** ****

**THIRD GENERATION ATM MACHINE USING ADVANCED COMPUTER VISION TECHNIQUES**

#### A PROJECT REPORT

***Submitted by***

**ARUN.L.K [REGISTER NO:211417104023] ALAGUPANDI.J [REGISTER NO:211417104011]**

**AAKASH.S [REGISTER NO:211417104001]**

***in partial fulfilment for the award of the degree of***

### BACHELOR OF ENGINEERING

**IN**

COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

Certified that this project report **“THIRD GENERATION ATM MACHINE USING ADVANCD COMPUTER VISION TECHNIQUES”** is the bonafide work of **“ ARUN.L.K(2017PECCS365), ALAGUPANDI.J (2017PECCS359),**

**AAKASH.S (2017PECCS353) ”** who carried out the project work under my supervision.

#### SIGNATURE SIGNATURE

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Mr. M.MOHAN,M.Tech.,(Ph.D.), HEAD OF THE DEPARTMENT ASSISTANT PROFESSOR(GRADE-I),** DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

2

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on

INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors

**Tmt.C.VIJAYARAJESWARI**, **Thiru.C.SAKTHIKUMAR,M.E.,** and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,**

for the support extended throughout the project.

We would like to thank my **Project Guide Mr.Sathish N** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project

**ARUN.L.K**

**ALAGUPANDI.J**

**AAKASH.S**

**Abstract**

Automated teller machines (ATMs) are well known devices typically used by individuals to carry out a variety of personal and business financial transactions and/or banking functions. ATMs have become very popular with the general public for their availability and general user friendliness. ATMs are now found in many locations having a regular or high volume of consumer traffic. For example, ATMs are typically found in restaurants, supermarkets, Convenience stores, malls, schools, gas stations, hotels, work locations, banking centres, airports, entertainment establishments, transportation facilities and a myriad of other locations. ATMs are typically available to consumers on a continuous basis such that consumers have the ability to carry out their ATM financial transactions and/or banking functions at any time of the day and on any day of the week.

In such ATM Security has always been one of the most prominent issues concerning the daily users and the not so frequent ones as well on. This situation is hypothetical, yet very possible scenario of an individual's ATM card falling into the wrong hands, and the PIN number being cracked by a theft perpetrating entity. Our proposed model uses certain factors which would be monitored right from the initiation, to the end of the respective transaction. With the help of these factors, we would declare the status of the transaction before proceeding with cash withdrawal. Such monitoring would assist the transaction with a secure approach to bank upon

## TABLE OF CONTENTS

## CHAPTER NO TITLE PAGE.NO

## 1. Introduction

## 1.1 overview 6

## 1.2 image processing 7

## 1.3 types of image 11

## 2. Literature survey 14

## 3. Existing system 18

## 4. Proposed system 18

## 5. Software requirements 18

## 5.1 Python 18

## 5.2 open cv 24

## 6. System Architecture 43

## 7. Face recognition technique

## 7.1 Introduction 44

## 7.2 Necessity of face recognition 44

## 7.3 Evolution 44

## 7.4 recent deployment 46

## 7.6 Face recognition 47

## 7.7 Filter band approach 48

## 8. UML diagrams

## 8.1 use case diagram 55

## 8.2 class diagram 56

## 8.3 interaction diagram 57

## 8.4 activity diagram 58

## 9. conclusionINTRODUCTION

**1.1 Overview**

Banks seek to reduce their infrastructure costs by shifting transactions of their customers to Automatic Teller Machines (ATMs) and Internet websites.. ATM is one such machine which made money transactions easy for customers to bank. The other side of this improvement is the enhancement of the culprit's probability to get his „unauthentic‟ share. Traditionally, security is handled by requiring the combination of a physical access card and a PIN or other password in order to access a customer's account. This model invites fraudulent attempts through stolen cards, badly-chosen or automatically assigned PINs, cards with little or no encryption schemes, employees with access to no encrypted customer account information and other points of failure. Our paper proposes an automatic teller machine security model that would combine a physical access card, a PIN, and electronic facial recognition. By forcing the ATM to match a live image of a customer‟s face with an image stored in a bank database that is associated with the account number, the damage to be caused by stolen cards and PINs is effectively neutralized. With the technological advances in financial infrastructure, most bank customers prefer to use Automatic Teller Machines (ATMs) and Internet websites for carrying out their banking transactions. Financial users especially utilize ATMs for physical transactions like cash withdrawal or cash deposit. However, just like any other system, ATMs are also suffering from numerous issues caused by users. Among these problems, card and/or cash forgetting (CCF) is a common issue. The main goal of our work is to propose a computer vision framework which uses the embedded ATM camera to perform face detection and recognition in order to prevent such unnecessary losses generated by CCF. In the studied scenario, we consider the case where a customer withdraws money from an ATM in a conventional setting. After the customer inserts the card into the ATM, the proposed system starts to perform face detection and builds a temporary face database for the customer using the camera located inside the ATM. If the customer leaves the ATM without taking his/her card or cash, the ATM waits for the customer to be back instead of retracting the forgotten item. If the system finds out there is a different customer approaching the ATM before the card/cash holder, the card/cash will be retracted at that moment. This scenario is fundamentally different than biometric authentication scenarios, in which a person’s image is matched to a gallery image acquired, possibly, a long time before the matching, under different conditions. In this scenario, the matching image and the gallery image are separated by mere minutes at most.

**1.2 Image processing**

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

These operations, taken [singly](http://www.answers.com/topic/singly) or in combination, are the means by which the image is enhanced, restored, or compressed. An image is enhanced when it is modified so that the information it contains is more clearly evident, but enhancement can also include making the image more visually appealing.

An example is noise smoothing. To smooth a [noisy](http://www.answers.com/topic/noisy) image, median filtering can be applied with a 3 × 3 pixel window. This means that the value of every pixel in the noisy image is recorded, along with the values of its nearest eight neighbours. These nine numbers are then ordered according to size, and the median is selected as the value for the pixel in the new image. As the 3 × 3 window is moved one pixel at a time across the noisy image, the filtered image is formed.

Another example of enhancement is contrast manipulation, where each pixel's value in the new image depends solely on that pixel's value in the old image; in other words, this is a point operation. Contrast manipulation is commonly performed by adjusting the brightness and contrast controls on a television set, or by controlling the exposure and development time in [printmaking](http://www.answers.com/topic/printmaking). Another point operation is that of [pseudo colouring](http://www.answers.com/topic/pseudocoloring) a black-and-white image, by assigning arbitrary colours to the gray levels. This technique is popular in [thermograph](http://www.answers.com/topic/thermography) (the imaging of heat), where hotter objects (with high pixel values) are assigned one color (for example, red), and cool objects (with low pixel values) are assigned another color (for example, blue), with other colours assigned to intermediate values.

Recognizing object classes in real-world images is a long standing goal in Computer vision. Conceptually, this is challenging due to large appearance variations of object instances belonging to the same class. Additionally, distortions from background clutter, scale, and viewpoint variations can render appearances of even the same object instance to be vastly different. Further challenges arise from interclass similarity in which instances from different classes can appear very similar. Consequently, models for object classes must be flexible enough to accommodate class variability, yet discriminative enough to sieve out true object instances in cluttered images. These seemingly paradoxical requirements of an object class model make recognition difficult. This paper addresses two goals of recognition are image classification and object detection. The task of image classification is to determine if an object class is present in an image, while object detection localizes all instances of that class from an image. Toward these goals, the main contribution in this paper is an approach for object class recognition that employs edge information only. The novelty of our approach is that we represent contours by very simple and generic shape primitives of line segments and ellipses, coupled with a flexible method to learn discriminative primitive combinations. These primitives are complementary in nature, where line segment models straight contour and ellipse models curved contour. We choose an ellipse as it is one of the simplest circular shapes, yet is sufficiently flexible to model curved shapes. These shape primitives possess several attractive properties. First, unlike edge-based descriptors they support abstract and perceptually meaningful reasoning like parallelism and adjacency. Also, unlike contour fragment features, storage demands by these primitives are independent of object size and are efficiently represented with four parameters for a line and five parameters for an ellipse.

Additionally, matching between primitives can be efficiently computed (e.g., with geometric properties), unlike contour fragments, which require comparisons between individual edge pixels. Finally, as geometric properties are easily scale normalized, they simplify matching across scales. In contrast, contour fragments are not scale invariant, and one is forced either to rescale fragments, which introduces aliasing effects (e.g., when edge pixels are pulled apart), or to resize an image before extracting fragments, which degrades image resolution.

In recent studies it is shown that the generic nature of line segments and ellipses affords them an innate ability to represent complex shapes and structures. While individually less distinctive, by combining a number of these primitives, we empower a combination to be sufficiently discriminative. Here, each combination is a two-layer abstraction of primitives: pairs of primitives (termed shape tokens) at the first layer, and a learned number of shape tokens at the second layer. We do not constrain a combination to have a fixed number of shape-tokens, but allow it to automatically and flexibly adapt to an object class. This number influences a combination’s ability to represent shapes, where simple shapes favor fewer shape-tokens than complex ones. Consequently, discriminative combinations of varying complexity can be exploited to represent an object class. We learn this combination by exploiting distinguishing shape, geometric, and structural constraints of an object class. Shape constraints describe the visual aspect of shape tokens, while geometric constraints describe its spatial layout (configurations). Structural constraints enforce possible poses/structures of an object by the relationships (e.g., XOR relationship) between shape-tokens.

