A

Project Report on

Remote navigation support System using Image Processing for Robotics system

By

Miss. Rutika Sanjay Bhagat

Under Guidance of

Prof. Sweta Kale

Department of Information Technology Engineering

Ms. Annapurna Agrawal (Scientist 'E')

Research and Development Establishment (Engineers)

Dighi, Pune

Sinhgad Institutes

Department of Information Technology Engineering

RMD Sinhgad Technical Institute Campus,

Warje, Pune-411058

[2021-21]

RMD SINHGAD TECHNICAL INSTITUTE CAMPUS,

WARJE, PUNE - 411 058

CERTIFICATE

This is to certify that Miss. Rutika Sanjay Bhagat has satisfactorily carried out and completed the project work entitled Remote navigation support System using Image Processing for Robotics system. It is submitted in the partial fulfilment of the prescribed syllabus in final year of Information Technology of Savitribai Phule Pune University, Pune for the academic year 2021-22.

Date :-

Place :-

Prof. Sweta Kale	Prof. Sweta Kale
(Guide)	(HOD, IT Dept.)

Dr. V. V. Dixit

(Principal, RMDSSOE)

ABSTRACT

In automated surveillance systems with multiple cameras, the system must be able to position the cameras accurately. Each camera must be able to pan-tilt such that an object detected in the scene is in a vantage position in the camera's image plane and subsequently capture images of that object. Typically, camera calibration is required. We propose an approach that uses only image-based information. Each camera is assigned a pan-tilt zero-position. Position of an object detected in one camera is related to the other cameras by homographs between the zero-positions while different pan-tilt positions of the same camera are related in the form of projective rotations. We then derive that the trajectories in the image plane corresponding to these projective rotations are approximately circular for pan and linear for tilt. The camera control technique is subsequently tested in a working prototype.

This project aims to develop remote navigation support system using image processing for robotics system. This includes Pan-Tilt movements which can be controller by various commands through programming. Also there is one camera mounted on that pan-tilt unit which is to display the visuals and we can see in on laptop. This can capture some random picture from that visual and can be save at any location as we want. After that we also able to open that picture on application window and can be perform image processing on that like grayscale, flip, edge detection etc.

ACKNOWLEDGEMENT

I would like to express our gratitude towards our project guide Prof. Sweta Kale (HOD) for her guidance during this project work, and to the college especially to the Information Technology Department for making all facilities available to us.

I take this opportunity to express our sincere thanks to Ms. Bani Hazra (Scientist ‘F’) , Ms. Annapurna Agrawal (Scientist ‘E’) and entire Robotics team for giving this great opportunity, encouragement and making all the facilities available to me, which helped me a lot in completing this project work. I am thankful to Mr. Alok Mukherjee (Scientist ‘G’) and R&DE (DRDO) team who were always a source of guidance and improvement during the project work.

Finally, I express my sincere thanks to all those who helped me directly or indirectly in this project.

Rutika Sanjay Bhagat

INDEX

CHAPTER – 1

INTRODUCTION

1.1 Problem Statement :

Remote navigation support system using image processing for robotics system.

1.2 Scope

Image processing has wide range of application in robotics systems such as obstacle avoidance, stereo vision, target detection and tracking etc. Under this project various image processing algorithms will be explored and implemented using C# and OpenCV.

1.3 Technologies involved :

Robotics, Image Processing, OpenCV, Computer vision

CHAPTER – 2

LITERATURE SURVEY

2.1 Pan Tilt Unit

The Pan Tilt Unit 20M (PTU) is a rugged 2-Degree of Freedom (DOF) actuation mechanism. The PTU is suitable for wide range of indoor and outdoor, fixed and mobile applications. The PTU is extremely compact, light weight, rugged and offers an off-shelf solution for accurate positioning of any type of small sensor or other payload.

The PTU 20M uses stepper motors for actuation. The body of the PTU is machined aluminium and provides very rigid, repeatable positioning. The controller board is integrated inside the unit and suits all weather conditions. Serial RS-232 communication protocol is available to interface with Host PC. Payloads can be mounted on the side and top depending on user applications and custom designed mounting brackets are also available.

The built-in command supports real-time control of actuators. The rich command set provides fine grained control of position (absolute and relative), velocity, acceleration, status and other unit functions.

Features of PTU :

- Military grade casing
- IP65 enclosure rating
- Ability to command in angle and pulses
- Simple command set
- Software configurable Pan and Tilt

Applications of PTU include :

- Robotics and Computer vision
- Border security and Law Enforcement
- Automated detection and tracking

- Mid and short range surveillance system
- Satellite communication systems
- Advanced monitoring system
- Highway and transportation monitoring
- Long range surveillance

Communicating with the PTU

Communication with the PTU 20M can be set up over a serial (RS232) communication link. The communication can be established using any embedded microcontroller board or using a host PC. When using a host PC the user can use any terminal program of his/her choice or write a high level program to send the necessary commands.

PTU Commands

In this section we describe the command set for operating the pan-tilt unit. The commands can be classified as position commands, mode commands and configuration commands.

Command Summary	
Position Commands	PP, TP, PPXXX, PP-XXX, TP-XXX, PO, TO, POXXX, TOXXX, PO-XXX, TO-XXX, PPXXXTPXXX, POXXXTOXXX, PM, TM, PMX, PM-X, TMX, TM-X, H
Configuration Commands	L, PLLXXX, PLUXXX, TLLXXX, TLUXXX, PS, TS, PSXXX, TSXXX, PA, TA, PAXXX, TAXXX, PD, TD, PDXXX, TDXXX, R, RQ, RP, RT, RE, RD, V, B, BXXX
Mode Commands	M, MP, MA

Table - PTU commands

i. The Commands are not case sensitive.
ii. For any commands or sequence of commands to be executed the CR LF characters have to be appended.
iii. Multiple commands can be entered separated by a single while space character. Entering more than one space may result in incorrect executing of commands.
iv. At most 30 commands can be entered at one time. In case more than 30 commands are entered, an error message is generated and more of the commands are executed.
v. While execution, if a wrong command is encountered, it is rejected with an error message. Any commands following the wrong command are also rejected. Commands prior to the wrong command are executed.

2.2 Serial Interface :

In this project we are going to use 9-pins serial communication for pan-tilt connection. There are two port require. One for pan-tilt commands and another is for zoom-in and zoom-out commands.

A COM port is simply an I/O interface that enables the connection of a serial device to a computer. You may also hear COM ports referred to as serial ports. Most modern computers are not equipped with COM ports, but there are many serial port devices still in use that use the interface. Lab instruments, medical equipment, and point-of-sale systems often make use of serial connections.

Figure - Design of COM port

USB interfaces have largely replaced COM ports as a faster way of performing serial data transmission. Most computers are equipped with internal modems, eliminating the need for external serial devices. Similarly, COM connected mice are no longer widely used to input data. Null-modem cables can be used with serial interfaces to connect two computers, but this method is rarely used anymore.

Serial ports are still used in many specialized situations. Some examples are surveillance cameras and equipment used for industrial automation implementations. They employ the RS232 interface which is still supported by the hardware on many modern computers. Since many cost-efficient serial over Ethernet software and hardware solutions enable COM ports to be used on machines that are not equipped with any, we can expect the RS232 protocol to be around for a while. A COM port also has the advantage of being able to power 15 meters of shielded cable. A comparable USB cable would be limited to a length of 5 meters.

The default configuration is already present in the software. So no need to configure the baud rate manually. Beyond this, there are four unknown rules. They are baud rate, data bit selection (framing), start-stop bit, and parity.

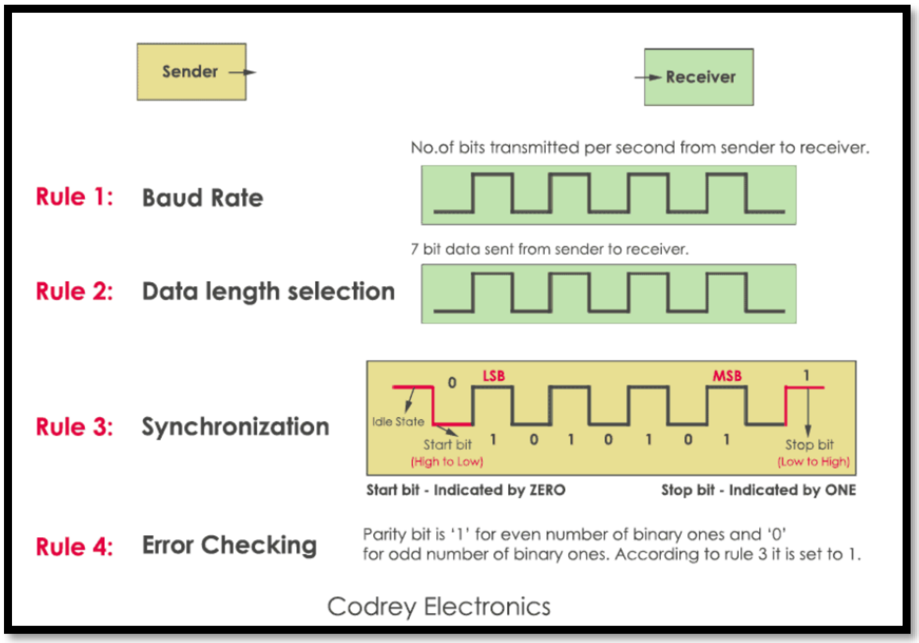

Sender
Receiver
No.of bits transmitted per second from sender to receiver.
Rule 1: Baud Rate
7 bit data sent from sender to receiver.
Rule 2: Data length selection
Rule 3: Synchronization
0
LSB
MSB
1
Idle State
Start bit
(High to Low)
1 0 1 0 1 0 1
Stop bit
(Low to High)
Start bit - Indicated by ZERO
Stop bit - Indicated by ONE
Rule 4: Error Checking
Parity bit is '1' for even number of binary ones and '0' for odd number of binary ones. According to rule 3 it is set to 1.
Codrey Electronics

Figure - Rules of Serial Communication

2.2.1 Signal assignments and connectors

In RS-232C, the connectors to use and the signal assignments have been defined and are standardized. The figure to the right describes the D-sub 9-pin signal assignments and signal lines.

