

EEES403 - PROGRAMMING LANGUAGES PROJECT DETAILED REPORT

3 JANUARY 2021

C# ATTENDANCE LIST PROGRAM

Prepared by: Kamil DİNLEYİCİ

Student ID: 2016513019



Table of Contents

Introduction	3
The inspiration behind project.....	3
Starting to build program	4
Libraries used in program	4
Briefly introduction to main program	5
Main.....	7
BusinessRecognition.cs.....	7
Classifier_Train.cs.....	10
Login screen with details (LoginForm.cs)	15
Main screen with details (Form1.cs)	18
Building the Attendance List.....	23
Driver code (Program.cs)	24
Resources File	25
Conclusion	25
Directly accessing program through Google Drive link	26
References.....	26

Introduction

The inspiration behind project

The education technologies are rapidly growing up with Artificial Intelligence(AI) applications. These AI applications can be count as image processing, handwriting recognition, face recognition, virtual reality ect.

I was interested in face recognition algorithm and to develop this project I thought that building a embedded camera system for attendance in class would be useful for instructors. There will be a camera above the board and taking attendance would not be problem for instructor, the faces will be detected and labelled for each course. The instructor could see their students entrance dates after the course.



Figure 1.1. The desired photo in my imagination

Here is an example for represent my project. My idea was building a system such above photo and also the program needs to label each person before entering the course. Once you trained your face to the program, you would not need to train your faces again and again.

Starting to build program

I started with searching face detection algorithms or programs to implement my project because I was not a professional programmer to build a face recognition program from zero.

I found a program that was making face detection and recognition and dive into that program. I made some changes in algorithm to adapt my own program.

Libraries used in program

Before starting to implement our code we need to add some libraries which will be used in the main code. There are two libraries which are named as **Emgu.CV** and **OpenCV** to load these you have to follow these steps: “**Tools – NuGet Package Manager – Manage NuGet Packages for Solution...**”.

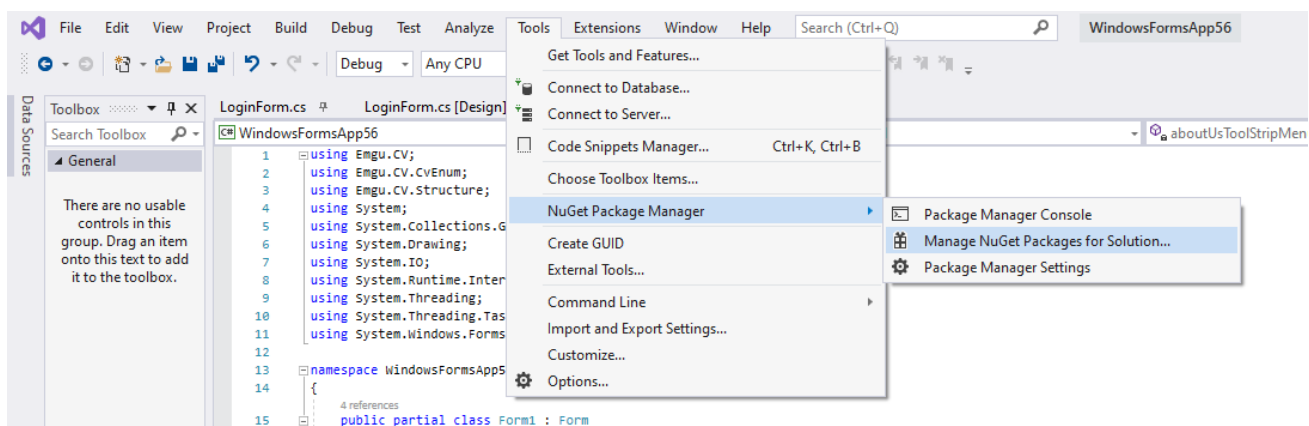


Figure 1.2. Access to NuGet packages

And then implement the Emgu.CV library to our code.

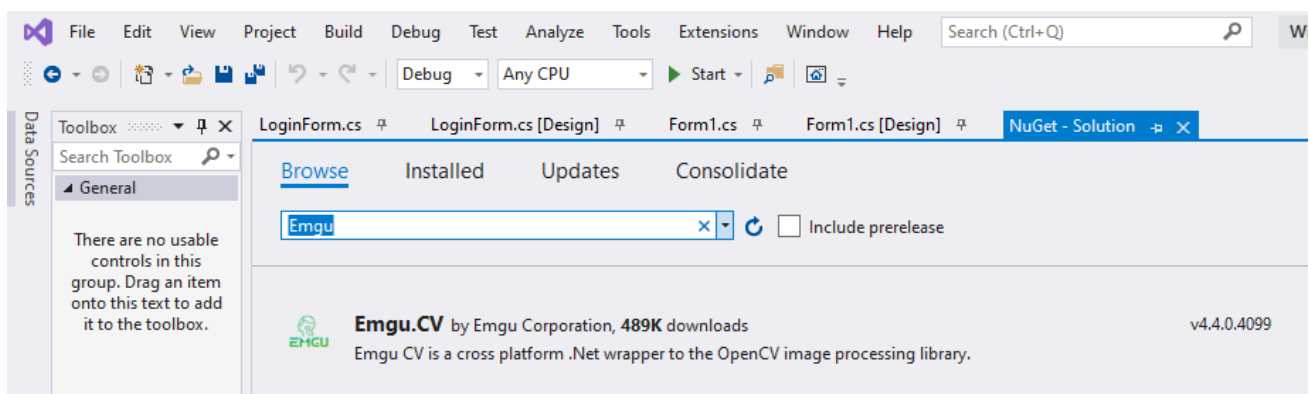


Figure 1.3. EmguCV implementation

As stated before we also need to add OpenCV library as shown below.