**1.3 CLASSIFICATION OF IMAGES:**

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Color Image

**1.3.1 BINARY IMAGE:**

A binary image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that has only two possible values for each [pixel](http://en.wikipedia.org/wiki/Pixel).  Typically the two colors used for a binary image are black and white though any two colors can be used.  The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as [gray scale images](http://en.wikipedia.org/wiki/Grayscale)

Binary images often arise in [digital image processing](http://en.wikipedia.org/wiki/Digital_image_processing) as [masks](http://en.wikipedia.org/w/index.php?title=Mask_(image_processing)&action=edit&redlink=1) or as the result of certain operations such as [segmentation](http://en.wikipedia.org/wiki/Segmentation_(image_processing)), [thresholding](http://en.wikipedia.org/wiki/Thresholding_(image_processing)), and [dithering](http://en.wikipedia.org/wiki/Dither). Some input/output devices, such as [laser printers](http://en.wikipedia.org/wiki/Laser_printer), [fax machines](http://en.wikipedia.org/wiki/Fax), and bi-level [computer displays](http://en.wikipedia.org/wiki/Visual_display_unit), can only handle bi-level images

**1.3.2 GRAY SCALE IMAGE**

A gray scale Image is [digital image](http://en.wikipedia.org/wiki/Digital_image) is an image in which the value of each [pixel](http://en.wikipedia.org/wiki/Pixel) is a single [sample](http://en.wikipedia.org/wiki/Sample_(signal)), that is, it carries only [intensity](http://en.wikipedia.org/wiki/Luminous_intensity) information. Images of this sort, also known as [black-and-white](http://en.wikipedia.org/wiki/Black-and-white), are composed exclusively of shades of [gray](http://en.wikipedia.org/wiki/Gray) (0-255), varying from black (0) at the weakest intensity to white (255) at the strongest.

Gray scale images are distinct from one-bit [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) images, which in the context of computer imaging are images with only the two [colors](http://en.wikipedia.org/wiki/Color), [black](http://en.wikipedia.org/wiki/Black), and [white](http://en.wikipedia.org/wiki/White) (also called bi-level or [binary images](http://en.wikipedia.org/wiki/Binary_image)). Gray scale images have many shades of gray in between. Gray scale images are also called [monochromatic](http://en.wikipedia.org/wiki/Monochromatic), denoting the absence of any [chromatic](http://en.wikipedia.org/wiki/Chromaticity) variation.

Gray scale images are often the result of measuring the intensity of light at each pixel in a single band of the [electromagnetic spectrum](http://en.wikipedia.org/wiki/Electromagnetic_spectrum) (e.g. [infrared](http://en.wikipedia.org/wiki/Infrared), [visible light](http://en.wikipedia.org/wiki/Visible_spectrum), [ultraviolet](http://en.wikipedia.org/wiki/Ultraviolet), etc.), and in such cases they are monochromatic proper when only a given [frequency](http://en.wikipedia.org/wiki/Frequency) is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

**1.3.3 COLOUR IMAGE:**

A (digital) color image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that includes [color](http://en.wikipedia.org/wiki/Color) information for each [pixel](http://en.wikipedia.org/wiki/Pixel). Each pixel has a particular value which determines it’s appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colors Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colors is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255).

In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colours. This allows the image to contain a total of 256x256x256 = 16.8 million different colours. This technique is also known as RGB encoding, and is specifically adapted to human vision

It is observable that our behaviour and social interaction are greatly influenced by emotions of people whom we intend to interact with. Hence a successful emotion recognition system could have great impact in improving human computer interaction systems in such a way as to make them be more user-friendly and acting more human-like.

Moreover, there are a number of applications where emotion recognition can play an important role including biometric authentication, high-technology surveillance and security systems, image retrieval, and passive demographical data collections.

It is unarguable that face is one the most important feature that characterises human beings. By only looking ones’ faces, we are not only able to tell who the2y are but also perceive a lot of information such as their emotions, ages and genders.

This is why emotion recognition by face has received much interest in computer vision research community over past two decades.

**2**.**LITERATURE SURVEY:**

**Aru, Okereke Eze, Ihekweaba Gozie**

There is an urgent need for improving security in banking region. With the birth of the Automatic Teller Machines, banking became a lot easier though with its own troubles of insecurity. Due to tremendous increase in the number of criminals and their activities, the ATM has become insecure. ATM systems today use no more than an access card and PIN for identity verification. The recent progress in biometric identification techniques, including finger printing, retina scanning, and facial recognition has made a great efforts to rescue the unsafe situation at the ATM. This research looked into the development of a system that integrates facial recognition technology into the identity verification process used in ATMs. An ATM model that is more reliable in providing security by using facial recognition software is proposed .The development of such a system would serve to protect consumers and financial institutions alike from intruders and identity thieves. This paper proposes an automatic teller machine security model that would combine a physical access card, a PIN, and electronic facial recognition that will go as far as withholding the fraudster‟s card. If this technology becomes widely used, faces would be protected as well as PINs. However, it obvious that man‟s biometric features cannot be replicated, this proposal will go a long way to solve the problem of Account safety making it possible for the actual account owner alone have access to his accounts. The combined biometric features approach is to serve the purpose both the identification and authentication that card and PIN do.

**Radha.S , Imaya.S**

This paper describes a novel biometric scenario, where a person is authenticated at an ATM, and has to be re-identified from a camera within a very short time period, under very challenging illumination and pose conditions I using Single Identifying System. We propose a multimodal authentication system that operates under the constraints imposed by this applications scenario, and implement face recognition and RFID Card appearance recognition to create a system that improves ATM behavior in case of forgotten card or cash by re-identifying the user from an embedded ATM camera. We focus on the scenario and the platform, and report tests with the proposed system under challenging conditions, obtained from ATMs placed in the field.

**Manpreet Kaur , Rajeev Kumar Bedi & S.K Gupta**

There's an urgent want for making improvements to safety in banking region. With birth of Automatic Teller Machines, banking became lots less difficult although with its possessing troubles of insecurity. Because of huge increase within quantity of criminals and their activities, ATM has become insecure. ATM methods in these days use no more than an access card and PIN for identity verification. Latest growth in biometric identification methods, including finger printing, retina scanning, and facial realization has made a first-rate efforts to rescue hazardous predicament at ATM. This study seemed into progress of a procedure that integrates facial recognition technological know-how into identity verification procedure used in ATMs. An ATM model that's extra trustworthy in delivering protection with aid of making use of facial awareness program is proposed. Progress of this sort of approach would serve to shield customers and economic associations alike from intruders and identity thieves. This article proposes an automatic teller machine protection model that will minimize motion blur utilizing STRA algorithm and further compare performance based on few parameters (False Negative, False Positive, and Accuracy). If this technology becomes widely used, faces would be saved as well as PINs.

**A. Ross and A. Jain**

User verification systems that use a single biometric indicator often have to contend with noisy sensor data, restricted degrees of freedom, non-universality of the biometric trait and unacceptable error rates. Attempting to improve the performance of individual matchers in such situations may not prove to be effective because of these inherent problems. Multi biometric systems seek to alleviate some of these drawbacks by providing multiple evidences of the same identity. These systems help achieve an increase in performance that may not be possible using a single biometric indicator. Further, multi biometric systems provide anti-spoofing measures by making it difficult for an intruder to spoof multiple biometric traits simultaneously. However, an effective fusion scheme is necessary to combine the information presented by multiple domain experts. This paper addresses the problem of information fusion in biometric verification systems by combining information at the matching score level. Experimental results on combining three biometric modalities (face, fingerprint and hand geometry) are presented.

**M. Baranitharan, R. Nagarajan, G. ChandraPraba**

Nowadays most of the 300 million surveillance cameras today are ‘blind’ and merely record videos for post incident manual analysis, So The system deals with the development of an application for automation of video surveillance in ATM machines and detect any type of potential criminal activities that might be arising with the automated system which would considerably decrease the inefficiency that are existing in the prevalent systems. An advanced Human detection system using Open Computer Vision technique and Artificial Intelligence would be utilized which would create phenomenal results in the detection of the activities and their categorization. The proposed system makes efficient utilization of Open CV which has more than 2500 optimized algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects finally ending up with the detection and identification of the necessary action for the prevention of such type of activities. The proposed system includes the specialized mechanisms for Camera tampering, Collision of human, Risky voice analysis, long time tracking. The entire mechanism takes place in real time decreasing the time complexity to a great extent making the system an efficient mechanism to prevent such anti-social activities.

**3. EXISTING SYSTEM:**

In existing system magnetic strips card is used as ATM card, IR sensor in order to sense the presence of the card holders and to turn on Fan and Light, if ATM is tampered then SMS is sent to two main stations via GSM. Based on WI fall detection get security, that network access is not that much secured.

**4. Proposed system**

* The study is focused on Design and Implementation of Face Detection based ATM Security System using Embedded Linux Platform.
* The system is implemented on the credit card size Raspberry Pi board with extended capability of open source Computer Vision (Open CV) software which is used for Image processing operation.
* High level security mechanism is provided by the consecutive actions such as initially system captures the human face and check whether the human face is detected properly or not. If the face is not detected properly, it warns the user to adjust him/her properly to detect the face.

**5.software requirements:**

* Python programming
* Open CV

**5.1 PYTHON**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for programming Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

**5.1.2 PYTHON FEATURES:**

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple programming paradigms, including object-oriented , [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive library. Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and aspect [-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by meta [programming](https://en.wikipedia.org/wiki/Metaprogramming) and [meta objects](https://en.wikipedia.org/wiki/Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

**5.1.3 PYTHON LIBRARIES**

Python's large [standard library](https://en.wikipedia.org/wiki/Standard_library), commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications,

many standard formats and protocols such as [MIME](https://en.wikipedia.org/wiki/MIME) and [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) are supported. It

includes modules for creating [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface), connecting to [relational databases](https://en.wikipedia.org/wiki/Relational_database), [generating pseudorandom numbers](https://en.wikipedia.org/wiki/Pseudorandom_number_generator), arithmetic with arbitrary precision decimals, manipulating [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), and [unit testin](https://en.wikipedia.org/wiki/Unit_testing)g.

