National College of Ireland

Project Submission Sheet – 2015/2016

School of Computing

Student Name:	Amit Bora		
Student ID:	14125013		
Programme:	MSc in Cloud Computing	**Year:**	Jan-2015
Module:	Dissertation		
Lecturer:	Dr. Arghir-Nicolae Moldovan		
Submission Due Date:	17th December 2015		
Assignment Title:	Improving real-time face detection and recognition techniques with the help of cloud computing in real time		
Word Count:	10,350		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Amit Bora

Date: 17th Dec 2015

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3 Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Improving real-time face detection and recognition techniques with the help of cloud computing in real time

Amit Bora

Submitted as part of the requirements for the degree
of MSc in Cloud Computing
at the School of Computing,
National College of Ireland
Dublin, Ireland.

January 2016

Supervisor Dr. Arghir-Nicolae Moldovan

Abstract

Cloud computing is transforming the business of information technology by deploying, managing and running all software on cloud which provide reliable, scalable and cost-effective technology. Cloud computing is gaining much popularity nowadays. Many organizations are switching to cloud to fulfill user requests as fast as possible. Cloud computing gives advantages such as mobility, low cost to user. Devices like laptop have low processing power and storage. Shifting processing power and storage of device to cloud, we can overcome drawbacks of devices. Face recognition and detection is a wide area of research but it is hard to achieve in real time. Many police departments are using face recognition for criminal person detection. As many people are using internet service, face detection with cloud computing is one of the important application. Devices like low configuration desktops face many challenges in their resources as low computing power, and storage.

The thesis proposes a software methodology that helps to overcome the challenges of low computing hardware. The thesis investigates the use of cloud computing to improve face detection and face recognition technology. The openstack platform was used to evaluate the performance of three different algorithms (haar cascade classifier, LBP cascade classifier, and eigenface) on four different databases. The face detection and recognition software was able to detect and recognize faces successfully. To conclude, we can use the software to any database which contains different number of images and hence we promote the usage of cloud to get the result of face detection and face recognition in real-time.

Keywords:- Cloud computing, Image processing, face detection, face recognition, Haar Cascade Classifier, LBP Cascade Classifier, and Eigenface.

Acknowledgment

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. Arghir-Nicolae Moldovan for the continuous support and flexibility in carrying out my research. His guidance helped me in completing this thesis in the best possible way. I couldn't have imagined a better mentor for this study.

Besides my supervisor, I would like to thank Dr Horacio Gonzalez-Velez and Dr. Adriana Hava Olariu. I am also thankful to Dr. Damien Mac Namara for his immediate response for any queries while completing my thesis.

I would like to thank all my friends for their motivation and support in carrying out my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The IT industry is rapidly transforming by cloud computing for developing software for millions to consume as a service without running it on individual computers. It allows users to access Software as a Service from anywhere in the world with the help of Internet on behalf of their individual computers. Face recognition is useful for many fields like security, psychology and image processing. As face detection and recognition application required much more processing power as it search and compare required face in database with input face. But face detection and recognition is very challenging as devices lack in energy, processing capability and storage. Cloud computing will solve the challenges of devices like processing and storage. When moving of processing power and data storage away from devices on cloud, will improve life of devices. As use of cloud for processing security of data will be improved because of centralized monitoring and maintenance in the cloud.

Introduction shows the steps which involved in recognition in order to understand how they contribute to day to day life. Face recognition is a comparison process which mainly involves detection of face and comparison with database. It is a form of biometric security which is gaining popularity. As discussed by Ayad et al. [2014], faces contain natural highlights of acknowledgment technologies as per psychological standard of human being. As per Ayad et al. [2014], the most instinctive way to deal with face recognition is in view of the geometric features of a face.

Ayad et al. [2014], Stojmenovic [2012] both agree that face recognition is divided in two parts, the first part is face detection which is taking into account the recognition of the area of the face in the picture paying little mind to size, position, brightness and condition. The second part deals with face recognition which is comparing the detected image with stored human faces from a database. Also, the main focus of this

dissertation is to detect and compare faces based on their facial expression without depending on resolution of image. As most of the articles state, face detection and recognition in real time is very difficult and challenging. This document discusses difficulties and challenges which is stated by journal articles and will also outline how to manage these troubles and challenges.

The document is structured as follows:

- **Chapter 2** discusses previous related research work on face detection and face recognition with cloud computing.
- **Chapter 3** presents the specification details which will elaborate on internal and external requirement of the system. It will also discuss about testing platform, algorithms and databases which are used to develop the system.
- **Chapter 4** presents implementation details of the algorithms in OpenStack.
- **Chapter 5** evaluates the system with the help of some test case scenario.
- **Chapter 6** concludes the dissertation and discuss the research research findings.

Chapter 2

Literature Review

The research problem that is going to be discussed for this dissertation is how can traditional face detection and recognition techniques be improved with the help of cloud computing in real time? This dissertation will also discuss why face recognition is challenging, along with how face detection and recognition happens? It will also explain how to improve performance to achieve it in real time. Cloud computing comes with its advantages such as storage and processing power, but cost for cloud usage which have to be paid by user. It is necessary to understand about cloud computing as the research problem relates to it.

2.1 Cloud Computing

Armbrust et al. [2010] define cloud computing as "both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services." Buyya et al. [2009] define a Cloud as "type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers." Cloud providers rely on cloud service models which are used to deliver their services to the cloud user.

According to Mell and Grance [2011], there are three cloud service models which are infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a Service (SaaS). Mell and Grance [2011] also provide the NIST definition for cloud computing as a pool of resources (e. g. networks, storages, servers and services) which is enabling on demand that can be rapidly scaled and released with minimal

effort. Carolan et al. [2009] provides details about the infrastructure service model as the user can choose an operating system to be installed in any provider's public or private cloud. After that, the subscriber is free to install any software on the operating system. The key provider for the IaaS model is amazon. It provides cloud platform for public with cost effective pay as you go In the PaaS model, the service provider provides all the resources which are required to build the application such as operating systems, compilers, programming languages, databases, web servers and application development tools. PaaS model allows developers to develop and deploy applications. The key providers of the PaaS model are Google and Microsoft. Microsoft Azure is used more by users for the PaaS cloud platform from Microsoft Inc. In SaaS the service provider hosts the application on their cloud and their subscribers can get to these applications through light applications, for example, browsers from their mobile phones or individual desktop system. A subscriber can't modify an application using this model. Saleforce.com is a key cloud provider for the SaaS model.

Cloud computing has many advantages. It improves the utilization rate of hardware by using same hardware by many users. It save idle time of hardware and save cost for extra hardware. With the help of cloud computing, speed of applications development is increased. Many resources can scale up and down dynamically as per requirement without changing of hardware. There are many people which are using cameras and cell phones to take images of person. For example, by using a mobile phone to take images and recognize a face, we can identify criminals. As face detection with recognition require more energy and processing power to compare facial image with input, it can't process on normal devices, so high processing devices are required. As cloud provide high processing power and many more advantages. So why we using cloud computing are discussed below.

2.1.1 Why Cloud Computing?

Marston et al. [2011] discusses some advantages about cloud computing. Cloud computing lowers the cost of entry level firms which require compute-intensive hardware. Cloud computing also allow to allocate resources dynamically. Some countries lack in IT solution, so cloud computing provide advantage to these countries by enabling IT services. Cloud computing is dynamic and virtualized. The infrastructure of cloud computing take different forms which are required by user in run time. As discussed by Rajaraman [2014], use of cloud computing by any organization saves much more money as they do not need to invest in large computing hardware. He also states that, user have to pay to provider for what organization uses of cloud.

As discussed by Marston et al. [2011], cloud computing provide immediate access to required hardware by user with both side scaling. In other words, cloud computing should be able to scale up and scale down automatically as required. This phenomenon reduces the cost of use of hardware and its resources. Scalability of resources at run-time as per the user request is most important in cloud computing and is best explained by both Armbrust et al. [2010], Buyya et al. [2009] in their articles. According to Armbrust et al. [2010], Buyya et al. [2009], change of scale of resources can help to measure correct cost of service without breaking or violating service Level Agreements (SLAs). There are many advantages explained by the author Armbrust et al. [2010] such as save money by paying correct cost of service, conserve resources and money, flexible pricing is common advantage discussed by both Armbrust et al. [2010], Buyya et al. [2009] to elaborate the importance of run time scalability of service.

As per Marston et al. [2011], cloud become shared by different end users and way of use of cloud is depends on each user. As there are many user per cloud, resources should be separated from each user. This phenomenon is provided by cloud as visualization and makes cloud very useful as saving resources. Cloud computing reduces IT barriers for startups which are coming in market with new innovations. As per Rajaraman [2014], cloud provider provides quality of service to user with some Service Level Agreement (SLAs). It assured of cloud infrastructure to user of cloud in sense of time, space, security, etc. As per Marston et al. [2011], cloud computing provide organizations to scale their services according to client demand. Author Marston et al. [2011] also discussed about recovery of data in cloud computing which uses distributed infrastructure, disaster recovery will be easy in crucial and high value applications. It is also possible as cloud computing save same data in multiple location to recover on demand. This means clod computing is capable of self healing in terms of data.

Having seen the basics of cloud computing, we will now look at face recognition which is main area of dissertation. But face detection is mainly depends on image processing. So next section will be Image processing.

2.2 Image Processing

As discussed by Plagman and Littlejohn [1992], image processing is a method to convert an image into digital format and perform the entire task related to that image. With the help of image processing, we can extract useful information and also enhance an image. It includes converting an image into a two dimensional matrix form and perform image processing enhancement on it. Image processing involves three basic components capture, analysis and manipulation and output. These components are interconnected

with each other by using microcomputers, minicomputers or mainframes depending on the input image.

Image processing includes three steps which are, capture: Using digital photography or optical scanner, we can capture the image. Analysis and Manipulation: It mainly includes compression and enhancement of the image with the help of image processing tools. And the last one is output: It includes report generation on the second stage and storing the output of the second stage.

As discussed by Prasada [1979], Image processing offers several advantages on raw images: Image processing improve complex image process for any task. Image processing encompasses many disciplines. It helps many application fields like medical diagnosis for example X-ray scan, robotics and manufacturing process and space exploration. Image processing also enable many people to work on the same data simultaneously to complete it as fast as possible.

As per Twogood and Sommer [1982], there are two types of image processing - Analog image processing and Digital image processing. Analog image processing can be used for a document which is required in hard forms like printouts and photos. Analog image processing is not restricted to just the area which is in study but it also depends on the knowledge of the analyzing person. It shows that image processing is a combination of the personal knowledge of the analyst and the data enhancement technique. Digital image processing helps to manipulate digital images with the help of digital tools. Satellite data are one type of digital image which in raw form, contains deficiencies. To remove all the impurities and get the original image information, it has to be overcome with digital image processing. Techniques used for it are pre-processing, enhancement and display, information extraction.

Hornak [2002], Analog image processing carried on hard copies like printouts and photographs. Analog image processing output depends on analyst of analog image. Analyst apply various interpretation visual techniques. As analog image processing is fully depends on analyst but it also depend on some collateral data of image processing. As per author Hornak [2002], Digital image processing carried out on digital images with the help of computers by using image processing algorithms. Raw digital image contains deficiencies and distortions. To remove such impurities, three phases of algorithms carried out on digital image. These three phases are pre-processing of image, enhancement and display and information extraction. After all these phases, we get original information as pure digital image without impurities.

