Abstract

This work is devoted to the development of an information system for pattern recognition. Methods for developing applications on OpenCV technology in the programming language Python are discussed in detail. Based on the features of the language described in the work, the application itself is developed on this platform and a new cascade is generated for further use. Detailed methods for recognizing faces when developing an application are described.

Main goals:

As an example, based on the OpenCV library in a virtual environment, demonstrate the cascade workout and develop an appropriate algorithm for the application. This paper considers three main tasks:

- Development of an algorithm for recognizing faces
- Development of an algorithm for object recognition
- Training of the cascade for further use in the form of an .xml document

Contents

INTRODUCTION

Technological progress has brought humankind to evolutionary stage, where complex tasks became much easier, rapid to produce. Smartphones, smart cars, smart houses and other smart gears were designed to help people to overcome complexity of everyday tasks.

However, this technological progress brought many complications, especially associated with securing data, authentication systems, threats to data in cloud and many more issues; a billionaire, entrepreneur and physicist Elon Musk suggest that this threat is only the beginning and by creation of AI, there will be much more consequences until humanity will be led by cyborgs. According to forecast provided by Gartner, security systems industry market will grow up almost up to 100 billion by the end of this year.

Therefore, from the perspective of computer engineer one can assume that the market of cybersecurity is growing and young. The main goal of a computer engineer – as a cybersecurity professional is to try to prevent these kinds of cyber-attacks, hackings, scam and phishing. There are plenty of programming tools for cybersecurity professionals to do so, but this work mainly concentrates on face and image detection.

At least one element from security systems technology appears almost everywhere. These include all sorts of camera systems, encryption methods in authentication systems, biometric data collection and implementation of specialized customized encryption methods, such as hashing algorithms and much more.

Let us concentrate on analyzing the time complexity of existing algorithms and APIs for camera systems technology – computer vision tools. One of the most broadly used APIs is OpenCV, which was created for multiple programming languages and includes methods for image processing, resizing, displaying and many other various complex operations on image.

CHAPTER 1. STATEMENTS OF FACE DETECTION PROBLEM MOTIVATION BEHIND FACE DETECTION PROBLEM

1.1. Face detection definition

Definition of face detection problem can vary from source to source but conceptually the definitions have very close meanings. Face detection is a computational technology aimed towards various applications but mainly it finds its application in detection of human faces, in both digital images and digital video-captures. It is also used in other fields, such as criminology and psychology for capturing and analyzing facial features to identify the psychology and character of person.

But, in computational engineering face detection problem is also connected with the problem of image processing so, the definition of image processing also should be given.

From perspective of imaging science, image processing is processing (which include current fields: Image analysis, Image sharpening, Image smoothing, Multidimensional systems, Near sets, Photo manipulation, Image compression) of digital images, videos and other raster images using mathematical operations and skills, signals processing for which the input is vector with image data, or with coordinates of current image, or even a video, with a photograph or a video frame; and output is a processed set of coordinates or an image, or even a video.

So as one can deduct from above said the problem of face detection is one of the most complex tasks in computer science and electrical engineering which involves both the skills of programmer (algorithmic and software part) and electrical engineer (hardware and physical installment part).

This thesis however assumes that the tools and physical installments have already been done, and the missing part is software. As hardware one can assume

that web camera and standard monitor is being used, and this whole work is mainly concentrated on software.

The whole work is dedicated to creation of a cascade using which one can detect different types of objects, such as cellphones. The work also covers some methods of face detection.

1.2. Some methods of image recognition

For optical pattern recognition, one can apply the methods of sorting the view of the object at different angles (from which perspective the camera is recording), scales, displacements, etc. If the problem is to recognize letters or some symbols, one needs to sort through the font, font properties, and so on.

The second approach is to find the outlier of an object and investigate its properties (connectivity, presence of angles, etc.).

Another approach is to use artificial neural networks. This method requires either a large number of examples of a recognition problem (with the correct answers and wrong ones) or a special structure of a neural network that takes into account the specific nature of the problem. This is one of the most developing fields nowadays.

1.3. Statement of pattern recognition problem

First, one should consider the informal statement of pattern recognition problem. The problem of image recognition has acquired an outstandingly significant importance in conditions of overloads of datasets. One can assume that with current technological progress the loads of information are going to grow and

reach a whole another level, and as predicted in future, datasets will take space up-to cities.

Now suppose a certain person gets a problem and when a person does not cope with linearly consistent understanding of incoming messages-such as coordinates fetched from multiple sensors (such as web camera itself), resulting in his brain switching to the mode of simultaneity of perception and thinking, to which such recognition is inherent.

It is no accident, thus, the problem of image recognition turned out to be in the field of interdisciplinary research - including the connections with the work on creating artificial intelligence, and the creation of technical embedded systems for image processing and its recognition, which attracts more attention.

These tasks are often in demand in such market as car number detection in road accidents, analog and digital installed cameras in robberies in stores, banks and enterprise facilities. The example scenario of such task can be, for example, when car is crossing or passing a street by traffic light signals. Recognizing the color of a lighted traffic light and knowing the rules of the road makes it possible to make the right decision about whether or not to cross the street. One more example of such task can be a robbery in a store and finding a person behind this theft.

In contrast with classical statement of problem, formal statement of image detection problem can be defined in a bit of a different direction. Classical statement is as follows: the image recognition and its processing is the assignment of source data (which can be extracted from sensors and is taken as an input array) to a specific class by extracting the necessary and essential characteristics that characterize this data from the total population of nonessential data. In other words, one has to extract the raw data from the given dataset and filter all the necessary part from the rest. Data cleaning is one of the most essential parts of data science and image recognition problem.

When setting recognition tasks, it is recommended to use the mathematical language – for formulation current problem; in contrast to the theory of artificial neural networks, where to get to the solutions the basis is to obtain the result by experiment – in image recognition professionals are striving to replace the experiment with logical reasoning and mathematical evidence.

The classical statement of the problem of pattern recognition is as follows:

- Given a set of objects.
- It’s necessary those objects appropriately.
- A set is then represented by subsets, which are called classes.
- Given: information about classes, description of the entire set and description of information about the object, belonging to a certain class is unknown.
- It is required, based on available information about classes and object description, to establish which class this object belongs to- in other words it is required to classify the objects based on the information available.

As the main algorithm of image recognition problem is described at a very low level, one should also give a more formal statement of current problem – in an explicit mathematical notions, involving functions and plane.

Before going to further formalizing, one has to consider the meaning of monochrome images. According to one definition from Wikipedia, monochrome image is an image containing light of one color (wavelength) and it is perceived as one hue (as opposed to a color image containing different colors).

As example, monochrome images, are drawings in ink, pencil or charcoal, black and white photographs, images on the screen of black and white TVs or computer monitors (regardless of the true color of their glow).

Monochrome images are considered most often in image recognition problems, which makes it possible to treat an image as a function on a plane.

So now, if one considers a point set in the plane ***T***, where the function ***f (x, y)*** expresses its characteristic - brightness, transparency, optical density at each point of the image, then such a function is Formal image recording.

The set of all possible functions ***f(x, y)*** on the plane ***T*** is a model of the set of all images ***X***. Introducing the notion of similarity between images, one can define the task of recognition. The specific form of such a formulation depends heavily on the subsequent steps in recognition if one considers this approach.

CHAPTER 2. NECESSARY TOOLS AND THEIR ANALYSIS

2.1. Tools, programming language and its API

One should consider that the crucial part that is interesting for people, in fact is not the part of getting the input data from sensors or outputting it in a file, or storing data in databases, but it's in fact the algorithm of checking and contrasting the features of given image with the other ones.

One of the most broadly used APIs is OpenCV, which was created for multiple programming languages and includes methods for image processing, resizing, displaying and many other various complex operations on image. There are tons of tools for Image Processing, but OpenCV has been chosen over others, because of simplicity of its implementation. Its implementation can be done in multiple programming languages, such as Python, C#, Java and even JavaScript.

However, technology itself is not that simple and face detection problem goes deep in thickets of Machine Learning. According to the official library documentation whole process of face recognition is based on one of four current algorithms: Discrete Adaboost , Real Adaboost, Gentle Adaboost and Logitboost or combination of those algorithms. The map of algorithms is shown in figure 2.1. These algorithms have a boost word in common so they are all boosting machine learning meta-algorithms that are specifically designed to decrease bias by converting weak learners to strong ones. As one can understand these meta-algorithms are based on weak learner, whereby the latter is by its nature a classifiers.

Figure 2.1. Map of boosting Algorithms

In order to understand the principle of its work one should consider those meta-algorithms Discrete Adaboost –adaptive boosting algorithm.

All the training operations must be executed in Linux environment, so in order to be able to use the training materials it is required to download and install virtual environment. There are lots of free virtual environments, but the choice has fallen on VirtualBox because the ease of its usage and its reputation – checked product.

VirtualBox or Oracle VM VirtualBox is a software product for virtualization for Microsoft Windows, Linux, FreeBSD, Mac OS X, Solaris / OpenSolaris, ReactOS, DOS and other operating systems. The program **Virtualbox** was created by Innotek using the Qemu source code. The first publicly available version of VirtualBox appeared on January 15, 2007. In February 2008, Innotek was acquired by Sun Microsystems, while the VirtualBox distribution model did not change. In January 2010, Sun Microsystems was absorbed by Oracle, the distribution model remained the same.

There are plenty of possibilities provided by Oracle VM VirtualBox:

- The software is cross-platform
- It is modular
- When host devices become available for guest operating systems (only in the proprietary version) USB 2.0 is supported
- Support for 64-bit guest systems (since version 2.0), even on 32-bit host systems (since version 2.1, this requires support for virtualization technology by the processor)
- Support for SMP on the side of the guest system (starting with version 3.0, for this it is necessary to support the technology of virtualization by the processor)

- Built-in RDP-server, as well as support for client USB-devices over RDP (only in the proprietary version)
- Experimental support of hardware 3D acceleration (OpenGL, DirectX 8/9 (using the wine code) (only in 32-bit Windows XP, Vista, 7 and 8), for guest DOS / Windows 3.x / 95/98 / ME support Hardware 3D acceleration is not provided)
- Support for images of VMDK (VMware) and VHD (Microsoft Virtual PC) hard disks, including snapshots (since version 2.1)
- ISCSI support (only in the proprietary version)
- Support for virtualization of audio devices (emulation of AC97 or SoundBlaster 16 or Intel HD Audio to choose from)
- Support for various types of network interaction (NAT, Host Networking via Bridged, Internal)
- Support for a chain of saved states of the virtual machine (snapshots), to which can be rolled back from any state of the guest system
- Support for Shared Folders for simple file exchange between host and guest systems (for guests of Windows 2000 and later, Linux and Solaris)
- Support for the integration of desktops (seamless mode) host and guest operating system
- OVF / OVA format support
- There is a choice of interface language (Russian-language interface is also supported)
- The basic version is fully licensed under the GNU GPL license, there are no restrictions on the use, which basically eliminates the problem of privacy.

