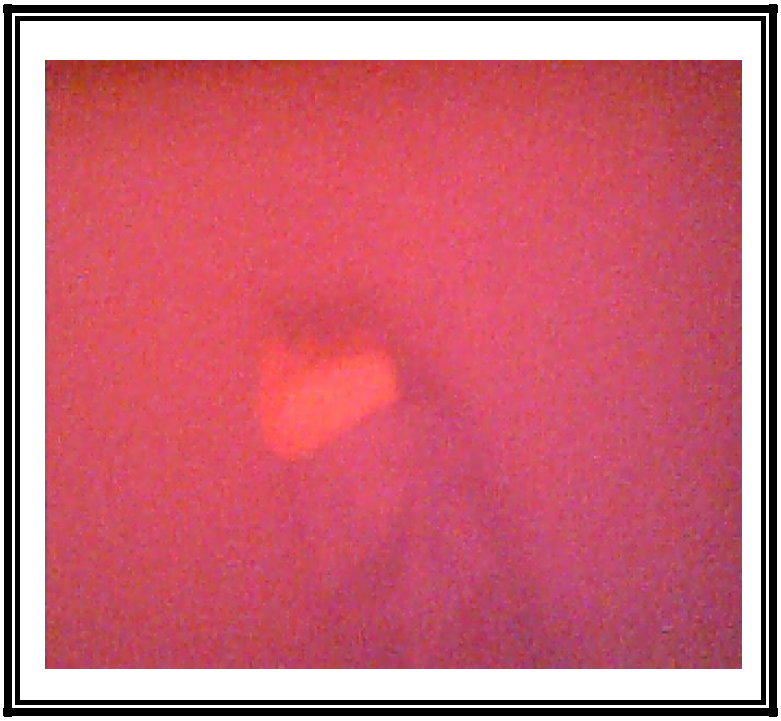


Figure 2.16: Red Filtered Image

Figure 2.17: Green Filtered Image

Figure 2.18: Blue Filtered Image

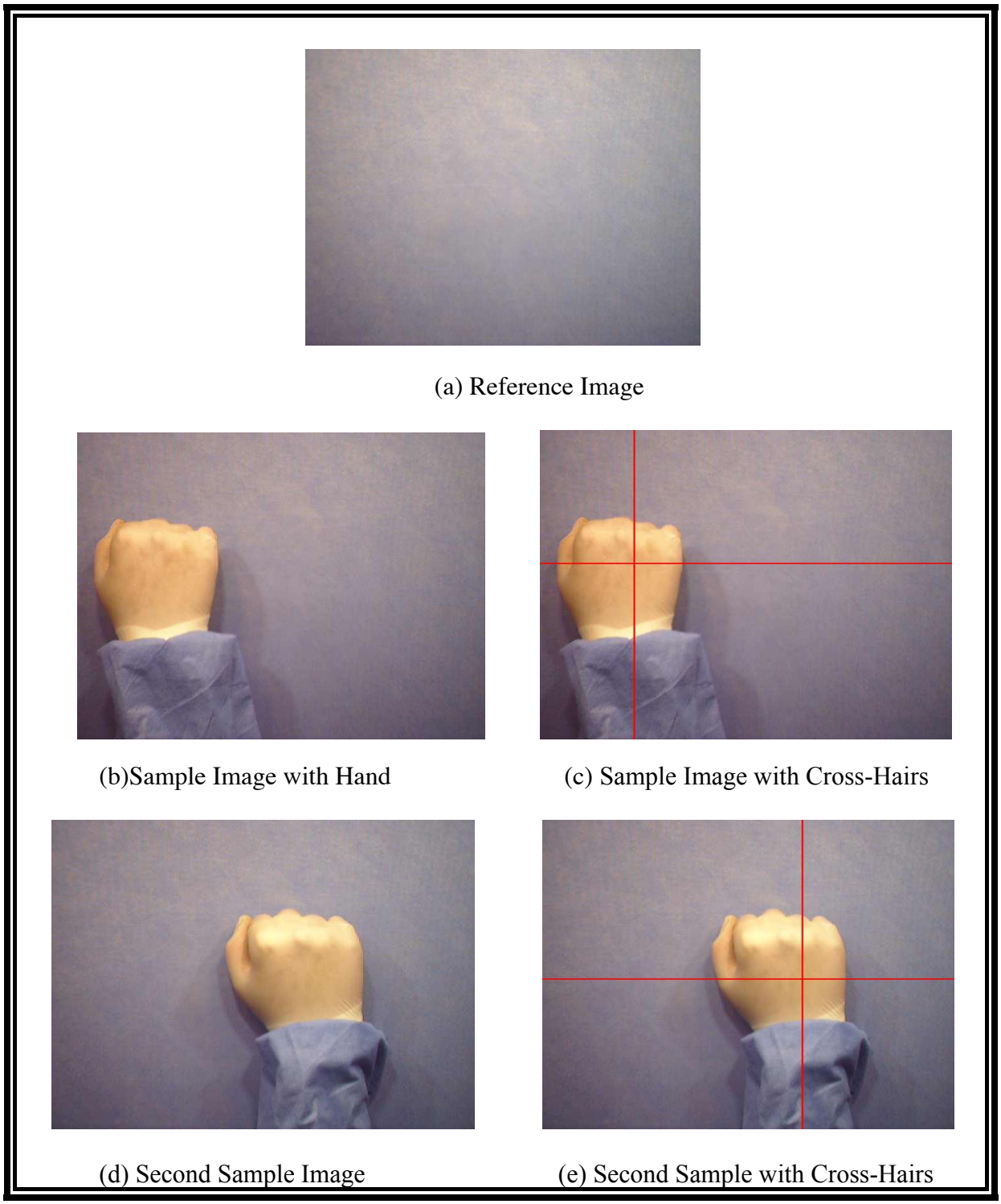
Chapter 3: Testing Platform and Research Development

The resulting platform was tested for use with both the surgeon's hand and PC Board applications over a development period of two years. With the system being reliable enough to track the relative position of the hand and missing chips on a PC Board, several other implementations have been possible.

3.1 HAND DETECTION

To consistently detect the center of a surgeon's hand, proper setup of a background and lighting was required to give good enough contrast and lack of interfering shadows. It was found that a surgeon's gown as background gave good results. Further, six lights in a row at the back of the table (see Figure 2.7) provided the best removal of shadows.

Figure 3.1 shows images, which capture the results of this application after JESSE processing (see section 1.2). The light filtering consisted of the use of a double green filter (transparencies). The resulting blue channel filtering is depicted in Figure 3.2. The hand was consistently detected but its detected position is not always placed in the same position relative to the hand (see Figure 3.3).



(a) Reference Image

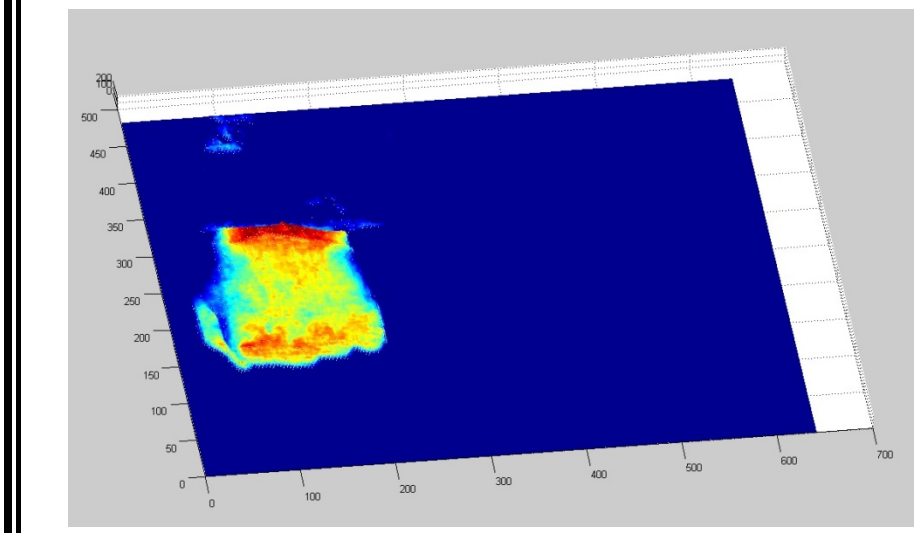
(b)Sample Image with Hand

(c) Sample Image with Cross-Hairs

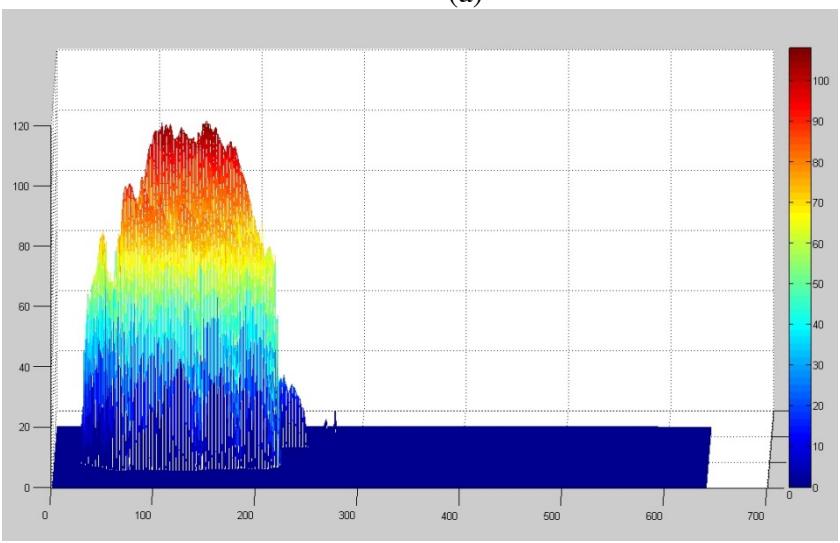
(d) Second Sample Image

(e) Second Sample with Cross-Hairs

Figure 3.1: Image used as a reference (a) and samples of a hand (b and d) as well as result images (c and e). The detected location of the hand in marked with the crosshairs on the image.



(a)



(b)

Figure 3.2: Blue-color plane view of hand. Top (a) as well as side (b) view of hand as seen in MATLAB.