As of March 2018, the [Python Package Index](https://en.wikipedia.org/wiki/Python_Package_Index) (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

* Graphical user interfaces
* Web frameworks
* Multimedia
* Databases
* Networking
* Test frameworks
* Automation
* Web scraping
* Documentation
* System administration
* Scientific computing
* Text processing
* Image processing

**5.1.5 DEPLOYMENT ENVIRONMENT**

Most Python implementations (including CPython) include a [read–eval–print loop](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop) (REPL), permitting them to function as a [command line interpreter](https://en.wikipedia.org/wiki/Command_line_interpreter) for

which the user enters statements sequentially and receives results immediately

Other shells, including [IDLE](https://en.wikipedia.org/wiki/IDLE_(Python)) and [IPython](https://en.wikipedia.org/wiki/IPython), add further abilities such as auto-completion, session state retention and [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting).

As well as standard desktop [integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment), there are [Web browser](https://en.wikipedia.org/wiki/Web_browser)-based IDEs; [Sage Math](https://en.wikipedia.org/wiki/SageMath) (intended for developing science and math-related Python programs); [Python Anywhere](https://en.wikipedia.org/wiki/PythonAnywhere), a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

**5.1.6 OPENCV-PYTHON**

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also. So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it

And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc. As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV. Remember, we together can make this project a great success !!!

**5.1.7 ARRAY ATTRIBUTES**

Array attributes reflect information that is intrinsic to the array itself. Generally, accessing an array through its attributes allows you to get and sometimes set intrinsic properties of the array without creating a new array. The exposed attributes are the core parts of an array and only some of them can be reset meaningfully without creating a new array.

For a 1-D array, this has no effect. (To change between column and row vectors, first cast the 1-D array into a matrix object.) For a 2-D array, this is the usual matrix transpose. For an n-D array, if axes are given, their order indicates how the axes are permuted (see Examples). If axes are not provided and a.shape = (i[0], i[1], ... i[n-2], i[n-1]), then a.transpose().shape = (i[n-1], i[n-2], ... i[1], i[0]).

**5.1.8 SCALARS**

Python defines only one type of a particular data class (there is only one integer type, one floating-point type, etc.). This can be convenient in applications that don’t need to be concerned with all the ways data can be represented in a computer. For scientific computing, however, more control is often needed. In NumPy, there are 24 new fundamental Python types to describe different types of scalars. These type descriptors are mostly based on the types available in the C language that CPython is written in, with several additional types compatible with Python’s types.

**5.1.9 METHODS**

Array scalars have exactly the same methods as arrays. The default behavior of these methods is to internally convert the scalar to an equivalent 0-dimensional array and to call the corresponding array method. In addition, math operations on array scalars are defined so that the same hardware flags are set and used to interpret the results as for ufunc, so that the error state used for ufuncs also carries over to the math on array scalars.

**5.1.10 DATA TYPE OBJECTS (DTYPE)**

A data type object (an instance of numpy.dtype class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted. It describes the following aspects of the data: 1. Type of the data (integer, float, Python object, etc.) 2. Size of the data (how many bytes is in e.g. the integer) 3. Byte order of the data (little-endian or big-endian) 4. If the data type is structured, an aggregate of other data types, (e.g., describing an array item consisting of an integer and a float), (a) what are the names of the “fields” of the structure, by which they can be accessed, (b) what is the data-type of each field, and (c) which part of the memory block each field takes. 5. If the data type is a sub-array, what is its shape and data type. To describe the type of scalar data, there are several built-in scalar types in Numpy for various precision of integers, floating-point numbers, etc. An item extracted from an array, e.g., by indexing, will be a Python object whose type is the scalar type associated with the data type of the array. Note that the scalar types are not dtype objects, even though they can be used in place of one whenever a data type specification is needed in Numpy. Structured data types are formed by creating a data type whose fields contain other data types. Each field has a name by which it can be accessed. The parent data type should be of sufficient size to contain all its fields; the parent is nearly always based on the void type which allows an arbitrary item size. Structured data types may also contain nested structured sub-array data types in their fields. Finally, a data type can describe items that are themselves arrays of items of another data type. These sub-arrays must, however, be of a fixed size. If an array is created using a data-type describing a sub-array, the dimensions of the sub-array are appended to the shape of the array when the array is created. Sub-arrays in a field of a structured type behave differently, see Field Access. Sub-arrays always have a C-contiguous memory layout.

**5.2 OpenCV**

OpenCV was started at Intel in 1999 by **Gary Bradsky** and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel’s Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

**5.2.1OpenCV-Python**

Python is a general purpose programming language started by **Guido van Rossum**, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task more easier. **Numpy** is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

**5.2.2 OpenCV-Python Tutorials**

OpenCV introduces a new set of tutorials which will guide you through various functions available in OpenCV-Python. **This guide is mainly focused on OpenCV 3.x version** (although most of the tutorials will work with OpenCV 2.x also).

A prior knowledge on Python and Numpy is required before starting because they won’t be covered in this guide. **Especially, a good knowledge on Numpy is must to write optimized codes in OpenCV-Python.**

This tutorial has been started by *Abid Rahman K.* as part of Google Summer of Code 2013 program, under the guidance of *Alexander Mordvintsev*.

**5.2.3 OpenCV Needs You !!!**

Since OpenCV is an open7 source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also.

So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it.

And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc.

As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV.

Remember, we **together** can make this project a great success !!!

**5.2.4 Contributors**

Below is the list of contributors who submitted tutorials to OpenCV-Python.

1. Alexander Mordvintsev (GSoC-2013 mentor)
2. Abid Rahman K. (GSoC-2013 intern)

**5.2.5 Additional Resources**

1. A Quick guide to Python - [A Byte of Python](http://swaroopch.com/notes/python/)
2. [Basic Numpy Tutorials](http://wiki.scipy.org/Tentative_NumPy_Tutorial)
3. [Numpy Examples List](http://wiki.scipy.org/Numpy_Example_List)
4. [OpenCV Documentation](http://docs.opencv.org/)
5. [OpenCV Forum](http://answers.opencv.org/questions/)

# Install OpenCV-Python in Windows

## Goals

**In this tutorial**

* We will learn to setup OpenCV-Python in your Windows system.

*Below steps are tested in a Windows 7-64 bit machine with Visual Studio 2010 and Visual Studio 2012. The screenshots shows VS2012.*

## 5.2.6 Installing OpenCV from prebuilt binaries

1. Below Python packages are to be downloaded and installed to their default locations.

1.1. [Python-2.7.x](http://python.org/ftp/python/2.7.5/python-2.7.5.msi).

1.2. [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download).

1.3. [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe) (*Matplotlib is optional, but recommended since we use it a lot in our tutorials*).

1. Install all packages into their default locations. Python will be installed to **C:/Python27/**.
2. After installation, open Python IDLE. Enter import numpy and make sure Numpy is working fine.
3. Download latest OpenCV release from [sourceforge site](http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.6/OpenCV-2.4.6.0.exe/download) and double-click to extract it.
4. Goto **opencv/build/python/2.7** folder.
5. Copy **cv2.pyd** to **C:/Python27/lib/site-packeges**.
6. Open Python IDLE and type following codes in Python terminal.
7. >>> import cv2
8. >>> print cv2.\_\_version\_\_

If the results are printed out without any errors, congratulations !!! You have installed OpenCV-Python successfully.

## 5.2.7 Building OpenCV from source

1. Download and install Visual Studio and CMake.

1.1. [Visual Studio 2012](http://go.microsoft.com/?linkid=9816768)

1.2. [CMake](http://www.cmake.org/files/v2.8/cmake-2.8.11.2-win32-x86.exe)

1. Download and install necessary Python packages to their default locations

2.1. [Python 2.7.x](http://python.org/ftp/python/2.7.5/python-2.7.5.msi)

2.2. [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download)

2.3. [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe) (*Matplotlib is optional, but recommended since we use it a lot in our tutorials.*)

**Note**

In this case, we are using 32-bit binaries of Python packages. But if you want to use OpenCV for x64, 64-bit binaries of Python packages are to be installed. Problem is that, there is no official 64-bit binaries of Numpy. You have to build it on your own. For that, you have to use the same compiler used to build Python. When you start Python IDLE, it shows the compiler details. You can get more [information here](http://stackoverflow.com/q/2676763/1134940). So your system must have the same Visual Studio version and build Numpy from source.

**Note**

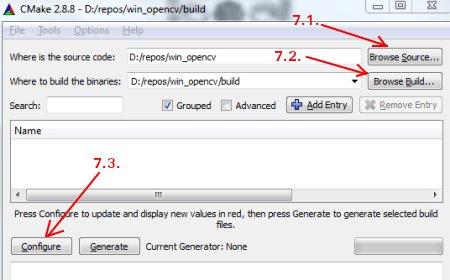
Another method to have 64-bit Python packages is to use ready-made Python distributions from third-parties like [Anaconda](http://www.continuum.io/downloads), [Enthought](https://www.enthought.com/downloads/) etc. It will be bigger in size, but will have everything you need. Everything in a single shell. You can also download 32-bit versions also.

1. Make sure Python and Numpy are working fine.
2. Download OpenCV source. It can be from [Sourceforge](http://sourceforge.net/projects/opencvlibrary/) (for official release version) or from [Github](https://github.com/Itseez/opencv) (for latest source).
3. Extract it to a folder, opencv and create a new folder build in it.
4. Open CMake-gui (*Start > All Programs > CMake-gui*)
5. Fill the fields as follows (see the image below):

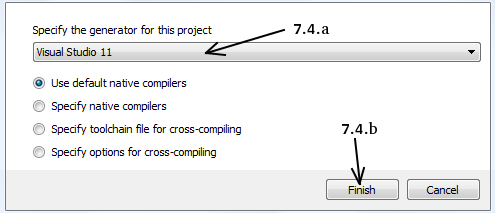
7.1. Click on **Browse Source...** and locate the opencv folder.

