

OPENFINDER

OPENFINDER



**Batch 2013-2016**

**Project Advisors**

**Mr. Saeed**

**Submitted By**

|  |  |
| --- | --- |
| Syed Saqib Abbas Rizvi | BS Seat # EP-1349112 |
| Sharukh Ahmed Khan | BS Seat # EP-1349092 |
| Abdul BasitRana | BS Seat # EP-1349001 |

Department of Computer Science

University of Karachi

Karachi

Mar 20, 2017

-

**ACKNOWLEDGEMENT**

We truly acknowledge the cooperation and help make by sir Saeed, Department of Computer Science University of Karachi. He has been a constant source of guidance throughout the course of this project. We would also like to thank all of our teachers for their help and guidance in understanding each and everything .We are also thankful to our friends and families whose silent support led us to complete our project.

(Signed)

Syed Saqib Abbas Rizvi EP-1349112

Sharukh Ahmed Khan EP-1349092

Abdul BasitRana EP-1349001

**Abstract**

Our project is based on image processing “OpenFinder” is a source for every type of User. Who is able for search a particular face.Team has successfully developed the idea to find the person in multiple videos , the idea is approximately same as youtube finder , its works on local videos and find the person in parallel to all videos.Open finder is an open source java based project , having simple Graphical User Interface where user have a chance to select multiple videos, and image of a person. now it is very easily to search a face in a multiple videos , if user want to find all the videos related to any star he only want to select his /her image and click “GO”. Instantly he gets the result that a person’s image he selected was present in the video or not. OpenFinder works on multiple threads , it deals all the videos in separate threads.

.

**CERTIFICATE**

**UNIVERSITY OF KARACHI**

**Department of Computer Science**

Certified that the project work entitled **OPENFINDER** carried out by Mr. Syed Saqib Abbas Rizvi, Mr. Shahrukhahmed khan and Mr. Abdul BasitRana are bona fide students of Department of Computer Science, University of Karachi. They did BSCS Final Year Project in partial fulfillment for the award of **Bachelor of Computer Science** during the year 2016. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

**Name & Signature of the Supervisor**

**Name of the examiners                                                                                          Signature with date**

**List Of Figures**

figure 1.0 google image finder...........................................2

figure 1.1 PIPL...................................................................2

figure 1.3 screenshot of path folder......................................4

figure 2.0 flowchart of internal class working.....................5

figure 2.1 flowchart of face recognition..............................5

figure 2.2 splash screen page............................................6

figure 2.3 application main page before selecting

videos and pictures.............................................7

figure 2.4 application processing selected videos

and pictures.........................................................7

figure 2.5 application showing results..................................8

figure 2.6 screenshots of showing credit..............................8

figure 4.0 configuring open cv (i).........................................17

figure 4.1 configuring open cv (ii)........................................18

figure 4.2 configuring open cv (iii).......................................18

figure 4.3 configuring open cv (iv).......................................19

figure 4.4 configuring open cv (v)........................................19

figure 4.5 configuring open cv (vi).......................................20

figure 5.0 face features.......................................................21

figure 5.1 all face features..................................................21

figure 5.2 example of face features....................................22

figure 5.3 different types of face features...........................23

figure 6.0 eigen faces.........................................................26

**Table Of Content**

Acknowledgements i

Abstract ii

Certificate iii

Table of Content iv

1 Introduction 1

1.1 Overview 1

1.2 Significance Of the Project 1

1.2.1 Google Image Finder 1

1.2.2 PIPL 2

1.3 Description of the Project 3

1.3.1 Youtube Finder 3

1.3.2 Live Surveillance system 3

1.3.3 Achievements 3

1.3.4 OpenFinder 4

1.3.5 Root Directory Of Openfider 4

1.4 Project Scope 4

2 WorkFolws 5

2.1 Struture of OpenFinder 5

2.2 How To Use Openfinder 5

2.3 OpenFinderRequirements 8

2.4 Project Management 8

3 Image Processing 9

3.1 Purposes of Image Processing 9

3.2 Types of Image Processing 10

4 OpenCV 12

4.1 Defination 12

4.2 Comparison Matlab and OpenCV 12

4.3 Why OpenCV 15

4.4 Installation and Integeration Of OpenCV 17

5 Face Detection 21

5.1 Introduction 21

5.2 Haar-cascade Detection 22

6 Face Recognition 24

6.1 Introduction 24

6.2 The Principal Component Analysis Theory 24

6.3 PCA-based Object Recognition 25

6.4 Eigenfaces 25

6.5 Steps follow in a PCA for Digital Image 26

6.6 PCA-based Object Recognition 26

7 Bibliography 27

**Chapter # 1**

**Introduction**

**1.1 Overview**

Title of our project is “Open Finder”. Here we are providing ability to our customer to search a face in a video(s). Having simple Graphical User Interface where user have a chance to select multiple videos, and image of a person. now it is very easily to search a face in a multiple videos , if user want to find all the videos related to any star he only want to select his /her image and click “GO”. Instantly he gets the result that a person’s image he selected was present in the video or not.

This application may be use in many purpose but our main focus is on YouTube searcher , because in a Criminal tracing it is being used but for a YouTube it is being unable to search a face.

According to the need of new intelligence system. There must be a system which can find a person from anywhere. As we look in spy movies that there is a bluish screen where the hacker guy types something and they find the required person. In the reality it is not that simple. There is no ideal system which can track or find a person in a while, or from anywhere in the world

We have tried to target the person in a video.

It can be used like you have a picture of a person and you have to find it that is in the video or not?

The other software’s which are open source and are on the internet claims that they can find person over the globe by email, name, contact number and region.

**1.2 Significance of the Project**

**1.2.1 Google image finder**

In this figure we searched image in Google image search. It told us that it is related to wedding reception. We were expecting something different. System should tell user the name and other details to be mentioned. If the internet is so fast and vast spread so it should keep record of its users at least.

