|  |
| --- |
| University of reading |
| A Face Authentication System |
| Final Year Project |
|  |

**Project Supervisor: Hong Wei**

**Project ID: A-FACE**

**Module Code: SE3IP11**

**Student Name: Tom Bedford**

**Student Number: 21009807**

**Submitted: TBD**

|  |
| --- |
|  |

**Abstract**

Computerised Face authentication is vastly being integrated into today’s technologies and systems. As an efficient and effortless method of character recognition this technology aspires to be prevalent in every-day activities with the aims of managing data and location access and delivering customized user experiences. This report illustrates the design, implementation and effectiveness of a proof of concept facial authentication system. The system utilises Viola-Jones methodology of feature extraction and PCA (Principal Component Analysis) to derive a fast and efficient means of digital face detection and verification. It features a configurable threshold acceptance value in order to handle a range of illumination levels which provide a challenging constraint in image analysis. The developed system can potentially complement an existing user authentication layer or operate as an independent authentication system. Its application can be easily extended to provide some form of greeting message or profile to future students attending a university open day

**Acknowledgments**

Hong Wei – project supervisor

Contents

[Glossary of Terms and Abbreviations 4](#_Toc449077896)

[Introduction 5](#_Toc449077897)

[Problem Articulation / Technical Specification 5](#_Toc449077898)

[The Solution 7](#_Toc449077899)

[Literature Reviews 9](#_Toc449077900)

[Implementation 11](#_Toc449077901)

[Login Interface 11](#_Toc449077902)

[Create Database Interface 15](#_Toc449077903)

[Training Interface 16](#_Toc449077904)

[Testing: Verification and Validation 20](#_Toc449077905)

[Face Detection 20](#_Toc449077906)

[Discussion 23](#_Toc449077907)

[Conclusion 24](#_Toc449077908)

[Project Commentary 24](#_Toc449077909)

[Social, Legal, Health & Safety and Ethical Issues 24](#_Toc449077910)

[Reflection 25](#_Toc449077911)

[References 25](#_Toc449077912)

[Appendices 28](#_Toc449077913)

[Appendix 1 28](#_Toc449077914)

[Primary Supervisor 28](#_Toc449077915)

[Secondary supervisor 28](#_Toc449077916)

[1 Introduction 30](#_Toc449077917)

[2 Background 30](#_Toc449077918)

[3 Problem Statement 31](#_Toc449077919)

[3.1 Objectives 31](#_Toc449077920)

[3.2 Constraints 31](#_Toc449077921)

[3.3 Assumptions 31](#_Toc449077922)

[4 Technical Products 31](#_Toc449077923)

[5 Crosscheck of approach 33](#_Toc449077924)

[6 Health and Safety 33](#_Toc449077925)

[7 Legal and Ethical 34](#_Toc449077926)

[8 Examination Products 34](#_Toc449077927)

[9 References 37](#_Toc449077928)

[10 Project Plan 38](#_Toc449077929)

[10.1 Technical Products 38](#_Toc449077930)

[10.2 Examination Products 38](#_Toc449077931)

[10.3 Time Plan for the proposed Project work 39](#_Toc449077932)

[Appendix 2 1](#_Toc449077933)

## Glossary of Terms and Abbreviations

|  |  |
| --- | --- |
| HCI | Human Computer Interaction |
| LDA | Linear Detection Algorithm |
| PID | Project Initiation Document |
| GUI | Graphical User Interface |
| MVC | Model View Controller |
| API | Application Program Interface |
| Open CV | Open Computer Vision |
| PCA | Principal Component Analysis |
| POS | Point of Sale |
| DDFD | Deep Dense Face Detection |

## Introduction

Biometric analysis has been a focused area of research for many years due to its distinctive features, ease of accessibility and potential industrial application such as government surveillance[1] and more commonly today HCI [2]. Technological advancements have provided more powerful and affordable hardware and open sourced software libraries allowing the technology to be researched more actively. This has resulted in the rise of fully automated recognition systems that are now defining their presence in the world today. Although biometric evaluation such as finger print recognition has delivered more reliable results than face recognition it requires additional expensive hardware to install and demands focused interaction from its user. With the least overhead for system instalment, invasiveness and efficiency, facial recognition has become the most popular choice of biometric analysis.

Facial recognition is the identification and verification of noticeable characteristics of a human face and in the field of image analysis it is leading the race in research as its speed and versatile application out weights previous methods.

It is a task performed effortlessly by humans on a daily basis. The sheer complexity of the human brain can truly be admired whilst replicating this complex functionality in machine software. Distinctive features of the human face such as the nose, mouth and eyes are algorithmically recognised by object detection and associated to known geometrical shapes.

Human to computer authentication is an integral functionality of many software systems as it manages data and/or location security

This project explores the application of face detection and verification and its effectiveness as an authentication based system using the open vision library developed by IBM [3]. The main algorithmic features of the developed face authentication system in this project focus on the detection and verification of a human user. The system efficiency, accuracy and speed is tested by serious of test on compiled face databases sourced from various establishments’ online repositories.

## Problem Articulation / Technical Specification

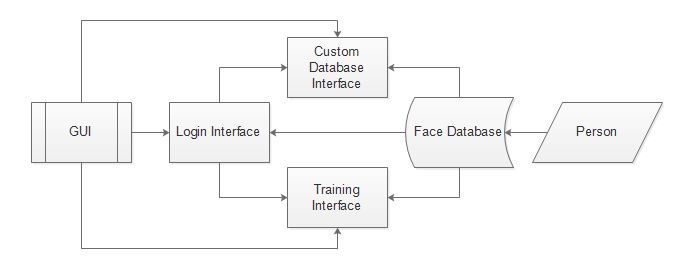
The fundamental goals of the developed system in this study is the successful detection of a user’s face and the verification of a known users within a targeted face databases. Acceptance criteria is satisfied by LDA analysis determining whether the users extracted geometrical features fall within a given threshold in order to grant them access to the application. The threshold of acceptance will be adaptable to suit the needs of the systems environment which substantially dictates acceptance rates of the recognition process. Illumination, user posture, direction, expression and face size are the greatest affecting factors in face recognition computation [4].

It is assumed that users will present there face at an offset no greater than a 35 degree angle and the camera will be of sufficient quality to capture images of an adequate resolution. The user will not be wearing items that obstruct large portions of the face. To provide a reliably secure authentication system recognition results must fall within a refined acceptance threshold that ensures access only to known users of the system. These assumptions and constraints where originally defined in the PID [Appendix 1] report referenced at the end of this document submitted at the beginning of this project.

To satisfy the project objectives derived in the PID each deliverable must meet its acceptance criteria.

The proposed solutions are based on addressing three main modules of the system’s functionality, the GUI, face detection and verification functionality.

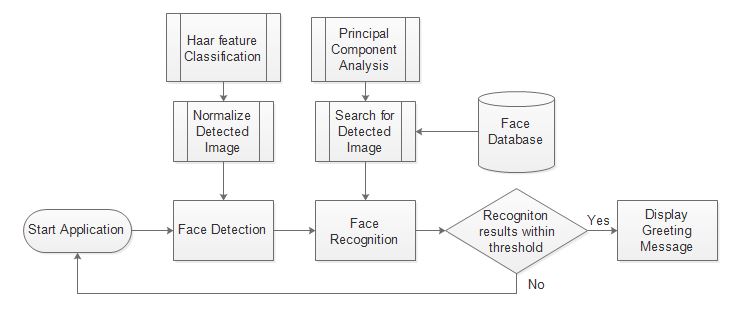
The GUI must be developed to provide a simple interface for the user to detect and capture images of their face to be submitted as part of the authentication process. The GUI will also need to host a training screen where the user can select, load and train face databases prior to user verification.



**Figure. 1** Face authentication graphical user Interface design.

Figure 1 illustrates the work flow of the proposed GUI. It can be seen that the programs user interface is built up of three main displays, the login, training and custom database screens. The login and training screens are essential to the systems core functionality for loading and training data and the custom database screen is to provide further customisation for creating and adapting databases. Developing the GUI in Java language will be less complex and time consuming than C++.

This solution will be developed in Java CV [5] a ported version of the natively C++ Open CV library. The Open CV java library essentially wraps the C++ functionality in Java so that it can be utilised in Java code. JavaFX [6] will be used to develop the systems GUI allowing it be implemented as an MVC framework to allow the systems API to be reused by other systems easily.



**Figure 2.** Recognition process of the facial authentication system.

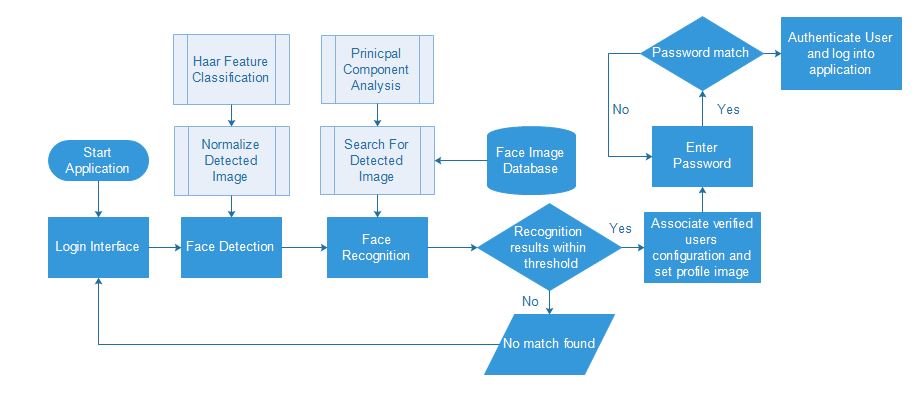
The face detection functionality will be capable of acknowledging a human face by recognising geometrical features of a human face with the use of trained Haar-like feature cascade classifiers provided by the Open CV library [7]. The classifier iterates through image pixel data verify any know geometrical shapes whilst optimising its search results using an AdaBoost algorithm [8]. To satisfy the systems acceptance criteria detection must successfully capture a frontal facing image within an offset of 35 degree angle at a success rate of 90%.