7.2. Click on **Browse Build...** and locate the build folder we created.

7.3. Click on **Configure**.

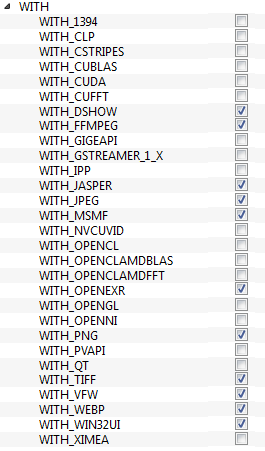


7.4. It will open a new window to select the compiler. Choose appropriate compiler (here, Visual Studio 11) and click **Finish**.

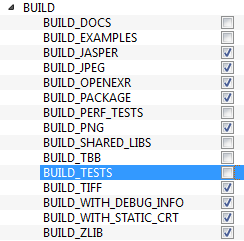


7.5. Wait until analysis is finished.

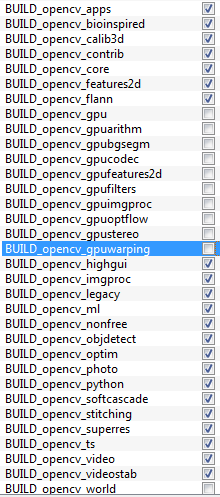
1. You will see all the fields are marked in red. Click on the **WITH** field to expand it. It decides what extra features you need. So mark appropriate fields. See the below image:



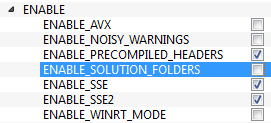
1. Now click on **BUILD** field to expand it. First few fields configure the build method. See the below image:



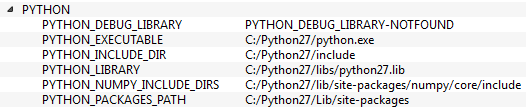
1. Remaining fields specify what modules are to be built. Since GPU modules are not yet supported by OpenCV-Python, you can completely avoid it to save time (But if you work with them, keep it there). See the image below:



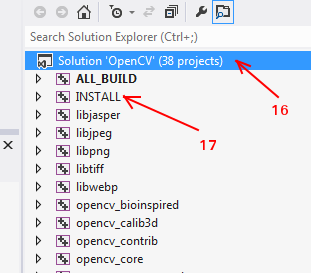
1. Now click on **ENABLE** field to expand it. Make sure **ENABLE\_SOLUTION\_FOLDERS** is unchecked (Solution folders are not supported by Visual Studio Express edition). See the image below:



1. Also make sure that in the **PYTHON** field, everything is filled. (Ignore PYTHON\_DEBUG\_LIBRARY). See image below:



1. Finally click the **Generate** button.
2. Now go to our **opencv/build** folder. There you will find **OpenCV.sln** file. Open it with Visual Studio.
3. Check build mode as **Release** instead of **Debug**.
4. In the solution explorer, right-click on the **Solution** (or **ALL\_BUILD**) and build it. It will take some time to finish.
5. Again, right-click on **INSTALL** and build it. Now OpenCV-Python will be installed.



1. Open Python IDLE and enter import cv2. If no error, it is installed correctly.

**Note**

We have installed with no other support like TBB, Eigen, Qt, Documentation etc. It would be difficult to explain it here. A more detailed video will be added soon or you can just hack around.

## Additional Resources

## Exercises

1. If you have a windows machine, compile the OpenCV from source. Do all kinds of hacks. If you meet any problem, visit OpenCV forum and explain your problem.

## Goals

**In this tutorial**

* We will learn to setup OpenCV-Python in your Fedora system. Below steps are tested for Fedora 18 (64-bit) and Fedora 19 (32-bit).

## Introduction

OpenCV-Python can be installed in Fedora in two ways, 1) Install from pre-built binaries available in fedora repositories, 2) Compile from the source. In this section, we will see both.

Another important thing is the additional libraries required. OpenCV-Python requires only **Numpy**(in addition to other dependencies, which we will see later). But in this tutorials, we also use **Matplotlib** for some easy and nice plotting purposes (which I feel much better compared to OpenCV). Matplotlib is optional, but highly recommended. Similarly we will also see **IPython**, an Interactive Python Terminal, which is also highly recommended.

## Installing OpenCV-Python from Pre-built Binaries

Install all packages with following command in terminal as root.

$ yum install numpy opencv\*

Open Python IDLE (or IPython) and type following codes in Python terminal.

>>> import cv2

>>> print cv2.\_\_version\_\_

If the results are printed out without any errors, congratulations !!! You have installed OpenCV-Python successfully.

It is quite easy. But there is a problem with this. Yum repositories may not contain the latest version of OpenCV always. For example, at the time of writing this tutorial, yum repository contains 2.4.5 while latest OpenCV version is 2.4.6. With respect to Python API, latest version will always contain much better support. Also, there may be chance of problems with camera support, video playback etc depending upon the drivers, ffmpeg, gstreamer packages present etc.

So my personnel preference is next method, i.e. compiling from source. Also at some point of time, if you want to contribute to OpenCV, you will need this.

## Installing OpenCV from source

Compiling from source may seem a little complicated at first, but once you succeeded in it, there is nothing complicated.

First we will install some dependencies. Some are compulsory, some are optional. Optional dependencies, you can leave if you don’t want.

### Compulsory Dependencies

We need **CMake** to configure the installation, **GCC** for compilation, **Python-devel** and **Numpy** for creating Python extensions etc.

yum install cmake

yum install python-devel numpy

yum install gcc gcc-c++

Next we need **GTK** support for GUI features, Camera support (libdc1394, libv4l), Media Support (ffmpeg, gstreamer) etc.

yum install gtk2-devel

yum install libdc1394-devel

yum install libv4l-devel

yum install ffmpeg-devel

yum install gstreamer-plugins-base-devel

### Optional Dependencies

Above dependencies are sufficient to install OpenCV in your fedora machine. But depending upon your requirements, you may need some extra dependencies. A list of such optional dependencies are given below. You can either leave it or install it, your call :)

OpenCV comes with supporting files for image formats like PNG, JPEG, JPEG2000, TIFF, WebP etc. But it may be a little old. If you want to get latest libraries, you can install development files for these formats.

yum install libpng-devel

yum install libjpeg-turbo-devel

yum install jasper-devel

yum install openexr-devel

yum install libtiff-devel

yum install libwebp-devel

Several OpenCV functions are parallelized with **Intel’s Threading Building Blocks** (TBB). But if you want to enable it, you need to install TBB first. ( Also while configuring installation with CMake, don’t forget to pass -D WITH\_TBB=ON. More details below.)

yum install tbb-devel

OpenCV uses another library **Eigen** for optimized mathematical operations. So if you have Eigen installed in your system, you can exploit it. ( Also while configuring installation with CMake, don’t forget to pass -D WITH\_EIGEN=ON. More details below.)

yum install eigen3-devel

If you want to build **documentation** ( *Yes, you can create offline version of OpenCV’s complete official documentation in your system in HTML with full search facility so that you need not access internet always if any question, and it is quite FAST!!!* ), you need to install **Sphinx** (a documentation generation tool) and **pdflatex** (if you want to create a PDF version of it). ( Also while configuring installation with CMake, don’t forget to pass -D BUILD\_DOCS=ON. More details below.)

yum install python-sphinx

yum install texlive

### Downloading OpenCV

Next we have to download OpenCV. You can download the latest release of OpenCV from [sourceforge site](http://sourceforge.net/projects/opencvlibrary/). Then extract the folder.

Or you can download latest source from OpenCV’s github repo. (If you want to contribute to OpenCV, choose this. It always keeps your OpenCV up-to-date). For that, you need to install **Git**first.

yum install git

git clone https://github.com/Itseez/opencv.git

It will create a folder OpenCV in home directory (or the directory you specify). The cloning may take some time depending upon your internet connection.

Now open a terminal window and navigate to the downloaded OpenCV folder. Create a new buildfolder and navigate to it.

mkdir build

cd build

### Configuring and Installing

Now we have installed all the required dependencies, let’s install OpenCV. Installation has to be configured with CMake. It specifies which modules are to be installed, installation path, which additional libraries to be used, whether documentation and examples to be compiled etc. Below command is normally used for configuration (executed from build folder).

cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local ..

It specifies that build type is “Release Mode” and installation path is /usr/local. Observe the -Dbefore each option and .. at the end. In short, this is the format:

cmake [-D <flag>] [-D <flag>] ..

You can specify as many flags you want, but each flag should be preceded by -D.

So in this tutorial, we are installing OpenCV with TBB and Eigen support. We also build the documentation, but we exclude Performance tests and building samples. We also disable GPU related modules (since we use OpenCV-Python, we don’t need GPU related modules. It saves us some time).

*(All the below commands can be done in a single cmake statement, but it is split here for better understanding.)*

* Enable TBB and Eigen support:
* cmake -D WITH\_TBB=ON -D WITH\_EIGEN=ON ..
* Enable documentation and disable tests and samples
* cmake -D BUILD\_DOCS=ON -D BUILD\_TESTS=OFF -D BUILD\_PERF\_TESTS=OFF -D BUILD\_EXAMPLES=OFF ..
* Disable all GPU related modules.
* cmake -D WITH\_OPENCL=OFF -D WITH\_CUDA=OFF -D BUILD\_opencv\_gpu=OFF -D BUILD\_opencv\_gpuarithm=OFF -D BUILD\_opencv\_gpubgsegm=OFF -D BUILD\_opencv\_gpucodec=OFF -D BUILD\_opencv\_gpufeatures2d=OFF -D BUILD\_opencv\_gpufilters=OFF -D BUILD\_opencv\_gpuimgproc=OFF -D BUILD\_opencv\_gpulegacy=OFF -D BUILD\_opencv\_gpuoptflow=OFF -D BUILD\_opencv\_gpustereo=OFF -D BUILD\_opencv\_gpuwarping=OFF ..
* Set installation path and build type
* cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local ..