Rovetta discusses about digital image processing which have many advantages over analog image processing. Digital image processing allows wider range of enhancement

algorithms which can applies on raw digital image. It also avoid problems such as noise (in data of digital image) building in the middle of processing and digital signal distortion during processing. Analog image processing can be only apply on 2D images but digital image processing can apply on multi-dimensional image.

Twogood and Sommer [1982] mention, the general purpose of image processing is divided into 5 groups which are explained below:

1. Visualization: Image processing helps to visualize and observe an image with naked eyes.
2. Image sharpening and restoration which helps to create a better enhanced image.
3. Image retrieval which means with the help of image processing, we can get the required image.
4. Measurement of pattern which also helps to measure different objects in an image which is not visible to human eyes.
5. Image recognition which helps to differentiate the objects in an image.

2.3 Face Technologies

As discussed by Ayad et al. [2014], face acknowledgment is a standout amongst the most common natural highlights of acknowledgment technologies as per the psychological standard of human beings. In the early days (1970) of face detection, simple heuristic and anthropometric techniques were used. These techniques had many limitations such as plain background, frontal face, and regular passport photograph Hjelms and Low [2001]. More strong division plans have been displayed, especially those using movement, shading, and generalized information. The use of statistics and neural networks have additionally empowered countenances to be identified from cluttered scenes at distinctive distances from the camera. Moreover, there are various advances in the design of highlight extractors, for example, the deformable layouts and the dynamic contours which can find and track facial features precisely.

Ayad et al. [2014] and Stojmenovic [2012] both agree that face recognition is divided in two parts, the first part is face detection which is taking into account the recognition of area of the face in the picture paying little mind to size, position, brightness and condition. The second part deals with face recognition which is comparing the detected image with stored human faces from a database.

2.3.1 Face Detection

Hjelms and Low [2001] discuss that face detection techniques can obtain data from the face which can be adequately sorted out into two general classes recognized by their distinctive methodology to using face information. The techniques in the first class makes utilization of face information and combines this information with traditional recognition method in which low level features are implied. The properties of face, for example, skin color and face geometry, are misused at distinctive system levels. These techniques for face detection are fulfilled by adjusting distance, angles and area which is generated from the image. As features are the main input to these techniques, these techniques are termed feature based approach. The second approach for face detection technique is pattern recognition techniques. These techniques use image based representation of faces without changing the features of the image likes distance, angles and area. With these techniques, an image is directly classified into two groups - face group without feature derivation and other the rest of the image. Unlike the feature-based approach, these generally new techniques have fused face knowledge into the system through mapping and preparing plans.

As per Ayad et al. [2014], the Haar classifier face detection algorithm is able to achieve a 95% accurate result for human faces. The Haar algorithm first looks for the human face. Once the facial features are found, it passes to the second stage which is the removal of the non-face elements. The cascade eliminates features by sticking to a requirement in the iteration of the algorithm. It will exit after each stage completed and each face is found or fails in any stage. Each stage of the algorithm goes through many distinct Haar features.

As per Belaroussi and Milgram [2012] more resolution face image require more time to detect same number of faces as compare to less resolution image. Belaroussi and Milgram [2012] use four frontal face detectors which are based on Haar filter readily available in OpenCV, and it is well described by Lienhart et al. [2003]. Belaroussi and Milgram [2012] use some pre-processing before face detection to achieve better result in face detection.

Zaqout et al. [2004] proposed new algorithm called as Lookup Table Algorithm. This algorithm first create three lookup tables which contain triple component ration of image histogram. The ratios are with color green:red, blue:red and blue:green. This algorithm crop image with help of lookup table. After this algorithm skin classifier is applied. Zaqout et al. [2004] uses PICS image database to test face detection.

Chen et al. [2008] proposed new algorithm to improve performance of face detection.

They use preprocessing on image and divide it in two new files namely skin map generation and Luminance Map and Lips Map Generation. On Skin map Generated image they perform Face search on the image and it equalized with local histogram which is generated by Luminance Map and Lips Map. They applied this technique on 1000 face images which are from Kodak database.

Technologies used by all authors mentioned above are summarized in Table 2.1.

Table 2.1: Face Detection Technologies

Authors	**Approach**	**Algorithm**	**Feature Used**	**Test Database**
Hjelms and Low [2001]	Image Based	Principle Component Analysis	Linear Subspace Method	7562 Custom Created
		Linear Discriminant Analysis		
		Factor Analysis		
		Multilayer Perceptron MLP	Neural Networks	
		Kullback Divergence	Statistical Approach	
Ayad et al. [2014]	Feature Based	Haar Classifier Detection	Haar Feature	220 Face Custom
Belaroussi and Milgram [2012]	Feature Based	Haar Classifier Detection (after preprocessing)	Haar Feature	200 Webcam Images
Chen et al. [2008]	Image Classification	LBP (PCA) algorithm	Local Histogram and Frontal Face Classification	1000 Custom Images
Hsu et al. [2002]	Feature Based	Skin Tone Color Algorithm	Facial Feature Detection after preprocessing	FERET and CMU databse
Zaqout et al. [2004]	Face Feature Based	Lookup Table Algorithm	Histogram Based	PICS Image Database

2.3.2 Face Recognition

As per Ayad et al. [2014], the most instinctive way to deal with face recognition is in view of the geometric features of a face. The first recognition system of a face is the marker point where points refer to the position of eyes, ears, nose and other features of a face which are used to generate a feature vector. A feature vector is the distance between points and the angle between them. The output is generated by calculating the Euclidean distance between the feature vector and the matching image.

Nowadays, many algorithms recognize an image using the Eigenface and fisherface algorithms. The Eigenface algorithm takes the detected faces received from the Viola Jones Face detector and converts them into a set of Eigenfaces, which are all eigenvectors done step by step. The training set is generated by a set of faces. After the generation of the training set, the mean of the training set is calculated and subtracted from each image of the training set to calculate a covariance matrix. Eigenvectors contain the data of faces and eigenvalues for each vector is calculated from the covariance matrix. From all faces, a face is selected which has the highest eigenvalues. If two appearances have comparable Eigenfaces, then they are more inclined to be the same face. This phenomena is proposed by Pissarenko [2002].

As per Peng and Shen [2013], every 2 out of 8 face recognition system depends on principle component analysis (PCA). PCA uses subspace method which is quite suitable for mobile device. As subspace uses low computation cost and memory usage for doing computation. But it has some disadvantage as it needs more time for processing on mobile device. And it will have limited database for comparing face. As per Stojmenovic [2012], previous research has done by various researcher on face recognition provides various techniques which are classified according to how and what kind of features they extract from face. He tested many algorithms and conclude that algorithm proposed in Orientation analysis for rotated human face detection (2002) is better. He also describes steps in face recognition that is, face detection from image is a first step which is followed by extracting suitable features from input image which will compares with database. He also describes that images taken in various resolution give different output results. He also states that facial expression is also depends on light condition when photo is captured.

Technologies used by all authors mentioned above are summarized in Table 2.2.

Table 2.2: Face Recognition Summery

Authors	**Approach**	**Algorithm**	**Feature Used**	**Test Database**
Ayad et al. [2014]	Principle Component Analysis (PCA)	Eigenface	Facial Features	220 Custom Created
Pissarenko [2002]	Principle Component Analysis (PCA)	Eigenface	Mouth and Face Feature	464 Custom Created
Peng and Shen [2013]	Principle Component Analysis (PCA)	Eigenfaces	Eigenvectors (Distance between objects)	Not described
Ahonen et al. [2006]	Local Binary Pattern (LBP)	LBP Method	Face Texture Description	FERET 14051 Images

2.4 Face Detection and Recognition on Cloud Computing

As per Stojmenovic [2012], a face detection is a biometric application which is depends on face. Detection system mainly focuses on one to one which mainly gives output whether a query matches with database. As per author, identification application of faces are more challenging because application have to match one to many faces stored in database. It also challenging to give partial match faces. He also describes that existing face recognition system have lack of real time performance. As face contain some key feature point, it is hard to identify in real time. Similarity of faces depends on relative distance between key feature point, such as distance between eyes, width of face etc. As this process requires more processing power which is not capable of day to day device such as normal desktop system. As per author, Sony Ericsson's Recognizer application enables user to recognize people who are only present in contact list with picture present. CroudSTag face detection application takes more response time which is averagely 22s to identify faces.

As per Pawle and Pawar, face recognition technique is biometric authentication techniques which require high processing power and centralization of database. Pawle and Pawar also discusses that face contain different features such as eyes, eyebrows, nose, lips, and chin which are not common to other user. As per [Pawle and Pawar], to authenticate person to system it should be recognized in real-time. As per Pawle and Pawar, usage of cloud to recognize face generates output in real-time.

Chapter 3

Design

3.1 Introduction

This chapter specifies the requirements and specifications set for face detection and recognition with cloud computing in real time. Face recognition with cloud computing in real time provides matching of faces with database. This section covers the entire project and its development technologies in detail. The final product developed using these technologies. As face detection and recognition require high processing power, cloud to be used. There are many types of cloud as discussed in section 2.1. Among all of these OpenStack provides some additional features like object storage, block storage, and networking. These services provided by OpenStack are free of cost, I have chosen OpenStack as my cloud provider.

3.2 OpenStack Cloud Platform

As described above, we use OpenStack. OpenStack [2014] is an open source cloud which is more extensively used to control compute, storage and networking resources. It also provide dashboard to manage all services from web interface with full control for administrator. It also provide command line tool to manage its facilities from command line.

OpenStack [2014] OpenStack is a project which was developed by NASA and Rackspace Hosting in 2010. Firstly, it is developed to help organizations to offer services of cloud computing. In 2011, developers of Ubuntu adopted OpenStack for further addition of development of OpenStack. In 2012, Red Hat also announced their OpenStack

distribution with Essex release. Red Hat also started commercial support to OpenStack in July 2013. In 2013 NASA officially announced that they are no longer active in OpenStack because of lack of technical progress but it uses services of OpenStack. In 2014, HP also joined OpenStack with release of HP Helion OpenStack. In these years many companies like AT&T, EMC, DELL and much more organization put their effort to take the OpenStack to the next stage. The coding part is been taken care by IBM and Red Hat.

OpenStack project is developing and contributing by large number of developers and cloud computing technologists. They are also producing open standard platform for both public as well as private clouds. OpenStack aims to deliver solutions for all types of clouds with simple to implement, massively scalable, and feature rich. OpenStack foundation put the effort to develop the OpenStack software and is been managed by the people of different organization with contributing by designing and coding their product. Now OpenStack foundations have thousands of members from all over the globe and its funded more than USD 10 million. Membership in OpenStack foundation is free and accessible to everyone.

OpenStack sets some principles which are described below;

1. OpenStack is freely available with full source code in standard Apache 2.0 license.
2. After every six months foundation of OpenStack gathers for design and gathers requirements for future release which is open to all public.
3. OpenStack is dedicated for gathering active developers and user in OpenStack Community. All the process of OpenStack is open, transparent and documented.

3.2.1 OpenStack Architecture

The following are the services of the OpenStack:

OpenStack Dashboard (Horizon): Dashboard is the graphical user interface (GUI) with the help of web browser from where users put their credentials to login into the OpenStack and configuring access controls to different services such as launch instance, upload image.

OpenStack Compute (Nova): It manages the life of compute instance. It is a fabric controller which is responsible for spawning, scheduling and life of virtual machine on user demand. KVM hypervisor which is an open source hypervisor is by default installed on it.

OpenStack Networking (Neutron): It is responsible for all network operation of OpenStack. It also provide interface to user to manage network. It also support many networking vendors and technologies. Neutron allows us to create VLANS, DHCP, assign Floating IP, IDS, Load balancing and creating VPN in OpenStack.