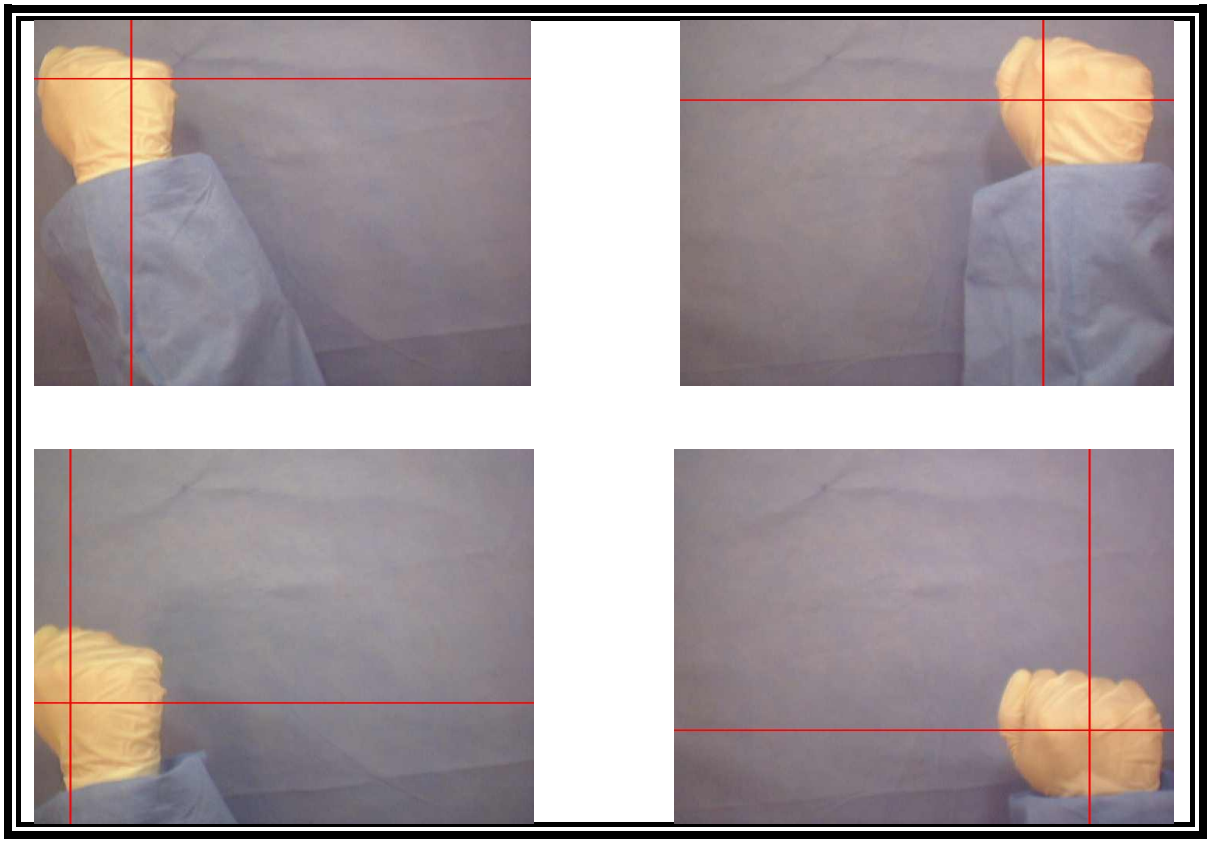
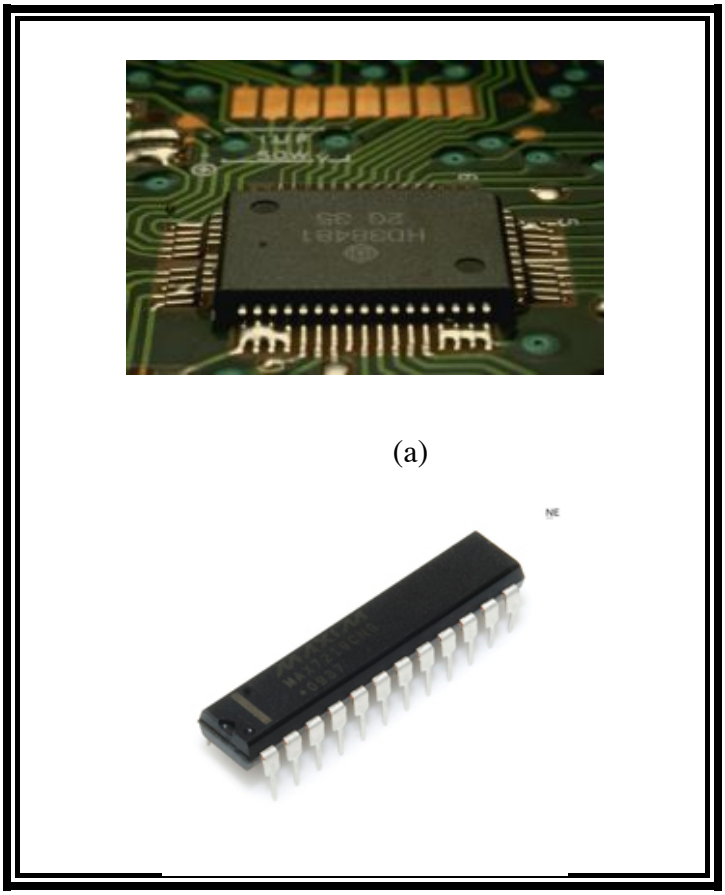


Figure 3.3: Hand placed on the 4 corners to test edge detection

3.2 PC Board Detection

At present printed circuit boards are typically populated using surface mount chips. An example is shown in Figure 3.4a. However, to facilitate the present work, it was decided to use dual in-line pin type (DIP) chips (see Figure 3.4b) to ease the process of placement/removable. The higher profile nature of the latter chips along with their less certain placement brought about a challenge in image acquisition, resulting in noise (see lower part of Figure 3.5). This noise was also in part due to the PC Board application's lighting setup. The best case light placement in this case consisted of 4 LED lights at the corners of the platform, resulting in some shadows. Chip labels and other minor differences could also be a noise source. However, to simplify the process, these were removed



(a)

Figure 3.4: PCB components. (a) Surface Mount Chips used on PCBs and (b) DIP Chips used for testing purposes.

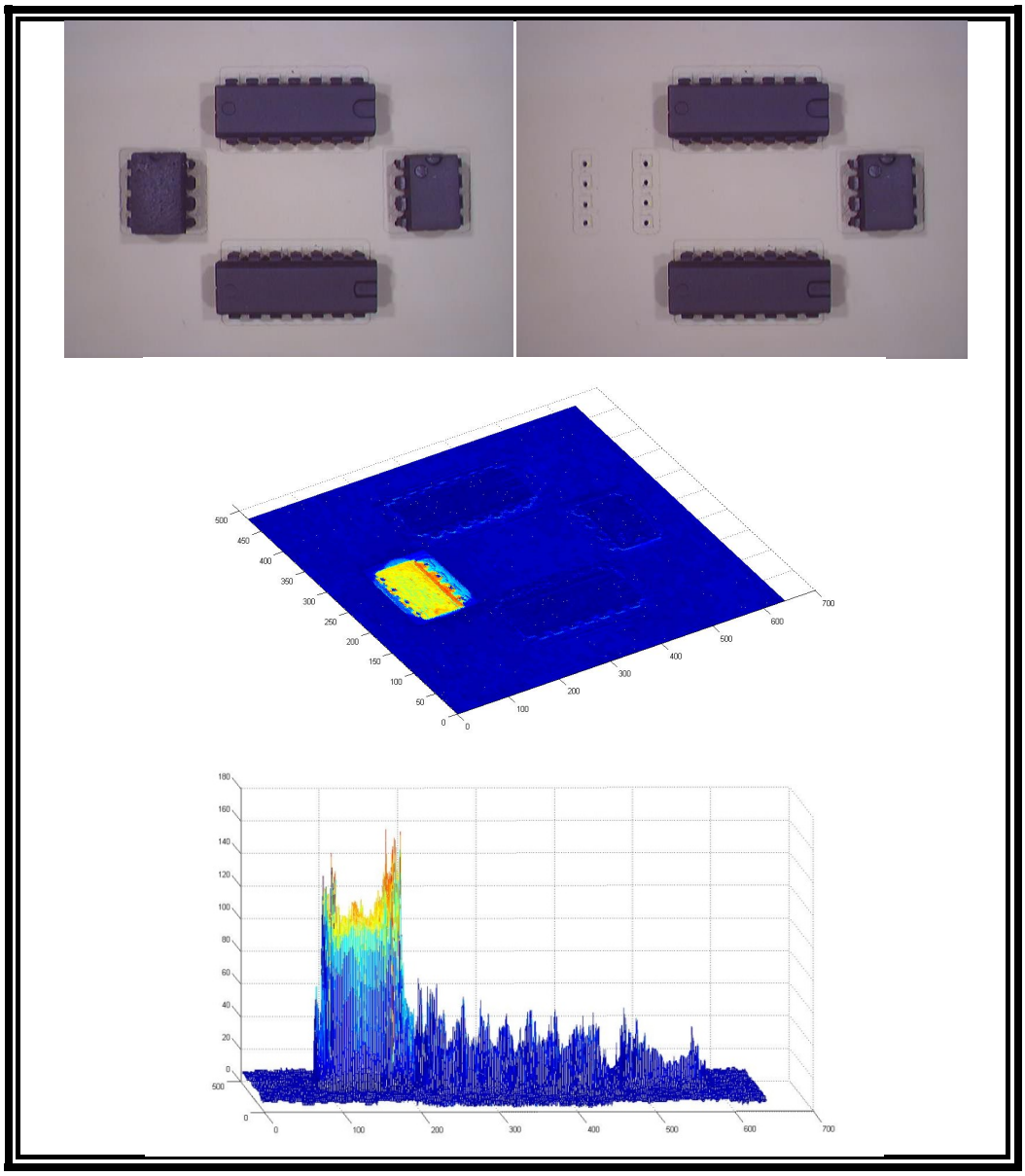


Figure 3.5: MATLAB Analysis of high resolution focused images and the noise detected

In order to compensate for such noise, the camera lens may be defocused. An example of the benefit of defocused images can be seen in Figure 3.6 where the noise level is diminished greatly as compared to the previous focused attempts in Figure 3.5.

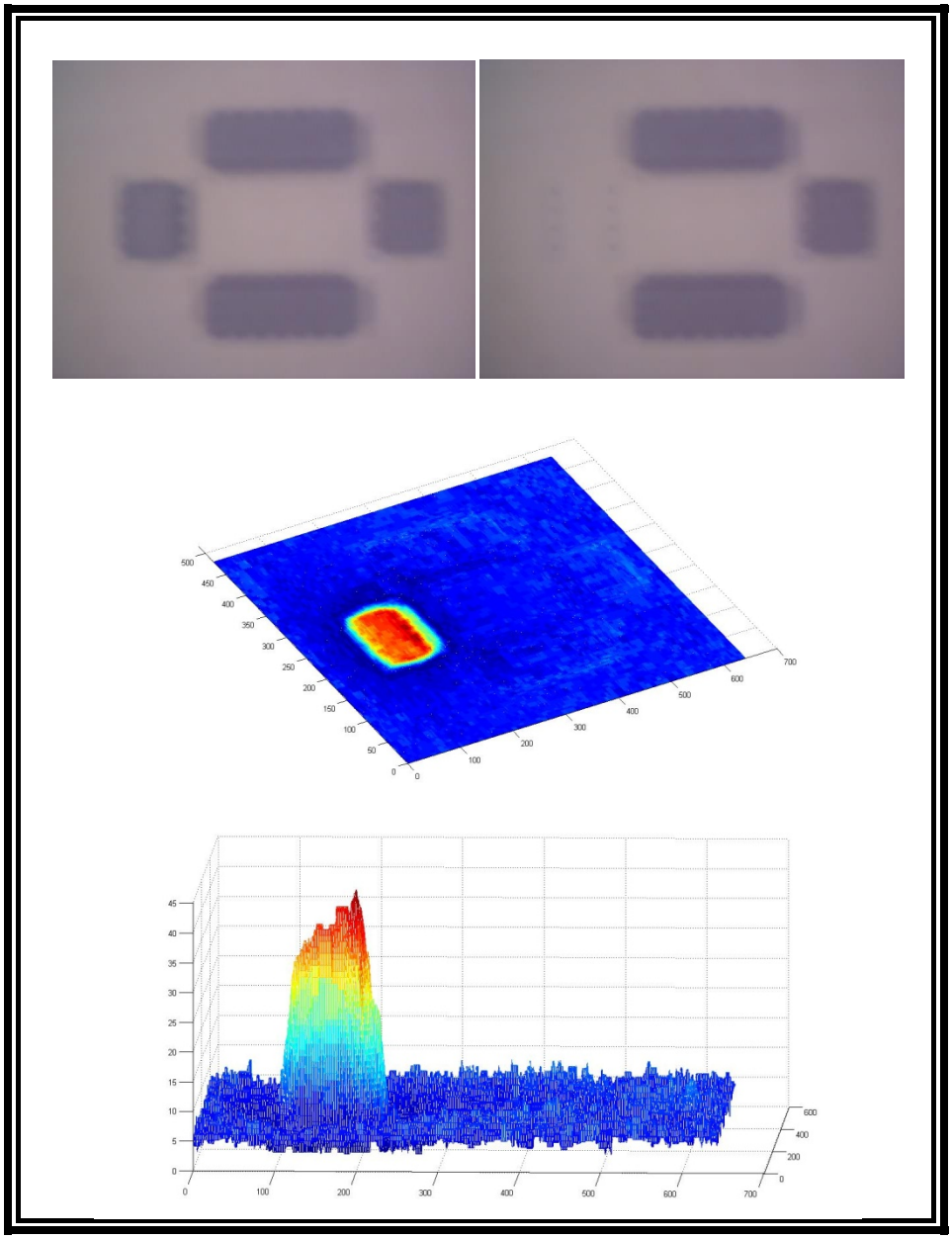


Figure 3.6: Defocused Images which provide noise reduction from through hole DIP Chips