Detected images will then need to be normalised to conform to standardized dimensions of the system and to reduce computation. This is achieved by resizing the detected image and then converting the colour channels to an average greyscale channel.

Principal component analysis will be used reduce and optimise data representation. It is the most popular and efficient LDA’s (Linear Detection Algorithm) in developing facial recognition technologies today. The PCA approach

## The Solution

**Figure. 3** Proposed system solution design



The proposed solution as displayed in figure.3 will be developed using a Java wrapped Open CV library Java CV to enable simpler development of the GUI’s so that efforts can be focused towards the main algorithmic features of the system. JavaFX library will be used to implement an MVC structure allowing the main functionality of the system to be utilised or integrated into other systems easily.

The open vison library supplies a variety of classifiers that have been trained to detect geometrical feature shapes of a human face. The classifier ‘haarClassifier\_frontalFace\_alt.xml’ that will be used to perform face detection is based on viola jones methodology [9]. The detected image is then normalized by rescaling and greyscale conversion to reduce computation and conform to dimensional standards of the recognition algorithm. As the Java CV library does not incorporate the Face Recognizer class [10] a custom PCA class will be developed by the use of Open CV and third party math libraries.

This solution aims to verify the user and associate them to the relevant profile though the login interface. A password is then entered to ensure a secure authentication system. It was decided not to use the facial authentication system developed in this project as a stand-alone security system and should instead compliment an existing system as an extra layer of security.

The main algorithm of the system will be principal component analysis to calculate Eigen Decomposition and effectively derive Eigen Faces [11]. The system will initially provide functionality to load directories of face images from local memory to the application and may be expanded to access online repositories depending on objective deliverance deadlines met.

The application GUI will be based on the module illustrated in figure.1 earlier in this document. The design will provide methods to standardize loaded images prior to verification or detection computation.

The original determined acceptance criteria of the PID documents technical products have been refined and illustrated below in figure.4.

|  |  |
| --- | --- |
| **Technical Product** | **Acceptance Criteria** |
| Graphical user interface | The GUI must provide functionality for the user to capture, load and train images. It must also provide a login screen for authentication and a platform where verification can be tested. |
| Capturing and storing images as graphical files | The system will feature functionality for the user to capture and save images and trained data sets. |
| Computing face detection | A user’s face will be successfully detected and represented in the video camera feed. To satisfy acceptance it should be able to detect a face on offset no greater than 35 degrees. |
| Computing face verification | Images from a trained database should be able to match features of an input image within an adequate threshold at 75% success rate with a minimal false negative match rate. |
| Integrate face database into system | The user will be able to load a face database from local memory into the application and functionality to train the data set. Images must be standardized to comply with system standards. |
| Display recognition results of acceptance and rejection rates | The system will display acceptance threshold rates and associated face matches. It will also show rejection rates of verifications outside of a set threshold. |

**Figure.4** Refined acceptance criteria of technical products

## Literature Reviews

Methods of human authentication have previously and still are performed using unique credentials such as personal ID cards, passports, unique passwords and phrases. These methods of authentication often require some form of physical interaction or focus from its user. The form of this uniquely identifiable data has largely been an extension of a person such as a passport. Biometrics offer the convenience of no extra materials, assets or memory cognition as they are physical features that individually identify a person.

The face is the most exposed definitive characteristic of a human and therefore can be analysed with little demand of interaction from the person, if any. A person need only look towards the scanning device which in this case is a camera for seconds and recognition can be successfully performed in real time or from a captured image frame on reasonable low quality images. Face recognition is a popular method of identification in government surveillance systems [12]. It can be carried out on multiples entities simultaneously with speed which is extremely effective in crowds of people.

In the infantile stages of facial analysis feature extraction was performed manually. The automation of this process in real time was introduced by Paul Viola and Michael Jones in 2001 [9]. Not only does the algorithm detect facial features in real time but performs this with a simpler complexity of computation allowing computers with low processing power to utilise the software. The Viola-Jones method of Haar-like feature classification revolutionised face detection techniques. All methods of computerised human identification operates on the basis that a set of uniquely identifiable data is submitted by a user to the system where it is compared and associated to its relevant data within a determined threshold of acceptance to verify authentication.

In December last year the company bioID [13] released a facial recognition software solution for apple hardware. The software allows the use of face authentication to authorize access to devices such as the IPad and IPhone and is steadily integrating it services towards bank payments, POS and other high security transactions. It features a patented liveness detection algorithm [14] to determine whether the user is fact a real person or not. Liveness detection is become a common practice in face detection in the efforts to prevent spoofing and unauthorized access. BioID’s software also host a rich development package that can be easily integrated into software development to encourage developers to incorporate facial authentication into future software applications. Other programming language versions of this software are slowly becoming available.

Cognitec is another company offering multipurpose face recognition system solutions for use in security authentication in fields such as law enforcement, border control and ID fraud protection. The company has been researching facial recognition technology in collaboration with the university of Surrey and Technical University of Dresden since 2010 [15].

In the stride for more secure and reliable biometric analysis recognition, methods are being expanded to extract further features that can determine an inanimate user. For instance to combat finger print scanner spoofing thermo-detection hardware is being added to scanners in order to register a heat signature that would suggest a real user is attempting authenticate [16]. In regards to enhancing face verification security with liveness detection, tracking eyelid movement is proving an effective means of countering falsified verification.

Again last year there was the release of an advanced facial recognition system called DDFD [17]. The detection algorithm has been extended from the foundations of Viola-Jones methodology of face detection and uses a convolution neural network to learn a composition of facial features. Impressively this algorithm has been trained on a database containing over 21 thousand images. This is one of the most robust face detection systems on the market today as it can successfully detect partial faces, upside down oriented faces and largely obstructed faces.

In a few months from now Microsoft windows 10 will be introducing a new component dubbed ‘Windows Hello’ [18] to their popular operating system. This new functionality will allow windows users to login using face recognition technologies. Microsoft claim this will be more secure than older authentication methods like passwords and will strengthen computer security as a whole.

There are many organisations investing largely in face recognition development as the vision of effortless human computer interaction becomes a reality. The United States National Intelligence Agency launched a development program in early 2014 named Janis [19]. The aims of Janis is to compile a broad database of facial morphology derived from online digital media and sources to understand and develop more sophisticated and optimal facial analysis techniques. This rich database will provide an intensive testing plane for facial recognition algorithms and will accelerate their development.

A high street store has invested in face recognition software to verify high profile customs such as celebrities so that a high quality customer service can be delivered. The investment was motivated after a well renowned celebrity entered the store one day and the manager serving them had no knowledge of their social status missing the opportunity of a lucrative sale and repeat business [20]. This example demonstrates the versatile application of face authentication systems being applied in services other than security.

Well renowned security conferences such as the Black Hat [21] and Def Con [22] conferences attract some of the world’s leading specialists in information security under one roof to diverge and discuss security practices and technologies. An article posted shortly after the Black Hat conference back in 2009 in Washington DC voiced concerns over facial biometric use for authentication. The salient features of the report demonstrated how early versions of face authentication integrated into Lenovo, Asus and Toshiba machines could be hacked and did not fulfil the necessary requirements. Although facial analysis methods are becoming more sophisticated and reliable for security there are still concerns over the handling and management of biometric data due to its permanent nature.

A group of biometric computation experts demonstrated at the 2015 Black Hat conference how retinal analysis could be re-engineered and potentially hacked to falsify authentication [23]. This form of re-engineering firstly required gaining access to the stored biometrical data but illustrated that even the most complex of biometric analysis could be exploited. Amongst the concerns of biometrical data management their also worries that the integration of facial recognition systems increase surveillance coverage and reducing the public privacy.

The huge leap in face analysis development has also attracted further developments in existing recognition techniques such as speech recognition [24]. As a desired means of HCI this is a contestant not far behind face recognition.

PCA is one of the most successful face detection algorithms to date. PCA is the method of reducing dimensionality of a data space (observational variables) and extracting data from the feature space (Independent Variables) which are needed to represent the data economically. The feature space in this case is the Eigen Space in which derived principal components are projected to deduce Eigen values.

For other image analysis systems who are also challenged by lighting factors technologies such as Infrared and thermal imaging is be introduced. For instance infrared technologies are currently being used in carparks to simulate a daytime like environment to extract car number plates in poor lighting conditions [25]. This allows image recognition methods to extract the characters of the number plate. Infrared scanners used in this capacity are expensive to install and maintain but would provide more mobility of the system and more potential recognition environments.