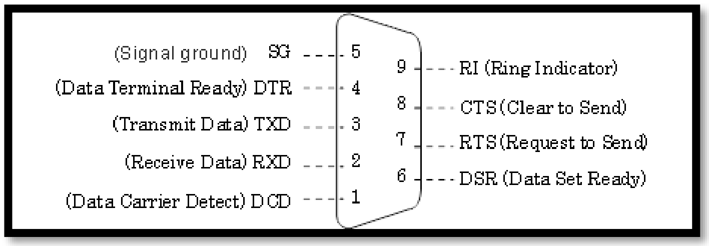


Figure - Pins and Connectors

2.2.2 Requirements

- **What is Baud Rate**

Baud rate is the speed of transferring data from the transmitter to a receiver in the form of bits per second. Some of the standard baud rates are 1200, 2400, 4800, 9600, 57600. You have to set the same baud rate on both sides.

In this project we require two baud rate. For Pan-Tilt commands we require baud rate as 115200 and for Zoom-In and Zoom-Out commands we require 9600 baud rate.

- **Stop bit length**

This sets the length of the bit that indicates the end of the data. This is normally selected as 1 bit, 1.5 bits, or 2 bits. The start bit length is fixed as 1 bit so this setting is not necessary.

- **Data bit length**

This specifies how many bits each item of data is composed from. This depends on the device being used, but normally specify 7 bits for alphanumeric characters and symbols, and specify 8 bits for 1 byte binary data.

- **Parity check setting**

This is a function to find errors in the data and is selected from "even parity check (EVEN)", "odd parity check (ODD)", or "no parity check (NONE)".

- **Parity check details**

On the sending side, a parity bit of "1" or "0" is added to the data so as to make the number of "1" data bits even for EVEN and odd for ODD. On the receiving side, the number of "1" data bits is counted and the data is judged as being correct if the number is even when EVEN and odd when ODD.

2.3 Camera Interface

The camera interface can be an essential part of this project. They can direct the attention to events. The camera which mounted on pan-tilt unit have ability to zoom-in and zoom-out feature. This can also be possible using program including zoom-in and zoom-out commands.

The commands are as follows-

X1	02 21 45 00 10 00 00 00 00 00 00 00 00 00 00 01 03 7A 22
X2	02 21 45 00 12 3C 00 00 00 00 00 00 00 00 00 01 03 68 75
X3	02 21 45 00 12 EC 00 00 00 00 00 00 00 00 00 01 03 D6 B7
X4	02 21 45 00 13 4C 00 00 00 00 00 00 00 00 00 01 03 D6 B7
X5	02 21 45 00 13 8C 00 00 00 00 00 00 00 00 00 01 03 D7 35
X6	02 21 45 00 13 D0 00 00 00 00 00 00 00 00 00 01 03 07 E5
X7	02 21 45 00 14 OC 00 00 00 00 00 00 00 00 00 01 03 EE 3C
X8	02 21 45 00 14 38 00 00 00 00 00 00 00 00 00 01 03 61 8D
X9	02 21 45 00 14 50 00 00 00 00 00 00 00 00 00 01 03 3E EC
X10	02 21 45 00 14 59 00 00 00 00 00 00 00 00 00 01 03 22 FO

Figure - Commands

CHAPTER – 3

SYSTEM DESIGN

The project includes mainly four module are as follow :

1. Pan-Tilt Controller
2. Camera Controller
3. Camera Feed
4. Image Processing

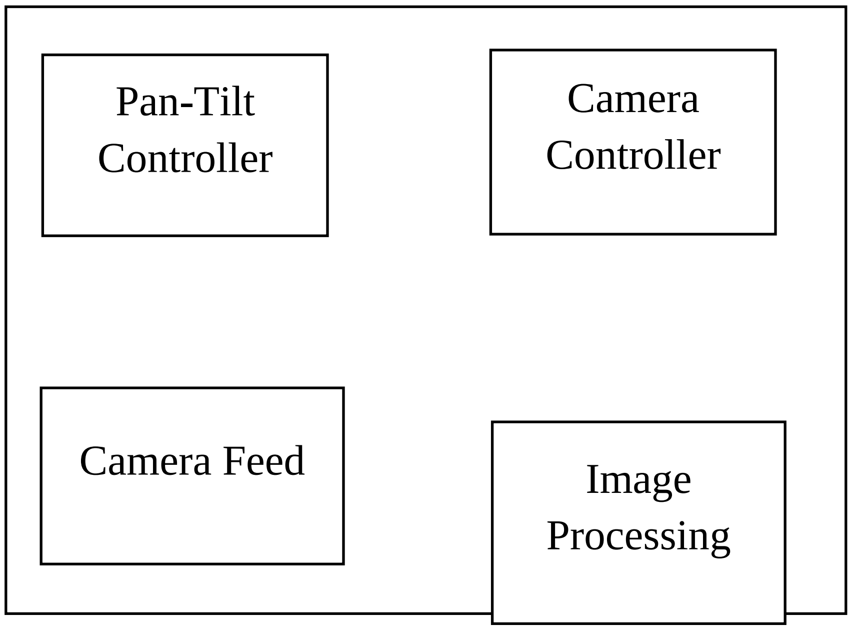


Figure - System Block Diagram

3.1 Pan-Tilt Controller

This contains pan and tilt motions. Whereas pan moves your device along flat, horizontal plane and tilt moves along a vertical plane. This pan-tilt motions done by commands through serial communication.

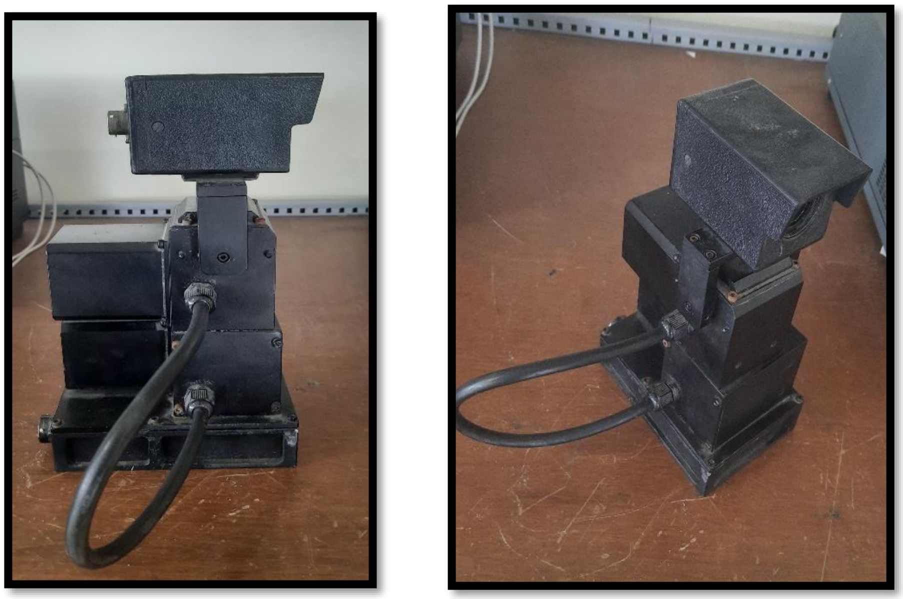


Figure – Pan Tilt Unit (PTU)

To operate the PTU the following are required:

I. PTU 20M Unit
II. Signal and power cable for PTU 20M
III. 24 VDC power supply rated at 3 amps
IV. Host computer with serial (RS232) communication port

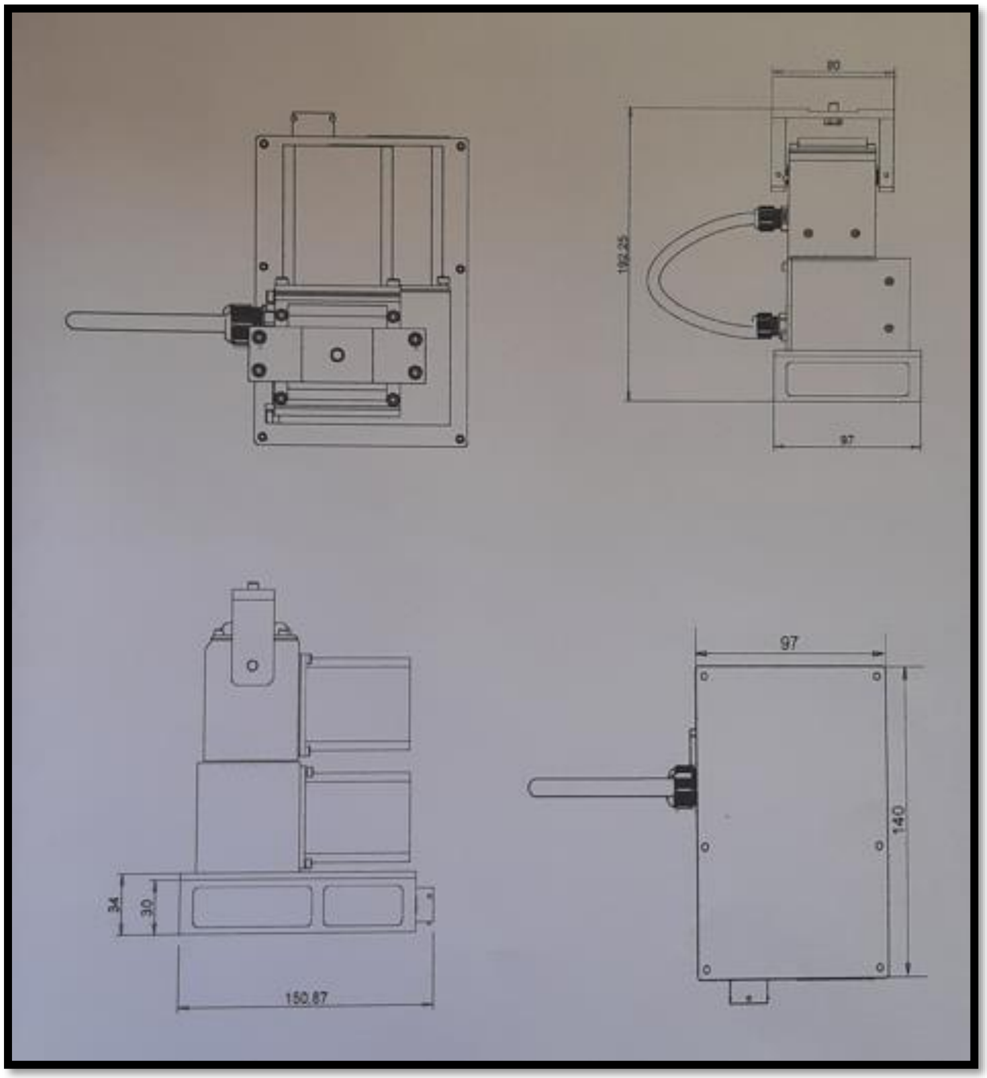
Following are some technical specifications required :

Maximum Payload	2kg nominal, (3kg on axis with optional bracket mount)
Position Resolution (both axis)	Upto 0.01 degree
Pan Limits	+160 degree to -160 degree
Tilt Limits	-50 degree to +50 degree
Speed (both axis)	10 degree to 60 degree per sec user configurable
Absolute Position	Yes
Communication Interface	Yes
Supply Voltage	24V DC @ 3.5 Amp

We are performing Up, Down, Right, Left and reset motions. For that the commands are:

i.	TO-XXX	:- Move XXX positions reverse from current position
ii.	PO-XXX	:- Move XXX position reverse from current position
iii.	TOXXX	:- Move XXX position forward from current position
iv.	POXXX	:- Move XXX position forward from current position
v.	R	:- For reset pan and tilt axis

Below diagram shows the mechanical drawing of PTU



3.2 Camera Controller

The camera controller block contains the zoom in and zoom out functionality. It also required serial communication to perform zoom in and zoom out. This interface can be an essential part of this project. They can direct the attention to events. The camera which mounted on pan-tilt unit have ability to zoom-in and zoom-out feature. This can also be possible using program including zoom-in and zoom-out commands. For that we required another serial port to sending camera control commands.