The PC Board application's lighting setup consisted of 4 LED lights at the corners of the platform. Since chips have labels and other minor differences, it is beneficial to defocus the camera. With the platform and lighting configuration described, one can also show that under less than ideal image conditions, a fuzzy camera, it is possible to detect a missing chip between two different boards.

Figure 3.7 shows a reference image of a PC Board with 4 chips properly placed and a second board with the same setup but a missing chip. The system was able to detect the missing chip and its

general location even though the resolution on the camera is set to one of the lowest settings and does not clearly define the chips.

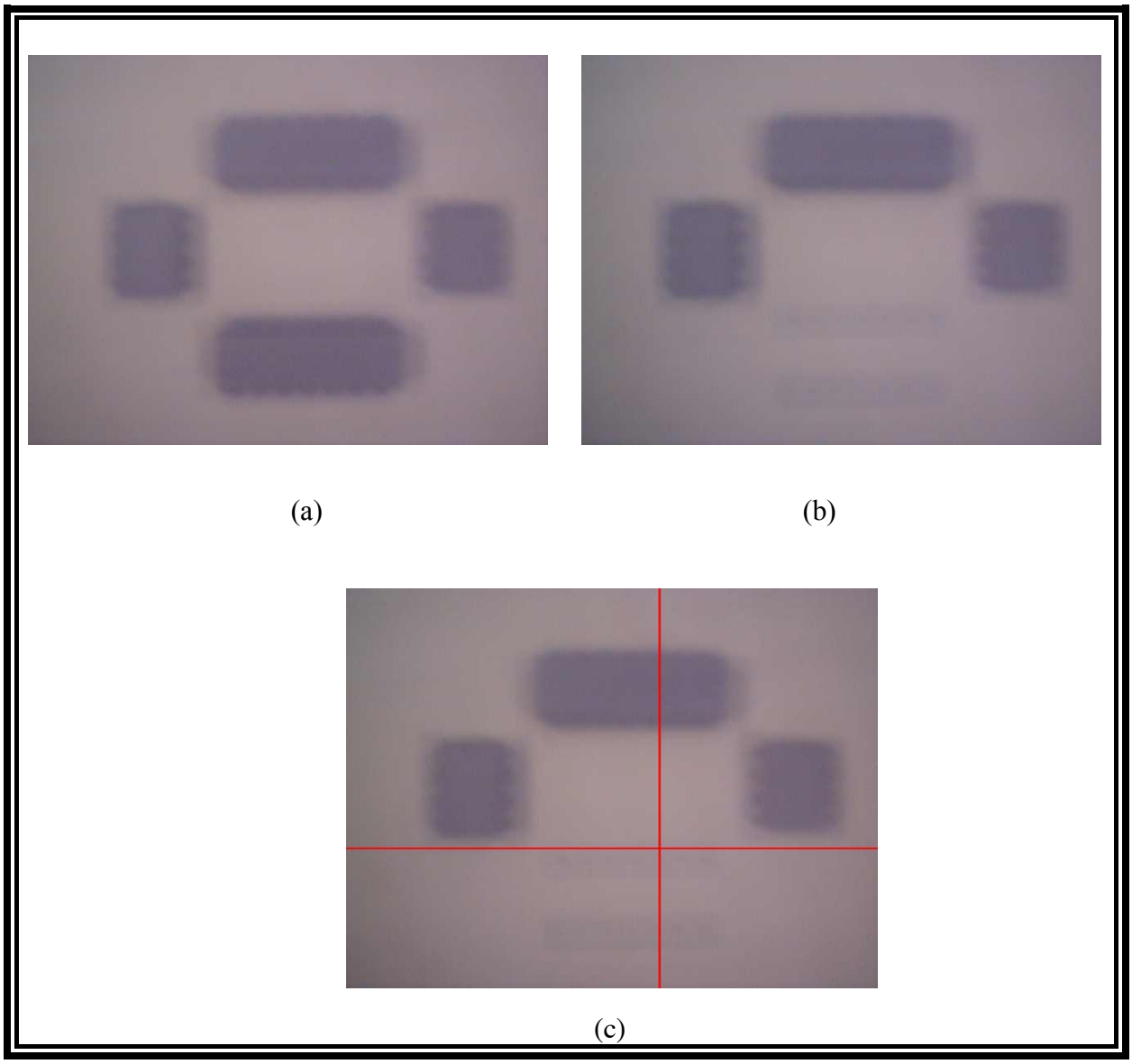


Figure 3.7: PC Board Application. (a) Reference with 4 chips present, (b) a second board with a missing chip and (c) the resulting outcome after running through the system.

3.3 EXTENSIONS

Two extensions of the surgeon's hand application have been conceived: a virtual mouse and a pan and tilt control on a second camera. As the surgeon's hand moves across the background, the x and y coordinates of the hand are constantly available. This process gives a location that could be used for

another application. For example, the coordinates could in turn be used as a base of operation for a microprocessor to control a 16x16 dot matrix display or the pan and tilt of two servos.

The microprocessor used to show these extensions was an Atmega16 with the properties described in Table 3.1. This microprocessor was part of a PC Board designed to function as a multiuse tool. This board contains a terminal block to allow its pins to be safely accessed by external components. The PC Board was designed using Proteus Professional and built in the Electronics shop of the Electrical and Computer Engineering Department at UTEP. The schematic of the design is shown in Figure 3.8. The 16x16 dot matrix board that was interfaced to the Atmega16 board is shown in Figure 3.9.

Pan and tilt servos have been positioned on the platform as well and have been mapped to move depending on the location of the surgeon's hand. One servo moves a flat surface up and down while the other servo moves left and right. This allows for the surgeon's hand to control both movements at the same time while moving through the background.

Table 3.1: Parameters of Atmega16

Parameter	Value
Flash Memory (Kbytes)	16
Pin Count	28
CPU	8 Bit
SPI	1
TWI	1
UART	1
Max Operating Frequency	16 MHz
ADC Channels	6
PWM Channels	6

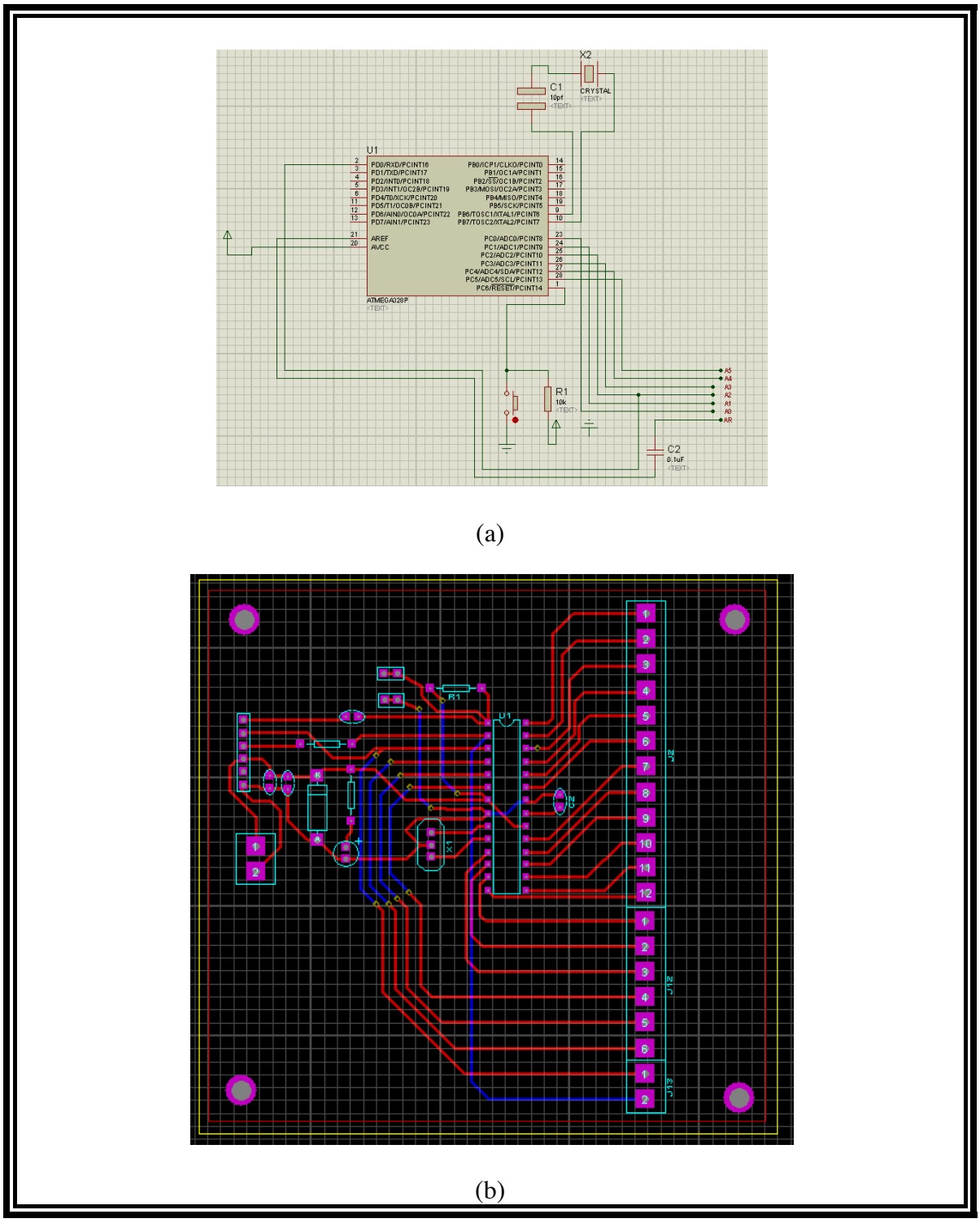


Figure 3.8: Atmega16 multiuse board. (a) Proteus Design of Atmega , (b) ARES PCB implementation of Design, (c) Pin configuration of Atmega16 and (d) the FTDI communication connection.

