﻿using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.CvEnum;

using System.IO;

using System.IO.Ports;

using System.Diagnostics;

namespace MultiFaceRec

{

public partial class FrmPrincipal : Form

{

//Declararation of all variables, vectors and haarcascades

Image<Bgr, Byte> currentFrame;

Capture grabber;

HaarCascade face;

HaarCascade eye;

MCvFont font = new MCvFont(FONT.CV\_FONT\_HERSHEY\_TRIPLEX, 0.5d, 0.5d);

Image<Gray, byte> result, TrainedFace = null;

Image<Gray, byte> gray = null;

List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>();

List<string> labels= new List<string>();

List<string> NamePersons = new List<string>();

int ContTrain, NumLabels, t;

string name, names = null;

private void label4\_Click(object sender, EventArgs e)

{

}

private void groupBox2\_Enter(object sender, EventArgs e)

{

}

private void serialPort1\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

}

public FrmPrincipal()

{

InitializeComponent();

//Load haarcascades for face detection

face = new HaarCascade("haarcascade\_frontalface\_default.xml");

//eye = new HaarCascade("haarcascade\_eye.xml");

try

{

//Load of previus trainned faces and labels for each image

string Labelsinfo = File.ReadAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt");

string[] Labels = Labelsinfo.Split('%');

NumLabels = Convert.ToInt16(Labels[0]);

ContTrain = NumLabels;

string LoadFaces;

for (int tf = 1; tf < NumLabels+1; tf++)

{

LoadFaces = "face" + tf + ".bmp";

trainingImages.Add(new Image<Gray, byte>(Application.StartupPath + "/TrainedFaces/" + LoadFaces));

labels.Add(Labels[tf]);

}

}

catch(Exception e)

{

//MessageBox.Show(e.ToString());

MessageBox.Show("Nothing in binary database, please add at least a face(Simply train the prototype with the Add Face Button).", "Triained faces load", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

//Capture Image

grabber = new Capture();

grabber.QueryFrame();

//Initialize the FrameGraber event

Application.Idle += new EventHandler(FrameGrabber);

}

private void button2\_Click(object sender, System.EventArgs e)

{

try

{

//Trained face counter

ContTrain = ContTrain + 1;

//Get a gray frame from capture device

gray = grabber.QueryGrayFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//Face Detector

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

//Action for each element detected

foreach (MCvAvgComp f in facesDetected[0])

{

TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();

break;

}

//resize face detected image for force to compare the same size with the

//test image with cubic interpolation type method

TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

trainingImages.Add(TrainedFace);

labels.Add(textBox1.Text);

//Show face added in gray scale

imageBox1.Image = TrainedFace;

//Write the number of triained faces in a file text for further load

File.WriteAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString() + "%");

//Write the labels of triained faces in a file text for further load

for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)

{

trainingImages.ToArray()[i - 1].Save(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");

File.AppendAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");

}

MessageBox.Show(textBox1.Text + "´s face detected and added :)", "Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

MessageBox.Show("Enable the face detection first", "Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

void FrameGrabber(object sender, EventArgs e)

{

label3.Text = "0";

//label4.Text = "";

NamePersons.Add("");

//Get the current frame form capture device

currentFrame = grabber.QueryFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//Convert it to Grayscale

gray = currentFrame.Convert<Gray, Byte>();

//Face Detector

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

//Action for each element detected

foreach (MCvAvgComp f in facesDetected[0])

{

t = t + 1;

result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

//draw the face detected in the 0th (gray) channel with blue color

currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

if (trainingImages.ToArray().Length != 0)

{

//TermCriteria for face recognition with numbers of trained images like maxIteration

MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

//Eigen face recognizer

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

3000,

ref termCrit);

name = recognizer.Recognize(result);

//Draw the label for each face detected and recognized

currentFrame.Draw(name, ref font, new Point(f.rect.X - 2, f.rect.Y - 2), new Bgr(Color.LightGreen));

}

NamePersons[t-1] = name;

NamePersons.Add("");

//Set the number of faces detected on the scene

label3.Text = facesDetected[0].Length.ToString();

/\*

//Set the region of interest on the faces

gray.ROI = f.rect;

MCvAvgComp[][] eyesDetected = gray.DetectHaarCascade(

eye,

1.1,

10,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(20, 20));

gray.ROI = Rectangle.Empty;

foreach (MCvAvgComp ey in eyesDetected[0])

{

Rectangle eyeRect = ey.rect;

eyeRect.Offset(f.rect.X, f.rect.Y);

currentFrame.Draw(eyeRect, new Bgr(Color.Blue), 2);

}

\*/

}

t = 0;

//Names concatenation of persons recognized

for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)

{

names = names + NamePersons[nnn] + ", ";

serialPort1.Open();

serialPort1.Write("A");

serialPort1.Close();

}

//Show the faces procesed and recognized

imageBoxFrameGrabber.Image = currentFrame;

label4.Text = names;

names = "";

//Clear the list(vector) of names

NamePersons.Clear();

}

}

}