OpenStack Object Storage (Swift): It is the storage system of OpenStack. It is scalable data storage used to save and retrieve data from the cluster which are widely distributed in OpenStack environment. It is highly effective for fault tolerance with replication and scalability. It is unlike file server.

OpenStack Block Storage (Cinder): This facilitates the OpenStack to provide persistent block storage. It also provides attaching and detaching the users own storage system which can be in the form of servers or SAN. It also provide facilities like creation and management of storage devices with the help of its pluggable driver architecture.

OpenStack Identity (Keystone): It is a central directory to provide accessibility. It provides centralized authentication and authorization to the users. It also provide catalog of endpoints for all services.

OpenStack Image Service (Glance): It accommodate disk image and provides feature to upload an image on OpenStack.

Telemetry Module (Ceilometer): This module is used to support billing systems on OpenStack. It gives details of services used in OpenStack.

OpenStack Orchestration (Heat): It is orchestrate multiple clouds with use of either its HOT template format or AWS Cloud Formation template.

The architecture of OpenStack Juno is shown in Figure 3.1.

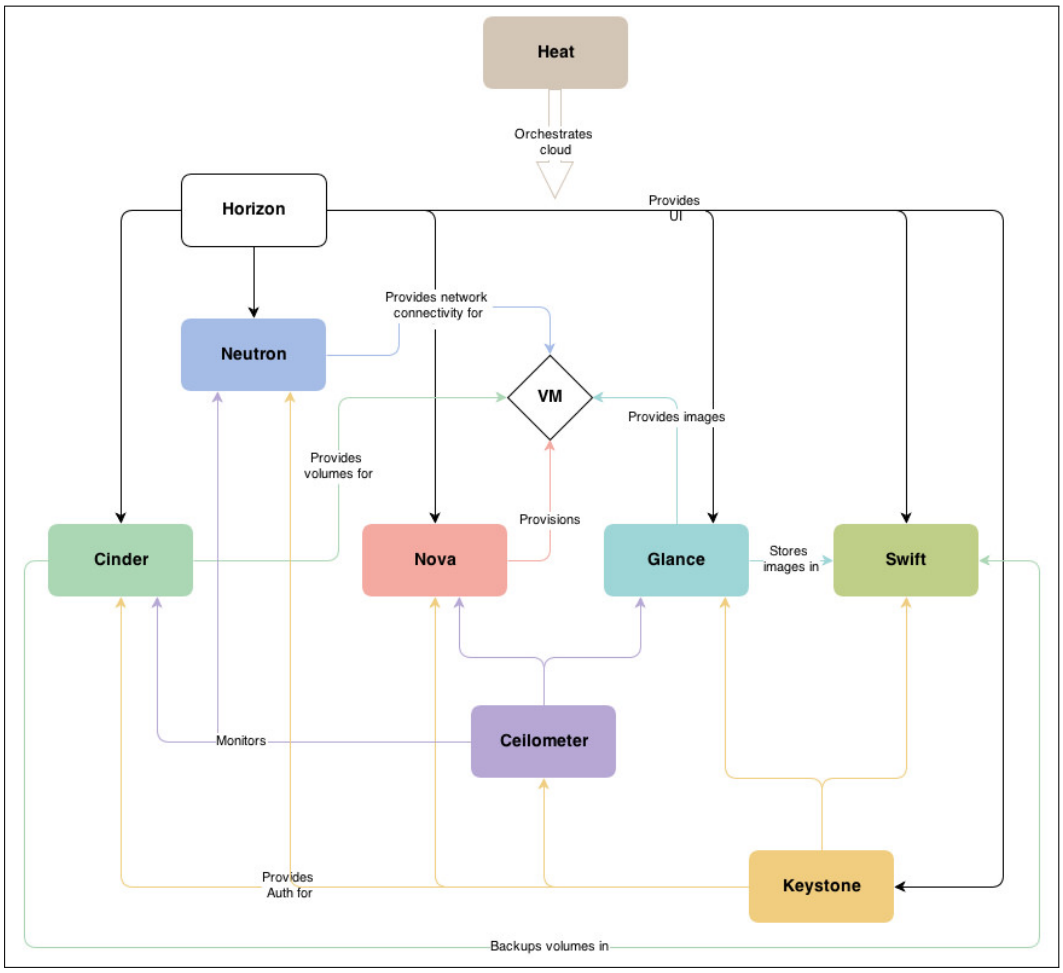

Figure 3.1: OpenStack Architecture(OpenStack [2014])

3.2.2 OpenStack Security

OpenStack also provide some security for both data and virtual machines. OpenStack is very robust role-based access control. Access and resource utilization can be controlled at the level of user, roles, and projects. The keystone identity service provides multiple form of authentication including user-name / password and token based authentication.

A Powerful Dashboard: the Dashboard is a web application that provides an intuitive interface for managing compute, storage and networking resources, allowing users and administrator to have a clear overview for the management of resource usage, currently active VM instance and users. Among much private cloud OpenStack proves to be more effective with security and GUI.

3.3 OpenCV

As per Bradski and Kaehler [2008], OpenCV is a library which is open source in nature. It is used as computer vision library which is available freely from [OpenCV, 1999]. OpenCV is written in C and C++. It is designed for computational efficiency with strong focus on real-time application. The goal of OpenCV is to help people by providing simple vision environment to build real-time applications. As OpenCV written in C and C++, it can take advantage of multi-core applications. OpenCV contains its own Machine Learning Library (MLL) which contains 500 functions that span many areas. OpenCV is open source but it can be used to build commercial product.

OpenCV is a computer vision library that provides conversion of still or video imaginary into other representation of same. These representation is done for new goals such as detection of face in image or finding person in a scene. Computer vision receives image in pixel format which represents some numbers of pixels. Computer vision does not provide built in pattern recognition or any other automatic controls. A person who uses OpenCV have to develop its required program code to acquire result.

HighGUI is main module of OpenCV architecture which is shown in Figure 3.2 . HighGUI module OpenCV [1999] provides GUI of OpenCV. HighGUI provides interface to user to create and manipulate windows which is used for display images. It also handles mouse events and keyboard commands on image display window. HighGUI is also used for reading and writing images as well as videos on memory.

All the functionality of OpenCV is done with the help of CXCORE module OpenCV [1999]. Core module is used for all mathematical as well as logical operations related to images and videos. Core provides support to draw function with the use of XML.

OpenCV is an initiative of INTEL to advance CPU applications. Bradski and Kaehler [2008] set some goals for OpenCV such as:

1. Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.

2. Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.

3. Advance vision-based commercial applications by making portable, performance optimized code available for freewith a license that did not require commercial applications to be open or free themselves.

The basic structure of OpenCV is shown in Figure 3.2.

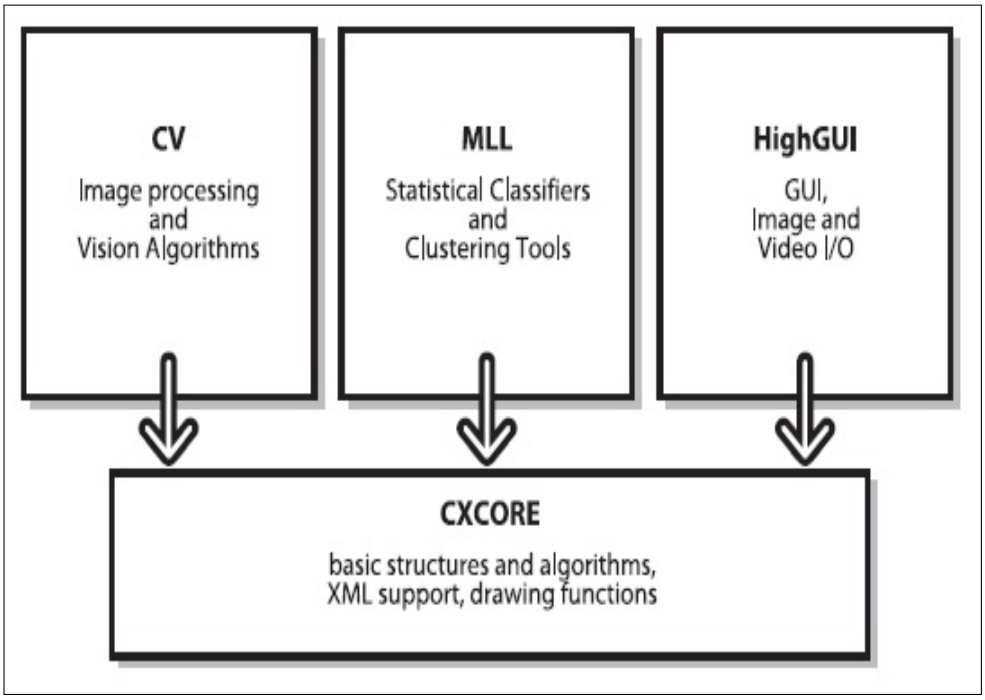

Figure 3.2: Basic Structure of OpenCV(Bradski and Kaehler [2008])

3.4 Algorithms for Face Detection and Face Recognition

There are many types of algorithm used to develop face detection as described in subsection 2.3.1. But Haar Cascade Classifier and LBP Cascade Classifier provides some advantages among all.

As Haar Cascade Classifier uses integral images which is sum of pixels calculated from subset of rectangular area of image, Haar takes minimum time to calculate image. As described in subsection 2.3.1, Haar Cascade Classifier have high accuracy compared to any other algorithm as it uses face features with integral image.

As LBP Cascade Classifier uses texture spectrum model means it calculate histograms of textures in images. This classifier is fastest among all algorithm with less accuracy than Haar. As both algorithms have different advantages which are measure in time and accuracy, I chose both algorithm for further studies.

As second part of dissertation is on Face Recognition, algorithm used by previous researcher for it is described in subsection 2.3.2. As described in literature review, many author suggest that Eigenface algorithm provides best accurate result in less time. Eigenface is a set of eigenvectors which are calculated on input image. Eigenface algorithm creates many gray-scale images of input image and overlaps on images of

database to check match. Eigenface algorithm recognize face with high accuracy as compared to other algorithms. Though the Eigenface algorithm was chosen for further study of face recognition. The next subsections will discussed in more details of the three algorithms used in this project.

3.4.1 Haar Cascade Classifier

This classifier algorithm was developed by Viola and Jones [2001]. This algorithm is used for object detection and recognition. Viola and Jones [2001] uses Haar wavelets which is sequence of rescaled square-shaped functions. Viola and Jones [2001] combined both techniques to develop the Haar cascade classifier. Haar cascade classifier changes representation of image which allows very fast feature evaluation. The change of image representation can be computed from operations on pixels image.

It is easy to compare Haar features with derived pixel image. Image represent large number of features which excluded by Haar cascade classifier and focuses on set critical features. Haar like features which is used by Haar cascade classifier create virtual adjacent rectangular regions in detection image. It calculates pixel intensities in all rectangular regions and differentiates between them. According to this difference, it categorize different features of image. As per Pan et al. [2009] Haar cascade classifier provides high accuracy and speed in real time due to its use of integral images.

3.4.2 LBP Cascade Classifier

Local Binary Pattern (LBP) is briefly discussed by Kadir et al. [2014]. LBP uses feature vector which is used to differentiate between image texture. This difference is calculated on each pixel. Texture operator labels each pixel of image by thresholding its adjacent pixel with center pixel. The result of this stored in binary number. As LBP is capable of making fine distinction between each pixels, makes it popular for various application. It is very robust to monotonic gray-scale image and also simple in computational power.