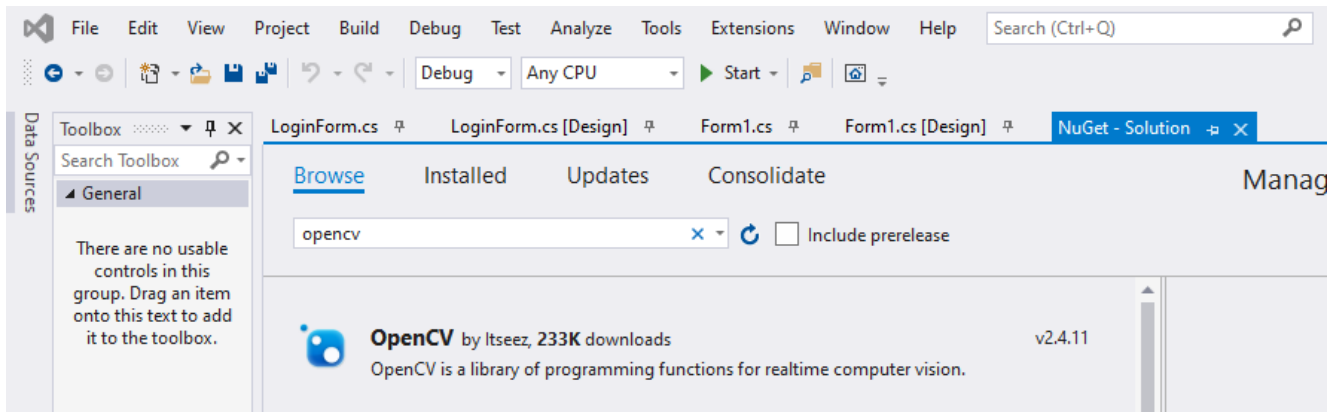


Figure 1.4. OpenCV implementation

Briefly introduction to main program

Emgu.CV library contains three types of face recognition algorithm. These are Eigenfaces, Fisherfaces and LBPH. In our code we use **Eigenfaces** algorithm, let's describe what is Eigenfaces algorithm. Eigenfaces takes the difference of the resulting image from the average by taking the average of all the pictures in the memory.

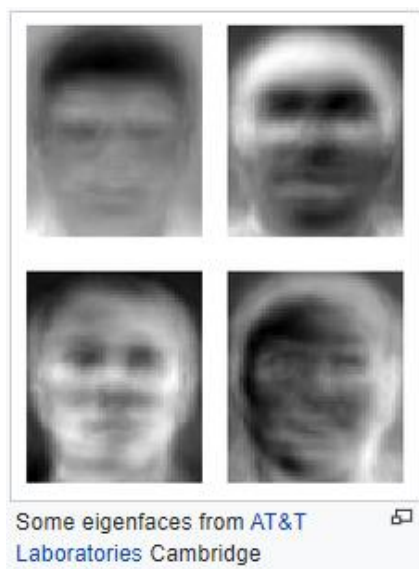


Figure 1.5. Some eigenfaces shown in process

Fisherfaces algorithm:

In order to find the combination of features that separates best between classes the Linear Discriminant Analysis maximizes the ratio of between-classes to within-classes scatter, instead of maximizing the overall scatter. The idea is simple: same classes should

cluster tightly together, while different classes are as far away as possible from each other in the lower-dimensional representation.

LBPH algorithm:

Think of things like scale, translation or rotation in images — your local description has to be at least a bit robust against those things. Just like SIFT, the Local Binary Patterns methodology has its roots in 2D texture analysis. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. You'll end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels you'll end up with 2^8 possible combinations, called *Local Binary Patterns* or sometimes referred to as *LBP codes*. The first LBP operator described in literature actually used a fixed 3 x 3 neighborhood just like this:

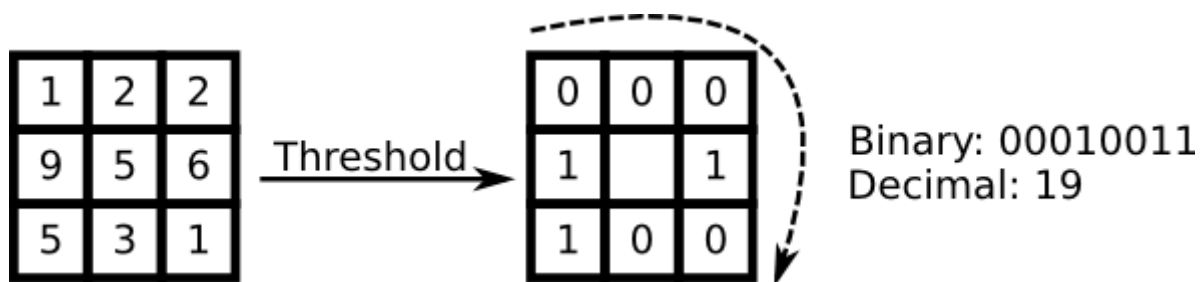


Figure 1.6. LBP operator

Face recognition and detection algorithms use two different class in the code. These two class are **BusinessRecognition.cs** and **Classifier_Train.cs**.

There is a **flow** for maintain the main program. Let us show the **steps**.

- **Training Faces**

The software's ability to take people's faces, store them in a certain memory in the background and organize these images in xml format is called "training".

Defined faces are assigned to the pictureBox2 object. These captured face images are sent to the SaveTrainingData method (in **BusinessRecognition.cs**) and the program is specially trained for the faces.

- **Comparing the trained face to the normal face**

At this stage, the Recognize method of the **Classifier_Train.cs** class will get involved. This method compares the sent image object with the memory and returns the information of the matching record.

- **Labelling**

Now it is time to work in the load event of the form. Here, in addition to camera image capture, face identification and emphasis, triggers are made to activate face recognition algorithms.

Main

Firstly, I will share my two important classes which are BusinessRecognition.cs and Classifier_Train.cs codes. Then, I will show my login screen and main code part (Form1). At the end of the main part, I will show you how I created the attendance list part.