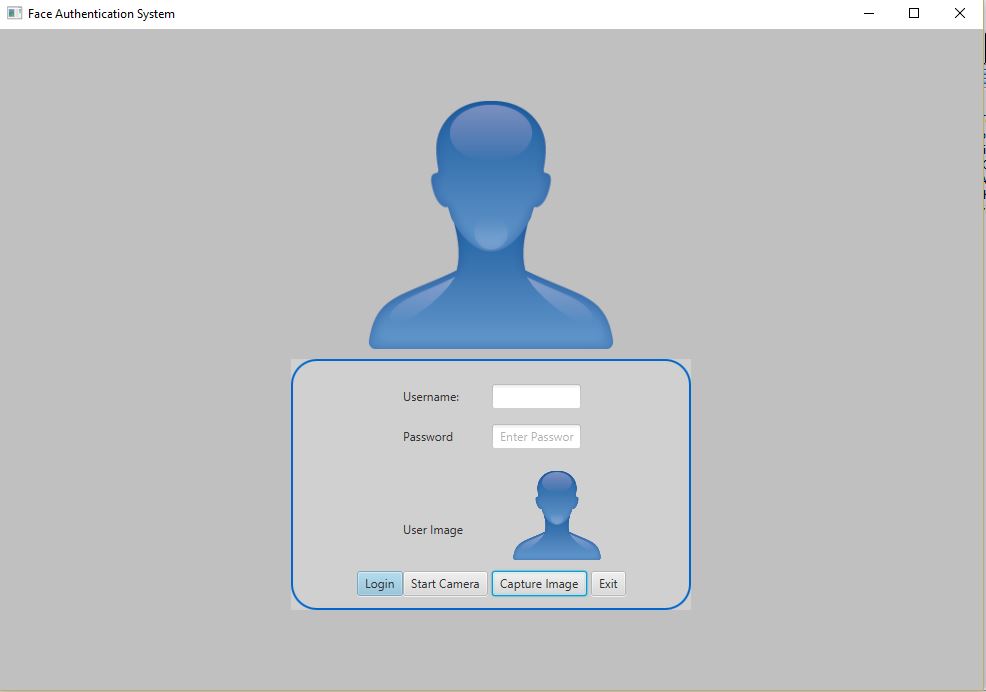
Extended face recognition research investigates the use of thermal cameras to collect image data. This allows a better representation of a 3D face model as it does not detect facial items such as glasses and records only heat signature data [26]. The system environments lighting conditions are a CCP of the systems functionality and its placement must be considered careful or additional technology will be needed.

## Implementation

### Login Interface

A basic GUI framework was initially implemented and later enhanced to satisfy the needs of interaction to the system. The JavaFX library used to develop the GUI allowed the creation of an MVC based model isolating the API and GUI allowing modular reuse of the system. Figure.5 illustrates a screen shot of the login interface.

**Figure.5** Screenshot of login screen graphical user interface



The login screen provides an area where the user can test user authentication. On first time login of the system access is granted using an administrative username and password so that an initial face image data set can be loaded and trained. After the trained data has been produced it is then written to memory so that then a known user can login using the facial verification functionality.

The login screen hosts a video capture stream activated by the ‘Start/Stop Camera’ button. The video capture objects implemented throughout the system is sourced from the ‘opencv.videoio’ library. Video stream data is handled in memory using Open CV’s Matrices object format ‘Mat’. To comply with Java AWT [Java AWT] image display standards ‘Mat’ objects are converted to a Java.awt Image using Open CV’s image codecs in order to display them in the GUI. This conversion functionality is displayed below in figure.6

|  |
| --- |
| **protected** Image Mat2Image(Mat frame) {  // temporary buffer  MatOfByte buffer = **new** MatOfByte();  // encode image frame into PNG format  Imgcodecs.*imencode*(".PNG", frame, buffer);  // build image from encoded buffered data  **return** **new** Image(**new** ByteArrayInputStream(buffer.toArray()));  } |

**Figure.6** Open CV Matrix to Java AWT conversion function

Each grabbed frame from the video stream is processed using the Face Detector class developed for this application. The face detector class loads into memory a trained cascade classifier ‘haarcascade\_FrontalFace\_alt.xml’ supplied by the Open CV library. The grabbed frame is converted to greyscale to reduce colour channels and effectively data to process. Greyscale co nversion functionality is incorporated in Open CV’s Imgproc module originally derived from OpenCV core module. During system runtime there are multiple areas of the application where image type conversion is computed. The class AppTools.class has been developed to manage these conversions and has been referenced in the Appendices of this report.

The face detection algorithm is performed on each grabbed frame of the video sequence to determine a human face through Haar-like feature cascade classification. This involves loading the classifier into the Cascade Classifier object supported by the Open CV library. The frame is then prepares for detection by converting the frame to greyscale and calculating the minimum face size. The following code seen in figure.7 illustrates this preparation.

|  |
| --- |
| //create cascade classifier  CascadeClassifier faceDetector = **new** CascadeClassifier();  //absolute path to classifiers  String classifierPath = "C:\\Users\\user\\workspace\\AFaceAuthenticationSystem\\src\\application\\Resources\\HaarCascades\\haarcascade\_frontalface\_alt.xml";  //load classifiers  faceDetector.load(classifierPath);    //Create Mat canvas to store detections  MatOfRect faceDetections = **new** MatOfRect();  Mat greyScaleImg = **new** Mat();    // convert the frame in gray scale  Imgproc.*cvtColor*(imageMat, greyScaleImg, Imgproc.***COLOR\_BGR2GRAY***);  // compute minimum face size (20% of the frame height)  **if** (absoluteFaceSize == 0)  {  **int** height = greyScaleImg.rows();  **if** (Math.round(height \* 0.2f) > 0)  {  absoluteFaceSize = Math.round(height \* 0.2f);  }  } |

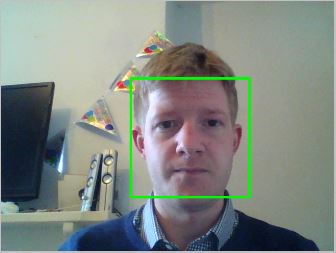
**Figure.7** Cascade classification preparation

Detection is then performed on the prepared frame. This process iterates through the frames pixel data to distinguish known geometrical features of the classifier.

|  |
| --- |
| // detect faces  faceDetector.detectMultiScale(greyScaleImg, faceDetections, 1.1, 2, 0 | Objdetect.***CASCADE\_SCALE\_IMAGE***, **new** Size(  absoluteFaceSize, absoluteFaceSize), **new** Size());  //create array of face detections computed  Rect[] faceArray = faceDetections.toArray();  //iterate through the image  **for** (**int** i=0; i<faceArray.length; i++)  {  //find rectangle contours of faces  Imgproc.*rectangle*(imageMat, faceArray[i].tl(), faceArray[i].br(), **new** Scalar(0, 250, 0, 255), 3);  //crop image of face  Mat crop = imageMat.submat(faceArray[i]); |

**Figure.8** Calculate face detections functionality from Face Detector class

Detected faces in each frame are then outlined with a green rectangle of the calculated absolute face dimensions in figure.9 to signify face detection to the user.



**Figure.9** Face Detection

Once the user has logged in for the first time added there detected face to a dataset and computed Eigen faces through the training interface an input image can then be verified against the applications trained data on login authentication. This is done by a similar process to training the data set where firstly the average face is subtracted from the input image and then the weights and distances of the image minus averages are calculated. Figure.10 and 11 demonstrate this processes logic.

|  |
| --- |
| **private** Matrix getWeights(Matrix matrix, **int** numOfEigenFaces) {  Matrix eigenFaces = **new** Matrix(cache.getEigenFaces());  Matrix selectedEigenFaces = eigenFaces.getMatrix(0,  eigenFaces.getRowDimension() <= numOfEigenFaces ? eigenFaces.getRowDimension() - 1 : numOfEigenFaces, 0,  eigenFaces.getColumnDimension() - 1);  **return** matrix.times(selectedEigenFaces.transpose());  } |

**Figure.10** Calculate weights of input image functionality

|  |
| --- |
| **private** **double**[] getDistances(Matrix inputWeights) {  **double**[] weightsData = inputWeights.getArray()[0];  **double**[][] tempWeights = subtractFromEachRow(cache.getEigenWeights(), weightsData);  tempWeights = squareNonMatrix(tempWeights);  **double**[] distances = **new** **double**[tempWeights.length];  **for** (**int** i = 0; i < tempWeights.length; ++i) {  **double** total = 0;  **for**(**int** j=0;j<tempWeights[i].length ;++j) {  total += tempWeights[i][j];  }  distances[i] = total;  }  **return** distances;  } |

**Figure.11** Calculate distances of input image functionality

Once the distance and weights of the input image have been determined the previously trained data set is read into the application in order to compare these values. Loading in the trained data into memory on the login process reduces computational time opposed to having to train datasets on each login attempt. The trained data is read back into memory with the functionality illustrated in figure.12.