LBP feature vector which is created on input image, divide input image into $N \times N$ cells. Each pixel of cell is compared to its 8 neighbor pixel. If value of center pixel is more than its neighbor's pixel value then LBP writes 1, otherwise writes 0. This will followed in clockwise or anticlockwise direction for all image. This will form 8 digit binary number for each pixel in cell. This number then converted in histogram of that pixel. All pixel's histogram differentiate features of cell. This process is followed for all

cells which is created for feature vector. Features of face is calculated from result of all cells to detect a face in image.

3.4.3 Eigenfaces

Principal Component Analysis (PCA) is a mathematical processing technique which is used most commonly in face recognition. Authors Turk and Pentland [1991] used Eigenface for recognition and face classification. Eigenface is a set of eigenvectors which are derived from covariance matrix of intensity distribution of pixel in image. Eigenfaces form basic set of images from training images and recognize by comparing it with input image. This algorithm require centered face image, same size as training images and Eigenfaces. This algorithm also require same pixel size of training images with input image.

Eigenfaces contains only standard face features such as eyes, nose, ears, lips, etc which are derived from many images which contain face. As Eigenface is not recorded as digital image, it won't take more space to store all Eigenfaces for each person. Eigenfaces are created in gray-scale as light and dark areas of face features. All different face features are evaluated and scored according to its pattern.

Eigenfaces is easy and cost effective to recognize person with database because of following reasons:

- The process of creating Eigenfaces and comparing it with database is automatic and easy to implement.
- The process of creating Eigenfaces has less complexity compared with other algorithm.
- Eigenfaces can be used for any database without depending on size of database.

As Eigenface have many more advantage, it also have some limitations which are listed below:

- Eigenfaces are calculated in gray-scale so it is very sensitive to day-light of image. It is also sensitive to resolution of image as it uses overlapping of image on other image.
- Eigenface can't handle facial expression of human and its change on it.

3.5 Databases

As different databases provide different results for each algorithm as discussed in section 2.3, we have to choose minimum two database to compare results. There are many databases available to use for research purposes. For both purpose namely detection and recognition, I chose two different databases of images. ColorFERET and ExtendedYaleB are used for face detection. 10K and AT&T are used for face recognition. The details of databases are discussed below.

3.5.1 Face Detection Databases

As face detection needs image which contain face in different day-light conditions and angles. ColorFERET provides images with different face angles whereas ExtendedYaleB provides images with different day-light conditions.

ColorFERET

A standard database of facial images is require to acquire high accuracy in face recognition programs. FERET database is created by gathering images interdependently from many researchers and algorithm developers of face recognition. FERET is also known as Face Recognition Technology. ColorFERET is new version of FERET database with facial images in color. FERET database is created from different sessions of capturing images between August 1993 and July 1996. FERET database contains 14,126 images of 1199 individuals. This database creation is started by Department of Defense (DoD) Counterdrug Technology Development Program Office and it maintained by NIST.

To obtain ColorFERET database, we have request NIST via email to color-feret@nist.gov in plain text format for request of account. More details for ColorFERET database available on [ColorFERET, 1996].

Figure 3.3: Sample Image from ColorFERET([ColorFERET, 1996])

ExtendedYaleB

ExtendedYaleB is addition to YaleB database which is developed by University of California, San Diego. This database is freely available from [YaleB, 2005] for research purpose only. It contains 16128 images of 28 human in 9 different pose 64 illumination conditions. Each image in database in .pgm (portable graymap) format.

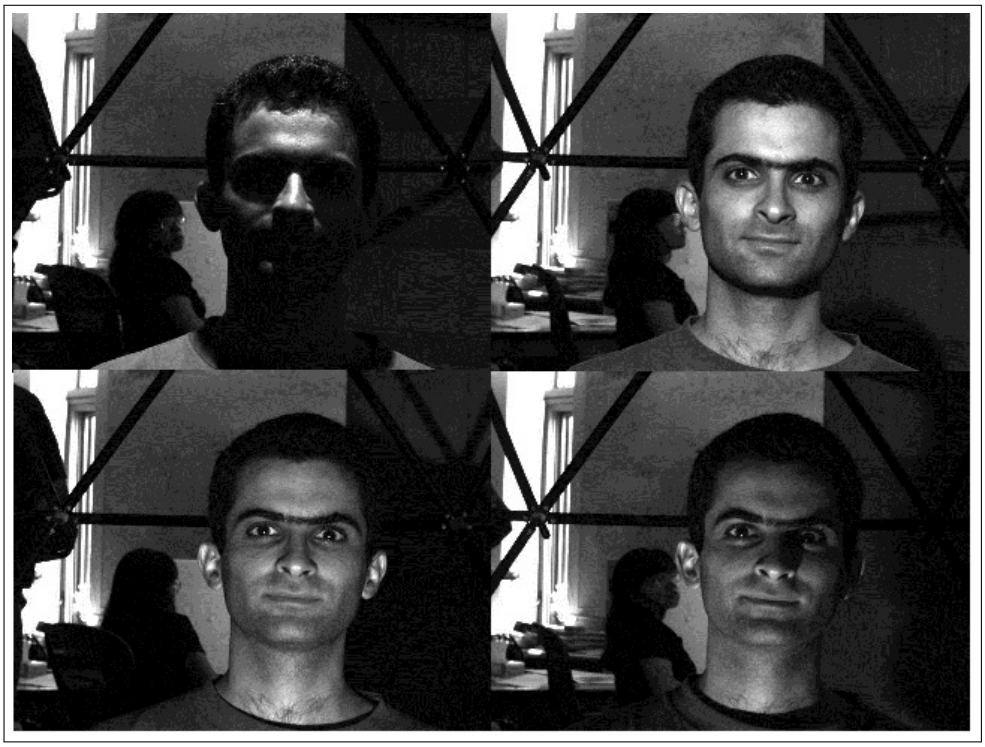

Figure 3.4: Sample Image from ExtendedYaleB([YaleB, 2005])

3.5.2 Face Recognition Databases

As face recognition require images which contain face with their names to identify person. 10K and AT&T provides images with name of person.

10K

This database is also known as 10k US Adult Faces Database. It contains 10168 natural face images of different people with name. This database is created by PhD student Wilma Bainbridge from Massachusetts Institute of Technology, Massachusetts by sampling Google images [10K, 2013]. On the basis of 1990 US Census, names of persons are given to each image.

To obtain 10K database, we have request Wilma Bainbridge via filling details on [10K, 2013] for request of account. More details for 10K database available on [10K, 2013].

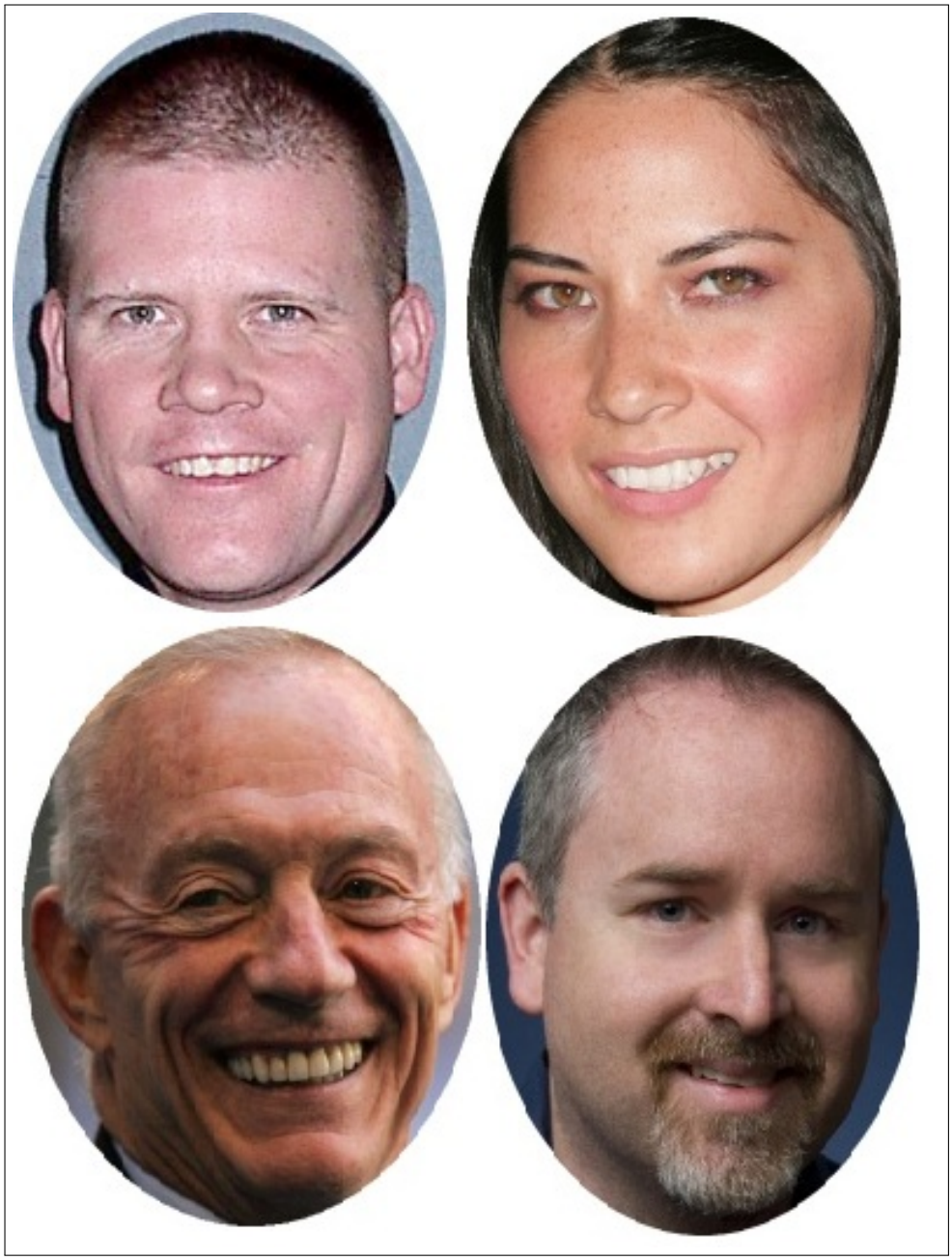

Figure 3.5: Sample Image from 10K([10K, 2013])

AT&T

This database is developed by AT&T Labs, Inc which is research division of AT&T American multinational telecommunications company. This database is hosted with Cambridge University Computer Laboratory. It formerly known as The ORL databases of faces which contains set of images taken from April 1992 to April 1994. This database contains 400 images of 40 different humans, 10 images each.

AT&T database directly available to download from [AT&T, 1994]. More details for AT&T database about images and its description available on [AT&T, 1994].