BusinessRecognition.cs

We will use BusinessRecognition class to mainly use SaveTrainingData method.

```
using Emgu.CV;
using Emgu.CV.Structure;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml;

namespace WindowsFormsApp56
{
    class BusinessRecognition
    {
        string Dizin;
        string KlasorAdi;
        string XmlVeriDosyasi;
        public BusinessRecognition()
        {
            KlasorAdi = "TrainedFaces";
            XmlVeriDosyasi = "TrainedLabels.xml";
            Dizin = Application.StartupPath + "/" + KlasorAdi + "/";
        }

        public BusinessRecognition(string Dizin, string KlasorAdi)
        {
            this.Dizin = Dizin + "/" + KlasorAdi + "/";
            Eigen_Recog = new Classifier_Train(Dizin, KlasorAdi);
        }
    }
}
```

```

public BusinessRecognition(string Dizin, string KlasorAdi, string XmlVeriDosyasi)
{
    this.Dizin = Dizin + "/" + KlasorAdi + "/";
    this.XmlVeriDosyasi = XmlVeriDosyasi;
    Eigen_Recog = new Classifier_Train(Dizin, KlasorAdi, XmlVeriDosyasi);
}

#region To capture 50 photos
List<Image<Gray, byte>> resultImages = new List<Image<Gray, byte>>();

#endregion

#region Classifier
Classifier_Train Eigen_Recog;
#endregion

#region XML Data sets
XmlDocument docu = new XmlDocument();
#endregion

#region SaveTrainingData method
public bool SaveTrainingData(Image face_data, string FaceName)
{
    try
    {
        string NAME_PERSON = FaceName;
        Random rand = new Random();
        bool file_create = true;
        string facename = "face_" + NAME_PERSON + "_" + rand.Next().ToString() + ".jpg";
        while (file_create)
        {
            if (!File.Exists(Dizin + facename))
            {
                file_create = false;
            }
            else
            {
                facename = "face_" + NAME_PERSON + "_" + rand.Next().ToString() +
".jpg";
            }
        }

        if (Directory.Exists(Dizin))
        {
            face_data.Save(Dizin + facename, ImageFormat.Jpeg);
        }
        else
        {
            Directory.CreateDirectory(Dizin);
            face_data.Save(Dizin + facename, ImageFormat.Jpeg);
        }
        if (File.Exists(Dizin + XmlVeriDosyasi))
        {
            //File.AppendAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", NAME_PERSON.Text + "\n\r");
            bool loading = true;
            while (loading)

```



```

{
    try
    {
        docu.Load(Dizin + XmlVeriDosyasi);
        loading = false;
    }
    catch
    {
        docu = null;
        docu = new XmlDocument();
        Thread.Sleep(10);
    }
}

//Get the root element
XmlElement root = docu.DocumentElement;

XmlElement face_D = docu.CreateElement("FACE");
XmlElement name_D = docu.CreateElement("NAME");
XmlElement file_D = docu.CreateElement("FILE");

//Add the values for each nodes
//name.Value = textBoxName.Text;
//age.InnerText = textBoxAge.Text;
//gender.InnerText = textBoxGender.Text;
name_D.InnerText = NAME_PERSON;
file_D.InnerText = facename;

//Construct the Person element
//person.Attributes.Append(name);
face_D.AppendChild(name_D);
face_D.AppendChild(file_D);

//Add the New person element to the end of the root element
root.AppendChild(face_D);

//Save the document
docu.Save(Dizin + XmlVeriDosyasi);
//XmlElement child_element = docu.CreateElement("FACE");
//docu.AppendChild(child_element);
//docu.Save("TrainedLabels.xml");
}
else
{
    FileStream FS_Face = File.OpenWrite(Dizin + XmlVeriDosyasi);
    using (XmlWriter writer = XmlWriter.Create(FS_Face))
    {
        writer.WriteStartDocument();
        writer.WriteStartElement("Faces_For_Training");

        writer.WriteStartElement("FACE");
        writer.WriteElementString("NAME", NAME_PERSON);
        writer.WriteElementString("FILE", facename);
        writer.WriteEndElement();

        writer.WriteEndElement();
        writer.WriteEndDocument();
    }
    FS_Face.Close();
}

return true;
}
catch (Exception ex)

```

```

        {
            return false;
        }
    }
}
#endregion
}
}

```

Classifier_Train.cs

```

using Emgu.CV;
using Emgu.CV.Structure;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Serialization;

namespace WindowsFormsApp56
{
    class Classifier_Train : IDisposable
    {
        /// <summary>
        /// Örnek Kullanım,
        /// D:\, D:\Klasor\
        /// </summary>
        string Dizin;
        string KlasorAdi;
        string XmlVeriDosyasi;
        public Classifier_Train(string Dizin, string KlasorAdi)
        {
            this.Dizin = Dizin + KlasorAdi;

            termCrit = new MCvTermCriteria(ContTrain, 0.001);
            _IsTrained = LoadTrainingData(this.Dizin);
        }
        public Classifier_Train(string Dizin, string KlasorAdi, string XmlVeriDosyasi)
        {
            this.Dizin = Dizin + KlasorAdi;
            this.XmlVeriDosyasi = XmlVeriDosyasi;

            termCrit = new MCvTermCriteria(ContTrain, 0.001);
            _IsTrained = LoadTrainingData(this.Dizin);
        }

        #region Variables

        //Eigen
        MCvTermCriteria termCrit;
        EigenObjectRecognizer recognizer;

        //training variables
        List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>(); //Images
        List<string> Names_List = new List<string>(); //labels

