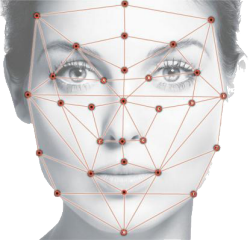


**IP Camera Control with Face detection and   
 Face Recognition**





JULY 2, 2017  
SUPERVISOR  
**Prof. Yousef B. Mahdy**

ACKNOWLEDGEMENT

first we should thanks Allah for helping us to complete this project and produce it in this acceptable view .

we would like to thank our faculty management team for provision computers and facilities all the time of developing the project.

We are proud of faculty of computer and information ,assiut university where this work has been performed.

We wish to express our deep gratitude to our supervisior

**Prof. Yousef B. Mahdy**

For his encourgement and helping us with valuable suggestions through prepration of this project special thanks for.

Finally,our greatest dept to our families for their outstanding support which allowed us to be what we are,they were responsible for getting all of the complex pieces put together instead of us to give us the time to be creative and successful.

**Best regards,  
Team of the project**

**Project team members:**

* **Mathew Samer**
* **Pola Yohanna**
* **Antonyos Alques Damian**
* **Fady Khayrat**
* **Beshoy Mousa**

**Chapter 1: Stream**

**Concepts of chapter**

1.1- Common System……………………………… 6

1.2- AForge.NET libarary ………………………… 6

1.3- Introduction to Capture ………………………… 18

1.4- Llibarary Used ……………………………… 21

**Chapter 2: Face detection**

**Concepts of chapter:**

2.1 Introduction ………………………………………….24

2.2 Algorithms……………………………………………….25

2.3 Common Algorithms…………………………………..26

2.4 Face Detection using Haar Cascades…………………….30

2.5 Cascading classifiers…………………………………..35

**Chapter 3: Face recognition**

**Contents of Chapter:**

3.1Introduction…………………………………… **.**36

3.2 Advantages and disadvantages…………………36

3.3 Recognition algorithm……..……………………40

3.4 Principals Component Analysis (PCA)…………46

**Chapter 4: Implementation**

**Contents of Chapter:**

4.1 Introduction …………………………**..............**49

4.2 Steps for making app …………….……………………49

4.3 Full implementation ……………………59

**Chapter 1  
 Stream**

**1.1 Common Systems**

**OpenCV :** one of the best systems working in face recognition and almost most used

**OpenCV** (*Open Source Computer Vision*) is a [library of programming functions](https://en.wikipedia.org/wiki/Library_(computing)) mainly aimed at real-time [computer vision](https://en.wikipedia.org/wiki/Computer_vision). Originally developed by [Intel](https://en.wikipedia.org/wiki/Intel_Corporation)

**OpenCV** : is written in [C++](https://en.wikipedia.org/wiki/C%2B%2B) and its primary interface is in C++, but it still retains a less comprehensive though extensive older [C interface](https://en.wikipedia.org/wiki/C_(programming_language)).

**EmguCV** : is a cross platform .Net wrapper to the [OpenCV](http://www.emgu.com/wiki/index.php/OpenCV) image processing library. Allowing [OpenCV](http://www.emgu.com/wiki/index.php/OpenCV) functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc.

**AForge.NET** is a [computer vision](https://en.wikipedia.org/wiki/Computer_vision) and [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) library originally for the [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework).

The framework's API includes support for

:[Computer vision](https://en.wikipedia.org/wiki/Computer_vision" \o "Computer vision), [image processing](https://en.wikipedia.org/wiki/Image_processing) and video processing Including a comprehensive image filter library

**1.2 AForge.NET libarary:**

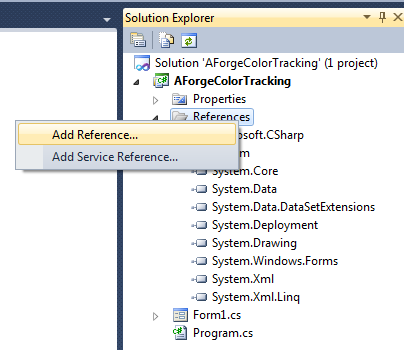
AForge.NET is a computer vision and artificial intelligence library originally developed by Andrew Kirillov for the .NET Framework.

The source code and binaries of the project are available under the terms of the Lesser GPL and the GPL (GNU General Public License).

We are going to walk through a demo on how to setup a project using Visual Studio. I will be using Visual Studio Express C# 2010 for this tutorial. However any version should work just fine.

To get started go t[o http://www.aforgenet.com/framework/downloads.html](http://www.aforgenet.com/framework/downloads.html) and download the installer

First we need to add a reference to the Aforge.NET. Right click on References and select Add Reference.



Then navigate to the Aforge.NET release folder: C:\Program Files\AForge.NET\Framework\Release We want to add reference to four libraries.

C:\Program Files\AForge.NET\Framework\Release\AForge.dll C:\Program Files\AForge.NET\Framework\Release\AForge.Imaging.dll C:\Program Files\AForge.NET\Framework\Release\AForge.Video.dll

C:\Program Files\AForge.NET\Framework\Release\AForge.Video.DirectShow.dll

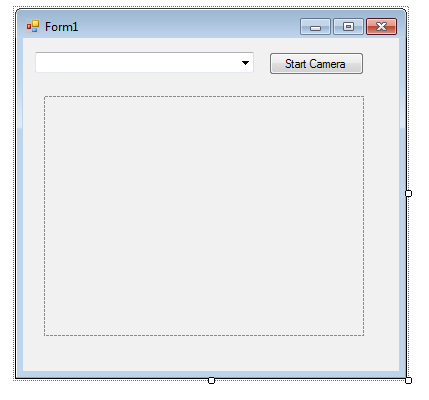
Next double click on Form1.cs in the Solution Explorer menu to the right. Then press F7 to go into view code.

And add the reference to the top of the code.

using AForge;

using AForge.Imaging; using AForge.Video;

using AForge.Video.DirectShow;

Add three controls to your userform, PictureBox, Button and ComboBox.

Rename each control.

ComboBox –> cboCamera Button → btnStartCamera PictureBox → picVideo

Change the text on the button to “Start Camera” and Set the Size for PictureBox to 320,240

Set PictureBox SizeMode to StretchImage

Next double click anywhere on the form where there is not a control. This will create a Load Event for the form.

// Add Global References

private FilterInfoCollection colCamera; private VideoCaptureDevice CaptureDevice;

private void Form1\_Load(object sender, EventArgs e)

{

FillInDropDownCamera();

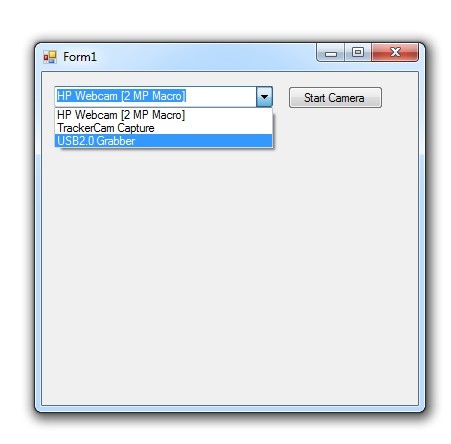
}

void FillInDropDownCamera()

{

colCamera = new FilterInfoCollection(FilterCategory.VideoInputDevice); foreach (FilterInfo Device in colCamera)

{

cboCamera.Items.Add(Device.Name); cboCamera.SelectedIndex = 0; CaptureDevice = new VideoCaptureDevice();

}

}

You should see the list of cameras available.

This will make an event for the button.

We want to add the code to the button

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

{

CaptureDevice = new VideoCaptureDevice(colCamera[cboCamera.SelectedIndex].MonikerString);

CaptureDevice.NewFrame += new NewFrameEventHandler(FinalFrame\_NewFrame); CaptureDevice.Start();

}

void FinalFrame\_NewFrame(object sender, NewFrameEventArgs eventArgs)

{

Bitmap video1 = (Bitmap)eventArgs.Frame.Clone(); picVideo.Image = video1;

}

Next we need to add two more events, We will start with a FormClosed event.

Select the border of the userform and click on the lightning bolt icon at the top of properties menu. Find the FormClosed event and double click next to it to add the event,

Next add an event for the ComboBox DropDown.

Select the ComboBox and select the lightning bolt. Double Click on SelectedIndexChanged

You will now see the two events added to your code.

Add the following code. This will ensure that the camera closes on exit. And when the ComboBox changes selection.

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

CloseCamera();

}

private void cboCamera\_SelectedIndexChanged(object sender, EventArgs e)

{

CloseCamera();

}

void CloseCamera()

{

if (CaptureDevice != null)

{

if (CaptureDevice.IsRunning == true)

{

CaptureDevice.Stop();

picVideo.Image = new Bitmap(640, 480);

}

}

Next let's add a second PictureBox to our Userform.

You can select the picturebox and copy and paste a copy with CTRL C and CTRL V. Move the box to the right and rename to picVideoProcessed.

Add a line to the CloseCamera subroutine to blank out the video on close

picVideoProcessed.Image = new Bitmap(640, 480); void CloseCamera()

{

if (CaptureDevice != null)

{

if (CaptureDevice.IsRunning == true)

{

CaptureDevice.Stop();

picVideo.Image = new Bitmap(640, 480);

→ picVideoProcessed.Image = new Bitmap(640, 480);

}

}

}

We need to add a reference to the second PictureBox to display the image.

Add the following code

Bitmap video2 = (Bitmap)eventArgs.Frame.Clone(); picVideoProcessed.Image = video2;

picVideoProcessed.Image = new Bitmap(640, 480);

void FinalFrame\_NewFrame(object sender, NewFrameEventArgs eventArgs)

{

Bitmap video1 = (Bitmap)eventArgs.Frame.Clone(); picVideo.Image = video1;

* Bitmap video2 = (Bitmap)eventArgs.Frame.Clone();
* picVideoProcessed.Image = video2;

}

void CloseCamera()

{

if (CaptureDevice != null)

{

if (CaptureDevice.IsRunning == true)

{

CaptureDevice.Stop();

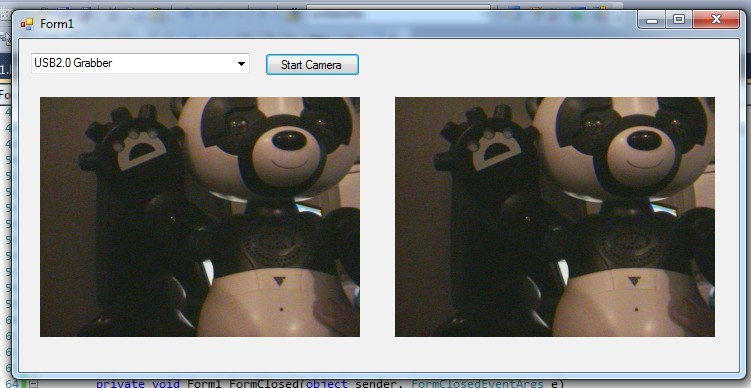
picVideo.Image = new Bitmap(640, 480);

→ picVideoProcessed.Image = new Bitmap(640, 480);

}

}

}



Now we can apply different filters to affect the processed video.

