

# ONLINE VERIFICATION OF MAKERERE UNIVERSITY STUDENTS EXAMINATION PERMITS

By

**OKILA NIXSON**

**2009/HD18/16163U**

Department of Computer Science

College of Computing and Information Sciences , Makerere University

E-mail: nixson.okila@gmail.com ; Mobile: +256-772- 855869

A Project Report Submitted to the School of Graduate Studies for the Study Leading  
to the award of the Degree of Master of Science in Computer Science Makerere  
University.

**OPTION: Computer Vision**

Supervisor: Dr. John Quinn

Department of Computer Science

College of Computing and Information Sciences, Makerere University

jquinn@cit.ac.ug, +256 773420920

March, 2014

# DECLARATION

I Okila Nixon, declare that this report is my original work and has never been published or submitted for any other award to any university/institution of higher learning.

Signed:..... Date:.....

Okila Nixon

Department of Computer Science

College of Computing and Information Sciences, Makerere University

# APPROVAL

This project report has been submitted for examination with the approval of the following supervisor:

.....

Dr. John Quinn (PhD)

Department of Computer Science

College of Computing and Information Sciences

Makerere University

# DEDICATION

To my family, friends and everyone who is interested in searching for knowledge.

# ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to the Almighty God for the wisdom and spiritual strength. My sincere thanks also goes to my supervisor Dr. John Quinn who has been tremendously mentoring me and encouraging my research. I am also grateful to Dr. Bainomugisha for his positive criticism of this work.

Finally, I would like to thank all my friends especially, Dr. Okello Denis and Mr. Kakama Victor for all their guidance and support throughout this project.

Okila Nixon

March 2014

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Document Verification . . . . .	1
1.1.1	Motivation . . . . .	1
1.1.2	Reasons for document verification . . . . .	2
1.2	Statement of the problem . . . . .	3
1.3	Objectives . . . . .	3
1.3.1	Main Objective . . . . .	3
1.3.2	Specific Objectives . . . . .	4
1.4	Scope of the study . . . . .	4
1.5	Significance of the study . . . . .	4
1.5.1	Industry . . . . .	4
1.5.2	Academia . . . . .	5
<b>2</b>	<b>Literature review</b>	<b>6</b>
2.1	Identification (ID) Documents Verification. . . . .	6
2.1.1	Barcodes . . . . .	8

2.1.1.1	Two dimensional (2D) barcodes . . . . .	9
2.2	Construction of Security Feature . . . . .	14
2.3	Object Detection and Recognition . . . . .	14
2.3.1	Applications of Object Detection and Recognition . . . . .	15
2.3.2	Object Recognition Algorithms . . . . .	16
<b>3</b>	<b>Methodology</b>	<b>22</b>
3.1	Encoding student's number on QR code . . . . .	24
3.1.1	Format information . . . . .	24
3.2	Server side cross reference . . . . .	26
3.3	Face Recognition . . . . .	26
3.3.1	ORB algorithm . . . . .	27
3.3.1.1	Feature detection using FAST . . . . .	27
3.3.1.2	Harris corner detector . . . . .	27
3.3.2	Keypoints matching . . . . .	28
3.3.3	Face matching & evaluation . . . . .	28
3.4	System Implementation and Evaluation . . . . .	28
3.4.1	System implementation . . . . .	28
3.4.2	Testing and validation . . . . .	28
3.4.2.1	Recognition accuracy . . . . .	29
3.4.2.2	Verification rate . . . . .	29

<b>4</b>	<b>System Discussion and Analysis of Results</b>	<b>30</b>
4.1	User Authentication . . . . .	30
4.2	Scan Option . . . . .	32
4.3	Face recognition . . . . .	34
4.3.1	Face recognition accuracy . . . . .	35
4.3.2	Potential problems with face recognition. . . . .	36
4.3.3	Verification rate . . . . .	37
4.4	Download option . . . . .	37
4.5	Server and Database . . . . .	39
4.5.1	Web Portal Login Screen . . . . .	40
4.5.2	Home Page Screen . . . . .	41
4.5.3	Students . . . . .	41
4.5.3.1	Create New Student . . . . .	42
4.5.3.2	List Students . . . . .	42
4.5.4	Courses . . . . .	43
4.5.4.1	Create New Course . . . . .	43
4.5.4.2	List Courses . . . . .	44
4.5.5	Course Units . . . . .	45
4.5.5.1	Create Course Units . . . . .	45
4.5.5.2	List Course Units . . . . .	45
4.5.6	Semesters . . . . .	46



4.5.6.1	Add a Semester . . . . .	46
4.5.6.2	Register Student . . . . .	47
4.5.7	Payment . . . . .	47
4.5.7.1	Create Financial Statement . . . . .	47
4.5.7.2	List Financial Statement . . . . .	48
<b>5</b>	<b>Conclusion &amp; Further Work</b>	<b>49</b>
5.1	Conclusion . . . . .	49
5.2	Future Work . . . . .	50
	<b>Bibliography</b>	<b>51</b>

# List of Figures

2.1	PDF417 . . . . .	10
2.2	Data Matrix bar code . . . . .	11
2.3	MaxiCode . . . . .	12
2.4	QR Code . . . . .	13
3.1	Verification process . . . . .	22
4.1	Authentication Screen . . . . .	31
4.2	Option Screen . . . . .	31
4.3	QR Code . . . . .	32
4.4	QR code Scanning . . . . .	33
4.5	Student registration details . . . . .	33
4.6	Photo recognition . . . . .	34
4.7	Recognition of dissimilar faces . . . . .	35
4.8	Download menu . . . . .	38
4.9	Download . . . . .	39
4.10	Offline Results . . . . .	39

4.11 Login Screen . . . . .	41
4.12 Homepage Screen . . . . .	41
4.13 Create New Student . . . . .	42
4.14 List Students Screen . . . . .	43
4.15 Create New Course Screen . . . . .	44
4.16 List Courses Screen . . . . .	44
4.17 Create Course Units Screen . . . . .	45
4.18 List Course Units screen . . . . .	46
4.19 Add Semester Screen . . . . .	46
4.20 Register Student Screen . . . . .	47
4.21 Create Financial Statement Screen . . . . .	48
4.22 List Financial Statement Screen . . . . .	48

# List of Tables

2.1	Storage Capacity of QR code . . . . .	13
4.1	Recognition Accuracy . . . . .	35
4.2	Verification speed . . . . .	37
4.3	Database Design . . . . .	40

# Nomenclature

*BRIEF* : Binary Robust Independent Elementary Feature

*CCD* : Charged Couple Device

*EBGM* : Elastic Bunch Graph Matching

*FAST* : Features from Accelerated Segment Test

*ICA* : Independent Component Analysis

*LEM* : Line Edge Map

*ORB* : Oriented Fast and Rotated BRIEF

*SIFT* : Scale Invariant Feature Transform

*SURF* : Speeded Up Robust Feature

# ABSTRACT

As a way to reduce identification document forgery and impersonation during a university examinations, there is need for a system which can retrieve the current student's registration details from a database and recognise the student's face automatically. Currently, student's face recognition and examination permit verification are done manually and by "eye" making it very difficult for examination invigilators to authenticate the student.

In this study, an online Android application has been developed which can: decode the student number from a Quick Response (QR) code and use it to retrieve the student's details from a database through a server, and recognise the student's face by matching the keypoints of the camera face image with the database face image using OpenCV ORB Feature Detector Algorithm. Server and database have been designed and integrated with the Android application to form a networked verification system.

# Chapter 1

## Introduction

### 1.1 Document Verification

#### 1.1.1 Motivation

Currently technology has advanced so greatly that counterfeit documents which are very difficult to distinguish from authentic documents can now be cheaply and easily scanned and printed with high quality. This has accelerated the rate of document forgery and impersonation majorly in institutions of higher learning. According to [1], Makerere University students do not only forge examination permits but have gone to the extent of forging police letters to prove that the receipts which they would have used as evidence of payment in order to be allowed to sit examinations got lost or misplaced. It is suspected that some of the students who forged police letters claiming that their examination permits got lost were actually mercenaries hired by students to do the examinations for them .

To curb document forgery and impersonation, there is need for a networked system which can perform the verification process automatically.

### 1.1.2 Reasons for document verification

Document verification is basically the determination of the authenticity and integrity of various types of documents. According to [2], document verification can either be accomplished by using an electronic verification machine to compare data contained in electronic circuits printed on the document to document data printed on the document or by transmitting signature data from the electronic circuits via electronic verification machine to a central computer for comparison with document data [3].

Documents are verified to ensure proof of the following:

- Document was issued by the relevant authority,
- The details recorded on the document correspond to the details held by the issuing authority, and
- The document is still valid.

Examination permits, taking the case of Makerere University, are documents detailing among others: the student's photo, registration status of the students for a particular course and course units, and tuition fees payment status. During a university examination period, students are required to have fully registered for the examinations by clearing all the outstanding tuition fees and thereby obtaining an examination permit detailing the tuition fees payment and course units registered and or paid for (for retake cases). Before students are let in the examination rooms, these permits are presented to the examination supervisors/invigilators for verification to avoid letting in students with uncompleted university tuition fees and to curb impersonation. This verification process is done manually and is still error prone. In addition, it is very difficult to verify these permits because of the large numbers of students and limited time.



## **1.2 Statement of the problem**

Verification of university students' examination permits before or during examinations is not only time consuming and stressful to both the students and invigilators due to big student numbers but it is also very difficult to authenticate the examination permits due to sophisticated document forgery now in place. According to [1], in some extreme cases, students suspected of forgery or malpractice have turned against the invigilators and physically assaulted them.