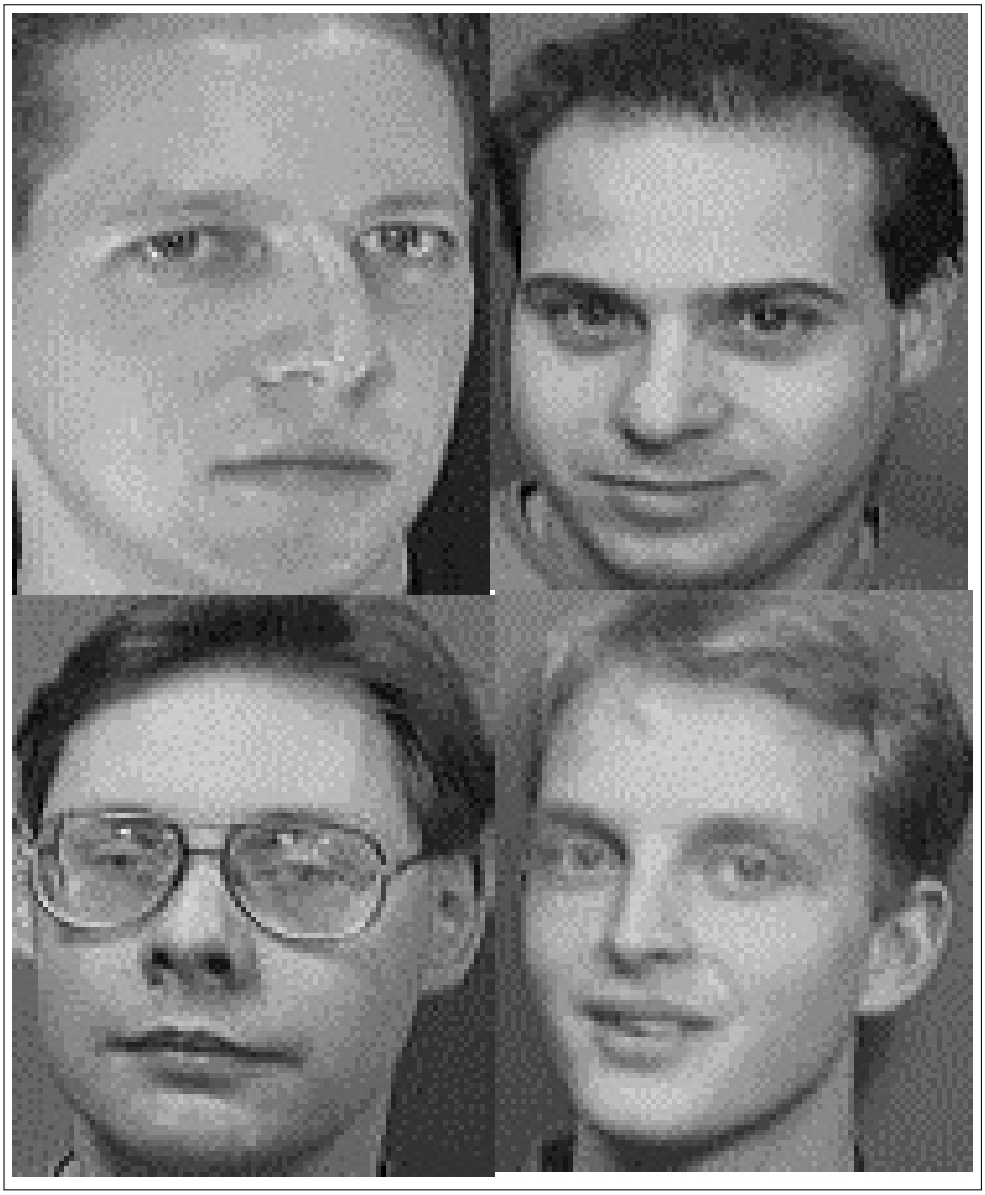

Figure 3.6: Sample Image from 10K([AT&T, 1994])

Chapter 4

Implementation

This chapter describes the step by step implementation of a system for detection and recognition of face. This chapter also describes different implementation and configuration of cloud. This chapter also describe some important code for face detection and recognition. First we will discuss flow diagrams of face detection and face recognition system in section 4.1. Next section 4.2 will discuss about instances which are created in cloud. section 4.3 discusses about configuration of OpenCV and its dependencies. Implementation details of face detection module is discussed in section 4.4 . section 4.5 discussed about implementation of face recognition module.

4.1 System Flow Chart

As system is develop for cloud, we have to get authenticated on OpenStack. As system runs on cloud, instance of operating system (OS) should be present. If instance of OS is not present on cloud, we have to create an instance on cloud and upload a face detection and face recognition project. Once project of face detection and recognition is uploaded, we have to run desired system. System will calculate time for execution of whole process. The flow of system is shown in Figure 4.1.

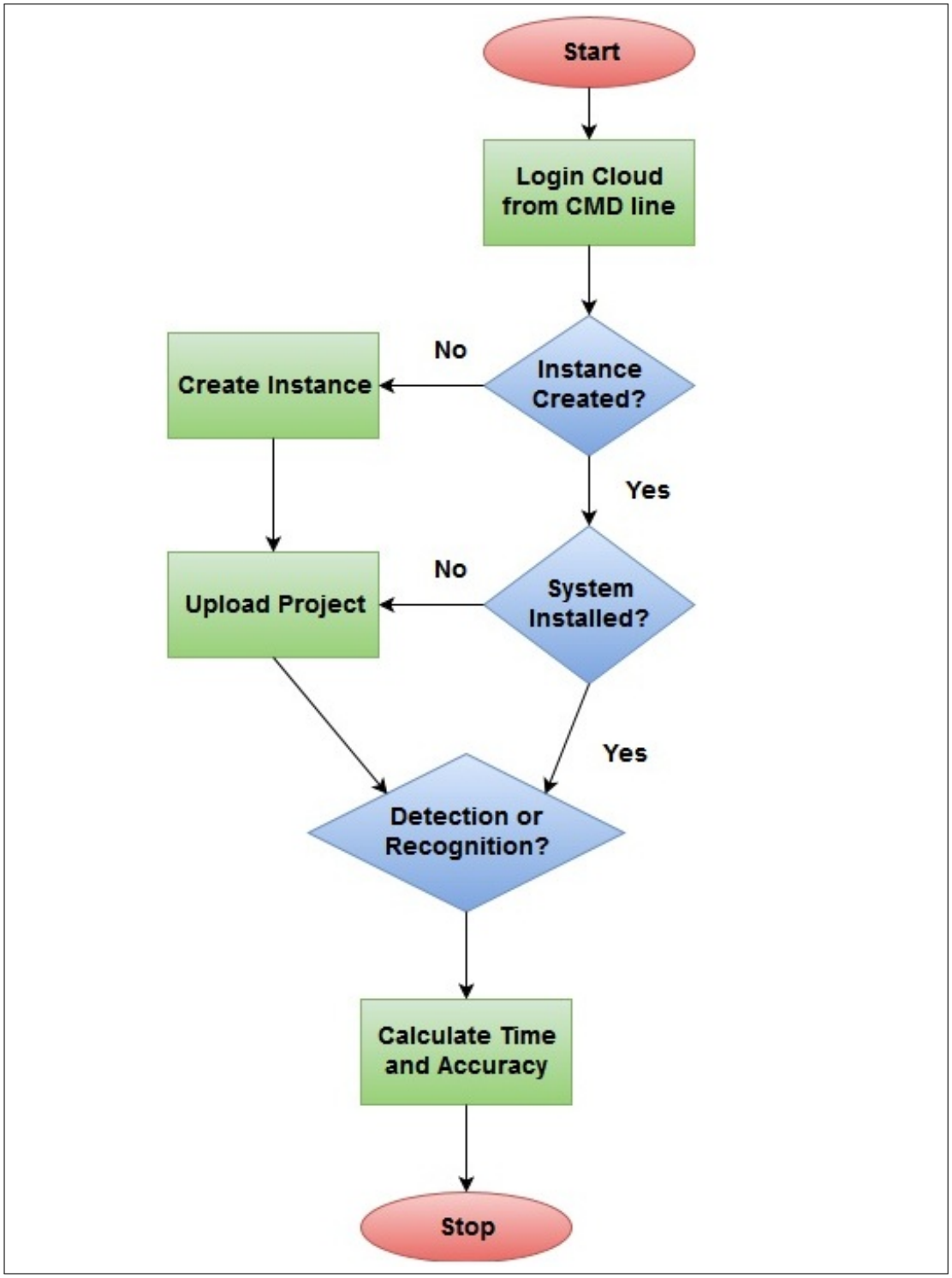

Figure 4.1: Flow Chart of System

4.2 Instance Specification

We have used OpenStack as cloud to deploy and to run a system. We have used different instances in OpenStack to test different scenarios. We use OpenStack as a private cloud provided by National College of Ireland. Medium instance have two core of Virtual CPU (VCPU) and 4 gigabytes of RAM. Large instance have 4 core and 8 GB of RAM whereas xlarge have 8 core and 16GB of RAM. These different instances whose configurations are summarized in Table 4.1.

Table 4.1: Configuration of Cloud Instances

Sr. No.	**Cloud Vendor**	**Instance Type**	**VCPU Core**	**RAM**
1	OpenStack	m1.medium	2 Core	4GB
2	OpenStack	m1.large	4 Core	8GB
3	OpenStack	m1.xlarge	8 Core	16GB

To launch an instance in OpenStack, we have to log on in OpenStack. There are many predefined flavors which are different size of image. Flavors define VCPU, RAM and total disk of image. After selecting flavor, we have to select an image of operating system (OS) which will be OS of virtual machine. The launch of instance also require key pair which is used as authentication of instance image. Key pair contains public key and private key of user for authentication. Instance also provide an IP address to access image from remote place. Figure 4.2 shows screen capture of launch of image. After launching an image, we have to run our system on instance. As instance is on cloud, we need a terminal emulator such as putty to run system.

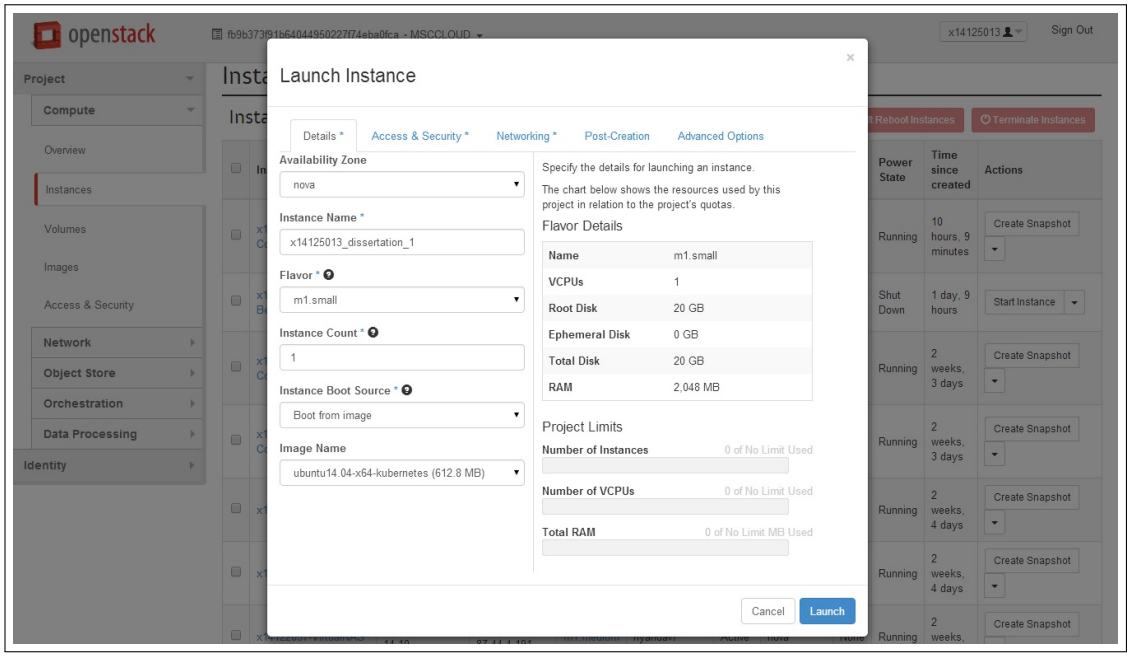

Figure 4.2: Instance Launch Snapshot

4.3 OpenCV Configuration

OpenCV is open source computer vision library which depends on some predefined library files. Face detection and recognition require multiplication and transformation of matrix as images are stored in matrix of pixel values. JavaCV provides matrix calculation which is already implemented for Java. JavaCV also provide image mathematical calculations in different jar files. These jar files are previous work which is done by researcher in the field of face detection and recognition on Java platform. We can add dependent libraries by adding libraries in properties window of project. Figure 4.3 shows project properties window for addition of libraries to project.