Each frame is process separately as the FinalFrame\_NewFrame event is triggered. Add a new filter

System.Drawing.Image MirrorImage(System.Drawing.Image Image1)

{

// create filter

Mirror filter = new Mirror(false, true);

// apply the filter filter.ApplyInPlace((Bitmap)Image1); return Image1;

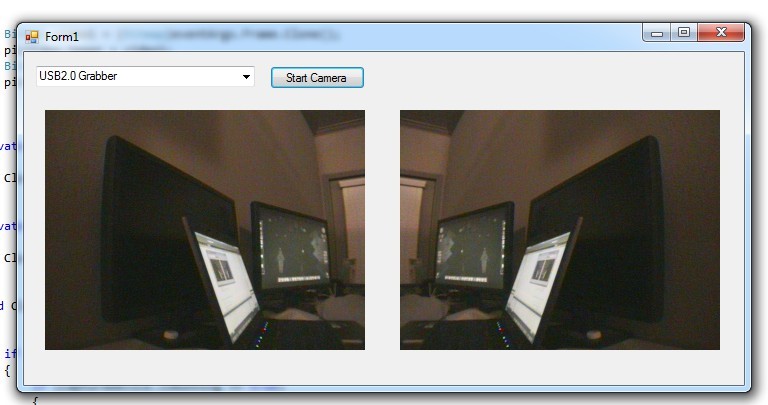
}

# And edit the line

picVideoProcessed.Image = video2;

# to

picVideoProcessed.Image = MirrorImage(video2);



Finally Let's Add a checkbox to our form and name it chkRaw and add the following code.

This code will try to track an object based on it's color. We can run it inline with other filters.

int intRedLower = 50; int intRedUpper = 100; int intGreenLower = 0; int intGreenUpper = 50; int intBlueLower = 0; int intBlueUpper = 50;

System.Drawing.Image CameraTracking(Bitmap video2)

{

// create filter

ColorFiltering colorFilter = new ColorFiltering();

// set color ranges to keep

colorFilter.Red = new IntRange(intRedLower, intRedUpper); colorFilter.Green = new IntRange(intGreenLower, intGreenUpper); colorFilter.Blue = new IntRange(intBlueLower, intBlueUpper);

// apply the filter colorFilter.ApplyInPlace(video2); BlobCounter BlobCounter = new BlobCounter();

// MinHeight doesn't seem to do anything? BlobCounter.MinHeight = 50;

BlobCounter.ObjectsOrder = ObjectsOrder.Size; BlobCounter.ProcessImage(video2);

Rectangle[] rect = BlobCounter.GetObjectsRectangles(); if (chkRaw.Checked)

{

if (rect.Length > 0)

{

for (int i = 0; i <= rect.Length - 1; i++)

{

Rectangle obj = rect[i];

{

Graphics graphic = Graphics.FromImage(video2); using (Pen Pen = new Pen(Color.White, 3))

{

graphic.DrawRectangle(Pen, obj);

}

graphic.Dispose();

}

}

}

}

else{

bool bolRectFound = false;

if (rect.Length > 0)

{

Rectangle obj = rect[0];

int X = obj.X + (obj.Width / 2); int Y = obj.Y + (obj.Height / 2);

// filter out small objects int intThreshhold = 5;

if (obj.Width > intThreshhold && obj.Height > intThreshhold)

{

Graphics graphic = Graphics.FromImage(video2); using (Pen Pen = new Pen(Color.White, 3))

{

graphic.DrawRectangle(Pen, obj);

}

graphic.Dispose(); bolRectFound = true;

}

}

}

return video2;

}

# and then we can run the code on each frame.

void FinalFrame\_NewFrame(object sender, NewFrameEventArgs eventArgs)

{

Bitmap video1 = (Bitmap)eventArgs.Frame.Clone(); picVideo.Image = video1;

Bitmap video2 = (Bitmap)eventArgs.Frame.Clone();

→ picVideoProcessed.Image = CameraTracking((Bitmap) MirrorImage(video2));

}

# You can adjust the color ranges by changing the intRefLower and intRedUpper Ranges.

**1.3 Introduction to Capture**

Recognition systems are more important for distinguishing Safety and secure for the firm whose using this system and checking who go in and out for many reasons and purposes like attendance , organize and other later uses which

participate in firm's reputation

One of the most important in Recognition Systems is the first phase which is capture phase which means capture the video directly from the camera

This phase important because all other phases depend specifically on its accuracy and quality

This phase is participating in all phases because every frame detected from it executed by a lot of tasks and image processing parts executable operations

So any frame less or more will change in results or accuracy that it wanted to be so for above mentions Capture phase is important for many reasons and above of all next operations that will be affected by its results

**Video Capture**

Video capture is the process of converting an analog video signal—such as that produced by a video camera or DVD player—to digital video. The resulting digital data are computer files referred to as a digital video stream, or more often, simply video stream. This is in contrast with screencasting, in which previously digitized video is captured while displayed on a digital monitor.

Our system depends heavily on capture which is important for next parts and operations of system because its accuracy participate in quality of next parts

Capture is process of display directly live scene detected from camera and show to user instantly

And convert scenes detected to input to enter in other operations and processing later mentioned and as mentioned earlier good capture quality and accuracy brings great results of other operations

Capture divided into parts as our system worked here as

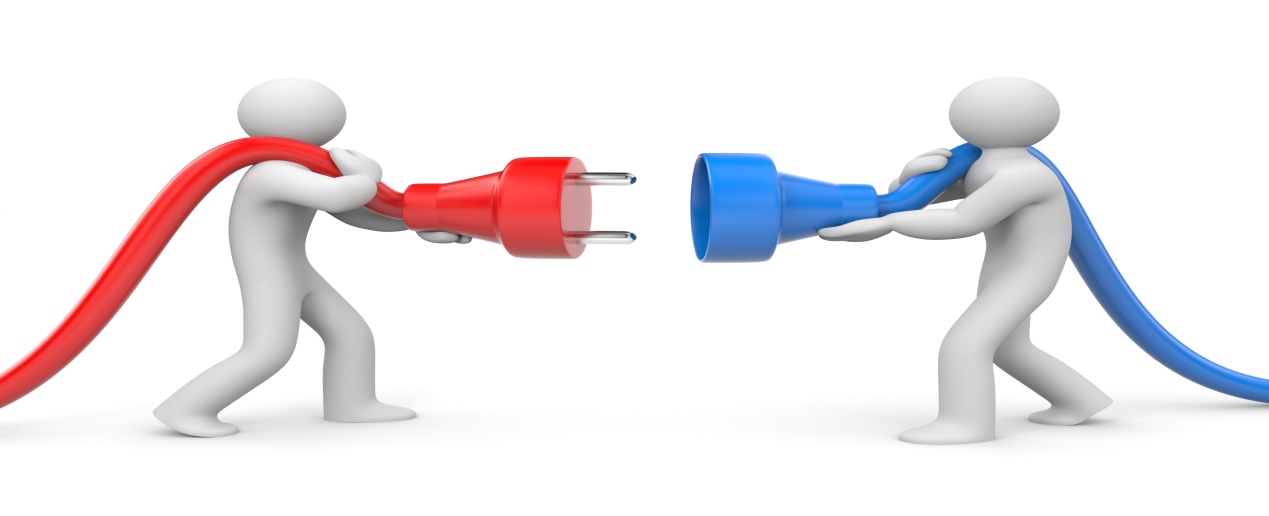
Our system works with ip camera connection and the parts of the process is: -

1- Web Connection

2- receive what comes from connection established

3- convert connection output to video and input to next operations

First: Web Connection :-



Ip camera need ip connection to connect to user by http link used to link to browser to begin displaying what camera detect but it is different manner for other displaying systems like our system that require specific http request to join the http connection and start receiving what to receive and display

Second : receive from connection established

The http request and the http link that established correctly start with display directly on ay browser but as for the system it will not display what it is expected to display

So our system after the http connection would be ready for receiving some frames and pictures because it will not receive what we want from it to receive so receive some frames with time is like receive some queries just as video include and formed from so system can process these queries and do what it will

Third: Display frames combined as video:- 

Third phase is converting frames our system getting by time to live video displaying directly to user and an input to specific input to other next operations that system do to complete its functionalities and this is done by get the frames along time and put them on the display directly after convert it to the specific form that would display accept it and it can join other previously

Frames and prepare every frame also to the operations will be done on it after or before getting out to the user watching it

**1.4 Libraries Used :**

First : System.net

The System.Net namespace provides a simple programming interface for many of the protocols used on networks today. The [WebRequest](https://msdn.microsoft.com/en-us/library/system.net.webrequest(v=vs.110).aspx) and [WebResponse](https://msdn.microsoft.com/en-us/library/system.net.webresponse(v=vs.110).aspx) classes form the basis of what are called pluggable protocols, an implementation of network services that enables you to develop applications that use Internet resources without worrying about the specific details of the individual protocols.

Classes in the System.Net namespace are affected by network isolation feature

Specified Classes

First WebRequest:-

Makes a request to a Uniform Resource Identifier (URI). This is an **abstract** class

Second WebResponse :-

Provides a response from a Uniform Resource Identifier (URI). This is an **abstract** class.

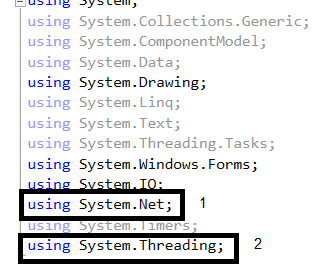
Second : Threading:-

The System.Threading namespace provides classes and interfaces that enable multithreaded programming. In addition to classes for synchronizing thread activities and access to data (Mutex, Monitor, Interlocked, AutoResetEvent, and so on), this namespace includes a ThreadPool class that allows you to use a pool of system-supplied threads, and a Timer class that executes callback methods on thread pool threads

Thread Class:-

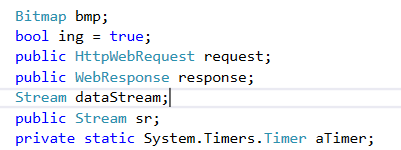
Creates and controls a thread, sets its priority, and gets its status.