```

```

int ContTrain, NumLabels;
float Eigen_Distance = 0;
string Eigen_label;
int Eigen_threshold = 0;

//Class Variables
string Error;
bool _IsTrained = false;

#endregion
public List<string> isimler()
{
    return Names_List;
}

#region Constructors
/// <summary>
/// Default Constructor, Looks in (Dizin) for traing data.
/// </summary>
public Classifier_Train()
{
    KlasorAdi = "TrainedFaces";
    Dizin = Application.StartupPath + "\\\" + KlasorAdi;
    XmlVeriDosyasi = "TrainedLabels.xml";

    termCrit = new MCvTermCriteria(ContTrain, 0.001);
    _IsTrained = LoadTrainingData(Dizin);
}

/// <summary>
/// Takes String input to a different location for training data
/// </summary>
/// <param name="Training_Folder"></param>
public Classifier_Train(string Training_Folder)
{
    termCrit = new MCvTermCriteria(ContTrain, 0.001);
    _IsTrained = LoadTrainingData(Training_Folder);
}
#endregion

#region Public
/// <summary>
/// <para>Return(True): If Training data has been located and Eigen Recogniser has been
trained</para>
/// <para>Return(False): If NO Training data has been located of error in training has
occured</para>
/// </summary>
public bool IsTrained
{
    get { return _IsTrained; }
}

/// <summary>
/// Recognise a Grayscale Image using the trained Eigen Recogniser
/// </summary>
/// <param name="Input_image"></param>
/// <returns></returns>
public string Recognise(Image<Gray, byte> Input_image, int Eigen_Thresh = -1)
{
    if (_IsTrained)
    {
        EigenObjectRecognizer.RecognitionResult ER = recognizer.Recognize(Input_image);
        //handle if recognizer.EigenDistanceThreshold is set as a null will be returned
        //NOTE: This is still not working correctley
    }
}

```

```

        if (ER == null)
        {
            Eigen_label = "Undefined";
            Eigen_Distance = 0;
            return Eigen_label;
        }
        else
        {
            Eigen_label = ER.Label;
            Eigen_Distance = ER.Distance;
            if (Eigen_Thresh > -1) Eigen_threshold = Eigen_Thresh;
            if (Eigen_Distance > Eigen_threshold) return Eigen_label;
            else return "Undefined";
        }
    }
    else return "";
}

/// <summary>
/// Sets the threshold confidence value for string Recognise(Image<Gray, byte>
Input_image) to be used.
/// </summary>
public int Set_Eigen_Threshold
{
    set
    {
        //NOTE: This is still not working correctley
        //recognizer.EigenDistanceThreshold = value;
        Eigen_threshold = value;
    }
}

/// <summary>
/// Returns a string containg the recognised persons name
/// </summary>
public string Get_Eigen_Label
{
    get
    {
        return Eigen_label;
    }
}

/// <summary>
/// Returns a float confidence value for potential false clasifications
/// </summary>
public float Get_Eigen_Distance
{
    get
    {
        //get eigenDistance
        return Eigen_Distance;
    }
}

/// <summary>
/// Returns a string contatining any error that has occured
/// </summary>
public string Get_Error
{
    get { return Error; }
}

```

```

    /// <summary>
    /// Saves the trained Eigen Recogniser to specified location
    /// </summary>
    /// <param name="filename"></param>
    public void Save_Eigen_Recogniser(string filename)
    {
        StringBuilder sb = new StringBuilder();

        (new XmlSerializer(typeof(EigenObjectRecognizer))).Serialize(new StringWriter(sb),
recognizer);
        XmlDocument xDoc = new XmlDocument();
        xDoc.LoadXml(sb.ToString());
        xDoc.Save(filename);
    }

    /// <summary>
    /// Loads the trained Eigen Recogniser from specified location
    /// </summary>
    /// <param name="filename"></param>
    public void Load_Eigen_Recogniser(string filename)
    {
        //introduce error checking
        FileStream EigenFS = File.OpenRead(filename);
        long Eflength = EigenFS.Length;
        byte[] xmlEBs = new byte[Eflength];
        EigenFS.Read(xmlEBs, 0, (int)Eflength);
        EigenFS.Close();

        MemoryStream xStream = new MemoryStream(xmlEBs);
        recognizer = (EigenObjectRecognizer)(new
XmlSerializer(typeof(EigenObjectRecognizer))).Deserialize(xStream);
        _IsTrained = true;
        //_eigenImages[Array_location] = (Image<Gray, Single>)(new
XmlSerializer(typeof(Image<Gray, Single>))).Deserialize(xStream);
    }

    /// <summary>
    /// Dispose of Class call Garbage Collector
    /// </summary>
    public void Dispose()
    {
        recognizer = null;
        trainingImages = null;
        Names_List = null;
        Error = null;
        GC.Collect();
    }

#endregion

#region Private
    /// <summary>
    /// Loads the traing data given a (string) folder location
    /// </summary>
    /// <param name="Folder_location"></param>
    /// <returns></returns>
    private bool LoadTrainingData(string Folder_location)
    {
        if (File.Exists(Folder_location + "\\\" + XmlVeriDosyasi))
        {
            try
            {
                //message_bar.Text = "";
                Names_List.Clear();
            }
            catch { }
        }
    }