Following are some technical specifications required :

- Baud rate : 9600
- Supply Voltage : 12 V DC

Connectors

Following connector we used in our project to communication of Pan-Tilt. This connector allows connection that will allow two cables to be securely connected while protecting the connection from the elements.

Fig. 4 cable mount female

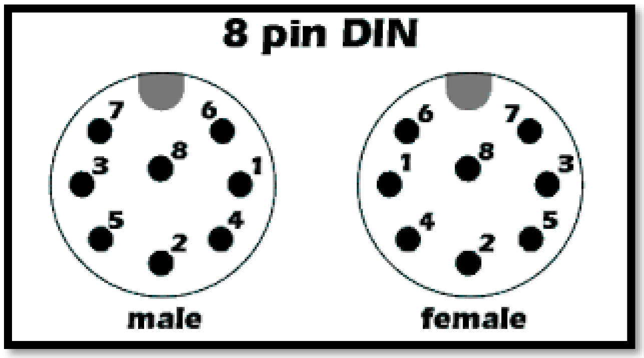


Fig. 5 Front view of Male female connector

Pin No.	Pin Description
A	Power : +24 VDC
B	Power : 0 VDC GND
C	RS232 Rx – Input

D	RS232 Tx – Output
E	RS232 Gnd – Ground
F	NC/Reset(For Firmware upgrade only)
G	NC
F	NC

Table 2 Connector Pin out

3.3 Camera Feed

The camera feed block supports video feature of camera. We can see the camera visuals on laptop screen. This can be happen using Emgu CV platform. For that we need to import some libraries that are –

1) Emgu.CV.dll
2) Emgu.CV.UI.dll
3) Emgu.Util.dll

With the help of this we can capture the camera feed of pan-tilt camera and see it on windows application.

3.4 Image Processing

Here in this we can capture any random image from camera feed. And able to save it at the specific device location and again able to open that image in image box. Image box is a tool that provide from Emgu CV. Also can do some image processing operation on that image like grayscale, edge detection. For this all we are using Emgu CV cross-platform.

Emgu CV is a cross-platform image-processing library. It is closely related to OpenCV because Emgu CV is a .NET wrapper to OpenCV. We can say Emgu CV is OpenCV in .NET. The amazing wrapper makes it possible for OpenCV functions to be called from .NET programming languages. C#, VB and VC++ are some of the languages supported.

Grayscale Image Processing

In image processing, grayscale means that the value of each pixel represents only the intensity information of the light. Such images typically display only the darkest black to the brightest white. In other words, the image contains only black, white, and gray colors, in which gray has multiple levels.

Below figure shows the exact difference between original image and grayscale image.

Figure – Example of grayscale image conversion

Edge Detection

Edge detection allows users to observe the features of an image for a significant change in the gray level. This texture indicating the end of one region in the image and the beginning of another. It reduces the amount of data in an image and preserves the structural properties of an image.

Edge Detection Operators are of two types:

- **Gradient** – based operator which computes first-order derivations in a digital image like, Sobel operator, Prewitt operator, Robert operator
- **Gaussian** – based operator which computes second-order derivations in a digital image like, Canny edge detector, Laplacian of Gaussian

Advantages of Image Processing :

- Provides generic color and depth image class.
- Automatic memory management
- XML-serialization image
- Both, invoking the image class and using OpevCV functions are directly supported
- Generic operations are provided for image pixels.

CHAPTER – 4
IMPLEMENTATION

4.1 Requirements

Development Interface : Microsoft Visual Studio 2019

Hardware : Pan Tilt, Camera, Analog to Digital Converter, OS-Windows

API : EmguCV, OpenCV

Programming Language : C#

4.2 Code

```
using System;
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;
using System.IO.Ports;
using System.Diagnostics;
using Emgu.CV;
using Emgu.CV.Util;
using Emgu.CV.Structure;
using Emgu.Util;
using Emgu.CV.UI;
using Emgu.CV.CvEnum;
using System.IO;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;

namespace PanTilt
{
    public partial class Form1 : Form
    {
        private Bitmap Image, Image2;
        private BitmapData ImageData, ImageData2, ImageData3, ImageData4;
        private byte[] buffer, buffer2, buffer3, buffer4;
        private int b, g, r, r_x, g_x, b_x, r_y, g_y, b_y, grayscale,
location, location2;
        private sbyte weight_x, weight_y;
        private sbyte[,] weights_x;
        private sbyte[,] weights_y;
        private IntPtr pointer, pointer2, pointer3, pointer4;
        private int location3, location4;

        private Size _pictureOriginal;

        private Capture _capture;
        private bool _captureInProgress;

        int count = 1;

        byte[] x1 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x10, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x7a, 0x22 };
        byte[] x2 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x12, 0x3c, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x68, 0x75 };
        byte[] x3 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x12, 0xec, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0xD6, 0xB7 };
        byte[] x4 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x13, 0x4C, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0xD6, 0xB7 };
        byte[] x5 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x13, 0x8C, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0xD7, 0x35 };
        byte[] x6 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x13, 0xd0, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0xE5 };
        byte[] x7 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x14, 0x0C, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0xee, 0x3c };
        byte[] x8 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x14, 0x38, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x61, 0x8d };
        byte[] x9 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x14, 0x50, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x3e, 0xec };
        byte[] x10 = new byte[] { 0x02, 0x21, 0x45, 0x00, 0x14, 0x59, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x22, 0xf0 };

        public Form1()
        {
            InitializeComponent();
            _pictureOriginal = pictureBox1.Size;


            weights_x = new sbyte[,] { { 1, 0, -1 }, { 2, 0, -2 }, { 1, 0, -1
} };
```

```
            weights_y = new sbyte[,] { { 1, 2, 1 }, { 0, 0, 0 }, { -1, -2, -1
} };


        }

        private void Form1_Load(object sender, EventArgs e)
        {

            string[] ports = SerialPort.GetPortNames();
            cboPortName.Items.AddRange(ports);
            cboPortName.SelectedIndex = 0;
            btnClose.Enabled = false;

            string[] ports2 = SerialPort.GetPortNames();
            cboPortName2.Items.AddRange(ports);
            cboPortName2.SelectedIndex = 0;
            btnDisconnect.Enabled = false;
        }

        private void btnOpen_Click(object sender, EventArgs e)
        {
            btnOpen.Enabled = false;
            btnClose.Enabled = true;
            try
            {
                serialPort1.PortName = cboPortName.Text;
                serialPort1.Open();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "unable to connect serial port 1
connection", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            btnOpen.Enabled = true;
            btnClose.Enabled = false;
            try
            {
                serialPort1.Close();

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "unable to disconnect serial port
1 connection", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            btnConnect.Enabled = false;
            btnDisconnect.Enabled = true;
            try
            {
                serialPort2.PortName = cboPortName2.Text;
                serialPort2.Open();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Unable to connect Serial Port 
2", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnDisconnect_Click(object sender, EventArgs e)
        {
            btnConnect.Enabled = true;
            btnDisconnect.Enabled = false;
            try
            {
                serialPort2.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Unable to disconnect Serial Port 
2", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (serialPort1.IsOpen)
                serialPort1.Close();
            if (serialPort2.IsOpen)
                serialPort2.Close();
        }


        private void btnUp_Click(object sender, EventArgs e)
        {
            string ch = "TO05\r\n";
            serialPort2.WriteLine(ch);

        }

        private void btnDown_Click(object sender, EventArgs e)
        {
            string ch = "TO-05\r\n";
            serialPort2.WriteLine(ch);
        }

        private void btnLeft_Click(object sender, EventArgs e)
        {
            string ch = "PO-05\r\n";
            serialPort2.WriteLine(ch);
        }
```

```
private void btnRight_Click(object sender, EventArgs e)
{
    string ch = "PO05\r\n";
    serialPort2.WriteLine(ch);
}

private void btnReset_Click(object sender, EventArgs e)
{
    string ch = "R\r\n";
    serialPort2.WriteLine(ch);
}


private void btnZoomIn_Click(object sender, EventArgs e)
{
    if (count == 1)
    {
        serialPort2.Write(x1, 0, x1.Length);
    }
    if (count == 2)
    {
        serialPort2.Write(x2, 0, x2.Length);
    }
    if (count == 3)
    {
        serialPort2.Write(x3, 0, x3.Length);
    }
    if (count == 4)
    {
        serialPort2.Write(x4, 0, x4.Length);
    }
    if (count == 5)
    {
        serialPort2.Write(x5, 0, x5.Length);
    }
    if (count == 6)
    {
        serialPort2.Write(x6, 0, x6.Length);
    }
    if (count == 7)
    {
        serialPort2.Write(x7, 0, x7.Length);
    }
    if (count == 8)
    {
        serialPort2.Write(x8, 0, x8.Length);
    }
    if (count == 9)
    {
        serialPort2.Write(x9, 0, x9.Length);

    }
    if (count == 10)
    {
        serialPort2.Write(x10, 0, x10.Length);
    }
```

```
                count += 1;
            }

            private void btnZoomOut_Click(object sender, EventArgs e)
            {

                if (count == 10)
                {
                    serialPort2.Write(x10, 0, x10.Length);
                }
                if (count == 9)
                {
                    serialPort2.Write(x9, 0, x9.Length);
                }
                if (count == 8)
                {
                    serialPort2.Write(x8, 0, x8.Length);
                }
                if (count == 7)
                {
                    serialPort2.Write(x7, 0, x7.Length);
                }
                if (count == 6)
                {
                    serialPort2.Write(x6, 0, x6.Length);

                }
                if (count == 5)
                {
                    serialPort2.Write(x5, 0, x5.Length);
                }
                if (count == 4)
                {
                    serialPort2.Write(x4, 0, x4.Length);
                }
                if (count == 3)
                {
                    serialPort2.Write(x3, 0, x3.Length);
                }
                if (count == 2)
                {
                    serialPort2.Write(x2, 0, x2.Length);

                }
                if (count == 1)
                {
                    serialPort2.Write(x1, 0, x1.Length);
                }
                count -= 1;
            }

             private void pictureBox_Click(object sender, EventArgs e)
               {

               }
```