In conclusion VirtualBox has the functionality to support the needed system for objects training if needed.

2.2. Face recognition technology

There are some remarkable examples of implementation of face detection technology and it is worth discussing them too.

- One of the first motivating tasks of image recognition was its concrete implementation, specifically on human faces. In fact, the task of identifying and recognizing faces itself is one of the first practical tasks, if not the first that stimulated the formation and development of the theory of object recognition and identification. There are nine categories of objects that correspond to the gnostic regions and cause visual images:
- Objects that can be manipulated by people (such as cup, keys, clock, etc.);
- Objects that can be partially manipulated by people (such as cars, materials, etc.);
- Objects that cannot be manipulated (trees, buildings, etc.);
- Faces;
- Expressions of faces;
- Living beings (such as animals, human figures from different perspectives);
- Printed characters (such as letters, symbols, signs);
- Handwritten images (such as italic or handwritten letters);
- Characteristics and location of light sources (regarded when angles are taken into account and sun, moon or lightbulb can be considered as an example).

Interest in the procedures underlying the process of pattern of recognition and face recognizing has always been significant, especially in connection with increasing practical needs: security systems, hacker invasions, verification in secured buildings, forensic expertise, teleconferences et al. Despite the clarity of the worldly fact that a person well identifies people's faces, in other words, there is no exact algorithm to follow, it is not at always clear how to teach a computer to conduct this procedure, including how to decode and store digital images of individuals. Even

less clear are estimates of the similarity of individuals, including their complex processing. There are several areas of research into the problem of face recognition:

- Neuropsychological models;
- Neurophysiological models;
- Information - procedural models;
- Computer models of recognition.

The problem of face recognition was considered in the early stages of computer vision. A number of companies for more than 40 years are actively developing now automatic systems for recognizing human faces: Smith & Wesson (ASID system - Automated Suspect Identification System); ImageWare (FaceID system); Imagis, Epic Solutions, Spillman Co, Miros (Trueface system); Vissage (Vissage Gallery system); Visionics (FaceIt system), Google (TensorFlow API) and last but not least OpenCV.

Face recognition technologies allow automatic search and recognition of faces in graphic files and video stream.

Main characteristics of face recognition problems are as follows: The ability to search and recognize multiple persons;

- Resistance to changes in the hairstyle, the presence / absence of a mustache and beard, glasses (except for sun protection), age changes (except for children), turns (up to 30 degrees)
- Practically linear scalability of performance when installed on multiprocessor, multi-core systems and computer clusters;
- The ability to bind to images of keywords (for example, "politician", "businessman", etc.) and a brief description for further automatic classification of the processed content;
- The possibility of multi-frame analysis of the video stream, which improves recognition accuracy;

- Output recognition results in plain text, or an XML document that includes information about the location and size of the detected persons, recognition results and timestamps;
- Weak dependence of the speed of work on the size of the used gallery of persons. For example, with an increase in the gallery from 100 to 1000 persons, the work speed decreases by less than 10%;
- Work with video in real time.

2.3. The reasons behind choosing Python

Python is a unique and simple high-level programming language that has its own philosophy. Python developers adhere to a certain programming philosophy called "The Zen of Python" ("Zen Python"). Zen philosophy's of is Tim Peters. As the text of the Zen is states:

- Code is beautiful rather than ugly.
- More explicit is better than implicit or not at all.
- KISS (keep it simple stupid) concept is far better than complicated.
- But take into account that complicated is better than confusing.
- Flat is better than embedded.
- Sparse –yes, no dense.
- Readability matters.
- Special cases are not so special if they do not violate the rules.
- In this case, practicality is more important than impeccability.
- Errors should never be skipped or hushed.
- No guess if ambiguous is met.
- There must be one - and, preferably, only one - an obvious way to do this.
- Although it may not be obvious at first, but for Dutch it is.
- Now better than never.

- Although sometimes never often better than right now.
- If the implementation is difficult to explain, there is something wrong with the idea. The idea is bad.
- If the implementation is easy to explain, the idea is definitely good.
- Namespaces are a great thing! We will make them more!

Python is a powerful programming language, as it supports several programming paradigms, including structural, object-oriented, functional, imperative, and aspect-oriented. The main architectural features are dynamic typing, automatic memory management (garbage collection in other high-level programming languages – C#, and Java), full introspection, exception handling mechanism, support for multi-threaded computing and convenient high-level data structures. The code in Python is organized into functions and classes that can be combined into modules (they in turn can be combined into packages)

Before diving into functional programming in python it's necessary to consider meaning of functional programming itself.

Functional programming is a section of computer science and programming paradigm in which the calculating processes are being interpreted as computing the values of functions in a mathematical sense, formalized versions (as opposed to functions as subroutines in procedural programming or other programming paradigms).

It contrasts the paradigm of imperative programming, which describes the computation process as a sequential change – systematic change of states (in a sense similar to that in the theory of automata and formal languages). Of course, if it is necessary, in functional programming, the entire set of consecutive states of the computational process is represented and depicted explicitly, for example, as a list.

Functional programming involves bypassing the calculation of the results of functions from the source data and the results of other functions, and does not imply an explicit storage of the program state. Accordingly, it does not imply the variability

of this state (in contrast to the imperative one, where one of the basic concepts is a variable that stores its value and allows it to be changed as the algorithm executes).

In practice, the conceptual difference between a standard mathematical function and the concept of “function”- “method” in imperative programming is that imperative functions can rely not only on arguments, but also on the state of variables external to the function, and also have side effects and change the state of external variables and even the objects themselves.

Thus, in imperative programming, when one calls the same function with the same parameters, but at different stages in the execution of certain algorithm, one can get different data at the output due to the effect on the state function of the variables. In addition, in the functional language, when one calls a function with the same arguments, it is outputting always the same result: the output depends only on the input. This allows the program execution environments in functional languages to cache the results of functions and call them in an order not determined by the algorithm and parallelizing them without any additional actions on the part of the programmer (which is provided by functions without side effects - pure functions).

Lambda calculus is the basis for functional programming, many functional languages can be considered as a "superstructure" over them. Python 3 has a powerful tool called Lambda expressions, and example of a lambda expression might be the following snip of code:

```
>>> addOneToArgument = lambda x: x + 1
>>> addOneToArgument (26101993)
26101994
```

Some concepts and paradigms are specifically aimed to functional programming and are largely foreign to imperative programming (including object-oriented programming itself). However, programming languages are usually a

hybrid of several programming paradigms, so "mostly imperative" programming languages can use any of these concepts.

2.4. OpenCV overview

OpenCV (English Open Source Computer Vision Library, library of computer vision with open source code) is a library of algorithms for computer vision, image processing and numerical algorithms of general purpose with open source. Implemented in C / C ++, also developed for Python, Java, Ruby, Matlab, Lua and other languages. It can be used freely for academic and commercial purposes - it is distributed under the terms of the BSD license.

Applications of OpenCV library are like standards of computer vision software, just like in case of OpenGl, used in different purposes. One of the main applications of OpenCV is to approve a common standard computer vision interface for applications in this field. To help increase the number of such applications and create new models of using PC.

To make Intel platforms attractive to developers of such applications due to additional acceleration of OpenCV with Intel® Performance Libraries (Now include IPP (low-level libraries for processing signals, images, and media codecs) and MKL (special version of LAPACK and FFTPack)). OpenCV is able to automatically detect the presence of IPP and MKL and use them to speed up processing.

There are few core libraries that are supported on one platforms, so one should know which platform is supported and which one is not supported. Due to flexibility of OpenCV, it is supported almost for every platform, including mobile platforms- Android from Google and iOS from Apple.

The libraries themselves are layered out like shown in table 2.1:

Table 2.1. Libraries for platforms

Name of library/platform	Description
Microsoft Windows	compilers are Microsoft Visual C ++ (6.0, .NET 2003), Intel Compiler, Borland C ++, Mingw (GCC 3.x)
Windows RT	ported to ARM by Itseez
Linux	GCC (2.9x, 3.x), Intel Compiler: "./configure-make-make install", RPM (spec file included in the package).
Mac OS X	GCC (3.x, 4.x)
Android	Java and Android Development kit
IOS	Used C and "lightweight" C ++. Pragmas and conditional compilation are very limited.

There are also tools provided for construction of corresponding graphical interface counted in table 2.3.1 for video capturing. The reason of such need in video capturing tools and its API is that OpenCV takes video or image capture as an input. In mathematical terms it should have an input data as the array, tuple or set of ***(x,y,z)*** coordinates and output the same set of processed data. The description of all necessary tools is shown in table 2.2

Table 2.2. Tools for GUI for video capturing

Platform	Tool
MS Windows	DirectShow, VfW, MIL, CMU1394.
Linux	V4L2, DC1394, FFMPEG.
Mac OS X	QuickTime

The format of documentation is provided in either PDF or plain static HTML format accordingly.

It is also required to consider all the necessary libraries provided by OpenCV for the project demonstrated in this thesis.

As in many libraries there are many versions of OpenCV, however the noticeable change has occurred in after version 2.2. After version 2.2, the library was fully reorganized. Instead of universal modules, including cxcore, cvaux, highGUI and others, several compact modules with a narrower specialization were created in aim to achieve high coherence and low coupling goals. As documentation of OpenCV states:

- **opencv_core** – has the main functionality. It includes basic structures and structural calculations (math functions, random number generators), it also includes linear algebra library, DFT, DCT, I / O for XML and YAML, and so on.
- **opencv_imgproc** – is basically used for standard image processing (processing includes image filtering, geometric transformations, transformation of color spaces, etc.).
- **opencv_highgui** - simple UI elements are provided by this library, input / output images and video.
- **opencv_ml** – meta-algorithms for machine learning models (for example SVM, decision trees, training with incentives).
- **opencv_features2d** – recognition and description of flat primitives (such as SURF (English), FAST and others, including a specialized framework).
- **opencv_video** – it is used for motion analysis and object tracking (such as optical flow, motion patterns, background elimination).

- **opencv_objdetect** – provides the detection of objects on the given image or video as stated in documentation (finding people, objects using the algorithm of Viola-Jones, recognizing people HOG and et cetera).
- **opencv_calib3d** – this object is used for camera calibration, stereo matching search and 3D data processing elements.
- **opencv_flann** – this is a library for quick search of nearest neighbors (FLANN 1.5) and OpenCV wrappers.
- **opencv_legacy** – it is used for obsolete code, stored for backward compatibility.
- **opencv_gpu** – this library is used for acceleration of some OpenCV functions due to CUDA, created with the support of NVidia.

