PROJECT REPORT

ON

INTERACTIVE COMPUTING USING HUMAN GESTURES

IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF

THE DEGREE IN BACHELOR OF ENGINEERING

(COMPUTER SCIENCE & ENGINEERING)

BATCH: 2006-2010

Submitted by:

Karthikeya Udupa K M (06/CS/118)

B.S. ANANGPURIA INSTITUTE OF TECHNOLOGY & MANAGEMENT
ALAMPUR, FARIDABAD (HARYANA)

(MAHARSHI DAYANAND UNIVERSITY, ROHTAK)

ACKNOWLEDGEMENT

I take this opportunity to thank all those who have helped me in completing the project successfully.

I sincerely thank Dr. S.S.Tyagi, Principal of B.S.Anangpuria Institue of Technology and Management, Alampur for providing a platform to build this project.

I would also like to show my gratitude to Mr. Pawan Bhadana (Asst. Prof. H.O.D. of the institute's Computer and IT department) for providing me with well trained faculty and giving the required resources and a healthy environment to carrying out the project work.

I am highly obliged to Ms. Neha Aggarwal (Faculty and Project Guide) for providing the continuous and invaluable guidance throughout the project. Her kind and elderly advice always inspired me in putting my best efforts to develop an efficient project.

Karthikeya Udupa K M (06/ CS/ 118)

B.S. Anangpuria Institute of Technology & Management

CERTIFICATE

This is to certify that the project work entitled, *Interactive Computing Using Human Gestures,* submitted by the student of *B.S. ANANGPURIA INSTITUTE OF TECHNOLOGY AND MANAGEMENT (2006-2010)* in partial fulfillment of the requirement for the degree B.E. of M.D. University, Rohtak, is a bonafide record of the work carried out by him under my supervision and is his original work.

Miss. Neha Aggarwal	**Mr. Pawan Bhadana**
(PROJECT GUIDE)	**(Asst. Prof H.O.D. CSE / IT)**

INDEX

Software Requirement Specification

1.1 Introduction

1.1.1 Purpose

The basic purpose of the project is to demonstrate the varied input source in the form of vision and ways to process it and analyze it and convert into defined input for the certain application. The advantage of this kind of project is allowing the user to allow people with basic idea of computers to operate it as in a real world situation with their movements and also to make usage of some application simpler then with the traditional inputs.

Also to allow the computer to be able to communicate with the user and provide a more soothing environment for his particular needs.

With the advancement in these technologies these things can be used in traffic and security cameras and detect threats and extract information as needed. This also would make lives lot easier for people who are not similar with computer and make learning new things easier.

This project also gives the base structure to other complex fields such as of artificial intelligence and robotics as vision is a key aspect for artificial intelligence and with the ability to see and understand humans and their expressions the field will certainly benefit.

1.1.2 Contact Information

Name : Karthikeya Udupa K M

Roll No : 06/CS/118

Email : karthikeyaudupa@gmail.com

1.2 Overall Description

1.2.1 Product Perspective

The basic idea of the project is to create an interface for using computer without having to restrict the input to mouse and keyboard. With the advancement in the field of hardware and the processing power in the recent years it has allowed using various other devices to give the computer as input. Allowing the system to be more related to real world and also allowing people with basic knowledge of computer to use it and do their day to day jobs with it.

1.1.2 Product Features

Some of the features that the software is supposed to have are as follows:

- Multiple human features detection.
- Tracking various features when needed.
- Identifying individuals and keep track of their information.

1.3 Hardware & Software Development Environment

1.3.1 Hardware Specification

- 1GHz Pentium 4 processor or better.
- 1GB RAM or more.
- 100MB Hard Disk Space.
- USB connectable Camera (Preferably 2.0 Mega Pixel or more).

1.3.2 Software Specification

- Visual C# 2008 Express Edition
- EmguCV/OpenCV Binaries
- Windows XP SP2 or above.

1.3.3 Front End Language Description

C# (pronounced "see sharp") is a multi-paradigm programming language encompassing imperative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure.

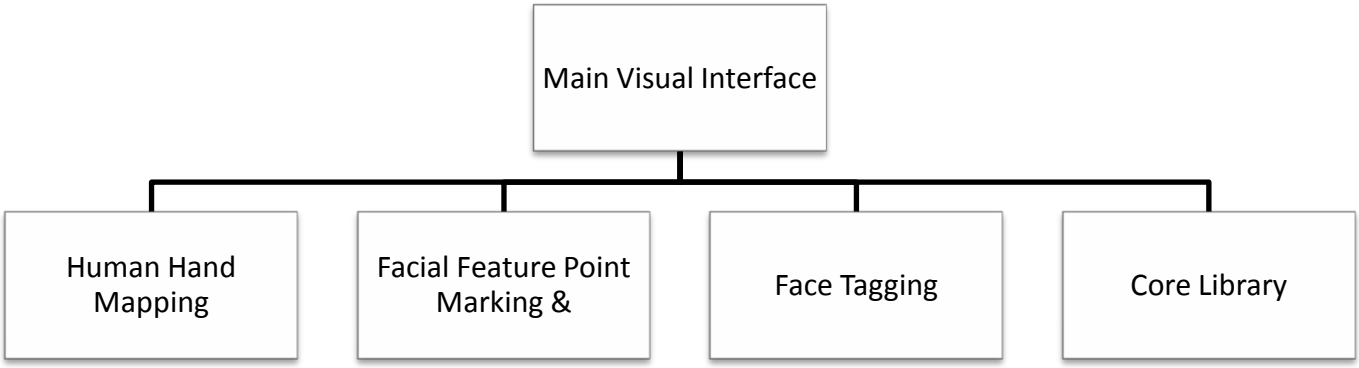
C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 4.0, which was released in April 12, 2010.

By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework.

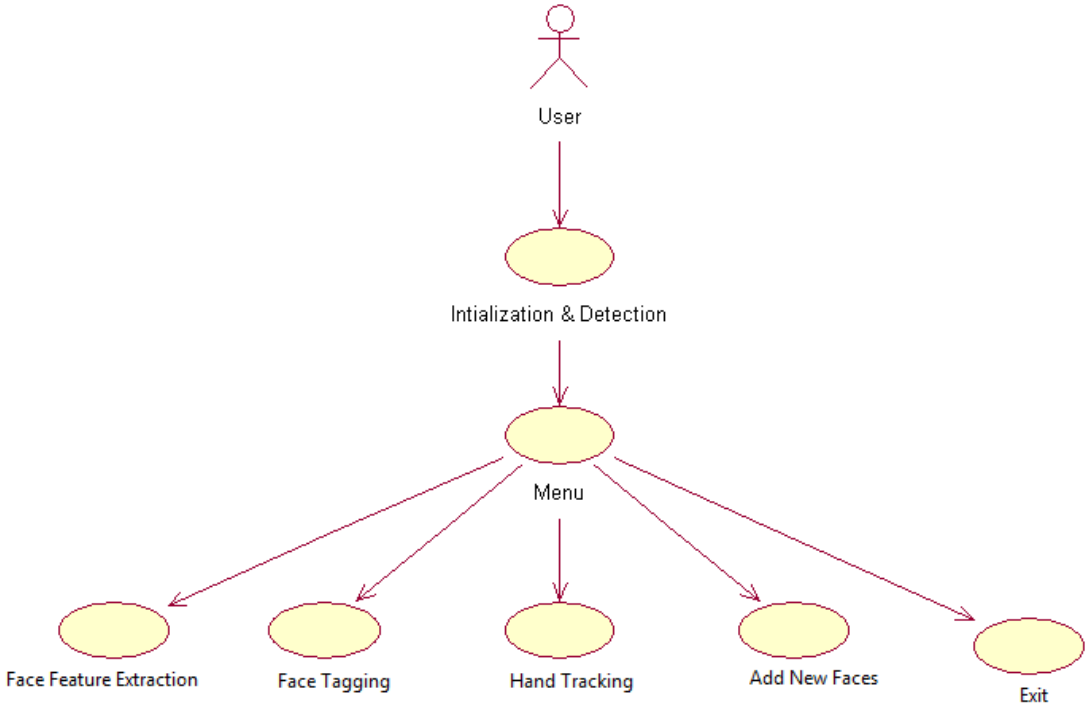
However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN.