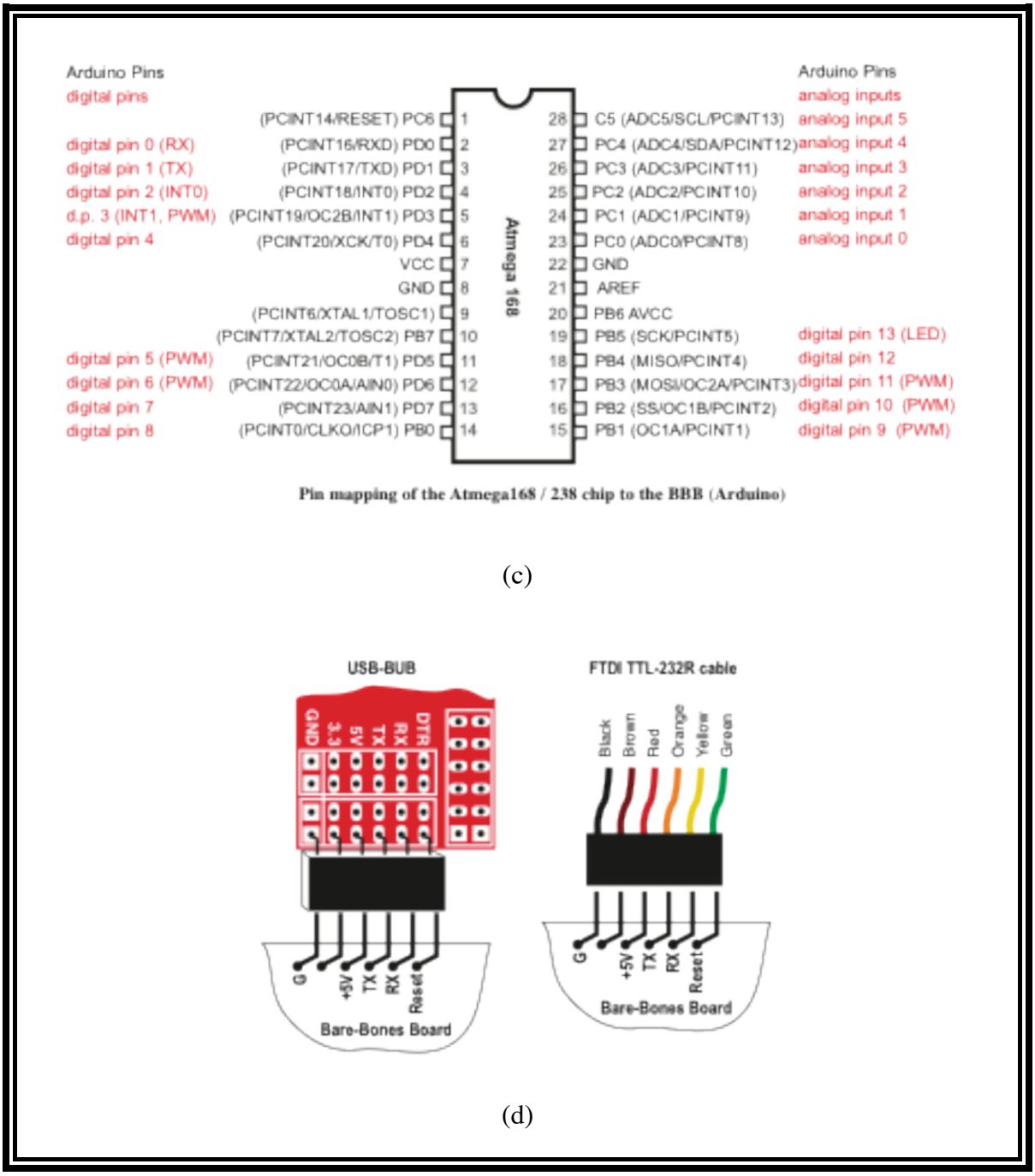


Figure 3.8 (cont.) : Atmega16 multiuse board. (a) Proteus Design of Atmega , (b) ARES PCB implementation of Design, (c) Pin configuration of Atmega16 and (d) the FTDI communication connection.

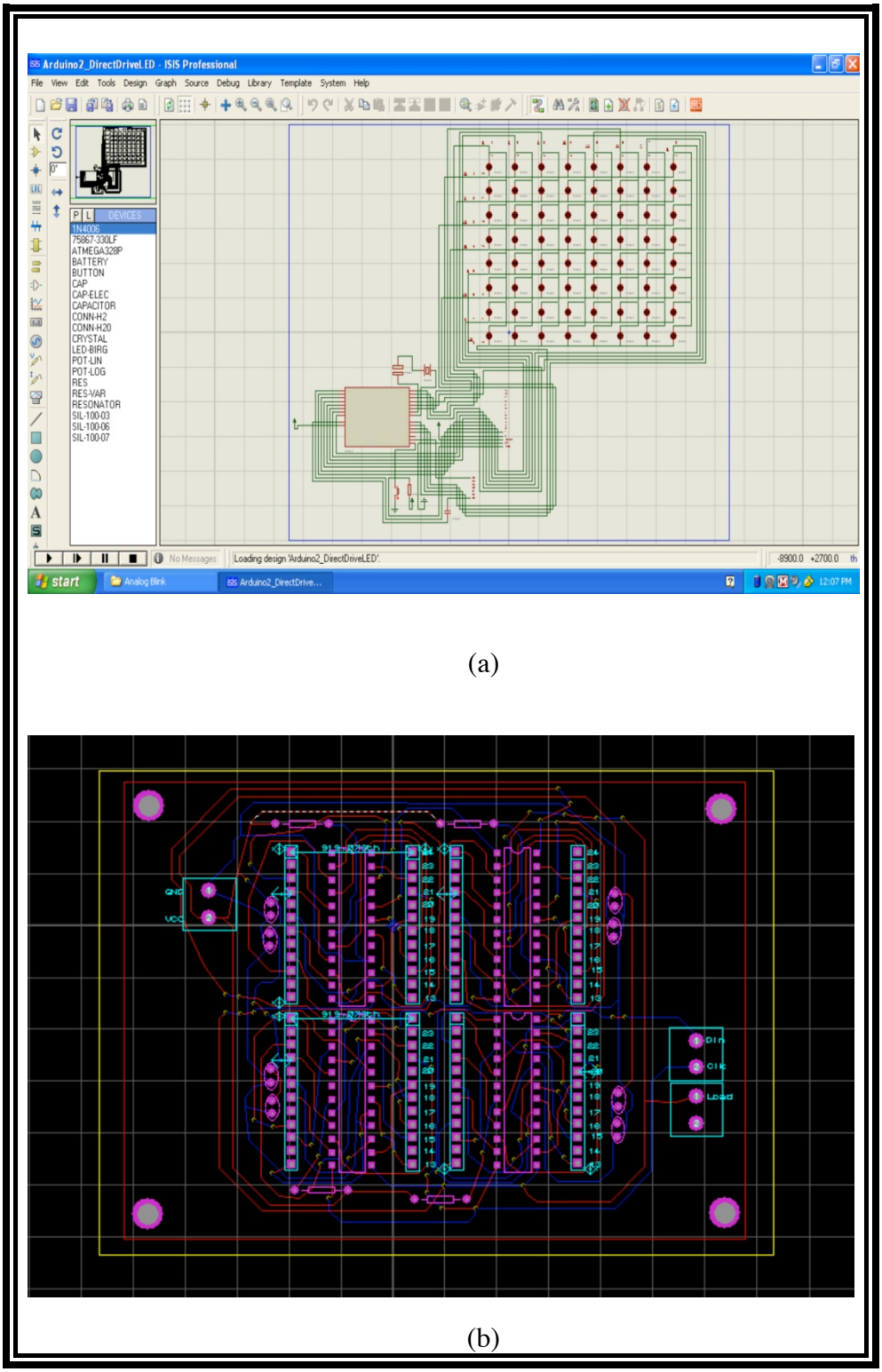
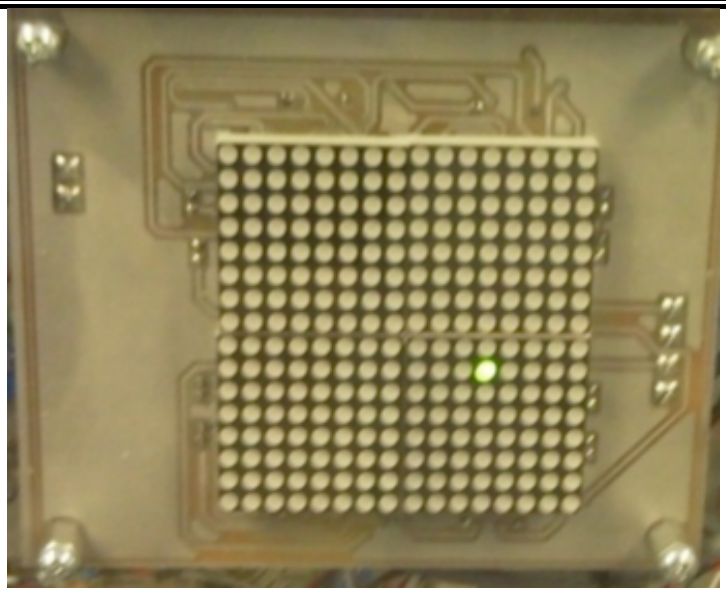


Figure 3.9: 16x16 Dot Matrix board. (a) Proteus Design of 16x16 Dot Matrix , (b) PCB Layout and (c) final product of Dot Matrix



(c)

Figure 3.9 (cont.): 16x16 Dot Matrix board. (a) Proteus Design of 16x16 Dot Matrix , (b) PCB Layout and (c) final product of Dot Matrix

Chapter 4: Conclusion and Future Work

4.1 Concluding Outcome

A simple and cost-effective system was presented that extends an approach for detecting changes in images. Possible applications in surgery and in PC Boards quality control were assessed, and it was shown that by utilizing appropriate lighting and/or filtering, the position of a surgeon's hand and missing chips on the PC Boards could be tracked. Further, it was possible to utilize the reference point given by the surgeon's hand to coordinate activity in external devices. Finally, this design implements techniques useful in visual recognition and motion detection without high cost and complicated programming.

4.2 Future Work

Future work will consist of converting the JESSE algorithm from MATLAB to C or C# and further extensions of the results. Some possibilities could be to locate a finger movement to change direction or indicate the click of virtual mouse or to create a device to take images of patients to detect potential skin cancer spots over a period of time.

References

1. Yun, F. and T.S. Huang. *hMouse: Head Tracking Driven Virtual Computer Mouse*. in *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*. 2007.
2. Xingfeng, W. and Q. Kaihuai. *A Six-Degree-of-Freedom Virtual Mouse Based on Hand Gestures*. in *Electrical and Control Engineering (ICECE), 2010 International Conference on*. 2010.
3. Li, W., C. Deng, and Y. Lv. *Implementation of virtual mouse based on machine vision*. in *Apperceiving Computing and Intelligence Analysis (ICACIA), 2010 International Conference on*. 2010.
4. Argyros, A. and M. I.A. Lourakis, *Vision-Based Interpretation of Hand Gestures*

for Remote Control of a Computer Mouse. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 2006. **3979**(Computer Vision in Human-Computer Interaction - ECCV 2006 Workshop on HCI, Proceedings): p. 40-51.