```

```

trainingImages.Clear();
FileStream filestream = File.OpenRead(Folder_location + "\\\" +
XmlVeriDosyasi);
    long filelength = filestream.Length;
    byte[] xmlBytes = new byte[filelength];
    filestream.Read(xmlBytes, 0, (int)filelength);
    filestream.Close();

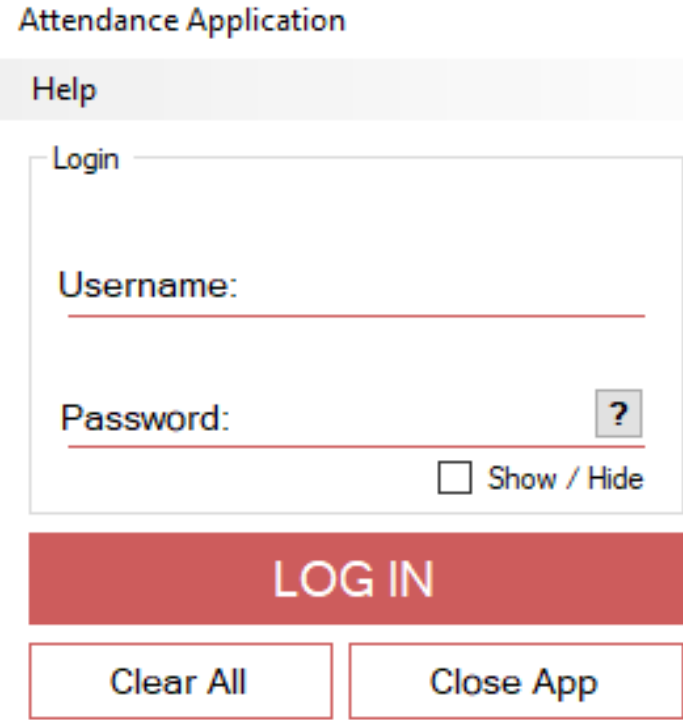
    MemoryStream xmlStream = new MemoryStream(xmlBytes);

    using (XmlReader xmlreader = XmlTextReader.Create(xmlStream))
    {
        while (xmlreader.Read())
        {
            if (xmlreader.IsStartElement())
            {
                switch (xmlreader.Name)
                {
                    case "NAME":
                        if (xmlreader.Read())
                        {
                            Names_List.Add(xmlreader.Value.Trim());
                            NumLabels += 1;
                        }
                        break;
                    case "FILE":
                        if (xmlreader.Read())
                        {
                            //PROBLEM HERE IF TRAININGG MOVED
                            trainingImages.Add(new Image<Gray, byte>(Dizin +
"\\\" + xmlreader.Value.Trim()));
                        }
                        break;
                }
            }
        }
    }
    ContTrain = NumLabels;

    if (trainingImages.ToArray().Length != 0)
    {
        //Eigen face recognizer
        recognizer = new EigenObjectRecognizer(trainingImages.ToArray(),
        Names_List.ToArray(), 5000, ref termCrit); //5000 default
        return true;
    }
    else return false;
}
catch (Exception ex)
{
    Error = ex.ToString();
    return false;
}
}
else return false;
}
}
#endregion
}

```


Login screen with details (LoginForm.cs)

	What was used in this screen? <ul style="list-style-type: none">1 Group box2 Text boxes2 Labels1 Menu Strip4 Buttons1 Check box1 Progress bar1 Timer
--	--

The login screen of the program is given at the above.

- There are two text boxes to enter “Username” and “Password”.
- One hint button is added to see if there is a hint for password.
- “Login” button triggers the timer and timer wakes up progress bar. If the username and password match as “admin” for username and “123” for password, you will enter the main program.

Related code for login screen is shared at the below also you can see the descriptions for given code parts with the green comment lines.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp56
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }
        // Exit button
        private void button2_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        // "Clear All" button
        private void button3_Click(object sender, EventArgs e)
        {
            textBox1.Clear();
            textBox2.Clear();
            label3.Visible = false;
        }
        // "Log In" button
        private void button1_Click(object sender, EventArgs e)
        {
            timer1.Enabled = true;
        }
        // When you press "Log In" button, the timer1 will be triggered
        private void timer1_Tick(object sender, EventArgs e)
        {
            progressBar1.Visible = true;
            progressBar1.Value = progressBar1.Value + 5;
            label3.Visible = true;
            label3.Text = "Logging in...";
            if(progressBar1.Value == progressBar1.Maximum)
            {
                if ((textBox1.Text == "admin") && (textBox2.Text == "123") )
                {
                    timer1.Enabled = false;
                    progressBar1.Visible = false;
                    progressBar1.Enabled = false;
                    progressBar1.Value = 0;
                    Form1 frm1 = new Form1();
                    this.Hide();
                    frm1.ShowDialog();
                }
                else if (textBox2.Text.Contains(textBox1.Text))
                {
                    progressBar1.Enabled = false;
                    timer1.Enabled = false;
                    progressBar1.Visible = false;
                    progressBar1.Value = 0;
                    label3.Text = "Password can not contain username!";
                    textBox1.Clear();
                    textBox2.Clear();
                }
            }
            else
            {
                progressBar1.Enabled = false;
                timer1.Enabled = false;
                progressBar1.Visible = false;
            }
        }
    }
}

```

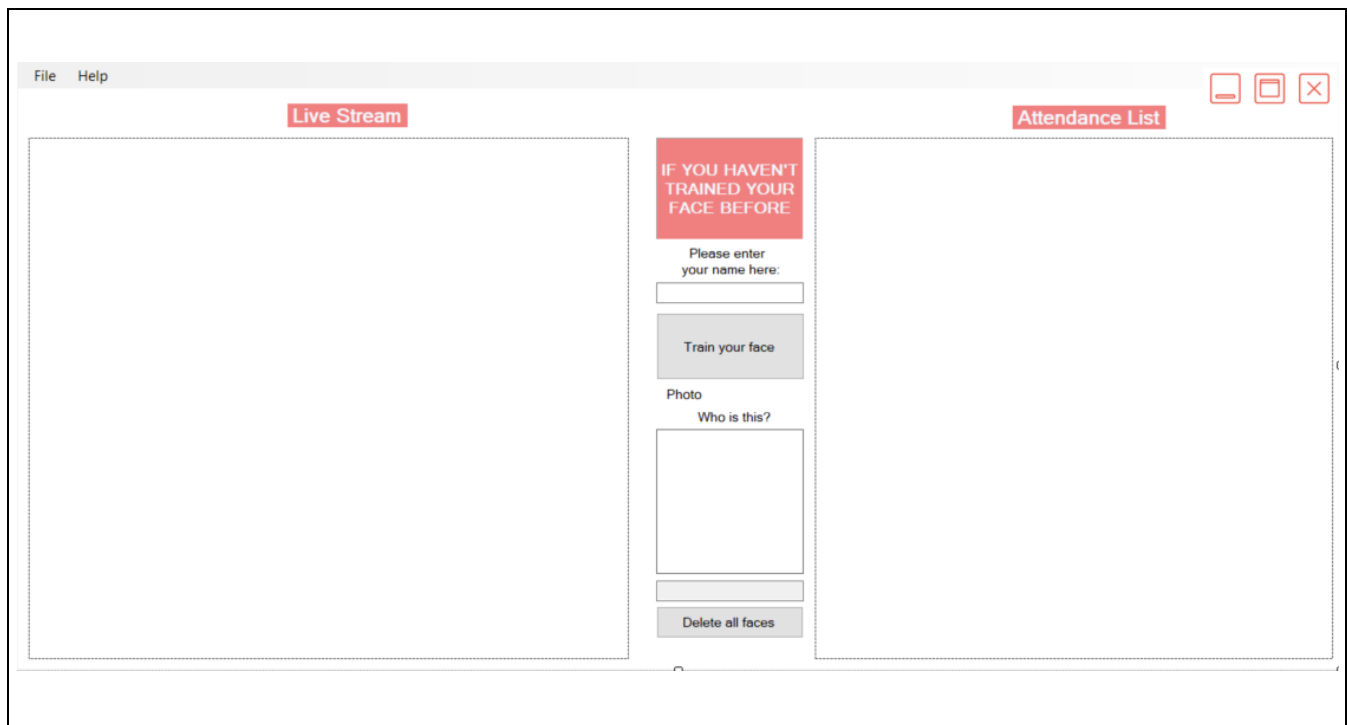
```