Design Methodology

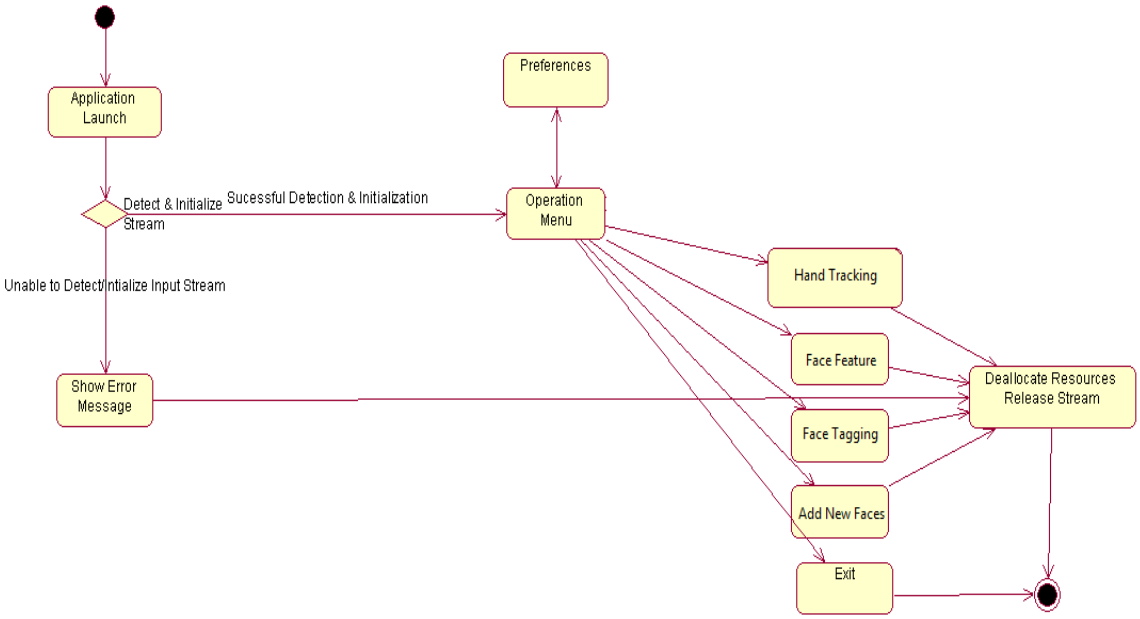
2.1 Structural Chart



2.2 Usecase Diagram



2.3 Activity Diagram



Coding

3.1 Update_DB.CS

```
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 Emgu;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.Util.TypeEnum;

namespace OpenCVApp
{

    public partial class UpdateDB : Form
    {
        SampleCollector sc;
        private Capture StreamCapture;
        Image<Bgr, byte> cur_frame;
        int click_cnt,cam_on = 0;
        string def_file_path = "C:\\Users\\Karthik\\Documents\\Code\\demofaces\\";

        public UpdateDB()
        {
            InitializeComponent();
            Application.Idle += handl;

             sc = new SampleCollector("haarcascade_frontalface_alt_tree.xml", 92, 112,
             "C:\\Users\\Karthik\\Documents\\Code\\facelist.xml");
            List<string> cur_rec = sc.MoveToPointer();
            populateboxes(cur_rec);
            click_cnt = 1;
        }

        private int InitCapture()
        {

            if (StreamCapture == null)
            {
                try
                {
                    StreamCapture = new Capture(0);
                    return 1;
                }
                catch (NullReferenceException excpt)
                {
                    MessageBox.Show(excpt.Message);
```

```
                return 0;
            }

        }
        else
        {
            return 1;
        }

    }

    private void Previous_Click(object sender, EventArgs e)
    {
        populateboxes(sc.movePrevious());
    }

    private void handl(object sender, EventArgs e)
    {
        if (cam_on == 1)
            cur_frame = StreamCapture.QuerySmallFrame();
            imageBox4.Image = cur_frame;

    }


    private void populateboxes(List<string> data)
    {
        txtID.Text = data[0];
        txtName.Text = data[1];
        txtDesc.Text = data[2];
        txtLocation.Text = data[3];
        loadimgs(data[3]);
    }

    private void loadimgs(string uribase)
    {


        imageBox2.Image = new Image<Gray, byte>(uribase + "2.pgm");
        imageBox1.Image = new Image<Gray, byte>(uribase + "1.pgm");
        imageBox3.Image = new Image<Gray, byte>(uribase + "3.pgm");
    }

    private void Next_Click(object sender, EventArgs e)
    {
        populateboxes(sc.moveNext());
    }

    private void button2_Click(object sender, EventArgs e)
    {
       if (txtID.Text != "" || txtName.Text != "" || txtDesc.Text != "" ||
txtLocation.Text != "")
        {
            sc.updatenode(txtName.Text, txtID.Text, txtLocation.Text, txtDesc.Text);
        }


    }

    private void button4_Click(object sender, EventArgs e)
    {
```

```
            if (txtID.Text != "" || txtName.Text != "" || txtDesc.Text != "" ||
        txtLocation.Text != "")
            {
                sc.writenode(txtName.Text, txtID.Text, txtLocation.Text, t      xtDesc.Text);
            }
            button4.Enabled = false;
            button1.Enabled = true;
            button6.Enabled = true;
            cam_on = 0;
            StreamCapture.Dispose();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            imageBox1.Image = null; imageBox2.Image = null; imageBox3.Image = null;
            txtID.Text = (sc.MAX +1).ToString();
            txtName.Text = "";
            txtDesc.Text = "";
            txtLocation.Text = def_file_path;
            button4.Enabled = true;
            button1.Enabled = false;
            button6.Enabled = true;
            button5.Enabled = true;
            InitCapture();
            cam_on = 1;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            sc.deletenode();
        }


        private void button5_Click(object sender, EventArgs e)
        {
            label1.Text = click_cnt.ToString();
            List<face_data> b = new List<face_data>();
            string loc = txtLocation.Text;
            FaceExtractor a = new FaceExtractor("haarcascade_frontalface_alt_tree.xml", 92,
    112, false);
            b = a.FaceDetector((cur_frame.Convert<Gray, byte>()));
            cur_frame.Draw(b[0].rect, new Bgr(Color.Aqua), 2);

            if (click_cnt == 1)
                System.IO.Directory.CreateDirectory(loc);

            switch (click_cnt)
            {
                case 1:
                    imageBox1.Image = b[0].img_GB_format;
                    break;
                case 2:
                    imageBox2.Image = b[0].img_GB_format;
                    break;
                case 3:
                    imageBox3.Image = b[0].img_GB_format;
                    break;
            }
                b[0].imgface.Save(loc+"\\"+click_cnt+".pgm");

            if(click_cnt == 3)
```

```
                button5.Enabled = false;
            click_cnt++;
        }

        private void button6_Click(object sender, EventArgs e)
        {
            folderBrowserDialog1.RootFolder = Environment.SpecialFolder.MyDocuments;
            folderBrowserDialog1.ShowDialog();
            txtLocation.Text = folderBrowserDialog1.SelectedPath;

        }

    }
}
```