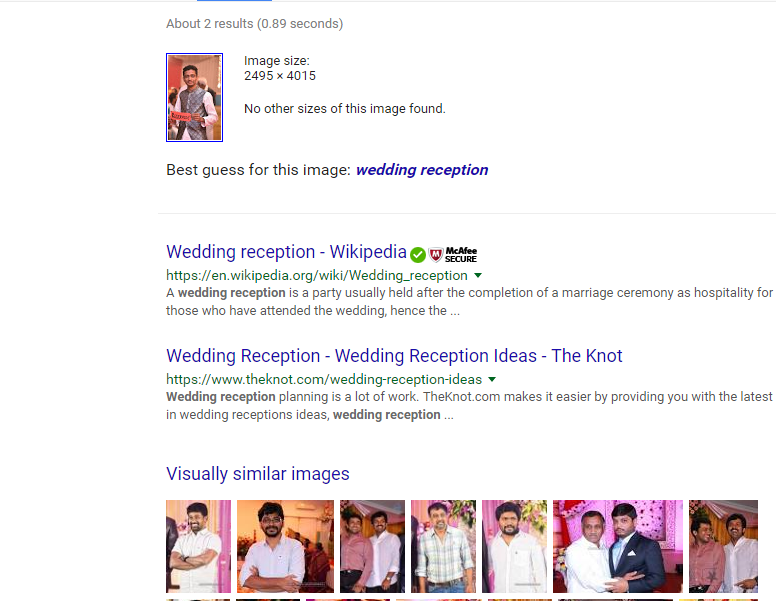


figure 1.0 google image finder

**1.2.2 PIPL**

It is also used for finding people over internet. Let’s have a look in this figure

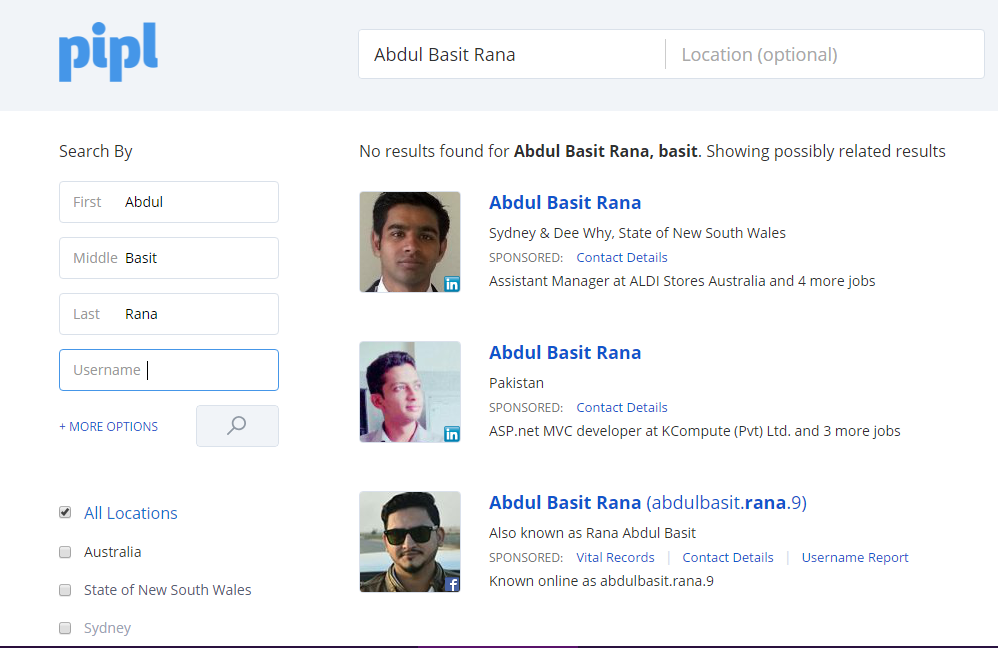


figure 1.1 PIPL page

Here the entered query was "ABDUL BASIT RANA”, one of our group member’sname

And you can see the result. It provided his LinkedIn profile.

**1.3 Description of the Project**

OpenFinder has multiple versions during its development , all are given below

**1.3.1 Youtube finder**

The idea and proposal of this version has to find the the person in youtube videos , the idea was there will be an another part application which fetch maximum no of youtube videos and store in our database , the second part is to analyze all of them and stores the calculated result in database ,Whenever user come to find a person , openfinder will give the match result from database.

**1.3.2 Live Surveillance system**

This version of open finder was surveillance system , the concept of project was to keep complete track of recognized person in a specific area with the help of multiple cameras connectd to a single server , all cameras are synced with each other , they provides signal to neighbour cameras to keep track of a person .User has to put the image of wanted person in system , as this person comes in the area , cameras recognize that person and keep track of that person

**1.3.3 Acheivements**

Team has developed the client server application for live stream , but this idea not contiued because need of high performenced computers and GPUs.

**1.3.4 OpenFinder**

Team has successfully developed the idea to find the person in multiple videos , the idea is approximately same as youtube finder , its works on local videos and find the person in parallel to all videos.

Open finder is an open source java based project , having simple Graphical User Interface where user have a chance to select multiple videos, and image of a person. now it is very easily to search a face in a multiple videos , if user want to find all the videos related to any star he only want to select his /her image and click “GO”. Instantly he gets the result that a person’s image he selected was present in the video or not.OpenFinder works on multiple threads , it deals all the videos in separate threads.

**1.3.5 Root Director of OpenFinder**

Bin-> includes all the library of java

Details-> includes the detail images view of recognized person

Faces-> it is the parents directory which hold all the faces came in the video

Lib-> it includes colt library which use for linear algebra operations.

Slider-> it includes slider images

Splash-> splash screen image

Src->includes all source code file of openfinder.

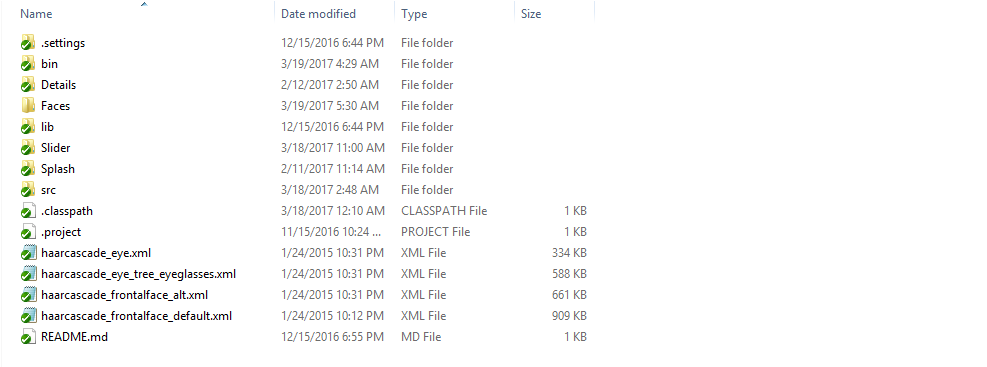


figure 1.2 screenshot of path folder page

**1.4 Project Scope**

As it is final year project , openfinder team decided to make something new and provide a solution to problem which community faces. As described above multiple application are here which can use to find a person ,but no one find the person in video ,but “OpenFinder does”

**Chapter # 2**

**Workflow**

**2.1 Structure Of OpenFinder**

Classes involved in implementation.