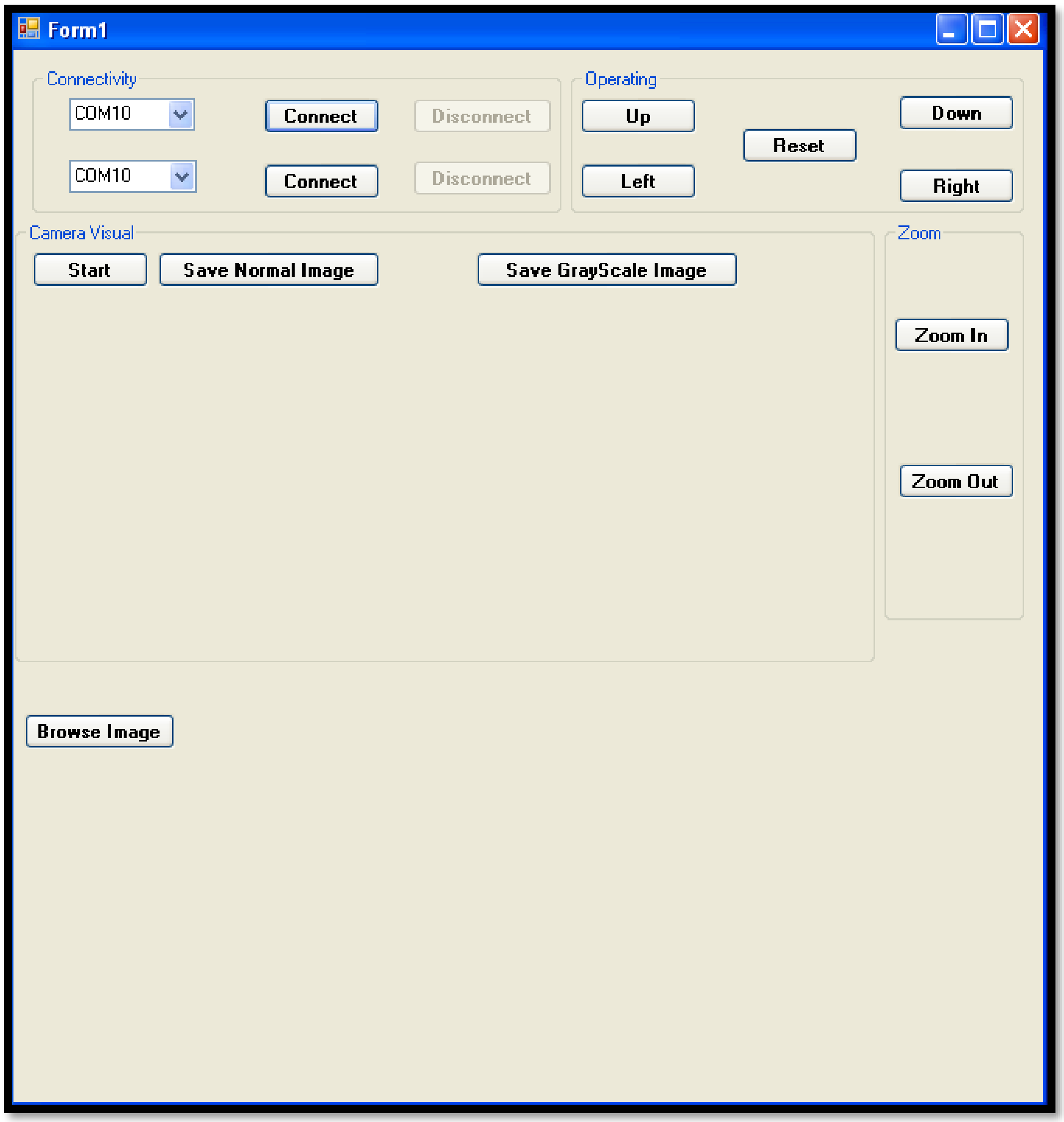
```
        private void Start_btn_Click(object sender, EventArgs e)
        {

            if (_capture == null)
            {
                try
                {
                    _capture = new Capture();

                }
                catch (NullReferenceException expt)
                {
                    MessageBox.Show(expt.Message);

                }
            }
                if (_capture != null)
                {
                    if (_captureInProgress)
                    {
                        Start_btn.Text = "Start";
                        Application.Idle -= ProcessFrame;

                    }
                    else
                    {
                       Start_btn.Text = "Capture";
                        Application.Idle += ProcessFrame;

                    }
                    _captureInProgress = !_captureInProgress;
                }


        }
        private void ProcessFrame(object sender, EventArgs arg)
        {
            Image<Bgr, Byte> OriginalFrame = _capture.QueryFrame();
            imageBox1.Image = OriginalFrame;

            Image<Gray, Byte> grayFrame = _capture.QueryFrame().Convert<Gray,
Byte>();
            imageBox2.Image = grayFrame;
        }

        private void btnSaveNormalImage_Click(object sender, EventArgs e)
        {
            SaveFileDialog save = new SaveFileDialog();
            save.Filter = "JPG(*.JPG|*.jpg";

            if (save.ShowDialog() == DialogResult.OK)
            {
                int width = Convert.ToInt32(imageBox1.Width);
                int height = Convert.ToInt32(imageBox1.Height);
                Bitmap bmp = new Bitmap(width, height);
                imageBox1.DrawToBitmap(bmp, new Rectangle(0, 0, Width,
Height));

                bmp.Save(save.FileName, ImageFormat.Jpeg);
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            SaveFileDialog save = new SaveFileDialog();
            save.Filter = "JPG(*.JPG|*.jpg";

            if (save.ShowDialog() == DialogResult.OK)
            {
                int width = Convert.ToInt32(imageBox2.Width);
                int height = Convert.ToInt32(imageBox2.Height);

                Bitmap bmp = new Bitmap(width, height);
                imageBox2.DrawToBitmap(bmp, new Rectangle(0, 0, Width,
Height));

                bmp.Save(save.FileName, ImageFormat.Jpeg);


            }
        }


        private void btnBrowse_Click_1(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                //Bitmap bit = new Bitmap(ofd.FileName);
                //pictureBox1.Image = bit;
                //pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

                Image = new Bitmap(ofd.FileName);
                Image2 = new Bitmap(Image.Width, Image.Height);
            }
            pictureBox1.Image = Image;
            pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
        }




        private void btnGray_Click(object sender, EventArgs e)
        {
            Bitmap copyBitmap = new Bitmap((Bitmap)pictureBox1.Image);
            processImage(copyBitmap);
            pictureBox2.Image = copyBitmap;

        }
        public bool processImage(Bitmap bmp)
        {
            for (int i = 0; i < bmp.Width; i++)
            {
                for (int j = 0; j < bmp.Height; j++)
                {
                    Color bmpColor = bmp.GetPixel(i, j);
                    int red = bmpColor.R;
                    int green = bmpColor.G;
                    int blue = bmpColor.B;
                    int gray = (byte)(.299 * red + .587 * green + .114 *
blue);

                    red = gray;
                    green = gray;
                    blue = gray;
                    bmp.SetPixel(i, j, Color.FromArgb(red, green, blue));

                }
            }
            return true;
        }



        private void btnSobel_Click(object sender, EventArgs e)
        {
            ImageData = Image.LockBits(new Rectangle(0, 0, Image.Width,
Image.Height), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            ImageData2 = Image2.LockBits(new Rectangle(0, 0, Image.Width,
Image.Height), ImageLockMode.WriteOnly, PixelFormat.Format24bppRgb);
            buffer = new byte[ImageData.Stride * Image.Height];
            buffer2 = new byte[ImageData.Stride * Image.Height];
            pointer = ImageData.Scan0;
            pointer2 = ImageData2.Scan0;
            Marshal.Copy(pointer, buffer, 0, buffer.Length);
            for (int y = 0; y < Image.Height; y++)
            {
                for (int x = 0; x < Image.Height * 3; x += 3)
                {
                    r_x = g_x = b_x = 0;
                    r_y = g_y = b_y = 0;
                    location = x + y * ImageData.Stride;

                    for (int yy = -(int)Math.Floor(weights_y.GetLength(0) /
2.0d), yyy = 0; yy <= (int)Math.Floor(weights_y.GetLength(0) / 2.0d); yy++,
yyy++)
                    {
                        if (y + yy >= 0 && y + yy < Image.Height)
                        {
                            for (int xx = -
(int)Math.Floor(weights_x.GetLength(1) / 2.0d) * 3, xxx = 0; xx <=
(int)Math.Floor(weights_x.GetLength(1) / 2.0d) * 3; xx += 3, xxx++)
                            {
                                if (x + xx >= 0 && x + xx <= Image.Width * 3
- 3)
```

```
                                {
                                    location2 = x + xx + (yy + y) *
ImageData.Stride;
                                    weight_x = weights_x[yyy, xxx];
                                    weight_y = weights_y[yyy, xxx];

                                    b_x += buffer[location2] * weight_x;
                                    g_x += buffer[location2 + 1] * weight_x;
                                    r_x += buffer[location2 + 2] * weight_x;
                                    b_y += buffer[location2] * weight_y;
                                    g_y += buffer[location2 + 1] * weight_y;
                                    r_y += buffer[location2 + 2] * weight_y;
                                }
                            }

                        }
                    }
                    b = (int)Math.Sqrt(Math.Pow(b_x, 2) + Math.Pow(b_y, 2));
                    g = (int)Math.Sqrt(Math.Pow(g_x, 2) + Math.Pow(g_y, 2));
                    r = (int)Math.Sqrt(Math.Pow(r_x, 2) + Math.Pow(r_y, 2));

                    if (b > 255) b = 255;
                    if (g > 255) g = 255;
                    if (r > 255) r = 255;

                    grayscale=(b + g + r) / 3;

                    buffer2[location] = (byte)grayscale;
                    buffer2[location + 1] = (byte)grayscale;
                    buffer2[location + 2] = (byte)grayscale;
                }

            }
            Marshal.Copy(buffer2, 0, pointer2, buffer.Length);
            Image.UnlockBits(ImageData);
            Image2.UnlockBits(ImageData2);
            pictureBox2.Image = Image2;

        }

        private void btnHorizontalFlip_Click(object sender, EventArgs e)
        {
            ImageData3 = Image.LockBits(new Rectangle(0, 0, Image.Width,
Image.Height), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            ImageData4 = Image2.LockBits(new Rectangle(0, 0, Image2.Width,
Image2.Height), ImageLockMode.WriteOnly, PixelFormat.Format24bppRgb);
            buffer3 = new byte[ImageData3.Stride * Image.Height];
            buffer4 = new byte[ImageData4.Stride * Image2.Height];
            pointer3 = ImageData3.Scan0;
            pointer4 = ImageData4.Scan0;
            Marshal.Copy(pointer3, buffer3, 0, buffer3.Length);
            for (int y = 0; y < Image.Height; y++)
            {
                for (int x = 0, xx = Image.Width * 3 - 3; x < Image.Width *
3; x += 3, xx -= 3)
                {
                    location3 = x + y * ImageData3.Stride;
                    location4 = xx + y * ImageData4.Stride;
                    buffer4[location4] = buffer3[location3];
                    buffer4[location4 + 1] = buffer3[location3 + 1];
                    buffer4[location4 + 2] = buffer3[location3 + 2];
                }
            }

            Marshal.Copy(buffer4, 0, pointer4, buffer3.Length);
            Image.UnlockBits(ImageData3);
            Image2.UnlockBits(ImageData4);
            pictureBox2.Image = Image2;

        }

        private void btnVerticalFlip_Click(object sender, EventArgs e)
        {
            ImageData3 = Image.LockBits(new Rectangle(0, 0, Image.Width,
Image.Height), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            ImageData4 = Image2.LockBits(new Rectangle(0, 0, Image2.Width,
Image2.Height), ImageLockMode.WriteOnly, PixelFormat.Format24bppRgb);
            buffer3 = new byte[ImageData3.Stride * Image.Height];
            buffer4 = new byte[ImageData4.Stride * Image2.Height];
            pointer3 = ImageData3.Scan0;
            pointer4 = ImageData4.Scan0;
            Marshal.Copy(pointer3, buffer3, 0, buffer3.Length);
            for (int y = 0, yy=Image.Height-1;y < Image.Height; y++,yy--)
            {
                for (int x = 0;x < Image.Width * 3; x += 3)
                {
                    location3 = x + y * ImageData3.Stride;
                    location4 = x + yy * ImageData4.Stride;
                    buffer4[location4] = buffer3[location3];
                    buffer4[location4 + 1] = buffer3[location3 + 1];
                    buffer4[location4 + 2] = buffer3[location3 + 2];
                }
            }

            Marshal.Copy(buffer4, 0, pointer4, buffer3.Length);
            Image.UnlockBits(ImageData3);
            Image2.UnlockBits(ImageData4);
            pictureBox2.Image = Image2;
        }


    }
}
```