3.2 FaceTag.cs

```
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 Emgu;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.Util.TypeEnum;
using Lastfm.Services;
using System.Net;
using System.IO;
namespace OpenCVApp
{
    public partial class Form2 : Form
    {

        private Capture StreamCapture;

        public Form2()
        {
            InitCapture();
            InitializeComponent();
            init_lastfm();
            Application.Idle += handl;
        }

        Image<Bgr, byte> cur_frame;
        private void handl(object sender, EventArgs e)
        {

            cur_frame = StreamCapture.QuerySmallFrame();//fetchstreamimg();
            imageBox3.Image = cur_frame;
        }

        private void button1_Click(object sender, EventArgs e)
        {
```

```
            FaceRecog a = new
FaceRecog("C:\\Users\\Karthik\\Documents\\Code\\facelist.xml",5000,30);
            a.populateimages();
            FaceRecog.ImageStruct found_match = new FaceRecog.ImageStruct();
            found_match = a.searchimg(c);
            if (found_match.loc != null)
                imageBox1.Image = new Image<Gray, byte>(found_match.loc + "1.pgm");
            label1.Text = found_match.name;
           // label2.Text = found_match.desc;

            fetchrecent(found_match.name);
          }
        Image<Gray, byte> c;
       private void button2_Click(object sender, EventArgs e)
        {
            Bitmap b;
            FaceExtractor a = new
FaceExtractor("haarcascade_frontalface_alt_tree.xml",92,112,false);
            b= a.FaceDetector((cur_frame.Convert<Gray, byte>()))[0].imgface;

            c = new Image<Gray, byte>(b);
            imageBox2.Image = c;
        }

        private int InitCapture()
        {

            if (StreamCapture == null)
            {
                try
                {
                    StreamCapture = new Capture(0);
                    return 1;
                }
                catch (NullReferenceException excpt)
                {
                    MessageBox.Show(excpt.Message);
                    return 0;
                }

            }
            else
            {
                return 1;
            }
        }

        //last.fm code
        Session session;
        private void init_lastfm()
        {
            string API_KEY = "08fdb45815c362be3ce6441a5bf7a0bc";
            string API_SECRET = "02860815b33edc24753f7849cc94f1c7";


            session = new Session(API_KEY, API_SECRET);

            session.Authenticate("insanoid", Lastfm.Utilities.md5("1123581321"));

        }
```

```
        private List<Track> fetchrecent(string uid)
        {

            List<Track> songlist = new List<Track>();
            Lastfm.Services.User u = new User(uid, session);
            Track[] t = u.GetRecentTracks(5);
            foreach (Track tr in t){
                songlist.Add(tr);
                label2.Text += "\n" + tr.Title + " - " +tr.Artist.Name;
                }
            return songlist;
        }

        private void fetchalbumart( List<Track> tracks)
        {
            List<string> uriArt= new List<string>();
            List<Bitmap> ArtList = new List<Bitmap>();

            for (int i = 0; i < 5; i++)
            {

                uriArt.Add(tracks.ElementAt(i).GetAlbum().GetImageURL());
            }

            foreach (string s in uriArt)
            {
                WebClient client = new WebClient();
                Stream stream = client.OpenRead(s);
                Bitmap bitmap = new Bitmap(stream);
                ArtList.Add(bitmap);
                stream.Flush();
                stream.Close();
            }
        }

    }
}
```

3.3 HandTracking.cs

```
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 Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using Emgu.CV.UI;

namespace OpenCVApp
{
    public partial class camshift : Form
    {
        private Capture StreamCapture;
```

```
        HaarCascade cascade_face = new HaarCascade("haarcascade_frontalface_alt_tree.xml");
        HaarCascade eye = new HaarCascade("haarcascade_eye.xml");
        HaarCascade nose = new HaarCascade("nose.xml");
        HaarCascade mouth = new HaarCascade("mouth.xml");
        HaarCascade eyer = new HaarCascade("ojoD.xml");
        HaarCascade eyel = new HaarCascade("ojoI.xml");
        HaarCascade agest = new HaarCascade("aGest.xml");
        private int InitCapture()
        {

            if (StreamCapture == null)
            {
                try
                {
                    StreamCapture = new Capture(1);
                    return 1;
                }
                catch (NullReferenceException excpt)
                {
                    MessageBox.Show(excpt.Message);
                    return 0;
                }

            }
            else
            {
                return 0;
            }


        }

        public camshift()
        {
            InitializeComponent();
            InitCapture();
            Application.Idle += DoStuff;
        }

        private void starttrack(object sender, MouseEventArgs e)
        {

        }
        Image<Bgr, byte> originalImage;
        Image<Hsv, byte> hsvImage;
        Image<Gray, byte> cannyImage, greyedImage, goodFeatures;
        List<Contour<Point>> largestContours = new List<Contour<Point>>();

        void DoStuff(object sender, EventArgs e)
        {
            originalImage = StreamCapture.QuerySmallFrame();

                ibVideoImage.Image = originalImage;
                greyedImage = Makemask(originalImage);
                imageBox1.Image = greyedImage;
                cannyImage = greyedImage.Canny(new Gray(1), new Gray(3)).SmoothGaussian(3);//
Dilate(10);

                goodFeatures = greyedImage.CopyBlank();
                ibProcessedImage.Image = cannyImage;
```

```
                using (MemStorage storage = new MemStorage())
                    for (Contour<Point> contours =
cannyImage.FindContours(Emgu.CV.CvEnum.CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_NONE,Emgu.CV.CvEnum
.RETR_TYPE.CV_RETR_LIST, storage);
                              contours != null;
                              contours = contours.HNext)
                    {
                        Contour<Point> currentContour = contours.ApproxPoly(contours.Perimeter
* 0.001, storage);
                        if (currentContour.Area > 700)
                        {
                            largestContours.Add(currentContour);

                            Seq<MCvConvexityDefect> defects =
currentContour.GetConvexityDefacts(storage, Emgu.CV.CvEnum.ORIENTATION.CV_CLOCKWISE);
                            int numOfDefects = defects.Total;
                            if (numOfDefects == 0)
                                continue;

                            MCvConvexityDefect[] defectArray = defects.ToArray();


                            for (int i = 0; i < numOfDefects; i++)
                            {
                                LineSegment2D line = new
LineSegment2D(defectArray[i].StartPoint, defectArray[i].DepthPoint);

                                PointF depthPoint = new
PointF((float)defectArray[i].DepthPoint.X,

(float)defectArray[i].DepthPoint.Y);

                                CircleF depthCircle = new CircleF(depthPoint, 5f);

                                PointF startPoint = new
PointF((float)defectArray[i].StartPoint.X,

(float)defectArray[i].StartPoint.Y);

                                CircleF startCircle = new CircleF(startPoint, 5f);
                                LineSegment2D line2 = new
LineSegment2D(defectArray[i].DepthPoint, defectArray[i].EndPoint);

                                originalImage.Draw(line, new Bgr(Color.Green), 1);
                                originalImage.Draw(depthCircle, new Bgr(Color.Green), 1);
                                originalImage.Draw(startCircle, new Bgr(Color.Cyan), 1);
                                originalImage.Draw(line2, new Bgr(Color.Green), 1);
                            }
                            originalImage.ROI = Rectangle.Empty;
                            ibVideoImage.Image = originalImage ;

                    }

            }


        }
        private static Image<Gray, Byte> Makemask(Image<Bgr, byte> img)
        {
            using (Image<Hsv, Byte> hsvImg = img.Convert<Hsv, Byte>())
            {
```

```
                Image<Gray, Byte>[] channel = hsvImg.Split();

                CvInvoke.cvInRangeS(channel[0], new MCvScalar(0), new MCvScalar(20),
channel[0]);
                CvInvoke.cvInRangeS(channel[1], new MCvScalar(30), new MCvScalar(150),
channel[1]);
                CvInvoke.cvInRangeS(channel[2], new MCvScalar(80), new MCvScalar(255),
channel[2]);
                channel[0]._Not();

                channel[1]._ThresholdBinary(new Gray(10), new Gray(255.0));

                CvInvoke.cvAdd(channel[0], channel[1], channel[0], IntPtr.Zero);
                channel[1].Dispose();
                channel[2].Dispose();

                return channel[0];
            }
        }
    }
}
```

3.4 FaceExtractor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Emgu;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.Util.TypeEnum;
using System.Drawing;

namespace OpenCVApp
{

    public struct face_data
        {
            public Bitmap imgface;
            public Rectangle rect;
            public Image<Gray, byte> img_GB_format;
        }

    class FaceExtractor
    {
        protected HaarCascade objectCascade;
        protected int nWidth, nHeight;
        public MCvAvgComp[][] facesDetected;
        protected Emgu.CV.CvEnum.HAAR_DETECTION_TYPE selType;



        public FaceExtractor(string oCascade,int nW,int nH,Boolean Multiple) {
            objectCascade = new HaarCascade(oCascade);
            nWidth = nW;
            nHeight = nH;
            if (Multiple == true)
```