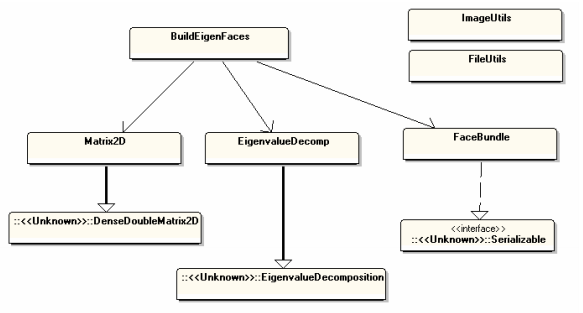


figure 2.0 flowchart of internal class working page 5

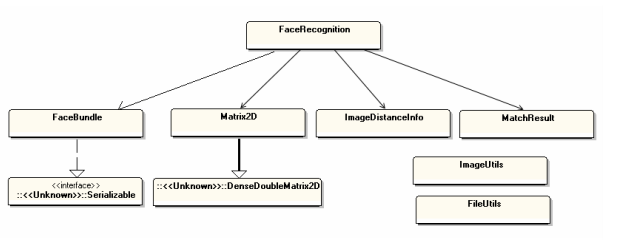


figure 2.1 flowchart of face recognition

**2.2 How To Use OpenFinder**

OpenFinder using two important concepts of computer graphics

Face detection

Face recognition

Face detection is done by openCV, while face recognition is done by PCA (eigen faces).Steps included

This is basically the start screen of OpenCV

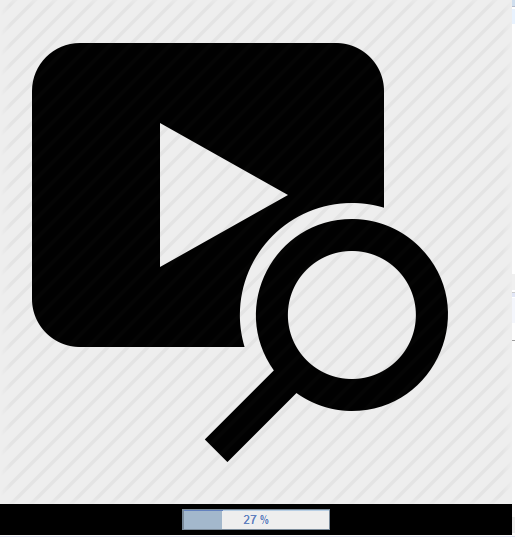


figure 2.2 splash screen

This is the basic UI consisting of image slider and user have the ability of selecting multiple videos.

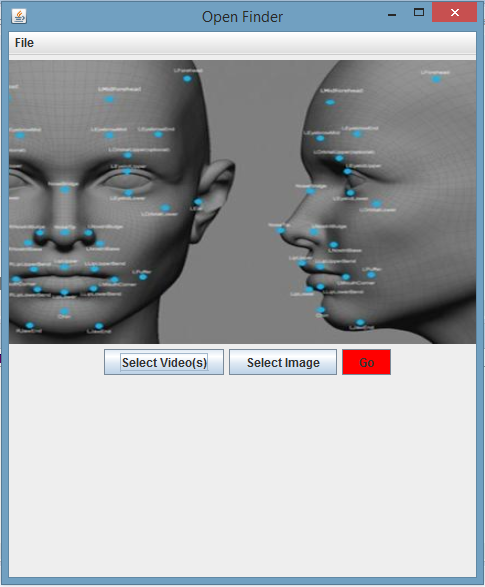


figure 2.3 application main page before selecting videos and pictures

Here we are providing a screenshot of multiple videos selected having started processing.

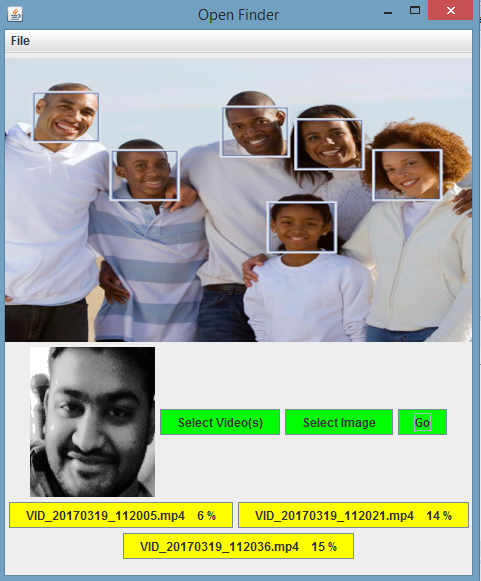


figure 2.4 application processing selected videos and pictures

Here this is a result screen where images is not found in 2 videos but found in 1st video.

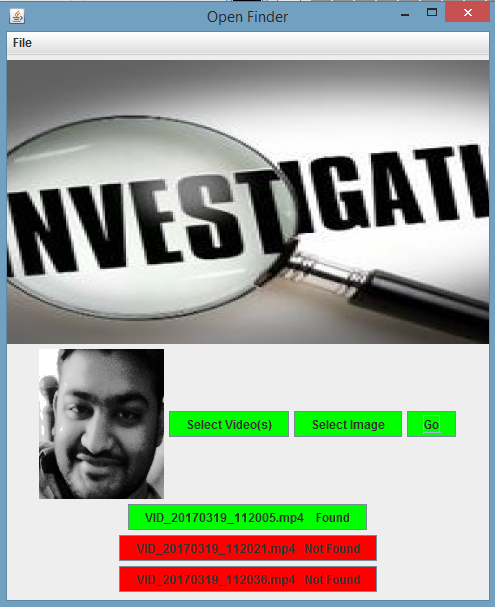
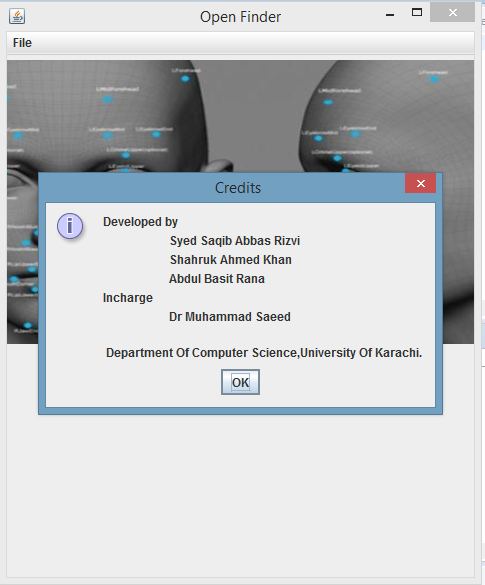


figure 2.5 application showing results

This is a screen of having credits.



**figure 2.6 screenshots of showing**

**2.3 OpenFinder Requirements**

Each and every system have particular requirements in case of implementation of OpenFinder, here our system have some of the basic requirements.