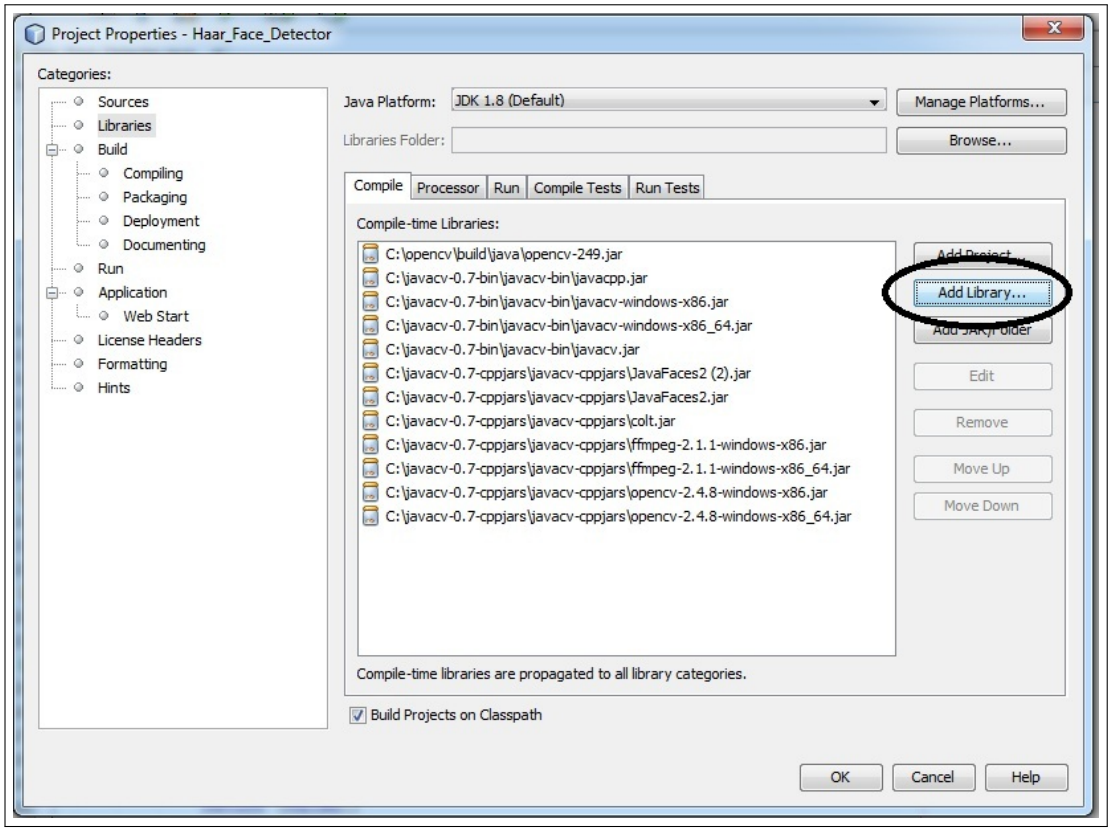

Figure 4.3: Addition of Libraries in Java

Having described about usage of jar libraries in project for implementing programs of face detection and recognition, next section will explain it clear with the help of some Java codes which are implemented in different jar file. Figure 4.4 shows source code of matrix calculations which is imported from “core” package of opencv-249.jar file.

```java
package org.opencv.core;

public class Rect {

    public int x, y, width, height;

    public Rect(int x, int y, int width, int height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

    public Rect() {
        this(0, 0, 0, 0);
    }

    public Rect(Point p1, Point p2) {
        x = (int) (p1.x < p2.x ? p1.x : p2.x);
        y = (int) (p1.y < p2.y ? p1.y : p2.y);
        width = (int) (p1.x > p2.x ? p1.x : p2.x) - x;
        height = (int) (p1.y > p2.y ? p1.y : p2.y) - y;
    }

    public Rect(Point p, Size s) {
        this((int) p.x, (int) p.y, (int) s.width, (int) s.height);
    }

    public Rect(double[] vals) {
        set(vals);
    }

    public void set(double[] vals) {
        if (vals != null) {
            x = vals.length > 0 ? (int) vals[0] : 0;
            y = vals.length > 1 ? (int) vals[1] : 0;
            width = vals.length > 2 ? (int) vals[2] : 0;
            height = vals.length > 3 ? (int) vals[3] : 0;
        } else {
            x = 0;
            y = 0;
            width = 0;
            height = 0;
        }
    }

    public Rect clone() {
        return new Rect(x, y, width, height);
    }

    public Point tl() {
        return new Point(x, y);
    }

    public Point br() {
        return new Point(x + width, y + height);
    }

    public Size size() {
        return new Size(width, height);
    }

    public double area() {
        return width * height;
    }
```

Figure 4.4: Source Code for Matrix Calculation

4.4 Face Detection Module

Face detection module detects faces within image which is provided from file chooser. We are detecting multiple faces in different image files from single directory by using dynamic array of files. To choose only image files within directory, we implement filter of image files which compares extensions of files with filter. To detect face by Haar cascade algorithm, we uses haarcascade_frontalface_alt.xml file which is provided by OpenCV. OpenCV also provide XML file to detect face from image by LBP cascade classifier. Small working code of haarcascade_frontalface_alt.xml file is shown in Figure 4.5.

The working code of face detection algorithm with image filter in Java is shown in Figure 4.6. Face detection module also calculate time for detecting faces from all input images and prints in format as " HH 'hours', mm 'mins,' ss 'seconds', SSS 'Milliseconds'".

```
<?xml version="1.0"?>
<opencv_storage>
<haarcascade_frontalface_alt type_id="opencv-haar-classifier">
  <size>20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>3 7 14 4 -1.</_>
                <_>3 9 14 2 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>4.0141958743333817e-003</threshold>
            <left_val>0.0337941907346249</left_val>
            <right_val>0.8378106951713562</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>1 2 18 4 -1.</_>
                <_>7 2 6 4 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>0.0151513395830989</threshold>
            <left_val>0.1514132022857666</left_val>
            <right_val>0.7488812208175659</right_val></_></_>
        <_>
          <!-- tree 2 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>1 7 15 9 -1.</_>
                <_>1 10 15 3 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>4.2109931819140911e-003</threshold>
            <left_val>0.0900492817163467</left_val>
            <right_val>0.6374819874763489</right_val></_></_></trees>
      <stage_threshold>0.8226894140243530</stage_threshold>
      <parent>-1</parent>
      <next>-1</next></_>
    <_>
      <!-- stage 1 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>5 6 2 6 -1.</_>
                <_>5 9 2 3 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>1.6227109590545297e-003</threshold>
            <left_val>0.0693085864186287</left_val>
            <right_val>0.7110946178436279</right_val></_></_>

```

Figure 4.5: Haar Cascade Classifier XML

```java
package haar_face_detector;

import java.io.File;
import java.io.FilenameFilter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.TimeZone;
import javax.swing.JFileChooser;
import org.opencv.core.*;
import org.opencv.highgui.Highgui;
import org.opencv.objdetect.*;

public class Haar_Face_Detector {

    public static void main(String[] args) {
        long startTime= System.currentTimeMillis();
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        System.out.println("\nRunning FaceDetector");
        JFileChooser chooser = new JFileChooser();
        chooser.setMultiSelectionEnabled(true);
        String i=null;
        System.out.println(System.getProperty("user.dir"));
        String dir = System.getProperty("user.dir");
        String abcd = dir + "/src/abc";
        System.out.println(abcd);
        File ab = new File (abcd);
        chooser.setCurrentDirectory(ab);
        File[] filesInDirectory1 = chooser.getCurrentDirectory().listFiles();
        String path = null;
        for ( File file : filesInDirectory1 ) {
            i=file.getName();
            if ("haarcascade_frontalface_alt.xml".equals(i))
            {
                path = file.getAbsolutePath();
            }
        }
        File[] filesInDirectory =
        chooser.getCurrentDirectory().listFiles(IMAGE_FILTER);
        for ( File file : filesInDirectory ) {
            String j =file.getAbsolutePath();
            CascadeClassifier faceDetector = new CascadeClassifier();
            faceDetector.load(path);
            Mat image = Highgui.imread(j);
            MatOfRect faceDetections = new MatOfRect();
            faceDetector.detectMultiScale(image, faceDetections);
            System.out.println(String.format("Detected %s faces",
            faceDetections.toArray().length));
            for (Rect rect : faceDetections.toArray()) {
                Core.rectangle(image, new Point(rect.x, rect.y), new Point(rect.x +
                rect.width, rect.y + rect.height), new Scalar(0, 255, 0));
            }
            String filename =  j + ".jpg";
            System.out.println(String.format("Writing %s", filename));
            Highgui.imwrite(filename, image);
        }

        long endtime = System.currentTimeMillis()-startTime;
        DateFormat df = new SimpleDateFormat("HH 'hours', mm 'mins,' ss 'seconds',
        SSS 'Milliseconds'");
        df.setTimeZone(TimeZone.getTimeZone("GMT+0"));
        System.out.println("\n Total Execution Time := ");
        System.out.println(df.format(new Date(endtime)));
    }
}
```

Figure 4.6: Face Detector Model

4.5 Face Recognition Module

Face recognition module recognize face from input image which is provided from file chooser. We are using publish databases which are free to access for research purpose to compare face of person in input image which is summarized in table 2.2. For comparing input image with multiple image files, we are using dynamic array of files. As Eigenface algorithm creates number of eigenfaces of input image and compare with databases by overlapping input image on eigenfaces. Some example of created Eigenface files are shown in Figure 4.5.

Figure 4.7: Eigenfaces

Face recognition module also depends on libraries provided by OpenCV. The working code of face Recognition algorithm in Java is shown in Figure 4.8. Face recognition module also calculate time for detecting faces from all input images and prints in format as " HH 'hours', mm 'mins,' ss 'seconds', SSS 'Milliseconds' ".

```java
package facerecognition.javafaces;

import java.awt.image.*;
import java.io.*;
import java.util.*;
import java.util.logging.FileHandler;
import java.util.logging.Handler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

import javax.imageio.ImageIO;

import facerecognition.utils.ValueIndexPair;


public class FaceRec{
    private boolean isImage(String imagefilename){
        boolean isimage = false;
        try{
                BufferedImage bi = ImageIO.read(new File(imagefilename));
                if (bi != null){
                    isimage = true;
                }
            }catch(Exception e){
                isimage = false;
            }
        return isimage;
    }
    public MatchResult processSelections(String faceImageName,String
    directory,String numofFaces,String threshold) {
        MatchResult result = null;
        int numFaces = 0;
        double thresholdVal = 0.0;
        try{
            validateSelectedImageFileName(faceImageName);
            validateSelectedFolderName(directory);
            numFaces = getNumofFacesVal(numofFaces);
            thresholdVal = getThresholdVal(threshold);
            String extension = getFileExtension(faceImageName);
            checkCache(directory,extension,numFaces);
            reconstructFaces(numFaces);
            result = findMatchResult(faceImageName,numFaces,thresholdVal);
        }catch(Exception e){
            result = new MatchResult(false,null,Double.NaN,e.getMessage());
        }
        return result;
    }

    public static void main(String[] args){
        long start = System.currentTimeMillis();
        String imgToCheck ="E:\\Face Recognition\\JavaFaces2\\andrew0.png";
        String imgDir = "E:\\Face Recognition\\JavaFaces2\\trainingImages";
        String numFaces = "4";
        String thresholdVal = "2";
        MatchResult r = new
        FaceRec().processSelections(imgToCheck,imgDir,numFaces,thresholdVal);
        if (r.getMatchSuccess()){
            debug(imgToCheck + " matches "+r.getMatchFileName()+" at distance=" +
            r.getMatchDistance());

        }else{
            printError("match failed:" + r.getMatchMessage());
        }
        long end = System.currentTimeMillis();
        debug("time taken ="+ (end - start) / 1000.0 +" seconds");
    }

}
```

Figure 4.8: Face Recognition Model

Chapter 5

Evaluation

The main aim of detection is to evaluate the performance of different algorithms on high computing hardware. In general, to evaluate the performance of algorithm there are different points needs to be taken consideration such as accuracy of algorithm, time for detection of face. We design experiments to evaluate each proposed approach for face detection. The difference in performance can not be identified by just looking at instance specification. Moreover, the user of system does not have to worry about size, extension, faces and persons in image. Intelligent system of detection of face solves the problem.