```
                selType = Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING;
            else
                selType = Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT;
        }


        public Bitmap ResizeBitmap(Bitmap b, int nW, int nH)
        {
            Bitmap result = new Bitmap(nW, nH);
            using (Graphics g = Graphics.FromImage((Image)result))
                g.DrawImage(b, 0, 0, nW, nH);
            return result;

        }
        public List<face_data> FaceDetector(Image<Gray, byte> grayframe)
        {
            List<face_data> replyFaces = new List<face_data>();
            facesDetected = grayframe.DetectHaarCascade(objectCascade, 1.1, 1, selType, new
Size(40, 40)); //FIND_BIGGEST_OBJECT
            foreach (MCvAvgComp face in facesDetected[0])
            {
                face_data temp_face = new face_data();

                grayframe.ROI = face.rect;
                Bitmap temp = new Bitmap(grayframe.ToBitmap());

                temp_face.imgface = ResizeBitmap(temp, nWidth, nHeight);
                temp_face.rect = face.rect;
                temp_face.img_GB_format = new Image<Gray, byte>(temp);

                replyFaces.Add(temp_face);
                grayframe.ROI = Rectangle.Empty;
            }

            return replyFaces;

        }


    }
}
```

3.5 CVEventClass.cs

```
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 Emgu;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.Util.TypeEnum;
using System.Xml;
using System.IO;
```

```
namespace OpenCVApp
{
    /// <summary>
    /// Eigen Face Recognition System.
    /// </summary>

    class FaceRecog
    {
        public struct ImageStruct {
           public int id;
           public string name;
           public string desc;
           public string loc;
        }

        public int Populate_cnt = 3;
        public  int MAX_ITERATION = 30;
        public double MAX_THRESHOLD = 4000;

        public string path;
        Image<Gray, byte>[] TrainingImageList;
        String[] TrainingLabelList;

        List<ImageStruct> TrainingData= new List<ImageStruct>();

        public FaceRecog(string XMLpath, double MAX_EIGEN_THRESHOLD, int MAXIMUM_ITERATION)
        {
            path = XMLpath;
            MAX_ITERATION = MAXIMUM_ITERATION;
            MAX_THRESHOLD = MAX_EIGEN_THRESHOLD;
        }

        public void populateimages(){

            int MAX, cnt;
            XmlDocument xmldoc = new XmlDocument();


            FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read,
FileShare.ReadWrite);
            xmldoc.Load(fs);

            XmlNodeList xmlnode = xmldoc.GetElementsByTagName("Face");
            MAX = xmlnode.Count;
            TrainingImageList = new Image<Gray,byte>[MAX*Populate_cnt];
            TrainingLabelList = new string[MAX*Populate_cnt];


            for(cnt=0;cnt<MAX;cnt++){
                 XmlAttributeCollection xmlattrc = xmlnode[cnt].Attributes;
                 TrainingLabelList[cnt] = xmlattrc[0].Value;
                 ImageStruct tempNew= new ImageStruct();
                 tempNew.id = int.Parse(xmlattrc[0].Value);
                 tempNew.name = xmlnode[cnt].FirstChild.InnerText;
                 tempNew.desc = (xmlnode[cnt].LastChild.InnerText);
                 string uricur = xmlnode[cnt].ChildNodes[1].InnerText;
                 tempNew.loc = uricur;

                 TrainingData.Add(tempNew);
                 for (int i = 0; i < Populate_cnt; i++)
                 {
```

```
                    TrainingImageList[cnt * Populate_cnt + i] = (new Image<Gray, byte>(uricur
+ (i+1) + ".pgm"));
                    TrainingLabelList[cnt * Populate_cnt + i] = tempNew.id.ToString();
                }
            }


        }



        public ImageStruct searchimg(Image<Gray, byte> srimg) {
            MCvTermCriteria termCrit = new MCvTermCriteria(MAX_ITERATION, 0.001);
            EigenObjectRecognizer recognizer = new EigenObjectRecognizer(TrainingImageList,
TrainingLabelList,MAX_THRESHOLD, ref termCrit);

            String Recg_label = recognizer.Recognize(srimg);
            ImageStruct foundVal = FetchID(Recg_label);
            return foundVal;
        }


        private ImageStruct FetchID(string Recg_label)
        {
            ImageStruct ret_Value = new ImageStruct();
            for (int cnt = 0; cnt < TrainingData.Count; cnt++)
            {
                if (TrainingData[cnt].id.ToString() == Recg_label)
                    ret_Value = TrainingData[cnt];
            }
            return ret_Value;
        }


    }

}
```

3.6 SampleCollector.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Emgu;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.Util.TypeEnum;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Xml;
namespace OpenCVApp
{
    class SampleCollector
    {
        FaceExtractor Faceext;
        private XmlDocument xmldoc;
        private string path; // = "C:\\Users\\Karthik\\Documents\\Code\\facelist.xml"
        public int xml_cur_pointer, MAX;
```

```
        public SampleCollector(string oCascade,int nW,int nH,string xmlpath)
         {
             Faceext = new FaceExtractor(oCascade, nW, nH, false);
             path = xmlpath;

             xmldoc = new XmlDocument();

            FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read,
FileShare.ReadWrite);
            xmldoc.Load(fs);
            xml_cur_pointer = 0;

         }

        public List<string> MoveToPointer()
        {
            List<string> reply_element = new List<string>();
            XmlNodeList xmlnode = xmldoc.GetElementsByTagName("Face");

            MAX = xmlnode.Count;

            XmlAttributeCollection xmlattrc = xmlnode[xml_cur_pointer].Attributes;

            reply_element.Add(xmlattrc[0].Value);
            reply_element.Add(xmlnode[xml_cur_pointer].FirstChild.InnerText);
            reply_element.Add(xmlnode[xml_cur_pointer].LastChild.InnerText);
            reply_element.Add(xmlnode[xml_cur_pointer].ChildNodes[1].InnerText);

            return reply_element;

        }

        public void updatenode(string name, string id, string loc, string desc)
        {

            XmlNodeList xmlnode = xmldoc.GetElementsByTagName("Face");

            XmlAttributeCollection xmlattrc = xmlnode[xml_cur_pointer].Attributes;
            xmlattrc[0].Value = id;
            xmlnode[xml_cur_pointer].FirstChild.InnerText = name;
            xmlnode[xml_cur_pointer].LastChild.InnerText = desc;
            xmlnode[xml_cur_pointer].ChildNodes[1].InnerText = loc;

            FileStream fsxml = new FileStream(path, FileMode.Truncate, FileAccess.Write,
FileShare.ReadWrite);
            xmldoc.Save(fsxml);

        }

        public void writenode(string name, string id, string loc, string desc)
        {

            XmlElement newcatalogentry = xmldoc.CreateElement("Face");
            XmlAttribute newcatalogattr = xmldoc.CreateAttribute("ID");

            newcatalogattr.Value = id;

            newcatalogentry.SetAttributeNode(newcatalogattr);
            XmlElement firstelement = xmldoc.CreateElement("Name");
```

```
            firstelement.InnerText = name;
            newcatalogentry.AppendChild(firstelement);


            XmlElement secondelement = xmldoc.CreateElement("baseuri");
            secondelement.InnerText = loc;
            newcatalogentry.AppendChild(secondelement);

            XmlElement thirdelement = xmldoc.CreateElement("Desc");
            thirdelement.InnerText = desc;
            newcatalogentry.AppendChild(thirdelement);

            xmldoc.DocumentElement.InsertAfter(newcatalogentry,
xmldoc.DocumentElement.LastChild);

            FileStream fsxml = new FileStream(path, FileMode.Truncate, FileAccess.Write,
FileShare.ReadWrite);
            xmldoc.Save(fsxml);

        }

        public void deletenode()
        {
             XmlNodeList xmlnode = xmldoc.GetElementsByTagName("Face");

             string dir = xmlnode[xml_cur_pointer].ChildNodes[1].InnerText;
             System.IO.Directory.Delete(dir, true);

             XmlNode n = xmldoc.SelectSingleNode("FaceStruct/Face[@ID='" + (xml_cur_pointer+1)
+ "']");

             n.ParentNode.RemoveChild(n);
             FileStream fsxml = new FileStream(path, FileMode.Truncate, FileAccess.Write,
FileShare.ReadWrite);
             xmldoc.Save(fsxml);
             xml_cur_pointer = 1;
             MoveToPointer();

        }

        public face_data extract_face(Image<Gray, byte> grayframe)
         {
             List<face_data> temp_face = new List<face_data>();
             temp_face = Faceext.FaceDetector(grayframe);
             if (temp_face.Count == 0) {
                 srsfailures(0);
                 return temp_face[0];
             }
             else
               return temp_face[0];
         }

        public List<string> moveNext()
        {
            if (xml_cur_pointer < MAX - 1)
              xml_cur_pointer++;
            else
              xml_cur_pointer = 0;

            return MoveToPointer();
```

```
        }

        public List<string> movePrevious()
        {
            if (xml_cur_pointer > 0)
             xml_cur_pointer--;
            else
             xml_cur_pointer = MAX - 1;

           return MoveToPointer();
        }

         private void srsfailures(int p)
         {
             throw new NotImplementedException();
         }
    }
}
```