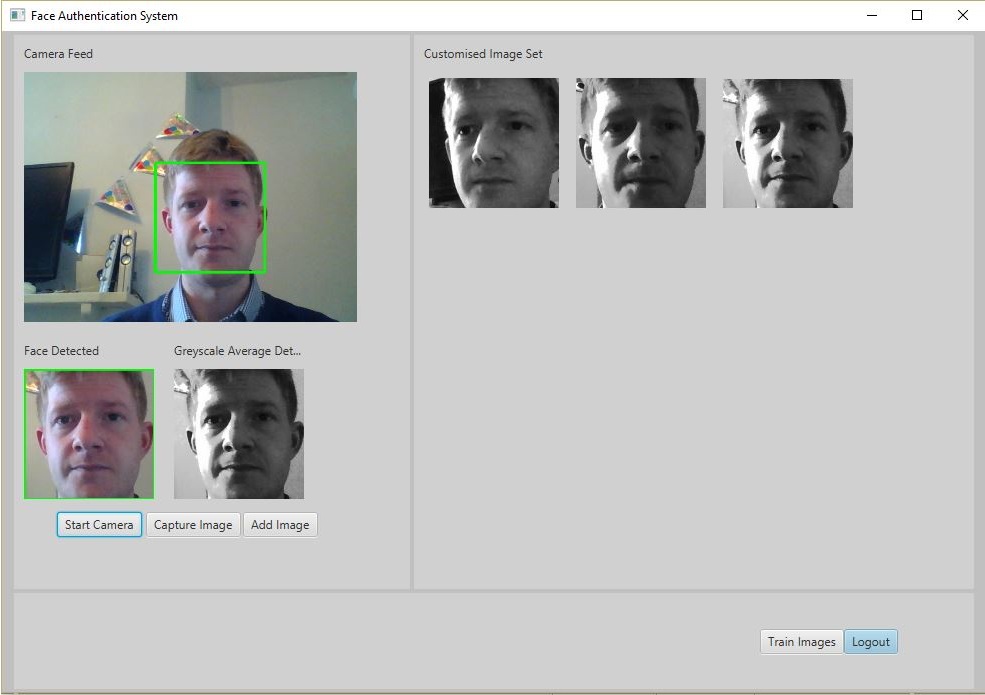
|  |
| --- |
| **private** **void** loadEigenCache() {  // read eigen cache into memory  FileInputStream fs;  **try** {  fs = **new** FileInputStream("C:\\Users\\user\\workspace\\AFaceAuthenticationSystem\\EigenCache\\eigenCache.db");  ObjectInputStream os = **new** ObjectInputStream(fs);  eigenCache = (EigenCache) os.readObject();  // close streams  os.close();  fs.close();  } **catch** (IOException e) {  e.printStackTrace();  } **catch** (ClassNotFoundException e) {  e.printStackTrace();  }  } |

**Figure.12** Load cached trained data into memory functionality

Here a comparison can be made of the calculated Eigen Vectors and Values to determine the distance from the average face. This is then accepted or rejected if the distance is within the set threshold.

### Create Database Interface

The create database interface provides an area where a first time or existing user can capture face detections and either create a customized database or add to an existing one. Essentially as set of images can be obtained and saved as displayed in figure.13. The interface utilises the Face detection algorithm and Open CV video capture stream components used in the login screen.



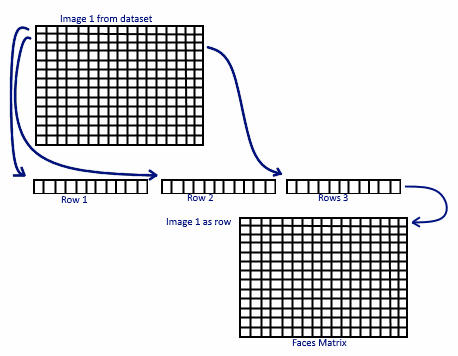
**Figure.13** Create database interface screenshot

### Training Interface

Limitations of the ported Open CV library Java CV where discovered here. The native C++ Open CV library supports a Face Recognizer class that features PCA (Principal Component Analysis) computation functionality. As this module of the Open CV has not yet been ported to the Java CV library a custom PCA class was implemented in Java with the aid of third party apache commons Math [27] and Jama Matrices libraries [28]. The custom PCA class developed for this project has referenced in the Appendices of at the end of the report.

The training interface provides the user the ability to load, train and write trained data sets to memory. It also provides a testing ground for trained data sets where the user can select an image to perform face verification on the loaded database. Matches, if any, are displayed along with their calculated thresholds. The thresholds represent the distance from the average face. These calculated thresholds are then accepted if they fall within a set threshold of acceptance. In order to manage the illumination changes of the different environments in which the system may be used a slider has been implemented in the training interface to allow the threshold of acceptance to be adjusted.

PCA concatenates each row of pixel data from an image of the dataset into a one dimensional array. Once each image is represented as a row a Matrix is built up of each image as a row. The logic to method can be seen in figure.14 below.



**Figure.14** Face matrix preparation

Once the faces matrix has been constructed pixel values of the data are normalised between zero and one. During the greyscale averaging computation the RGB values are combined to create large decimal numbers. Normalization is performed to unify variable proportions and standardize data in the system. Figure.15 displays the system normalization function.

|  |
| --- |
| **private** **double**[][] normalizeImageData(**int**[][] image) {  **double**[][] returnData = **new** **double**[image.length][image[0].length];  // Normalise data between 0 and 1  **for** (**int** faceInd = 0; faceInd < image.length; ++faceInd) {  **double** min = (**double**) getMinValue(image[faceInd]);  **double** max = (**double**) getMaxValue(image[faceInd]);  **for** (**int** j = 0; j < image[faceInd].length; ++j)  returnData[faceInd][j] = ((((**double**) image[faceInd][j])) - min) / (max - min);  }  **return** returnData;  } |

**Figure.15** Normalisation function

Next the mean is subtracted from each row and a covariance matrix is compiled. This was carried out with the aid of the Apache Commons Math library. The Apache library is then again used to calculate Eigen vectors and values. Eigen Vectors represent the direction of data and the associated Eigen Values represent the distance of the data from the average face. The average face used to determine Eigen decomposition is produced by finding the average data from the compiled face matrix as described in figure.16. The calculate Eigen decomposition function.

|  |
| --- |
| **private** **void** calculateEigenVectorsAndValues() {  // Log.append("Computing covariance matrix...");  // Compute covariance matrix  RealMatrix matrix = **new** Covariance(**new** BlockRealMatrix(faceMatrixMinusAverages).transpose())  .getCovarianceMatrix();  // Log.append("Computing eigen decomposition...");  // Get the eigenvalues and eigenvectors  EigenDecomposition eigen = **new** EigenDecomposition(matrix);  eigenValues = eigen.getRealEigenvalues();  // Transpose so that eigenvalues are in vectors/columns  eigenVectors = eigen.getV().transpose().getData();  } |

**Figure.16** calculate Eigen decomposition function

The Eigen vectors that represent the most variance are then removed and a set amount of Eigen vectors are selected to represent the Principal Components. A function was developed to compute the principal components and then sort them into descending order prior to selection as displayed in figure.17

|  |
| --- |
| **private** **void** calculatePrincipalComponents() {  **int** numOfComponents = eigenVectors.length;  // Get principle components  ArrayList<PrincipalComponent> principalComponents = **new** ArrayList<PrincipalComponent>();  **for** (**int** i = 0; i < numOfComponents; i++) {  **double**[] eigenVector = **new** **double**[numOfComponents];  **for** (**int** j = 0; j < numOfComponents; j++) {  eigenVector[j] = eigenVectors[i][j];  }  principalComponents.add(**new** PrincipalComponent(eigenValues[i], eigenVector));  }  // sort components  Collections.*sort*(principalComponents);  Iterator<PrincipalComponent> iterator = principalComponents.iterator();  **int** count = 0;  **double**[][] tempVectors = **new** **double**[3][eigenVectors.length];  **double**[] tempValues = **new** **double**[3];  **while** (iterator.hasNext()) {  PrincipalComponent pc = iterator.next();  **if** (count < 3) {  tempVectors[count] = pc.eigenVector;  tempValues[count] = pc.eigenValue;  } **else** {  eigenVectors[count - 3] = pc.eigenVector;  eigenValues[count - 3] = pc.eigenValue;  }  count++;  }  } |

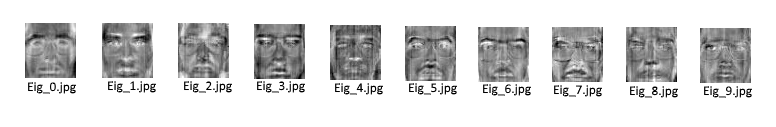
**Figure.17** calculate Principal Component function

|  |
| --- |
| **private** **void** calculateEigenFacesAndWeights () {  **int** pixelTotal = faceMatrixMinusAverages[0].length;  **int** imageTotal = faceMatrixMinusAverages.length;  **int** vectorTotal = eigenVectors.length;  // calculate eigen faces  **double**[][] eigenFace = **new** **double**[vectorTotal][pixelTotal];  **for** (**int** i = 0; i < vectorTotal; ++i) {  **double** squaredSum = 0;  **for** (**int** j = 0; j < pixelTotal; ++j) {  **for** (**int** k = 0; k < imageTotal; ++k) {  eigenFace[i][j] += faceMatrixMinusAverages[k][j] \* eigenVectors[i][k];  }  squaredSum += eigenFace[i][j] \* eigenFace[i][j];  }  **double** norm = Math.*sqrt*(squaredSum);  **for** (**int** j = 0; j < pixelTotal; j++) {  eigenFace[i][j] /= norm;  }  }  // get specified amount of eigen faces  **this**.eigenFaces = **new** Matrix(eigenFace).getMatrix(0,  eigenFace.length <= numOfEigenFacesSelected ? eigenFace.length - 1 : numOfEigenFacesSelected, 0,  eigenFace[0].length - 1).getArray();  **this**.eigenWeights = **new** Matrix(faceMatrixMinusAverages).times(**new** Matrix(eigenFaces).transpose()).getArray();  } |

**Figure. 18** calculate Eigen face and weights function

Figure.18 illustrates the programming logic use to calculate the Eigen faces and corresponding weights. The use of a third party library called JAMA was used to support matrices computation. The derived Eigen faces are converted back to Java AWT buffered images and written as Jpeg format into memory in order to visualize the variance. A sample of 10 computed Eigen faces are displayed in figure.19.