**Source Code**



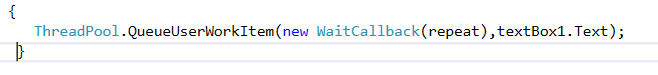
1- System.net that used for initial connection

2- System.Threading used for receiving and streaming over time



HttpWebrequest and WebResponse that both used for initialize connection and begin receiving from server that stream the live video

DataStream that get the stream and the streamed frames from connection established already



Thread that continue streaming over time

One for every live video and camera in this system



First statements declare the connection with the corresponding http server link

And but then the response in stream object that take frame and save it in buffer to

**Chapter 2  
 Face detection**

**2.1 Introduction**

Face detection and tracking is the process of determining whether or not a face is present in an image. Unlike face recognition—which distinguishes different human faces, face detection only indicates whether or not a face is present in an image. In addition, face tracking determines the exact location of the face. Face detection and tracking has been an active research area for a long time because it is the initial important step in many different applications, such as video surveillance, face recognition, image enhancement, video coding, and energy conservation. The applicability of face detection in energy conservation is not as obvious as in other applications. However, it is interesting to learn how a face detection and tracking system allows power and energy to be saved. Suppose one is watching a television and working on other tasks simultaneously. The face detection system is for checking whether or not the person is looking directly at the TV. If the person is not directly looking at the TV within some time period (i.e. 15 minutes), the TV’s brightness is reduced to save energy. When the person turns back to look at the TV, the TV’s brightness can be increased back to original. In addition, if the person looks away for too long (i.e. more than one hour), then the TV will be automatically turned off. Different approaches to detect and track human faces—including feature-based, appearance-based, and color-based have been actively researched and published in literature. The feature-based approach detects a human’s face based on human facial features—such as eyes and nose. Because of its complexity, this method requires lots of computing and memory resources. Although compared to other methods this one gives higher accuracy rate, it is not suitable for power-limited devices. Hence, a color-based algorithm is more reasonable for applications that require low computational effort. In general, each method has its own advantages and disadvantages. More complex algorithm typically gives very high accuracy rate but also requires lots of computing resources.

**Definition :**

**Face detection** is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene .

* 1. **Algorithms:**

Face detection can be regarded as a specific case of [object-class detection](https://en.wikipedia.org/wiki/Object-class_detection). In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.

Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

A reliable face-detection approach based on the [genetic algorithm](https://en.wikipedia.org/wiki/Genetic_algorithm) and the [eigen-face](https://en.wikipedia.org/wiki/Eigenface) technique:

Firstly, the possible human eye regions are detected by testing all the valley regions in the gray-level image. Then the genetic algorithm is used to generate all the possible face regions which include the eyebrows, the iris, the nostril and the mouth corners.

Each possible face candidate is normalized to reduce both the lightning effect , which is caused by uneven illumination; and the shirring effect, which is due to head movement. The fitness value of each candidate is measured based on its projection on the eigen-faces. After a number of iterations, all the face candidates with a high fitness value are selected for further verification. At this stage, the face symmetry is measured and the existence of the different facial features is verified for each face candidate.

* 1. **Common Algorithms:**

## **Finding faces in images with controlled background:**

This is the easy way out. Use images with a plain monocolour background, or use them with a predefined static background – removing the background will always give you the face boundaries. The rest is easy going…

### **Finding faces by color:**

If you have access to color images, you might use the typical skin color to find face segments. The disadvantage: doesn’t work with all kind of skin colors, and is not very robust under varying lighting conditions…

# Locating and tracking faces using color

# A system for detecting and tracking faces was previously described . It combined motion detection by spatio-temporal filtering with an appearance-based face model in the form of a neural net. Multiple person tracking was performed using time-symmetric matching and Kalman filtering. In this section, the use of colour as a cue for detection and tracking is described. Colour provides a computationally efficient yet effective method which is robust under rotations in depth and partial occlusions. It can be combined with motion and appearance-based face detection.

Human skin forms a relatively tight cluster in colour space even when different races are considered [[5](http://web.archive.org/web/20040815172250/http:/www.dcs.qmul.ac.uk/research/vision/publications/papers/bmvc97/node7.html#hunkwaib94)]. Figure 3 shows the colour distribution of three faces in hue-saturation (H-S) space. Face colour distributions were modelled as Gaussian mixtures of the form:

  equation415

The mixing parameter *P*(*j*) corresponds to the prior probability that the data, tex2html_wrap_inline679, was generated by component *j*. Each mixture component, tex2html_wrap_inline683, is a Gaussian with mean tex2html_wrap_inline685and covariance matrix . Given *n* face pixels tex2html_wrap_inline691, tex2html_wrap_inline693, Expectation-Maximisation (EM) provides an effective maximum-likelihood algorithm for learning a Gaussian mixture model [[9](http://web.archive.org/web/20040815172250/http:/www.dcs.qmul.ac.uk/research/vision/publications/papers/bmvc97/node7.html#redn84)]. An expectation (E) step consists of evaluating the posterior probabilities tex2html_wrap_inline695for each mixture component. Let the sum of these probabilities be tex2html_wrap_inline697. A maximisation (M) step then updates the mixture components as follows:

equation417

equation419

The E and M steps are iterated until convergence. If *M*=1, the parameters of the Gaussian are estimated directly.

In practice, an H-S model of a single person functions well with other races. The mixture model is used to assign a probability to each pixel in an image and faces are detected by grouping suitably sized areas of high probability. A face is tracked by estimating the position as the mean tex2html_wrap_inline707and the spatial extent as the vertical and horizontal standard devaitions tex2html_wrap_inline709of the local colour probability distribution in the image plane. For a given frame t, the box position tex2html_wrap_inline707is estimated as an offset from the position tex2html_wrap_inline713:

equation421

where tex2html_wrap_inline723ranges over all image coordinates in the region of interest and is the colour point at image position tex2html_wrap_inline723. To improve accuracy, probabilities tex2html_wrap_inline729are thresholded. Values lower than the threshold are taken to be background and are consequently set to zero in order to nullify their influence on the estimation of tex2html_wrap_inline707and tex2html_wrap_inline709. The size of the bounding box is estimated by computing the standard deviation weighted by the pixel probabilities:

equation423

Figure [2](http://web.archive.org/web/20040815172250/http:/www.dcs.qmul.ac.uk/research/vision/publications/papers/bmvc97/node2.html#figtrackedseqs) shows a sequence of a face being tracked with a moving camera against a cluttered background. The tracker's ability to deal with changes in scale, large rotations in depth and partial occlusion are all clearly demonstrated.

The colour-based tracking system has been implemented on a 200MHz Pentium PC equipped with a Matrox Meteor frame grabber and a Sony EVI-D31 active camera. The camera can be driven by maintaining the mean position, **m**, at the centre of the image. Tracking is performed at approximately 15 frames per second. Some problems are inevitably caused by large changes in the spectral composition of scene illumination. It has been found necessary to use at least two colour models, one for interior lighting and one for exterior natural daylight.

### **Finding faces by motion:**

If you are able to use real-time video, you can use the fact that a face is almost always moving in reality. Just calculate the moving area, and here you go.  
Disadvantages: What if there are other objects moving in the background?

**Motion Detection :**

To detect and analyse movement in a video sequence, we perform to following four steps:

1. Frame differencing

2. Thresholding

3. Noise removal

4. Add up pixels on each line in the motion image

First we find the difference between the current frame in the video sequence and the previous. If the difference between the pixel values are greater than $\mathit{(colors\,\,used)}/10$ , the movement has been significant and the pixel is set to black. If the change is less than this threshold, the pixel is set to white. This image (figure [2](http://web.archive.org/web/20080522171806/http:/www.ansatt.hig.no/erikh/papers/hig98_6/node2.html#bevdet)) now indicates if something has moved and and where the movement is located.

In the thresholded image, there may be noise. To remove the noise, we scan the image with a $3\times3$ . window and remove all black pixels which are isolated in a white area. If the center pixel of the $3\times3$frame is black and less than three of the pixels in the frame are black, we remove the black center pixel because it is probably noise. Otherwise the pixel remains black. This way we detect only "large" moving objects.

$60\times60$$80\times80$$\mathit{(average\,\,width)}/2$$40\times40$This motion image is used to add up how many black pixels there are on each line (figure [3](http://web.archive.org/web/20080522171806/http:/www.ansatt.hig.no/erikh/papers/hig98_6/node2.html#bevdetlinje)). We use this image to find the upper moving object in the images. If there are three lines with movement greater than fifteen pixels below each other, we assume this is an object, not just single pixels with movement. By using the information about how much motion there is on each line, a point in the middle of the upper moving object is calculated. This is done by calculating the center of the object within a square of a fixed size ( pixels). The average width of the object is calculated and the center pixel is where this cross the twentieth line from the top of the moving object. This procedure is repeated for frame sizes . and .

## **Finding faces in unconstrained scenes:**

Well here we go – this is the main thing, the top of them all, the most complicated thing maybe in whole object recognition: Given a black and white still image, where is the face? Humans can do it, so where’s the perfect algorithm that can do it, too? Here is some work on it:

1. [Model-based Face Tracking](http://www.cs.rutgers.edu/%7Edecarlo/facetrack.html)  
   There seems to be a revival of edge-based methods, using geometric models. Two top performing methods have been published in the early 2000s:
   * [Real-Time Face Detection Using Edge-Orientation Matching](http://link.springer.com/chapter/10.1007%2F3-540-45344-X_12)  
     Fröba, Küblbeck: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.
   * [Robust Face Detection Using the Hausdorff Distance](https://facedetection.com/wp-content/uploads/AVBPA01BioID.pdf)  
     Jesorsky, Kirchberg, Frischholz: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.
     + [Genetic Model Optimization for Hausdorff Distance-Based Face Localization](https://facedetection.com/wp-content/uploads/BIOMET02HumanScan.pdf)  
       Kirchberg, Jesorsky, Frischholz: International ECCV Workshop on Biometric Authentication, Springer, Lecture Notes in Computer Science, LNCS-2359, pp. 103-111, Copenhagen, Denmark, June 2002.
2. [Weak classifier cascades](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.8268&rep=rep1&type=pdf)  
   The breakthrough in face detection happened with [Viola & Jones](http://en.wikipedia.org/wiki/Viola-Jones_object_detection_framework). Using a cascade of “weak-classifiers”, using simple Haar features, can – after excessive training – yield impressive results. This approach is now the **most commonly used algorithm for face detection.** A basic implementation is included in OpenCV.
   1. **Face Detection using Haar Cascades**

## **Basics**

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

****

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by **Adaboost**.



For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Wow.. Isn't it a little inefficient and time consuming? Yes, it is. Authors have a good solution for that.

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of **Cascade of Classifiers**. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. How is the plan !!!

Authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is actually obtained as the best two features from Adaboost). According to authors, on an average, 10 features out of 6000+ are evaluated per sub-window.

So this is a simple intuitive explanation of how Viola-Jones face detection works. Read paper for more details or check out the references in Additional Resources section.

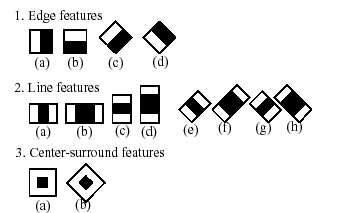
# 2.5 Cascading classifiers

**Cascading** is a particular case of [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) based on the concatenation of several [Classifiers](https://en.wikipedia.org/wiki/Statistical_classification), using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Unlike voting or stacking ensembles, which are multiexpert systems, cascading is a multistage one.

Cascading Classifiers are trained with several hundred "positive" sample views of a particular object and arbitrary "negative" images of the same size. After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location for the classifier. This process is most commonly used in [image processing](https://en.wikipedia.org/wiki/Image_processing) for object detection and tracking, primarily [facial detection](https://en.wikipedia.org/w/index.php?title=Facial_detection&action=edit&redlink=1) and recognition.

The first cascading classifier is the face detector of [Viola and Jones (2001)](https://en.wikipedia.org/wiki/Viola-Jones_object_detection_framework). The requirement was that the classifier be fast in order to be implemented on low CPU systems, such as cameras and phones.