False verification of the student's permit and failure to establish if a student has fully paid up tuition fees for the cases of purported loss of receipts, have led to non qualified (or unregistered) students sitting for the university examinations. This does not only impact negatively on the Makerere University economy but is leading to rise in the cases of examination malpractice at the university, threatening to erode the academic reputation that the university has built over the years.

Face recognition system which can verify if a student in the database is the one in the input image would solve some problems, but with current examination permits and student IDs there is no easy way to verify the student's face and registration status automatically.

## **1.3 Objectives**

### **1.3.1 Main Objective**

To develop and implement an online face recognition system for verifying Makerere University students' examination permits using mobile phones.

### **1.3.2 Specific Objectives**

The following are the specific objectives of the project:-

1. To encode student's number on Quick Response (QR) code.
2. To design database and server module for storing and retrieving student's registration details.
3. To develop face recognition module for recognising student's face.
4. To evaluate the system on its recognition accuracy and verification rate.

## **1.4 Scope of the study**

The study used Quick Response (QR) code generated by standalone application for retrieving student's registration information, and ORB openCV for recognising student's face. It also considered both online and offline verification processes. Android smart phone was used for the deployment of the application.

## **1.5 Significance of the study**

### **1.5.1 Industry**

- It will be able to reduce document forgery and impersonation.
- The system could be integrated into desk top application.

### **1.5.2 Academia**

The study will enable scholars improve the accuracy of object recognition using Android application.

# Chapter 2

## Literature review

### 2.1 Identification (ID) Documents Verification.

Document verification is the process of determining the authenticity of an identification document and this can be done in various ways depending on the kind of security features in the document. The security features can include information such as a photographic image, a bar code (which may contain information specific to a person whose image appears in the photographic image, and/or information that is the same from ID document to ID document), variable personal information (such as an address, signature, and/or birth date, biometric information associated with the person whose image appears in the photographic image), a magnetic stripe (which, for example, can be on a side of the ID document that is opposite to the side with a photographic image), and various security pattern designs (such as a fine-line printed security pattern as is used in the printing of banknote paper, stock certificates, and the like) [4].

If the document contains passport photo, then there is need to use face /object recognition technique which compares document photo with the database photo. This tech-

nique applies feature matching of the two objects and determine similarity measure. However, for documents with personal information, an application which can retrieve the personal details can be used. According to [5], biometric identification as a way of ID document verification has gained interest in recent years because certain personal characteristics have been found to be substantially unique to each person and difficult to reproduce by an imposter. Further, the recording and analysis of biometric data is generally susceptible to automation owing to the increased use of computer controlled electronics and digital recording techniques. Some of the biometric characteristics most investigated today for use in a personal identification system include fingerprints, hand or palm prints, retina scans, signatures and voice patterns. Hand or palm print techniques typically evaluate the shape of a person's hand or other significant features such as creases in the palm, but these techniques may be fooled by templates or models of the hand of an authorized person. Retina scanning techniques evaluate the pattern of blood vessels in a person's retina. In [5][6], a drawback of this technique is that the blood vessel pattern may vary over time, for example, when alcohol is in the blood stream or during irregular use of glasses or contact lenses. Also, a user may feel uneasy about having his or her eye illuminated for retina scanning or the possibility of eye contamination if there is contact between the eye and the scanning apparatus. Signatures can be forged easily and must usually be evaluated by a human operator, although work has been done on automated systems that evaluate the dynamics of a person's handwriting, such as the speed and the force of hand movement, pauses in writing [6]. Using voice patterns as the identifying characteristic encounters difficulties owing to the wide variations in a person's voice over time, the presence of background noise during an evaluation and the potential for an imposter to fool the system with a recording of the voice of an authorized person. The most commonly used biometric characteristic and the one that has been the most investigated and developed is the fingerprint [5].

### 2.1.1 Barcodes

Barcodes are a series of vertical bars of different widths in which digits from zero to nine are represented in dissimilar pattern of bars forming a code that can be read only by a laser scanner. Today the barcodes have many application areas and the decision to use barcodes in business processes is that procedures can be automated in order to increase productivity and reduce human errors [7]. Text and data can be securely and inexpensively stored using barcodes but the amount of data that can be encoded will vary depending upon the type of data, the compaction type, the error correction level chosen, and the limitation of the scanner or printer being used. Barcodes are cost effective, quick and easy, standardized, give correct information (no reading and writing errors), and lead to increased productivity. A bar code contains information which is encoded according to specific conventions (symbology) and graphically presents this information within the barcode field in the form of coloured stripes or bars and colourless spaces [8]. Normally a barcode contains no descriptive data, instead it consists of a reference number of numbers or characters, depending on the type of barcode which makes up a reference number. With the aid of the reference number, information deposited in the form of a data set can be classified and retrieved. Originally barcodes represented data by varying the widths and spacing of one-dimensional (1D) parallel lines. Later they evolved into rectangles, dots, hexagons, and other geometric patterns in 2D [9]. Barcodes originally were scanned by special optical scanners called barcode readers, later, scanners and interpretive software became available on devices including desktop printer and smartphone. The data in a bar code is a reference number which the computer uses to look up associated computer disk record(s) which contain descriptive data and other pertinent information. So, barcodes typically have only 1D data in them which is used by the computer to look up all the pertinent detailed data associated with the 1D data.

### 2.1.1.1 Two dimensional (2D) barcodes

According to [9], 2D barcodes have gained popularity due to: very high identification speed (0.3-1 s), small bit error rate (1/1000, 000), and very low printing cost. In addition, they can be read at any angle of rotation or 60 degrees skewed with a CCD camera, while 1D traditional linear barcode can only be read by scanning from left to right with a laser beam [7]. The 2D barcodes can be read from the back or reverse side of a glass or clear bottle but 1D barcode does not have this capability. Furthermore, 2D barcodes can hold 100 times more data than traditional barcode because of its capacity to increase in size both horizontally and vertically but 1D barcode can only increase its capacity by expanding horizontally. Unlike 1D barcode, 2D barcodes contain special feature called Error Detection And Correction (EDAC) which is a highly sophisticated Reed Solomon algorithms used in decoding software to give redundancy capability to detect and to correct symbol damages where the code can be damage up to 25% and still produce an accurate reading [8]. Finally, 2D barcode can be directly marked on a variety of medias particularly on shinny or metal surface because it uses a CCD camera for reading but 1D barcode is not recommended on those materials because of the reactive nature of a laser beam for reading. The 2D barcodes discussed here are: PDF417, Data Matrix, Maxi Code, and QR Code.

The PDF417 (Portable Data File 417) barcode is a two-dimensional (2D), high-density symbology capable of encoding relatively large amount of text, numbers, files, and actual data bytes [10].

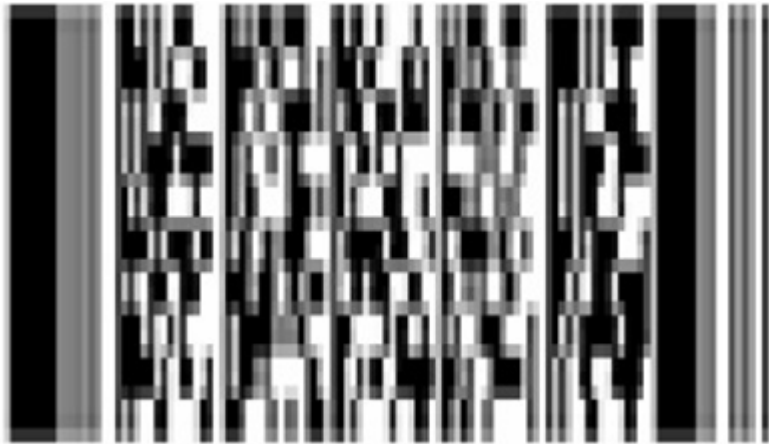


Figure 2.1: PDF417

The printed symbol consists of several linear rows of stacked codeword. Each codeword represents 1 of 929 possible values from one of three different clusters. A different cluster is chosen for each row, repeating after every three rows. Because the codeword in each cluster is unique, the scanner is able to determine what line each cluster is from. It uses Reed Solomon error correction instead of check digits which allows the symbol to endure some damage without causing loss of data [9]. The error correction levels range from 0 to 8 and depend on the amount of data that needs to be encoded, the size, and the amount of symbol damage that could occur. PDF417 has three data compaction modes; byte, text, and numeric. Like other stacked symbologies, it is sensitive to scanner tilt and is not as space efficient as matrix codes and has a maximum storage capacity of only 1Kb.

Data matrix barcodes are two-dimensional barcodes that store information and are often used to identify individual components of a larger item or to help track items between a sender and recipient [9].





Figure 2.2: Data Matrix bar code

It has a variable rectangular size in the form of a matrix. Adjacent **L** frame solid bar encloses one side which is used as a finder for orientation of reading by a detection system. If this frame is damaged, the matrix code cannot be recognized. It is a very compact and reliable code with powerful built in error correction algorithm [8]. The reconstruction of the data content is achievable even after damage to the total code symbol of up to 25%. The codes may be generated and placed on an extraordinarily small space and still be readable, even at only a 20 percent contrast ratio. The code is believed to be the more secure (less hackable) and is favoured where high security is deemed important [8]. The drawbacks of the code are: the maximum storage capacity is only 1.5 Kb, and only readable by an image-based reader which are more expensive. MaxiCode is a variant of the matrix code with a fixed size of 25.4 mm x 25.4 mm.