2.5. Mathematics of algorithm

The process of detecting faces itself can serve as a concrete implementation of discrete adaptive boosting algorithm[11], where faces are compared with given face by more than 100.000 features-criterions and are trained repeatedly by set of weak learners in order to output a strong learner. Let us discuss the algorithm and its math behind the discrete adaboost.

Consequently, analysis of algorithm begins with considering an input of sample tuple:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Similar to the definition of inputs, one should define also the outputs:

$$Y = (y_1, y_2, y_3, \dots, y_n)\ ,$$

where $y \in \{-1, 1\}$. This algorithm is based on weighing and renormalizing the weights, so initial weights are also required:

$$X = (w_{11}, w_{21}, w_{31}, \dots, w_{n1}),$$

where they are assigned to $w_{11} = \frac{1}{n}, w_{21} = \frac{1}{2}, w_{31} = \frac{1}{3}, \dots w_{n1} = \frac{1}{n}$.

Error function is required, so corresponding exponential one is as follows:

$$E(f(x), y, i) = e^{-y_i f(x_i)}.$$

It is necessary to assume that weak learners are assigned as follows:

$$h: x \to [-1,1].$$

Algorithm of discrete adaptive boosting algorithm works as follows:

at each iteration of t in (1,T), it's required to:

- Train a weak classifier as follows:

$$h_t = min_h \sum_{i=1}^{n} w_{t-1}(i), \qquad assuming\ that y_i \neq h(x_i))$$

- Calculate the error of h_t : $er_t = \sum_{i=1}^{n} w_{t-1}(i),\ assuming\ that\ y_i \neq h(x_i))$
- Set $\propto_t = \frac{1}{2} \log(\frac{1-er_t}{er})$
- If $y_i = h_{t+1}(x_i)$, then:
  - $w_{t+1}(i) = \frac{w_t}{2*(1-err_t)}$;

else if $y_i \neq h_{t+1}(x_i)$, then: $w_{t+1}(i) = \frac{w_t}{2*err_t)}$.

Strong classifier H(x) will serve as an output of current schema. It can be found, as follows:

$$H(x) = \sum_{k=1}^{T} \alpha_k * h_k(x)$$

So, as it can be seen above, this adaboost is machine-learning algorithms designed to work in conjunction with other learning algorithms in order to improve their performance. Real Adaboost, Gentle Adaboost and Logitboost have the similar working principle as Discrete Adaboost. Let us implement the library(API) that uses this algorithm for processing and detecting faces.

2.6. Haar-Cascade model

Features of Haar are features of a digital image used in pattern, object and face recognition. Their name is due to an intuitive resemblance to the Haar wavelets-math

function that allows to analyze different frequency component of data. The Haar attributes were used in the first real-time face detector.

Historically, algorithms that work only with image intensity (for example, the RGB value is in each pixel) have a large computational complexity. Practically large amount of brute-force face detection algorithms is either $O(n^c)$,where c is a constant or even $O(c^n)$. Importance of algorithmic complexity comes when either n or c grow as shown on figure 2.2.

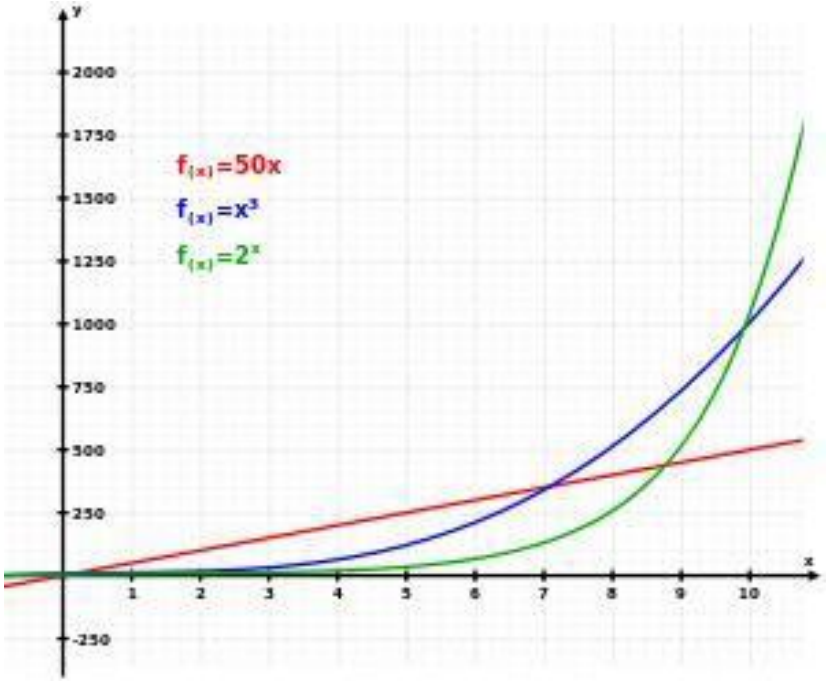


Figure 2.2. Comparison of three algorithmic complexities

Viola and Jones adapted the idea of using Haar wavelets and developed what was called Haar's properties. The Haar feature consists of adjacent rectangular areas. They are positioned on the image, then the intensities of the pixels in the areas are summed, after which the difference between the sums is calculated. In very basic notions, this difference will be the value of a certain characteristic, a certain size, in a certain way positioned on the image. In the method of Viola-Jones, the basis is made up of the Haar primitives, which are a breakdown of a given rectangular region into sets of diverse rectangular sub regions. As it is depicted on figure 2.3, it is

necessary to find the intersection between the one feature and the other using overlay.

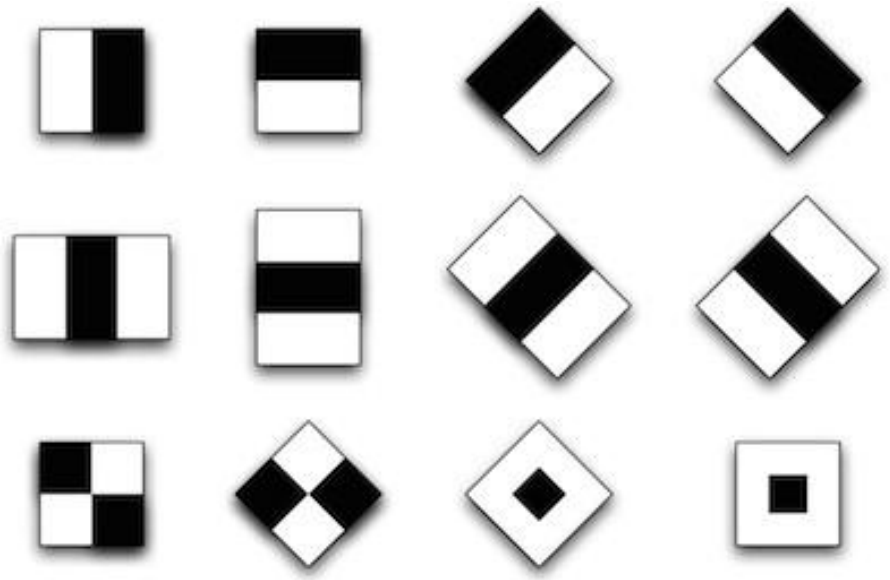


Figure 2.3. Overlaying different properties

For example, consider a database with human faces. Common to all images is that the area in the eye area is darker than the area around the cheeks. Therefore, the common hallmark of Haar for individuals is the two adjacent rectangular regions lying on the eyes and cheeks. In the detection phase in the Viola-Jones method, the window of the set size moves along the image, and for each area of the image over which the window passes, the Haar sign is calculated. The presence or absence of an object in the window is determined by the difference between the value of the characteristic and the threshold being learned. Since Haar's signs are not suitable for training or classification (the quality is slightly higher than for a random normally distributed quantity), more features are needed to describe the object with sufficient accuracy. Therefore, in the method of Viola-Jones, the Haar signs are organized into a cascade classifier.

The key feature of Haar's signs is the highest, in comparison with the remaining signs, speed. Haar's attributes can be calculated for a constant time (about 60 processor instructions per attribute from two areas) in some images.

Now assume that there is a rectangular frame and inside of that frame is a picture that contains or does not contain a face, as shown in figure 2.4.

Figure 2.4. Necessary image (Photo is taken from gamer's free online resource)

In order to step into the main stage – face detection stage, one has to resize and apply the black-and-white filter for decreasing the algorithmic complexity of further processing as shown in figure 2.5.

Figure 2.5. Black and white filter applied version

The size of given picture is **720X405=291 600** points. So there are up 291 600 points in this picture, but one has to consider the size of a face according to HaarCascade. Usually, the size of a face is **20X20,** so as a result it is obvious that there will be **20** points decrease in the given frame. Now if one takes a whole window and presents it asks how a small rectangles of size **20X20** can walk around the whole picture of **720X405** then it is obvious that the following resulting points **(720-20)X(405-20) = 269 500** will be left**.** Given square must walk around the whole picture, and as it detects a face return true. A very brief algorithm is presented below:

1) get frame from pc
2) convert to points and store in a data type
3) walk through the points by 20X20 rectangle
4) IF face is detected
    a. show it
5) ELSE
    a. continue walking until detecting a face. The graphical representation of algorithm is shown as squares on the figure 2.5.

Figure 2.5. Visual representation of algorithm

Now if the algorithm is applied on the Figure 2.5 one can assume that the face will be found, as shown on Figure 2.6.

Figure 2.6. Visual representation of algorithm on a real-world example

As one can see, the facial features are depicted on a picture as a black rectangle. Similarly to the face detection one can deduce that object is done similarly if trained appropriately.

CHAPTER 3. PRACTICAL PART OF OBJECT DETECTION

3.1. Installation of necessary tools

In order to install the python environment one has to refer to its official website that provides the necessary installation packages depending on the operating system. Personal computer that this work is executed on is Windows based, so the installation package should be appropriate for Windows. As soon as the package has downloaded one must execute the package and install following the steps shown on the next windows. And the last step would be to insert python variable in environmental variables which can be executed through My Computer.

The next step is to install the required application-programming interface to the list of existing Python libraries. It is a little bit complicated to install all necessary libraries on Windows because sometimes **pip** command does not work appropriately for libraries like OpenCV. So in order to install them one must refer to the special website – repository of all Python libraries. It's necessary to find a desired library and download its **.whl** file as shown on figure 3.1.