* a classifier (namely a *cascade of boosted classifiers working with haar-like features*) is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size
* After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.
* The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (*stages*) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques (weighted voting). Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are supported. The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifiers, and are calculated as described below. The current algorithm uses the following Haar-like features:



The feature used in a particular classifier is specified by its shape (1a, 2b etc.), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over a rectangular regions are calculated rapidly using integral images

**Chapter 3  
 Face recognition**

**3.1 INTRODUCTION**

Recently the identification and authentication of every individual is important task in day today’s life. In the massive number of the individuals, we can use the electronic based signal system [1] for the identification methods. For these the biometric based systems have the more importance in present days. The physical characteristics of the human beings like face geometry, fingerprint, iris identity, palm structure can be used as the individual identification for the every individual. The face reorganization based systems have the more advantages than the others. In the other methods like iris and finger detection method we have to go for the primitive actions like physical contact with the system, exact iris position etc., but for face recognition the slide change in face orientation may also be acceptable. In the face recognition system we are taking the set of different oriented images with high pixel camera and used for the feature extraction. The face recognition system has the more advantages over the other biometric system like iris detection, fingerprint recognition.

A **face recognition system** is a [computer application](https://en.wikipedia.org/wiki/Application_software) capable of [identifying](https://en.wikipedia.org/wiki/Identification_of_human_individuals) or [verifying](https://en.wikipedia.org/wiki/Authentication) a person from a [digital image](https://en.wikipedia.org/wiki/Digital_image) or a [video frame](https://en.wikipedia.org/wiki/Film_frame) from a [video](https://en.wikipedia.org/wiki/Video) source. One of the ways to do this is by comparing selected [facial features](https://en.wikipedia.org/wiki/Face) from the image and a face [database](https://en.wikipedia.org/wiki/Database_management_system).It is typically used in [security systems](https://en.wikipedia.org/wiki/Burglar_alarm) and can be compared to other [biometrics](https://en.wikipedia.org/wiki/Biometrics) such as [fingerprint](https://en.wikipedia.org/wiki/Fingerprint) or eye [iris recognition](https://en.wikipedia.org/wiki/Iris_recognition) systems.[[1]](https://en.wikipedia.org/wiki/Facial_recognition_system#cite_note-Animetrics-1) Recently, it has also become popular as a commercial identification and marketing tool

**3.2 Advantages and disadvantages**

***Compared to other technologies***

Among the different biometric techniques, face recognition may not be most reliable and efficient.[citation needed] However, one key advantage is that it does not require the cooperation of the test subject to work. Properly designed systems installed in airports, multiplexes, and other public places can identify individuals among the crowd, without passers-by even being aware of the system. Other biometrics like fingerprints, iris scans, and speech recognition cannot perform this kind of mass identification. However, questions have been raised on the effectiveness of face recognition software in cases of railway and airport security.[citation needed]

***Weaknesses***

Face recognition is far from perfect and struggles to perform under certain conditions. Ralph Gross, a researcher at the Carnegie Mellon Robotics Institute, describes one obstacle related to the viewing angle of the face: "Face recognition has been getting pretty good at full frontal faces and 20 degrees off, but as soon as you go towards profile, there've been problems."

Current face recognition still often misidentifies people which can sometimes led to controversy. Google was criticized for racism in its system when a black couple were misidentified as gorillas. Face recognition software generally doesn't do as well in identifying minorities when most of the subjects used in testing the technology were from the majority group.

Other conditions where face recognition does not work well include poor lighting, sunglasses, hats, scarves, beards, long hair, makeup or other objects partially covering the subject’s face, and low resolution images.

Another serious disadvantage is that many systems are less effective if facial expressions vary. Even a big smile can render the system less effective. For instance: Canada now allows only neutral facial expressions in passport photos.

There is also inconstancy in the datasets used by researchers. Researchers may use anywhere from several subjects to scores of subjects, and a few hundred images to thousands of images. It is important for researchers to make available the datasets they used to each other, or have at least a standard dataset.

***Effectiveness***

Critics of the technology complain that the London Borough of Newham scheme has, as of 2004, never recognized a single criminal, despite several criminals in the system's database living in the Borough and the system having been running for several years. "Not once, as far as the police know, has Newham's automatic face recognition system spotted a live target."[22][31] This information seems to conflict with claims that the system was credited with a 34% reduction in crime (hence why it was rolled out to Birmingham also).[32] However it can be explained by the notion that when the public is regularly told that they are under constant video surveillance with advanced face recognition technology, this fear alone can reduce the crime rate, whether the face recognition system technically works or does not. This has been the basis for several other face recognition based security systems, where the technology itself does not work particularly well but the user's perception of the technology does.

An experiment in 2002 by the local police department in Tampa, Florida, had similarly disappointing results.

A system at Boston's Logan Airport was shut down in 2003 after failing to make any matches during a two-year test period.

As of 2016, facial recognition is still not effective for most applications even though the accuracy has been substantially improved. Although systems are often advertised as having accuracy near 100%, this is misleading as the studies often uses much smaller sample sizes than would be necessary for large scale applications. Because facial recognition is not completely accurate, it creates a list of potential matches. A human operator must then look through these potential matches and studies show the operators pick the correct match out of the list only about half the time. This causes the issue of targeting the wrong suspect.

***Privacy issues***

Civil rights right organizations and privacy campaigners such as the Electronic Frontier Foundation and the ACLU express concern that privacy is being compromised by the use of surveillance technologies. Some fear that it could lead to a “total surveillance society,” with the government and other authorities having the ability to know the whereabouts and activities of all citizens around the clock. This knowledge has been, is being, and could continue to be deployed to prevent the lawful exercise of rights of citizens to criticize those in office, specific government policies or corporate practices. Many centralized power structures with such surveillance capabilities have abused their privileged access to maintain control of the political and economic apparatus, and to curtail populist reforms.

Face recognition can be used not just to identify an individual, but also to unearth other personal data associated with an individual – such as other photos featuring the individual, blog posts, social networking profiles, Internet behavior, travel patterns, etc. – all through facial features alone. Moreover, individuals have limited ability to avoid or thwart face recognition tracking unless they hide their faces. This fundamentally changes the dynamic of day-to-day privacy by enabling any marketer, government agency, or random stranger to secretly collect the identities and associated personal information of any individual captured by the face recognition system.

Social media web sites such as Facebook have very large numbers of photographs of people, annotated with names. This represents a database which may be abused by governments for face recognition purposes. Face recognition was used in Russia to harass women allegedly involved in online pornography. In Russia there is an app 'FindFace' which can identify faces with about 70% accuracy using the social media app called VK. This app would not be possible in other countries which do not use VK as their social media platform photos are not stored the same way as with VK.

In July 2012, a hearing was held before the Subcommittee on Privacy, Technology and the Law of the Committee on the Judiciary, United States Senate, to address issues surrounding what face recognition technology means for privacy and civil liberties.

In 2014, the National Telecommunications and Information Association (NTIA) began a multi-stakeholder process to engage privacy advocates and industry representatives to establish guidelines regarding the use of face recognition technology by private companies. In June 2015, privacy advocates left the bargaining table over what they felt was an impasse based on the industry representatives being unwilling to agree to consent requirements for the collection of face recognition data. The NTIA and industry representatives continued without the privacy representatives, and draft rules are expected to be presented in the spring of 2016.

States have begun enacted legislation to protect citizen's biometric data privacy. Illinois enacted the Biometric Information Privacy Act in 2008. Facebook's DeepFace has become the subject of several class action lawsuits under the Biometric Information Privacy Act, with claims alleging that Facebook is collecting and storing face recognition data of its users without obtaining informed consent, in direct violation of the Biometric Information Privacy Act. The most recent case was dismissed in January 2016 because the court lacked jurisdiction. Therefore, it is still unclear if the Biometric Information Privacy Act will be effective in protecting biometric data privacy rights.

**3.3 Recognition algorithms**

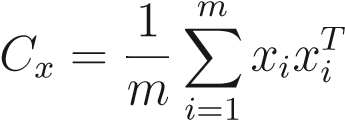
#### **Principal Component Analysis**

One of the most used and cited statistical method is the Principal Component Analysis (PCA) . It is a mathematical procedure that performs a dimensionality reduction by extracting the *principal components* of the multi-dimensional data. The first principal component is the linear combination of the original dimensions that has the highest variability. The *n*-th principal component is the linear combination with the maximum variability, being orthogonal to the *n-1* first principal components. The idea of PCA is illustrated in figure 1.7.

The greatest variance of any projection of the data lies in the first coordinate. The *n*-st coordinate will be the direction of the *n*-th maximum variance - the *n*-th principal component.

Usually the mean *~~x~~* is extracted from the data, so that PCA is equivalent to Karhunen-Loeve Transform (KLT). So, let *Xnxm* be the the data matrix where *x*1*,...,xm* are the image vectors (vector columns) and *n* is the number of pixels per image. The KLT basis is obtained by solving the eigenvalue problem

|  |  |
| --- | --- |
| *Cx* = ΦΛΦ*T*  where *Cx* is the covariance matrix of the data | (1.7) |

 (1.8)

Φ = [*φ*1*,...,φn*] is the eigenvector matrix of *Cx*. Λ is a diagonal matrix, the eigenvalues *λ*1*,...,λn* of *Cx* are located on its main diagonal. *λi* is the variance of the data projected on *φi*.

PCA can be computed using Singular Value Decomposition (SVD). The SVD of the data matrix *Xnxm* is

Figure 1.7: Face image and its DCT

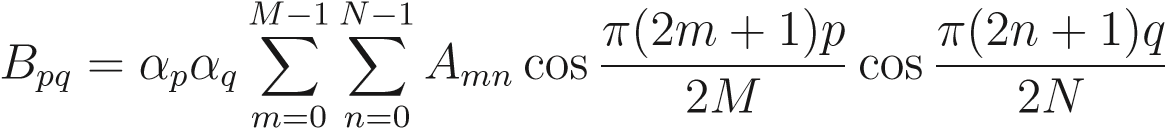
*X* = *UDV T* (1.9)

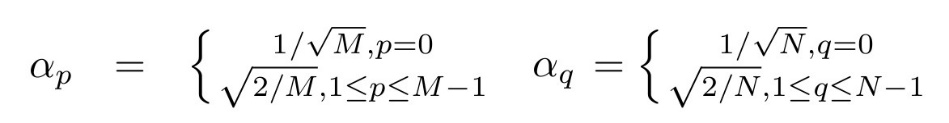
It is known that *U* = Φ . This method allows efficient implementation of PCA without having to compute the data covariance matrix *Cx* -knowing that *Cx* = *UT X*. The embedding is done by *yi* = *UT xi*, thus obtaining the mapped points *y*1*,...,ym*.

#### **Discrete Cosine Transform**

The Discrete Cosine Transform [2] DCT-II standard (often called simply DCT) expresses a sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. It has strong energy compaction properties. Therefore, it can be used to transform images, compacting the variations, allowing an effective dimensionality reduction. They have been widely used for data compression. The DCT is based on the Fourier discrete transform, but using only real numbers.