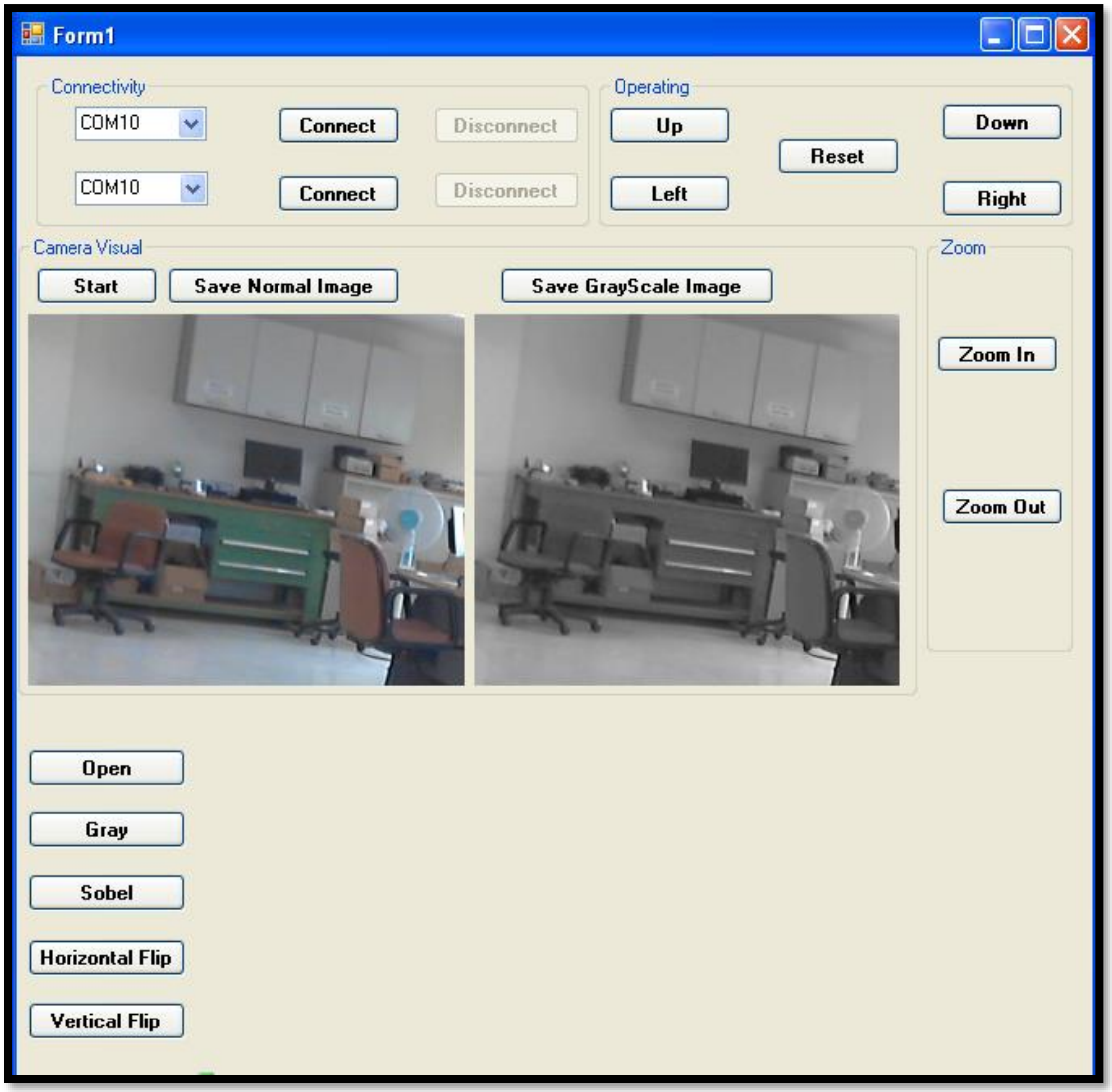
3.5 Output

Below figure shows the final setup of this project.

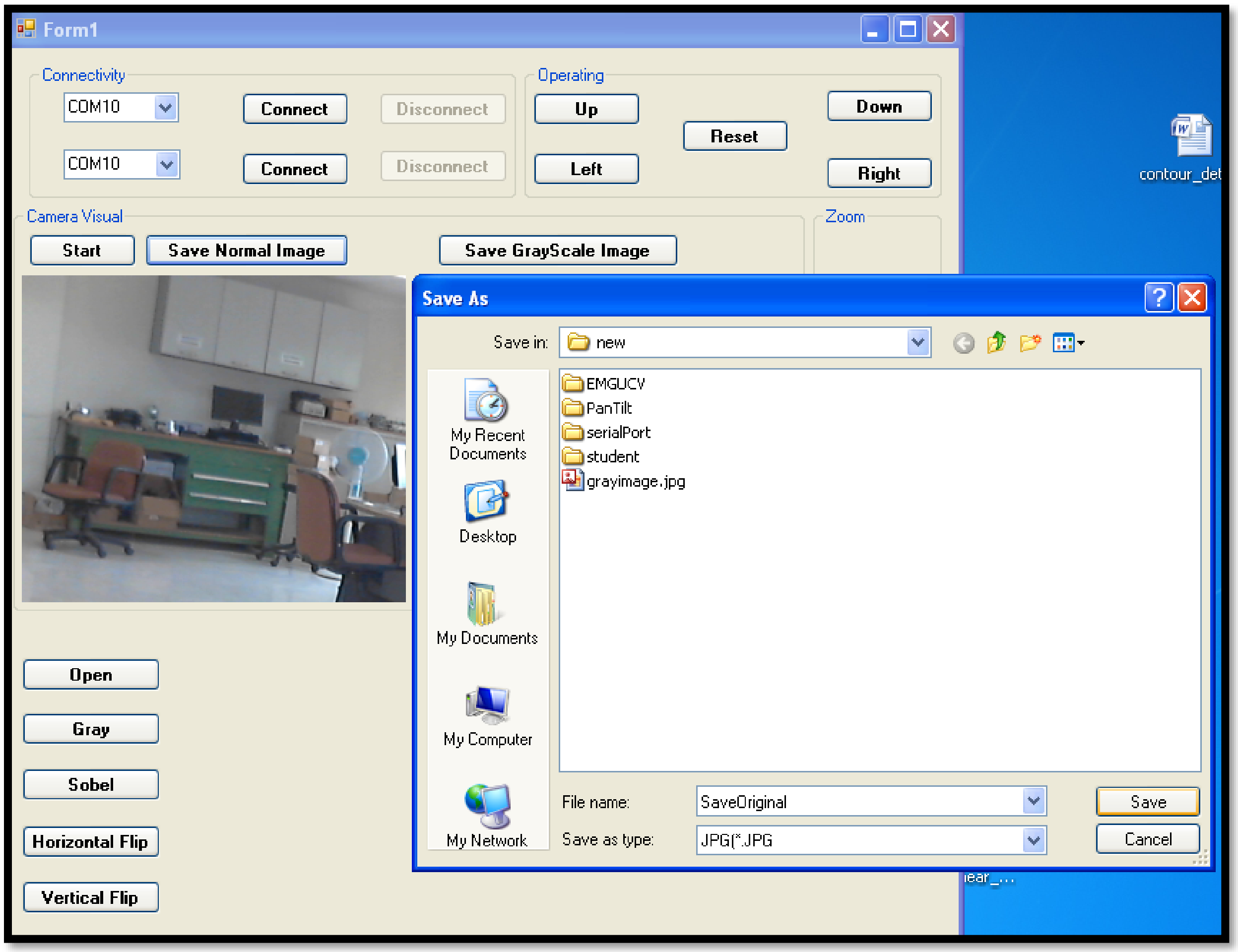
1) Normal output window of my project. In that there is two serial port connections, one is for to give proper commands for Pan-Tilt unit and second is for Zoom-In and Zoom-Out. There are five buttons are at the top of right i.e. Up, Down, Left, Right. We can move Pan-Tilt camera positions as per customers requirements. And another two buttons zoom-in and zoom-out are for zoom-in and out visuals.



