Multiple Face Detection and Recognition in Real Time



[**Sergio Andrés Gutiérrez Rojas**](https://www.codeproject.com/script/Membership/View.aspx?mid=5558099)

**Rate me:**

https://www.codeproject.com/script/Ratings/Images/star-fill-lg.png

https://www.codeproject.com/script/Ratings/Images/star-fill-lg.png

https://www.codeproject.com/script/Ratings/Images/star-fill-lg.png

https://www.codeproject.com/script/Ratings/Images/star-fill-lg.png

https://www.codeproject.com/script/Ratings/Images/star-fill-lg.png

https://www.codeproject.com/script/Ratings/Images/star-empty-lg.png

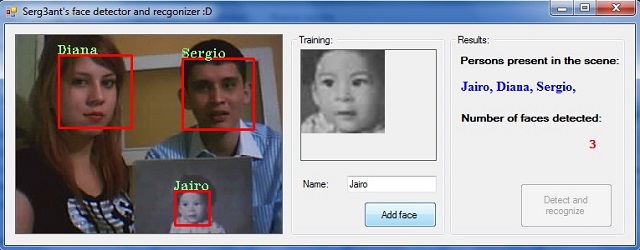
4.91/5 (353 votes)

5 Jan 2022[CPOL](http://www.codeproject.com/info/cpol10.aspx)10 min read

Face detection and recognition with support of multiples faces in the same scene and other interesting features using C# and EmguCV

In this article, I will show and explain the easiest way in which to implement a face detector and recognizer in real time for multiple persons using Principal Component Analysis (PCA) with eigenface for implementing it in multiple fields.

* [**Download FaceRecPro Optimized Version: source code, demo: All binaries references, algorithm enhanced - 7.4 MB**](https://www.codeproject.com/KB/Articles/FaceRecPro/FaceRecProOV.zip)
* [**Download source - 153.8 KB**](https://www.codeproject.com/KB/Articles/FaceRecPro/FaceRecProOV_Src.zip)
* [**Download demo - 2 MB**](https://www.codeproject.com/KB/Articles/FaceRecPro/FaceRecProOV_Demo.zip)



Introduction

The facial recognition has been a problem worked on around the world for many persons; this problem has emerged in multiple fields and sciences, especially in computer science, other fields that are very interested in this technology are: Mechatronic, Robotic, criminalistics, etc. In this article, I work on this interesting topic using EmguCV cross platform .NET wrapper to the Intel OpenCV image processing library and C# .NET, these libraries allow me to capture and process image of a capture device in real time. The main goal of this article is to show and explain the easiest way in which to implement a face detector and recognizer in real time for multiple persons using Principal Component Analysis (PCA) with eigenface for implementing it in multiple fields.

Background

Facial recognition is a computer application composed for complex algorithms that use mathematical and matricial techniques, these get the image in raster mode (digital format) and then process and compare pixel by pixel using different methods for obtaining faster and reliable results. Obviously, these results depend on the machine used to process this due to the huge computational power that these algorithms, functions and routines require, these are the most popular techniques used for solving this modern problem.

Techniques

**Traditional**

Some facial recognition algorithms identify faces by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face detection. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation. Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distill an image into values and comparing the values with templates to eliminate variances. Popular recognition algorithms include Principal Component Analysis with eigenface, Linear Discriminate Analysis, Elastic Bunch Graph Matching fisherface, the Hidden Markov model, and the neuronal motivated dynamic link matching.

Text taken from **[1]**

An example of EigenFaces:



Image taken from **[4]**

**3-D**

A newly emerging trend, claimed to achieve previously unseen accuracies, is three-dimensional face recognition. This technique uses 3-D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin. One advantage of 3-D facial recognition is that it is not affected by changes in lighting like other techniques. It can also identify a face from a range of viewing angles, including a profile view. Even a perfect 3D matching technique could be sensitive to expressions. For that goal, a group at the Technion applied tools from metric geometry to treat expressions as isometries.

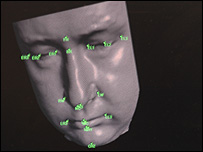


Image taken from **[2]**

**Skin Texture Analysis**

Another emerging trend uses the visual details of the skin, as captured in standard digital or scanned images. This technique, called skin texture analysis, turns the unique lines, patterns, and spots apparent in a person’s skin into a mathematical space Tests have shown that with the addition of skin texture analysis, performance in recognizing faces can increase 20 to 25 percent. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems.