3.7 FaceMap.cs

```
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 Emgu;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.Util.TypeEnum;
using System.Diagnostics;

namespace OpenCVApp
{
    public partial class FaceMap : Form
    {
        private Capture StreamCapture;
        HaarCascade cascade_face = new HaarCascade("haarcascade_frontalface_alt_tree.xml");
        HaarCascade eye = new HaarCascade("haarcascade_eye.xml");
        HaarCascade nose = new HaarCascade("nose.xml");
        HaarCascade mouth = new HaarCascade("mouth.xml");
        HaarCascade eyer = new HaarCascade("ojoD.xml");
        HaarCascade eyel = new HaarCascade("ojoI.xml");

        PointF fc1, fc2;

        #region init_frames

        public FaceMap()
        {
            InitializeComponent();

            int replyCode;
            replyCode = InitCapture();
```

```
            if (replyCode == 1)
            {
                Application.Idle += handl;
            }
            else
            {
                MessageBox.Show("Error in initializing the Stream,Please check your camera.");
                Application.Exit();
            }

            System.Threading.Thread.Sleep(5000);
            facecapinit();
            populatepoints();
            opFlowCal();
        }

        private int InitCapture()
        {

            if (StreamCapture == null)
            {
                try
                {
                    StreamCapture = new Capture(1);

                    return 1;
                }
                catch (NullReferenceException excpt)
                {
                    MessageBox.Show(excpt.Message);
                    return 0;
                }

            }
            else
            {
                return 0;
            }


        }

        private Image<Bgr, Byte> fetchstreamimg()
        {

            Image<Bgr, Byte> frame = StreamCapture.QuerySmallFrame();//
StreamCapture.QueryFrame();
            return frame;

        }

        #endregion
        Image<Gray, byte> PLeye, PReye;

        Point leye = new Point(); Point leye_T = new Point(); Point leye_B = new Point();
Point leye_R = new Point(); Point leye_L = new Point();
        Point reye = new Point(); Point reye_T = new Point(); Point reye_B = new Point();
Point reye_R = new Point(); Point reye_L = new Point();
        Point noseC1 = new Point();
        Point Pleye = new Point();
        Point Preye = new Point();
```

```
        Point mouthC1 = new Point(); Point mouthCL = new Point(); Point mouthCR = new Point();
Point mouthCT = new Point(); Point mouthCB = new Point();
        Rectangle r_rec = new Rectangle(); Rectangle l_rec = new Rectangle();

        public PointF[][] actPoints;
        public PointF[] newPoints;
        public float[] TrackError;
        public Byte[] Status;
        Image<Bgr, byte> opflowFrame;

        private void facecapinit()
        {
            Image<Gray, byte> frame = fetchstreamimg().Convert<Gray, byte>();
            MCvAvgComp[][] facesDetected = frame.DetectHaarCascade(cascade_face, 1.1, 1,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(20, 20));

            foreach (MCvAvgComp f in facesDetected[0])
            {

              // frame.Draw(f.rect, new Gray(1.5), 2);

              #region Left Eye
              Rectangle Cur_rect_L = new Rectangle(f.rect.X, f.rect.Y, f.rect.Width * 3 / 5,
f.rect.Height);
              frame.ROI = Cur_rect_L;
              MCvAvgComp[][] leyeDetected = frame.DetectHaarCascade(eyel, 1.1, 1,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(5, 5));

              foreach (MCvAvgComp e in leyeDetected[0])
              {
                  {
                      Rectangle eyeRect = e.rect;
                      l_rec = eyeRect;
                      // frame.Draw(eyeRect, new Gray(0.2), 1);
                      Pleye = leye;
                      leye = new Point(e.rect.X + (e.rect.Width / 2), e.rect.Y +
(e.rect.Height / 2));
                      leye_B = new Point(e.rect.X + (e.rect.Width / 2), e.rect.Y +
(e.rect.Height));
                      leye_T = new Point(e.rect.X + (e.rect.Width/2), e.rect.Y);

                      leye_T.Offset(Cur_rect_L.Location);
                      leye_B.Offset(Cur_rect_L.Location);
                  }

              }
              frame.ROI = Rectangle.Empty;
              #endregion

              #region Right Eye
              Rectangle Cur_rect_R = new Rectangle(f.rect.X + (f.rect.Width * 3 / 5),
f.rect.Y, f.rect.Width * 3 / 5, f.rect.Height);
              frame.ROI = Cur_rect_R;
              MCvAvgComp[][] ReyeDetected = frame.DetectHaarCascade(eyer, 1.1, 1,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(5, 5));

              foreach (MCvAvgComp e in ReyeDetected[0])
              {
                  {
                      Rectangle eyeRect = e.rect;
                      r_rec = eyeRect;
```

```
                    // frame.Draw(eyeRect, new Gray(0.2), 1);
                    Preye = reye;
                    reye = new Point((e.rect.X + (e.rect.Width / 2)) +
Cur_rect_R.Location.X, (e.rect.Y + (e.rect.Height / 2)) + Cur_rect_R.Location.Y);
                    reye_B = new Point(e.rect.X + (e.rect.Width / 2), e.rect.Y +
(e.rect.Height));
                    reye_T = new Point(e.rect.X + (e.rect.Width / 2), e.rect.Y);

                    reye_T.Offset(Cur_rect_R.Location);
                    reye_B.Offset(Cur_rect_R.Location);
                }
            }
            frame.ROI = Rectangle.Empty;
            #endregion

            #region Mouth
                Rectangle Cur_rect_M = new Rectangle(f.rect.X + f.rect.X * 1 / 4, f.rect.Y +
f.rect.Height * 3 / 5, f.rect.Width * 2 / 4, f.rect.Height * 2 / 5);
                frame.ROI = Cur_rect_M;

                MCvAvgComp[][] MouthDetected = frame.DetectHaarCascade(mouth, 1.1, 1,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(5, 5));
                foreach (MCvAvgComp e in MouthDetected[0])
                {
                    Rectangle MouthRect = e.rect;
                   // frame.Draw(MouthRect, new Gray(0.2), 1);

                    mouthC1 = new Point((e.rect.X + (e.rect.Width / 2)) +