When a DCT is performed over an image, the energy is compacted in the upper-left corner. An example can be found in image 1.8. The face has been taken from the ORL database[95], and a DCT performed over it. Let *B* be the DCT of an input image *ANxM:*

 (1.10)

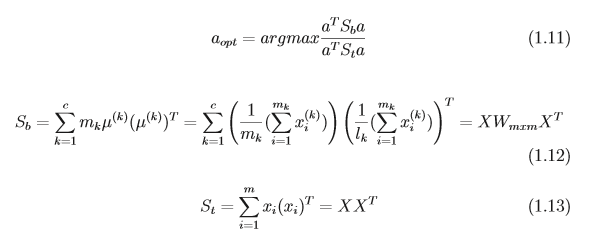


where *M* is the row size and *N* is the column size of *A*. We can truncate the matrix *B*, retaining the upper-left area, which has the most information, reducing the dimensionality of the problem.

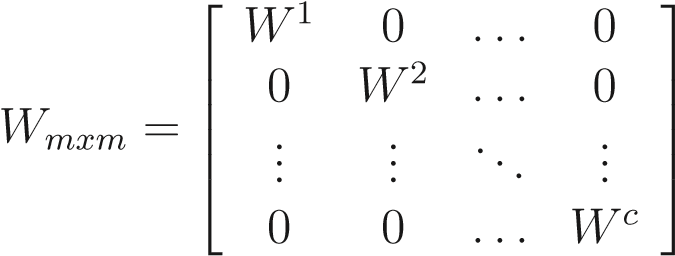
#### **Linear Discriminant Analysis**

LDA is widely used to find linear combinations of features while preserving class separability. Unlike PCA, LDA tries to model the differences between classes. Classic LDA is designed to take into account only two classes. Specifically, it requires data points for different classes to be far from each other, while point from the same class are close. Consequently, LDA obtains differenced projection vectors for each class. Multi-class LDA algorithms which can manage more than two classes are more used.

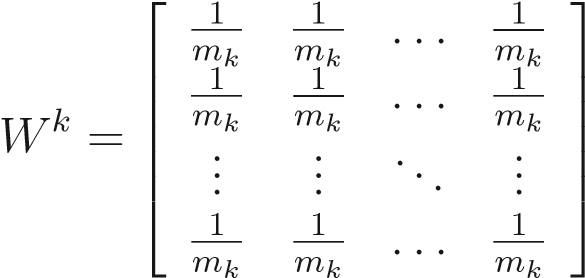
Suppose we have *m* samples *x*1*,*...,x*m* belonging to *c* classes; each class has *mk* elements. We assume that the mean has been extracted from the samples, as in PCA. The objective function of the LDA can be defined [22] as



where *Wmxm* is a diagonal matrix defined as

 (1.14)

and *Wk* is a *mk* × *mk* matrix

 (1.15)

Finally, we can write the eigenproblem:

 (1.16)

The solution of this eigenproblem provides the eigenvectors; the embedding is done like the PCA algorithms does.

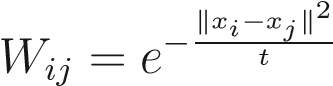
#### **Locality Preserving Projections**

The Locality Preserving Projections (LPP) was introduced by He and Niyogi [42]. It’s an alternative to PCA, designed to preserve locality structure. Pattern recognition algorithms usually make a search for the nearest pattern or neighbors. Therefore, the locality preserving quality of LPP can quicken the recognition.

Let *m* be the number of points that we want to map. In our case, those points correspond to images. The LPP algorithm has four steps:

Constructing the adjacency map: A graph *G* with *m* nodes is built using, for example, k-NN algorithm.

Choosing the weights: Being *Wij*a weight matrix, we can build it using a Heat kernel of parameter *t*-if nodes *i* and *j* are connected, put

 (1.17)

Solving the eigenproblem. *D* is a diagonal matrix where it’s elements are defined as *dii* = *j wij*, and *L=D-W* is the Laplacian matrix. The following eigenproblem must be solved:P

*λa* = *XDXT*(*XLXT*)−1 (1.18)

The embedding process and the PCA’s embedding process are analogous.

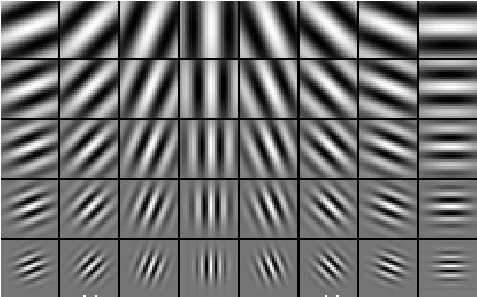
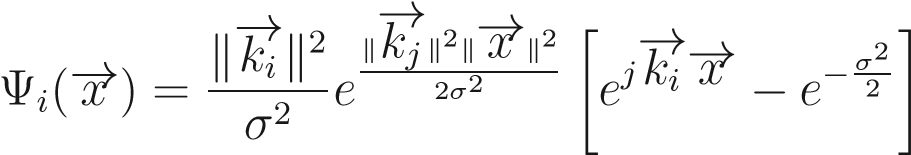


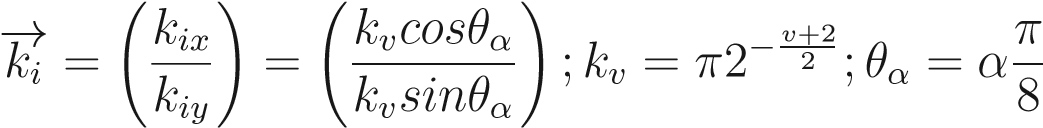
Figure 1.8: Gabor filters.

#### **Gabor Wavelet**

Neurophysiological evidence from the visual cortex of mammalian brains suggests that simple cells in the visual cortex can be viewed as a family of self-similar 2D Gabor wavelets. The Gabor functions proposed by Daugman [28, 63] are local spatial bandpass filters that achieve the theoretical limit for conjoint resolution of information in the 2D spatial and 2D Fourier domains. Daugman generalized the Gabor function to the following 2D form :

 (1.19)

Each Ψ*i* is a plane wave characterized by the vector *ki* enveloped by a Gaussian function, where *σ* is the standard deviation of this Gaussian. The center frequency of *i*-th filter is given by the characteristic wave vector,

 (1.20)

where the scale and orientation is given by (*kv,θα*), being *v* the spatial

frequency number and *α* the orientation.

An image is represented by the Gabor wavelet transform in four dimensions, two are the spatial dimensions, and the other two represent spatial frequency structure and spatial relations or orientation. So, processing the face image with Gabor filters with 5 spatial frequency (*v* = 0*,..,*4) and 8 orientation (*α* = 0*,..,*7) captures the whole frequency spectrum - see image

1.9. So, we have 40 wavelets. The amplitude of Gabor filters are used for recognition.

Once the transformation has been performed, different techniques can be applied to extract the relevant features, like high-energized points comparisons [55].

#### **Independent Component Analysis**

Independent Component Analysis aims to transform the data as linear combinations of statistically independent data points. Therefore, its goal is to provide an independent rather that uncorrelated image representation. ICA is an alternative to PCA which provides a more powerful data representation [67]. It’s a discriminant analysis criterion, which can be used to enhance PCA.

The ICA algorithm is performed as follows [25]. Let *cx* be the covariance matrix of an image sample *X*. The ICA of *X* factorizes the covariance matrix *cx* into the following form: *cx* = *F*∆*FT* where ∆ is diagonal real positive and *F* transforms the original data into *Z* (*X* = *FZ*). The components of *Z* will be the most independent possible. To derive the ICA transformation *F*,

 (1.21)

where *X* and Λ are derived solving the following eigenproblem:

*cx* = ΦΛΦ*T* (1.22)

Then, there are rotation operations which derive independent components minimizing mutual information. Finally, a normalization is carried out.

**3.4 Principals Component Analysis (PCA)**

Principal Component Analysis is proposed by Turk and Pent land in 1991, which is often used for extracting features of image. Principal Component Analysis is most widely used method considering with the face image extraction in image processing. Basic idea behind the PCA [4] is, the set of images are initially transformed into Eigenfacec i.e. lower data space by using the K-L transform method. This method includes the linear transformation of the higher data space into the lower data space using linear transformation method. This extracted lower dimensional image preserves the most of the data or information from the original higher dimensional facial image. This mapped lower data space is called as the Eigenface. Then the test Eigenfaces is projected on the trainee Eigenfaces to get the correct match.

For PCA, two-dimensional image matrix must be first transformed to a one-dimensional vector with high order. While the number of training sample is small, it is very difficult to calculate covariance matrix of training sample accurately. Furthermore, structure information will be lost during processing. The Eigen faces are Principal Components of a distribution of faces, or equivalently, the Eigen vectors of the covariance matrix of the set of the face images. Find Principal Components of the distribution of faces, or the Eigen Vectors of the covariance matrix of the set of face images. Each image location Contributes to each Eigen vector, so that we can display the Eigen vector as a sort of face. Each face image can be represented exactly in terms of linear combination of the Eigen faces. The number of possible Eigen faces is equal to the number of face image in the training set. The faces can also be approximated by using best Eigen face, those that have the largest Eigen values, and which therefore account for most variance between the set of face images. The primary reason for using fewer Eigen faces is computational efficiency.

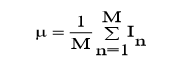
A. Brief Discussion of PCA-

1) First, consider the set of images in the column matrix or the row matrix format, named A



Where, M is the total number of objects present in total database.

2) Find the average of the defined matrix A



Hear, n =is the total number of images in single object of Database

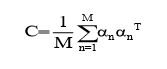
µ=Mean of the defined matrix A

3) Then find the differential distance between the trainee images and the mean calculated



Hear, we will get the α , scatter matrix for each image.

4) Find the covariance matrix C as follows,



5) Find the eigenvector and eigenvalue of the covariance matrix C. These eigenvectors are arranged in descending order and the weighted vector is selected having the highest eigenvalue for the feature extraction.

6) This feature vector is consist of the extracted data of all the images present in database and is compared with the vector of test image.

**Chapter 4   
Implementation**

**4.1 Introduction**

So now we arrive to how simulation Camera control with adding features for it

this features are detect face of person when camera capture person and then recognize it to see if this person belong to u or not all this is done by using some tools this tools include

1) IP webcam: software that run on phone and simulate the camera.

2) Emgu.cv: is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible language such as c#.

**4.2 Steps for making app**

We make this app by implement 3 stages that we will describe in details

First: capture video from ip webcam and load it into picture box

Second: detect face from video we run on picture box

Third: recognize face to see if this person belong to us or not.