        progressBar1.Value = 0;
        label3.Text = "Try again!";
        textBox1.Clear();
        textBox2.Clear();
    }
}
// Help menu
private void aboutUsToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Prepared by Kamil Dinleyici,\n ID:2016513019 \nÇukurova
University EEMB");
}
// Check box for show/hide password
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        textBox2.UseSystemPasswordChar = false;
    }
    else
    {
        textBox2.UseSystemPasswordChar = true;
    }
}
// When you press Enter button at password part, "Log In" button will perform
private void textBox2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        button1.PerformClick();
        e.SuppressKeyPress = true;
    }
}
// The help button next to password
private void button4_MouseHover(object sender, EventArgs e)
{
    lblHelp.Visible = true;
}
// The help button next to password
private void button4_MouseLeave(object sender, EventArgs e)
{
    lblHelp.Visible = false;
}

private void progressBar1_Click(object sender, EventArgs e)
{
}
}
}

```

Main screen with details (Form1.cs)



What was used in this screen?

- 2 Picture boxes
- 2 Text box
- 6 Labels
- 1 Menu Strip
- 2 Buttons
- 1 Check box
- 1 Progress bars
- 1 FlowLayoutPanel

When you passed the login screen, you will see this main code screen.

- At the left hand side there is one picture box that captures video from your webcam as stated below the “Live Stream” label.
- You have write your name if you have never writed your name in text box. Once you write your name and train your face you don’t have to do all this process again.

- Your face will take place at the picture box as labelled with your name. For example trained faces shown like this;

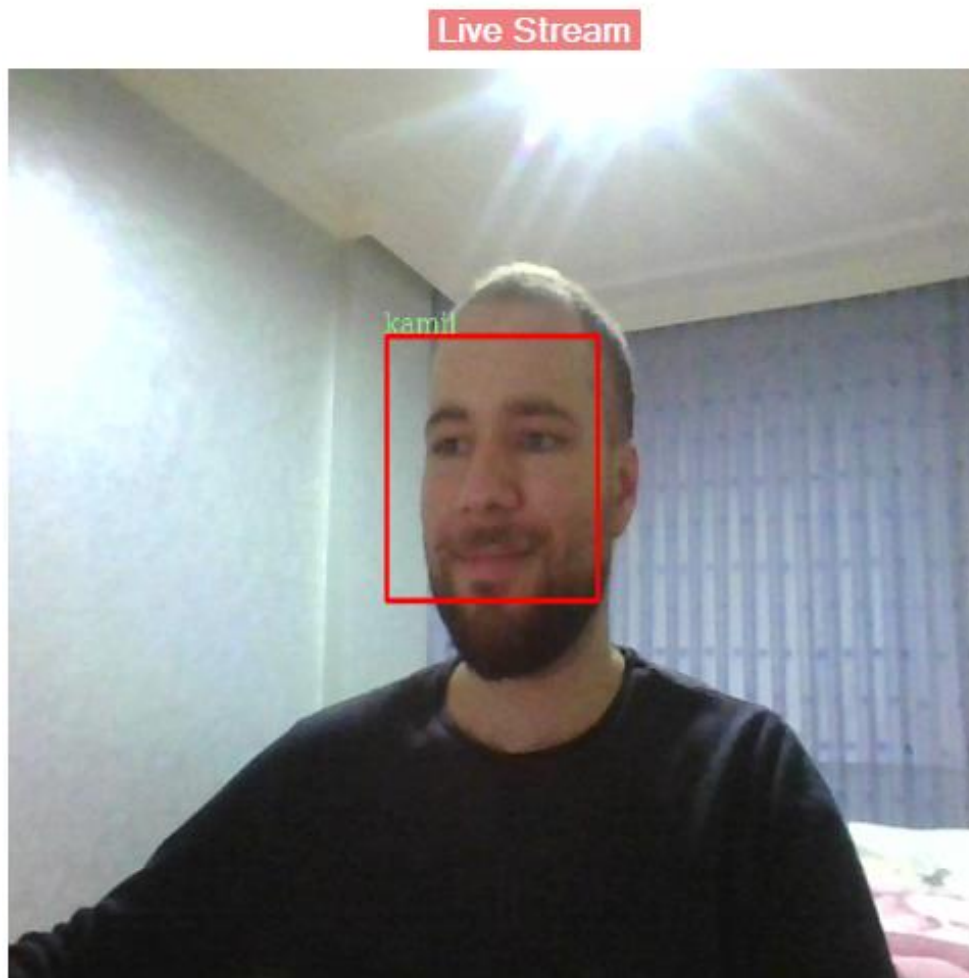


Figure 2.1. Picture box shows the labelled names

- After your face appear in the left side picture box, your name and some instructions transfer to the attendance list with a small photo. Here is an example;

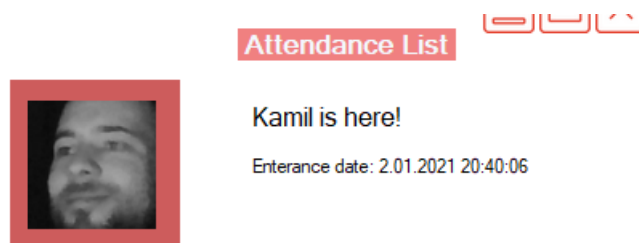


Figure 2.2. Attendance list

- If you press “Delete all trained faces” button, the whole faces added to file will be disappear and you have to train your face again when you restart your program.