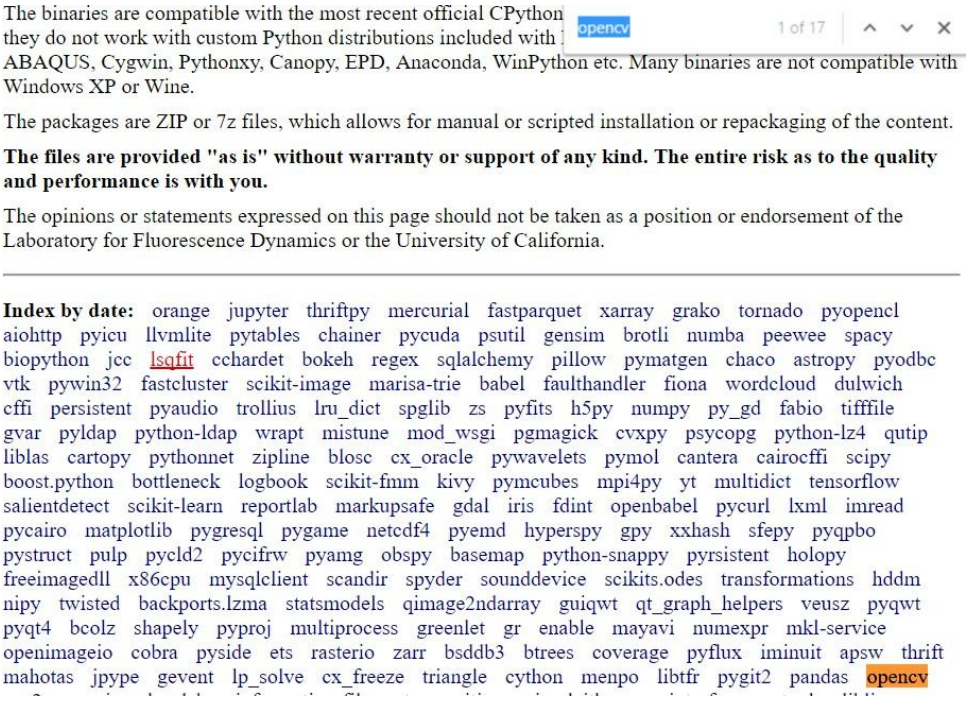


Figure 3.1.a: The necessary library

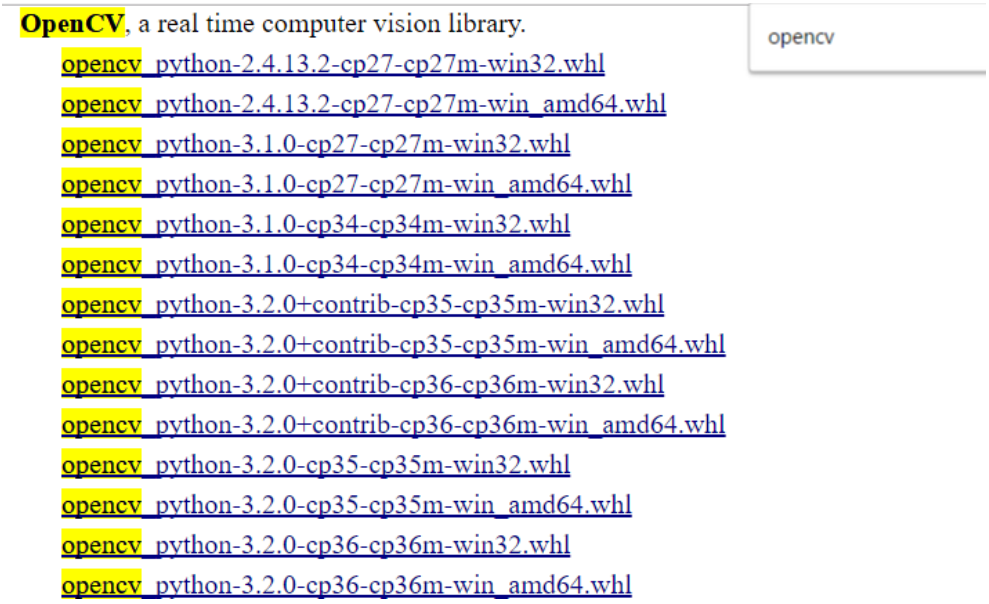


Figure 3.1.b: The necessary library

Now as the necessary library has been downloaded one needs to execute it using pip install command in command prompt in as shown on Figure 3.2.

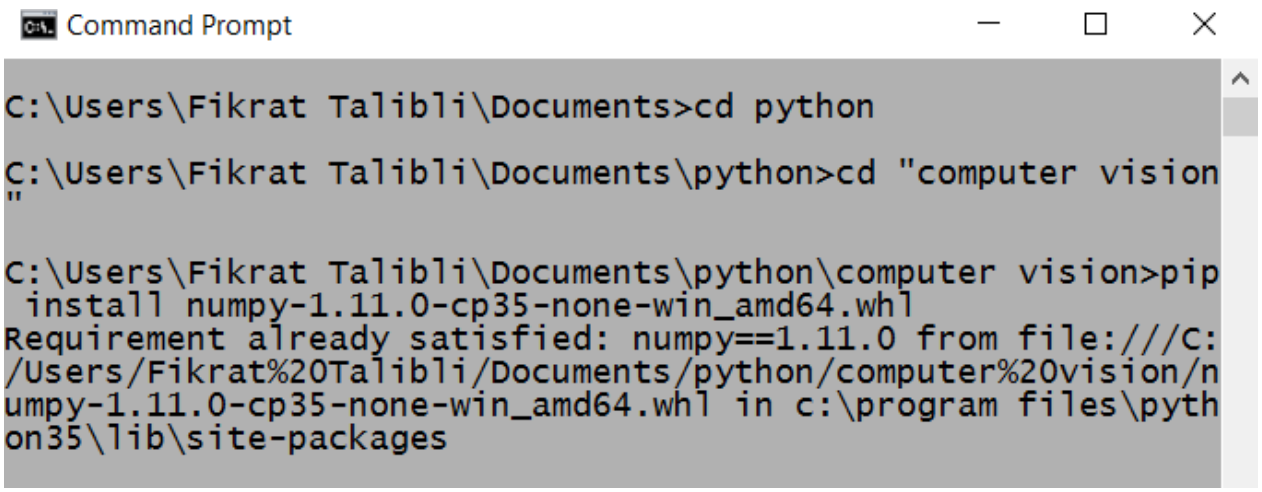


Figure 3.2. Installation of required package using **.whl** file

As the necessary libraries have been installed it is require to create a python source code file and populate it with necessary algorithmic code. Now it is required to go to the Start Menu, open IDLE, create a new source code file, and call it **comp_vision.py** as shown on figure 3.3.

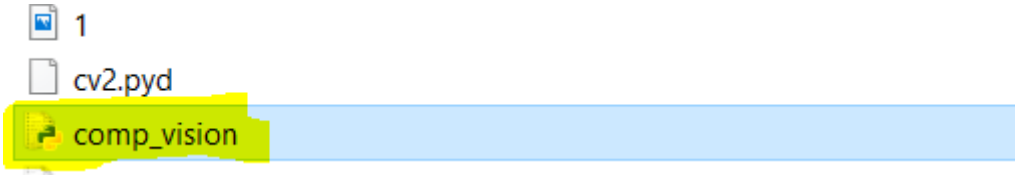


Figure 3.3. Source code file

Also one should consider downloading the Haar-Cascade from the open-source .xml data files from GitHub called haarcascade_frontalface_default.xml and haarcascade_eye.xml and place it right next to **comp_vision.py** file, as shown on figure 3.4.

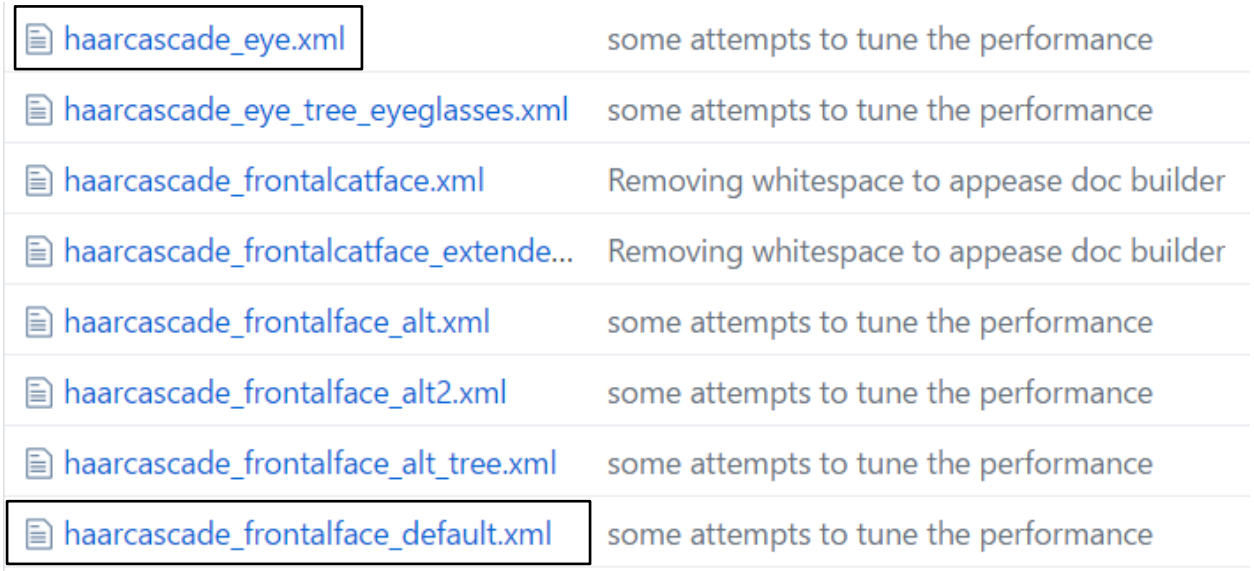


Figure 3.4.a. Necessary xml files from GitHub repo

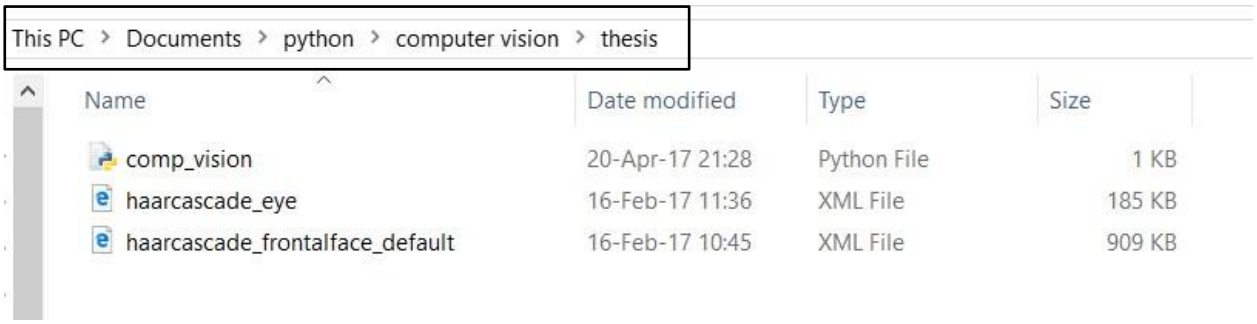


Figure 3.4.b. Necessary xml files from GitHub repo

Now it is necessary to populate given file with source code.