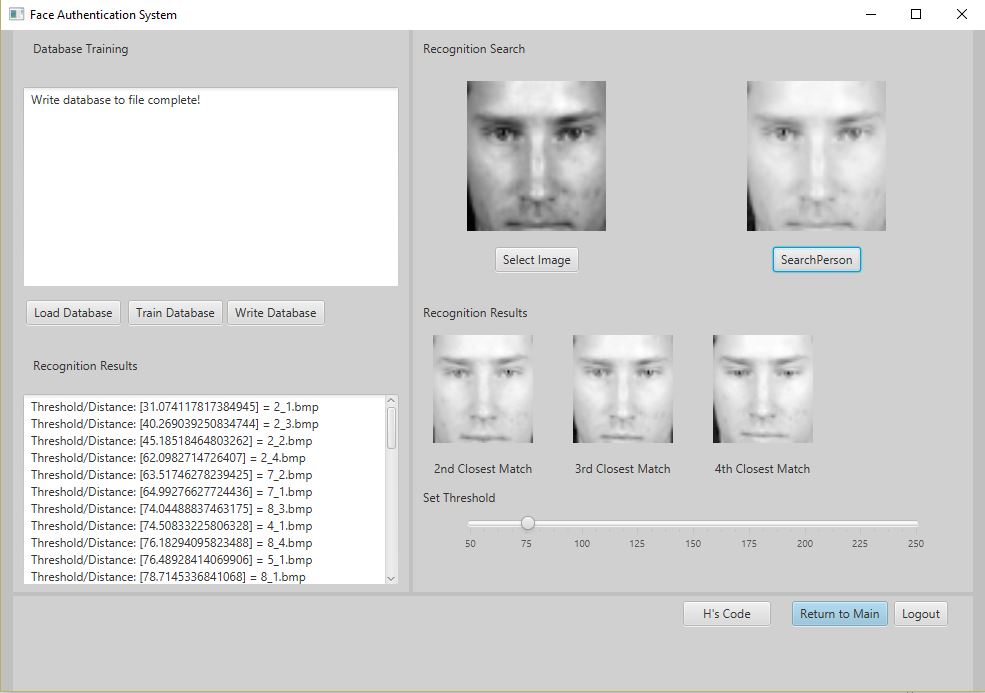
|  |
| --- |
| // construct buffered image from eigen faces  BufferedImage[] constructedEFaces = **new** BufferedImage[faceMatrix\_array.length];  **for** (**int** i = 0; i < faceMatrix\_array.length; i++) {  constructedEFaces[i] = getImageByEigenValues(eigenWeights[i], eigenFaces);  } |

**Figure.19** Constructing Eigen Faces to Jpeg images.

**Figure.20** Calculated Eigen faces for visualizing variance.

From observing the Eigen faces in Figure.20 It can be seen that the first few Eigen faces represent the most amount of variance leading to the least variance.

Figure.21 illustrates a screen shot of face verification matches and thresholds results after performing a search.

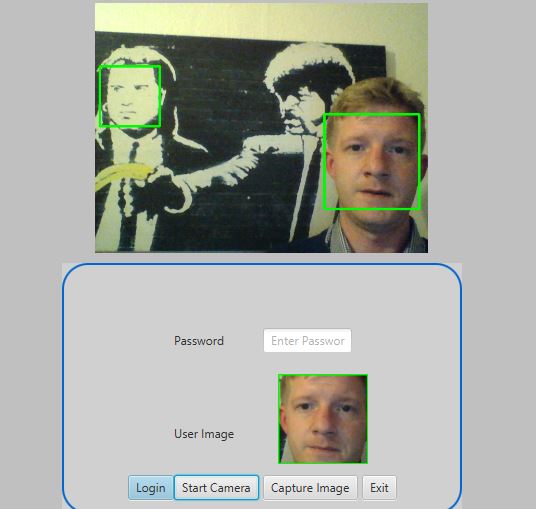


**Figure.21** Screen shot of Face verification results after search performed

## Testing: Verification and Validation

### Face Detection

During face detection runtime multiple faces where detected and highlighted in the visible frames of the video stream as demonstrated in Figure.22.



**Figure.22** Multiple faces detected

As the system only wishes to process one user at a time the most prominent face is selected as the detected face. Also as seen in figure.22 the robustness of the cascade classifier can be observed as the algorithm has successfully located a human face in the background from a picture with very little facial detail.

Face detection is performed successfully within the specified offset and adequate illumination. Without adequate lighting conditions the face detection success rate drops dramatically.

With large data sets Eigen decomposition can take a substantial amount of time.

A serious of tests where carried using the XM2VS database compliments of the University of Surrey [29], the AT&T face database supplied by the University of Cambridge [30] and the Yale database from the UCSD Computer Science Engineering Department [31]. Each of these database contain frontal facing greyscale channel images. The Yale database has much more variation in face gestures and lighting than the other two. The statistics and length of time taken to train of each database is recorded in figure.23

|  |  |  |  |
| --- | --- | --- | --- |
| **Database** | **Number of Images** | **Training Time (minutes)** | **Training Time Per Image (milliseconds)** |
| Xm2vts | 2,361 | 12.654 | 322.00 |
| Xm2vts Sub Sample | 40 | 0.214 | 5.35 |
| Yale | 165 | 0.924 | 5.60 |
| AT&T | 401 | 2.276 | 5.67 |

**Figure.23** Training statistics of three test face databases

The face recognition system was tested with a range of thresholds to determine percentages of acceptance rates. The bar chart illustrated in figure.24 shows these results.

**Figure.24** Recognition results for xm2vts and AT&T database

The results show better acceptance rates for the xm2tvs compared to the AT&T database. Illumination levels largely dictate the effectiveness of feature extraction. Images in the AT&T library are slightly lower quality of illumination than those in the xm2tvs library which could explain the lower acceptance rates of the results.

It can be deduced from figure.24 that if the threshold is raise to high then the number of rejections and essentially false matches increase. The threshold can be adjusted to suit the environments lighting to a certain degree although for optimal use of the system is advised that the illumination of the environment where the application will be used should be taken into considerate and accommodated for.

In relation to the refined acceptance criteria of technical products illustrated in figure.4 a table has been comprised below to display whether determined criteria where met.

|  |  |  |
| --- | --- | --- |
| **Technical Products** | **Criteria Acceptance achieved** | **Criteria Satisfaction description** |
| Graphical User Interface | Achieved | The graphical user interface has delivered interfaces for the user to capture, load and train image data sets. The GUI host an area to test verification on a trained data set and a login authentication page to test the verification security strength. |
| Capturing and Storing Images as Graphical files | Achieved | The user can successfully save captured images and load images into application memory throughout the system. |
| Compute Face Detection | Achieved | A face detection algorithm was successfully developed and achieved the expected thresholds of a 85% acceptance rate for frontal images within an offset of a 35 degree angle. |
| Compute Face Recognition | Achieved | The most demanding task of computing PCA without the support of the open CV library was achieved with the use of third party math and matrices libraries. |
| Integrate Face Database | Achieved | Functionality was created to allow a user to load databases into the application memory. Three database from various Universities where loaded into the application to train and test the verification functionality |
| Display Recognition results displaying acceptance and rejection rates | Achieved | The Training interface featured a text area where thresholds of acceptance and rejection where displayed along with verification statistics. |

**Figure.25** reflection of technical products and their achievements

## Discussion

On the basis of the deliverance of the proposed project deliverables the project was a reasonable success. All besides a few minor short falls the main demands of the system where met.

Some data is being lost when computing the Eigen faces. It is suspected that during the normalization process variables of type double are being converted to integers resulting in the rounding of decimal numbers and thus this loss of data can be seen illustrated in figure.26. This small loss of data has a minimal effect on the face recognition results although it could be refined to optimise the algorithms effectiveness given more time.



F**igure.26** Data loss through normalization and de-normalisation rounding

The strength of a stand-alone face recognition system used as a main authentication tool doesn’t appear to be the safest method of computer recognition as the face detection can be spoofed.

Inanimate images/photos can be held in front of the camera and a face is successfully detected. Open CV’s trained classifiers are robust enough to detect images with very little features. This poses a security issue as user recognition can be spoofed without the necessary precautions being implemented to counter such forgery.

For the proposed application purpose of an open day student verification and greeting system it would be a realistic assumption that the database to train would only contain a few hundred images opposed to the 2,361 images trained from the xm2vts database. This would reduce the training time exceptionally and could allow for a new face to be detect, added to the database, trained then verified to display a greeting message in real time. As this would pose limitations to the growth of the database it would most probably be better to train the database prior to recognition to ensure the most effective result.

## Conclusion

Naturally without the exposure and vision of completed project the originally determined success criteria in the PID document where very general and needed further definition. Time deadlines were also unrealistic as technical products such as the face verification functionality appeared to be far more demanding than first estimated. Although, with in consideration the project manage to deliver a fully functioning face authentication system and meet set objectives.

As there is no liveness detection functionality featured the system can be fooled by holding a photo of a face which can be successfully detected. This concludes that this application should be used to compliment an existing security system as an extra layer and not as stand-alone security system due to potentially falsifying authentication demonstrated in the testing of this system.

Extended goals such as integration of an online face database repository where not achieved due to time constraints and other coursework commitments.