```

using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Runtime.InteropServices;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp56
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            Control.CheckForIllegalCrossThreadCalls = false;

            InitializeComponent();
        }
        //These are for moving the program
        public const int WM_NCLBUTTONDOWN = 0xA1;
        public const int HT_CAPTION = 0x2;

        [DllImportAttribute("user32.dll")]
        public static extern int SendMessage(IntPtr hWnd,
            int Msg, int wParam, int lParam);
        [DllImportAttribute("user32.dll")]
        public static extern bool ReleaseCapture();

        // "Train your face" button's clicked part
        private async void btnEgit_Click(object sender, EventArgs e)
        {
            await Task.Run(() =>
            {
                for (int i = 0; i < 50; i++)
                {
                    if (!recognition.SaveTrainingData(pictureBox2.Image, txtFaceName.Text))
                        MessageBox.Show("Error", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    Thread.Sleep(100);
                    lblEgitilenAdet.Text = (i + 1) + " times photo taken.";
                }

                recognition = null;
                train = null;

                recognition = new BusinessRecognition("D:\\", "Faces", "yuz.xml");
                train = new Classifier_Train("D:\\", "Faces", "yuz.xml");
            });

            BusinessRecognition recognition = new BusinessRecognition("D:\\", "Faces", "yuz.xml");
            Classifier_Train train = new Classifier_Train("D:\\", "Faces", "yuz.xml");

            private void Form1_Load(object sender, EventArgs e)
            {
                webcamProcess();
            }
            //The pictureBox1 runs the webcam process
            private void webcamProcess()
            {
                List<string> nameList = new List<string>();
                Capture capture = new Capture();
            }
        }
    }
}

```



```

capture.Start();
capture.ImageGrabbed += (a, b) =>
{
    var image = capture.RetrieveBgrFrame();
    var grayimage = image.Convert<Gray, byte>();
    HaarCascade haaryuz = new HaarCascade("haarcascade_frontalface_alt2.xml");
    MCvAvgComp[][] Yuzler = grayimage.DetectHaarCascade(haaryuz, 1.2, 5,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING, new Size(15, 15));
    MCvFont font = new MCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 0.5, 0.5);
    foreach (MCvAvgComp yuz in Yuzler[0])
    {
        var sadeyuz = grayimage.Copy(yuz.rect).Convert<Gray, byte>().Resize(100,
100, INTER.CV_INTER_CUBIC);
        pictureBox2.Image = sadeyuz.ToBitmap();
        if (train != null)
            if (train.IsTrained)
            {
                string name = train.Recognise(sadeyuz);
                if (!nameList.Contains(name) && name != "Undefined")
                {
                    this.Invoke((MethodInvoker)delegate ()
                    {
                        //Make the name uppercase
                        string newName = char.ToUpper(name[0]) + name.Substring(1);
                        //Calls the method to transfer data to attendance list
                        populateItems(newName, pictureBox2.Image);
                    });
                }
                nameList.Add(name);
                txtWho.Text = name;
                int match_value = (int)train.Get_Eigen_Distance;
                image.Draw(name + " ", ref font, new Point(yuz.rect.X - 2,
yuz.rect.Y - 2), new Bgr(Color.LightGreen));
            }
            image.Draw(yuz.rect, new Bgr(Color.Red), 2);
        }
        pictureBox1.Image = image.ToBitmap();
    };
}
//This function for adding new items to the right hand side that is attendance list / It
takes the name and pictureBox2 image to add attendance list
private void populateItems(string TitleName, Image IconName)
{
    ListItem[] listItems = new ListItem[1];
    for (int i = 0; i < listItems.Length; i++)
    {
        listItems[i] = new ListItem();
        listItems[i].Title = TitleName + " is here!";
        listItems[i].Icon = IconName;
        listItems[i].Message = "Entrance date: " + Convert.ToString(DateTime.Now);
        flowLayoutPanel1.Controls.Add(listItems[i]);
        WriteLogin(TitleName);
    }
}
//This function creates a file which is named as "Login.txt" in main file and records
the logins from attendance list
private void WriteLogin(string TitleName)
{
    StreamWriter sw = new StreamWriter("../Login.txt", true);
    sw.WriteLine(TitleName + " has entered the class at " +
Convert.ToString(DateTime.Now));
    sw.Close();
}
private void label1_Click(object sender, EventArgs e)

```