See figure 4.1 for app

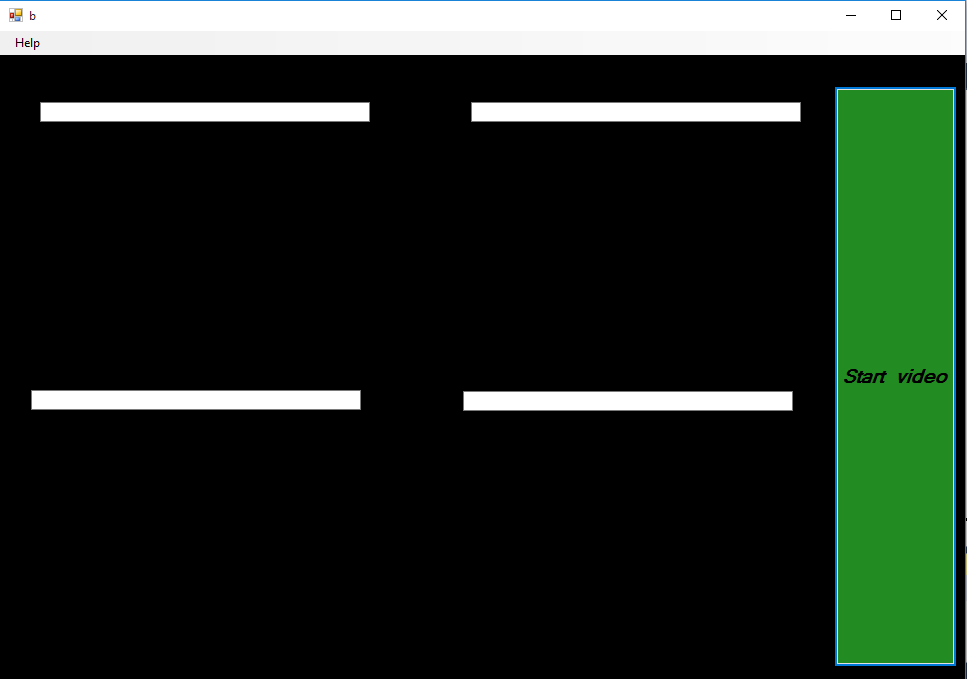


Figure 4.1

**4.2.1 first stage: capture video**

So we have 4 picture box with 4 text box we need every picture box load its video from ip webcam and we know that’s every webcam has its ip so every text box

Have ip an example for URL: is “http://192.168.43.1:8080/shot.jpg";

now we all know the video is a collection of frames so we will make http request for get one frame at time with while loop we will get video that considers collection of sequences of frames.

We need to know that every picture box load its video using threads

See part of implementation below:

// start load frames which h1 is url for ip webcam1 after press start video

th1 = new Thread (() => repeat1 (pictureBox1, h1, a));

th1.Start ();

we will know later what a is refer to, now look repeat1 function:

void repeat1 (Picture Box p, string url,int framenum)

{ while (true)

{

framenum++;

WebRequest requestPic = WebRequest. Create (url);

WebResponse responsePic = requestPic.GetResponse (); p.Image = Image.FromStream (responsePic.GetResponseStream())

}

}

Figure 4.2

So we load video from ip webcam1 in picture box1

**4.2.2 first stage: detect face**

Now we have video on picture box so we move to second stage that’s the face detection so which algorithm we will use Hear Feature-based Cascade Classifiers

We have to understand this basics

1. HaarCascade Haar; = new HaarCascade("haarcascade\_frontalface\_alt\_tree.xml");

This line of code load xml file that contain lot of positive images (images of faces) and negative images (images without faces) to train the classifier to use it for compare.

1. var faces = Grayframe.DetectHaarCascade(Haar, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

This line is very important it take image from video that run on picture box

and use canny to determine filter this image and extract edges from it then compare this features with that found in xml to say if there is face or not in image .

// Summary of this method:

// Finds rectangular regions in the given image that are likely to contain objects

//the cascade has been trained for and returns those regions as a sequence of rectangles.

// The function scans the image several times at different scales

// Each time it considers overlapping regions in the image and applies the classifiers

// to the regions using HaarClassifierCascade. It may also apply some heuristics

// to reduce number of analyzed regions, such as Canny prunning. After it has proceeded

// and collected the candidate rectangles (regions that passed the classifier cascade),

// it groups them and returns a sequence of average rectangles for each large enough

// group. The default parameters (scale\_factor=1.1, min\_neighbors=3, flags=0) are

// tuned for accurate yet slow object detection. For a faster operation on real

// video images the settings are: scale\_factor=1.2, min\_neighbors=2, flags=CV\_HAAR\_DO\_CANNY\_PRUNING,

// min\_size=<minimum possible face size> (for example, ~1/4 to 1/16 of the image

// area in case of video conferencing).

//

// Parameters:

// haarObj:

// Haar classifier cascade in internal representation

//

// scaleFactor:

// The factor by which the search window is scaled between the subsequent scans,

// for example, 1.1 means increasing window by 10%

//

// minNeighbors:

// Minimum number (minus 1) of neighbor rectangles that makes up an object. All

// the groups of a smaller number of rectangles than min\_neighbors-1 are rejected.

// If min\_neighbors is 0, the function does not any grouping at all and returns

// all the detected candidate rectangles, which may be useful if the user wants

// to apply a customized grouping procedure

//

// flag:

// Mode of operation. Currently the only flag that may be specified is CV\_HAAR\_DO\_CANNY\_PRUNING.

// If it is set, the function uses Canny edge detector to reject some image regions

// that contain too few or too much edges and thus can not contain the searched

// object. The particular threshold values are tuned for face detection and in this

// case the pruning speeds up the processing.

//

// minSize:

// Minimum window size. By default, it is set to the size of samples the classifier

// has been trained on (~20x20 for face detection)

//

// Returns:

// The objects detected, one array per channel

Note: you must ask yourself if you make detect on every frame or you can lose some of it so to make your app more speed not more accuracy you need to ignore some frames

So the full implementation:

void repeat1(PictureBox p, string url,int framenum)

{

while (true)

{

framenum++;

WebRequest requestPic = WebRequest.Create(url);

WebResponse responsePic = requestPic.GetResponse();

if (framenum % 3 == 0)

{ classification s = new classification(p, Image.FromStream(responsePic.GetResponseStream()));

lock (this)

{

Thread.Sleep(3);

mynewframe dm = new mynewframe(loadmyvideo);

AsyncResult m = dm.BeginInvoke(s, null, null);

p.Image = dm.EndInvoke(m);

}

}

else

{ p.Image = Image.FromStream(responsePic.GetResponseStream());}}}

Figure 4.3

So the difference from figure 4.2 we add the detect face phase that is found in loadvideo method

We need to notice to thing

1. framenum is acounter to make us ignore some frame in detection
2. we use 3 thing

first: deleget myframe that point to all method that take object as parameter and return bitmap

second: classification structure that contain two parameters image and picturebox to tell me this image belong to which picturebox

third: using asynchronization deleget Calling functions asynchronously causes the system to execute them in the background on a secondary thread while the calling function continues to do other work with lock to prevent any another thread for doing same code at same time .

Now:we move to method of detection loadvideo

private Bitmap loadmyvideo(Object s)

{

classification n = (classification)s;

currentFrame= new Image<Bgr, Byte>((Bitmap)n.myimg);

if (currentFrame != null)

{

Image<Gray, byte> Grayframe = currentFrame.Convert<Gray, byte>();

var faces = Grayframe.DetectHaarCascade(Haar, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

foreach (var face in faces)

{

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

}

}

return currentFrame.Bitmap;

}

Figure 4.5

so we convert the image in picture box to gray image the reason is clear that’s because I need not to consider color or light or any RGB when I want to move to third stage so if app found face using currentframe.Draw to draw rectangle on face that found then return new image with this rectangle on face.

### **4.2.3 Third stage:** [**Facial recognition**](https://en.wikipedia.org/wiki/Facial_recognition_system)

### In this stage we need to understand what is training set here? How we make in app? So training set collection of images that store and we use to compare between it and between images we get from video and here we use mysql database that store name and image in table called face and our app introduce this features when u press to help result in figure 4.6

### 

### Figure 4.6 after u add start we will now move to add face with name look at figure 4.7

### 

### Figure 4.7

### So you need to add name and start saving and this the most important thing

### All what we do we will add something in method on figure 4.4

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

try

{

byte[] bArr = imgToByteConverter(result.Bitmap);

insert\_img(bArr,textBox2.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

### }

### Figure 4.8

### We must convert image toarry of byte then store it by using this method below

public static byte[] imgToByteConverter(Image inImg)

{

ImageConverter imgCon = new ImageConverter();

return (byte[])imgCon.ConvertTo(inImg, typeof(byte[]));

### }

### And then we insert it in database using method insert\_img

### public void insert\_img(byte[] x,String name)

### {MySqlConnection con = new MySqlConnection("server=localhost;database=test;uid=root;pwd=root");

### con.Open();

byte [] img =x;

MySqlCommand cmd;

cmd = new MySqlCommand("insert into face values(@p1,@p2)", con);

cmd.Parameters.Add("@p1", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("@p2", MySqlDbType.Blob);

cmd.Parameters["@p1"].Value = name;

cmd.Parameters["@p2"].Value = img;

cmd.ExecuteNonQuery();

con.Close();

### }

### So after we store images we need to retrieve all of them in form1.cs to start compare so on load

con.Open();

cmd = new MySqlCommand("select \* from face", con);

MySqlDataReader de = cmd.ExecuteReader();

while (de.Read())

{

byte[] x = (byte[])de["image"];

MemoryStream xx = new MemoryStream(x);

Bitmap myimg= (Bitmap)Image.FromStream(xx);

trainingImages.Add(new Image<Gray, byte>(myimg));

labels.Add(de["name"].ToString());

counter++;

}

con.Close();

}

### Which labels list of names and trainingImages are list of face images

### Here most important question coming which features we compare between image we have now and the images in database and what is algorithm first we use PCA (Principle Component Analysis)

foreach (MCvAvgComp f in faces)

{

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

{

MCvTermCriteria termCrit = new MCvTermCriteria(counter, .001);

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 2000, ref termCrit);

name = recognizer.Recognize(result);

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

if (name.Equals("unknown"))

{

System.Media.SoundPlayer player = new System.Media.SoundPlayer(@"alarm.wav");

player.Play();

}

### }

### EigenObjectRecognizer is avery important class that will make all stages of pca algorithm which are

* Stage 1: Subtract the Mean of the data from each variable (our adjusted data)
* Stage 2: Calculate and form a covariance Matrix
* Stage 3: Calculate Eigenvectors and Eigenvalues from the covariance Matrix
* Stage 4: Chose a Feature Vector (a fancy name for a matrix of vectors)
* Stage 5: Multiply the transposed Feature Vectors by the transposed adjusted data

So if name = recognizer.Recognize(result); will calculate engine distance of all images and engine distance for my image to see which best one if name return unknown that mean not matched alarm will start

**4.3 Full implementation**

**Form 1**

using System;

using System.Drawing;

using System.Windows.Forms;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.CvEnum;

using System.Net;

using System.ComponentModel;

using System.IO;

using System.Threading;

using System.Collections.Generic;

using System.Threading;

using MySql.Data.MySqlClient;

namespace WindowsFormsApplication6

{

struct classification

{

public PictureBox container;

public Image myimg;

public classification(PictureBox a, Image b)

{

container = a;

myimg = b;

}

};

public partial class Form1 : Form

{

MySqlConnection con;

MySqlCommand cmd;

DateTime thisDay = DateTime.Today;

String date1;

String time1,h,m,s;

Image<Gray, byte> result;

public delegate Bitmap mynewframe(object b);