1. Installed and fully configure OpenCV
2. Minimum Windows 7
3. Have fast processer if not than it will take time for processing (for efficient results)

**2.4 OpenFinder Management**

We have managed our project on GIT for better improvement experience ,since we are working on the same time so we created repository on GIT and by using tortoisegit.

This is the link of repository.

Link : https://github.com/saqibabbas/OpenFinder

**Chapter # 3**

**Image Processing**

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

**3.1 PURPOSES OF IMAGE PROCESSING:**

The main five purposes of image processing are:

**3.1.1 Visualization:**

Observe the object that are not visible

**3.1.2 Image sharpening and restoration:**

To create a better image.

**3.1.3 Image retrieval:**

Seek for the image of interest.

**3.1.4 Measurement of pattern:**

Measures various objects in an image.

**3.1.5 Image Recognition:**

Distinguish the objects in an image.

**3.2 USES OF IMAGE PROCESSING**

Following are some of the uses of image processing:

**3.2.1 Pattern Recognition:**

It involves study from image processing and from various other fields that includes machine learning. In pattern recognition, image processing is used for identifying the objects in an images and then machine learning is used to train the system for the change in pattern.

**3.2.2 Color Processing:**

Color processing includes processing of colored images and different color spaces that are used. For example RGB color model. It also involves studying transmission, storage, and encoding of these color images.

**3.2.3 Video Processing:**

The very fast movement of pictures is known as Video. Video processing involves noise reduction, detail enhancement, motion detection, etc. The quality of the video depends on the number of frames/pictures per minute and the quality of each frame being used.

**In other opinion**

Image processing is processing of images using mathematical operations by using any form of [processing](https://en.wikipedia.org/wiki/Signal_processing) for which the input is an image, a series of images, or a video, such as a photographor video frameDigital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

**3.3 Types of Image Processing**

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

Image processing is also used in other perspectives like gaming, virtual reality.360, satellite tracking, x-ray images etc.

MATLAB supports image processing a lot. Now it is very easy to recognize a face and do some work on it using MATLAB .There are many open source libraries which provides numerous effects.In new models of smartphones and tablets and digital cameras there are many applications which are using image processing for better picture quality.

Some of the application are

1. Snapchat
2. Retrica
3. CandyCam
4. B612
5. Sweetselfie

They can add some cartoonic faces on the camera's canvas, they can merge two pictures, they can change the overall due saturation and contrast of the image and many more.

Infact there is a lot of work going on image processing. It is not an unknown word anymore.

There are more examples on image processing just like biotechnology,astronomy,geology,robotics,civil engineering and many other fields are having benefits from it.Like which bolt is having right shape or not , automatic molecule checker, movement of planets and space materials etc.

**Chapter #4**

**OpenCV**

**4.1 Definition**

If you already know OpenCV then stick with OpenCV.OpenFinder using the java wrapper of open cv, Currently OpenCV is the most comprehensive open source library for computer vision and it has large user community. OpenCV has more functions for computer vision than Matlab. Many of its functions are implemented on GPU. The library is being continuously updated (an updated version is released approximately every 3 to 4 months). In general C++ OpenCV code runs faster than Matlab code (if it's not fast enough, you can make it faster by optimizing the source code).Matlab is useful for rapid prototyping and Matlab code is very easy to debug. It has good documentation and support. However, as others have mentioned, Matlab is not open source, its licence is pretty pricey, and its programs are not portable. Matlab is an interpreted language and it negatively affects its performance. Performance matters a lot in computer vision, especially if you are doing real time video processing. Its programs can be made fast too, however you will have to rely on high-level functions (i.e. built-in functions professionally written in C), mex functions (your own compiled C code), and you'll have to learn how to vectorize your code to achieve decent speed.

**4.2 COMPARISION OF MATLAB AND OPENCV**

**4.2.1 Ease of Use**

Matlab is a relatively easy language to get to grips with. Matlab is a pretty high-level scripting language, meaning that you don’t have to worry about libraries, declaring variables, memory management or other lower-level programming issues. As such, it can be very easy to throw together some code to prototype your image processing idea. Say for example I want to read in an image from file and display it.

NeckbeardIndex:  
Matlab **9/10**  
OpenCV **3/10**

**4.2.2 Speed**

Matlab is a pretty high-level scripting language, meaning that you don’t have to worry about memory management or other lower-level programming issues. However, this is due to the fact that Matlab is built on Java, and Java is built upon C. So when you run a Matlab program, your computer is busy trying to interpret all that Matlab code. Then it turns it into Java, and then finally executes the code.OpenCV is basically a library of functions written in C.  You are closer to directly provide machine language code to the computer to get executed. So ultimately you get more image processing done for your computers processing cycles, and not more interpreting. As a result of this, programs written in OpenCV run much faster than similar programs written in Matlab. For example, we might write a small program to detect peoples smiles in a sequence of video frames. In Matlab, we would typically get 3-4 frames analyzed per second. In OpenCV, we would get at least 30fps, resulting in real-time detection.

Neckbeard Index:  
Matlab **2/10**  
OpenCV **9/10**

**4.2.3 ResourcesNeeded**

Closely related to speed, the resources used by both programming environments needs to be looked at. Image and video processing in its nature can be pretty resource heavy due to the amount of pixel data in each image. However, there is a wide gulf on the resource scale between MatlabvsOpenCV. Due to the high level nature of Matlab, it is a resource hog. We have had Matlab code that has required over a gig of RAM to run through video. In comparison, our typical OpenCV programs only require 70mb of RAM to run in real-time (assuming your code has no “leaks”, see “Memory Management” below). Similarly, a full installation of Matlab plus all the toolboxes will take up a couple of gigs on your computer. In comparison, OpenCV only requires a gig at most. 

Neckbeard Index:  
Matlab **4/10**  
OpenCV **9/10**  
  
**4.2.4 Development Environment**

Matlab comes with its own development environment,As the Matlab development environment is based on Java, Mathworks have made versions of their IDE (Integrated development environment) for Windows, Linux and OSX. As such, when you install Matlab, you are in essence installing the programming environment and the IDE as well. We have found the Matlab development environment in Windows and Ubuntu to be a nicely laid out affair. However, in Apple OSX, it can be somewhat awkward to use due to its non-native support of cocoa and having use of X11 instead. For OpenCV, there is no particular IDE that you have to use. Instead, you have a choice of any C programming IDE depending on whether you are using Windows, Linux, or OS X. For Windows, Microsoft Visual Studio or NetBeans is the typical IDE used for OpenCV. In Linux, its Eclipse or NetBeans, and in OSX, we use Apples Xcode. To install OpenCV with your IDE of choice on a particular platform will require following a guide of various complexity. We have been happy with using the various IDEs for OpenCV, but due to Mathworks exclusive IDE for Matlab and its cross-platform ability, Matlab pips OpenCV to the post here. 