Each time you enter cmake statement, it prints out the resulting configuration setup. In the final setup you got, make sure that following fields are filled (below is the some important parts of configuration I got). These fields should be filled appropriately in your system also. Otherwise some problem has happened. So check if you have correctly performed above steps.

-- GUI:

-- GTK+ 2.x: YES (ver 2.24.19)

-- GThread : YES (ver 2.36.3)

-- Video I/O:

-- DC1394 2.x: YES (ver 2.2.0)

-- FFMPEG: YES

-- codec: YES (ver 54.92.100)

-- format: YES (ver 54.63.104)

-- util: YES (ver 52.18.100)

-- swscale: YES (ver 2.2.100)

-- gentoo-style: YES

-- GStreamer:

-- base: YES (ver 0.10.36)

-- video: YES (ver 0.10.36)

-- app: YES (ver 0.10.36)

-- riff: YES (ver 0.10.36)

-- pbutils: YES (ver 0.10.36)

-- V4L/V4L2: Using libv4l (ver 1.0.0)

-- Other third-party libraries:

-- Use Eigen: YES (ver 3.1.4)

-- Use TBB: YES (ver 4.0 interface 6004)

-- Python:

-- Interpreter: /usr/bin/python2 (ver 2.7.5)

-- Libraries: /lib/libpython2.7.so (ver 2.7.5)

-- numpy: /usr/lib/python2.7/site-packages/numpy/core/include (ver 1.7.1)

-- packages path: lib/python2.7/site-packages

-- Documentation:

-- Build Documentation: YES

-- Sphinx: /usr/bin/sphinx-build (ver 1.1.3)

-- PdfLaTeX compiler: /usr/bin/pdflatex

--

-- Tests and samples:

-- Tests: NO

-- Performance tests: NO

-- C/C++ Examples: NO

Many other flags and settings are there. It is left for you for further exploration.

Now you build the files using make command and install it using make install command. make install should be executed as root.

make

su

make install

Installation is over. All files are installed in /usr/local/ folder. But to use it, your Python should be able to find OpenCV module. You have two options for that.

1. **Move the module to any folder in Python Path** : Python path can be found out by entering import sys;print sys.path in Python terminal. It will print out many locations. Move /usr/local/lib/python2.7/site-packages/cv2.so to any of this folder. For example,
2. su mv /usr/local/lib/python2.7/site-packages/cv2.so /usr/lib/python2.7/site-packages

But you will have to do this every time you install OpenCV.

1. **Add ``/usr/local/lib/python2.7/site-packages`` to the PYTHON\_PATH**: It is to be done only once. Just open ~/.bashrc and add following line to it, then log out and come back.
2. export PYTHONPATH=$PYTHONPATH:/usr/local/lib/python2.7/site-packages

Thus OpenCV installation is finished. Open a terminal and try import cv2.

To build the documentation, just enter following commands:

make docs

make html\_docs

Then open opencv/build/doc/\_html/index.html and bookmark it in the browser

**6. System architecture**

### **C:\Users\user\Pictures\atm_block_modified.PNG**

First we are taking the input as an image of the face it undergoes prepossessing where the distortion in the image is restored and undergoes face feature extraction when the feature is extracted it goes to classifier where it compares features from the data which is extracted from the database if its same authenticated it will proceed if not it will intimation goes to the email.

**7 FACE RECOGNITION TECHNIQUE**

**7.1 INTRODUCTION**

The face recognition process includes mainly three-task: a*cquisition, normalization andrecognition.The term acquisition means,*the detection and tracking of face-like image patches in a dynamic scene*. Normalizationis* thesegmentation, alignment and normalization of the face images, and finally r*ecognition is* the representation and modeling of face images as identities, and the association of novel face images with known models.

**7.2 NECESSITY FOR FACE RECOGNITION**

The are many ways that humans can identify each other, and so is for machines. There are many different identification technologies available, many of which have been in commercial use for years. The most common person verification and identification methods today are Password/PIN known as Personal identification number systems. The problem with that or other similar techniques, are not unique, and is possible for somebody to forget loose or even have it stolen for somebody else. In order to overcome these problems there has developed considerable interest in "biometrics" identification systems, which use pattern recognition techniques to identify people using their characteristics. Some of those methods are fingerprints and retina and iris recognition, but these techniques are not easy to use. This can be overcome by using spatial face recognition approaches.

**7.3 EVOLUTION OF FACE RECOGNITION**

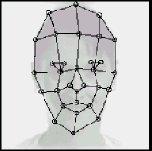
Face recognition rises from the moment that machine started to become more and more "intelligent" and had the advance of fill in, correct or help the lack of human abilities and senses.

The subject of face recognition is as old as computer vision and both because of the practical importance of the topic and theoretical interest from cognitive science. Face recognition is not the only method of recognizing other people. Even humans between each other use senses in order to recognize others. Machines have a wider range for recognition purposes, which use thinks such as fingerprints, or iris scans. Despite the fact that these methods of identification can be more accurate, face recognition has always remains a major focus of research because of its non-invasive nature and it is people's primary method of person identification.

Since the start of that field of technology two main approaches are followed.

* Geometrical approach
* Pictorial approach.

The geometrical approach uses the spatial configuration of facial features. This means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances and angles between features. On the other hand, the pictorial approach uses templates to recognize facial features. This pictorial method uses the templates of the major facial features and entire face to perform recognition on frontal views of faces. Many of the projects were based on these two approaches have some common extensions, which handle different poses backgrounds. Apart from these two techniques the other recent template-based approaches, which form templates from the image gradient, and the principal component analysis approach, which can be read as a sub-optimal template approach.



**Figure 3.1** Pictorial and Geometric approaches

The pictorial and feature geometry approaches has been applied to faces at varying pose and expression is shown in the figure 3.1

Aftergeometrical approach or pictorial approach methods, the other methods like eigenfaces, Principal Component Analysis, and methods that process images in combination with neural networks.

In the recognition stage, the input is compared against all selected model views of each person. To compare the input against a particular model view, the face is first geometrically aligned with the model view. An affine transform is applied to the input to bring the facial features automatically located by the system into correspondence with the same features on the model. A technique based on the optical flow between the transformed input and the model is used to compensate for any remaining small transformation between the two. Templates from the model are then compared with the image using normalized correlation. Both the model and input images are pre-processed with a differential operator.

**7.4 RECENT DEVELOPMENT:**

With the rapid evolution of the technology and the commercialization of technological achievements, face recognition became more and more popular, not only for research but also for the use of security systems. That gave motive to many researchers, and also companies in order to develop techniques for automatically recognizing faces that would find many applications, including security and human-computer interaction. For instance, a face recognizing machine could allow automated access control for buildings or enable a computer to recognize the person sitting at the console. Most existing face recognition systems, however, work only for frontal or nearly frontal images of faces. By recognizing faces under varying pose, one makes the conditions under which face recognition systems operate less rigid.

**7.5 FACE RECOGNITION AND FACE DETECTION**

Face recognition is a technique of recognizing faces but it is not necessary to "freeze" the user in order to take a picture. Though there is a problem with recognizing faces when the pose of the face is different, but in particular, there is a limit on face rotations in depth, which include left and right and up and down rotations. Face recognition itself is difficult because it is a fine discrimination task among similar objects. Adding pose variation naturally makes the problem more difficult. This is because the appearance of a person's face changes under rotation since the face has a complex 3D structure.

A distinguish have to be made between face recognition and face detection. Many people thing that these two terms are the same. Thoughface recognition and face detection have many similar techniques, based on the same idea and algorithms, they differ from each other. The main difference is the fact that, face recognition is detecting faces and search through a dataset in order to find an exact match.

Face recognition is computationally and psychophysically more appropriate to consider them as a set of co-operative visual modules with closed-loop feedback. In order to realize such a system, an integrated approach has been adopted which will perform acquisition, normalization and recognition in a coherent way. Images of a dynamic scene are processed in real-time to acquire normalize and aligned face sequences. In essence, this process is a closed-loop module that includes the computation and fusion of three different Visual cues: motion, color and face appearance models.

In general much research effort has been concentrated on face recognition tasks in which only a single image or at most a few images of each person are available. A major concern has been scalability to large databases containing thousands of images. However, large intra-subject variability casts doubt upon the possibility of scaling face recognition, at least in this form, to very large.

The tasks of face recognition mostly require recognition to be performed using sequences acquired and normalize automatically in poorly constrained dynamic scenes. These are characterized by low resolution, large-scale changes, variable illumination and occasionally inaccurate cropping and alignment. Recognition based upon isolated images of this kind is highly inconsistent and unreliable. However, accumulating recognition scores over time can compensate the poor quality of the data.

Face recognition is an active research area involving different fields such as physics, psychology, biology, mathematics, computer science and several others. A wide range of problems has been approached, resulting in many interesting applications.

**7.6 FILTER BANK APPROACH:**

In filter bank approach, difference of Gaussian is used to overcome lighting and facial expression problems. Along with DOG, Weber local descriptor and Gabor filter bank can be used to increase the efficiency of the system.