2) I can start my default camera by clicking on ‘Start’ button and I can get these two camera visuals i.e. 1st is normal visual and 2nd is grayscale visual. If we click again on that button then we can pause the camera visual and the cycle will run continue.



3) After capture option, here we can save a normal image by clicking on ‘Save Normal Image’. We can save it on any location of computer as we want.

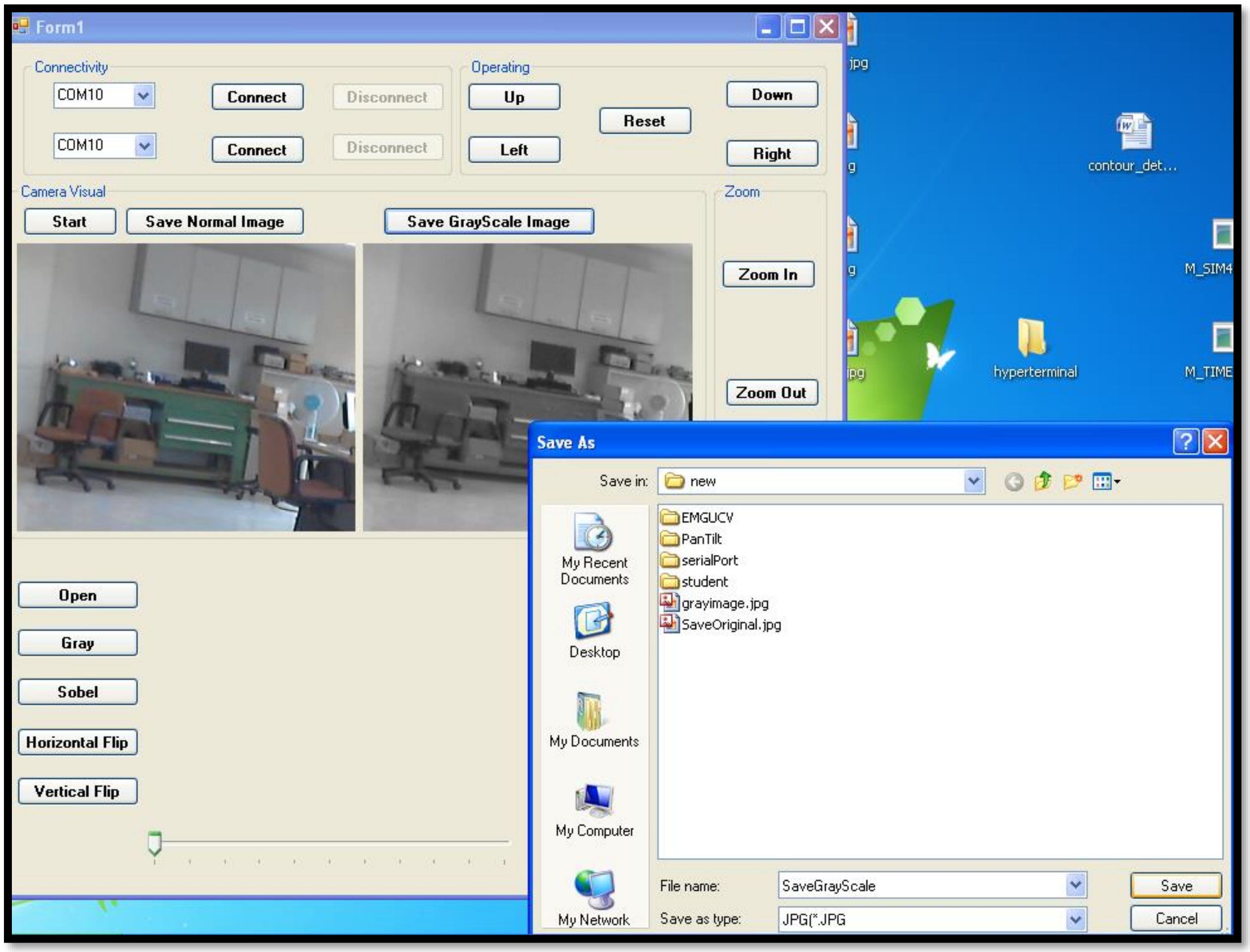


Save in: new

EMGUCV
PanTilt
serialPort
student
grayimage.jpg

File name: SaveOriginal
Save as type: JPG(*.JPG

4) Same, here we can save grayscale image by clicking on ‘Save Gray Scale Image’. We can save it on any location of computer as we want.