3.2. Main logic of program

One of the advantageous functionalities of this library is the possibility to detect any object with certain precision percentage using cascade files. These files can either be created manually or downloaded from OpenCV's official GitHub account. The decision whether to download or create new cascade entirely depends on the given task. There are some popular pre-created cascades that do not need to be downloaded, such as human-face, human-face with glasses, upper-body, human profile-face etc.

First it is required to consider a more general task - human face and eyes detection problem. This task can easily be solved using the cv2 library, and cascade.xml in order to find the face. This technology works on previously mentioned boosting algorithms. Full description of cascade is as follows: Haar feature-based cascade classifier is an effective object detection method and functionality proposed by Paul Viola[2] and Michael Jones in their paper, "Rapid Object Detection using a *Boosted Cascade* of Simple Features" in 2001.

First one has to make sure that one has corresponding files and everything is installed. Then it is necessary to discuss the full algorithm of detecting faces on Python looks as follows in pseudo-code:

1) *# import necessary libraries*
2) *# read corresponding cascades for face and eyes*
3) *# display capturing face from camera*
4) *#at every iteration (loop is infinite, it breaks out only when ESC is #pressed) take image captured from web-camera, grayscale it and #assign it to variable faces*

4.1) #at every iteration of (x,y,w,h) in faces do the following:

4.1.1) draw a rectangle along one's face and

4.1.2) #de-serialize data from eyes xml and store it in variable eyes

4.1.3) #print them

4.1.4) #at every iteration of (ex,ey,ew,eh) in eyes do the following:

4.1.4.1) #draw rectangle along my eyes

4.1.4.2) print eyes as a vector-tuple acquired from camera

5) exit the algorithm

The concrete implementation is shown in figure 3.5:

comp_vision.py - C:\Users\Fikrat Talibli\Documents\python\computer vision\comp_vision.py (3.5.2)

File Edit Format Run Options Window Help

```
import cv2
import numpy as np

face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade=cv2.CascadeClassifier('haarcascade_eye.xml')

capture = cv2.VideoCapture(0)

while True:
    ret, imageFromCam=capture.read()
    grayscaled = cv2.cvtColor(imageFromCam, cv2.COLOR_BGR2GRAY)
    faces=face_cascade.detectMultiScale(grayscaled,1.26,5)
    for (x,y,w,h) in faces:
        cv2.rectangle(imageFromCam, (x,y),(x+w, y+h),(255,0,0),2)
        roi_grayscaled=grayscaled[y:y+h,x:x+w]
        roi_color=imageFromCam[y:y+h,x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_grayscaled)
        print("Faces detected: "+str((x,y,w,h))+str())
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color, (ex,ey),(ex+ew,ey+eh),(0,255,0),2)
            print("Eyes detected: "+str((ex,ey,ew,eh)))
    k=cv2.waitKey(30) & 0xff
    if k==27:
        break

capture.release()
cv2.destroyAllWindows()
```

Figure 3.5. Concrete implementation of algorithm

The output of the following code will be the window displaying the real-time video taken from the web-camera and the coordinates displayed in console. Now it is required to see what is displayed on console as the camera captures my face:

Faces detected: (237, 210, 165, 165)
Eyes detected: (26, 115, 84, 42)
Faces detected: (241, 208, 172, 172)
Faces detected: (267, 228, 136, 136)
Eyes detected: (16, 20, 104, 52)
Faces detected: (267, 212, 154, 154)
Faces detected: (269, 212, 160, 160)
Faces detected: (269, 209, 167, 167)

Eyes detected: (36, 42, 89, 44)
Faces detected: (275, 209, 156, 156)
Eyes detected: (45, 21, 83, 41)
Faces detected: (280, 213, 151, 151)
Faces detected: (277, 212, 154, 154)
Eyes detected: (38, 19, 80, 40)
Eyes detected: (28, 33, 107, 53)
Faces detected: (274, 208, 159, 159)
Faces detected: (271, 207, 164, 164)
Faces detected: (268, 207, 168, 168)
Eyes detected: (48, 35, 70, 35)
Faces detected: (267, 200, 175, 175)
Eyes detected: (43, 31, 85, 43)
Faces detected: (268, 205, 165, 165)
Eyes detected: (45, 31, 73, 36)
Faces detected: (271, 207, 158, 158)
Eyes detected: (38, 26, 79, 40)
Faces detected: (267, 208, 159, 159)
Eyes detected: (44, 26, 76, 38)
Faces detected: (270, 212, 157, 157)
Eyes detected: (37, 18, 87, 43)
Faces detected: (268, 209, 160, 160)
Eyes detected: (43, 36, 75, 37)
Faces detected: (260, 205, 165, 165)
Eyes detected: (42, 27, 83, 42)
Faces detected: (260, 205, 163, 163)
Eyes detected: (56, 38, 47, 23)
Eyes detected: (41, 19, 88, 44)
Faces detected: (266, 205, 152, 152)
Eyes detected: (36, 26, 78, 39)
Faces detected: (265, 206, 152, 152)
Eyes detected: (36, 21, 81, 40)
Faces detected: (264, 205, 152, 152)
Eyes detected: (41, 27, 68, 34)
Faces detected: (265, 210, 145, 145)
Eyes detected: (30, 18, 87, 43)
Faces detected: (266, 209, 146, 146)
Eyes detected: (30, 16, 82, 41)
Faces detected: (262, 207, 148, 148)
Eyes detected: (39, 26, 70, 35)
Faces detected: (260, 206, 153, 153)
Eyes detected: (39, 25, 71, 36)
Faces detected: (260, 207, 146, 146)
Eyes detected: (50, 32, 46, 23)
Eyes detected: (37, 18, 82, 41)
Faces detected: (259, 205, 149, 149)
Eyes detected: (35, 21, 79, 40)
Faces detected: (260, 206, 149, 149)

Eyes detected: (36, 23, 75, 37)
Faces detected: (259, 207, 149, 149)
Eyes detected: (85, 106, 40, 20)
Eyes detected: (38, 22, 74, 37)
Faces detected: (258, 207, 149, 149)
Eyes detected: (33, 24, 78, 39)
Faces detected: (259, 208, 147, 147)
Eyes detected: (92, 95, 38, 19)
Eyes detected: (37, 16, 77, 39)
Faces detected: (259, 206, 147, 147)
Eyes detected: (66, 115, 42, 21)
Eyes detected: (45, 34, 46, 23)
Eyes detected: (38, 21, 76, 38)
Faces detected: (258, 205, 149, 149)
Eyes detected: (35, 20, 75, 38)
Faces detected: (260, 207, 147, 147)
Eyes detected: (41, 30, 52, 26)
Eyes detected: (30, 17, 81, 40)
Faces detected: (258, 206, 147, 147)
Eyes detected: (41, 27, 63, 31)
Faces detected: (258, 207, 148, 148)
Eyes detected: (34, 18, 83, 41)
Faces detected: (260, 209, 142, 142)
Eyes detected: (40, 24, 63, 31)
Faces detected: (258, 207, 149, 149)
Eyes detected: (41, 24, 65, 32)
Faces detected: (259, 208, 147, 147)
Eyes detected: (34, 21, 78, 39)
Faces detected: (259, 208, 144, 144)
Eyes detected: (30, 18, 86, 44)
Faces detected: (259, 208, 146, 146)
Eyes detected: (33, 25, 74, 37)
Faces detected: (258, 206, 149, 149)
Eyes detected: (89, 96, 39, 20)
Eyes detected: (36, 22, 78, 39)
Faces detected: (257, 205, 152, 152)
Eyes detected: (46, 36, 47, 23)
Eyes detected: (34, 20, 87, 44)
Faces detected: (259, 208, 146, 146)
Eyes detected: (37, 23, 70, 35)
Eyes detected: (30, 42, 79, 40)
Faces detected: (257, 208, 149, 149)
Eyes detected: (44, 28, 54, 27)
Eyes detected: (34, 16, 83, 41)
Faces detected: (257, 208, 147, 147)
Eyes detected: (39, 23, 67, 33)
Eyes detected: (25, 39, 90, 45)
Faces detected: (257, 210, 144, 144)

Eyes detected: (44, 32, 49, 24)
Eyes detected: (37, 16, 76, 38)
Faces detected: (257, 210, 145, 145)
Eyes detected: (86, 85, 45, 22)
Eyes detected: (42, 28, 56, 28)
Faces detected: (258, 211, 140, 140)
Eyes detected: (81, 76, 48, 24)
Eyes detected: (38, 22, 67, 33)
Faces detected: (258, 210, 144, 144)
Eyes detected: (38, 22, 67, 34)
Faces detected: (256, 208, 147, 147)
Eyes detected: (32, 17, 85, 43)
Faces detected: (257, 208, 149, 149)
Eyes detected: (33, 18, 85, 42)
Faces detected: (259, 214, 140, 140)
Eyes detected: (36, 20, 69, 35)
Faces detected: (253, 209, 153, 153)
Eyes detected: (46, 31, 55, 27)
Eyes detected: (39, 20, 83, 42)
Faces detected: (249, 208, 156, 156)
Eyes detected: (53, 31, 53, 26)
Faces detected: (244, 209, 159, 159)
Eyes detected: (95, 82, 47, 24)
Eyes detected: (47, 28, 63, 32)
Faces detected: (244, 208, 160, 160)
Faces detected: (257, 216, 143, 143)
Faces detected: (256, 221, 148, 148)
Eyes detected: (32, 38, 80, 40)
Eyes detected: (53, 64, 79, 40)
Eyes detected: (17, 65, 113, 56)
Faces detected: (263, 230, 135, 135)
Eyes detected: (32, 33, 64, 32)
Eyes detected: (12, 48, 115, 57)
Faces detected: (256, 227, 146, 146)
Eyes detected: (81, 68, 37, 18)
Eyes detected: (42, 37, 61, 30)
Faces detected: (257, 226, 150, 150)
Eyes detected: (27, 36, 92, 46)
Faces detected: (246, 222, 165, 165)
Eyes detected: (94, 85, 47, 24)
Eyes detected: (15, 63, 53, 26)
Faces detected: (255, 233, 147, 147)
Eyes detected: (100, 29, 40, 20)
Faces detected: (251, 211, 172, 172)
Eyes detected: (38, 42, 102, 51)
Faces detected: (268, 221, 152, 152)
Faces detected: (293, 212, 169, 169)
Eyes detected: (40, 39, 99, 49)