In [imaging science](http://en.wikipedia.org/wiki/Imaging_science), difference of Gaussian is a [feature](http://en.wikipedia.org/wiki/Feature_(computer_vision)) enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original. In the simple case of [grayscale images](http://en.wikipedia.org/wiki/Grayscale), the blurred images are obtained by [convolving](http://en.wikipedia.org/wiki/Convolution) the original [grayscale images](http://en.wikipedia.org/wiki/Grayscale) with Gaussian kernels having differing standard deviations. Blurring an image using a [Gaussian](http://en.wikipedia.org/wiki/Gaussian_blur) [kernel](http://en.wikipedia.org/wiki/Convolution_kernel) suppresses only [high-frequency spatial](http://en.wikipedia.org/wiki/Spatial_frequencies) information. Subtracting one image from the other preserves spatial information that lies between the range of frequencies that are preserved in the two blurred images. Thus, the difference of Gaussians is a [band-pass filter](http://en.wikipedia.org/wiki/Band-pass_filter) that discards all but a handful of spatial frequencies that are present in the original grayscale image

As a [feature](http://en.wikipedia.org/wiki/Feature_(computer_vision)) enhancement algorithm, the difference of Gaussians can be utilized to increase the visibility of edges and other detail present in a digital image. A wide variety of alternative [edge sharpening filters](http://en.wikipedia.org/wiki/Edge_detection) operate by enhancing high frequency detail, but because [random noise](http://en.wikipedia.org/wiki/Random_noise) also has a high spatial frequency, many of these sharpening filters tend to enhance noise, which can be an undesirable artefact. The difference of Gaussians algorithm removes high frequency detail that often includes random noise, rendering this approach one of the most suitable for processing images with a high degree of noise

The next step in face recognition is to pass the images of Gaussian filter to Weber Local Descriptor (WLD). It is based on the fact that human perception of a pattern depends not only on the change of a stimulus (such as sound, lighting) but also on the original intensity of the stimulus. Specifically, WLD consists of two components: differential excitation and orientation. The differential excitation component is a function of the ratio between two terms: one is the relative intensity differences of a current pixel against its neighbours; the other is the intensity of the current pixel. The orientation component is the gradient orientation of the current pixel.

**8 UML DIAGRAMS**

UML stands for Unified Modeling Language. It’s a rich language to model software solutions, application structures, system behavior and business processes. There are 14 UML diagram types to help you model these behaviors. Unified Modeling Language™ (UML®) is a standard visual modeling language intended to be used for

* modeling business and similar processes,
* analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

Specification explained that process:

* provides guidance as to the order of a team’s activities,
* specifies what artifacts should be developed,
* directs the tasks of individual developers and the team as a whole, and
* offers criteria for monitoring and measuring a project’s products and activities.

UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).UML is not complete, and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be

intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

Name of an abstract classifier is shown in italics while final classifier has no specific graphical notation, so there is no way to determine whether classifier is final or not from the diagram.

#### 9.1 List of UML Diagram Types

So, what are the different UML diagram types? There are two main categories; structure diagrams and behavioral diagrams. Click on the links to learn more about a specific diagram type.

#### Structure Diagrams

Structure diagrams show the things in the modeled system. In a more technical term, they show different objects in a system. Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

#### Class Diagram

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom.

In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

#### Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. The image below shows a component diagram.

#### Deployment Diagram

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.

#### Package Diagram

As the name suggests, a package diagram shows the dependencies between different packages in a system. Check out this wiki article to learn more about the dependencies and elements found in package diagrams.

#### Composite Structure Diagram

Composite structure diagrams are used to show the internal structure of a class. For a detailed explanation of composite structure diagrams, click here.

#### Use Case Diagram

As the most known diagram type of the behavioral UML diagrams, use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system. You can create use case diagrams using our tool and/or get started instantly using our use case templates.

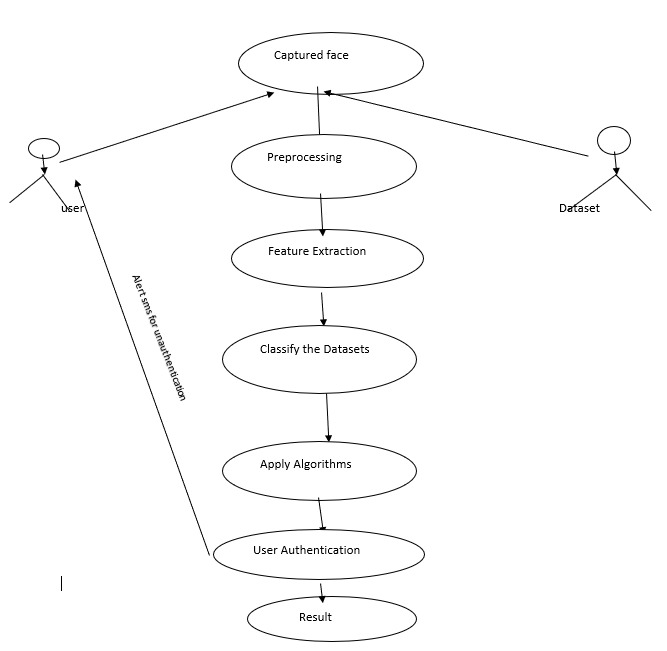
#### Activity Diagram

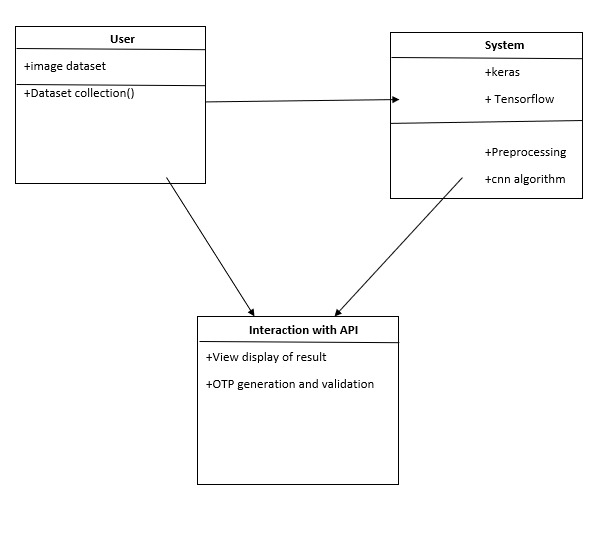
Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams. Check out this wiki article to learn about symbols and usage of activity diagrams.

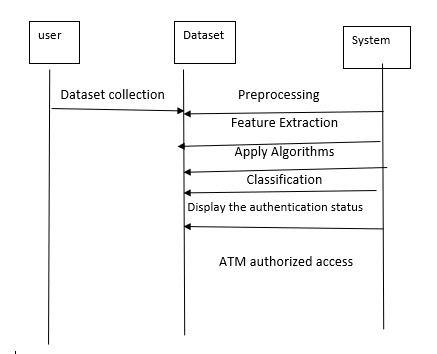
#### Sequence Diagram

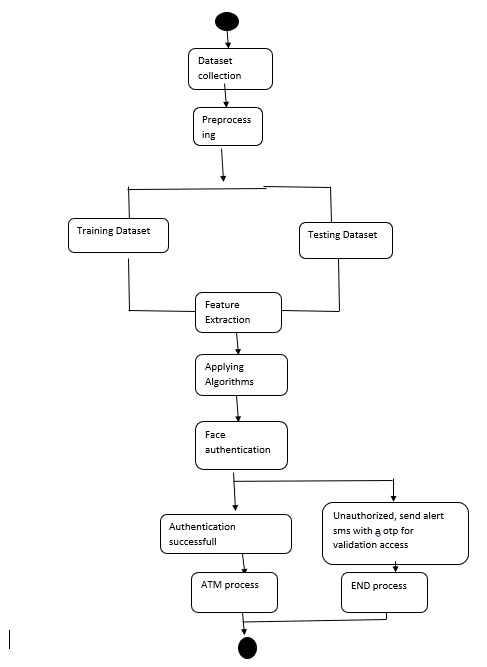
Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a scenario. The processes are represented vertically, and interactions are shown as arrows. This article explains the purpose and the basics of Sequence diagrams. Also, check out this complete Sequence Diagram Tutorial to learn more about sequence diagrams. You can also instantly start drawing using our sequence diagram templates.