Figure 2.3: MaxiCode

In the middle there is always a search pattern consisting of three concentric circles, which serves to orient the reading process. Around this search pattern are 866 hexagons, arranged like a honeycomb in 33 rows, which contain the data [10]. Maxicode is a very compact code and reliable due to its powerful built-in error correction algorithm [7]. Reconstruction of the data content after damage to the total code symbol of upto 25%. The fixed physical size of the image simplifies the facilities required for both printing and scanning the symbols. However, there are limitations in that: it has fixed parameters and only readable with image processing systems (Image Reader/2D scanners). In addition, it has a relatively low information content.

QR (Quick Response) code is a 2D barcode/matrix code consisting of black square pattern on a white background created and used primarily for tracking purposes and is popular for mobile applications [7].



Figure 2.4: QR Code

It has a Position Detection Patterns (PDP) characteristic. Compared to most data matrix code, QR code has a relatively high storage capacity for information of 3 Kb. The storage capacity of the QR code is summarized in the table below.

QR Code Data Capacity	
Numeric Only	Max, 7089 characters
Alphanumeric	Max, 4296 caharcters
Binary(8 bits)	Max, 2953 characters

Table 2.1: Storage Capacity of QR code

In addition to better storage capacity of QR code, it can be read anytime and anywhere with mobile phones. No special scanner is required, camera of mobile phones can scan and present the information contained in the codes [8]. The code also allows multiple usages thereby simplifying access without need to type anything. Additionally, when damaged, the QR code can still recover from 30 to 35% of the damaged data, words, or symbols, making the QR code far superior in the capabilities of restoring data, or recovering information which has been lost or damaged for any reason. The reason for this resides in the Error Correction (EC) levels. The EC of all Data Matrix in all versions is around 30% while QR codes have 4 different EC levels. The QR code in the left has only 7% additional data areas for EC while the Data Matrix has 30% additional

data for EC. The additional 23% causes the Data Matrix to be larger than the QR code at the end.

## 2.2 Construction of Security Feature

According to [4], security feature is constructed using two differently emission-decaying inks. The inks are arranged so as to convey a first pattern when they are both exited. A second pattern emerges as the first and more rapidly decaying ink decays, but while a second and relatively longer decaying ink still provides emissions. The second pattern can be alphanumeric characters, a barcode, a digital watermark, and pattern that will yield a predictable frequency domain representation. In one implementation, the first pattern is a first machine-readable code and the second pattern is a second, different machine-readable code. In [6], a system for reliably authenticating a document includes a device having a decryption key therein that, upon application to information provided by a user, reveals not only a plain text message indicating the source of the authentication but, in addition, provides the decryption key for use with the information provided by the mailer.

## 2.3 Object Detection and Recognition

In computer vision, object detection is the process of finding where is the object while object recognition is the process of determining what is the object. All these processes involve feature detection and matching which is the heart of several computer vision algorithms. The algorithms first extract some features from a reference image and compare with the features of the other image in order to determine the similarity measure. To find feasible matches between object features and image features, keypoints

have to be first determined. Keypoints are those pixels whose average intensity values differ much greatly from the immediate neighbours. The keypoints have to be clear, have well defined position in the image space and are stable under local and global perturbations in the image domain so as to be reliably computed with high degree of reproducibility. These tasks still present a challenge to computer vision systems and to that effect, many approaches to the tasks have been implemented over multiple decades [11].

### **2.3.1 Applications of Object Detection and Recognition**

One of the commonest applications is face detection and recognition which is useful in; passport photo verification, image retrieval, and surveillance. It involves identifying the person in the input image from a database of an input face image [12]. The method for acquiring face images depends upon the underlying application for instance, surveillance applications may best be served by capturing face images by means of a video camera while image database investigations may require static intensity images taken by a standard camera. Some other applications, such as access to top security domains, may even necessitate the foregoing of the nonintrusive quality of face recognition by requiring the user to stand in front of a 3D scanner or an infra-red sensor. Face recognition technique which is feature-based first processes the input image to identify and extract (and measure) distinctive facial features such as the eyes, mouth, and nose then compute the geometric relationships among those facial points and then employ standard statistical pattern recognition technique to match faces using the measurements [13].

### 2.3.2 Object Recognition Algorithms

The object recognition techniques/algorithms discussed here are: Eigenfaces, Line Edge Map (LEM), Lapacianfaces, Elastic Bunch Graph Matching (EBGM), Kernel methods, Bayesian, Naive similarity, Scale-invariant feature transform (SIFT), Speeded Up Robust Feature (SURF) and Oriented Fast and Rotated BRIEF (ORB) algorithms.

Eigenfaces algorithm extracts the relevant information of an image and encodes it as efficiently as possible [14]. Mathematically, the algorithm calculates the eigenvectors of the covariance matrix of the set of face images and determines the face space if it is close enough [2]. This algorithm when tried on 30-eigenvectors yielded a recognition rate of 65.12%. For a better recognition rate, this algorithm requires a very large data base [14]. This can overload the database leading to a very high search time.

LEM algorithm uses physiological features from human faces like mouth, nose and eyes to solve the problem. Here the face images are first converted into grey-scale pictures and then encoded into binary edge maps using Sobel edge detection algorithm [14]. According to [15], this algorithm has a low memory requirement due to the kind of data used and is less sensitive to illumination changes. When LEM was tried for different face poses, it yielded a recognition rate of 72.09% [13].

Lapacianfaces algorithm is an appearance -based face recognition algorithm which maps the face images into a face subspace for analysis [16]. It finds an embedding that preserves local information, and obtains a face subspace that best detects the essential face manifold structure. This algorithm when performed on 28 dimensions, its recognition rate was at 89.7% compared to Eigenfaces which yielded 74.7% on 33 dimensions and Fisherfaces which yielded 80% on 14 dimensions [17].

EBGM algorithm considers that all human faces share a similar topological structure [18]. Faces are represented as graphs, with nodes positioned at fiducial points (for

example tip of noses, corners of mouth) and edges labeled with two dimensional (2D) distance vectors. Each node contains a set of 40 complex Gabor wavelet coefficients at different scales and orientations (phase, amplitude) which are called “jets”. Recognition is based on labeled graphs. A labeled graph is a set of nodes connected by edges, nodes are labeled with jets while edges are labeled with distances. This algorithm extracts model graphs from the gallery images and image graphs from the probe images then compares the two graphs and picks the one with the highest similarity value. When EBGM was experimented on Bochum database of 108 neutral frontal views as a model gallery and the other images as probe galleries, an average recognition rate of 96.0% was obtained [18][19].

Kernel methods take into account the representations of subspace based on higher order statistics and Independent Component Analysis (ICA) which maximizes the degree of statistical independence of output variables using contrast functions like Kullback-Leibler divergence, Negentropy, and Cumulants [20][21]. When tested on Yale data base of 400 images of 40 subjects, a recognition rate of 93.75% was obtained.

Bayesian Face Recognition algorithm is a direct visual image matching algorithm which uses a probabilistic measure of similarity based primarily on a Bayesian analysis of image differences [22] . Other than relying on fiducial points like the eye/nose/mouth corners which yield low recognition rates, this algorithm considers visual representations making use of the appearance or texture of facial images often as raw 2D inputs. When tested on a collection of images from the ARPA FERET face data base consisting of 74 pairs of gallery images and 38 pairs of probe images, it yielded a recognition rate of 89.5%.

Naive similarity algorithm selects the two images (the reference image and the image to be compared), normalizes them to the same size (300x300), extracts the features, calculates the distances between features of the two images and then finally shows these

distances in order of less distant to most distant [23]. Similar features are less distant. However this algorithm does not perform well under different circumstances.

Despite all the above efficient recognition rates yielded by different face recognition techniques in document verification, face recognition as a way of document verification still presents a challenging task to researchers in the field of image analysis and computer vision in that human faces are not an invariant characteristics, it changes greatly in short intervals. Also different face may look alike and thus a discrimination task is required. The analysis of the same face is always challenged by the changes in illumination and variability in facial expressions like presence of accessories (glasses, beard) [6]. Also, the rotation of the face may change many facial characteristics [13]. However SIFT, SURF and ORB are invariant to location, scale and rotation.

According to [11], SIFT has the ability to find distinctive keypoints that are invariant to location, scale and rotation, and robust to affine transformations (changes in scale, rotation, shear, and position) and changes in illumination. SIFT first extracts the keypoints from a set of reference image and store it in a database. An object is then recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vector. To increase robustness, matches are rejected for those keypoints for which the ratio of the nearest neighbour distance to the second nearest neighbour distance is greater than 0.8 [24]. This discards many of the false matches arising from background clutter. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. Each cluster of 3 or more features that agree on an object and its pose is then subjected to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable



false matches [25]. Object matches that pass all these tests can be identified as correct with high confidence. The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbour to the distance of the second closest. To obtain the keypoints, the image is convolved with Gaussian filters at different scales, and then the differences of successive Gaussian-blurred images are taken [24]. In [11], Keypoints are then taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. This is done by comparing each pixel in the DoG images to its eight neighbours at the same scale and nine corresponding neighbouring pixels in each of the neighbouring scales. If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint. The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures which is achieved through interpolation of the nearby data for accurate position. According to [24] interpolation is done using the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function,  $D(x,y,\sigma)$  with the candidate keypoint as the origin. This Taylor expansion is given by:  $D(X) = D + \frac{\partial D^T}{\partial x} X + \frac{1}{2} X^T \frac{\partial^2 D}{\partial x^2} X$  where  $D$  and its derivatives are evaluated at the candidate keypoint and  $X=(x,y,\sigma)$  is the offset from this point. Points that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge are rejected. These points give a value of the second-order Taylor expansion less than 0.03. However in terms of speed, SURF has proved to perform better than SIFT [25].