Text taken from **[1]**

EmguCV



Emgu CV is a cross platform .NET wrapper to the Intel OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython, etc. The wrapper can be compiled in Mono and run on Linux / Mac OS X.

Text taken from **[3]**

In my own words, EmguCV is an awesome Wrapper, this allows to make very interesting things and tasks of computer vision. This library set lets do an unlimited amount of wonderful projects in this field, EmguCV have many functions that let us work with CPU and GPU increases the performance dramatically with the latest mentioned.

This awesome SW project let work and do:

* Optical Character Recognition(OCR)
* Face Detection
* Pedestrian Detection
* Kinect projects
* 3D reconstruction
* SURF feature detector... between many others interesting tasks

EmguCV Basics: How I Start to Work?

If you have never worked with this wrapper, you want see how to add references to project or solve problems, take a look at this good article/tutorial by **C\_Johnson**:

* [Creating Your First EMGU Image Processing Project](http://www.codeproject.com/Articles/257502/Creating-Your-First-EMGU-Image-Processing-Project)

Another interesting Web/blog with multiple tutorials to star with emguCV, image processing, and face recognition by **mehwish87** is:

* [EmguCV and basic image processing tutorials](http://fewtutorials.bravesites.com/tutorials)

Using the Code

First, declare all variables and important objects to use:

C#

Copy Code

*//Declaration of all variables, vectors and haarcascades*

Image<bgr,> currentFrame;

Capture grabber;

HaarCascade face;

HaarCascade eye;

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

Image<gray,> result, TrainedFace = null;

Image<gray,> gray = null;

List<image<gray,>> trainingImages = new List<image<gray,>>();

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

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

int ContTrain, NumLabels, t;

string name, names = null;

Then load the haarcascades for face detection, then I do a little “procedure” to load previous trained faces and labels for each image stored previously:

C#

Shrink ▲   Copy Code

*//Load haarcascades for face detection*

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

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

try

{

*//Load of previous trained faces and labels for each image*

string Labelsinfo =

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

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

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

ContTrain = NumLabels;

string LoadFaces;

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

{

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

trainingImages.Add(new Image<gray,>

(Application.StartupPath + "/TrainedFaces/" + LoadFaces));

labels.Add(Labels[tf]);

}

}

catch(Exception e)

{

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

MessageBox.Show("Nothing in binary database,

please add at least a face", "Trained faces load",

MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

Initialize the capture device, and FrameGrabber event that performs the detection and process of images for each frame captured:

C#

Copy Code

grabber = new Capture();

grabber.QueryFrame();

*//Initialize the FrameGraber event*

Application.Idle += new EventHandler(FrameGrabber);

button1.Enabled = false;

Passing to FrameGrabber event (main part of prototype), we use the most important methods and objects: DetectHaarCascade and EigenObjectRecognizer and perform operations for each face detected in one frame:

C#

Shrink ▲   Copy Code

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

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

new Size(20, 20));

*//Action for each element detected*

foreach (MCvAvgComp f in facesDetected[0])

{

t = t + 1;

result = currentFrame.Copy(f.rect).Convert<gray,>().

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

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

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

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

{

*//TermCriteria for face recognition with*

*//numbers of trained images like maxIteration*

MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

*//Eigen face recognizer*

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

5000,

ref termCrit);

name = recognizer.Recognize(result);

*//Draw the label for each face detected and recognized*

currentFrame.Draw(name, ref font,

new Point(f.rect.X - 2, f.rect.Y - 2), new Bgr(Color.LightGreen));

}

}

**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 have 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 many edges and thus cannot 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).

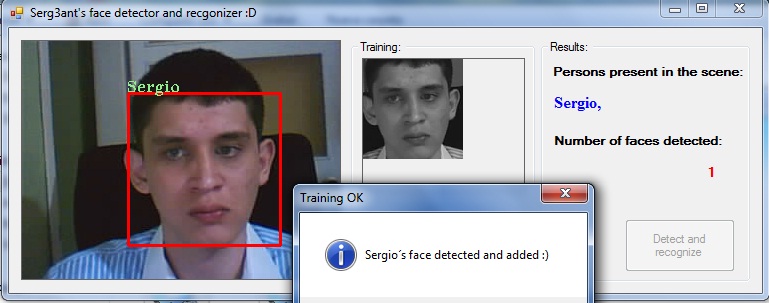
How to Train the Prototype?

I do this part in the easiest way possible, the prototype detects faces constantly (Each frame) and you can add this detected face in the image database with one label respectably, the face trained image will show in the imageBoxFrameGrabber and the process will be finished!!