HaarCascade Haar;

HaarCascade Haar2;

HaarCascade Haar3;

Image<Bgr, Byte> currentFrame;

int a, b, c, d;

Thread th1, th2, th3, th4;

private static string h1,h2,h3,h4;

bool isvideosrc = false;

bool s1, s2, s3, s4;

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

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

int counter=0;

MCvFont font = new MCvFont(FONT.CV\_FONT\_HERSHEY\_TRIPLEX, 1.5d, 1.5d);

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

{

addfaces f = new addfaces();

f.Show();

this.Hide();

}

private void label1\_Click(object sender, EventArgs e)

{

}

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

con = new MySqlConnection("server=localhost;database=test;uid=root;pwd=root");

this.FormClosed += new FormClosedEventHandler(f\_FormClosed);

label1.Visible = false;

label2.Visible = false;

label3.Visible = false;

label4.Visible = false;

try

{

Haar = new HaarCascade("haarcascade\_frontalface\_alt\_tree.xml");

Haar2 = new HaarCascade("haarcascade\_frontalface\_default.xml");

Haar3 = new HaarCascade("haarcascade\_profilefaced.xml");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

load\_databases();

}

private void load\_databases()

{

con.Open();

cmd = new MySqlCommand("select \* from face", con);

MySqlDataReader de = cmd.ExecuteReader();

while (de.Read())

{

byte[] x = (byte[])de["image"];

MemoryStream xx = new MemoryStream(x);

Bitmap myimg= (Bitmap)Image.FromStream(xx);

trainingImages.Add(new Image<Gray, byte>(myimg));

labels.Add(de["name"].ToString());

counter++;

}

con.Close();

}

void repeat1(PictureBox p, string url,int framenum)

{

while (true)

{

framenum++;

WebRequest requestPic = WebRequest.Create(url);

WebResponse responsePic = requestPic.GetResponse();

if (framenum % 3 == 0)

{

classification s = new classification(p, Image.FromStream(responsePic.GetResponseStream()));

lock (this)

{

Thread.Sleep(3);

mynewframe dm = new mynewframe(loadmyvideo);

IAsyncResult m = dm.BeginInvoke(s, null, null);

p.Image = dm.EndInvoke(m);

}

}

else

{

p.Image = Image.FromStream(responsePic.GetResponseStream());

}

}

}

private Bitmap loadmyvideo(Object s)

{

classification n = (classification)s;

currentFrame= new Image<Bgr, Byte>((Bitmap)n.myimg);

if (currentFrame != null)

{

Image<Gray, byte> Grayframe = currentFrame.Convert<Gray, byte>();//.Flip(Emgu.CV.CvEnum.FLIP.HORIZONTAL);

var faces = Grayframe.DetectHaarCascade(Haar, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

var faces2 = Grayframe.DetectHaarCascade(Haar2, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

var faces3 = Grayframe.DetectHaarCascade(Haar3, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

bool x = false,y=false;

foreach (var face in faces)

{

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

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

x = true;

}

if (x)

{

String name = "";

foreach (MCvAvgComp f in faces)

{

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

{

MCvTermCriteria termCrit = new MCvTermCriteria(counter, .001);

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 2000, ref termCrit);

name = recognizer.Recognize(result);

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

if (name.Equals("unknown"))

{

System.Media.SoundPlayer player = new System.Media.SoundPlayer(@"alarm.wav");

player.Play();

}

}

}

}

if (!x)

{

foreach (var face in faces2)

{

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

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

y = true;

}

if (y)

{

String name1 = "";

foreach (MCvAvgComp f in faces2)

{

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

{

MCvTermCriteria termCrit = new MCvTermCriteria(counter, .001);

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 2000, ref termCrit);

name1 = recognizer.Recognize(result);

currentFrame.Draw(name1, ref font, new Point(f.rect.X - 2, f.rect.Y - 2), new Bgr(Color.GreenYellow));

if (name1.Equals("unknown"))

{

System.Media.SoundPlayer player = new System.Media.SoundPlayer(@"alarm.wav");

player.Play();

}

}

}

}

bool z = false;

if (!y)

{

foreach (var face in faces3)

{

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

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

z = true;

}

}

if (z)

{

String name = "";

foreach (MCvAvgComp f in faces3)

{

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

{

MCvTermCriteria termCrit = new MCvTermCriteria(counter, .001);

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 2000, ref termCrit);

name = recognizer.Recognize(result);

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

if (name.Equals("unknown"))

{

System.Media.SoundPlayer player = new System.Media.SoundPlayer(@"alarm.wav");

player.Play();

}

}

}

}

}

}

return currentFrame.Bitmap;

}

public static byte[] imgToByteConverter(Image inImg)

{

ImageConverter imgCon = new ImageConverter();

return (byte[])imgCon.ConvertTo(inImg, typeof(byte[]));

}

private void timer1\_Tick(object sender, EventArgs e)

{

date1 = Convert.ToString(thisDay.Day + "/" + thisDay.Month + "/" + thisDay.Year);

h = Convert.ToString(DateTime.Now.Hour);

m= Convert.ToString(DateTime.Now.Minute);

s = Convert.ToString(DateTime.Now.Second);

time1 = h + ":" + m + ":" + s;

label1.Text = "day:"+date1 +" time:"+time1;

label2.Text = "day:" + date1 + " time:" + time1;

label3.Text = "day:" + date1 + " time:" + time1;

label4.Text = "day:" + date1 + " time:" + time1;

}

private void button1\_Click(object sender, EventArgs e)

{

h1 = "http://" + textBox1.Text + ":8080/shot.jpg";

h2 = "http://" + textBox2.Text + ":8080/shot.jpg";

h3 = "http://" + textBox3.Text + ":8080/shot.jpg";

h4 = "http://" + textBox4.Text + ":8080/shot.jpg";

if (textBox1.Text != "")

{

isvideosrc = true;

s1 = true;

a = 1;

// start load frames whuch h1 is url for ip webcam1

th1 = new Thread(() => repeat1(pictureBox1, h1,a));

th1.Start();

textBox1.Visible = false;

label1.Visible = true;

}

if (textBox2.Text != "")

{

isvideosrc = true;

s2 = true;

b = 1;

th2 = new Thread(() => repeat1(pictureBox2, h2,b));

th2.Start();

textBox2.Visible = false;

label2.Visible = true;

}

if (textBox3.Text != "")

{

isvideosrc = true;

s3 = true;

c = 1;

th3 = new Thread(() => repeat1(pictureBox3, h3,c));

th3.Start();

textBox3.Visible = false;

label3.Visible = true;

}

if (textBox4.Text != "")

{

isvideosrc = true;

s4 = true;

d = 1;

th4 = new Thread(() => repeat1(pictureBox4, h4,d));

th4.Start();

textBox4.Visible = false;

label4.Visible = true;

}

if (!isvideosrc)

{

MessageBox.Show("please enter ip for camera");

}

}

private void f\_FormClosed(object sender, EventArgs e)

{

MessageBox.Show("thank u for using our app");

if (s1)

{

th1.Abort();

textBox1.Visible = true;

textBox1.Text = null;

pictureBox1.Image = null;

label1.Visible = false;

}

if (s2)

{

th2.Abort();

textBox2.Visible = true;

textBox2.Text = null;

pictureBox2.Image = null;

label2.Visible = false;

}

if (s3)

{

th3.Abort();

textBox3.Visible = true;

textBox3.Text = null;

pictureBox3.Image = null;

label3.Visible = false;

}

if (s4)

{

th4.Abort();

textBox4.Text = null;

textBox4.Visible = true;

pictureBox4.Image = null;

label4.Visible = false;

}

}

}

}

**Add face form**

using System;

using System.Drawing;

using System.Windows.Forms;

using Emgu.CV;

using Emgu.CV.Structure;

using Emgu.CV.CvEnum;

using System.Net;

using System.ComponentModel;

using System.IO;

using System.Threading;

using System.Collections.Generic;

using System.Threading;

using MySql.Data.MySqlClient;

namespace WindowsFormsApplication6

{

public partial class addfaces : Form

{

private static string h1;

Thread th;

public delegate Bitmap mynewframe(object b);

Image<Bgr, Byte> currentFrame;

bool startthread;

HaarCascade Haar;

HaarCascade Haar2;

HaarCascade Haar3;

int framenum = 1;

List<Image> trainingset;

bool startsaving = false;

Image<Gray, byte> result;

public addfaces()

{

InitializeComponent();

}

private void f\_FormClosed(object sender, EventArgs e)

{

if (startthread == true)

th.Abort();

Application.Exit();

}

private void addfaces\_Load(object sender, EventArgs e)

{

button4.Visible = false;

label1.Visible = false;

textBox2.Visible = false;

this.FormClosed += new FormClosedEventHandler(f\_FormClosed);

button1.Visible = false;

button2.Visible = false;

try

{

Haar = new HaarCascade("haarcascade\_frontalface\_alt\_tree.xml");

Haar2 = new HaarCascade("haarcascade\_frontalface\_default.xml");

Haar3 = new HaarCascade("haarcascade\_profilefaced.xml");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

void repeat()

{

while (true)

{

framenum++;

WebRequest requestPic = WebRequest.Create(h1);

WebResponse responsePic = requestPic.GetResponse();

if (framenum % 3 == 0)

{

mynewframe dm = new mynewframe(loading);

IAsyncResult m = dm.BeginInvoke(Image.FromStream(responsePic.GetResponseStream()), null, null);

pictureBox1.Image = dm.EndInvoke(m);

}

else

{

pictureBox1.Image = Image.FromStream(responsePic.GetResponseStream());

}

}

}

private Bitmap loading(object e)

{

Image myimg = (Image)e;

currentFrame = new Image<Bgr, Byte>((Bitmap)myimg);

if (currentFrame != null)

{

Image<Gray, byte> Grayframe = currentFrame.Convert<Gray, byte>();//.Flip(Emgu.CV.CvEnum.FLIP.VERTICAL);

var faces = Grayframe.DetectHaarCascade(Haar, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

var faces2 = Grayframe.DetectHaarCascade(Haar2, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

var faces3 = Grayframe.DetectHaarCascade(Haar3, 1.2, 8, HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING, new Size(pictureBox1.Height / 10, pictureBox1.Width / 10))[0];

Bitmap bmpinput = Grayframe.ToBitmap();

bool x = false, y = false;

foreach (var face in faces)

{

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

if (startsaving)

{

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

pictureBox2.Image = result.Bitmap;

try

{

byte[] bArr = imgToByteConverter(result.Bitmap);

insert\_img(bArr,textBox2.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

x = true;

}

if (!x)

{

foreach (var face in faces2)

{

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

y = true;

if (startsaving)

{

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

pictureBox2.Image = result.Bitmap;

try

{

byte[] bArr = imgToByteConverter(result.Bitmap);

insert\_img(bArr,textBox2.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

if (!y)

{

foreach (var face in faces3)

{

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

if (startsaving)

{

result = currentFrame.Copy(face.rect).Convert<Gray, byte>().Resize(50, 50, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

pictureBox2.Image = result.Bitmap;

try

{

byte[] bArr = imgToByteConverter(result.Bitmap);

insert\_img(bArr,textBox2.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

}

}

}

return currentFrame.Bitmap;