The acceptance objectives where satisfied to produce a functioning face authentication application.

## Project Commentary

Developing a custom PCA class consumed more project time than originally designated. This had an effect on the amount of time spent developing other system features. In Hein sight I would of used the JNI to utilise Open CV C++ functionality. The native program language Open CV library contains a face recognizer class that can determine age and gender of a user. It also features a PCA class that would have saved much time when computing Eigen decomposition. Alternatively the project could have been developed using the native C++ library which would of provide more of a code learning curve but aided complicated computation such as principal component analysis.

There was a steep learning curve in matrix mathematics and its application in code which has provided a motivational challenge.

GIT Version control software was used to throughout the development of the project as online code back up and for rolling backing versions of code.

## Social, Legal, Health & Safety and Ethical Issues

There are many ethical arguments into the analysis and storage of biometrical data. As biometrics uniquely identify a person and cannot be replaced as sparingly as credentials like a password. If your authentication password or forms of identity such as bank card become compromised they can easily be changed although human like features such as the face are much more permanent.

The ethical and social use of face recognition has been questioned much already as the public feel that their civil liberties and privacy are being invaded by the continuous surveillance and gathering of public data. With the ever growing technical strive to innovate methods of HCI and biometric analysis its view by some that we are slowly losing our privacy. An article released by abc news in late December 2015 voiced concerns about this privacy invasion [32].

With current events in the world today, surveillance should make the public feel safer. It is unfortunate that quite often technology is abused in the wrong way but this shouldn’t affect the public decision to compromise and grow with technology.

## Reflection

This project has delivered invaluable knowledge and experience working with an open source computer vision library. It has provided many challenges that has enhanced my academic skills in programming, time management, research and documentation.

Many technical competencies have been exercised and improved through the completion of this project. I know have a clearer understanding and approach to unfamiliar programming language, libraries and practices.

My managerial skills have excelled to meet the demands of this project which has supported the growth and refinement of my skill set.

## References

[1] Jose Pagliery. (2014). *FBI launches face recognition system.* Available: http://money.cnn.com/2014/09/16/technology/security/fbi-facial-recognition/. Last accessed 12th Jan 2016.

[2] Reza Azad, Babak Azad, Nabil Belhaj Khalifa, Shahram Jamali. (2014). Real-Time Human-Computer Interaction Based on Face and Hand Gesture Recognition. *International Journal in Foundations of Computer Science & Technology*. 4 (4), 11.

[3] OpenCV Developers Team. (2016). *About OpenCV.* Available: http://opencv.org/about.html. Last accessed 13th Jan 2016.

[4] Hewlett-Packard Development Company. (2015). *HP IDOL.* Available: https://my.vertica.com/docs/IDOL/Servers/IDOLServer/10.9/Guides/html/Expert/index.html#IDOLExpert/Improve/FaceDetect\_factors.htm. Last accessed 21st Jan 2016.

[5] OpenCV Developers Team. (2016). *OpenCV 3.0.0.* Available: http://docs.opencv.org/java/3.0.0/. Last accessed Feb 2nd 2016.

[6] Oracle Developers Team. (2011). *Oracle Java Documentation.*Available: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784. Last accessed Feb 2nd 2016.

[7] Open CV Dev Team. (2016). *Cascade Classification.* Available: http://docs.opencv.org/2.4/modules/objdetect/doc/cascade\_classification.html. Last accessed Feb 10th 2016.

[8] The MathWorks, inc. (2016). *AdaBoost.* Available: http://uk.mathworks.com/discovery/adaboost.html?refresh=true. Last accessed Feb 20th 2016.

[9] Wikipedia. (2016). *Viola–Jones object detection framework.* Available: https://en.wikipedia.org/wiki/Viola%E2%80%93Jones\_object\_detection\_framework. Last accessed Feb 20th 2016.

[10] Open CV Dev Team. (2016). *Face Recognizer.* Available: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec\_api.html. Last accessed Feb 21th 2016.

[11] sis26@drexel.edu. (Unknown). *Eigen Face Tutorial.* Available: http://www.pages.drexel.edu/~sis26/Eigenface%20Tutorial.htm. Last accessed Feb 27th.

[12] Emily Miller. (2015). *The Australian Government Has Started Using Facial Recognition Technology, And Hardly Anyone Is Talking About It.*Available: http://junkee.com/the-australian-government-has-started-using-facial-recognition-technology-and-hardly-anyone-is-talking-about-it/67366. Last accessed March 8th.

[13] BioID Group. (2016). *Biometric Solutions.* Available: https://www.bioid.com/Solutions. Last accessed March 8th.

[14] Gang Pan, Zhaohui Wu and Lin Sun (2008). Liveness Detection for Face Recognition, Recent Advances in Face Recognition, Kresimir Delac, Mislav Grgic and Marian Stewart Bartlett (Ed.), ISBN: 978-953-7619-34-3, InTech, Available from: http://www.intechopen.com/books/recent\_advances\_in\_face\_recognition/liveness\_detection\_for\_face\_recognition

[15] Cognitec Systems Group. (2015). *Research.* Available: http://www.cognitec.com/research.html. Last accessed March 23rd 2016.

[16] Stephanie Schuckers, Larry Hornak, Tim Norman, Reza Derakhshani, Sujan Parthasaradhi. (2015). *Liveness Detection For Face Recognition.*Available: http://www.biometrics.org/bc2002/2\_bc0130\_DerakhshabiBrief.pdf. Last accessed March 9th 2016.

[17] Sachin Sudhakar Farfade, Mohammad Saberian, Li-Jia Li. (2015). *Multi-view Face Detection Using Deep Convolutional Neural Networks.*Available: http://arxiv.org/pdf/1502.02766v3.pdf. Last accessed March 15th.

[18] Joe Belfiore. (2015). *Making Windows 10 More Personal and More Secure with Windows Hello.* Available: https://blogs.windows.com/windowsexperience/2015/03/17/making-windows-10-more-personal-and-more-secure-with-windows-hello/. Last accessed March 23rd 2016.

[19] Michael Cooney. (2013). *US intelligence wants to radically advance facial recognition software.* Available: http://www.networkworld.com/article/2225788/applications/us-intelligence-wants-to-radically-advance-facial-recognition-software.html. Last accessed March 30th 2016.

[20] Brenda Salinas. (2013). *High-End Stores Use Facial Recognition Tools To Spot VIPs.* Available: http://www.npr.org/sections/alltechconsidered/2013/07/21/203273764/high-end-stores-use-facial-recognition-tools-to-spot-vips. Last accessed March 15th.

[21] Black Hat Group. (2016). *Black Hat Homepage.* Available: https://www.blackhat.com/. Last accessed April 10th 2016.

[22] Def Con Dev Group. (2016). *Def Con Hompage.* Available: https://www.defcon.org/. Last accessed April 15th 2016.

[23] Marcus Colon. (2012). *Black Hat: Biometric experts demonstrate reverse-engineering capability in iris scanning systems.* Available: http://www.scmagazineuk.com/black-hat-biometric-experts-demonstrate-reverse-engineering-capability-in-iris-scanning-systems/article/252153/. Last accessed April 17th 2016.

[24] Banking tech. (2011). *Biometrics: the case for convenience.* Available: http://www.bankingtech.com/47982/Biometrics-the-case-for-convenience/. Last accessed Feb27th 2016.

[25] wikipedia. (2016). *Automatic number plate recognition.* Available: https://en.wikipedia.org/wiki/Automatic\_number\_plate\_recognition. Last accessed April 15th 2016.

[26] Wikipedia. (2016). *Facial recognition system.* Available: https://en.wikipedia.org/wiki/Facial\_recognition\_system. Last accessed April 17th.

[27] Apache Software Foundation. (2016). *Commons Math: The Apache Commons Mathematics Library.* Available: https://commons.apache.org/proper/commons-math/. Last accessed April 17th 2016.

[28] Jama Dev Group. (2012). *JAMA : A Java Matrix Package.* Available: http://math.nist.gov/javanumerics/jama/. Last accessed April 20th 2016.

[29] AT&T Laboratories Cambridge. (2002). *The Database of Faces.*Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. Last accessed April 20th 2016.

[30] Dr Chi Ho Chan. (2016). *xm2vts database.* Available: http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/. Last accessed March 2nd.

[31] UCSD Group. (2016). *Yale Database.* Available: http://vision.ucsd.edu/content/yale-face-database. Last accessed March 2nd.

[32] Simon Lauder. (2015). *The Capability: Government's national facial recognition plan raises privacy concerns.* Available: http://www.abc.net.au/news/2015-12-17/government's-facial-recognition-system-sparks-privacy-concerns/7035980. Last accessed April 20th 2016.