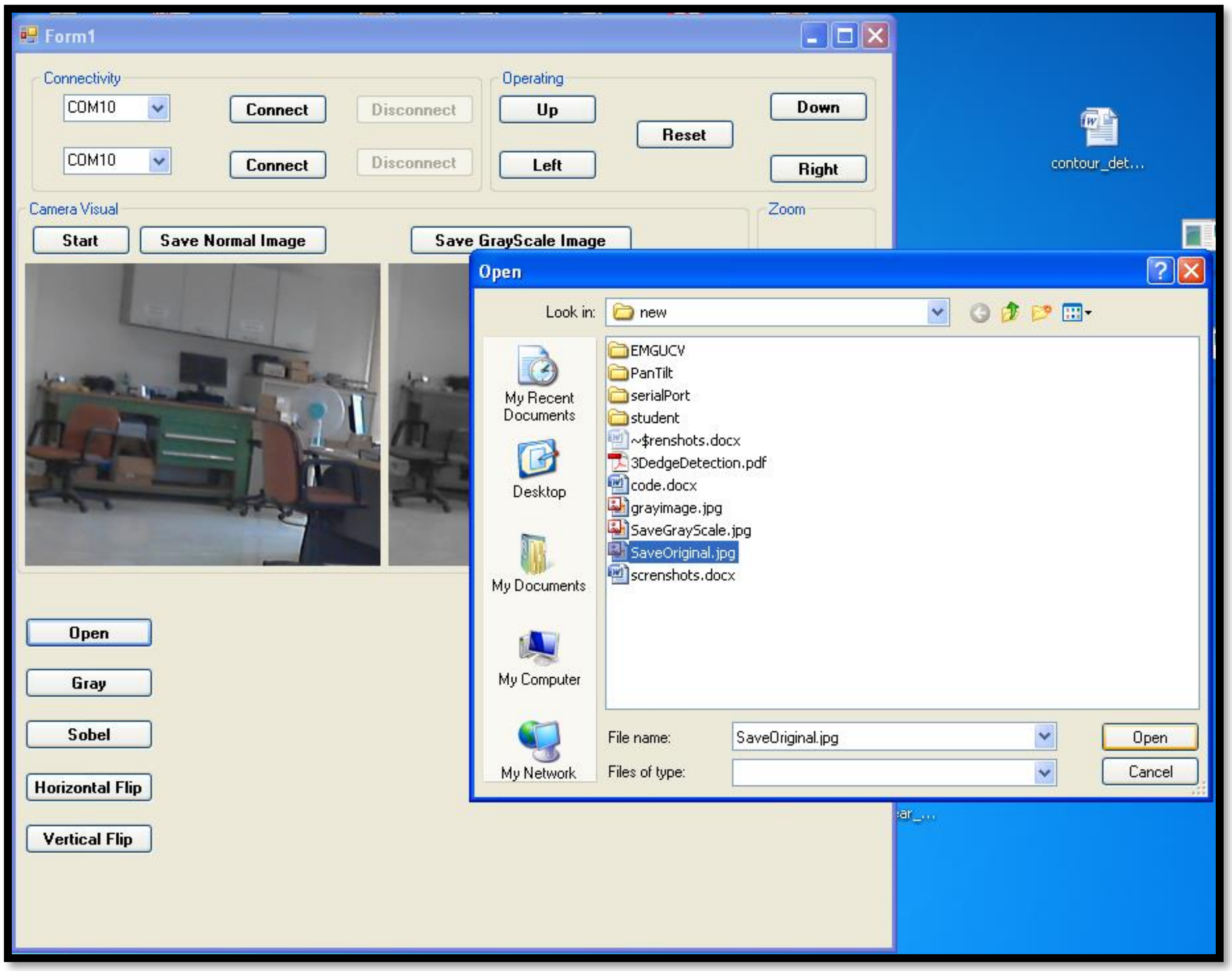


Save in: new

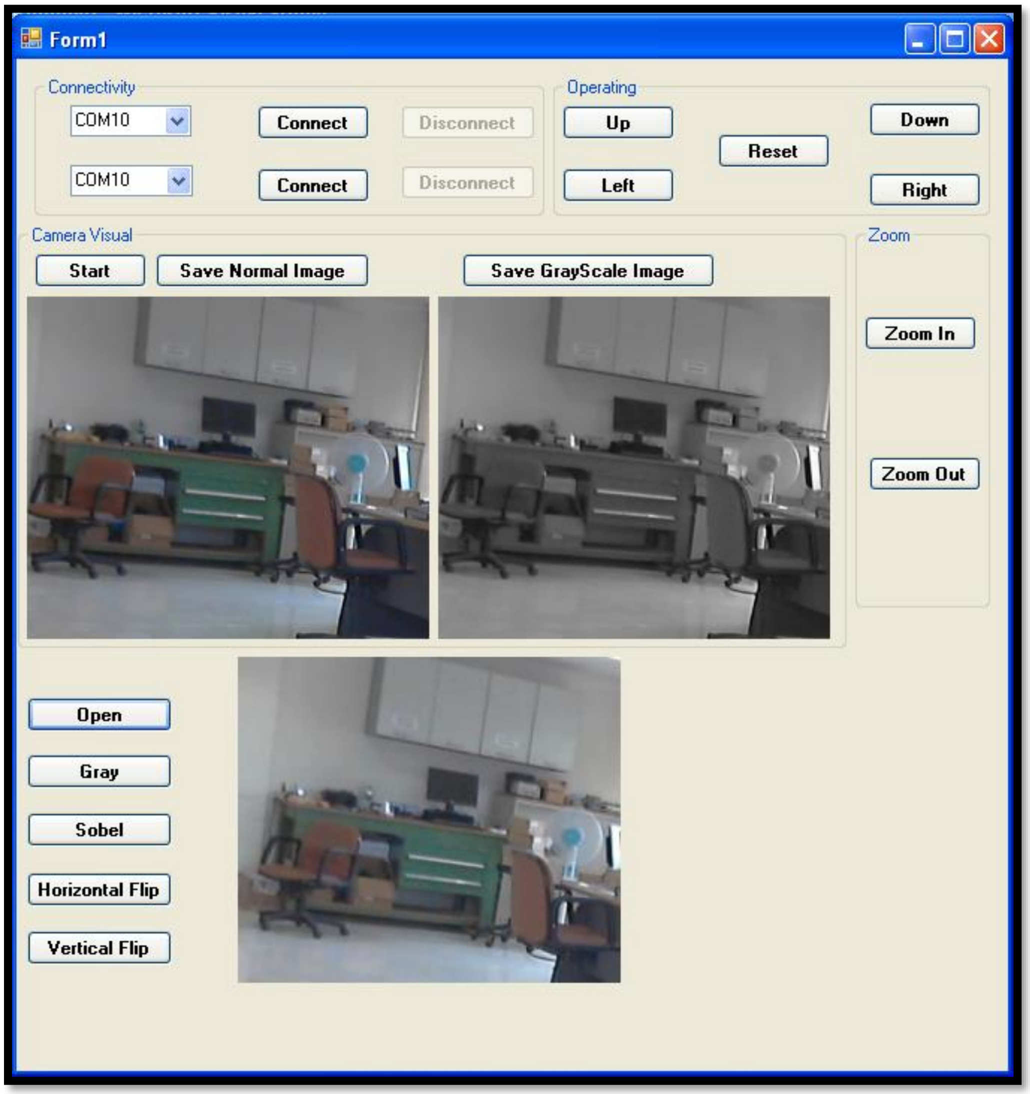
EMGUCV
PanTilt
serialPort
student
grayimage.jpg
SaveOriginal.jpg

File name: SaveGrayScale
Save as type: JPG(*.JPG

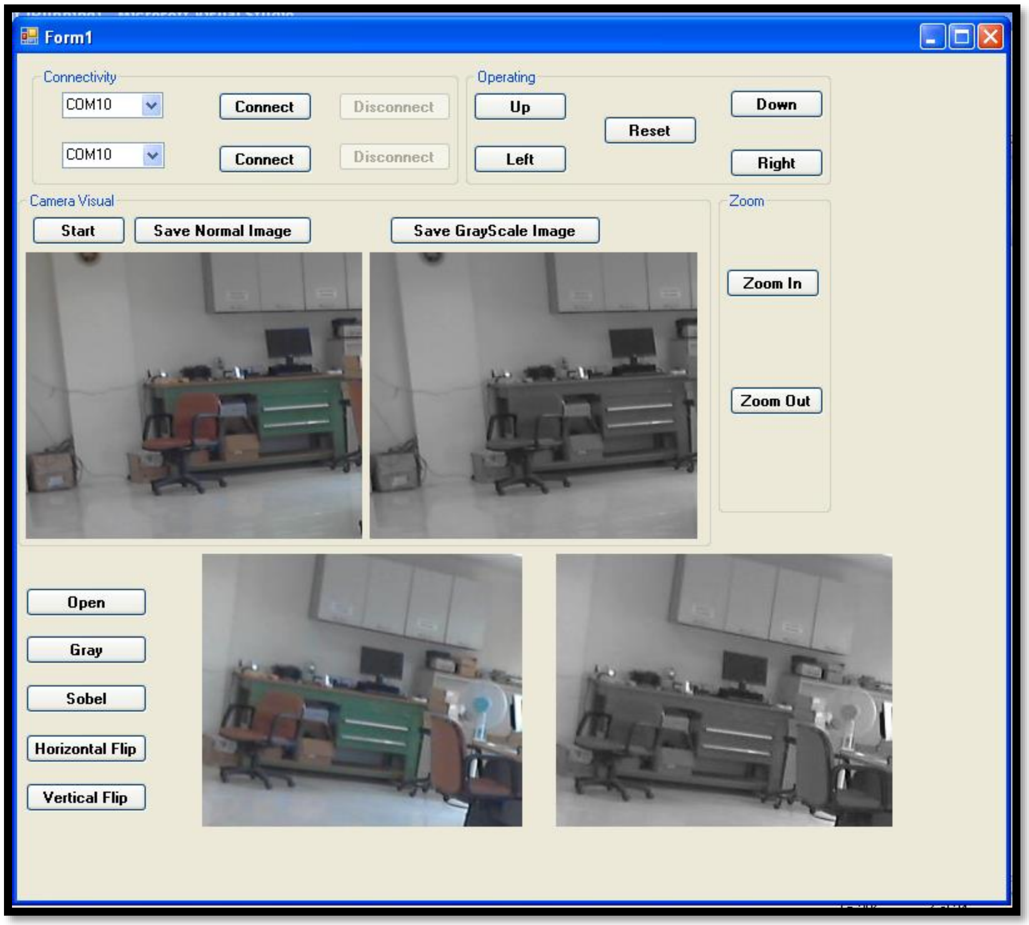
5) Here we can access that saved image from computer by clicking on ‘Browse Image’ button and we can see it in the below image box.



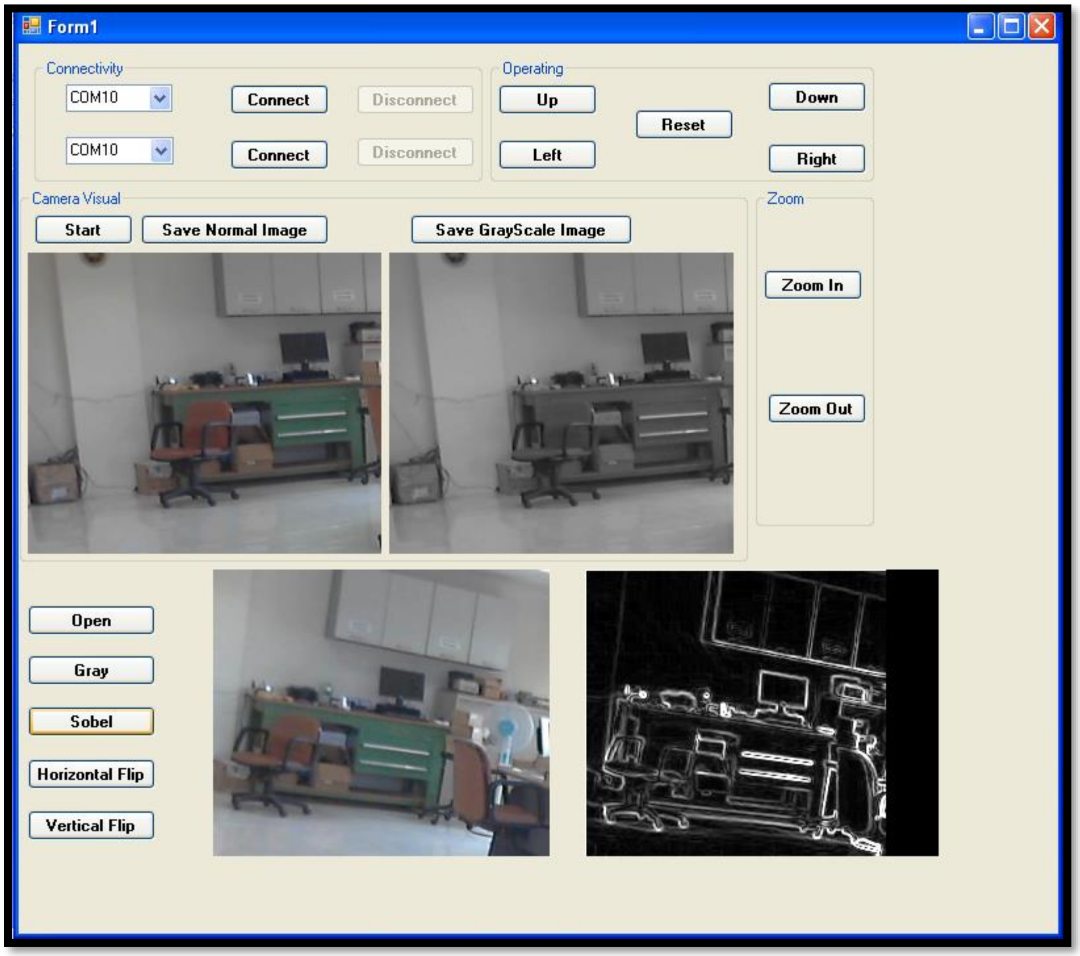
6) Here is the output of Open button. The image opened successfully.