```

{
}
private void pictureBox1_Click(object sender, EventArgs e)
{
}
private void txtFaceName_TextChanged(object sender, EventArgs e)
{
}
//To minimize the window - button
private void button1_Click(object sender, EventArgs e)
{
    WindowState = FormWindowState.Minimized;
}
//To minimize and maximize window - button
private void button2_Click(object sender, EventArgs e)
{
    if (WindowState == FormWindowState.Normal)
    {
        WindowState = FormWindowState.Maximized;
    }
    else if (WindowState == FormWindowState.Maximized)
    {
        WindowState = FormWindowState.Normal;
    }
}
//Exit button
private void button3_Click(object sender, EventArgs e)
{
    Application.Exit();
}
//To move the application
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        ReleaseCapture();
        SendMessage(Handle, WM_NCLBUTTONDOWN, HT_CAPTION, 0);
    }
}

private void label5_Click(object sender, EventArgs e)
{
}
//Help us part in menu strip
private void aboutUsToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Prepared by Kamil Dinleyici,\n
University EEMB");
}
//Restart application button in menu strip
private void restartToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Restart();
    Environment.Exit(0);
}
//Delete button to delete all trained faces
private void btnDelete_Click(object sender, EventArgs e)
{
    DirectoryInfo di = new DirectoryInfo(@"D:\Faces");
    foreach (FileInfo file in di.GetFiles())

```

```

        {
            file.Delete();
        }
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
}
}

```

Building the Attendance List

I created a User Control in the program as named ListItem.cs . This user control contains the detected face's name, entrance date and a small photo. The code for that User Control is given at the below.

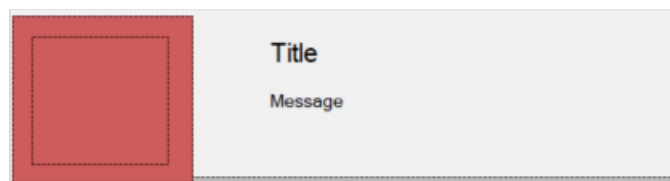


Figure 2.3. User Control box

```

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

namespace WindowsFormsApp56
{
    public partial class ListItem : UserControl
    {
        public ListItem()
        {
            InitializeComponent();
        }

        #region Properties
        private string _title;
        private string _message;
        private Image _icon;
        [Category("Custom Props")]
        public string Title
        {
            get { return _title; }
        }
    }
}

```

```

        set { _title = value; lblTitle.Text = value; }
    }
    [Category("Custom Props")]
    public string Message
    {
        get { return _message; }
        set { _message = value; lblMessage.Text = value; }
    }
    [Category("Custom Props")]
    public Image Icon
    {
        get { return _icon; }
        set { _icon = value; pictureBox1.Image = value; }
    }
}
#endregion

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

```

When the algorithm recognize the face populateltItems method in the main code will be executed and prepared User Control box will be added to the right hand sided FlowLayoutPanel. This FlowLayoutPanel is shown under the label of “Attendance List”. You can see the User Control added FlowLayoutPanel version at Figure 2.2.

Driver code (Program.cs)

Here is the code that is starting at the beginning as you can see it runs the LoginForm. The main code starts when the LoginForm is closed.

```

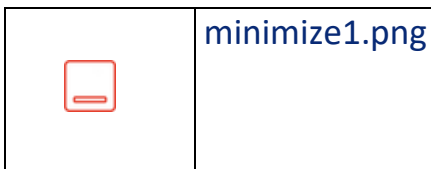
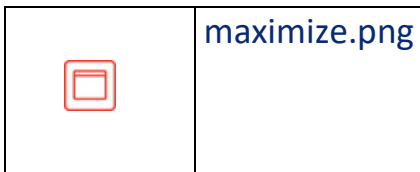
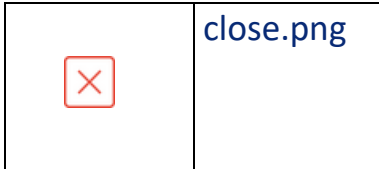
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp56
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LoginForm());
        }
    }
}

```

Resources File

I used extra images for designing the application. I am not sure if it might be a trouble that is working without this image files. Here is the images I used at the project:



Conclusion

In this part, I want to talk about what I learned when I was preparing this program.

- I learned how to design in C#. I made a lot of researches to get some knowledgements about designing a C# program with details. I added some mini pictures to Close, Minimize, Maximize bar as you can see at main program. I've changed the colors.
- Working with OOP makes your job faster. Classes are useful to implement your code if I haven't used classes in this project it will take more time.
- I enhanced my attention about Artificial Intelligence sub-subject like Machine Learning, Deep Learning with working on Image Processing.
- I used lots of stuffs in my program that I've seen at the Programming Languages course. In my opinion making practice makes learning permanent.

Directly accessing program through Google Drive link

I've tried to add all codes to this report. If you download and implement needed libraries from NuGet and do the necessary works I've mentioned in this report you will not get any error. But for making all calculations I've created a public Google Drive file and added the last version of program.

If you need you can use this Google Drive link to access program:
https://drive.google.com/drive/folders/1KXUusggqbfV49r_Gf5TdhUCQU4BKYEP4?usp=sharing

References

- <https://medium.com/@agustinddev/face-recognition-with-opencv-280ec1213ffd>
- <https://docs.microsoft.com/tr-tr/dotnet/api/system.windows.forms.flowlayoutpanel?view=net-5.0>
- <https://www.codeproject.com/Articles/239849/Multiple-Face-Detection-and-Recognition-in-Real-2>
- https://www.youtube.com/watch?v=F-vKCycPmUk&ab_channel=Gen%C3%A7ay%C4%B1d%C4%B1z
- <https://en.wikipedia.org/wiki/Eigenface>
- <https://www.gencayyildiz.com/blog/emgucv-multiple-face-recognitioncoklu-yuz-tanima/>