Neckbeard Index:  
Matlab **8/10**  
OpenCV **6/10**

**4.2.5 Memory Management**

OpenCV is based on C. As such, every time you allocate a chunk of memory you will have to release it again. If you have a loop in your code where you allocate a chunk of memory in that loop and forget release it afterwards, you will get what is called a “leak”. This is where the program will use a growing amount of memory until it crashes from no remaining memory. Due to the high-level nature of Matlab, it is “smart” enough to automatically allocate and release memory in the background.

Neckbeard Index:  
Matlab **9/10**  
OpenCV **4/10**

**4.2.6 Portability**

Both Matlab and the OpenCV development environment can be installed and run on Linux, Windows, and OS X. In our experience, we have had little issue with OpenCV and Matlab programs developed on one platform and running them on another platform. In some cases headers may have to be changed in OpenCV, or different codecs may have to be used for reading and writing video in Matlab. To the advantage of OpenCV, ports have been written for many additional platforms: here at Fixational we have had OpenCV running on everything from The Cloud (Amazon EC2 instances) to embedded devices such as Apple iPods. This reduces the barrier from a programs prototyping stage, to the final stage of embedding it on a device for real-time video applications.  

Neckbeard Index:  
Matlab **3/10**  
OpenCV**8/10**

**4.2.7 Development of useful programming skills**

Both Matlab and OpenCV are great programming languages to develop image-processing skills. However, as OpenCV is written in C, it’s a great way to simultaneously develop C programming skills. This can be good if you need to develop C programs for other applications, or even as a good jumping off-point if you want to learn Java or C++ for example. As OpenCV programs can be written using the common IDEs such as Netbeans, Xcode or Eclipse, it is also a great way to get to know your way around them for other programming applications that share the same IDEs.   

NeckbeardIndex:  
Matlab **3/10**  
OpenCV **8/10**

**4.2.8 Debugging**

Many of the standard debugging operations can be used with both Matlab and OpenCV: breakpoints can be added to code, the execution of lines can be stepped through, variable values can be viewed during code execution etc. Matlab however, offers a number of additional debugging options over OpenCV. One great feature is that if you need to quickly see the output of a line of code, the semi-colon at the end can be omitted. Also, as Matlab is a scripting language, when execution is stopped at a particular line, the user can type and execute their own lines of code on the fly and view the resulting output without having to recompile and link again. Added to this is are Matlabs powerful functions for displaying data and images, resulting in Matlab being our choice for the easiest development environment for debugging code. 

NeckbeardIndex:  
Matlab **9/10**  
OpenCV **5/10**

**4.3 Conclusion**

**Table: 4.1**

|  |  |  |
| --- | --- | --- |
| **Condition** | **Matlab** | **OpenCV** |
| Ease of use | 9 | 3 |
| Speed | 2 | 9 |
| Resources Needed | 4 | 9 |
| Development Environment | 8 | 6 |
| Memory Management | 9 | 4 |
| Portability | 3 | 8 |
| Development of useful programming skills | 3 | 8 |
| Debugging | 9 | 5 |
| **Total** | **48** | **52** |

From the final scores we can see that OpenCV has the edge over Matlab for image and video processing development . Although Matlab has an easy learning curve, built in memory management, a great help section, it is very slow to execute code, and is expensive to get started in. While OpenCV can be difficult to debug and requires much “housework code” needed for memory management, header files, etc., it wins out due to its free cost, the magnitude of sample code available on the internet, the short development path from prototype code to embedding code, the useful programming skills learnt from its use, and its super-fast speed. Matlab is a more “generic” programming language in that it was designed for many uses, demonstrated by its numerous toolboxes ranging from financial to specialized DNA analyzing tools. On the other hand, OpenCV was made for image processing. Each function and data structure was designed with the image processing coder in mind.   
As such, if you are new to the whole image and video processing area or looking to make a change in your programming environment, we would recommend that you get started in OpenCV.

OpenCV stands for Open Source Computer Vision Library

It was officially launched in 1998 by Intel. It was initially an Intel’s initiative to advance CPU-intensive applications it helped them to render 3D display walls.

Application are given below

* 2D and 3D feature toolkits
* Facial recognition system
* Gesture recognition
* Motion understanding
* Object identification
* Motion tracking
* Augmented reality

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV is written in C++ . There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl,Haskell and Ruby have been developed to encourage adoption by a wider audience.

All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

It is also been used as Hardware Acceleration , If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. A CUDA-based GPU interface has been in progress since September 2010.

**4.4 InstallingOpenCV and Configuring**

Download opencv installer from the link <http://opencv.org/downloads.html>

* Run the basic installation steps.
* Set the active director for opencv installation usually ‘c’ drive where the windows is installed
* Now you have easy steps away from using it on ‘C++’ application
* But for using it for java application you have to create an enviourment variables and restart
* Finally steps ahead for using “opencv” in your application