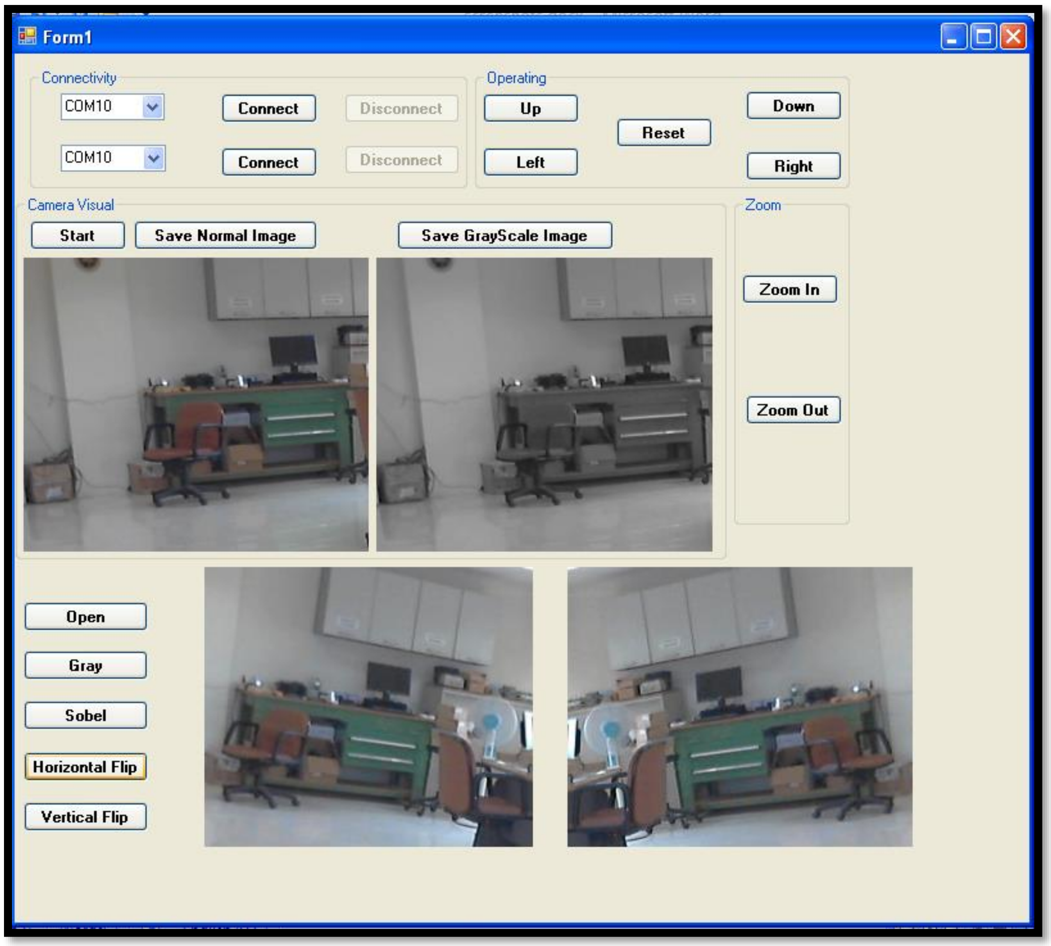
7) Below figure shows the output of gray button. It converts the original image to gray scale image.



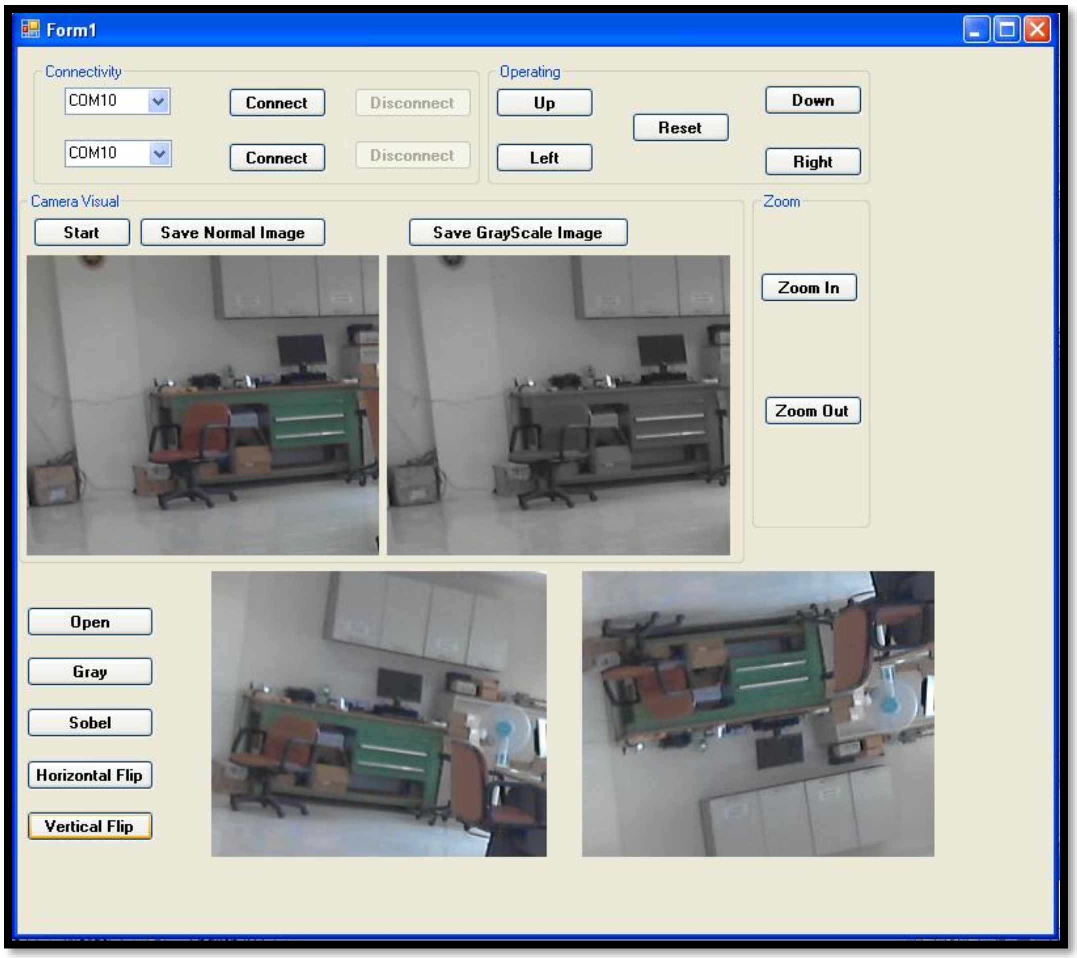
8) Below figure shows the sobel detection. Sobel detection is one of the type of edge detection.



9) Below figure shows the horizontally flip image.



10) Below figure shows the vertically flip image.



CHAPTER – 5

TROUBLESHOOTING

PTU mounted on ROV			
Sr. No.	**Name**	**Cause**	**Resolution**
1.	Mechanical reset not initiated at power on	Connector is loose or not connected	Check connector is attached properly
		DC power supply issue	On the harness connector, check pin A(+24VDC) and pin B(Ground) for power supply
2.	After mechanical reset PTU does not return to start position	Power supply issue or data communication error	Power down system and power up again. Normal reset should occur
		Mechanical alignment error due to excess external force	Unit needs to be serviced by authorized personnel
3.	Tilt axis or pan axis has overshot the mechanical limit range	PTU reset interrupted by power down	Switch off the ROV immediately. On next restart axis should return to normal reset position (though very slowly)
4.	Tilt axis or pan axis motor is making noise and axis is stuck at extreme mechanical range	PTU reset interrupted by power down	Switch off the ROV immediately. On next power ON, axis should return to normal position (though very slowly)
		Internal mechanical issue	Unit needs to be serviced by authorized personnel

PTU Stand Alone

Sr. No.	Issue	Cause	Resolution
1.	Mechanical reset not initiated at power on	Connector is loose or not connected	Check power and data connector is attached properly
		DC power supply issue	On the harness connector, check pin A(+24VDC) and pin B(Ground) for power supply. If power not present, troubleshoot the power supply lines.
		Internal RESET upon power ON is disabled	Check if internal reset upon power ON is enabled. Use the command **rq**. Reply should be **RST DISABLE**. Send command **re**.
2.	After mechanical reset PTU does not return to start position	Power supply issue or data communication error	Power down system and power up again. Normal reset should occur.
		Mechanical alignment error due to excess external force	Unit needs to be serviced by authorized personnel
3.	Tilt axis or pan axis has overshot the mechanical limit range	PTU reset interrupted by power down	Switch off power immediately. On next restart axis should return to normal reset position (though very slowly)
4.	Tilt axis or pan axis motor is making noise and axis is stuck at extreme mechanical range	PTU reset interrupted by power down	Switch off power immediately. On next power ON, axis should return to normal position (though very slowly)
		Internal mechanical issue	Unit needs to be serviced by authorized personnel

CHAPTER – 6

CONCLUSION

In this project, we have presented an image-based pan-tilt camera control technique by deriving the underlying characteristics of the projective rotations corresponding to camera pan-tilt. This image-based model greatly simplifies the task of camera control. This contribution would facilitate future development of a multi-camera surveillance system. Effectiveness and reliability of the technique have also been tested rigorously in a real-time pro-to type system.

The current Pan Tilt system has been designed and developed with the theoretical and practical understanding obtained in the beginning along with an earlier familiarity of working.

CHAPTER – 7

REFERENCES

- Wedler, M. Chalon, and et al., “DLRs dynamic actuator modules for robotic space applications” in Proc.:41st Aerospace Mechanisms Symposium (AMS), rev. Edward Boesiger (Lockheed Martin Space), vol. 41, 2012 Pasadena JPL/USA

- D. Griffiths, A. J. Coates, R. Jaumann, H. Michaelis,G. Paar, D. Barnes, J. Josset et. a Context for the ESA ExoMars Rover: the Panoramic Camera (PanCam) Instrument, International Journal of Astrobiology 5 (3) : 269– 275 (2006)

- Wedler, M. Chalon, and et al., “DLR’s space qualifiable multi-figered dexhand,” in Proc.:11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA), vol. 11, ESA. ESA/ESTEC, Noordwijk, the Netherlands: ESA, 12 14 April 2011, p. Session 3a

- Preusche, D. Reintsema, K. Landzettel, M. Fischer, and G. Hirzinger, “DLR on the way towards telepresent on-orbit servicing,” in Proc. Mechatronics & Robotics 2004, 2004

- Preusche, D. Reintsema, K. Landzettel, and G. Hirzinger, “ROKVISS - towards telepresence control in advanced space missions,” in Proc. 3rd. International Conference on Humanoid Robots (Humanoids 2003), Munich and Karlsruhe, Oct. 2003

- S. Gehrig, F. Eberli and T. Meyer, A Real-Time Low-Power Stereo Vision Engine Using SemiGlobal Matching, in International Conference on Computer Vision Systems (ICVS), October 2009 in Liege, Belgium