5. Gai, Y., H. Wang, and K. Wang, *A virtual mouse system for mobile device*. MUM '05 Proceedings of the 4th international conference on Mobile and ubiquitous multimedia, 2005: p. 127-131.
6. Guoqing, X., W. Yangsheng, and F. Xuetao. *A Robust Low Cost Virtual Mouse Based on Face Tracking*. in *Pattern Recognition, 2009. CCPR 2009. Chinese Conference on*. 2009.
7. Ji, E., H. Yoon, and Y.J. Bae, *Touring into the picture using hand shape recognition* Proceeding

MULTIMEDIA '00 Proceedings of the eighth ACM international conference on Multimedia, 2000: p. 388-390.

8. Kocejko, T., A. Bujnowski, and J. Wtorek. *Complex human computer interface for LAS patient*. in *Human System Interactions, 2009. HSI '09. 2nd Conference on*. 2009.
9. Kumar, S., et al. *Vision based human interaction system for disabled*. in *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*. 2010.
10. Robertson, P., R. Laddaga, and M. Van Kleek, *Virtual mouse vision based interface*. IUI '04 Proceedings of the 9th international conference on Intelligent user interfaces, 2004: p. 177-183.
11. Palleja, T., et al. *Simple and robust implementation of a relative virtual mouse controlled by head movements*. in *Human System Interactions, 2008 Conference on*. 2008.
12. Malik, S. and J. Laszlo, *Visual touchpad: a two-handed gestural input device*. Proceeding ICMI '04 Proceedings of the 6th international conference on Multimodal interfaces, 2004: p. 289-296.
13. Van Kleek, M., P. Robertson, and R. Laddaga, *Kiosk application for multimedia enhancement of public spaces*. WSEAS Transactions on Information Science and Applications, 2005. **2**(8): p. 1150-1156.
14. Okada, Y., *IntelligentBox as component based development system for body action 3D games*. ACE '05 Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology, 2005: p. 454-457.
15. Tsang, W.W.M. and P. Kong-Pang. *A finger-tracking virtual mouse realized in an embedded system*. in *Intelligent Signal Processing and Communication Systems, 2005. ISPACS 2005. Proceedings of 2005 International Symposium on*. 2005.
16. Mukherjee, A., K. Chakraborty, and A. Basu, *SweepSticks: An Adaptive Virtual Mouse for People*

With Neuromotor Disorders. Assistive Technology, 2008: p. 111-124.

17. Navab, N., et al. *An on-line evaluation system for optical see-through augmented reality*. in *Virtual Reality, 2004. Proceedings. IEEE*. 2004.
18. Sanghi, A., et al. *A fingertip detection and tracking system as a virtual mouse, a signature input device and an application selector*. in *Devices, Circuits and Systems, 2008. ICCDCS 2008. 7th International Caribbean Conference on*. 2008.
19. Roh, M.-C., S.-J. Huh, and S.-W. Lee. *A Virtual Mouse interface based on Two-layered Bayesian Network*. in *Applications of Computer Vision (WACV), 2009 Workshop on*. 2009.
20. Ren, H. and G. Xu. *Human action recognition in smart classroom*. in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*. 2002.
21. Voida, S., et al., *A study on the manipulation of 2D objects in a projector/camera-based augmented reality environment*. Proceeding CHI '05 Proceedings of the SIGCHI conference on Human factors in computing systems, 2005: p. 611-620.
22. Moya, J. and D. Saenz, *Vehicular Traffic Monitoring Via a Biologically-Inspired Approach*. WSEAS Transactions on Signal Processing, 2005. **1**(13): p. 385-391.

Appendix

C328 Image Commands and Functions

Sync Function

```
private void SYNC()
        {
            //Clear global function checker
            Response = false ;

            //Showing action to the user
            Result.Text = "Syncing...";
            Result.Refresh();

            //Sending sync command for supported baudrates
            for (i = 0; i != 2; ++i)
            {
                serialPort1.BaudRate = Supported_Bauds[i];

                for (counter = 0; counter != 100 & Response != true; ++counter)
                {
                    //Sending SYNC commands for a few turns and checking if sync acquired
                    Response = SYNC_Command();
                }

                if (Response == true){ break; }

            }

            //Show the result to user
            if (Response == true)
            {
                //Order the buttons
                Result.Text = "SYNC...";
                Result.Refresh();

                //BaudRate_Command();

            }
            else
            {
                //if unsuccesfull show the problem to the user
                Result.Text = "Reset the camera or Retry";
                Result.Refresh();
            }

        }
```

JPEG Function

```
        private void JPEGSize()
        {
```

```
            //Clear global function checker
            Response = false;

            //Getting the preview size from JPEGSize Box

            Init[5] = Convert.ToByte((2 * 3) + 1);

            //Sending init command
            Response = Command(Init);

            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "JPEG size set successfully...";
                Result.Refresh();

            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to set the JPEG size.Resync or retry please...";
                Result.Refresh();
            }
        }
```

PackSize Function

```
private void PackSize()
        {
            //Clear global function checker
            Response = false;

            //Getting the package size from the combobox and splitting the value
            upperbyte = (Convert.ToInt32(512)) >> 8;
            lowerbyte = Convert.ToInt32(512) & 0xFF;

            //Setting the package size
            Pack[3] = Convert.ToByte(lowerbyte); Pack[4] = Convert.ToByte(upperbyte);

            //Sending package command
            Response = Command(Pack);

            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "Package size set...";
                Result.Refresh();


            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to set the package size.Resync or retry please...";
                Result.Refresh();
```

```
            }
        }
```

Snapshot Function

```
private void SnapShot_Click()
        {
            //Clear global function checker
            Response = false;

            //Sending snapshot command
            Response = Command(Snap);

            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "Snapshot taken...";
                Result.Refresh();


            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to take the picture.Resync or retry please...";
                Result.Refresh();

            }
        }
```

Preview Function

```
private void Preview_Click()
        {
            Int32 Response = 0;

            Response = Preview_Command();

            if (Response != 0)
            {
                Result.Text = Response.ToString();  //if succesful show user that initialization
ac quired
                Result.Refresh();

                Construct_JPEG(Response);




            }
            else
            {
                Result.Text = "Unable to preview the picture.Resync or retry please..."; //if
unsucessfull show user the problem
```

```
                Result.Refresh();


            }
        }
```

Sync Command

```
bool SYNC_Command()
        {
            //Arranging acknowledge
            ACK[2] = SYNC_[1];

            //Sending sync command
            serialPort1.Write(SYNC_, 0, 6);
            Thread.Sleep(50);

            //Checking if receive buffer is empty?
            if (serialPort1.BytesToRead != 0)
            {
                Thread.Sleep(1);
                serialPort1.Read(RcvBuffer12, 0, 12);

                //checking if true response received
                if (RcvBuffer12[0]==SYNC_Resp[0] & RcvBuffer12[1]==SYNC_Resp[1]
&RcvBuffer12[2]==SYNC_Resp[2] &
                    RcvBuffer12[4]==SYNC_Resp[4] & RcvBuffer12[5]==SYNC_Resp[5]
&RcvBuffer12[6]==SYNC_Resp[6] &
                    RcvBuffer12[7]==SYNC_Resp[7] & RcvBuffer12[8]==SYNC_Resp[8]
&RcvBuffer12[9]==SYNC_Resp[9] &
                    RcvBuffer12[10]==SYNC_Resp[10] & RcvBuffer12[11]==SYNC_Resp[11] )
                {
                    // if yes sending ack command and returning true
                    serialPort1.Write(ACK, 0, 6);
                    return true;
                }
                else
                {
                    //else sending false
                    return false;
                }
            }

            return false;

        }
```

Preview Command

```
Int32  Preview_Command()
       {

           //Writing getpicture string to com port
```

```
            serialPort1.Write(GetPic, 0, 6);
            Thread.Sleep(100);

            //Checking receive buffer
            if (serialPort1.BytesToRead != 0)
            {
                //Reading the receive buffer
                serialPort1.Read(RcvBuffer12, 0, 12);
            }

            //Checking if true response received
            if ( RcvBuffer12[0] == GetPic_Resp [0] & RcvBuffer12[1] == GetPic_Resp [1] &
RcvBuffer12[2] == GetPic_Resp [2] &
                 RcvBuffer12[4] == GetPic_Resp [4] & RcvBuffer12[5] == GetPic_Resp [5] &
RcvBuffer12[6] == GetPic_Resp [6] &
                 RcvBuffer12[7] == GetPic_Resp [7] & RcvBuffer12[8] == GetPic_Resp [8] )
            {

                //if yes sending the image data length
                bytes = Convert.ToInt32(RcvBuffer12[11]);
                bytes <<= 8;
                bytes = bytes | Convert.ToInt32(RcvBuffer12[10]);
                bytes <<= 8;
                bytes = bytes | Convert.ToInt32(RcvBuffer12[9]);
                return bytes;
            }
            else
            {
                //else sending zero
                return 0;
            }
        }
```

Constru JPEG Command

```
void Construct_JPEG(Int32 bytes)
        {

            Byte[] Jpeg = new Byte[bytes];                          //Null jpeg stream

            //Clearing the global counters
            counter = 0; counter2 = 0; x = 0; index = 0 ;

            //Clearing the upper and lower bytes
            upperbyte = 0; lowerbyte = 0;


            for ( ;counter != bytes ;  )   //loop until getting all image data
            {

                Jpeg_ACK[4] = Convert.ToByte(lowerbyte);
                Jpeg_ACK[3] = Convert.ToByte(upperbyte);

                // send the acknowledge command
                serialPort1.Write(Jpeg_ACK, 0, 6);
                Thread.Sleep((3+1)*20);
```

```
            if ( (x = serialPort1.BytesToRead) != 0)
            {
                //read the values
                Byte[] Temp = new Byte[serialPort1.BytesToRead];
                serialPort1.Read(Temp, 0, serialPort1.BytesToRead);

                //control if true package got
                if (Temp[0] == lowerbyte & Temp[1] == upperbyte)
                {
                    for (int i = 4 ; i != x - 2; ++i)
                    {
                        //cut the unused bytes
                        Jpeg[index] = Temp[ i ];
                        ++index;
                    }

                }
                else
                {
                    // if error occurs show it to user
                    Result.Text = "Corruption occurred..";
                    Result.Refresh();
                    break;
                }

                //arrange the controllers
                counter = counter + x - 6;
                ++counter2;
                upperbyte = counter2 >> 8;
                lowerbyte = counter2 & 0x00FF;
            }


        }

        // Send the last command
        Jpeg_ACK[4] = Convert.ToByte(lowerbyte);
        Jpeg_ACK[3] = Convert.ToByte(upperbyte);
        serialPort1.Write(Jpeg_ACK, 0, 6);


        //Display the picture

        Stream image = new MemoryStream(Jpeg);


        pictureBox1.Image = Image.FromStream(image);



    }
```