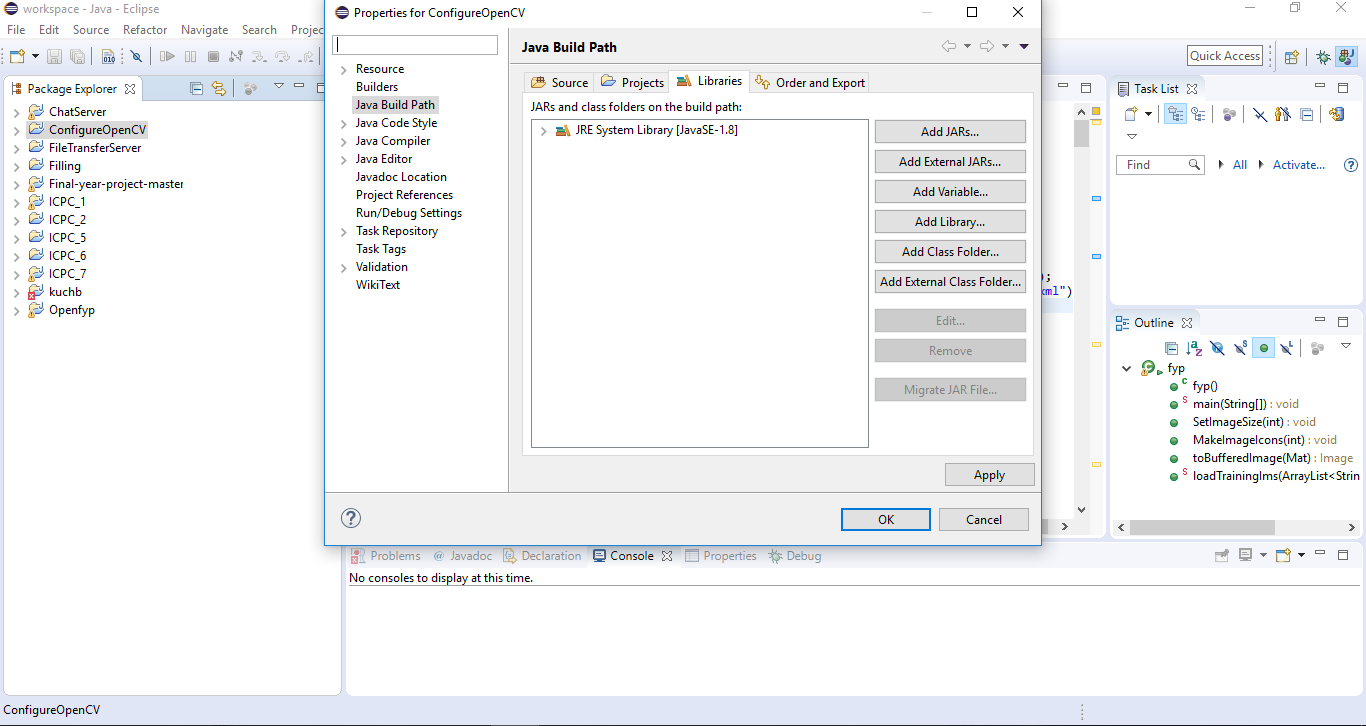


figure 4.0 configuring open cv (i)

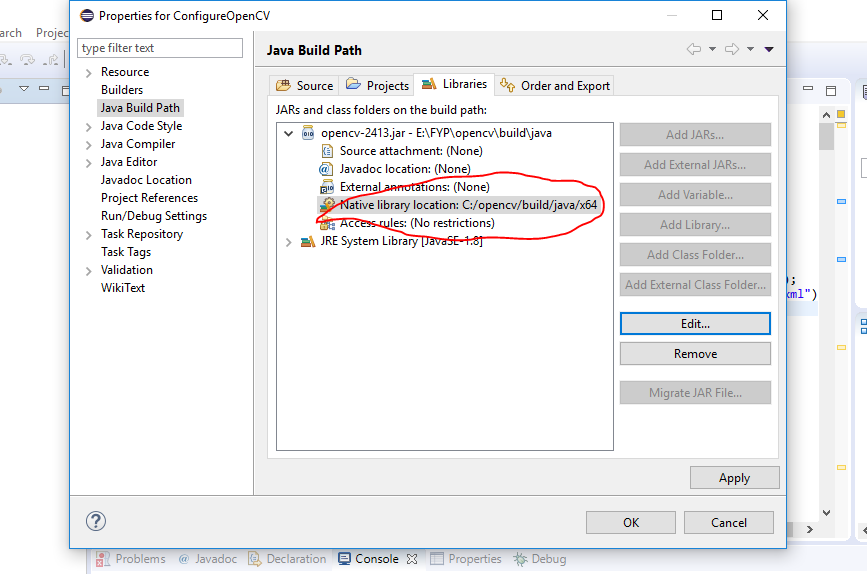
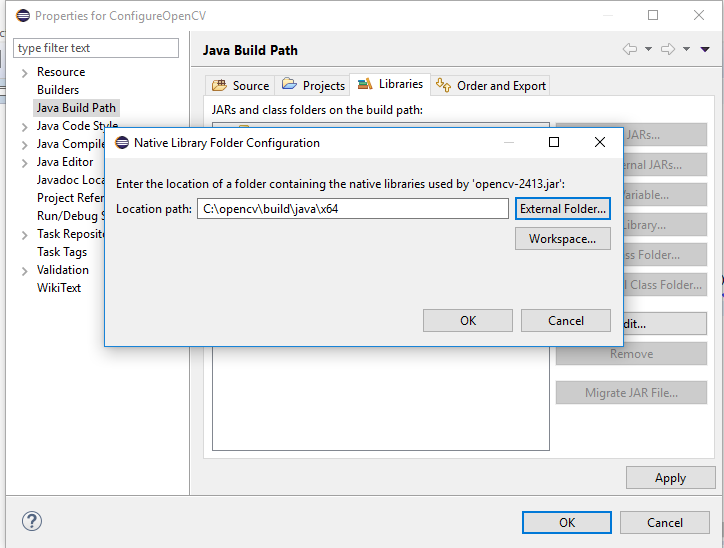
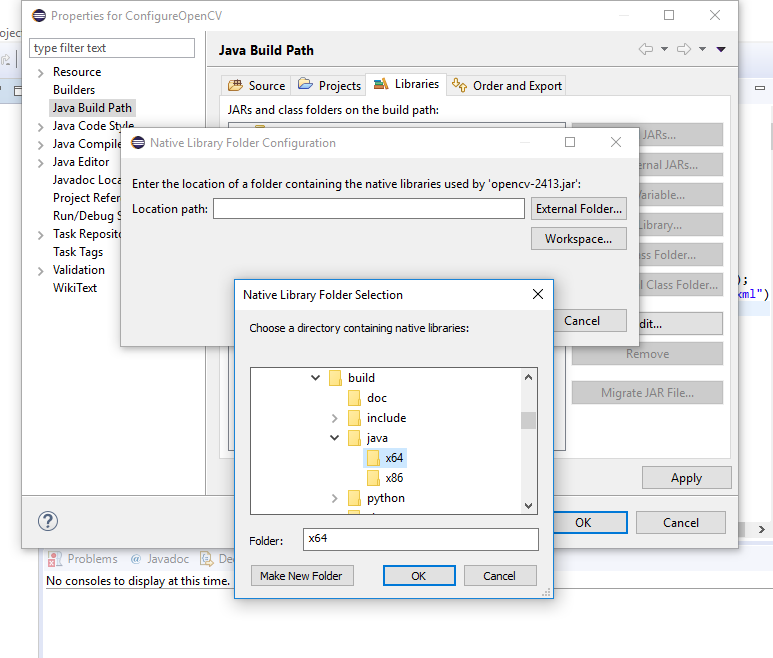
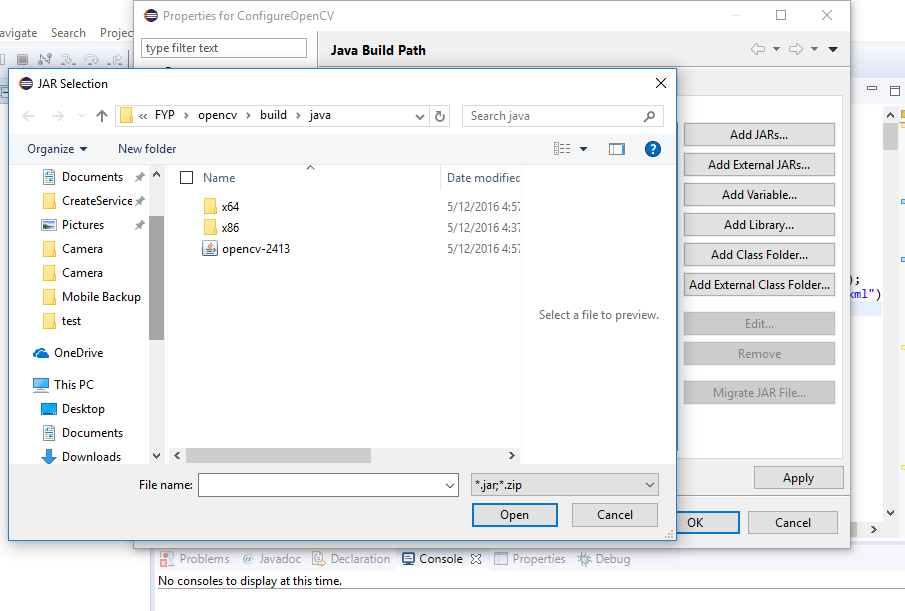
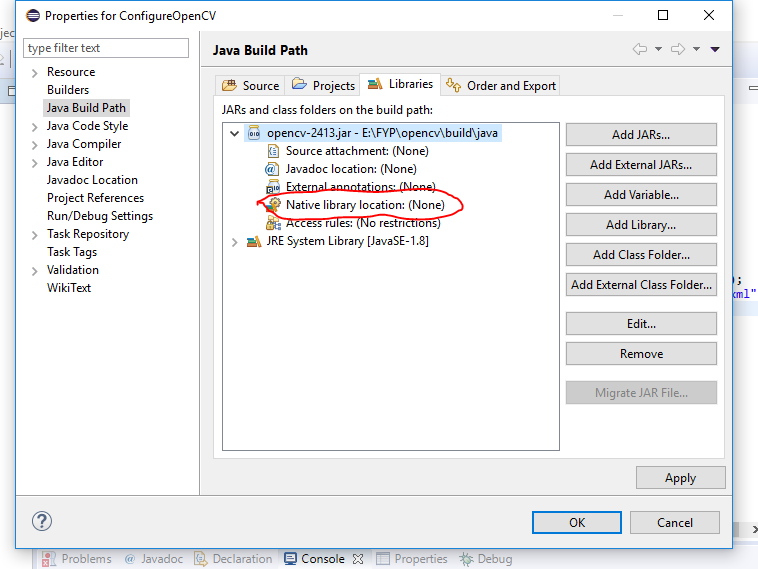