The order of growth of given algorithm is cubical - $O(n^3)$. Let us display the complexity on the graph using numpy(see figure 3.6):

```
import pylab as plt
mySamples=[]
myCubic=[]
for i in range(0,30):
    mySamples.append(i)
    myCubic.append(i**3)

plt.figure('cubic')
plt.plot(mySamples,myCubic)
```

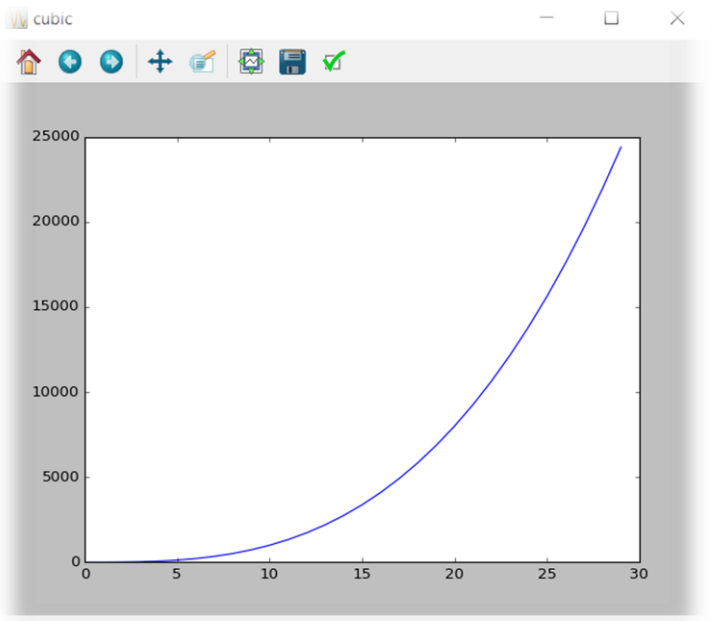


Figure 3.6. Algorithmic complexity of given function

Even though the price of current algorithmic complexity of current function is expensive, the cascade – solution is considered as one of the most powerful object detection solutions. The data given above can be used as test data and serialized in plain text files or can be stored in database.

As the program outputted the results with the corresponding points, one should notice that it outputs the position of face in rectangle. Nevertheless, what if the desired object is not the face or the ones in GitHub repository? Then OpenCV has created a possibility to train other objects and create custom **.xml** cascades, using OpenCV's library.

3.3. Getting raw data for negative images

In order to get negative images one has to refer to free open source image library. There are plenty of those on the internet; however, for this work **image-net.org-** a great open-source library for images was chosen. It is also required to write a small script that would take negative images and store it in some folder. In order to do so, one should refer to image library's API. Plus it is required to apply black-and-white filter on each of those images, and in some cases one must remove the pictures that contain nothing. So, there is lots going on in cleaning the raw data, as data scientists refer: Cleaning data is the ugly side of data science. Now to structure everything above said one has to:

- Download the images
- Find redundant images
- Create Positive and negative images

Here comes the functional programming in hand. Now one should script it as shown on figure 3.7.

```
import cv2
import numpy as np
import os

def download_images():
    negativeImagesLink="http://image-net.org/api/text/imagenet.synset.geturls?wn
    negativeImageUrls=urllib.request.urlopen(negativeImagesLink).read().decode()

    if not os.path.exists('negatives'):
        os.makedirs('negatives')

    picture_number=1
    file = open("negatives/downloaded_data.txt","w")
    for i in negativeImageUrls.split('\n'):
        try:
            file.write("Image" + i + " is downloaded.")
            print("Image" + i + " is downloaded.")
            urllib.request.urlretrieve(i,"negatives/"+str(picture_number)+".jpg"
            image=cv2.imread("negatives/"+str(picture_number)+".jpg",cv2.IMREAD_
            resized_image=cv2.resize(image,(100,100))
            cv2.imwrite("negatives/"+str(picture_number)+".jpg",resized_image)
            picture_number+=1
        except Exception as e:
            file.write("Couldn't download, because of " + str(e))
            print("Couldn't download, because of " + str(e))
```

Figure 3.7.a. Download images part

```
def find_redundant():
    for file_type in ['negatives']:
        for img in os.listdir(file_type):
            for redundant in os.listdir('redundant'):
                try:
                    current_image_path=str(file_type)+"/"+str(img)
                    redundant=cv2.imread('redundant/'+str(redundant))
                    question=cv2.imread(current_image_path)
                    if redundant.shape == question.shape and not(np.bitwise_xor(
                        print('Oops! This image contains question mark!')
                        print(current_image_path)
                        os.remove(current_image_path)
                except Exception as e:
                    print(str(e))
```

Figure 3.7.b. Find redundant images and remove them

```
def create_positive_and_negative_images():

    for file_type in ['negatives']:

        for img in os.listdir(file_type):
            if file_type=='negatives':
                line=file_type+'/'+img+'\n'
                with open('bg.txt','a') as f:
                    f.write(line)
            elif file_type=='pos':
                line=file_type+'/'+img+'1 0 0 50 50\n'
                with open('info.dat','a') as f:
                    f.write(line)
    print('done')
```

Figure 3.7.c. Create positive and negative images database

To track the download process I have created a file called **downloaded_data.txt**. The contents of the file is where the pictures are downloaded from depicted on Figure 3.8.

```
Imagehttp://farm4.static.flickr.com/3186/2606823711_b07495378f.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3360/3236502886_6a505502c5.jpg is
downloaded.Imagehttp://farm5.static.flickr.com/4022/4322083945_bc2db8ccf3.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3121/2556445732_0f905b11e4.jpg is
downloaded.Couldn't download, because of C:\projects\opencv-python\opencv\modules\imgproc
\src\imgwarp.cpp:3492: error: (-215) ssize.width > 0 && ssize.height > 0 in function
cv::resize
Imagehttp://farm4.static.flickr.com/3551/3443533285_2a70b92f2c.jpg is downloaded.Couldn't
download, because of C:\projects\opencv-python\opencv\modules\imgproc\src\imgwarp.cpp:3492:
error: (-215) ssize.width > 0 && ssize.height > 0 in function cv::resize
Imagehttp://farm1.static.flickr.com/224/491407069_8dbdc006c4.jpg is
downloaded.Imagehttp://farm1.static.flickr.com/33/42539714_c899059d27.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3512/3248952529_0bda5b70eb.jpg is
downloaded.Imagehttp://farm3.static.flickr.com/2461/3707202944_74e7a04f77.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3053/2664253302_956cb8356d.jpg is
downloaded.Couldn't download, because of C:\projects\opencv-python\opencv\modules\imgproc
\src\imgwarp.cpp:3492: error: (-215) ssize.width > 0 && ssize.height > 0 in function
cv::resize
```

Figure 3.8.a.The contents of downloaded_data.txt

```
Imagehttp://farm4.static.flickr.com/3236/3060322139_39a255d95c.jpg is
downloaded.Imagehttp://farm1.static.flickr.com/150/406427892_3776e7e6ef.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3609/4008974057_d0d7989bda.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3055/2658058589_618ac5486b.jpg is
downloaded.Imagehttp://farm3.static.flickr.com/2620/4205522549_02cbdc5c40.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3309/3615966336_1f0c670b76.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3278/2940713745_7f3ed00c42.jpg is
downloaded.Imagehttp://farm4.static.flickr.com/3446/3244281742_3c6af3fef7.jpg is
downloaded.Couldn't download, because of C:\projects\opencv-python\opencv\modules\imgproc
\src\imgwarp.cpp:3492: error: (-215) ssize.width > 0 && ssize.height > 0 in function
cv::resize
```

Figure 3.8.b.The contents of downloaded_data.txt (continued)

As the data is ready, it is time to train the necessary files, however it is impossible to do in Windows. Unfortunately, it is impossible to train the files in other environment than Linux environment, so it is required to install Linux either on physical machine or on Virtual Machine. Both procedures are quite bulky. It is a preference to use Virtual Machine, as it is safer for personal files.

Linux environment has many various distributions, so one more issue that rises here is the choice of a concrete Linux system. Nevertheless, there was no much complains about the choice as it is a matter of bulkiness and personal preference, so the choice has fallen on Ubuntu MATE – lightweight Ubuntu distribution of Linux that has all functionality of the latter.

3.4. Linux Installation in Virtual environment

In order to begin training one must download and install VirtualBox. In order to do so one must go to VirtualBox official website and download the corresponding version of VirtualBox, as shown in figure 3.9. Then by performing the steps above, one should also install the file correspondingly.

Figure 3.9. VirtualBox 5.1 website

Now it is important to download and install Ubuntu MATE Linux on VirtualBox. To do so one should first go to Ubuntu MATE website and download the Linux distribution itself. The website of Ubuntu MATE operating system based on Linux platform is shown in figure 3.10.

Figure 3.10. Ubuntu MATE 16.04.2 LTS

As it is stated in website (see figure 3.10) the minimum capacity required for this system is 512 MB of RAM, however due to the force of the PC that executed this work, 2500 MB of RAM has been chosen. So now, it is necessary to open up the VirtualBox and make necessary installations, as shown in figure 3.11.