**Keep in mind**: The face recognition algorithms based in PCA (Principal Component Analysis) do multiple comparisons and matches between a face detected and the trained images stored in binary database for this reason, and for improving the accuracy of recognition, you should add several images of the same person in different angles, po

sitions and luminance conditions, this training makes this prototype solid and very accurate.

Example:



Code of training button (This performs the adding of training faces and labels for each):

C#

Shrink ▲   Copy Code

try

{

*//Trained face counter*

ContTrain = ContTrain + 1;

*//Get a gray frame from capture device*

gray = grabber.QueryGrayFrame().Resize

(320, 240, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

*//Face Detector*

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

10,

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

new Size(20, 20));

*//Action for each element detected*

foreach (MCvAvgComp f in facesDetected[0])

{

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

break;

}

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

*//test image with cubic interpolation type method*

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

trainingImages.Add(TrainedFace);

labels.Add(textBox1.Text);

*//Show face added in gray scale*

imageBox1.Image = TrainedFace;

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

File.WriteAllText(Application.StartupPath +

"/TrainedFaces/TrainedLabels.txt",

trainingImages.ToArray().Length.ToString() + "%");

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

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

{

trainingImages.ToArray()[i - 1].Save

(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");

File.AppendAllText(Application.StartupPath +

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

}

MessageBox.Show(textBox1.Text + "´s face detected and added :)",

"Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

MessageBox.Show("Enable the face detection first",

"Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

'Enable the face detection first' warning?

For add a face to the model you need initiate the camera and face recognition engine first, you only should press the "Detect and recognize" button.

when the camera shows your face on live image you can add and train the model.

How to Improve the Recognition?

The default parameters (scale\_factor=1.1, min\_neighbors=3, flags=0) are tuned for accurate yet slow object detection.

Also, you may modify the size for a big value, modify this in the code:

C#

Copy Code

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.1,

3,

0,

new Size(20, 20));

Additionally, modify the third param to **2500** or **3000** instead 5000, this modification makes the EigenObjectRecognizer be more strict/accurate.

C#

Copy Code

*//Eigen face recognizer*

EigenObjectRecognizer recognizer = new EigenObjectRecognizer(

trainingImages.ToArray(),

labels.ToArray(),

5000,

ref termCrit);

How to Improve the Performance for Slower CPUs?

All image processing algorithms demand many computational power, in this case, the internals process carried on for the CPU with this sw prototype are so hard for slower or monocore CPUS, the easy way for improving the performance of this demo is modify the

parameters that use the DetectHaarCascade method, these allow to decrement the

number of iterations, critic sections and comparisons of the real time image captured for the

Webcam improving notoriously the application performance.

Keep in mind: reduce the values of these parameters will affect the efficiency of recognition Algorithms.

First Option

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> (for example, ~1/4 to 1/16 of the image area in case of video conferencing).

Also, you may modify the Minsize parameter for a big value.

C#

Copy Code

*// DetectHaarCascade Config for optimal performance*

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(

face,

1.2,

2,

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

new Size(20, 20));

Second Option

Get a “thumbnail”or resize the original image capture for reducing the time of processing in the FrameGrabber method modify the size values for a minor size (originally is 320x240).

Example:

C#

Copy Code

*//Get the current frame form capture device*

currentFrame = grabber.QueryFrame().Resize

(260, 200, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

Remember to do the same in the Training button:

C#

Copy Code

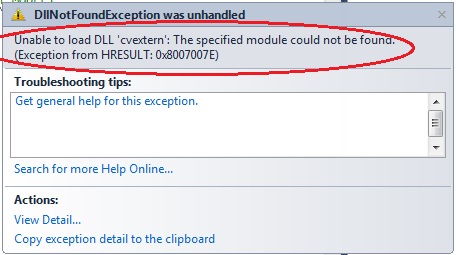
gray = grabber.QueryGrayFrame().Resize(260, 200, Emgu.CV.CvEnum.INTER.CV\_INTER\_CUBIC);

The Prototype Does Not Show the Strangers as Unknown, Why?

Is a limitation of Eigenface approach, isn't error, use LBPH algorithm instead (have pros and shortcomings) available on EmguCV 2.4.2 and above.

What Do You Need to Run/Use this Project without Errors?