figure 4.1 configuring open cv (ii),4.2 configuring open cv,4.3 configuring open cv (iv),4.4 configuring open cv (v),4.5 onfiguring open cv (vi) respectively

Here finally you have access to all native actions of “opencv” features.

**Chapter# 5**

**Face Detection**

**5.1 Introduction**

As can be assumed, detecting a face is simpler than recognizing a face of a specific person. In order to be able to determine that a certain picture contains a face (or several) we need to be able to define the general structure of a face. Luckily human faces do not greatly differ from each other; we all have noses, eyes, foreheads, chins and mouths; and all of these compose the general structure of a face.

Consider the following 5 figures:

figure 5.0 face features

Each of these figures represents a general feature of a human face. Combining all the features together we, indeed, receive something that resembles a face.



figure 5.1 all face features

By determining if each of these features is similar to some part of our picture, we can conclude if the picture contains a face or not. Notice that this does not have to be an accurate match; we just need to know if, roughly, each of these features corresponds to some part of the image. The technique used for this purpose is Template Matching.

By gathering statistics about which such features compose faces and how, we can train our algorithm to use the right features in the right positions; and thus detect faces.

Let's see an example. See in the figures below how the above features can be used to detect a face (namely, the face of former President Barack Obama).

figure 5.2 example of face features

In order for this process be quick, we design it in such a way that we first check the coarse features which represent the coarse structure of a face; and only if these features match, we continue to the next iteration and use finer features. In each such iteration we can quickly reject areas of the picture which do not match a face, and keep checking those which we are not sure about. In every iteration we increase the certainty that the checked area is indeed a face, until finally we stop and make our determination. The method depicted here is an over-simplified description of the Viola-Jones method (also known as Haar cascades).

**5.2 Haar-cascade Detection**

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. OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: Cascade Classifier Training.Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let's create face and eye detector with OpenCV.First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.F rst, 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:

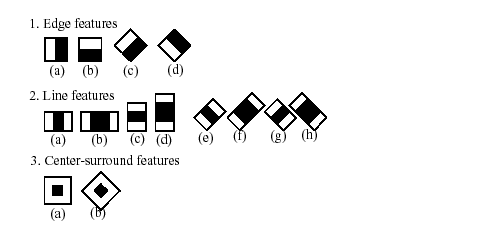


figure 5.3 different types of face features

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 (see below and the integral() description).

**Chapter#6**

**Face Recognition**

**6.1 Introduction**

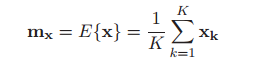
Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of (number of original variables or number of observations). This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

**6.2 The PCA Theory**

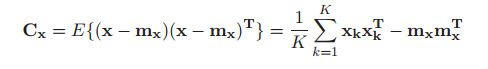
Principal component analysis in signal processing can be described as a transform of a given set of n input vectors (variables) with the same length K formed in the n-dimensional vector x = [x1, x2, ...xn] T into a vector y according to

(3.1)

This point of view enables to form a simple formula but it is necessary to keep in the mind that each row of the vector x consists of K values belonging to one input. The vector mx in Eq. (1) is the vector of mean values of all input variables defined by relation

(3.2)

Matrix A is determined by the covariance matrix Cx. Rows in the A matrix are formed from the eigenvectors e of Cx ordered according to corresponding eigenvalues in descending order. The evaluation of the Cx matrix is possible according to relation

(3.3)

As the vector x of input variables is n-dimensional it is obvious that the size of Cx is n x n. The elements Cx(i, i) lying in its main diagonal are the variances

(3.4)

of x and the other values Cx(i, j) determine the covariance between input variables xi, xj.