Object recognition using SURF is scale and rotation invariant which makes it very powerful [26]. The implementation is done using openCV library which loads two images, finds the SURF keypoints and descriptors, compares them and finds a matching if there is any. The algorithm extracts some keypoints and descriptors from an image and later uses it to detect the same image. SURF descriptor describes how pixels intensities are distributed within a scale dependent neighbourhood of each interest point. By using

the descriptor, a relation between the reliable key points and the neighbouring pixels are obtained. Once the descriptors and the key points of the two images are calculated, then they can be compared using the nearest neighbour algorithm. SURF uses an intermediate image representation called integral image, which is computed from the input image and is used to speed up the calculations in any rectangular area (SURF relies on integral images for image convolutions to reduce computation time). SIFT builds an image pyramids by filtering each layer with Gaussians of increasing sigma values and taking the difference while SURF creates a “stack” [25]. According to [27], SURF tackles the problem of point and line segment correspondences between two images of the same scene or object. First, keypoints are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. Next, the neighborhood of every keypoint is represented by a feature vector. This descriptor has to be distinctive. At the same time, it should be robust to noise, detection errors, and geometric and photometric deformations. Finally, the descriptor vectors are matched among the different images.

Despite being invariant to scale and rotation, SIFT and SURF algorithms are computationally very intensive, unsuitable for low power devices, too slow for use in real-time for low memory devices like smart phones and patented [26]. To make descriptors faster to compute, more compact while remaining robust to scale, rotation and noise, ORB algorithm which is an efficient alternative to SIFT and SURF was developed [26]. Additionally, ORB is free from the licensing restrictions of SIFT and SURF. According to [26][28], ORB is built on FAST keypoint detector and BRIEF descriptor. Unlike ORB, many keypoint detectors (SIFT and SURF inclusive) include an orientation operator which is either computationally demanding, or in the case of SURF, yields poor approximations.

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it uses FAST to find keypoints, then

applies Harris corner measure to find top  $N$  points among them. It also uses pyramid to produce multiscale-features. But one problem is that, FAST doesn't compute the orientation. [26] made modification such that ORB compute the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. To improve the rotation invariance, moments are computed with  $x$  and  $y$  which should be in a circular region of radius  $r$ , where  $r$  is the size of the patch. For descriptors, ORB "steers" BRIEF according to the orientation of keypoints. For any feature set of  $n$  binary tests at location  $(x_1, y_1)$ , it defines a  $2 \times n$  matrix,  $S$  which contains the coordinates of these pixels. Then using the orientation of patch,  $\theta$ , it finds the rotation matrix and rotates  $S$  to get steered (rotated) version  $S_\theta$ . ORB discretizes the angle to increments of  $\frac{2\pi}{30}$  (12degrees), and constructs a lookup table of precomputed BRIEF patterns. As long as the keypoint orientation,  $\theta$  is consistent across views, the correct set of points  $S_\theta$  will be used to compute its descriptor. Matching of features is achieved by using Brute-Force matcher. It takes the descriptor of one feature in first set and matches with all other features in second set using Hamming distance calculation and returns the closest [28].

Conclusively, previous research from scholars has shown that with advanced scanning and printing technologies, paper-based document fraud can easily be conducted without significant high cost. Therefore there is still a great challenge of inaccurate recognition and verification of documents. Basing on the fact that no work has yet been done on integrating image retrieval and face recognition algorithms using mobile phones as a way of verifying Makerere University student's examination permit, this study is therefore mainly focused on integrating image retrieval and face recognition algorithms using mobile phones to verify the student's examination permit. The study will encode student's number using Quick Response (QR) code (for confidentiality) and integrate the QR code application with image retrieval and face recognition algorithms.

# Chapter 3

## Methodology

This chapter illustrates steps taken to verify student's examination permit. The **Fig 3.1** below shows the class diagram of the computational steps.

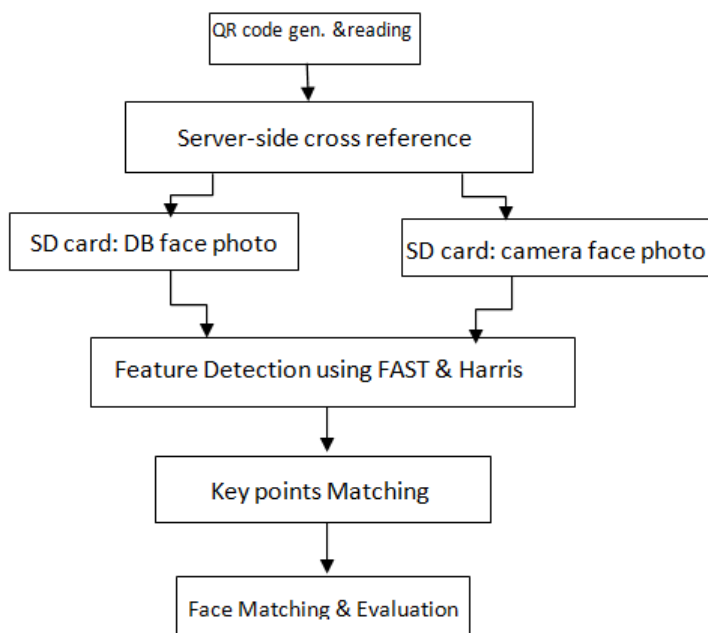


Figure 3.1: Verification process

- *QR code generation and reading:* The Quick Response (QR) code was generated with encoded student's number on it. The application scanned and decoded the

number from the QR code.

- *Server-side cross reference:* This connected the application to the database. Using the decoded student's number, the application queried the database and retrieved the student's registration details.
- *SD card:* This stored the retrieved student's face from the database and the captured student's face by the application.
- *Feature Detection using FAST and Harris:* Pixels of the key features for the two faces were detected and distinct ones extracted as two sets of key points.
- *Key points Matching:* Key points with close distance (similar key points) were matched.
- *Face Matching and Evaluation:* The number of matches were counted. If the matches were atleast 50, then the two faces matched and hence similar, otherwise not.

In order to achieve the above steps, the following tools and libraries were used:

- Java programming language
- Apache-Tomcat-7.0.41 , Grails 2.1.3 and
- NetBeans IDE, IntelliJ IDEA 12.1.6 and Eclipse IDE.
- Android phone
- com.google.zxing.\*, and java.io.\*,org.opencv.features2d.\*, and org.opencv.core.\*

## 3.1 Encoding student's number on QR code

The study used a standalone application to generate and print Quick Response (QR) code with encoded student's number. The study considered QR code pixel size  $25 \times 25$  pixels, symbol version 2 with  $M$  level of error correction. This level has a recovery capability for data damaged upto 15% [9]. For the QR image to be effectively scanned and read by the application, it should have a minimum size determined by scanning distance and size of the data dots in the QR code. This size was calculated from the formula:

$$\text{MinimumQRcodeSize} = \left( \frac{\text{ScanningDistance}}{\text{DistanceFactor}} \right) * \text{DataDensityFactor}, \text{ where } \text{DataDensityFactor} = \frac{\text{NumberOfColumnsOfDots}}{21}$$

Using the ratio  $\text{ScanDistance} : \text{MinimumQRcodeSize} = 10 : 1$ .  $\text{MinimumQRcodeSize}$  was set to  $2.5\text{cm}(1\text{inch})$  for effective  $\text{ScanDistance}$  of about  $250\text{mm}(10\text{inches})$ [8]

### 3.1.1 Format information

The generated QR symbol contained 26 bytes of information, some used to store the student's number and some for error correction. The first two bits of format information give the error correction, the next three bits select the masking pattern to be used in the data area, and the remaining ten bits are for correcting errors in the format itself.

During the encoding process, the following steps were followed:

- The student's number was encoded according to version 2  $M$  level mode to form bit stream.
- The bit streams were then divided into codewords and codewords further divided into blocks.

- Error correction codewords were then added to each block.
- The codewords were then put into a matrix and masked with mask pattern using *exclusive -or (XOR)* operation; for  $i$  rows and  $j$  columns:
 
$$\text{Mask} : 000 \Rightarrow (i + j)\%2 = 0$$

$$\text{Mask} : 011 \Rightarrow (i + j)\%3 = 0$$

$$\text{Mask} : 111 \Rightarrow ((i \star j)\%3 + i + j)\%2 = 0$$
- Finally the function patterns were added into the QR symbol.

During the QR code reading, the application performed the following steps:

- Binarization of gray scale images of the QR code by thresholding.
- The approximate region of the QR code was obtained, and coarse positioning for QR image according to the finder patterns was implemented.
- Accurate positioning according to the alignment patterns was implemented

For alignment patterns:  $P_1, P_2, \dots, P_n$  each with central coordinate  $(x_i, y_j)$

$$L_x = \text{Horiz} - \text{centre} - \text{to} - \text{centre} - \text{distance} - \text{of} - P_i, P_j, i \neq j$$

$$L_y = \text{Vert} - \text{centre} - \text{to} - \text{centre} - \text{distance} - \text{of} - P_i, P_j, i \neq j$$

$$d = \text{distance} - \text{between} - P_i, P_j, i \neq j$$

$$\therefore \text{modulePitches}, CP_x = \frac{L_x}{d}, \text{ and } CP_y = \frac{L_y}{d}$$

- The angle of inclination to rotate QR image was calculated and rectification processing implemented.
- Version number was obtained and self-adaptive sampling implemented as:

Let  $D$  be the distance between the centers of the upper left position detector pattern and of the upper right position detection pattern

$W_{UL} = \text{widthOfUpperLeftPattern}$

$W_{UR} = \text{widthOfUpperRightPattern}$ .

nominal dimension of the symbol,  $X = \frac{W_{UL}+W_{UR}}{14}$

module size ,  $CP_{UR} = \frac{W_{UL}}{7}$

version of the symbol,  $V = \frac{(D/X)-10}{4}$

- Finally the QR code based on corrected image was decoded.