For avoid errors like this:



1. First, download and decompress *FaceRecPro\_Demo.zip* it has “external” OS libraries(DLLs) used for some OpenCV functions, after copying these DLLs in *C:/Windows/System32* or in *bin* folder of this project.
2. Then download EmguCV(Wrapper of OpenCV for C#) at <http://sourceforge.net/projects/emgucv/files/>, install it and then go to *C:\Emgu\emgucv-windows-x86 2.2.1.1150\bin* folder and copy: opencv\_calib3d220, opencv\_contrib220, opencv\_core220, opencv\_features2d220, opencv\_ffmpeg220, opencv\_flann220, opencv\_gpu220, opencv\_highgui220, opencv\_imgproc220, opencv\_legacy220, opencv\_ml220, opencv\_objdetect220 and opencv\_video220 in *C:/Windows/System32* or in *bin* folder of this project.

Remember these DLLs are OpenCV Libraries and are necessary to run any project that uses EgmuCV.

**This is the easiest way to run the demo or project, find and copy these files to folder demo or bin folder (if you want "run" the source code)**

*opencv\_calib3d220.dll*, *opencv\_contrib220.dll*, *opencv\_core220.dll*, *opencv\_features2d220.dll*, *opencv\_ffmpeg220.dll*, *opencv\_flann220.dll*, *opencv\_gpu220.dll*, *opencv\_highgui220.dll*, *opencv\_imgproc220.dll*, *opencv\_legacy220.dll*, *opencv\_ml220.dll*, *opencv\_objdetect220.dll* and *opencv\_video220.dll*(DLLs from OpenCV Content in EmguCV download) and *Emgu.CV.dll* *Emgu.CV.UI.dll Emgu.Util.dll cv110.dll cvaux110.dll cvextern.dll cxcore110.dll highgui110.dll* (Content in demo download zip).

Keep in mind: you can copy these files in *Windows/System32/* folder and forget your problems of dependencies for this and other project that use Emgu or openCV (*Emgu.CV.dll*, *Emgu.CV.UI.dll*, *Emgu.Util.dll* SHOULD always go in the *bin* or *.exe* folder)**or download all ready projects (Optimized version) and files here**:

* [**Download FaceRecPro Optimized Version: source code, demo: All binaries references, algorithm enhanced - 6.5 MB**](http://www.mediafire.com/?q1g00owxbcaxyow)

Points of Interest

I had many problems with vectors use with EmguCV, for this reason, I had learnt how use list of components like vectors and it really works for my project.

I learn many of image processing, PCA and EigenFaces and to optimize the code, due to the huge demand of resources for part of artificial vision algorithms.

This project idea emerges after see an Iron man scene…XD

What Can You Do With This Article and OpenCV?

Something like this....Big Grin | <img src=

[**http://youtu.be/T4z11lIKnDU**](http://youtu.be/T4z11lIKnDU)

Books About It

The official reference book for OpenCV is: "Learning OpenCV: Computer Vision with the OpenCV Library", from O'Reilly (2008).

There are hundreds of other good books for computer vision that are not specific to OpenCV:

* "Computer Vision: A Modern Approach" by Forsyth and Ponce (2002)
* "Computer Vision: Algorithms and Applications" by Szeliski (2011)
* "Digital Image Processing" by Gonzalez and Woods (2001)
* "The Essential Guide to Image Processing", by Bovik (2009)
* "Computer Vision and Applications: A Guide for Students and Practitioners", by Jähne and Haußecker (2000)
* HIPR2 Image Processing Worksheets (simple explanations of many computer vision topics)

References

* [1] http://en.wikipedia.org/wiki/Facial\_recognition\_system
* [2] "Firms point to biometric future" By Dominic Bailey, BBC News, at Biometrics 2006
* [3] http://www.emgu.com/wiki/index.php/Main\_Page
* [4] http://www.shervinemami.info/faceRecognition.html

License

This article and prototype is only for academic purposes, not use for commercial or public releases without author permissions.

History

* 11-07-2011: Initial release
* 28-01-2012: Final release (optimized version)

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License (CPOL)](http://www.codeproject.com/info/cpol10.aspx)

Share

About the Author



[**Sergio Andrés Gutiérrez Rojas**](https://www.codeproject.com/Members/Sergio-A-Gutierrez)

|  |  |  |
| --- | --- | --- |
|  | Software Developer (Senior) Learn Production AI |  |
| Colombia Colombia |

MSc. Artificial intelligence  
Systems engineer  
Senior software developer  
Networks technologist  
Artificial vision, AI and robotic lover  
Technology enthusiast  
Gamer  
  
Learn Production AI founder  
Motion Soft Co-founder  
MIA PC Friend Developer  
  
Take practical professional artificial vision courses (desktop and mobile) in Learn Production AI: https://www.learnproductionai.tech