**8.1 Use case diagram:**

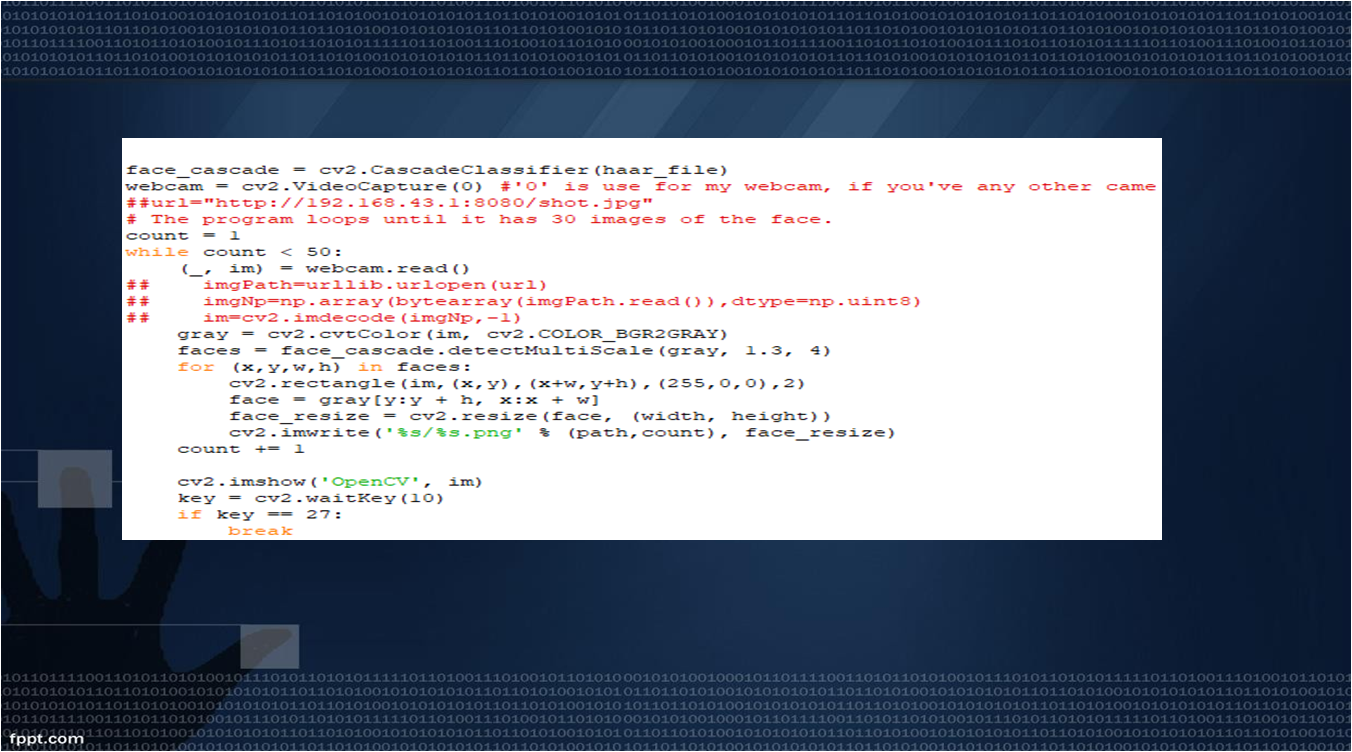
****

**8.2 Class diagram**

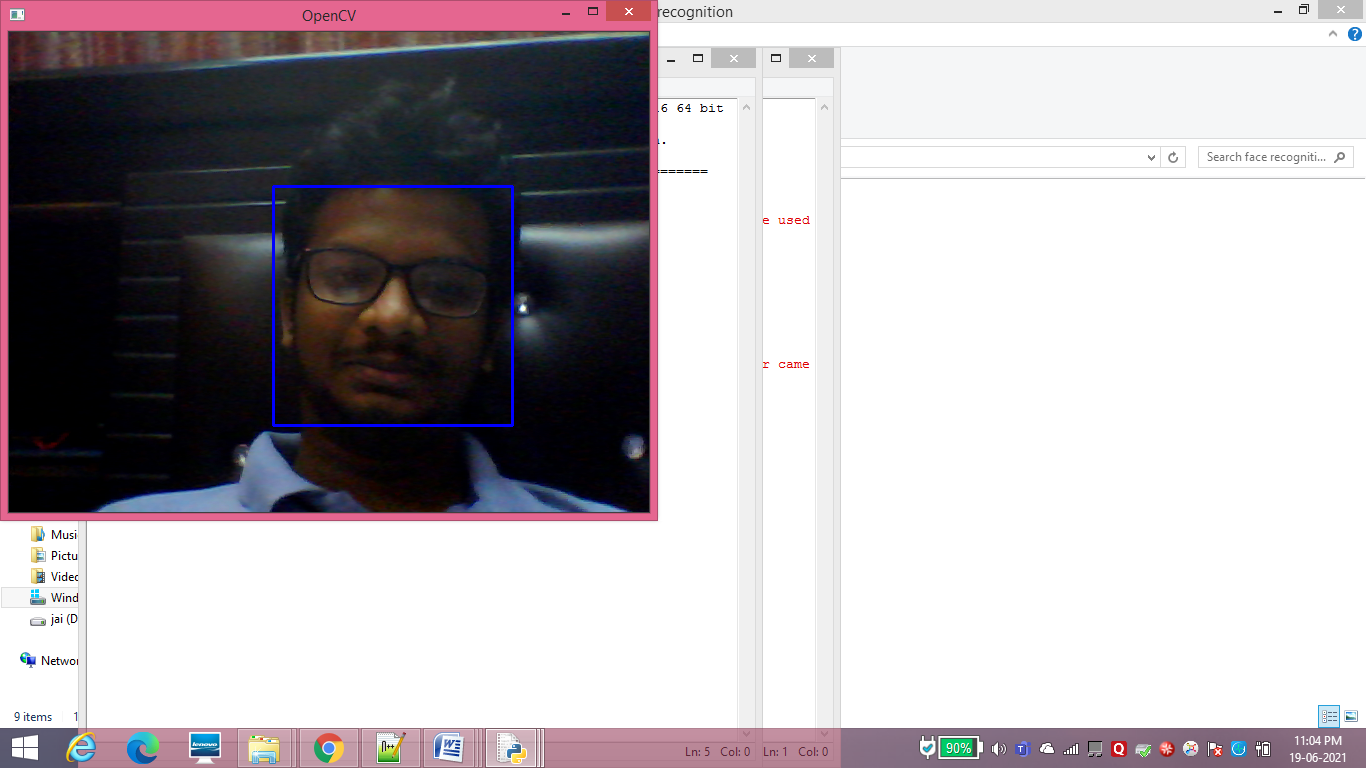
**8.3 Interaction diagram:**

**8.4 Activity diagram**

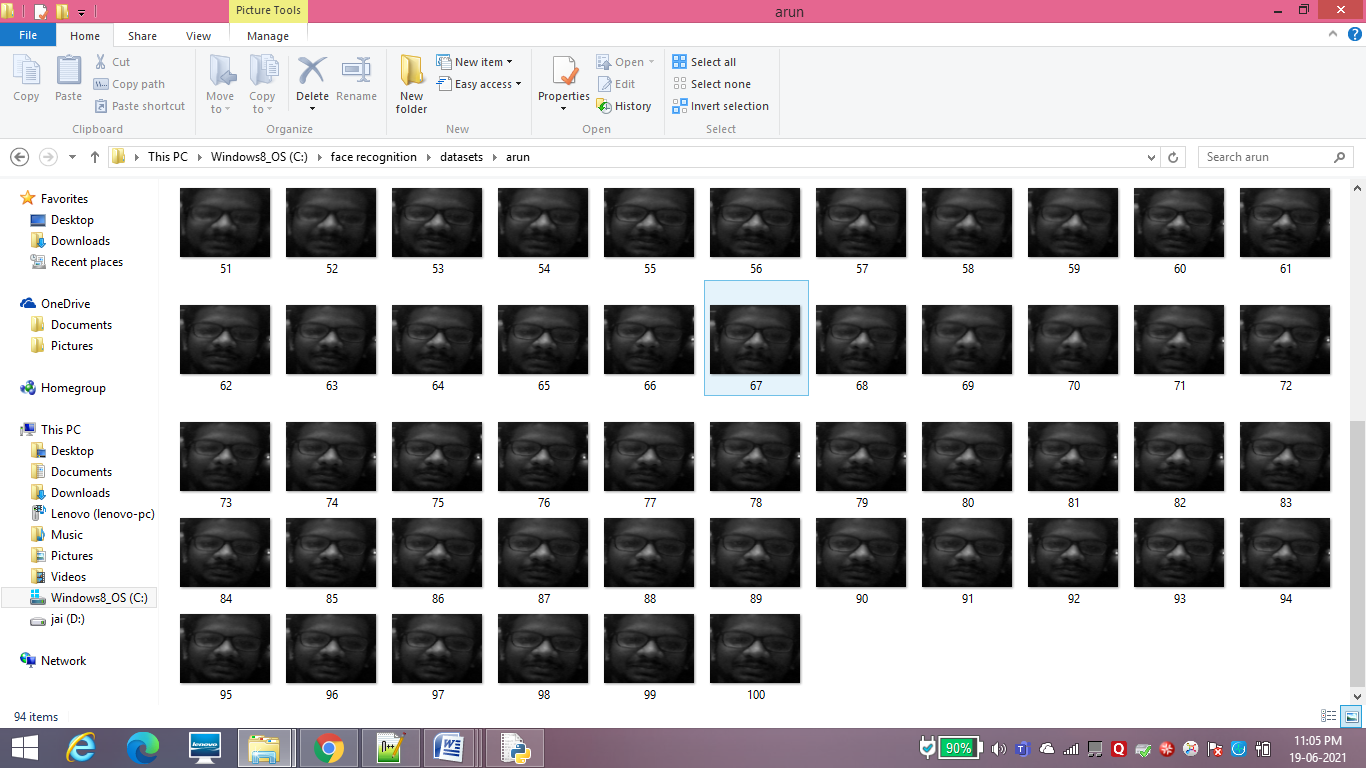
**Data base creation:**

****

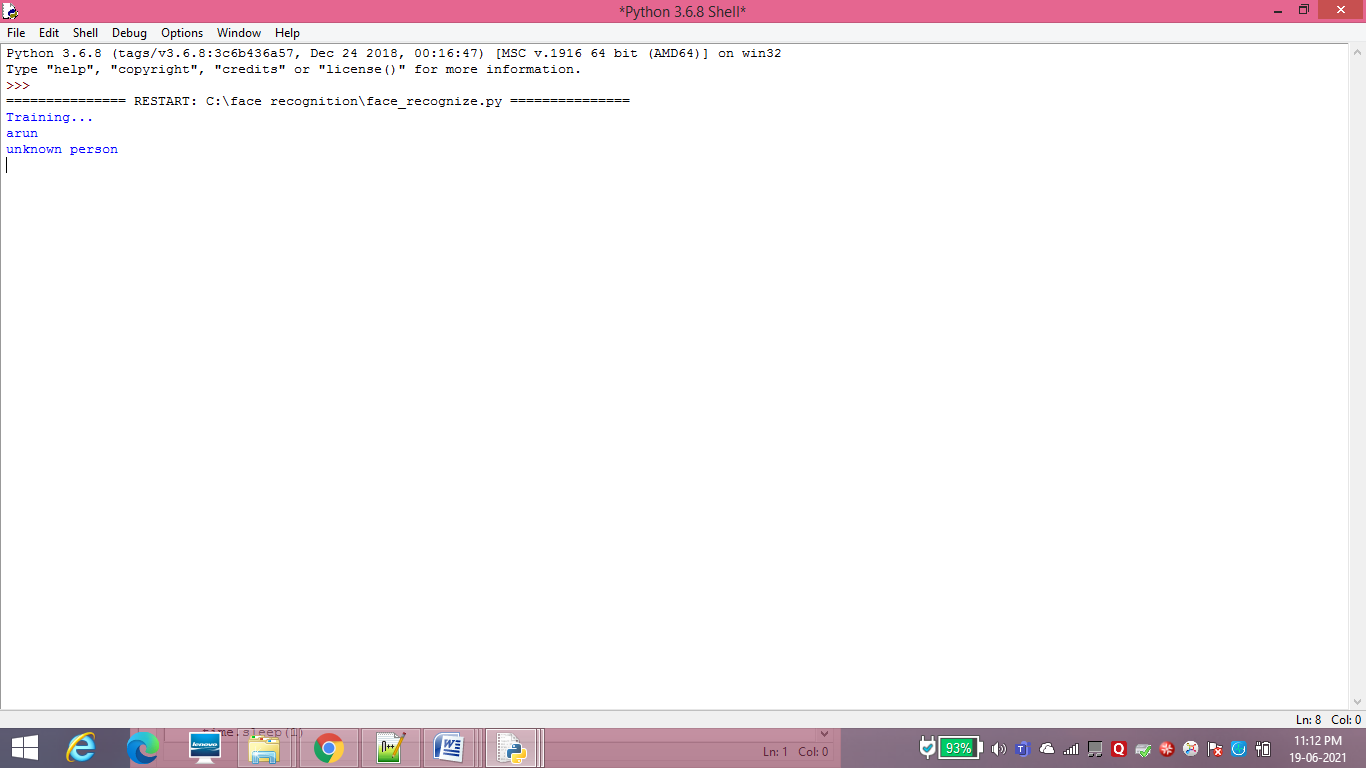
**accepting data in data base:**

****

**data stored in data base:**

****

**face recognizing from the input:**

****

**source code :**

**#Creating database**

import cv2, sys, numpy, os

import urllib.request

import numpy as np

haar\_file = 'haarcascade\_frontalface\_default.xml'

datasets = 'datasets' #All the faces data will be present this folder

sub\_data = 'lk'

####sub\_data = 'hai' #These are sub data sets of folder, for my faces I've used my name

path = os.path.join(datasets, sub\_data)

if not os.path.isdir(path):

os.mkdir(path)

(width, height) = (130, 100) # defining the size of images

face\_cascade = cv2.CascadeClassifier(haar\_file)

webcam = cv2.VideoCapture(0) #'0' is use for my webcam, if you've any other camera attached use '1' like this

##url="http://192.168.43.1:8080/shot.jpg"

# The program loops until it has 30 images of the face.

count = 1

while count < 101:

(\_, im) = webcam.read()

## imgPath=urllib.urlopen(url)

## imgNp=np.array(bytearray(imgPath.read()),dtype=np.uint8)

## im=cv2.imdecode(imgNp,-1)

gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 4)

for (x,y,w,h) in faces:

cv2.rectangle(im,(x,y),(x+w,y+h),(255,0,0),2)

face = gray[y:y + h, x:x + w]

face\_resize = cv2.resize(face, (width, height))

cv2.imwrite('%s/%s.png' % (path,count), face\_resize)

count += 1

cv2.imshow('OpenCV', im)

key = cv2.waitKey(10)

if key == 27:

break

**# Facerec.py**

import cv2, sys, numpy, os

import urllib

import numpy as np

import time

import os

from subprocess import call

import time

import os

import glob

import smtplib

import base64

from email.mime.image import MIMEImage

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import sys

gmail\_user = "fadinaleem@gmail.com"

gmail\_pwd = "dadofboi12"

FROM = 'fadinaleem@gmail.com'

TO = ['arunkarunanithi4121999@gmail.com'] #must be a list

def mail():

msg = MIMEMultipart()

time.sleep(1)

msg['Subject'] ="SECURITY"

#BODY with 2 argument

#body=sys.argv[1]+sys.argv[2]

body="THIS IS FROM your atm regarding to security"

msg.attach(MIMEText(body,'plain'))

time.sleep(1)

###IMAGE

fp = open("1.jpg", 'rb')

time.sleep(1)

img = MIMEImage(fp.read())

time.sleep(1)

fp.close()

time.sleep(1)

msg.attach(img)

time.sleep(1)

try:

server = smtplib.SMTP("smtp.gmail.com", 587) #or port 465 doesn't seem to work!

print ("smtp.gmail")

server.ehlo()

print ("ehlo")

server.starttls()

print ("starttls")

server.login(gmail\_user, gmail\_pwd)

print ("reading mail & password")

server.sendmail(FROM, TO, msg.as\_string())

print ("from")

server.close()

print ('successfully sent the mail')

except:

print ("failed to send mail")

size = 4

haar\_file = 'haarcascade\_frontalface\_default.xml'

datasets = 'datasets'

print('Training...')

# Create a list of images and a list of corresponding names

(images, labels, names, id) = ([], [], {}, 0)

for (subdirs, dirs, files) in os.walk(datasets):

for subdir in dirs:

names[id] = subdir

subjectpath = os.path.join(datasets, subdir)

for filename in os.listdir(subjectpath):

path = subjectpath + '/' + filename

label = id

images.append(cv2.imread(path, 0))

labels.append(int(label))

id += 1

(width, height) = (130, 100)

# Create a Numpy array from the two lists above

(images, labels) = [numpy.array(lis) for lis in [images, labels]]