On reading the QR code, the application through the server, queried the database using the decoded student's number and displayed student's details.

## 3.2 Server side cross reference

A database for performing basic Create, Read, Update, and Delete (CRUD) operations was developed using MySQL database management system and Grails Object Relational Mapping (GORM) for the persistence of the domain model. The following models/domains were created: Student, Course, CourseUnit, SemesterRegistration and FinancialStatement. As a way of securing the application, a mechanism for authenticating and authorising users was created using grails Spring Security. The User, Role and UserRole domains were generated for handling the application security.

## 3.3 Face Recognition

The study has used Oriented Fast and Rotated BRIEF (ORB) algorithm for finding, describing and matching features of the student's database face photo with the camera



face photo.

### 3.3.1 ORB algorithm

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor. It first uses FAST to find keypoints and then apply Harris corner measure to find top  $n$  points among them [29].

#### 3.3.1.1 Feature detection using FAST

The algorithm detected keypoints by finding keypoint pixels. It assumed  $P$  to be the candidate pixel (pixel at the corner). The algorithm then set Intensity and threshold as:  $Intensity = I_p, threshold = t$ . For pixels:  $P_1, P_2, \dots, P_n$  equidistant from  $P$ , if there exist pixels:  $P_1, P_2, \dots, P_k, k < n$  which are all brighter than  $I_p + t$ , or all darker than  $I_p - t$ , then  $P$  is a corner, else not a corner.

#### 3.3.1.2 Harris corner detector

This algorithm determines  $n$  distinct keypoints from the set of keypoints detected by FAST. It computed the shift in intensity for every pair of keypoints of the same image. It considered keypoints  $k_1, k_2, \dots, k_n$  located at  $(x_i, y_j)$  in an intensity image  $I$ . The change of intensity for the shift  $(u, v)$  is  $E(u, v) = \sum w(x, y)[I(x + u, y + v) - I(x, y)]^2, \forall x, y$ ; where  $w(x, y)$  is a Gaussian function. The shifted intensity,  $S = [I(x + u, y + v) - I(x, y)]^2$ .  $S = 0$  for indistinct keypoints and  $S \gg 0$  for very distinct keypoints.

### 3.3.2 Keypoints matching

For two sets of image keypoints :  $p = p_1, p_2, \dots, p_n$  and  $q = q_1, q_2, \dots, q_n$ , the Euclidean distances between every pair of keypoints were calculated from the formula  $d(p, q) = \sqrt{(p_i - q_i)^2 + (p_j - q_j)^2}$ . The minimum distance between the keypoints was calculated,  $d_{min} = \min [d(p, q)]$ . For every pair of keypoints  $(p_k, q_k)$ , if  $d(p, q) \leq 2 * d_{min}$ , then  $p_k$  and  $q_k$  are considered to be a match, otherwise there is no match.

### 3.3.3 Face matching & evaluation

For two faces with keypoints  $(p_k, q_k)$ , such that  $d(p_k, q_k) \leq 2 * d_{min}, k = 1, 2, \dots, n$ , the algorithm drew  $m$  matches. If  $m \geq 50$ , then the faces were considered to be similar.

## 3.4 System Implementation and Evaluation

### 3.4.1 System implementation

The system implementation was achieved by integrating the Android phone with the database into a networked system.

### 3.4.2 Testing and validation

Face recognition accuracy and verification rate against the internet signal strength (how long the system took to retrieve the student's data) of the application were used to evaluate the system.

### 3.4.2.1 Recognition accuracy

Recognition accuracy was calculated using  $Accuracy = \frac{No.OfTrueOutcomes}{No.OfTestCases}$

The study took count of the true outcomes using 12 pairs of test faces (human-to-human face) and 6 sets of other objects (human-to-other object). The recognition accuracies for human-to-human face and human-to-other object were computed and recorded in **table 4.1**

### 3.4.2.2 Verification rate

The study took three cases of internet signal strengths: Low, Good and Excellent. The time taken for the application to retrieve the student's details from the database were recorded against the internet signal strengths in **table 4.2**

# Chapter 4

## System Discussion and Analysis of Results

This section elaborates system demonstration and the results attained including the challenges faced in the implementations

### 4.1 User Authentication

The application was configured to authenticate users with user name *admin* and password *admin* as shown in fig 4.1. There is an option also for a new user to signup.

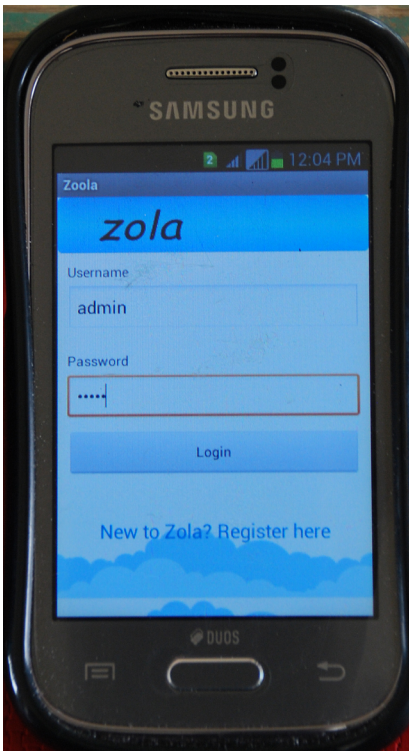


Figure 4.1: Authentication Screen

On successful login, the application opened the option screen. Here there are three options; search, scan and download as shown in **fig 4.2**

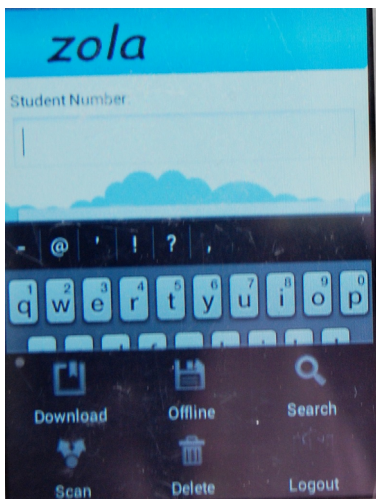


Figure 4.2: Option Screen

## 4.2 Scan Option

This allowed the user to scan the Quick Response (QR) code from the student's examination permit. The QR code image on the demo examination permit appeared as shown in **fig 4.3** below



Figure 4.3: QR Code

On focusing the QR code, the application scanned the code, decoded student's number from it and handed it to the search student function. The scanning process is shown in **fig 4.4**

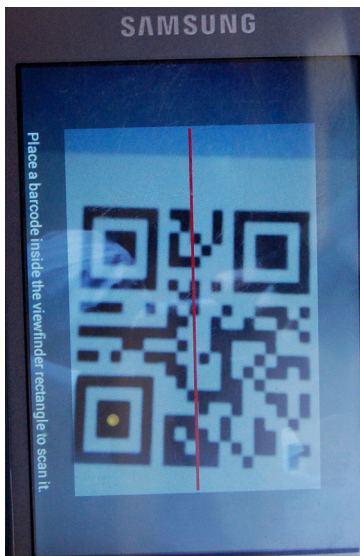


Figure 4.4: QR code Scanning

The search student function through the server, retrieved and displayed student's registration details as in **fig 4.5**

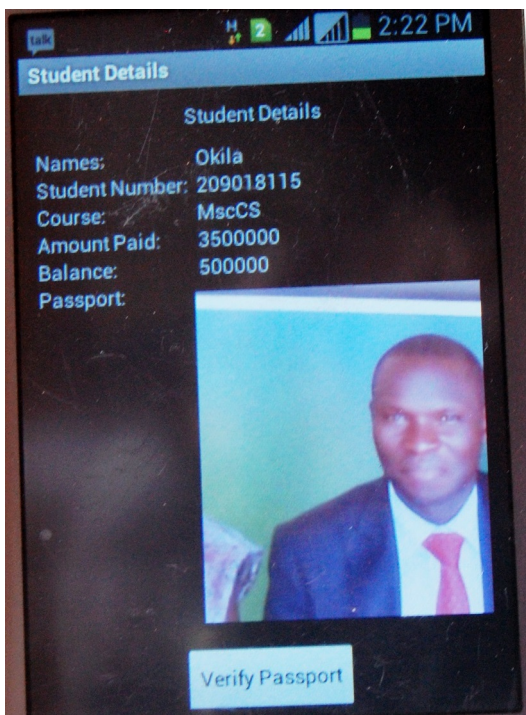


Figure 4.5: Student registration details

The details displayed are: Names, Student number, Course, Amount Paid, Balance and Passport. From this, the User can confirm if the student has fully paid all the tuition and therefore qualified to sit for an examination other than relying on the printout of the examination permit of which its authenticity can be questionable.

### 4.3 Face recognition

To rule out impersonation, the user tapped *Verify Passport* button on the photo and then focused the student's face. On clearly focusing the student's face, the user touched the phone's screen to capture the face. The application captured the student's face and matched it with the database face. If the number of good matches were at least 50, the application recognised the faces to be similar otherwise not similar.