5.1 Evaluation of Face Detection Tool

We can evaluate face detection algorithm in three ways namely accuracy of algorithm, time taken by algorithm, impact of size of image on algorithm. Accuracy of algorithm can be measured as ratio of number of faces detected in input to actual number of faces in input. We use subset of database which contain 1000 images as a input. Accuracy of algorithm is discussed in subsection 5.1.1.

5.1.1 Accuracy

Accuracy of detecting faces by using Haar and LBP algorithm is different. Accuracy of YaleB database is more when use of LBP Cascade classifier compared to Haar cascade classifier. Accuracy of FERET database is 49.9% as it contains faces with different angles which varies from -90°to +90°. As FERET database have image in PPM format which is high in size and resolution, but it contains face with 90°angle so accuracy of

FERET database is less compare to YaleB database. LBP cascade classifier is high accurate as it uses local binary pattern. Local binary pattern use subset of image and compare pixel values. So feature of faces were compared properly. Accuracy of FERET database by using Haar Cascade classifier and LBP Cascade classifier is shown in Figure 5.1.

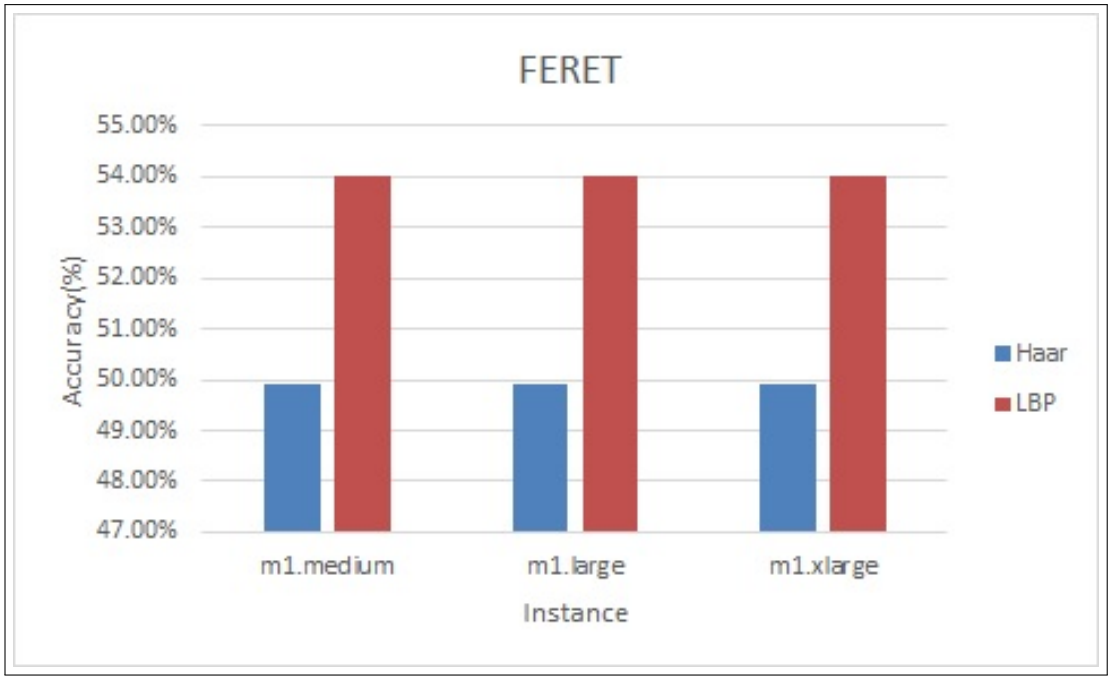

Figure 5.1: Accuracy of FERET database

Accuracy of both algorithm improve when use of YaleB database. Accuracy of YaleB database is more as compare to FERET. Images in YaleB database is PGM files which contains gray-scale values of pixel. Accuracy of YaleB database also more when use of Haar Cascade classifier as compared to FERET database on same machine. Accuracy of YaleB database by using Haar Cascade classifier and LBP Cascade classifier is shown in Figure 5.2.

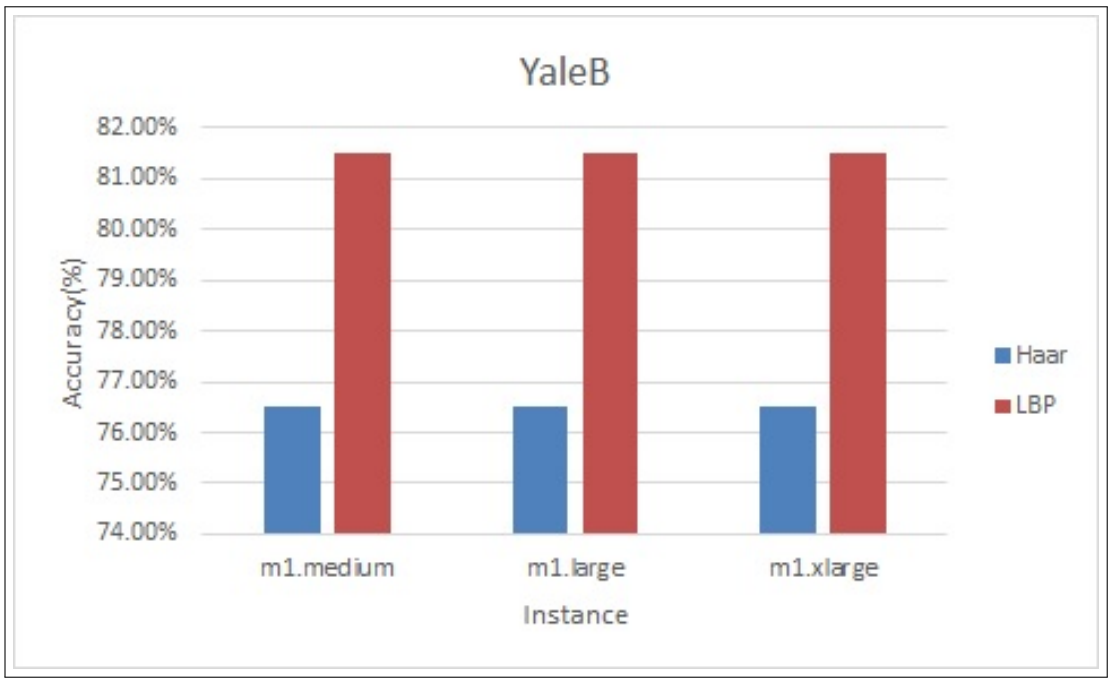

Figure 5.2: Accuracy of Extended YaleB database

Both graphs represents x axis as instance and y axis as percentage of accuracy. Having discussed about the accuracy of face detection algorithms, we discuss about the time taken by face detection.

The conclusion can be drawn for the result of accuracy is that LBP cascade classifier is more accurate as compared to Haar cascade classifier. Accuracy of LBP is more when use of YaleB database as it contains images of different luminance condition.

5.1.2 Time

There is a lot of difference between time taken to detect faces by Haar algorithm and LBP algorithm. Haar cascade classifier and LBP cascade classifier algorithm shows decrease in time to run application with different database. Time of FERET database is more when use of Haar Cascade classifier compared to LBP cascade classifier. As FERET database have high size image, so time taken by FERET database is more compared to YaleB database. Time taken to detect faces on m1.xlarge instance which is of 8 VCPU and 16 GB of RAM is minimum compared to other instances on same database.

LBP cascade classifier takes minimum time. As LBP cascade classifier is based on Local binary pattern which uses image subset and compare pixel values with other pixel and

generates fast results. Time taken by FERET database by using Haar Cascade classifier and LBP Cascade classifier is shown in Figure 5.3.

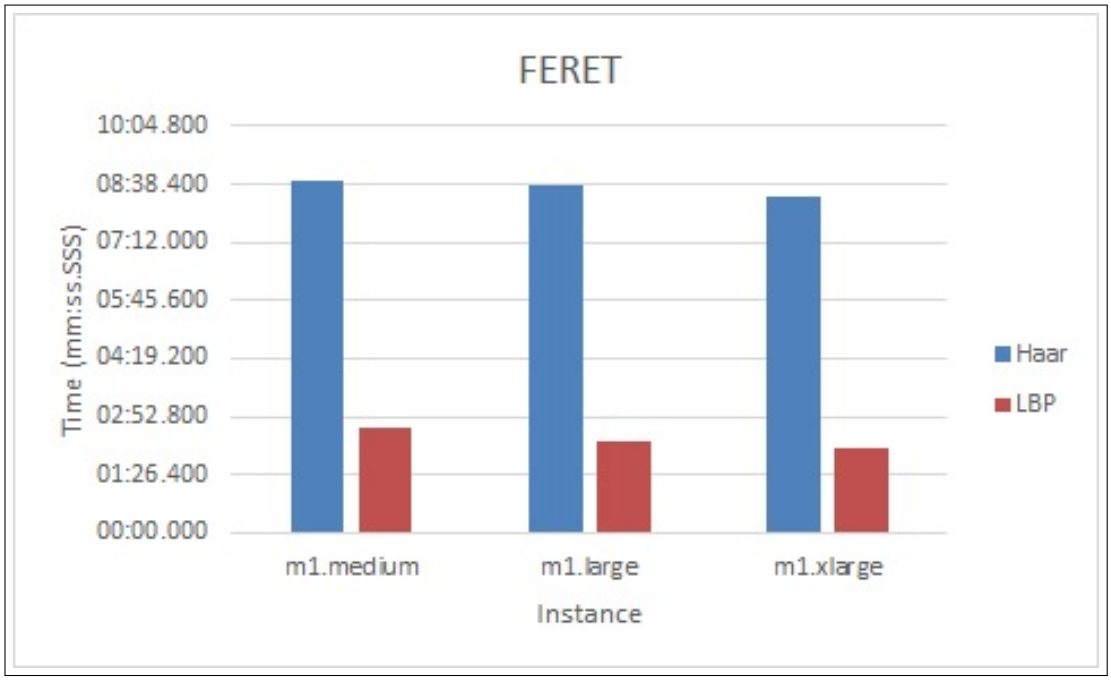

Figure 5.3: Time taken by FERET database

Time taken to run both algorithms improve when use of YaleB database. Time taken by YaleB database is less as compare to FERET. As discussed above, images of YaleB database are gray-scale pixel values which is stored in file format. So time taken to perform operations such as conversion of image to pixel value to read image is more easy and fast as compared to FERET. Time taken by YaleB database is less as compared to FERET database. Time taken by YaleB database by using Haar Cascade classifier and LBP Cascade classifier is shown in Figure 5.4.