Main Command

```
public bool Command(Byte[] Parameter)
```

```
        {

            //Arranging acknowledge
            ACK[2] = Parameter[1];

            //Sending parameters to com1
            serialPort1.Write(Parameter, 0, 6);

            //Waiting the device to send response
            Thread.Sleep(40);

            //Checking receive buffer
            if (serialPort1.BytesToRead != 0)
            {
                Thread.Sleep(1);

                //Reading the receive buffer
                serialPort1.Read(RcvBuffer, 0, 6);
            }


            //Show the result to user
            if (RcvBuffer[0] == ACK[0] & RcvBuffer[1] == ACK[1] & RcvBuffer[2] == ACK[2] &
                RcvBuffer[4] == ACK[4] & RcvBuffer[5] == ACK[5])
            {
                return true;
            }
            else
            {
                return false;
            }

        }

    }
```

MATLAB ALGORITHM MAPPING

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Mapping of Coordinates to send through Com Port
     if y>=0 && y<30
r=17
elseif y>=30 && y<60
r=18
elseif y>=60 && y<90
r=19
elseif y>=90 && y<120
r=20
elseif y>=120 && y<150
r=21
elseif y>=150 && y<180
r=22
elseif y>=180 && y<210
r=23
elseif y>=210 && y<240
r=24
elseif y>=240 && y<270
```

```
r=25
elseif y>=270 && y<300
r=26
elseif y>=300 && y<330
r=27
elseif y>=330 && y<360
r=28
elseif y>=360 && y<390
r=29
elseif y>=390 && y<420
r=30
elseif y>=420 && y<450
r=31
elseif y>=450 && y<=480
r=32
 end



  if z>=0 && z<30
c=1
elseif z>=30 && z<60
c=2
elseif z>=60 && z<90
c=3
elseif z>=90 && z<120
c=4
elseif z>=120 && z<150
c=5
elseif z>=150 && z<180
c=6
elseif z>=180 && z<210
c=7
elseif z>=210 && z<240
c=8
elseif z>=240 && z<270
c=9
elseif z>=270 && z<300
c=10
elseif z>=300 && z<330
c=11
elseif z>=330 && z<360
c=12
elseif z>=360 && z<390
c=13
elseif z>=390 && z<420
c=14
elseif z>=420 && z<450
c=15
elseif z>=450 && z<=480
c=16
  end