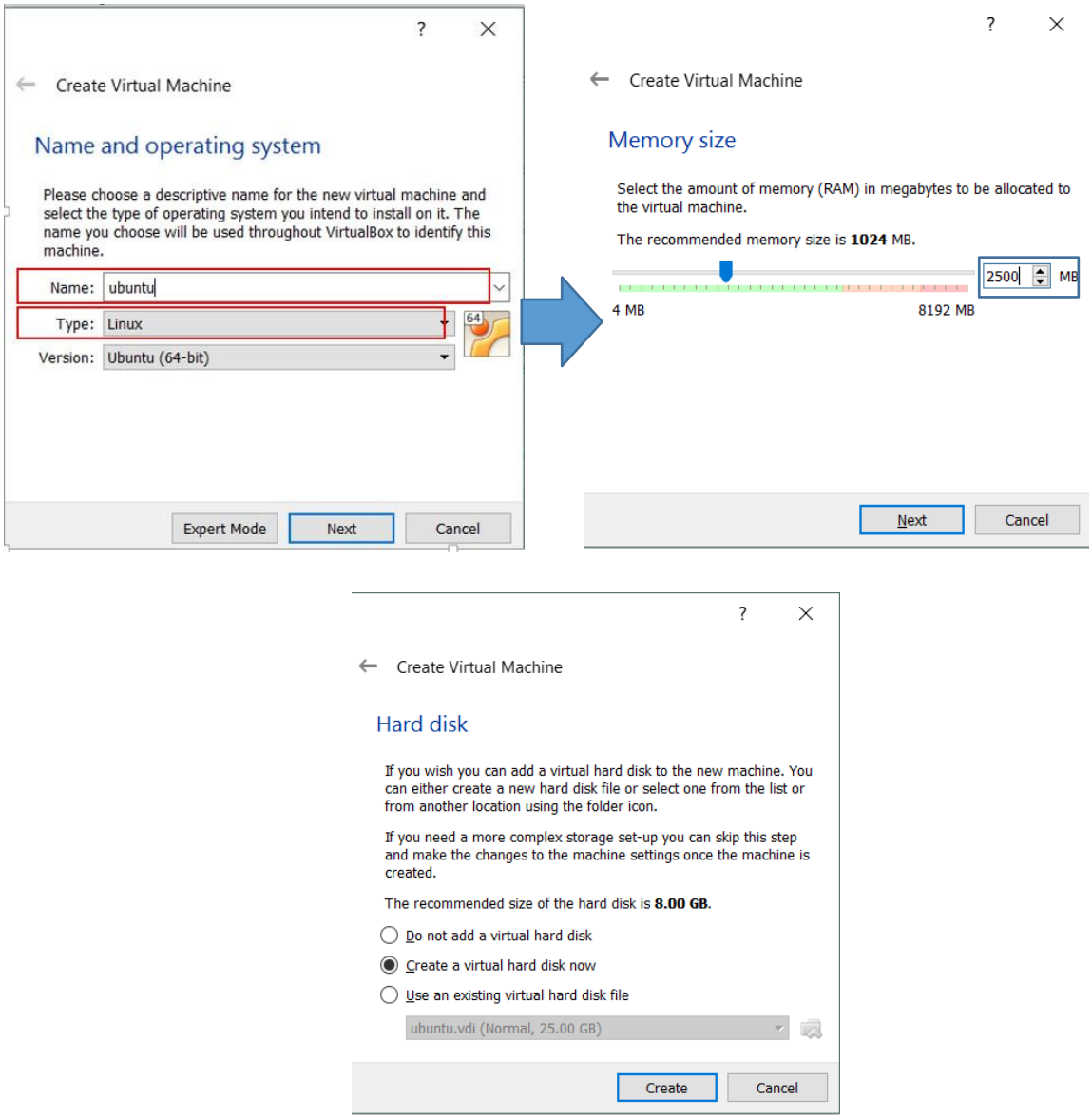


Figure 3.11.a. Ubuntu MATE 16.04.2 LTS installation

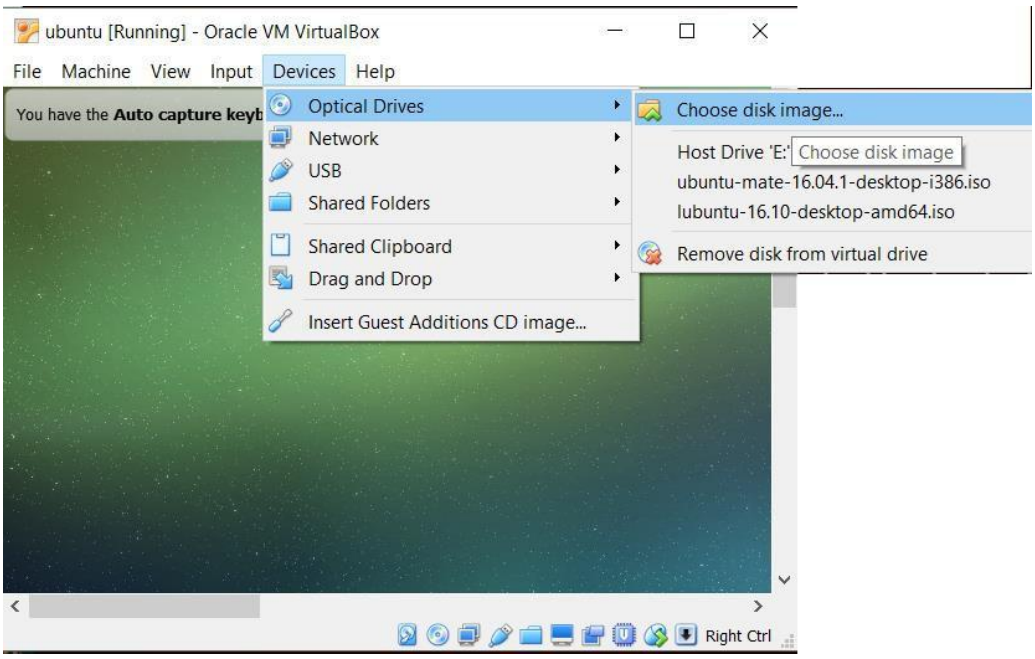


Figure 3.11.b. Ubuntu MATE 16.04.2 LTS installation

After the step performed in figure 3.11 one has to follow the corresponding instructions on the screen in order to install the necessary system. Now as everything is set up, it is time to execute training and as a result create a custom cascade.

3.5. Cascade creation

Now in order to create a cascade[32] one should download a GitHub repository of the OpenCV and execute it in following setup of environment, as shown in figure 3.12.

Compiler: `sudo apt-get install build-essential`

Libraries: `sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev`

Python bindings: `sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev`

OpenCV development library:

`sudo apt-get install libopencv-dev`

Figure 3.12. Necessary tools to install

One has to assume that everything is being executed on the Ubuntu MATE terminal in order to be able to train the cascade.

Now the final step is to generate necessary vector files from downloaded load of images. Make sure to copy everything from Windows to Ubuntu MATE. Now assume that one needs to detect phones analogically to detecting faces. So, now that everything has been copied, one has to perform the tasks, as shown in figure 3.13.

```
feeka@feeka-VirtualBox:~/opencv_workspace$ opencv_createsamples -img record.jpg
-bg bg1.txt -info info/info.lst -pngoutput info_pic -maxxangle 0.5 -maxyangle 0.
5 -maxzangle 0.5 -num 1000
Info file name: info/info.lst
Img file name: record.jpg
Vec file name: (NULL)
BG  file name: bg1.txt
Num: 1000
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 0.5
Max y angle: 0.5
Max z angle: 0.5
Show samples: FALSE
Width: 24
Height: 24
Create test samples from single image applying distortions...
Done
```

Figure 3.13. Info pictures collection

Now it is time to train the cascade as shown in figure 3.14.

```
feeka@feeka-VirtualBox:~/opencv_workspace$ opencv_traincascade -data newdata -ve
c positives.vec -bg bg1.txt -numPos 800 -numNeg 400 -numStages 10 -w 27 -h 40 &
[1] 3787
feeka@feeka-VirtualBox:~/opencv_workspace$ PARAMETERS:
cascadeDirName: newdata
vecFileName: positives.vec
bgFileName: bg1.txt
numPos: 800
numNeg: 400
numStages: 10
precalcValBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: HAAR
sampleWidth: 27
sampleHeight: 40
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC
```

```
===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   800 : 800
NEG count : acceptanceRatio    400 : 1
c positives.vec -bg bg1.txt -numPos 800 -numNeg 400 -numStages 10 -w 27 -h 40 &^
C
feeka@feeka-VirtualBox:~/opencv_workspace$ opencv_traincascade -data newdata -ve
c positives.vec -bg bg1.txt -numPos 800 -numNeg 400 -numStages 10 -w 27 -h 40 &
[2] 3795
feeka@feeka-VirtualBox:~/opencv_workspace$ PARAMETERS:
cascadeDirName: newdata
vecFileName: positives.vec
bgFileName: bg1.txt
numPos: 800
numNeg: 400
numStages: 10
precalcValBufSize[Mb] : 256
precalcIdxBufSize[Mb] : 256
stageType: BOOST
featureType: HAAR
sampleWidth: 27
sampleHeight: 40
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   800 : 800
NEG count : acceptanceRatio    400 : 1
Precalculation time: 19
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 33
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|   0.1725|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 2 minutes 35 seconds.
```

```
===== TRAINING 1-stage =====
<BEGIN
POS count : consumed   800 : 800
NEG count : acceptanceRatio    400 : 0.191205
Precalculation time: 33
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 33
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|  0.99875|    0.465|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 5 minutes 13 seconds.
```

```
===== TRAINING 2-stage =====
<BEGIN
POS count : consumed   800 : 801
NEG count : acceptanceRatio    400 : 0.0935454
|   2|  0.99875|    0.465|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 5 minutes 10 seconds.

===== TRAINING 2-stage =====
<BEGIN
POS count : consumed   800 : 801
NEG count : acceptanceRatio    400 : 0.0935454
Precalculation time: 36
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 35
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|        1|     0.53|
+----+---------+---------+
|   3|        1|     0.53|
+----+---------+---------+
|   4|  0.99625|   0.3425|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 10 minutes 29 seconds.
```