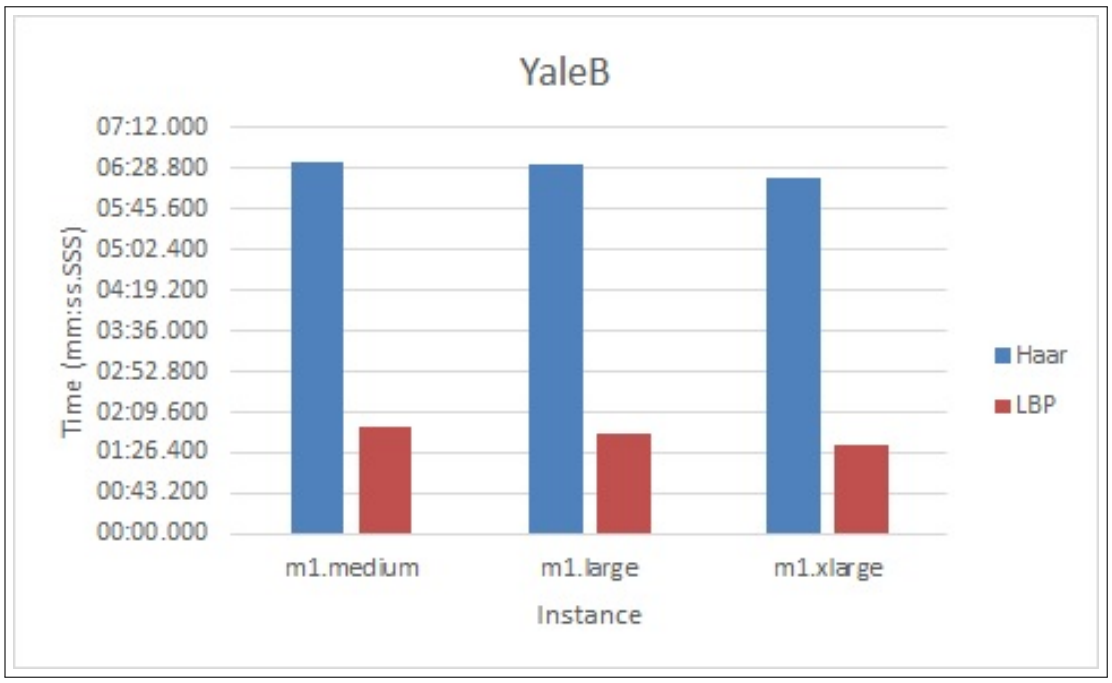

Figure 5.4: Time taken by YaleB database

The conclusion can be drawn on the basis of above result for time is that LBP cascade classifier is less time consuming as compared to Haar cascade classifier. Time taken by LBP cascade classifier for YaleB database less as compared to Haar cascade classifier.

5.1.3 Impact of Image Size

As discussed above, size of image impacts on results of algorithms. Haar cascade classifier takes more time on FERET database image as compared to LBP cascade classifier. Whereas YaleB database image which is smaller in size takes less time. Time taken by single image of different database on different algorithm is shown in Figure 5.5.

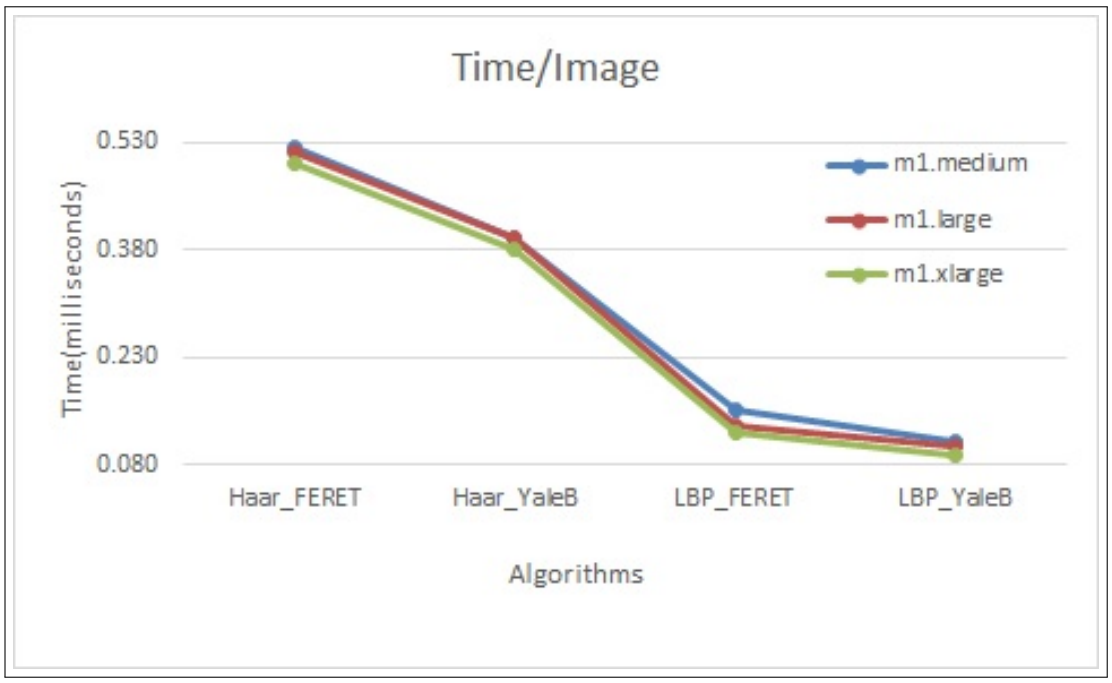

Figure 5.5: Impact of Image size

The conclusion can be drawn on the basis of above graph that size of image is less impacted on time taken by algorithm as Haar cascade classifier takes more time for both databases as compared to LBP database.

Chapter 6

Conclusion

Cloud computing is the latest high computing technology which is delivering the IT services over internet. This dissertation, we proposed a software solution which helps to efficiently detect and recognize face from image. The main aim of this software solution is to promote adoption of high computing device to overcome problem of low configuration infrastructure. Software system which is deployed on high-end computing hardware can be beneficial to low configuration device such as laptop. The cloud platform provides features to proposed software solution such as scaling of storage, load balancing over cores and real-time that supports use of high database. However it quite challenging to detect face and recognize it on cloud as it involves developing application which uses scalable data.

The thesis identified the need to develop a face detection and face recognition. The thesis also provides the designs for both Face detector and Face Recognizer. The tools were implemented as per designed and face detection worked as expected. The software was evaluated as its accuracy and time to perform. The research shows that tools provide good results with front facing face. It also noticed that available algorithms such as LBP Cascade Classifier runs in very low time with high accuracy. Haar Cascade Classifier runs slow and less accurate as compared to LBP cascade classifier. Hence to conclude, it is possible to efficiently use a Face detection application on cloud to get results in real-time.

6.1 Future Work

Currently Face detection and recognition can detect and recognize faces on cloud which require to upload input image on cloud. Face recognition in real time with use of mobile

cloud computing will availability of system. Mobile device works as mediator which will take input image form users' mobile device and send it to cloud to recognize faces. In present system for mobile device, algorithms takes input from device and compare to only one image which is present in directories of mobile device. Face recognition with mobile cloud computing in real-time will increase processing power of mobile device to process high size database on cloud. In addition, we can also implement some age factor which compare for different age stages of user. Real time face recognition with mobile cloud will be an efficient technique to compare and detect faces at different age of faces. The face detection and recognition tool could also be improved to have mobility of software so that software can be used anywhere on mobile device.

Bibliography

10K. Official website of 10k, 2013. URL http://www.wilmabainbridge.com/facememorability2.html.

T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010. ISSN 0001-0782.

AT&T. Official website of at&t, 1994. URL http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

Marwa Ayad, Mohamed Taher, and Ashraf Salem. Real-time mobile cloud computing: A case study in face recognition. pages 73–78. IEEE, 2014.

Rachid Belaroussi and Maurice Milgram. A comparative study on face detection and tracking algorithms. *Expert Systems With Applications*, 39(8):7158–7164, 2012.

Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

Rajkumar Buyya, Chee S. Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

Jason Carolan, Steve Gaede, James Baty, Glenn Brunette, Art Licht, Jim Remmell, Lew Tucker, and Joel Weise. Introduction to cloud computing architecture. *White Paper, 1st edn. Sun Micro Systems Inc*, 2009.

Tse-Wei Chen, Chi-Sun Tang, and Shao-Yi Chien. *Highly Efficient Face Detection in Color Images*, volume 5353, pages 919–922. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 9783540897958;354089795X;.

ColorFERET. Official website of colorferet, 1996. URL http://www.nist.gov/itl/iad/ig/colorferet.cfm.

Erik Hjelms and Boon K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001.

Joseph P Hornak. *RF Magnetic Field Mapping.* Wiley Online Library, 2002.

Rein-Lien Hsu, M. Abdel-Mottaleb, and A. K. Jain. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):696–706, 2002.

Kushsairy Kadir, Mohd K. Kamaruddin, Haidawati Nasir, Sairul I. Safie, and Zulkifli A. K. Bakti. A comparative study between lbp and haar-like features for face detection using opencv. pages 335–339. IEEE, 2014.

R. Lienhart, Luhong Liang, and A. Kuranov. A detector tree of boosted classifiers for real-time object detection and tracking. volume 2, pages II–277. IEEE, 2003. ISBN 9780780379657;0780379659;.

Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing the business perspective. *Decision Support Systems*, 51(1):176–189, 2011.

Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

OpenCV. Official website of opencv, 1999. URL http://SourceForge.net/projects/opencvlibrary.

OpenStack. Openstack releases juno, 2014.

Yaozhang Pan, Shuzhi S. Ge, Hongsheng He, and Lei Chen. Real-time face detection for human robot interaction. pages 1016–1021, 2009. ISBN 1944-9445.

Akshay A Pawle and Vrushsen P Pawar. Face recognition system (frs) on cloud computing for user authentication.

Peng Peng and Yehu Shen. Efficient face verification in mobile environment using component-based pca. volume 2, pages 753–757. IEEE, 2013.

Dimitri Pissarenko. Eigenface-based facial recognition. *December 1st*, 2002.

BK Plagman and MV Littlejohn. Image processing. *Internal Auditor*, 49:64–64, 1992.

B. Prasada. Review of 'digital image processing' (gonzalez, r. c., and wintz, p.; 1977), 1979.

V. Rajaraman. Cloud computing. *Resonance*, 19(3):242–258, 2014.

Stefano Rovetta. Introduction to image processing. URL http://www.engineersgarage.com/articles/image-processing-tutorial-applications.

M. Stojmenovic. Mobile cloud computing for biometric applications. pages 654–659. IEEE, 2012. ISBN 1467323314;9781467323314;.

Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1): 71–86, 1991.

R. E. Twogood and F. G. Sommer. Digital image processing. *IEEE Transactions on Nuclear Science*, 29(3):1075–1086, 1982.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–I. IEEE, 2001. ISBN 1063-6919.

Extended YaleB. Official website of yaleb, 2005. URL http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html.

Ihab Zaqout, Roziati Zainuddin, and Sapian Baba. Human face detection in color images. *Advances in Complex Systems (ACS)*, 7(3):369–383, 2004.