    fid = fopen('xy.txt', 'w');
```

```
    fprintf(fid,'%d\r%d\r',r, c);
    fclose(fid);
```

VISUAL STUDIO COMPLETE GUI PROGRAM

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;



namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        #region Variables
        Byte[] RcvBuffer   = new Byte[6] { 0, 0, 0, 0, 0, 0 };      //6 byte global receive
array
        Byte[] RcvBuffer12 = new Byte[12] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };//12 byte
global receive array
        Byte[] SYNC_       = new Byte[6] { 170, 13, 0, 0, 0, 0 };   //Sync command string
        Byte[] SYNC_Resp   = new Byte[12] { 170, 14, 13, 0, 0, 0, 170, 13, 0, 0, 0, 0 };
//Expected response after sync command
        Byte[] Baud        = new Byte[6] { 170, 7, 15, 1, 0, 0 };   //BaudRate command
string(Default 115200 bit/sec)
        Byte[] Init        = new Byte[6] { 170, 1, 0, 7, 0, 7 };    //Default init command
string(Default 640x480)
        Byte[] Pack        = new Byte[6] { 170, 6, 8, 0, 2, 0 };    //Default package command
string(Default 512 bytes)
        Byte[] Snap        = new Byte[6] { 170, 5, 0, 0, 0, 0 };    //Snapshot command string
        Byte[] Reset       = new Byte[6] { 170, 8, 0, 0, 0, 0 };    //RESET command string
        Byte[] GetPic      = new Byte[6] { 170, 4, 1, 0, 0, 0 };    //Getpicture command string
        Byte[] GetPic_Resp = new Byte[12] { 170, 14, 4, 0, 0, 0, 170, 10, 1, 0, 0, 0 };
//Expected response after getpicture command
        Byte[] ACK         = new Byte[6] { 170, 14, 13, 0, 0, 0 };  //Default ACK command string
        Byte[] Jpeg_ACK    = new Byte[6] { 170, 14, 0, 0, 0, 0 };   //ACK command string to get
all the image data
        bool Response      = false;                                 //Command error checker
        Int32[] Supported_Bauds = new Int32[2] {115200, 115200};   //Baudrates to be checked
        Int32 upperbyte = 0, lowerbyte = 0;                         //Variables to split value
into two bytes
        Int16 i = 0;                                                //Baudrate selector & index
holder to cut the unneeded bytes
        Int32 counter = 0, counter2 = 0, x = 0, index = 0;          //General use counters
        Int32 bytes = 0;                                            //Value that shows the
number of bytes of the picture
        bool State;
        #endregion
```

```
#region Initialize Form


public Form1()
{
    InitializeComponent();


}

private void Form1_Load(object sender, EventArgs e)
{


    //Center the image when displayed
    pictureBox1.SizeMode = PictureBoxSizeMode.CenterImage;


}

#endregion

#region Connect_Click

private void Connect_Click(object sender, EventArgs e)
{
    State = true;
    //Check if com port 1 is busy
    if (serialPort1.IsOpen)
    {
        //if busy show it to user
        Result.Text = "Com1 is busy. Close other application...";
        Result.Refresh();

    }
    else
    {
        // if not open, open port


        serialPort1.Open();
        Result.Text = "Com1 opened successfully...";
        Result.Refresh();

        SYNC_Click();
        Thread.Sleep(80);
        JPEGSize_Click();
        Thread.Sleep(80);
        PackSize_Click();
        Thread.Sleep(80);
        SnapShot_Click();
        Thread.Sleep(80);
        Preview_Click();
        Thread.Sleep(80);
        Save_Ref();//saves in this method the reference image
        Connect.Enabled = true;
        Disconnect.Enabled = true;
        PauseForMilliSeconds(2000);
```

```
        while (State)
        {
            SYNC_Click();
            Thread.Sleep(80);
            JPEGSize_Click();
            Thread.Sleep(80);
            PackSize_Click();
            Thread.Sleep(80);
            SnapShot_Click();
            Thread.Sleep(80);
            Preview_Click();
            Thread.Sleep(80);
            Save_Click();//saves in this method the reference image
            Check_Marked();
            Connect.Enabled = true;
            Disconnect.Enabled = true;
            PauseForMilliSeconds(3000);
        }
    }
}

#endregion

#region Disconnect_Click
private void Disconnect_Click(object sender, EventArgs e)
{
    State = false;

    //check if com port 1 is closed
    if (serialPort1.IsOpen)
    {
        // if not closed, close port
        serialPort1.Close();
        Result.Text = "Com1 closed successfully...";
        Result.Refresh();
        Connect.Enabled = true;
        Disconnect.Enabled = false;
    }
    else
    {
        //if closed,show it to user
        Result.Text = "Com1 is already closed...";
        Result.Refresh();

    }
}
#endregion

#region SYNC_Click

private void SYNC_Click()
{
    //Clear global function checker
    Response = false ;

    //Showing action to the user
    Result.Text = "Syncing...";
    Result.Refresh();

    //Sending sync command for supported baudrates
```

```
    for (i = 0; i != 2; ++i)
    {
        serialPort1.BaudRate = Supported_Bauds[i];

        for (counter = 0; counter != 100 & Response != true; ++counter)
        {
            //Sending SYNC commands for a few turns and checking if sync acquired
            Response = SYNC_Command();
        }

        if (Response == true){ break; }

    }

    //Show the result to user
    if (Response == true)
    {
        //Order the buttons
        Result.Text = "SYNC...";
        Result.Refresh();

        //BaudRate_Command();

    }
    else
    {
        //if unsuccesfull show the problem to the user
        Result.Text = "Reset the camera or Retry";
        Result.Refresh();
    }

    //Order the buttons

}

#endregion

#region JPEGSize_Click

private void JPEGSize_Click()
{
    //Clear global function checker
    Response = false;

    //Getting the preview size from JPEGSize Box

    Init[5] = Convert.ToByte((2 * 3) + 1);

    //Sending init command
    Response = Command(Init);

    //Show the result to user
    if (Response == true)
    {
        //if succesful show user that initialization acquired
        Result.Text = "JPEG size set successfully...";
        Result.Refresh();

    }
    else
```

```
    {
        //if unsucessfull show user the problem
        Result.Text = "Unable to set the JPEG size.Resync or retry please...";
        Result.Refresh();
    }
}

#endregion

#region PackSize_Click


private void PackSize_Click()
{
    //Clear global function checker
    Response = false;

    //Getting the package size from the combobox and splitting the value
    upperbyte = (Convert.ToInt32(512)) >> 8;
    lowerbyte = Convert.ToInt32(512) & 0xFF;

    //Setting the package size
    Pack[3] = Convert.ToByte(lowerbyte); Pack[4] = Convert.ToByte(upperbyte);

    //Sending package command
    Response = Command(Pack);

    //Show the result to user
    if (Response == true)
    {
        //if succesful show user that initialization acquired
        Result.Text = "Package size set...";
        Result.Refresh();


    }
    else
    {
        //if unsucessfull show user the problem
        Result.Text = "Unable to set the package size.Resync or retry please...";
        Result.Refresh();

    }
}


#endregion

#region Snapshot_Click



private void SnapShot_Click()
{
    //Clear global function checker
    Response = false;

    //Sending snapshot command
    Response = Command(Snap);
```

```
            //Show the result to user
            if (Response == true)
            {
                //if succesful show user that initialization acquired
                Result.Text = "Snapshot taken...";
                Result.Refresh();


            }
            else
            {
                //if unsucessfull show user the problem
                Result.Text = "Unable to take the picture.Resync or retry please...";
                Result.Refresh();

            }
        }

        #endregion

        #region Preview_Click


        private void Preview_Click()
        {
            Int32 Response = 0;

            Response = Preview_Command();

            if (Response != 0)
            {
                Result.Text = Response.ToString();  //if succesful show user that initialization
ac quired
                Result.Refresh();

                Construct_JPEG(Response);




            }
            else
            {
                Result.Text = "Unable to preview the picture.Resync or retry please..."; //if
unsucessfull show user the problem
                Result.Refresh();


            }
        }

        #endregion

        #region Save_Click


        private void Save_Click()
        {
```

```
            //Create new folder for the picture
            if (!(File.Exists(@"c:\M_files\new_pics\1.jpg")))
            {
                try
                {
                    // File.Delete(@"c:\M_files\new_pics\ref.jpg");

                    Image image = pictureBox1.Image;
                    image.Save("c:\\M_files\\new_pics\\1.jpg",
System.Drawing.Imaging.ImageFormat.Jpeg);
                    PauseForMilliSeconds(6000);
                    File.Delete(@"c:\M_files\new_pics\marked.jpg");
                }
                catch (System.NullReferenceException de)  //catch exceptions and just disconnect
                {
                    serialPort1.Close();
                    Result.Text = "Com1 closed successfully...";
                    Result.Refresh();
                    Connect.Enabled = true;
                    Disconnect.Enabled = false;

                }
            }
            else if ((File.Exists(@"c:\M_files\new_pics\1.jpg")))
            {
                try
                {
                    File.Delete(@"c:\M_files\new_pics\1.jpg");

                    Image image = pictureBox1.Image;
                    image.Save("c:\\M_files\\new_pics\\1.jpg",
System.Drawing.Imaging.ImageFormat.Jpeg);
                }
                catch (System.NullReferenceException de)  //catch exceptions and just disconnect
                {
                    serialPort1.Close();
                    Result.Text = "Com1 closed successfully...";
                    Result.Refresh();
                    Connect.Enabled = true;
                    Disconnect.Enabled = false;

                }
            }
        }

        #endregion

        #region Save_Ref


        private void Save_Ref()
        {


            //Create new folder for the picture
            if (!(File.Exists(@"c:\M_files\new_pics\ref.jpg")))
            {
                try
                {
                    File.Delete(@"c:\M_files\new_pics\marked.jpg");
```

```
                    Image image = pictureBox1.Image;
                    image.Save("c:\\M_files\\new_pics\\ref.jpg",
System.Drawing.Imaging.ImageFormat.Jpeg);
                }
                catch (System.NullReferenceException de)  //catch exceptions and just disconnect
                {
                    serialPort1.Close();
                    Result.Text = "Com1 closed successfully...";
                    Result.Refresh();
                    Connect.Enabled = true;
                    Disconnect.Enabled = false;

                }
            }
            else if ((File.Exists(@"c:\M_files\new_pics\ref.jpg")))
            {
                try
                {
                    File.Delete(@"c:\M_files\new_pics\ref.jpg");
                   // File.Delete(@"c:\M_files\new_pics\marked.jpg");
                    Image image = pictureBox1.Image;
                    image.Save("c:\\M_files\\new_pics\\ref.jpg",
System.Drawing.Imaging.ImageFormat.Jpeg);
                }
                catch (System.NullReferenceException de)  //catch exceptions and just disconnect
                {
                    serialPort1.Close();
                    Result.Text = "Com1 closed successfully...";
                    Result.Refresh();
                    Connect.Enabled = true;
                    Disconnect.Enabled = false;

                }
            }


        }

        #endregion

        #region Check_Marked


        private void Check_Marked()
            {


                //Create new folder for the picture
                if (!(File.Exists(@"c:\M_files\new_pics\marked.jpg")))
                {
                    try
                    {
                        MLApp.MLAppClass matlab = new MLApp.MLAppClass();
                        matlab.Execute("cd c:\\M_files");          //changes directory
                        matlab.Execute("jesse_script");

                        pictureBox1.Image = Image.FromFile("c:\\M_files\\new_pics\\marked.jpg");

                        List<string> list = new List<string>();
```

```
                    using (StreamReader reader = new StreamReader("c:\\M_files\\xy.txt"))
                    {
                        string line;
                        while ((line = reader.ReadLine()) != null)
                        {
                            list.Add(line);          // Add to list.
                            Console.WriteLine(line); // Write to console.
                        }
                    }
                    Byte byteX = Convert.ToByte(list[0]);
                    Byte byteY = Convert.ToByte(list[1]);
                    File.Delete("c:\\M_files\\new_pics\\xy.txt");
                    port.Open();
                    // Write a set of bytes
                    port.Write(new byte[] { byteX, byteY }, 0, 2);
                    // Close the port
                    port.Close();

                }
                catch (System.NullReferenceException de)  //catch exceptions and just
disconnect
                {
                    Result.Text = "Running Jesse...";
                    Result.Refresh();
                    Connect.Enabled = true;
                    Disconnect.Enabled = false;
                }
            }
            else if ((File.Exists(@"c:\M_files\new_pics\marked.jpg")))
            {
                try
                {
                    File.Delete(@"c:\M_files\new_pics\marked.jpg");

                    MLApp.MLAppClass matlab = new MLApp.MLAppClass();
                    matlab.Execute("cd c:\\M_files");          //changes directory
                    matlab.Execute("jesse_script");

                    pictureBox1.Image = Image.FromFile("c:\\M_files\\new_pics\\marked.jpg");

                    List<string> list = new List<string>();
                    using (StreamReader reader = new StreamReader("c:\\M_files\\xy.txt"))
                    {
                        string line;
                        while ((line = reader.ReadLine()) != null)
                        {
                            list.Add(line);          // Add to list.
                            Console.WriteLine(line); // Write to console.
                        }
                    }
                    Byte byteX = Convert.ToByte(list[0]);
                    Byte byteY = Convert.ToByte(list[1]);
                    File.Delete("c:\\M_files\\new_pics\\xy.txt");
                    port.Open();
                    // Write a set of bytes
                    port.Write(new byte[] { byteX, byteY }, 0, 2);
                    // Close the port
                    port.Close();

                }
```

```
                    catch (System.NullReferenceException de)  //catch exceptions and just
disconnect
                    {
                        //serialPort1.Close();
                        Result.Text = "Running Jesse...";
                        Result.Refresh();
                        Connect.Enabled = true;
                        Disconnect.Enabled = false;

                    }
                }


            }

        #endregion

        #region Pause


        public static DateTime PauseForMilliSeconds(int MilliSecondsToPauseFor)
        {


            System.DateTime ThisMoment = System.DateTime.Now;
            System.TimeSpan duration = new System.TimeSpan(0, 0, 0, 0, MilliSecondsToPauseFor);
            System.DateTime AfterWards = ThisMoment.Add(duration);


            while (AfterWards >= ThisMoment)
            {
                System.Windows.Forms.Application.DoEvents();
                ThisMoment = System.DateTime.Now;
            }


            return System.DateTime.Now;
        }

        #endregion

        #region Commands

        #region SYNC_Command

////////////////////////////////////////////////////////////////////////////////////////////
 /* Function: SYNC_Command
 * usage: used to acquire synchronization between the c328 camera.Must be used first
 * */
        bool SYNC_Command()
        {
            //Arranging acknowledge
            ACK[2] = SYNC_[1];

            //Sending sync command
            serialPort1.Write(SYNC_, 0, 6);
            Thread.Sleep(50);

            //Checking if receive buffer is empty?
            if (serialPort1.BytesToRead != 0)
```

```
            {
                Thread.Sleep(1);
                serialPort1.Read(RcvBuffer12, 0, 12);

                //checking if true response received
                if (RcvBuffer12[0]==SYNC_Resp[0] & RcvBuffer12[1]==SYNC_Resp[1]
&RcvBuffer12[2]==SYNC_Resp[2] &
                    RcvBuffer12[4]==SYNC_Resp[4] & RcvBuffer12[5]==SYNC_Resp[5]
&RcvBuffer12[6]==SYNC_Resp[6] &
                    RcvBuffer12[7]==SYNC_Resp[7] & RcvBuffer12[8]==SYNC_Resp[8]
&RcvBuffer12[9]==SYNC_Resp[9] &
                    RcvBuffer12[10]==SYNC_Resp[10] & RcvBuffer12[11]==SYNC_Resp[11] )
                {
                    // if yes sending ack command and returning true
                    serialPort1.Write(ACK, 0, 6);
                    return true;
                }
                else
                {
                    //else sending false
                    return false;
                }
            }

            return false;

        }
        #endregion


        #region Preview_Command



////////////////////////////////////////////////////////////////////////////////////////////
///
/*Function name :Preview_Command
* usage: used to preview the snapshot picture
* */
       Int32  Preview_Command()
       {

           //Writing getpicture string to com port
           serialPort1.Write(GetPic, 0, 6);
           Thread.Sleep(100);

           //Checking receive buffer
           if (serialPort1.BytesToRead != 0)
           {
               //Reading the receive buffer
               serialPort1.Read(RcvBuffer12, 0, 12);
           }

           //Checking if true response received
           if ( RcvBuffer12[0] == GetPic_Resp [0] & RcvBuffer12[1] == GetPic_Resp [1] &
RcvBuffer12[2] == GetPic_Resp [2] &
                RcvBuffer12[4] == GetPic_Resp [4] & RcvBuffer12[5] == GetPic_Resp [5] &
RcvBuffer12[6] == GetPic_Resp [6] &
                RcvBuffer12[7] == GetPic_Resp [7] & RcvBuffer12[8] == GetPic_Resp [8] )
           {
```

```
                //if yes sending the image data length
                bytes = Convert.ToInt32(RcvBuffer12[11]);
                bytes <<= 8;
                bytes = bytes | Convert.ToInt32(RcvBuffer12[10]);
                bytes <<= 8;
                bytes = bytes | Convert.ToInt32(RcvBuffer12[9]);
                return bytes;
            }
            else
            {
                //else sending zero
                return 0;
            }
        }

 #endregion

        #region Construct_JPEG



////////////////////////////////////////////////////////////////////////////////////////////
/* Function:Construct_JPEG(Response);
* usage: used to set construct jpeg
* */
        void Construct_JPEG(Int32 bytes)
        {

            Byte[] Jpeg = new Byte[bytes];                          //Null jpeg stream

            //Clearing the global counters
            counter = 0; counter2 = 0; x = 0; index = 0 ;

            //Clearing the upper and lower bytes
            upperbyte = 0; lowerbyte = 0;


            for ( ;counter != bytes ;  )   //loop until getting all image data
            {

                Jpeg_ACK[4] = Convert.ToByte(lowerbyte);
                Jpeg_ACK[3] = Convert.ToByte(upperbyte);

                // send the acknowledge command
                serialPort1.Write(Jpeg_ACK, 0, 6);
                Thread.Sleep((3+1)*20);

                if ( (x = serialPort1.BytesToRead) != 0)
                {
                    //read the values
                    Byte[] Temp = new Byte[serialPort1.BytesToRead];
                    serialPort1.Read(Temp, 0, serialPort1.BytesToRead);

                    //control if true package got
                    if (Temp[0] == lowerbyte & Temp[1] == upperbyte)
                    {
                        for (int i = 4 ; i != x - 2; ++i)
                        {
                            //cut the unused bytes
                            Jpeg[index] = Temp[ i ];
```

```
                    ++index;
                }

            }
            else
            {
                // if error occurs show it to user
                Result.Text = "Corruption occurred..";
                Result.Refresh();
                break;
            }

            //arrange the controllers
            counter = counter + x - 6;
            ++counter2;
            upperbyte = counter2 >> 8;
            lowerbyte = counter2 & 0x00FF;
        }


    }

    // Send the last command
    Jpeg_ACK[4] = Convert.ToByte(lowerbyte);
    Jpeg_ACK[3] = Convert.ToByte(upperbyte);
    serialPort1.Write(Jpeg_ACK, 0, 6);


    //Display the picture

    Stream image = new MemoryStream(Jpeg);


    pictureBox1.Image = Image.FromStream(image);




}


#endregion

 #region Command


/// <Function name= "Command"><used to send commands that returns only 6 byte
acknowledge>
/// <param name="Parameter"><contains the 6 byte parameter string>
/// <returns><if succesfull returns true else returns false>
///
public bool Command(Byte[] Parameter)
{

    //Arranging acknowledge
    ACK[2] = Parameter[1];

    //Sending parameters to com1
    serialPort1.Write(Parameter, 0, 6);
```

```
            //Waiting the device to send response
            Thread.Sleep(40);

            //Checking receive buffer
            if (serialPort1.BytesToRead != 0)
            {
                Thread.Sleep(1);

                //Reading the receive buffer
                serialPort1.Read(RcvBuffer, 0, 6);
            }


            //Show the result to user
            if (RcvBuffer[0] == ACK[0] & RcvBuffer[1] == ACK[1] & RcvBuffer[2] == ACK[2] &
                RcvBuffer[4] == ACK[4] & RcvBuffer[5] == ACK[5])
            {
                return true;
            }
            else
            {
                return false;
            }

        }




    }
       #endregion

        #endregion
}
```