}

public static byte[] imgToByteConverter(Image inImg)

{

ImageConverter imgCon = new ImageConverter();

return (byte[])imgCon.ConvertTo(inImg, typeof(byte[]));

}

private void button3\_Click(object sender, EventArgs e)

{

h1 = "http://" + textBox1.Text + ":8080/shot.jpg";

th = new Thread(() => repeat());

th.Start();

startthread = true;

button1.Visible = true;

button2.Visible = true;

textBox1.Visible = false;

button3.Visible = false;

button4.Visible = true;

label1.Visible = true;

textBox2.Visible = true;

}

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

{

if (startthread == true)

th.Abort();

Form1 f = new Form1();

f.Show();

this.Dispose();

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox2.Text =="")

{

MessageBox.Show("please enter name of person");

}

else

{

startsaving = true;

trainingset = new List<Image>();

}

}

public void insert\_img(byte[] x,String name)

{

MySqlConnection con = new MySqlConnection("server=localhost;database=test;uid=root;pwd=root");

con.Open();

byte [] img =x;

MySqlCommand cmd;

cmd = new MySqlCommand("insert into face values(@p1,@p2)", con);

cmd.Parameters.Add("@p1", MySqlDbType.VarChar, 255);

cmd.Parameters.Add("@p2", MySqlDbType.Blob);

cmd.Parameters["@p1"].Value = name;

cmd.Parameters["@p2"].Value = img;

cmd.ExecuteNonQuery();

con.Close();

}

private void button4\_Click(object sender, EventArgs e)

{

startsaving = false;

textBox2.Text = null;

MessageBox.Show("thank u u can save another persons");

}

private void button5\_Click(object sender, EventArgs e)

{

}

}

}

**class EigenObjectRecognizer**

using System;

using System.Diagnostics;

using Emgu.CV.Structure;

using Emgu.CV;

namespace WindowsFormsApplication6

{

/// <summary>

/// An object recognizer using PCA (Principle Components Analysis)

/// </summary>

[Serializable]

public class EigenObjectRecognizer

{

private Image<Gray, Single>[] \_eigenImages;

private Image<Gray, Single> \_avgImage;

private Matrix<float>[] \_eigenValues;

private string[] \_labels;

private double \_eigenDistanceThreshold;

/// <summary>

/// Get the eigen vectors that form the eigen space

/// </summary>

/// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you know what you are doing</remarks>

public Image<Gray, Single>[] EigenImages

{

get { return \_eigenImages; }

set { \_eigenImages = value; }

}

/// <summary>

/// Get or set the labels for the corresponding training image

/// </summary>

public String[] Labels

{

get { return \_labels; }

set { \_labels = value; }

}

/// <summary>

/// Get or set the eigen distance threshold.

/// The smaller the number, the more likely an examined image will be treated as unrecognized object.

/// Set it to a huge number (e.g. 5000) and the recognizer will always treated the examined image as one of the known object.

/// </summary>

public double EigenDistanceThreshold

{

get { return \_eigenDistanceThreshold; }

set { \_eigenDistanceThreshold = value; }

}

/// <summary>

/// Get the average Image.

/// </summary>

/// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you know what you are doing</remarks>

public Image<Gray, Single> AverageImage

{

get { return \_avgImage; }

set { \_avgImage = value; }

}

/// <summary>

/// Get the eigen values of each of the training image

/// </summary>

/// <remarks>The set method is primary used for deserialization, do not attemps to set it unless you know what you are doing</remarks>

public Matrix<float>[] EigenValues

{

get { return \_eigenValues; }

set { \_eigenValues = value; }

}

private EigenObjectRecognizer()

{

}

/// <summary>

/// Create an object recognizer using the specific tranning data and parameters, it will always return the most similar object

/// </summary>

/// <param name="images">The images used for training, each of them should be the same size. It's recommended the images are histogram normalized</param>

/// <param name="termCrit">The criteria for recognizer training</param>

public EigenObjectRecognizer(Image<Gray, Byte>[] images, ref MCvTermCriteria termCrit)

: this(images, GenerateLabels(images.Length), ref termCrit)

{

}

private static String[] GenerateLabels(int size)

{

String[] labels = new string[size];

for (int i = 0; i < size; i++)

labels[i] = i.ToString();

return labels;

}

/// <summary>

/// Create an object recognizer using the specific tranning data and parameters, it will always return the most similar object

/// </summary>

/// <param name="images">The images used for training, each of them should be the same size. It's recommended the images are histogram normalized</param>

/// <param name="labels">The labels corresponding to the images</param>

/// <param name="termCrit">The criteria for recognizer training</param>

public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, ref MCvTermCriteria termCrit)

: this(images, labels, 0, ref termCrit)

{

}

/// <summary>

/// Create an object recognizer using the specific tranning data and parameters

/// </summary>

/// <param name="images">The images used for training, each of them should be the same size. It's recommended the images are histogram normalized</param>

/// <param name="labels">The labels corresponding to the images</param>

/// <param name="eigenDistanceThreshold">

/// The eigen distance threshold, (0, ~1000].

/// The smaller the number, the more likely an examined image will be treated as unrecognized object.

/// If the threshold is &lt; 0, the recognizer will always treated the examined image as one of the known object.

/// </param>

/// <param name="termCrit">The criteria for recognizer training</param>

public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, double eigenDistanceThreshold, ref MCvTermCriteria termCrit)

{

Debug.Assert(images.Length == labels.Length, "The number of images should equals the number of labels");

Debug.Assert(eigenDistanceThreshold >= 0.0, "Eigen-distance threshold should always >= 0.0");

CalcEigenObjects(images, ref termCrit, out \_eigenImages, out \_avgImage);

/\*

\_avgImage.SerializationCompressionRatio = 9;

foreach (Image<Gray, Single> img in \_eigenImages)

//Set the compression ration to best compression. The serialized object can therefore save spaces

img.SerializationCompressionRatio = 9;

\*/

\_eigenValues = Array.ConvertAll<Image<Gray, Byte>, Matrix<float>>(images,

delegate (Image<Gray, Byte> img)

{

return new Matrix<float>(EigenDecomposite(img, \_eigenImages, \_avgImage));

});

\_labels = labels;

\_eigenDistanceThreshold = eigenDistanceThreshold;

}

#region static methods

/// <summary>

/// Caculate the eigen images for the specific traning image

/// </summary>

/// <param name="trainingImages">The images used for training </param>

/// <param name="termCrit">The criteria for tranning</param>

/// <param name="eigenImages">The resulting eigen images</param>

/// <param name="avg">The resulting average image</param>

public static void CalcEigenObjects(Image<Gray, Byte>[] trainingImages, ref MCvTermCriteria termCrit, out Image<Gray, Single>[] eigenImages, out Image<Gray, Single> avg)

{

int width = trainingImages[0].Width;

int height = trainingImages[0].Height;

IntPtr[] inObjs = Array.ConvertAll<Image<Gray, Byte>, IntPtr>(trainingImages, delegate (Image<Gray, Byte> img) { return img.Ptr; });

if (termCrit.max\_iter <= 0 || termCrit.max\_iter > trainingImages.Length)

termCrit.max\_iter = trainingImages.Length;

int maxEigenObjs = termCrit.max\_iter;

#region initialize eigen images

eigenImages = new Image<Gray, float>[maxEigenObjs];

for (int i = 0; i < eigenImages.Length; i++)

eigenImages[i] = new Image<Gray, float>(width, height);

IntPtr[] eigObjs = Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages, delegate (Image<Gray, Single> img) { return img.Ptr; });

#endregion

avg = new Image<Gray, Single>(width, height);

CvInvoke.cvCalcEigenObjects(

inObjs,

ref termCrit,

eigObjs,

null,

avg.Ptr);

}

/// <summary>

/// Decompose the image as eigen values, using the specific eigen vectors

/// </summary>

/// <param name="src">The image to be decomposed</param>

/// <param name="eigenImages">The eigen images</param>

/// <param name="avg">The average images</param>

/// <returns>Eigen values of the decomposed image</returns>

public static float[] EigenDecomposite(Image<Gray, Byte> src, Image<Gray, Single>[] eigenImages, Image<Gray, Single> avg)

{

return CvInvoke.cvEigenDecomposite(

src.Ptr,

Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages, delegate (Image<Gray, Single> img) { return img.Ptr; }),

avg.Ptr);

}

#endregion

/// <summary>

/// Given the eigen value, reconstruct the projected image

/// </summary>

/// <param name="eigenValue">The eigen values</param>

/// <returns>The projected image</returns>

public Image<Gray, Byte> EigenProjection(float[] eigenValue)

{

Image<Gray, Byte> res = new Image<Gray, byte>(\_avgImage.Width, \_avgImage.Height);

CvInvoke.cvEigenProjection(

Array.ConvertAll<Image<Gray, Single>, IntPtr>(\_eigenImages, delegate (Image<Gray, Single> img) { return img.Ptr; }),

eigenValue,

\_avgImage.Ptr,

res.Ptr);

return res;

}

/// <summary>

/// Get the Euclidean eigen-distance between <paramref name="image"/> and every other image in the database

/// </summary>

/// <param name="image">The image to be compared from the training images</param>

/// <returns>An array of eigen distance from every image in the training images</returns>

public float[] GetEigenDistances(Image<Gray, Byte> image)

{

using (Matrix<float> eigenValue = new Matrix<float>(EigenDecomposite(image, \_eigenImages, \_avgImage)))

return Array.ConvertAll<Matrix<float>, float>(\_eigenValues,

delegate (Matrix<float> eigenValueI)

{

return (float)CvInvoke.cvNorm(eigenValue.Ptr, eigenValueI.Ptr, Emgu.CV.CvEnum.NORM\_TYPE.CV\_L2, IntPtr.Zero);

});

}

/// <summary>

/// Given the <paramref name="image"/> to be examined, find in the database the most similar object, return the index and the eigen distance

/// </summary>

/// <param name="image">The image to be searched from the database</param>

/// <param name="index">The index of the most similar object</param>

/// <param name="eigenDistance">The eigen distance of the most similar object</param>

/// <param name="label">The label of the specific image</param>

public void FindMostSimilarObject(Image<Gray, Byte> image, out int index, out float eigenDistance, out String label)

{

float[] dist = GetEigenDistances(image);

index = 0;

eigenDistance = dist[0];

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

{

if (dist[i] < eigenDistance)

{

index = i;

eigenDistance = dist[i];

}

}

label = Labels[index];

}

/// <summary>

/// Try to recognize the image and return its label

/// </summary>

/// <param name="image">The image to be recognized</param>

/// <returns>

/// String.Empty, if not recognized;

/// Label of the corresponding image, otherwise

/// </returns>

public String Recognize(Image<Gray, Byte> image)

{

int index;

float eigenDistance;

String label;

FindMostSimilarObject(image, out index, out eigenDistance, out label);

return (\_eigenDistanceThreshold <= 0 || eigenDistance < \_eigenDistanceThreshold) ? \_labels[index] : "unknown";

}

}

}