In **fig 4.6** below, the application found 89 good matches and recognised the two faces as similar.

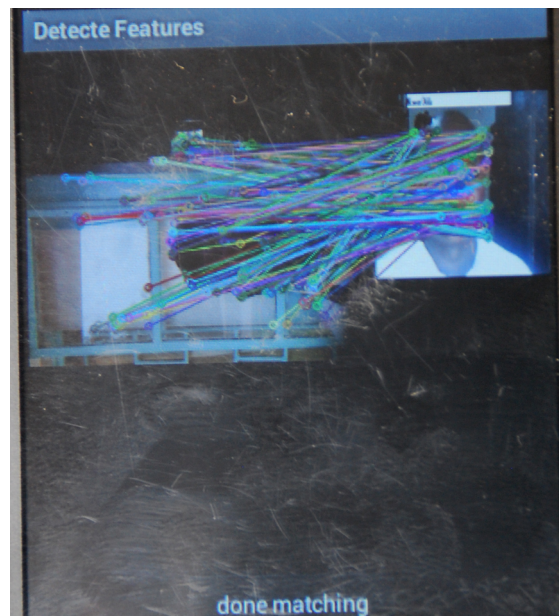


Figure 4.6: Photo recognition



In **fig 4.7** below, the application found 42 good matches and recognised the two faces as not similar

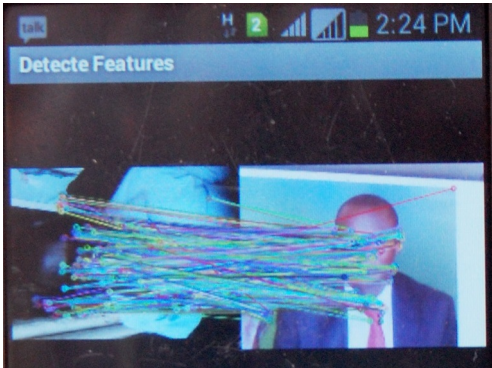


Figure 4.7: Recognition of dissimilar faces

### 4.3.1 Face recognition accuracy

With the threshold of number good matches set to  $\geq 50$ , the application gave the following recognition accuracies.

Object Class	Number of test cases	No.of true outcome	No. of false outcome	Acuracy (%)
Human face-to-Human face	12	8	4	66.7
Human face-to-other objects	6	5	1	83.3

Table 4.1: Recognition Accuracy

The recognition accuracy was very high when human face was compared with other objects because the contours on human faces are quite distinct from that of other objects but the accuracy was lower when human faces were compared because human faces have similar contours.

### 4.3.2 Potential problems with face recognition.

The false recognitions where similar faces were recognised as not similar and dissimilar faces recognised as similar were due to the potential problems:

- **Appropriate threshold value for number of good matches.** With low threshold value ( $< 50$ ), other objects with curved contours were recognised as similar to the retrieved student database face photo. And all test human faces were recognised as similar to the student database face photo. On setting the threshold value to  $\geq 50$ , most other objects could be recognised as not similar to human face but some human faces were recognised as similar to the retrieved student's database face photo. And also some similar faces were recognised as not similar. This is because human faces are so similar that choosing a threshold value for the number of good matches to be used for human face recognition still poses a great challenge.
- **Strength of the lens camera.** If the lens is not strong enough, the captured student image may be blurry and the application may not easily identify the key features. This can compromise the recognition accuracy.
- **Light source.** If the student's photo is taken from a low light intensity environment, the application may not match all the keypoints as some may not be clearly visible.
- **Background.** If the student's background has many distracting details, then the captured image will have a lot of noise in the background. These noise will also be matched with the database photo and therefore compromising the recognition accuracy.

- **Dimension of the database photo.** Improper dimension of the database photo reduces the recognition accuracy with the captured photo.

### 4.3.3 Verification rate

The time taken by the application to retrieve the student's face from the database was recorded against different signal strengths.

Signal Strength	Duration (sec)
Low	6
Good	3
Excellent	2

Table 4.2: Verification speed

## 4.4 Download option

This option downloaded a list of student's details and stored it in the SQLite database when the application was online. The list was retrieved during offline mode. On logging in, the user selected the download option as in **fig 4.8**.

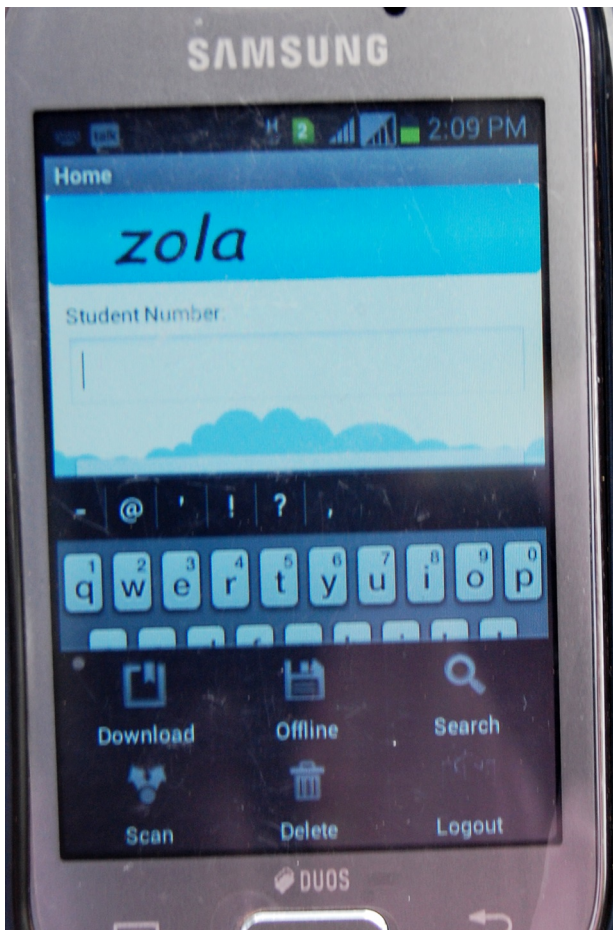


Figure 4.8: Download menu

On tapping the download button, the user entered course code and tapped the Download as shown in **fig 4.9**

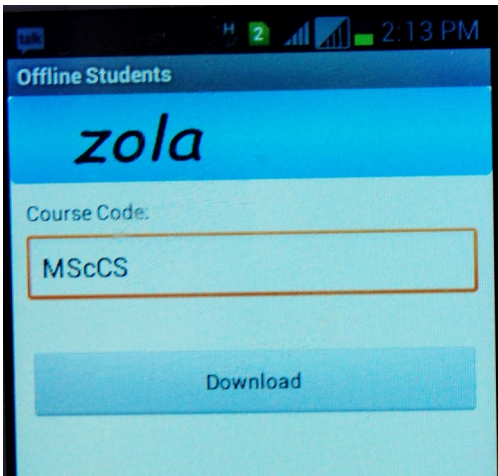


Figure 4.9: Download

A list of student registered for the course was displayed as in **fig 4.10**

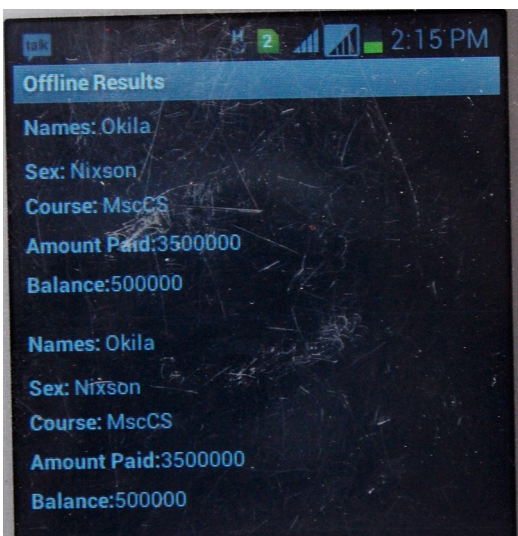


Figure 4.10: Offline Results

## 4.5 Server and Database

This module was developed using grails framework and MySQL database management system. The database used Grails Object Relational Mapping (GROM) and had the

following entities:

Table Name	Table Fields	Field Description	Data Type
Student	id	Unique Student Id.	VARCHAR
	std_num	Student Number	VARCHAR
	name	Student Names	VARCHAR
	sex	Student Sex	VARCHAR
	passport	Student Photot	byte[]
Course	id	Unique Course Id.	VARCHAR
	name	Course Name	VARCHAR
	code	Course Code	VARCHAR
	tuition	Tuition fees for the course	INT
CourseUnit	id	Unique Course Unit Id	VARCHAR
	name	Course Unit Name	VARCHAR
	code	Course Unit Code	VARCHAR
SemesterRegistration	id	Unique Sem.Registration Id	VARCHAR
	name	Sem. Registration Name	VARCHAR
FinancialStatement	id	Unique Fin.Statement Id.	VARCHAR
	amount_paid	Amount paid for course	INT
	balace	Balance Due	INT

Table 4.3: Database Design

#### 4.5.1 Web Portal Login Screen

This loaded a default Login screen which required the user to enter *admin* as username and *admin* as password as in **fig 4.11**

**Please Login**

Username:

Password:

Remember me

Figure 4.11: Login Screen

On successful login, this portal linked to home page screen

## 4.5.2 Home Page Screen

This page showed the modules/domains as in fig 4.12

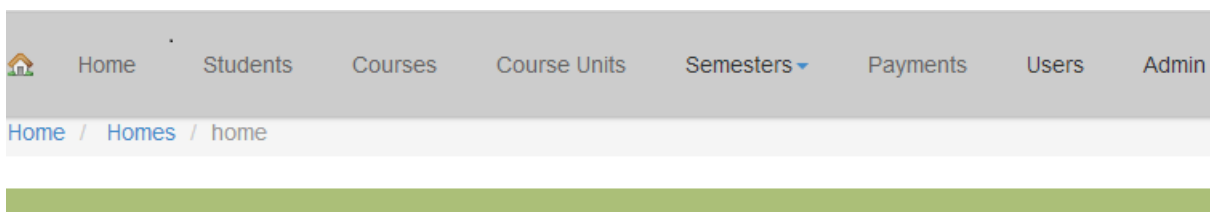


Figure 4.12: Homepage Screen

## 4.5.3 Students

This web interface had two links to the database:

### 4.5.3.1 Create New Student

For capturing and storing student's informations into the database as in fig 4.13

Home / students / create

Home Student List

## Create Student

**Passport \***

Choose File No file chosen

**Names** OKila Nixson

**Sex** Male

**Course \*** MSc. in Computer Science ▼

**Student Number** 209018115

Create

Figure 4.13: Create New Student

### 4.5.3.2 List Students

For viewing a list of registered students from the database as in fig 4.14






Student Number	Student Names	Sex	Passport	Course
205017449	Amwiine Emmanuel	Male		MSc. in Data Communications and Software Engineering
209018115	Okila Nixson	Male		MSc. in Computer Science
209018116	Mutebe Alex	Male		MSc. in Data Communications and Software Engineering

Figure 4.14: List Students Screen

## 4.5.4 Courses

This had two links to the database:

### 4.5.4.1 Create New Course

For capturing and storing course details into the database as in **fig 4.15**

Home Course List

## Create Course

Name

Code

Course Units [Add CourseUnit](#)

Students [Add Student](#)

Tuition \*

Create

Figure 4.15: Create New Course Screen

#### 4.5.4.2 List Courses

For viewing a list of courses from the database as in fig 4.16

New Course

## Course List

Name	Code	Tuition
MSc. in Data Communications and Software Engineering	MscDCSE	3,500,000
MSc. in Computer Science	MCSC	3,500,000

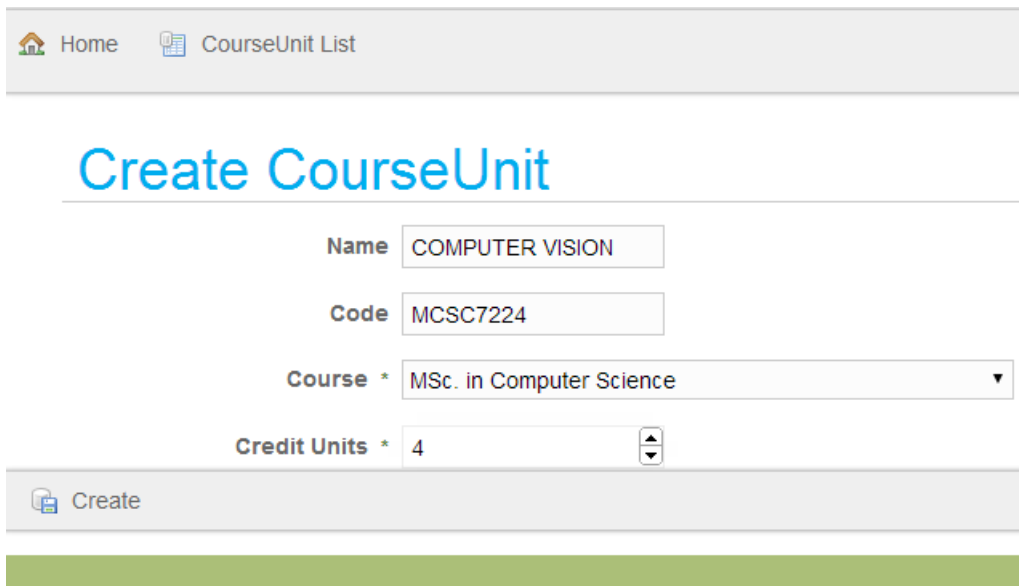
Figure 4.16: List Courses Screen

## 4.5.5 Course Units

This has two links to the database:

### 4.5.5.1 Create Course Units

For capturing and storing course unit details into the database as in fig 4.17



The screenshot shows a web application interface for creating a course unit. At the top, there is a navigation bar with a home icon and the text 'Home', and a document icon with the text 'CourseUnit List'. Below this is a large blue heading 'Create CourseUnit'. The form contains four input fields: 'Name' with the value 'COMPUTER VISION', 'Code' with the value 'MCSC7224', 'Course \*' with a dropdown menu showing 'MSc. in Computer Science', and 'Credit Units \*' with a spinner box set to '4'. At the bottom of the form is a 'Create' button with a document icon. The entire form is set against a light gray background with a green bar at the very bottom.

Figure 4.17: Create Course Units Screen

### 4.5.5.2 List Course Units

For viewing a list of course units from the database as in fig 4.18

New CourseUnit

## CourseUnit List

Name	Code	Course	Credit Units
COMPUTER VISION	MCS 8200	MSc. in Data Communications and Software Engineering	4
ADVANCED PROGRAMMING	MCS7118	MSc. in Computer Science	4
COMPUTER VISION	MCSC7224	MSc. in Computer Science	4
MASTER'S PROJECT I IN COMPUTER SCIENCE	MCSC8200	MSc. in Computer Science	4

Figure 4.18: List Course Units screen

## 4.5.6 Semesters

This had two links to the database:

### 4.5.6.1 Add a Semester

For capturing and storing semester details into the database as in **fig 4.19**

Home / [semesterRegistrations](#) / index

New SemesterRegistration

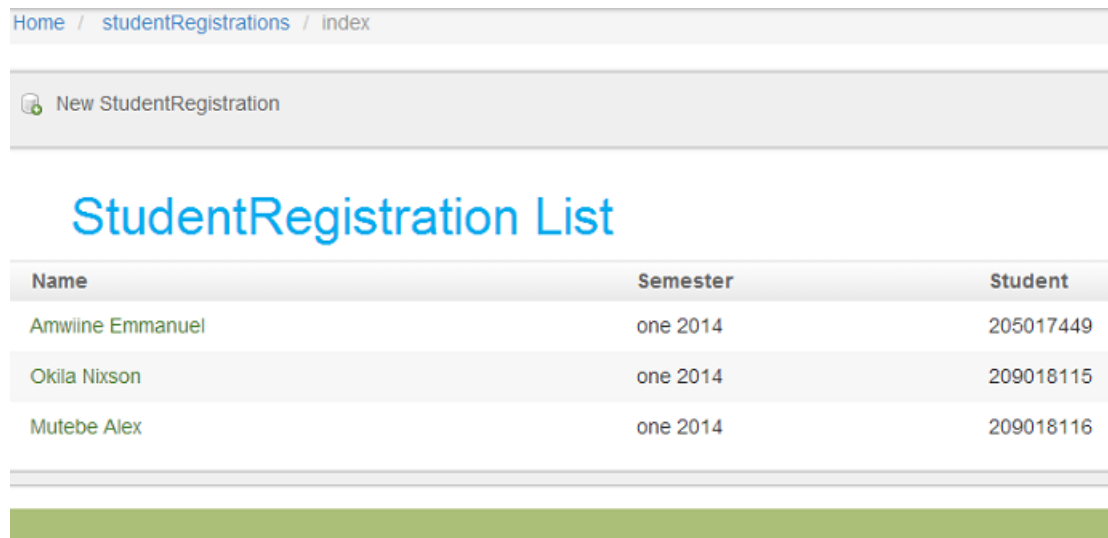
## SemesterRegistration List

Year	Name
2014	one
2014	two

Figure 4.19: Add Semester Screen

#### 4.5.6.2 Register Student

This registered a student for a particular semester as in **fig 4.20**



The screenshot shows a web application interface. At the top, there is a breadcrumb trail: Home / studentRegistrations / index. Below this is a header area with a green icon and the text 'New StudentRegistration'. The main content area features a large blue heading 'StudentRegistration List'. Underneath the heading is a table with three columns: 'Name', 'Semester', and 'Student'. The table contains three rows of data. At the bottom of the page, there is a solid green horizontal bar.

Name	Semester	Student
Amwiine Emmanuel	one 2014	205017449
Okila Nixon	one 2014	209018115
Mutebe Alex	one 2014	209018116

Figure 4.20: Register Student Screen

#### 4.5.7 Payment

This had two links to the database:

##### 4.5.7.1 Create Financial Statement

This created financial statement for each student and stored into the database as in **fig 4.21**

Home / financialStatements / create

Home FinancialStatement List

## Create FinancialStatement

Amount Paid \*

Balance \*

Semester Registration \*

Student \*

Create

Figure 4.21: Create Financial Statement Screen

#### 4.5.7.2 List Financial Statement

For viewing a list of financial statements from the database as shown in fig 4.22

Home / financialStatements / index

New FinancialStatement

## FinancialStatement List

Amount Paid	Balance	Semester Registration	Student
3,500,000	500,000	one 2014	Amwiine Emmanuel
3,500,000	0	one 2014	Okila Nixson
2,500,000	1,000,000	one 2014	Mutebe Alex

Figure 4.22: List Financial Statement Screen

# Chapter 5

## Conclusion & Further Work

### 5.1 Conclusion

Though face recognition has been one of the challenging tasks in computer vision field, as a way of document verification this study has presented *ORB* algorithm for student's face recognition coupled with deployment of the application on a smart (Android) phone. The study used Quick Response (QR) code to encode student's number for confidentiality purpose. On scanning the QR code, the student's face photo was retrieved from the database and the algorithm compared this database face photo with the the captured face photo by extracting the keypoints and matching them. Some recognitions were false due to environmental factors, hardware (camera) capabilities, and facial appearances. A comparative study of recognition accuracy for 12 sets of test faces was done. With the threshold value for number of good matches  $\geq 50$ , the study found a recognition accuracy of 66.7% . The accuracy was greatly affected by the similarity in human faces, environmental factors (background objects and light intensity), and strength of lens camera.