## Appendices

### Appendix 1

**Project Initiation Document**

A Face Authentication System

**Author: Thomas Bedford**

**Module: SE3IP11-15-6A**

**Project ID: A-FACE**

Date: 9/10/2015 10:30am

Document Version: 1.0

The current status of this document is: Draft

|  |  |  |  |
| --- | --- | --- | --- |
| Primary Supervisor | Hong Wei | | |
| Secondary supervisor | James Ferryman | | |
| Supervisor’s check: For each item below the supervisor should circle ✓ if it is satisfactory for this stage of the project; if it is not satisfactory then the supervisor should circle 🗶 and add a suitable comment to indicate the deficiency. The supervisor should then sign below to confirm the ✓,🗶 and comments. | | | |
| **CHECKLIST** |  | Comments, concerns and recommendations | |
| Background | **✓ 🗶** |  | |
| Problem statement | **✓ 🗶** |  | |
| Technical Products | **✓ 🗶** |  | |
| Crosscheck | **✓ 🗶** |  | |
| Purchases | **✓ 🗶** |  | |
| Health & Safety | **✓ 🗶** |  | |
| Social, Legal & Ethical | **✓ 🗶** |  | |
| References | **✓ 🗶** |  | |
| Project Plan | **✓ 🗶** |  | |
| **Supervisor’s Signature** |  | | **Date:** |
| **Submission rules: The PID, with this form completed and signed by the supervisor, must be handed in to the student information centre (G47) AND submitted online on Blackboard by Friday 9th October 2015. If the deadline is not met, the student will face a penalty of 5 marks being deducted from their overall module mark. It is expected that students will address any comments noted by their supervisor on this form in due course.** | | | |

# Introduction

This document initialises the project originally mandated in [1]. It clarifies *what* is expected and *how* it is to be done. As in any project initiation there may be uncertainties and they are identified where possible using the term TBD – to be determined.

The background motivating this project is summarised below in §2. A concise statement of the objectives of this project is presented in §3. How these objectives are expected to be satisfied is summarised with a headline list of *products*[[1]](#footnote-1) listed in §4.

The project is undertaken under the auspices of the final year project module SE3IP11-15-6A and this mandates that the examination products listed in §11 are delivered as well as the technical products listed in §4.

An assurance that the intended technical products are likely to be adequate is given in §5 where the objectives and technical products are crosschecked. §12 provides an outline plan and shows the plausibility that delivering the products (both technical and examination) is feasible.

The purchases expected in the plan are listed in §6. Mandatory safety, social, legal and ethical concerns are addressed in §7 and §8 respectively. All references referred to in this document are listed in §10.

# Background

Technological advances striving to replicate human processes has resulted in biometric analysis being a focused area of research over the past twenty years. Biometric analysis of fingerprints, palm, eyes and faces have been a concentrated area of research. Humans perform face recognition on a daily basis effortlessly. Face recognition is one of the most non-intrusive methods of biometric analysis which makes it one of the most relevant applications.

With so many applications such as video surveillance, human computer interaction and law enforcement biometric analysis is a focused area of research. The main issues regarding computing face recognition is the subject’s posture, age and lighting which restricts the usage of such systems to a controlled environment. With technological advances in hardware we draw closer to more efficient and reliable face authentication systems.

# Problem Statement

The proposed project has been analysed to extract any assumption §3.3 that can be made about the project. The constraints of which this project will be completed in have been considered in §3.2.

On the basis of the soundness of the following assumptions, §3.3, it is deemed worthwhile to satisfy these objectives, §3.1, by delivering technical products subject to these constraints, §3.2, on how they may be developed and/or delivered.

## Objectives

1) Develop a graphical user interface to capture and display images.

2) Capture images from a camera device and save images as graphic files.

3) Implement Face detection algorithms on camera feed.

4) Implement Face recognition algorithms on the camera feed.

5) Integrate face data base into face authentication system.

6) Display recognition results displaying rejection and acceptance rates.

## Constraints

1) The system will have to be in a controlled environment i.e good illumination.

2) Hardware such as camera resolution will dictate the quality of analysis results.

3) Faces captured for recognition should be of a neutral gesture and posture.

4) Faces captured for recognition will be frontal facing or within 20 degree angle.

5) Project deadlines and delivery date.

## Assumptions

1) Experiments will be carried out in a suitable environment i.e good lighting conditions, no foreign objects between the subject and camera.

2) Images of faces stored in database will be frontal facing. Images captured for authentication will be frontal facing.

3) System subjects will not be wearing obstructive facial items i.e. sunglasses, burka, head scarf.

4) Participants using the system will be aware of the handling of sensitive data

# Technical Products

**Product 1:** Graphical User Interface

The graphical user interface will provide a method of interaction from the user to the system.

The user interface will display a camera feed and stored images for comparison. It will provide mechanisms for the user to customise and interact with the authentication system.

No risks are involved in the completion of this product.

**Product 2:** Capturing and storing images as graphical files.

This product will enable a camera feed to be displayed through the GUI. It will allow the user to capture and store an image as a graphical file.

Successful implementation of the camera feed into the GUI and the ability to capture and save graphical images with satisfy this products completion.

Exceeding memory limits of storage, hardware failures and violation of privacy are the rick factors of the completion of this product.

**Product 3:** Computing Face Detection

The detection software will locate and track the face of a user. It will compute and draw a rectangle encompassing the contour features of the face.

For this products completion the face detection software will correctly be able to locate and track a human user’s face. It will be able to illustrate through the camera feed the area of image that is being detected visible for the user.

Hardware failures and loss of code are key risks to the completion of the product.

**Product 4:** Computing Face Recognition

The face recognition software will be able to compare the user’s captured image with a previously stored image of the user.

The system should be able to successfully compare and recognise similarities of biometrical features within two images of a user’s face.

Risk to the products completion include a weak understanding of supporting libraries such as image processing libraries and bad time management.

**Product 5:** Integrate Face Database into system

The integration of an online database that stores images and details of users faces into the system.

The product must deliver the following tasks: Setup connections to an online database through the face authentication system. Some aspects of this products completion are TBD.

Key Risk to its completions are connection issues with the online database and the completion of the previous products.

**Product 6:** Display recognition results displaying rejection and acceptance rates.

The product will display acceptance and rejection rates and recognition results to the user interface.

Displaying the acceptance, rejection and recognition results will satisfy the completion of this product.

Risks to this products completion are bad design of code restricting the developer from extracting such figures and late delivery of previous products.

# Crosscheck of approach

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Product 1 | Product 2 | Product 3 | Product 4 | Product 5 | Product 6 |
| Objective 1 | X |  |  |  |  |  |
| Objective 2 |  | X |  |  |  |  |
| Objective 3 |  |  | X |  |  |  |
| Objective 4 |  |  |  | X |  |  |
| Objective 5 |  |  |  |  | X |  |
| Objective 6 |  |  |  |  |  | X |

The products in which this project have been broken down into §4 satisfy the objectives proposed §3.1. Each product provides a detailed solution to the tasks of each of the objectives stated in this document. Each product is associated with its own individual objective.

The time plan illustrated in the tables of §10.3 details the time allocated for each technical and examination product. The technical deliverables are scheduled to finish 2 weeks before the deadline of the final report examination product. This for extra testing, documentation and project conclusions. The deliverable products §4 have been designated adequate proportions of time §10.3 in which they can be completed and delivered in.

# Health and Safety

1. Computing for extensive amounts of time without regular breaks and unsupported seating position can cause eye strain, neck and back injuries and muscle strains.
2. Using the software system in an inadequate environment i.e. outdoors when raining, a wet room.
3. Using the camera in a cluttered area with objects on the floor (trip hazard). The area in which the camera will be used needs to be cleared from any trip hazards as the user will focusing on the interface.

# Legal and Ethical

When using the face authentication system subjects need to be aware that the system will be handling there biometrics. Likewise participants need to be aware they are being monitored as uniformed used would be intrusive and a violation of privacy. Permission needs to be granted by the user before carrying out any analysis.

An individual’s biometrics is a unique set of data that needs to be handle extremely sensitively and securely. You can get a new bank card with ease but it’s much harder to replace your face.

The face authentication system proposed in this project should not be used within critical systems unless results can be guaranteed. There is no place for ambiguity in critical systems. For example the use of a recognition system that cannot guarantee reliable results in a critical system may falsely identify someone for a crime they did not commit. This can be damaging to a person reputation and the repercussions can be severe.

# Examination Products

The mandated examination products are as follows. Risks identified that may obstruct their completion are listed in [2] with cross-reference to this section.

**EX1. Name**: PID

* **Short Description**: The document that initiates this project and guides it thereafter.
* **Acceptance Criteria**: The PID must be written as a Microsoft Word or PDF document in compliance with the format and content instructions indicated in this document (the PID Master). All sections should be complete to a reasonable depth and quality, consistent with the stage of the project, that satisfies the technical judgement of the project supervisor and, if appropriate, the Company Partner. The sign offs on the cover page will be completed.

**EX2. Name**: Autumn Term Week 6 Logbook Check

* **Short Description**: A formal check of project progress based on the logbook.
* **Acceptance Criteria**: The Student Information Centre staff, and Project Co-Ordinator if necessary, must be convinced the project progress has been sufficient to date and that this progress has been adequately recorded in the logbook. The logbook must include a sign-off sheet that indicates that the logbook has been assessed by the supervisor on a weekly basis.

**EX3. Name:** Project Progress Review

* **Short Description**: A formal check of project progress based on a checklist form.
* **Acceptance Criteria**: The ‘Project Progress Review Individual Form’ must be completed by the student and supervisor and signed and submitted by the appropriate deadline.

**EX4. Name**: Spring Term Week 6 Demonstration

* **Short Description**: A formal presentation of the executable technical products of the project to the project supervisor and internal moderator.
* **Acceptance Criteria**: During Week 6 of Spring Term, the student must demonstrate their project to their supervisor and internal moderator. The demonstration should show that the project’s technical products are on track to be completed by the end of the project.

**EX5. Name**: Poster

* **Short Description**: A poster summarising a significant aspect of your work.
* **Acceptance Criteria**: A poster is constructed based on the template (TBD). It will be of a quality typical of an academic conference poster presentation. The content will introduce and explain some aspect, claim of other facet of the project that the student deems most interesting and which is agreed by the supervisor.