Cur_rect_M.Location.X, (e.rect.Y + (e.rect.Height / 2)) + Cur_rect_M.Location.Y);
                    mouthCL = new Point(e.rect.X + Cur_rect_M.Location.X, (e.rect.Y +
(e.rect.Height / 2)) + Cur_rect_M.Location.Y);
                    mouthCR = new Point(e.rect.X + Cur_rect_M.Location.X + e.rect.Width,
(e.rect.Y + (e.rect.Height / 2)) + Cur_rect_M.Location.Y);
                    mouthCT = new Point((e.rect.X + (e.rect.Width / 2)) +
Cur_rect_M.Location.X, e.rect.Y + Cur_rect_M.Location.Y);
                    mouthCB = new Point((e.rect.X + (e.rect.Width / 2)) +
Cur_rect_M.Location.X, e.rect.Y + Cur_rect_M.Location.Y + e.rect.Height);
                }
                frame.ROI = Rectangle.Empty;
                #endregion

            #region Nose
                frame.ROI = f.rect;
                MCvAvgComp[][] NoseDetected = frame.DetectHaarCascade(nose, 1.1, 1,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(5, 5));
                foreach (MCvAvgComp e in NoseDetected[0])
                {
                    Rectangle NoseRect = e.rect;
                    noseC1 = new Point(((e.rect.Width) / 2 + e.rect.X) + f.rect.X,
((e.rect.Height) / 2 + e.rect.Y) + f.rect.Y);
                   // frame.Draw(NoseRect, new Gray(0.2), 1);


                }
                frame.ROI = Rectangle.Empty;
                #endregion

            #region drawing
                LineSegment2DF l1 = new LineSegment2DF();
                leye.Offset(Cur_rect_L.Location);
                l1.P1 = leye; l1.P2 = reye;
```

```
                frame.Draw(l1, new Gray(0.2), 2);


                frame.Draw(new LineSegment2D(reye_B, reye_B), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(reye_T, reye_T), new Gray(0.2), 4);

                frame.Draw(new LineSegment2D(leye, leye), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(reye, reye), new Gray(0.2), 4);

                frame.Draw(new LineSegment2D(leye_B,leye_B), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(leye_T, leye_T), new Gray(0.2), 4);

                LineSegment2DF l2 = new LineSegment2DF();
                l2.P1 = mouthC1; l2.P2 = mouthC1; frame.ROI = Rectangle.Empty;
                frame.Draw(l2, new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(mouthCL, mouthCL), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(mouthCR, mouthCR), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(mouthCT, mouthCT), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(noseC1,noseC1), new Gray(0.2), 4);
                frame.Draw(new LineSegment2D(mouthCB, mouthCB), new Gray(0.2), 4);

              //lines
                frame.Draw(new LineSegment2D(leye_B, leye), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(leye_T, leye), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(reye_B, reye), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(reye_T, reye), new Gray(0.2), 2);

                frame.Draw(new LineSegment2D(mouthCT, mouthCL), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCR, mouthCT), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCR, mouthCB), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCL, mouthCB), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCR, mouthC1), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCL, mouthC1), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCT, mouthC1), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(mouthCB, mouthC1), new Gray(0.2), 2);

                frame.Draw(new LineSegment2D(mouthCT, noseC1), new Gray(0.2), 2);

                frame.Draw(new LineSegment2D(noseC1,leye), new Gray(0.2), 2);
                frame.Draw(new LineSegment2D(noseC1, reye), new Gray(0.2), 2);
              //
                #endregion

            }

            imageBox1.Image = frame;
        }

        private void opFlowCal() {
            Image<Bgr, byte> frame_bgr = fetchstreamimg();
            Image<Gray, byte> frame = frame_bgr.Convert<Gray,Byte>();

            Image<Bgr, byte> frame_new_bgr = fetchstreamimg();
            Image<Gray, byte> frame_new = frame_new_bgr.Convert<Gray, Byte>();

            //frame.FindCornerSubPix(actPoints, new System.Drawing.Size(10, 10), new
System.Drawing.Size(-1, -1), new MCvTermCriteria(30, 0.3d));
            OpticalFlow.PyrLK(frame, frame_new, actPoints[0], new System.Drawing.Size(10, 10),
1, new MCvTermCriteria(20, 0.03d), out newPoints, out Status, out TrackError);
            opflowFrame = new Image<Bgr, Byte>(frame.Width, frame.Height);
            opflowFrame = frame_new_bgr.Copy();
```

```
            for (int i = 0; i < actPoints[0].Length; i++)
                DrawFlowVectors(i);
            imageBox2.Image = opflowFrame;
            actPoints[0] = newPoints;
        }

        private void populatepoints() {


           actPoints = new PointF[][]{ new PointF[]
{leye,reye,leye_B,leye_L,leye_R,leye_T,reye_B,reye_L,reye_R,reye_T,noseC1,mouthC1,mouthCB,mout
hCL,mouthCR,mouthCT}};

        }
        private void handl(object sender, EventArgs e)
        {

                opFlowCal();
        }



        private void DrawFlowVectors(int i)
        {

            System.Drawing.Point p = new Point();
            System.Drawing.Point q = new Point();

            p.X = (int)actPoints[0][i].X;
            p.Y = (int)actPoints[0][i].Y;
            q.X = (int)newPoints[i].X;
            q.Y = (int)newPoints[i].Y;

            double angle;
            angle = Math.Atan2((double)p.Y - q.Y, (double)p.X - q.X);

            LineSegment2D line = new LineSegment2D(p, q);
            opflowFrame.Draw(line, new Bgr(255, 0, 0), 1);

            p.X = (int)(q.X + 6 * Math.Cos(angle + Math.PI / 4));
            p.Y = (int)(q.Y + 6 * Math.Sin(angle + Math.PI / 4));
            opflowFrame.Draw(new LineSegment2D(p, q), new Bgr(255, 0, 0), 1);
            p.X = (int)(q.X + 6 * Math.Cos(angle - Math.PI / 4));
            p.Y = (int)(q.Y + 6 * Math.Sin(angle - Math.PI / 4));
            opflowFrame.Draw(new LineSegment2D(p, q), new Bgr(255, 0, 0), 1);

        }
        int j = 0;

        private void button1_Click(object sender, EventArgs e)
        {

            facecapinit();
            populatepoints();
            j = 1;
        }

    }
}
```