# OpenCV trains a model from the images

# NOTE FOR OpenCV2: remove '.face'

model = cv2.face.FisherFaceRecognizer\_create()

model.train(images, labels)

# Part 2: Use fisherRecognizer on camera stream

face\_cascade = cv2.CascadeClassifier(haar\_file)

##with open("1.txt", mode='a') as file:

webcam = cv2.VideoCapture(0)

##url="http://192.168.43.1:8080/shot.jpg"

while True:

(\_, im) = webcam.read()

## imgPath=urllib.urlopen(url)

## imgNp=np.array(bytearray(imgPath.read()),dtype=np.uint8)

## im=cv2.imdecode(imgNp,-1)

gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:

cv2.rectangle(im,(x,y),(x+w,y+h),(255,255,0),2)

face = gray[y:y + h, x:x + w]

face\_resize = cv2.resize(face, (width, height))

#Try to recognize the face

prediction = model.predict(face\_resize)

cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)

if prediction[0]<100:

#port.write('B')

print (names[prediction[0]])

cv2.putText(im,names[prediction[0]],(x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN,1,(0, 255, 0))

if names[prediction[0]]=='' or '':

print(names[prediction[0]])

else:

print("unknown person")

cv2.imwrite('1.jpg',im)

mail()

a=input("Enter the pin:")

if a=="2000":

print("Correct user")

else:

cv2.putText(im,'Scanning',(x-10, y-10), cv2.FONT\_HERSHEY\_PLAIN,1,(0, 255, 0))

print("unknown person")

cv2.imwrite('1.jpg',im)

mail()

a=input("Enter the pin:")

if a=="1234":

print("Correct user")

cv2.imshow('OpenCV', im)

key = cv2.waitKey(10)

**9. CONCLUSION**

To avoid ATM robberies and provide security for ATM , To secure such a complex system will be even more difficult than design it. And now people just begin to discuss some issues of ATM security. It will provide some experience for us to implement security services in ATM network.

**REFERENCE**

H. R. Babaei, O. Molalapata and A. A. Pandor, Face Recognition Application for Automatic Teller Machines (ATM), in ICIKM, 3rd ed. vol.45, pp.211-216, 2012.

Aru, O. Eze and I. Gozie, Facial Verification Technology for Use in ATM Transactions, in American Journal of Engineering Research (AJER), [Online] 2013, pp. 188-193, Available:http://www.ajer.org/papers/v2 (5)/Y02501880193.pdf ¬ K.

J. Peter, G. Nagarajan, G. G. S. Glory, V. V. S. Devi, S. Arguman and K. S. Kannan, Improving ATM Securityvia Face Recognition, in ICECT, Kanyakumari, 2011,vol.6, pp.373-376.

E. Derman, Y. K. Gec¸ici and A. A. Salah, Short Term Face Recognition for Automatic Teller Machine (ATM) Users, in ICECCO 2013, Istanbul, Turkey, pp.111-114

A. Ross and A. Jain, Information Fusion in Biometrics,in Pattern Recognition Letters, vol.24, pp.2115-2125,2003.

Ing. Ibrahim Nahhas, Ing. Filip Orsag, Ph.D “Real Time Human Detection And Tracking”, Bruno University of Technology.

[5] Mohamed Hussein, WaelAbd-Almageed, Yang Ran, Larry Davis “Real-Time Human Detection in Uncontrolled Camera Motion Environments” Institute for Advanced Computer Studies University of Maryland.

[6] Wongun Choi, Caroline Pantofaru, Silvio Savarese “Detecting and Tracking People using an RGB-D Camera via Multiple Detector Fusion” Electrical and Computer Engineering, University of Michigan, Ann Arbor, USA.

[7] Rupesh Mandal, Nupur Choudhury “Automatic video surveillance for theft detection in ATM machine : An enhanced approach”, Computer Science and Engineering Sikkim Manipal Institute of Technology India.

[8] EunSom Jeon, Jong-suk Choi, Ji Hoon Lee, Kwang Yong Shin, YeongGon Kim, ToanThanh Le And Kang Ryoung Park,” Human Detection Based on the Generation of a Background Image by Using a Far-Infrared Light Camera”, Division Of Electronics And Electrical Engineering, Dongguk University