## 5.2 Future Work

Future studies for improvement of this application can be done on the recognition accuracy of human faces. Major focus should be on:

- Appropriate threshold value for the number of good matches to achieve a reliable recognition accuracy.
- How to deal with changes in light intensity and background of face photo.



# Bibliography

- [1] The Observer, “Makerere students forge permits to sit exams,” Weekly Observer, Tech. Rep., 28 Sept.2012.
- [2] M. A. Turk and A. P. Pentland , “Face recognition using eigenfaces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586-591, 1991.
- [3] F. William and Behm et al, “Document verification system,” in *Document verification*, 1995.
- [4] L. Robert Jones,M. Alasatir Reed, “Emerging security features for identification documents,” US 2003-0173406, Tech. Rep., 2007.
- [5] E Keneth Taylor, B Jeff Glickman, “Apparatus and method for matching image characteristics such as fingerprint minutiae,” US 4896363 A, Tech. Rep., 1990.
- [6] John Smith, “Reliable document authentication system,” *Public Key Cryptography*, pp. 189–218, 1983.
- [7] Maykin Warasart and Pramote Kuacharoen, “Paper-based document authentication using digital signature and QR code,” in *4Th Internatinal Conference on Computer Engineering and Technology*, 2012.

- [8] J. Z Gao, “Understanding 2-D barcode technology and applications in m-commerce design and implementation of a 2-D barcode processing solution,” in *Proceedings of international conference on computer software and application*, 2007.
- [9] Nicole Pontius, “Data matrix codes vs QR codes,” in *Data Matrix Codes vs QR codes*, 2012.
- [10] “Public key cryptography for financial services industry, the elliptic curve digital signature algorithm(ecdsa), 2005.”
- [11] Lowe and G. David , “Object recognition from local scale-invariant features,” *Proceedings of the International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.
- [12] Rabia Jafri and R. Hamid Arabnia, “A survey of face recognition techniques,” in *Information Processing System*, 5(2) 2009.
- [13] O. De Vel and Aeberhard, “Line-based face recognition under varying pose,” *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 21, pp. 1081–1088, 1999.
- [14] Jorge Orts, “Face recognition technique,” ECE533-Image processing project, Tech. Rep., 1996.
- [15] Youngsheng Gao and M. K. H. Leung , “Face recognition using line edge map,” *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 24, pp. 764–779, 2002.
- [16] Xiaofei He , Shuicheng Yan , Yuxiao Hu , Partha Niyogi and Hong-Jiang Zhang, “Face recognition using lapcaianfaces,” *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 27(3), 2005.

- [17] P.N. Belhumeur , J.P. Hespanha and D.J. Kriegman, “Recognition using class specific linear projection,” *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 19(7), pp. 711–720, 1997.
- [18] L. Wiskot , Fellous J. M, Krueger N and Von der Malsburg C, “Face recognition by elastic bunch graph matching,” *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 19(7), pp. 775–779, 1997.
- [19] H. I .Hahn and J.K .Joung , “Implementation of algorithm to decode two-dimensional barcode pdf-417,” in *International conference on signal processing*, 2002.
- [20] M.H Yang, “Face recognition using kernel methods,” *Proceedings of the IEEE on Automatic Face and Gesture Recognition*, pp. 215–220, 2002.
- [21] A. Hyvarienan , J. Karhunen and E. Oja , “Independent component analysis,” in *Wiley- Interscience*, 2001.
- [22] B. Moghaddam , T. Jebara and A. Pentland , “Bayesian face recognition,” *Pattern recognition*, vol. 33(11), pp. 1771–1782, 2000.
- [23] Rafael Santos, “Java image processing cookbook.”
- [24] J. Beis and D.G. Lowe , “Shape indexing using approximate nearest-neighbour search in high-dimensional spaces,” *Computer Vision and Pattern Recognition*, pp. 1000–1006, 1997.
- [25] L. Juan and O. Gwun , “A comparison of SIFT, PCA-SIFT and SURF,” *International Journal of Image Processing (IJIP)*, vol. 3(4), pp. 143–152, 2009.
- [26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski., “ORB:anefficient alternative to SIFT or SURF. 2011.”

- [27] H. Bay , “From wide-baseline point and line correspondences to 3D,” Ph.D. dissertation, Swiss Federal Institute of Technology, Switzerland, 2006.
- [28] M. Calonder, V. Lepetit, C. Strecha, and P. Fua., “Brief: Binary robust independent elementary features,” in *European Conference on Computer Vision*, 2010.
- [29] Alexander Mordvintsev and K. Abid , “Opencv-python tutorials documentation,” Tech. Rep., 2014.
- [30] Edward Rosten and Tom Drummond, “Features from accelerated segment test,” in *Machine learning for high-speed corner detection*, 2006.

# Appendix

## Code

### C.1 QR code generation

```
public void generateQrCode(String text) {  
  
    BitMatrix bitMatrix = new QRCodeWriter().encode(text, BarcodeFormat.QR_CODE,  
    width, height);  
  
    MatrixToImageWriter.writeToStream(bitMatrix, imageFormat, new FileOutputStream(new  
    File(""));
```

### C.2 User authentication

```
public class LoginActivity extends Activity {  
  
    private EditText usernameTxt;  
  
    private EditText passwordTxt;  
  
    public void onClick(View view) {  
  
        String url = ZoolaNetworkBridge.URL+"login";  
  
        String username = usernameTxt.getText().toString();
```

```
String password = passwordTxt.getText().toString();
```

### **C.3 Display of student's registration details**

```
public class StudentDetailsActivity extends Activity {  
  
    public static Bitmap passport;  
  
    LinkedHashMap<String, String> map = convertToMap(data2);  
  
    String names = map.get("names");  
  
    byte []img = Base64.decode(map.get("photo"),Base64.DEFAULT);  
  
    Bitmap bmp = BitmapFactory.decodeByteArray(img, 0, img.length);  
  
    imgPreview.setImageBitmap(bmp);  
  
    passport = bmp;
```

### **C.4 Face recognition**

#### **C.4.1 Setting application camera ready for capturing and storing face photo.**

```
public class MyCameraView extends JavaCameraView implements PictureCallback {  
  
    private static final String IMAGE_DIRECTORY_NAME = "zoola-img";  
  
    private Uri fileUri;  
  
    public static final int MEDIA_TYPE_IMAGE = 1;  
  
    private String mPictureFileName;  
  
    FileOutputStream fos = new FileOutputStream(getOutputMediaFile(MEDIA_TYPE_IMAGE));
```

### C.4.2 Taking student's face photo and converting it openCV format

```
public class TakePhotoActivity extends Activity implements CvCameraViewListener2,
OnTouchListener {

private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {

mOpenCvCameraView = (MyCameraView)

public Mat onCameraFrame(CvCameraViewFrame inputFrame) {

Mat mRgba = inputFrame.rgba();

Mat mRgbaT = mRgba.t();

Core.flip(mRgba.t(), mRgbaT, 1);

return mRgbaT;
```

### C.4.3 Matching the two faces

```
public class PreviewImageActivity extends Activity {

FeatureDetector detector= FeatureDetector.create(FeatureDetector.ORB);

DescriptorExtractor descriptorExtractor = DescriptorExtractor .create(DescriptorExtractor.ORB);

DescriptorMatcher matcher= DescriptorMatcher .create

(DescriptorMatcher.BRUTEFORCE_HAMMING);

Mat capturedImageDescriptor = new Mat();

Mat savedImagedescriptor = new Mat();

BitmapFactory.Options options = new BitmapFactory.Options();

byte[] data = intent.getBytesExtra("image");

Bitmap bm = BitmapFactory .decodeByteArray(data, 0, data.length, options);
```

```

Bitmap passport = StudentDetailsActivity.passport;

private void doTheMagic2(Bitmap captured, Bitmap downloaded) {

captured = scaleBitMap(captured);

capturedImage = convertBitMapToMat(captured);

capturedImageKeypoints = getKp(capturedImage);

descriptorExtractor.compute(capturedImage, capturedImageKeypoints, capturedImageDescriptor);

descriptorExtractor.compute(savedImage, savedImageKeyPoints, savedImagedescriptor);

Mat resultImage = new Mat(); MatOfByte drawnMatches = new MatOfByte();

Features2d.drawMatches();

Bitmap featureImage = Bitmap.createBitmap(resultImage.cols(),

private boolean checkSimilarity(MatOfDMatch matches2) {

List<DMatch> rMatches = matches2.toList();

double maxDist = 0, minDist = 100;

int rowCount = rMatches.size();

for (int i = 0; i < rowCount; ++i)

List<DMatch> goodMatches = new ArrayList<DMatch>();

double goodDist = 2*minDist; for (int i = 0; i < rowCount; i++)

if (rMatches.get(i).distance < goodDist) goodMatches.add(rMatches.get(i));

if (goodMatches.size() >= 50 ) ;similar = true;

```