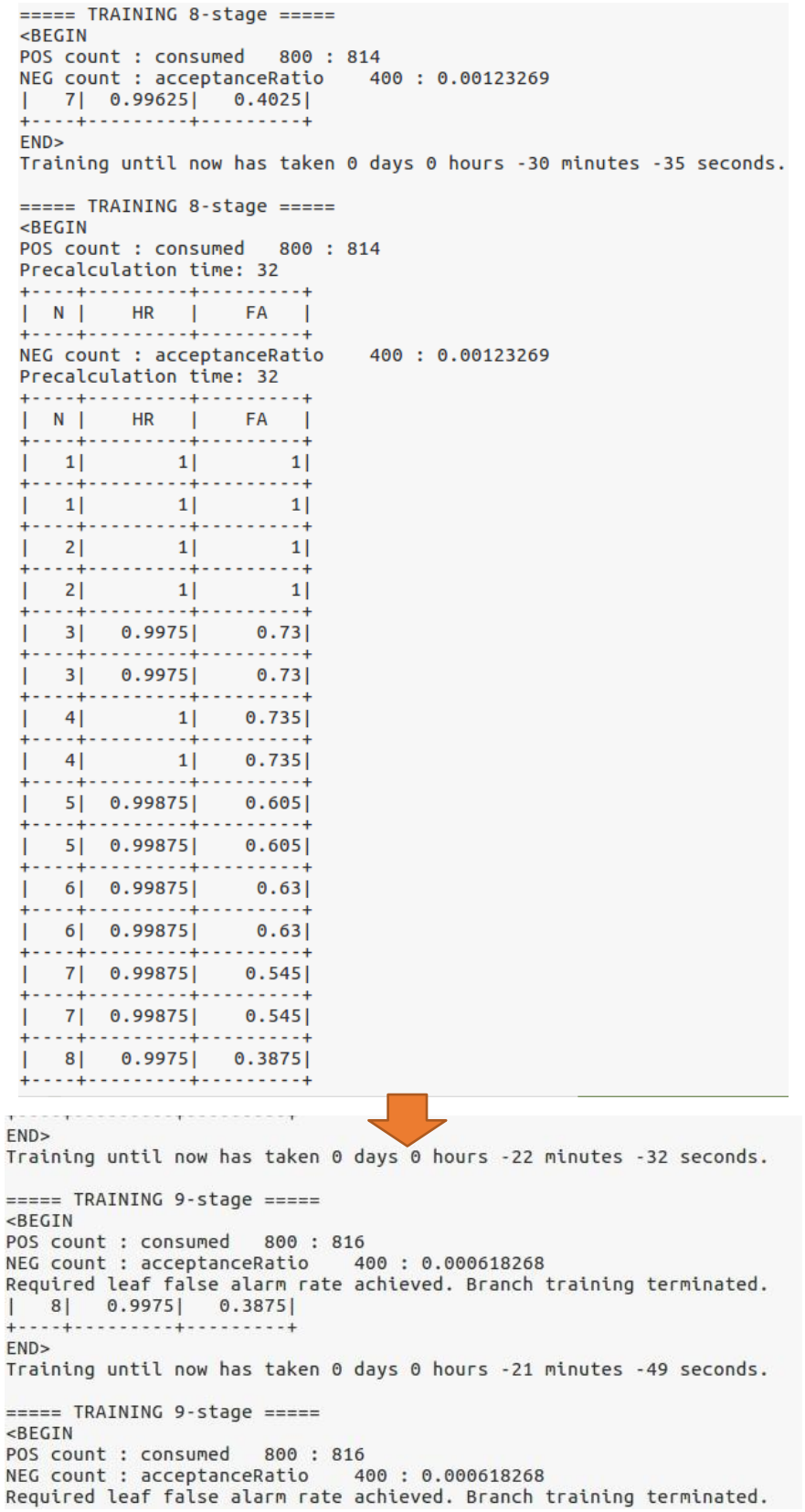
```
===== TRAINING 3-stage =====
<BEGIN
POS count : consumed   800 : 804
NEG count : acceptanceRatio    400 : 0.0347041
|   4|  0.99625|   0.3425|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 10 minutes 19 seconds.

===== TRAINING 3-stage =====
<BEGIN
POS count : consumed   800 : 804
NEG count : acceptanceRatio    400 : 0.0347041
Precalculation time: 35
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 34
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|   0.9975|     0.71|
+----+---------+---------+
|   3|   0.9975|     0.71|
+----+---------+---------+
|   4|   0.9975|   0.7025|
+----+---------+---------+
|   4|   0.9975|   0.7025|
+----+---------+---------+
|   5|   0.9975|   0.4075|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 16 minutes 30 seconds.
```

```
===== TRAINING 4-stage =====
<BEGIN
POS count : consumed   800 : 806
NEG count : acceptanceRatio    400 : 0.0159198
NEG count : acceptanceRatio    400 : 0.0159198
Precalculation time: 33
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 34
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|   0.9975|   0.6925|
+----+---------+---------+
|   3|   0.9975|   0.6925|
+----+---------+---------+
|   4|   0.9975|   0.6925|
+----+---------+---------+
|   4|   0.9975|   0.6925|
+----+---------+---------+
|   5|   0.9975|    0.465|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 22 minutes 16 seconds.
```

```
===== TRAINING 5-stage =====
<BEGIN
POS count : consumed   800 : 808
NEG count : acceptanceRatio    400 : 0.00939541
|   5|   0.9975|    0.465|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 22 minutes 36 seconds.

===== TRAINING 5-stage =====
<BEGIN
POS count : consumed   800 : 808
NEG count : acceptanceRatio    400 : 0.00939541
Precalculation time: 32
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 34
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|  0.99875|    0.655|
+----+---------+---------+
|   2|  0.99875|    0.655|
+----+---------+---------+
|   3|  0.99875|     0.59|
+----+---------+---------+
|   3|  0.99875|     0.59|
+----+---------+---------+
|   4|  0.99875|     0.45|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 26 minutes 59 seconds.
```

```
===== TRAINING 6-stage =====
<BEGIN
POS count : consumed   800 : 809
NEG count : acceptanceRatio    400 : 0.0045986
|   4|  0.99875|     0.45|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 27 minutes 22 seconds.

===== TRAINING 6-stage =====
<BEGIN
POS count : consumed   800 : 809
NEG count : acceptanceRatio    400 : 0.0045986
Precalculation time: 31
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 32
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|        1|        1|
+----+---------+---------+
|   3|        1|        1|
+----+---------+---------+
|   4|   0.9975|   0.6925|
+----+---------+---------+
|   4|   0.9975|   0.6925|
+----+---------+---------+
|   5|   0.9975|    0.485|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 32 minutes 37 seconds.
```

```
===== TRAINING 7-stage =====
<BEGIN
POS count : consumed   800 : 811
NEG count : acceptanceRatio    400 : 0.0025777
|   5|   0.9975|    0.485|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 33 minutes 6 seconds.

===== TRAINING 7-stage =====
<BEGIN
POS count : consumed   800 : 811
NEG count : acceptanceRatio    400 : 0.0025777
Precalculation time: 33
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
Precalculation time: 34
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|  0.99875|     0.73|
+----+---------+---------+
|   3|  0.99875|     0.73|
+----+---------+---------+
|   4|  0.99875|     0.73|
+----+---------+---------+
|   4|  0.99875|     0.73|
+----+---------+---------+
|   5|  0.99625|    0.555|
+----+---------+---------+
|   5|  0.99625|    0.555|
+----+---------+---------+
|   6|  0.99875|     0.57|
+----+---------+---------+
|   6|  0.99875|     0.57|
+----+---------+---------+
|   7|  0.99625|   0.4025|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours -31 minutes -11 seconds.
```



```
===== TRAINING 8-stage =====
<BEGIN
POS count : consumed   800 : 814
NEG count : acceptanceRatio    400 : 0.00123269
|   7|  0.99625|   0.4025|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours -30 minutes -35 seconds.

===== TRAINING 8-stage =====
<BEGIN
POS count : consumed   800 : 814
Precalculation time: 32
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
NEG count : acceptanceRatio    400 : 0.00123269
Precalculation time: 32
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|   0.9975|     0.73|
+----+---------+---------+
|   3|   0.9975|     0.73|
+----+---------+---------+
|   4|        1|    0.735|
+----+---------+---------+
|   4|        1|    0.735|
+----+---------+---------+
|   5|  0.99875|    0.605|
+----+---------+---------+
|   5|  0.99875|    0.605|
+----+---------+---------+
|   6|  0.99875|     0.63|
+----+---------+---------+
|   6|  0.99875|     0.63|
+----+---------+---------+
|   7|  0.99875|    0.545|
+----+---------+---------+
|   7|  0.99875|    0.545|
+----+---------+---------+
|   8|   0.9975|   0.3875|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours -22 minutes -32 seconds.

===== TRAINING 9-stage =====
<BEGIN
POS count : consumed   800 : 816
NEG count : acceptanceRatio    400 : 0.000618268
Required leaf false alarm rate achieved. Branch training terminated.
|   8|   0.9975|   0.3875|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours -21 minutes -49 seconds.

===== TRAINING 9-stage =====
<BEGIN
POS count : consumed   800 : 816
NEG count : acceptanceRatio    400 : 0.000618268
Required leaf false alarm rate achieved. Branch training terminated.
```

Figure 3.14. Training process of cascade

As a result, one can see the corresponding xml file in cascade folder that has described all its stages.

As one can see similarly to the eyes detection within faces, this trained object is also going to work for phones detection. This sophisticated procedure has quite a bulky algorithmic complexity, however it is considered as one of the most efficient algorithms in modern era’s technology and solves many computer vision problems.

CONCLUSION

An essence of OpenCV application programming interface is re-using the pre-invented technology, algorithms and methods in order solve the appropriate task with ease. The Haar feature-based cascade classifiers proposed by Paul Viola and Michael Jones are effectively helping to detect objects based on their features. It is also possible to create a cascade and train it using a variety of tools.

Those tools included the VirtualBox, custom Linux distribution operating system – Ubuntu MATE and finally the OpenCV library itself.

A real-world example, written on Python has demonstrated the appropriate ways of usage of given tools.

REFERENCES

1-https://en.wikipedia.org/wiki/Face_detection
2-https://en.wikipedia.org/wiki/Image_processing
3-https://en.wikipedia.org/wiki/Analysis_of_algorithms
4-https://en.wikipedia.org/wiki/Optical_character_recognition
5-https://en.wikipedia.org/wiki/Artificial_neural_network
6-https://en.wikipedia.org/wiki/Pattern_recognition
7-http://rduin.nl/papers/CSChallanges_07_PRScience.pdf
8-https://en.wikipedia.org/wiki/Pattern_recognition#Probabilistic_classifiers
9-https://en.wikipedia.org/wiki/Monochrome
10-http://docs.opencv.org/2.4/modules/refman.html
11-http://docs.opencv.org/2.4/modules/ml/doc/boosting.html?highlight=adaboost
12-https://www.virtualbox.org/
13-https://en.wikipedia.org/wiki/Outline_of_object_recognition
14-https://en.wikipedia.org/wiki/Face_perception
15-https://docs.python.org/3/
16-https://en.wikipedia.org/wiki/Functional_programming
17-https://docs.python.org/2/howto/functional.html
18-https://en.wikipedia.org/wiki/Imperative_programming
19-http://www.secnetix.de/olli/Python/lambda_functions.hawk
20-http://opencv.org/platforms/
21-https://en.wikipedia.org/wiki/OpenCV
22-https://en.wikipedia.org/wiki/Haar-like_features
23-http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html
24-https://en.wikipedia.org/wiki/Cascading_classifiers
25-https://pypi.python.org/pypi
26-http://www.lfd.uci.edu/~gohlke/pythonlibs/
27-https://www.python.org/downloads/
28-https://github.com/opencv/opencv/tree/master/data/haarcascades
29-http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf
30-http://image-net.org/
31-https://ubuntu-mate.org/
32-http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_traincascade.html