ARDUINO DOT MATRIX PROGRAM

```
/*****************************************************************************
* The program displays to the external Dot Matrix
*****************************************************************************/

#include <LedControl.h> //Library to Control Max7221
/*****************************************************************************
*********
*  Pin numbers vary depending on what is available
*  pin 9is connected to the DataIn
*  pin 8 is connected to the CLK
*  pin 7 is connected to LOAD
*  Sample for LedControl is
*  LedControl lc=LedControl(DataIn Pin, CLK Pin, LOAD Pin, # of Devices);
*  lc.setLed(address, row, column);
```

```
*  Max number of Devices is 8 for this library.
*****************************************************************************
*******/
LedControl lc=LedControl(9,8,7,4);

//Global variables used to store values
int s;
int s1;
int c;
int r;
int a=3;
byte val;
byte rval;
byte cval;
 int temp;

//Function to set the row on the Dot Matrix
void row(){
if (rval ==17 or rval ==25)
{
   r=0+s;
}

else if (rval ==18 or rval == 26)
{
  r=1+s;
}
else if(rval ==19 or rval ==27)
{
  r=2+s;
}
else if (rval ==20 or rval== 28)
{
   r=3+s;
}

else if (rval ==21 or rval== 29)
{
  r=4+s;
}
else if (rval ==22 or rval ==30)
{
  r=5+s;
}
else if (rval ==23 or rval ==31)
{
   r=6+s;
}
```

```
else if (rval ==24 or rval ==32)
{
  r=7;
}
 }

//Function to set the Column on the Dot Matrix
  void cols(){
if (cval == 1 or cval == 9)
{
   c=0+s1;
}
else if (cval == 2 or cval == 10)
{
  c=1+s1;
}
else if (cval == 3 or cval == 11)
{
  c=2+s1;
}
else if (cval == 4 or cval == 12)
{
   c=3+s1;
}

else if (cval == 5 or cval == 13)
{
  c=4+s1;
}
else if (cval == 6 or cval == 14)
{
  c=5+s1;
}
else if (cval == 7 or cval == 15)
{
   c=6+s1;
}

else if (cval == 8 or cval == 16)
{
  c=7;
}
 }
//Function to set which  of the 4 matrices is displaying
   void address(){
if (cval>=1 && cval<=8 && rval>=17 && rval<=24)
{
```

```
   a=3;
}

else if (cval>=9 && cval<=16 && rval>=17 && rval<=24)
{
  a=2;
}
else if (cval>=1 && cval<=8 && rval>=25 && rval<=32)
{
  a=1;
}
else if (cval>=9 && cval<=16 && rval>=25 && rval<=32)
{
   a=0;
}
 }
//Function to read and display serial communcation data
 void com(){
   if (Serial.available())
 {
   // read the most recent byte (which will be from 0 to 255):
   val = Serial.read();
   // set the brightness of the LED:

   if (val>16 && val<=32){
       rval=val;
    //   digitalWrite(ledPin,  HIGH);
      // delay(500);
       // digitalWrite(ledPin, LOW);
      }
      else if(val>0 && val<=16){
        cval=val;
      //digitalWrite(ledPin1,  HIGH);
      //delay(500);
      //digitalWrite(ledPin1, LOW);
      }
   Serial.print(val, BYTE);

 }
 }

//Function to setup only runs once
   void setup() {

 //Set the baud rate for the serial communication and Dot Matrix
   Serial.begin(9600);
  //we have already set the number of devices when we created the LedControl
  int devices=lc.getDeviceCount();
```

```
  //we have to init all devices in a loop
  for(int address=0;address<devices;address++) {
    //The MAX72XX is in power-saving mode on startup so we have to wake it up
    lc.shutdown(address,false);
    //Set the brightness to a medium values
    lc.setIntensity(address,8);
    //and clear the display
    lc.clearDisplay(address);
   }

   void loop() {
// Begins reading data from the com port
com();
//sets the address ,row, and column depending on the two values coming in
   address();
    row();
    cols();
    lc.setLed(a,r,c, true);
// Clear display after every run to make the led on the Dot Matrix blink
     delay(200);
     lc.clearDisplay(3);
     lc.clearDisplay(2);
     lc.clearDisplay(1);
     lc.clearDisplay(0);
   }
```

ARDUINO SERVO PROGRAM

```
/*****************************************************************************

*   This program reads data from the communication port from the computer then displays the to

the External 16x16 Dot matrix as well as to the 2 servos for pan and tilt.

*****************************************************************************/

#include <LedControl.h> //Library to Control Max7221
/*****************************************************************************
*  Pin numbers vary depending on what is available
*  pin 9is connected to the DataIn
*  pin 8 is connected to the CLK
*  pin 7 is connected to LOAD
*  Sample for LedControl is
*  LedControl lc=LedControl(DataIn Pin, CLK Pin, LOAD Pin, # of Devices);
*  lc.setLed(address, row, column);
*  Max number of Devices is 8 for this library.
*****************************************************************************/
```

```
LedControl lc=LedControl(9,8,7,4);
//Global variables used to store values

int c;
int r;
int a=3;
byte val;
byte rval;
byte cval;
 int temp;

//Function to set the row on the Dot Matrix
void row(){
if (rval ==17 or rval ==25)
{
   r=0;
}

else if (rval ==18 or rval == 26)
{
  r=1;
}
else if(rval ==19 or rval ==27)
{
  r=2;
}
else if (rval ==20 or rval== 28)
```

```
{
  r=3;
}

else if (rval ==21 or rval== 29)
{
 r=4;
}
else if (rval ==22 or rval ==30)
{
 r=5;
}
else if (rval ==23 or rval ==31)
{
  r=6;
}

else if (rval ==24 or rval ==32)
{
 r=7;
}
 }

//Function to set the Column on the Dot Matrix
 void cols(){
if (cval == 1 or cval == 9)
{
```

```
   c=0;
}

else if (cval == 2 or cval == 10)
{
  c=1;
}
else if (cval == 3 or cval == 11)
{
  c=2;
}
else if (cval == 4 or cval == 12)
{
   c=3;
}

else if (cval == 5 or cval == 13)
{
  c=4;
}
else if (cval == 6 or cval == 14)
{
  c=5;
}
else if (cval == 7 or cval == 15)
{
   c=6;
```

```
}

else if (cval == 8 or cval == 16)
{
  c=7;
}
 }
//Function to set which  of the 4 matrices is displaying
   void address(){
if (cval>=1 && cval<=8 && rval>=17 && rval<=24)
{
   a=3;
}

else if (cval>=9 && cval<=16 && rval>=17 && rval<=24)
{
  a=2;
}
else if (cval>=1 && cval<=8 && rval>=25 && rval<=32)
{
  a=1;
}
else if (cval>=9 && cval<=16 && rval>=25 && rval<=32)
{
   a=0;
}
```

```
}


//Function to read and display serial communcation data
void com(){
  if (Serial.available())
{
  // read the most recent byte (which will be from 0 to 255):
  val = Serial.read();
  // set the brightness of the LED:

  if (val>16 && val<=32){
      rval=val;
   //   digitalWrite(ledPin,  HIGH);
     // delay(500);
     // digitalWrite(ledPin, LOW);
    }
    else if(val>0 && val<=16){
      cval=val;
    //digitalWrite(ledPin1,  HIGH);
    //delay(500);
    //digitalWrite(ledPin1, LOW);
    }
  // rem Serial.print(val, BYTE);

}
}
```

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
Servo myservo2;                // a maximum of eight servo objects can be created
void tilt(){
switch (rval){
  case 17:
    myservo.write(5.625);
    break;
  case 18:
    myservo.write(16.875);
    break;
  case 19:
    myservo.write(28.125);
    break;
  case 20:
    myservo.write(39.375);
    break;
  case 21:
    myservo.write(50.625);
    break;
  case 22:
    myservo.write(61.875);
    break;
  case 23:
    myservo.write(73.125);
    break;
```

```
case 24:
  myservo.write(84.375);
  break;
case 25:
  myservo.write(95.625);
  break;
case 26:
  myservo.write(106.875);
  break;
case 27:
  myservo.write(118.125);
  break;
case 28:
  myservo.write(129.375);
  break;
case 29:
  myservo.write(140.625);
  break;
case 30:
  myservo.write(151.875);
  break;
case 31:
  myservo.write(163.125);
  break;
case 32:
  myservo.write(174.375);
  break;
```

```
  }
}
void pan(){
switch (cval){
  case 16:
    myservo2.write(2.813);
    break;
  case 15:
    myservo2.write(8.438);
    break;
  case 14:
    myservo2.write(14.063);
    break;
  case 13:
    myservo2.write(19.688);
    break;
  case 12:
    myservo2.write(25.313);
    break;
  case 11:
    myservo2.write(30.938);
    break;
  case 10:
    myservo2.write(36.563);
    break;
  case 9:
    myservo2.write(42.188);
```

```
    break;
  case 8:
    myservo2.write(47.813);
    break;
  case 7:
    myservo2.write(53.438);
    break;
  case 6:
    myservo2.write(59.063);
    break;
  case 5:
    myservo2.write(64.688);
    break;
  case 4:
    myservo2.write(70.313);
    break;
  case 3:
    myservo2.write(75.938);
    break;
  case 2:
    myservo2.write(81.563);
    break;
  case 1:
    myservo2.write(87.188);
    break;
  }
}
```

```
//Function to setup only runs once
   void setup() {
 myservo.attach(3);  // attaches the servo on pin 9 to the servo object
  myservo2.attach(5);
 //Set the baud rate for the serial communication and Dot Matrix
   Serial.begin(9600);
  //we have already set the number of devices when we created the LedControl
  int devices=lc.getDeviceCount();
  //we have to init all devices in a loop
  for(int address=0;address<devices;address++) {
    //The MAX72XX is in power-saving mode on startup so we have to wake it up
    lc.shutdown(address,false);
    //Set the brightness to a medium values
    lc.setIntensity(address,8);
    //and clear the display
    lc.clearDisplay(address);
   }



   }
   void loop() {



// Begins reading data from the com port
```

```
        //com();
        //cval = 1;
        //rval = 18;
        //Sets position of Servos
        //pan();
        //tilt();
        //sets the address ,row, and column depending on the two values coming in
         //  address();
           // row();
           // cols();
           // lc.setLed(a,r,c, true);
        // Clear display after every run to make the led on the Dot Matrix blink
            //delay(200);
           // lc.clearDisplay(3);
           // lc.clearDisplay(2);
           // lc.clearDisplay(1);
          //  lc.clearDisplay(0);
            //lc.setLed(0,0,0, true);
           // myservo2.write(180);
}
```

Curriculum Vita

Son of Maria Rafaela Sanchez and Gerardo Miguel Chaidez, Miguel Angel Chaidez was born in Gonzalez Ortega, Zac, Mx. He lived in his birthplace until the age of 7 when he moved to the United States. He was raised for the latter part of his life in El Paso, Tx where he continued his pursuit of higher education and graduated with a Bachelor's of Science in Electrical Engineering in May of 2010. From a continuation of his senior project under the direction of Dr. John Moya he continued his education and obtained a Master's in Computer Engineering in the summer of 2011.

He has received an opportunity to work for Texas Instruments under the DLP department in Plano, Tx and will be moving there in the fall of 2011. He plans to pursue his career path and complete a Ph.D in Computer Engineering at the University of Texas in Dallas in the coming years.

Permanent address: 10200 Hedgerow Apt. 21
El Paso, Tx, 79925

This thesis was typed by Miguel Angel Chaidez