(3.5)

between input variables xi, xj . The rows of A in Eq. (1) are orthonormal so the inversion of PCA is possible according to relation

(3.6)

The kernel of PCA defined by Eq. (1) has some other interesting properties resulting from the matrix theory which can be used in the signal and image processing to fulfill various goals as

One of the advance useage of PCA is image Compression .PCA Use for Image Compression Data volume reduction is a common task in image processing. There is a huge amount of algorithms [1, 2, 4] based on various principles leading to the image compression. Algorithms based on the image color reduction are mostly lossy but their results are still acceptable for some applications. The image transformation from color to the gray-level (intensity) image I belongs to the most common algorithms. Its implementation is usually based on the weighted sum of three color components R, G, B according to relation

**I = w1R + w2G + w3B** (3.7)

**6.3 PCA-based Object Recognition**

Throughout the waking day we are bombarded with information from the visual environment. Mostly we make sense of that information, which usually involves identifying or recognizing objects or faces that surround us. Object and face recognition typically occurs so effortlessly that it is hard to believe this is actually a rather complex achievement. In spite of the complexities of object recognition, we can generally go beyond simply identifying objects in the visual environment. For example, we can normally describe what an object would look like if viewed from a different angle, and we know its uses and functions. All in all, there is more to object and face recognition than might initially be supposed. Principal component Analysis is a mathematical technique which transforms the original image data, typically highly correlated, to a new set of uncorrelated variables called principal components. These new components are linear combinations of the original image bands and are derived in decreasing order of importance so that, for example, the first principal component accounts for as much as possible of the variation in the original data. Each principal component is called an eigen channel. The benefits of the technique from a geological standpoint are principally that information not visible in false color composite images can be highlighted in one of the resulting component images.The ability to recognize faces is very important. Face recognition differs in important ways from other forms of object recognition. Face recognition involves more holistic processing , processing that involves strong integration across the whole object. This is important because specific features of a face may be shared by different individuals, or may be subject to change. In the inversion effect, faces are harder to identify when presented upside-down than when upright. In the part–whole effect, memory for a face part is more accurate when it is presented within the whole face rather than on its own. In the composite effect, performance is only impaired when the two halves of different faces are aligned.

**6.4 Eigenfaces**

Eigenfaces is the name given to a set of [eigenvectors](https://en.wikipedia.org/wiki/Eigenvector) when they are used in the [computervision](https://en.wikipedia.org/wiki/Computer_vision) problem of human [facerecognition](https://en.wikipedia.org/wiki/Facial_recognition_system).The approach of using eigenfacesfor [recognition](https://en.wikipedia.org/wiki/Facial_recognition_system) was developed by Sirovich and Kirby (1987) and used by Matthew Turk and [AlexPentland](https://en.wikipedia.org/wiki/Alex_Pentland) in face classification.The eigenvectors are derived from the [covariancematrix](https://en.wikipedia.org/wiki/Covariance_matrix) of the [probabilitydistribution](https://en.wikipedia.org/wiki/Probability_distribution) over the high-[dimensional](https://en.wikipedia.org/wiki/Dimension) [vectorspace](https://en.wikipedia.org/wiki/Vector_space) of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix.

**6.4.1 Computation of the Eigenfaces**

Step 1: obtain face images I1 , I2 , ..., IM (training faces) (very important: the face images must be centered and of the same size)

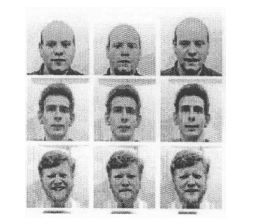


figure 6.0 eigen faces

Step 2: represent every image Ii as a vector Γi

Step 3: compute the average face vector Ψ: Ψ = 1 M M i=1 Σ Γi

Step 4: subtract the mean face: Φi = Γi − Ψ

Step 5: compute the covariance matrix C: C = 1 M M n=1 Σ ΦnΦ T n = AAT (N 2 xN 2 matrix) .

where A = [Φ1 Φ2 ... ΦM] (N 2 xM matrix)

Step 6: compute the eigenvectors ui of AAT The matrix AAT is very large --> not practical !!

Step 6.1: consider the matrix A T A (MxM matrix)

Step 6.2: compute the eigenvectors vi of A T A A T Avi = µ ivi What is the relationship between usi and vi ? A T Avi = µ ivi => AAT Avi = µ i Avi =>CAvi = µ i Avi or Cui = µ iui where ui = Avi Thus, AAT and A T A have the same eigenvalues and their eigenvectors are related as follows: ui = Avi !! Note 1: AAT can have up to N 2 eigenvalues and eigenvectors. Note 2: A T A can have up to M eigenvalues and eigenvectors. Note 3: The M eigenvalues of A T A (along with their corresponding eigenvectors) correspond to the M largest eigenvalues of AAT (along with their corresponding eigenvectors).

Step 6.3: compute the M best eigenvectors of AAT : ui = Avi (important: normalize ui such that ||ui || = 1)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

**6.5 STEPS FOLLOW IN A PCA FOR DIGITAL IMAGE**

The steps normally followed in a PCA of a digital image can now be established.

**6.5.1:**

In the computational model of a digital image, in expression 1, the variables *X*1*, X*2*,...,X*p are the columns of the image. The PCA is begun by coding (correcting) the image to that its columns have zero means and unitary variances. This is common, in order to avoid one or the other of the columns having undue influence on the principal components.

|  |
| --- |
| ***Image corrected by the mean = image – mean of the image*** |

**6.5.2**:

The covariance matrix C is calculated using expression 6, implemented computationally, that is:

|  |
| --- |
| ***covImage = image corrected by the mean x (image corrected by the mean)T*** |

**6.5.3:**

The eigenvalues λ1,λ2,...,λp and the corresponding eigenvectors *a*1*, a*2*,..., a*p. are calculated.

**6.5.4:**

The value of a vector of characteristics is obtained, a matrix with vectors containing the list of eigenvectors (matrix columns) of the covariance matrix.

***Vc = (av1,av2,av3,…,avn)***

**6.5.5**:

The final data are obtained, that is, a matrix with all the eigenvectors (components) of the covariance matrix.

|  |
| --- |
| ***Finaldata = vcT x (Image - mean)T*** |

**6.5.6:**

The original image is obtained from the final data without compression using the expression

|  |
| --- |
| ***ImageT = (vc)T x finaldata + meanT*** |

**6.5.7:**

Any components that explain only a small portion of the variation in data for the effect of image compression are discarded. The eliminations have the effect of reducing the quantity of eigenvectors of the characteristics vectors and can produce final data with a smaller dimension. The use of expression 13 in these conditions allow the recovery of the original image with compression.

**Chapter#7**

**Bibliography**

# Biography

<http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html>

<https://web.stanford.edu/class/ee368/Project_03/Project/reports/ee368group01.pdf>

<http://eyalarubas.com/face-detection-and-recognition.html>

<http://encyclopedia2.thefreedictionary.com/image+processing>

<http://fivedots.coe.psu.ac.th/~ad/jg/?%3F>

<https://en.m.wikipedia.org/wiki/Facial_recognition_system>

<http://sciencenetlinks.com/science-news/science-updates/wasp-face-processing/>