3.8 Window1.xaml.cs (splashscreen)

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace ProjectCode
{
		public partial class Window1 : Window
	{
		public Window1()
		{
			this.InitializeComponent();

		}


private void launchparent(object sender, System.Windows.Input.MouseButtonEventArgs e)
	{

			MainWindow a = new MainWindow();
			a.Show();
			this.Close();
		}
	}
}
```

3.9 MainWindow.xaml.cs (Main/Option Window)

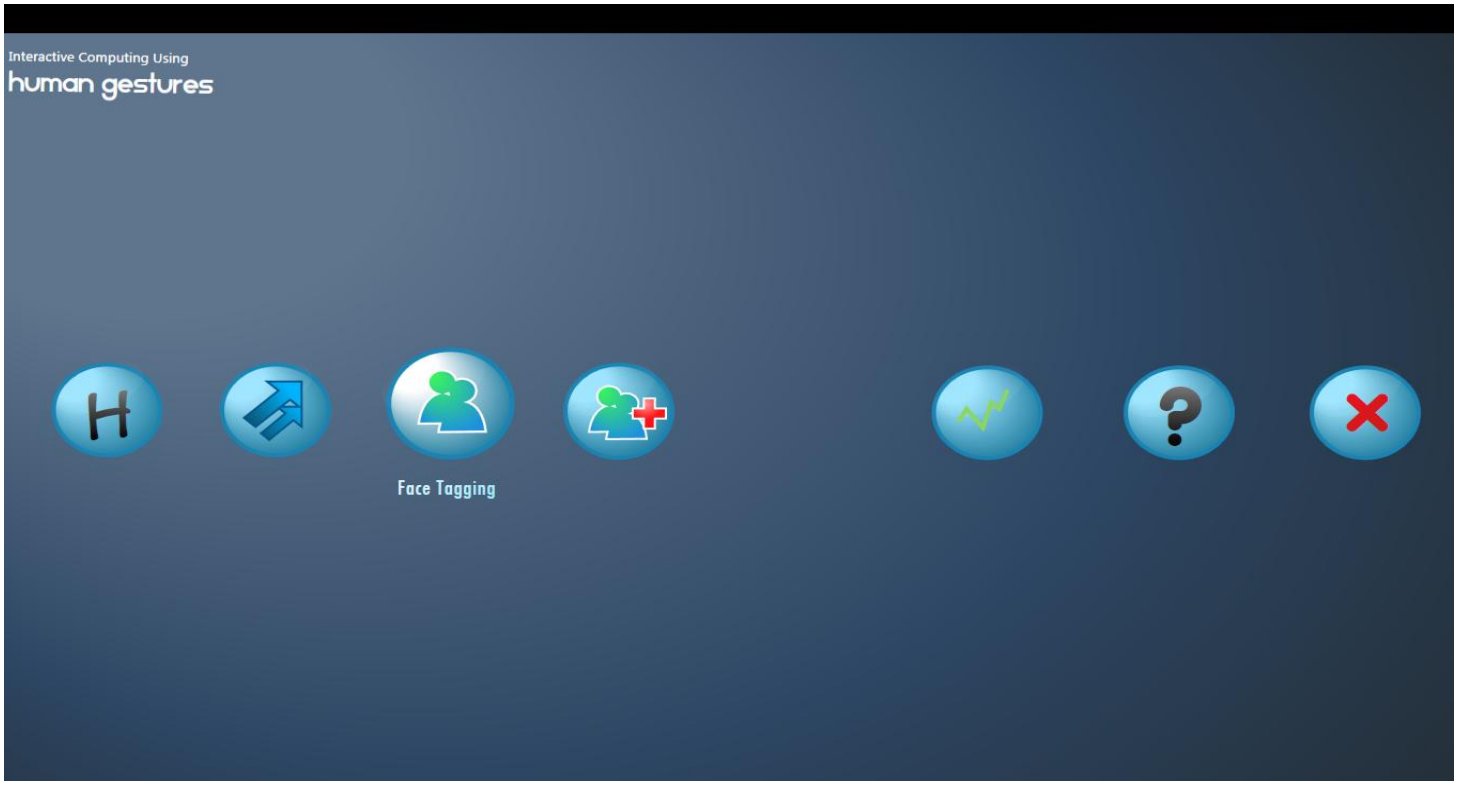
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace ProjectCode
{
	public partial class MainWindow : Window
	{
		public MainWindow()
		{
			this.InitializeComponent();
		}


		private void Exit_app(object sender, System.Windows.RoutedEventArgs e)
```

```
            {
                  if(MessageBox.Show(this,"Do You Really Want To Exit?","Exit
Application",MessageBoxButton.YesNo,MessageBoxImage.Exclamation) ==
System.Windows.MessageBoxResult.Yes){this.Close();}
            }

            private void abtbutton(object sender, System.Windows.RoutedEventArgs e)
            {
             About a = new About();
                  a.Show();
            }

            private void handtbutton(object sender, System.Windows.RoutedEventArgs e)
            {
              handtracking a = new handtracking ();
                  a.Show();
            }

            private void updatedbbutton(object sender, System.Windows.RoutedEventArgs e)
            {
              Update_DB a = new Update_DB ();
                  a.Show();
            }
            private void facetbutton(object sender, System.Windows.RoutedEventArgs e)
            {
              FaceTag a = new FaceTag ();
                  a.Show();
            }
      }
}
```

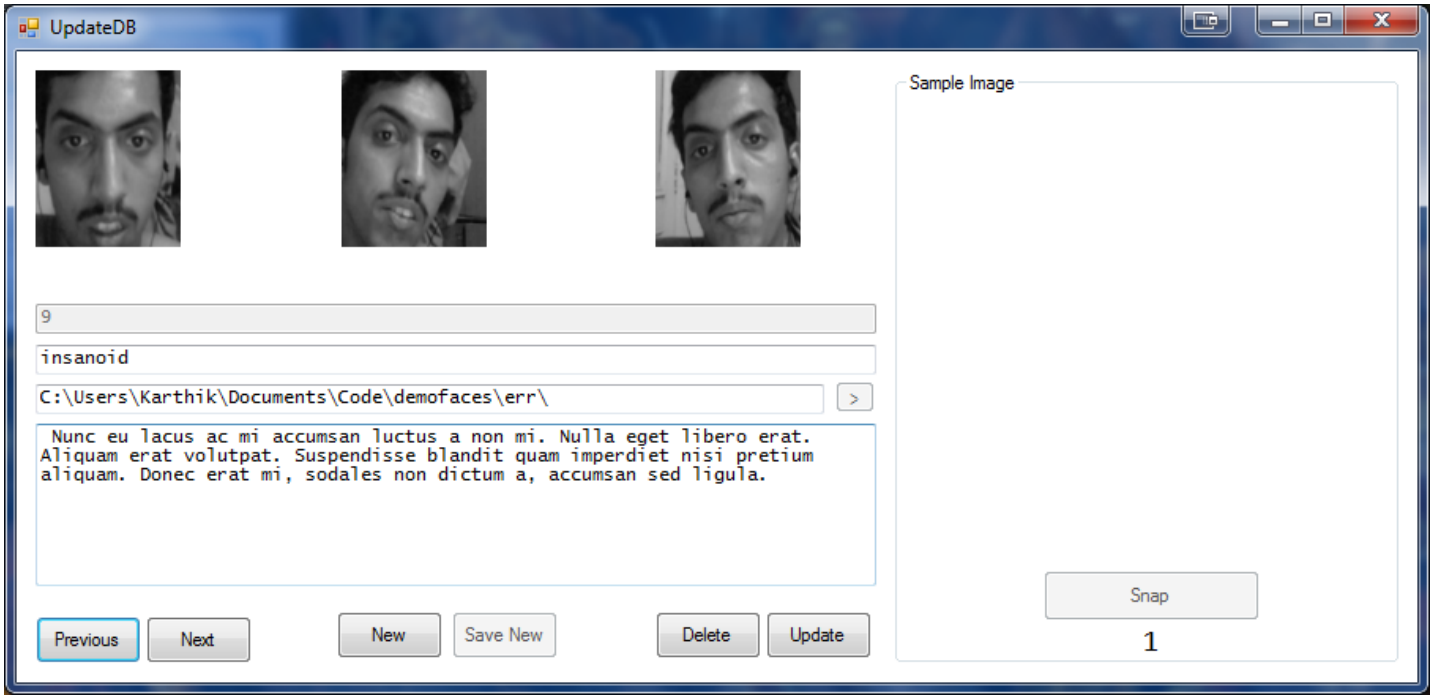
Results

4.1 Splash Screen (window1.xaml)

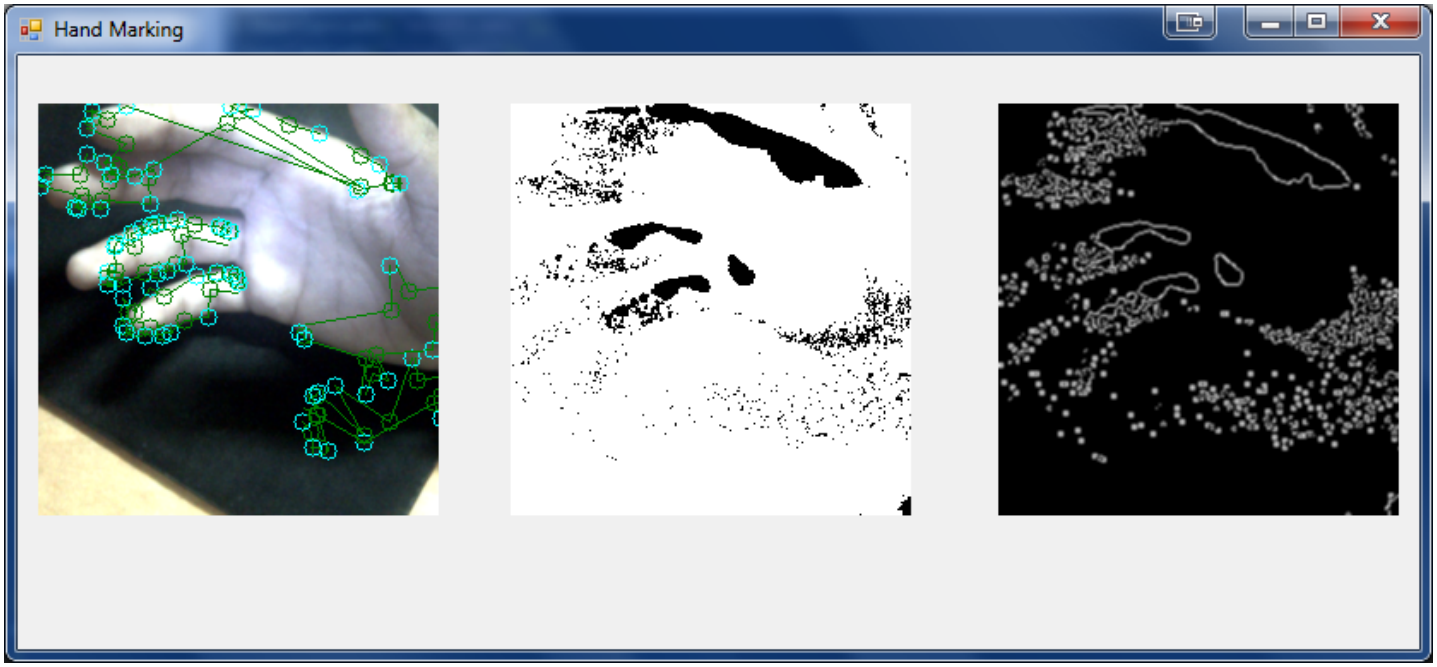
4.2 Main/Option Screen (MainWindow.xaml)



4.3 Tagged Faces(Update_DB.cs)

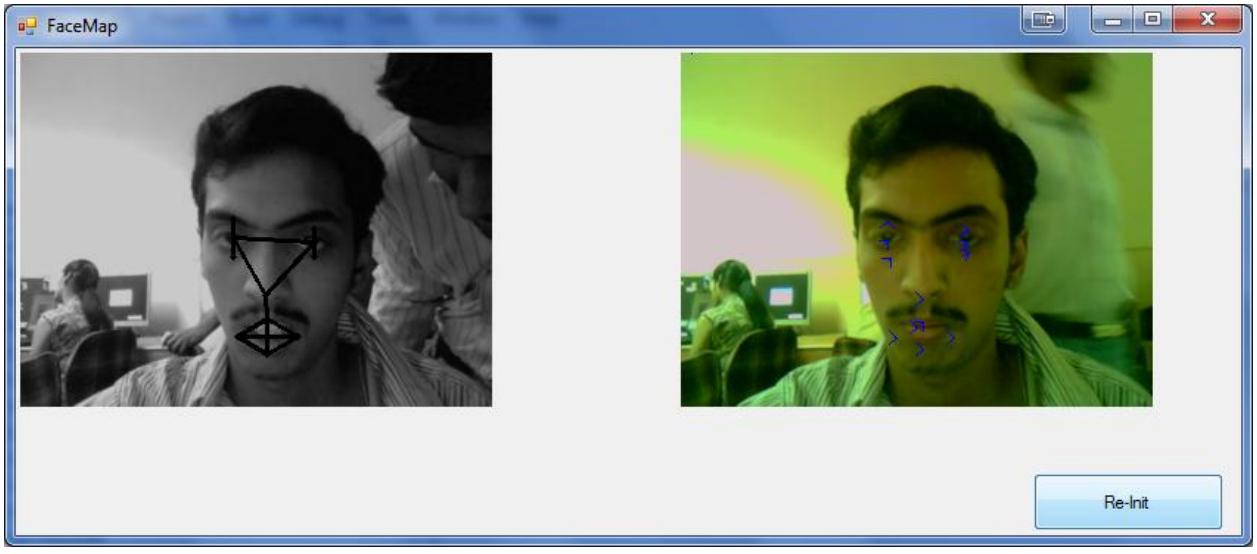


4.3 Hand Tracker (handtracking.cs)



4.4 Tag Faces (TagFaces.cs)

4.5 Facial Feature Extraction (FaceMap.cs)



4.6 About (About.xaml)

Testing

5.1 Brief Description of Testing

Testing can be categorized into following forms:

1. Unit testing
2. Module testing
3. System testing
4. Alpha/ Beta testing
5. White Box/ Black Box testing

UNIT TESTING:

The methodology of unit testing was followed in the manner that each and every piece of code was tested as soon as it was written. The individual components were tested to ensure that they operate correctly and independently. Let's have a closer look at this kind of pattern:
Each component is tested independently without the other system components. It is the first level of testing and the different modules are tested against the specifications. Unit testing is essential for verification of the code product during the coding phase, and hence the goal is to test the internal logic of the module. Unit testing focuses in the verification effort of the smallest unit of software design module.

MODULE TESTING:

A module is a collection of dependent components such as procedures and functions. A module encapsulates related components so that it can be tested without other system modules.
Module testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The program is constructed and testing in small segments where errors are easier to isolate and correct interfaces are more likely to be tested completely.

SYSTEM TESTING:

Collections of modules make a sub-system and collection of sub-systems makes the entire system. This phase involves testing of collection of sub-systems, which have been integrated into the entire system. Aim is to find out errors, which normally result from unanticipated interaction between sub-systems and components. It is also concerned with validating that the system meets its functional and non-functional requirements.
Generally, it begins with low volumes of transactions based on live data. System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. In system testing, performance and acceptance standards are developed.

ALPHA/BETA TESTING:

Alpha Testing- In this of testing the system is tested with the user provided data. Customer tests the system at developer site. Developer "looks over shoulder" and records errors & usage problems. Test is conducted in a controlled environment.

Beta Testing – This kind of testing involves delivering the system to a number of potential customers who give their feedback regarding the software. The system is then modified with respect to the feedback given. Repairing the system may introduce new bugs or errors in the system, thus testing has to be repeated after the system has been modified. Beta testing is conducted at one or more customer's site by the end user. Unlike in alpha testing, the developer is generally not present.

WHITE BOX & BLACK BOX TESTING:

White Box Testing is concerned with the implementation of the program. The idea is to test different programming and data structures used in the program. This test mainly focuses on the code rather than on the specifications. The test cases must be designed in such a way that it is able to locate most of the errors with minimum amount of time and efforts. White box testing is a design method that uses the control structure of the procedural design to derive test cases.
Here, in the project few proto were developed and tested thoroughly. All the modules were designed only using those protos.
Using the White Box method, we derive the test case that

- Guarantee that all independent paths within a module have been exercised at once.
- Exercise all logical decisions on the true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structure to ensure their validity.

Black Box Testing method focuses on the functional requirements of the software. A set of input conditions are derived that will fully exercise all functional requirements for a program. It is not an alternative to white box but is a complementary approach that uncovers a different class of errors than white box methods.
Black Box method attempts to find out errors relating to incorrect or missing functions, Interface errors, errors in data structures, performance errors, initialization and termination errors. White box testing is performed at earlier stages but black box testing is applied at the later stages of testing.

We also began with unit testing procedures. In this we tested individual forms developed to meet different requirements, to find and remove the errors. Following problems were encountered while making our project:

- Syntax error caused due to misspelling or missing clause.
- Unrecognized command verb, phrase and keywords.
- Data types mismatched.
- Expected errors.
- Connectivity errors and logic errors.

These errors were removed using following techniques:

- Monitoring the compilation of the program.
- By appropriate statements in the program.
- By applying the right logic.

Finally the project is run to test the connectivity and integrity of different options given in the menu to choices given by the user. This is to ensure that the system is responding well and giving best results as per user expectations and requirements

The testing for this project is done through unit testing only, no testing tools have been used since they weren't deemed necessary for testing.
The visual components were tested to correctly map the required identities as required and then used to extract them as needed, changes were made as required.

Conclusion

The aim of this project was to make the visual input more feasible and to be used in day to day basis and also to provide future developer a quick start in the interaction development.The non traditional input source can be used in a better way for the use of not only normal day to day use but also to the use of handicapped personal.

With the ability to identify people it also allows us to change preferences of people as per their needs without them having to do it so themselves.This can also be used to track the behaviour of the person, and can also be used in security systems.

The project has achived its initial goal of making an efficent input soruce and to allow the source to be reused and create various application on its basis.

Bibliography And Webliography

1. http://msdn.microsoft.com/
2. http://www.google.com/
3. http://opencv.willowgarage.com/
4. http://www.emgu.com/