**EX6. Name**: SCARP Abstract

* **Short Description**: An abstract in a form typical of an academic conference.
* **Acceptance Criteria**: The abstract is written according to the academic template (TBD). It will summarise the project with a focus on its achievements with respect to the objectives and technical products and what was learned.

**EX7. Name**: SCARP Paper

* **Short Description**: A short paper in a form typical of an academic conference.
* **Acceptance Criteria**: The paper is written according to the academic template (TBD). It will describe the project with a focus on its achievements with respect to the objectives and technical products and what was learned.

**EX8. Name**: Final Report

* **Short Description**: The academic write up of the work achieved by the project in addressing the technical problem.
* **Acceptance Criteria**: The report shall be formatted according to provided rules (TBD). It shall have content assessable according to the criteria (TBD). It will include sections such as Introduction, Literature Survey, Problem Analysis, Solution Analysis, Implementation, Evaluation (detail TBD).

**EX9. Name**: Demonstration to Internal Examiners

* **Short Description**: A formal presentation of the executable technical products of the project.
* **Acceptance Criteria**: The working demonstration must be convincing to the examiners in terms of: showing a significant satisfaction of the objectives through the operation of the developed technical products; showing that the work is substantially the product of the student’s efforts.

**EX10. Name**: SCARP Presentation

* **Short Description**: A 10 minute presentation of your work in a conference setting.
* **Acceptance Criteria**: The paper is presented in a clear, coherent manner in the allotted time and plausible answers are given to such questions that are possible in the conference schedule (typically two or three questions).

**EX11. Name**: Project Archive

* **Short Description**: A CD containing all project documents.
* **Acceptance Criteria**: The CD must contain all the files and folders specified in the guidelines (TBD).

# References

[1] A Face Authentication system, A-FACE, [SE3IP11-15-6A: Individual Project (2015/16)](https://www.bb.reading.ac.uk/webapps/blackboard/execute/launcher?type=Course&id=_113324_1&url=)/Course Documents/[SE3IP11 List of Staff Proposed Projects 2015-16 v2.pdf](https://www.bb.reading.ac.uk/bbcswebdav/pid-2778826-dt-content-rid-4576387_2/xid-4576387_2), available from (https://www.bb.reading.ac.uk/webapps/blackboard/content/listContent.jsp?course\_id=\_113324\_1&content\_id=\_2570572\_1&mode=reset).

[2] RA1 Risk Assessment forms, [SE3IP11-15-6A: Individual Project (2015/16)](https://www.bb.reading.ac.uk/webapps/blackboard/execute/launcher?type=Course&id=_113324_1&url=)/Course Documents/[RISK\_ASSESSMENT\_FORM\_RA2.docx](https://www.bb.reading.ac.uk/bbcswebdav/pid-2570600-dt-content-rid-4564128_2/xid-4564128_2), available from (https://www.bb.reading.ac.uk/webapps/blackboard/content/listContent.jsp?course\_id=\_113324\_1&content\_id=\_2570572\_1&mode=reset).

[3] Wikipedia, Face Recognition System, available from (https://en.wikipedia.org/wiki/Facial\_recognition\_system), visited on [01/10/2015].

[4] Nova Next, The limits of face recognition, available from (http://www.pbs.org/wgbh/nova/next/tech/the-limits-of-facial-recognition/), visited on [28/09/2015].

[5] PFC-IonMarques.pdf, available from (www.ehu.eus/ccwintco/uploads/e/eb/PFC-IonMarques.pdf), visited on [03/10/2015]

[6] Face Rec Algorithms, available from (http://www.face-rec.org/algorithms/), visited on [31/09/2015].

# Project Plan

## Technical Products

Table 12‑1 Technical Products

|  |  |  |  |
| --- | --- | --- | --- |
| Product Delivered | Task  ID | Task Description | Effort  (weeks) |
|  | TP1.1 | Specify, Design | 1 |
|  | TP1.2 | Implement and test | 2 |
| 1 | TP1.3 |  |  |
|  | TP2.1 | Display camera feed in GUI | 1 |
|  | TP2.2 | Capture and save images as graphical files | 2 |
| 2 | TP2.3 |  |  |
|  | TP3.1 | Install and setup libraries and tools | 1 |
|  | TP3.2 | Compute face detection | 2 |
|  | TP3.3 |  |  |
|  | TP3.4 | Run and test face detection on stored images and camera feed | 2 |
| 3 | TP3.5 |  |  |
|  | TP4.1 | Install and setup libraries and tools |  |
|  | TP4.2 | Compute face recognition | 2 |
|  | TP4.3 |  |  |
|  | TP4.4 | Run and test face recognition on stored images and camera feed | 2 |
|  | TP4.5 |  |  |
| 4 | TP4.6 |  |  |
|  | TP5.1 | Setup and configure connections to database | 2 |
|  | TP5.2 | Test sets of data from online database in the authentication system | 3 |
|  | TP5.3 |  |  |
|  | TP5.4 |  |  |
| 5 | TP5.4 |  |  |
|  | TP6.1 | Extract and display acceptance and rejection rates and recognition statistics | 2 |
| 6 | TP6.2 |  |  |

## Examination Products

Table 12‑2 Examination Products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product Delivered | Task  ID | Task Description | Effort  (weeks) | Who |
|  | EX1.1 | Research and complete PID | 2 |  |
| 1 | EX1.2 |  |  | Hong Wei |
|  | EX2.1 | Write up log book and project progress review | 3 |  |
| 2 | EX2.2 |  |  | Hong Wei |
|  | EX3.1 | Complete progress report checklist | 3 |  |
|  | EX3.2 |  |  |  |
| 3 | EX3.3 |  |  | Hong Wei |
|  | EX4.1 | Project review demonstration of system progress | 3 |  |
| 4 | EX4.2 |  |  | Hong Wei |
|  | EX5.1 | Design and create poster | 1 | Hong Wei |
| 5 | EX6.1 | Write up scarp extract | 1 | Hong Wei |
| 6 | EX7.1 | Write up scarp paper | 1 | Hong Wei |
| 7 | EX8.1 | Write up final report | 2 |  |
|  | EX8.2 |  |  |  |
| 8 | EX9.1 | Preparation for final demo to examiners | 1 | Hong Wei |
| 9 | EX10.1 | Preparation for scarp presentation | 1 |  |
| 10 | EX11.1 | Archive project | 1 |  |
| 11 | EX11.2 |  |  | Hong Wei |

## Time Plan for the proposed Project work

**Plan for the proposed Project work**

**10.3.1**

Table 12‑3 Time Plan Technical Products

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Product / Product phase** | **START DATE: <enter the project start date here>** | | | | | | | | |  | | | | | | | |
| Pre-Term | AUTUMN TERM (weeks) | | | | | | Break | SPRING TERM (weeks) | | | | | | | Break | SUMMER TERM  (exams) |
| 1-2 | 3-4 | 5-6 | 7-8 | 9-10 | 11 | 1-2 | | 3-4 | 5-6 | 7-8 | 9-10 | 11 |  |
| Product 1: Graphical User Interface |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Product 2: Capturing and storing images as graphical files. |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Product 3: Computing Face Detection |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Product 4: Computing Face Recognition |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Product 5: Integrate Face Database into system |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Product 6: Display recognition results displaying rejection and acceptance rates. |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |

10.3.2

Table 12‑3 Time Plan Examination Products

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Product / Product phase** | **START DATE: <enter the project start date here>** | | | | | | | | |  | | | | | | | |
| Pre-Term | AUTUMN TERM (weeks) | | | | | | Break | SPRING TERM (weeks) | | | | | | | Break | SUMMER TERM  (exams) |
| 1-2 | 3-4 | 5-6 | 7-8 | 9-10 | 11 | 1-2 | | 3-4 | 5-6 | 7-8 | 9-10 | 11 |  |
| PID |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Autumn Term Week 6 Logbook Check |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Project Progress Review |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Spring Term Week 6 Demonstration |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Poster |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| SCARP Abstract |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| SCARP Paper |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Final Report |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Demonstration to Internal Examiners |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| SCARP Presentation |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| Project Archive CD |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |

### Appendix 2

package application;

import java.awt.image.BufferedImage;

import java.awt.image.DataBufferByte;

import java.awt.image.Raster;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.nio.charset.Charset;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

import org.opencv.core.Mat;

import org.opencv.core.MatOfByte;

import org.opencv.imgcodecs.Imgcodecs;

import Jama.Matrix;

import javafx.scene.image.Image;

import javafx.scene.image.PixelWriter;

import javafx.scene.image.WritableImage;

public class AppTools {

/\*\*

\* Converts a buffered image into 2d array

\* @param bufferdImage bi

\* @return double[][] returnArray

\*/

protected int[][] buffImg2array(BufferedImage bi) {

int cols = bi.getWidth();

int rows = bi.getHeight();

int returnArray[][] = new int[cols][rows];

for (int i = 0; i < cols; i++) {

for (int j = 0; j < rows; j++) {

returnArray[i][j] = bi.getRGB(i, j);

}

}

return returnArray;

}

protected WritableImage buffToWriteImage(BufferedImage bi) {

WritableImage wr = null;

if (bi != null) {

wr = new WritableImage(bi.getWidth(), bi.getHeight());

PixelWriter pw = wr.getPixelWriter();

for (int x = 0; x < bi.getWidth(); x++) {

for (int y = 0; y < bi.getHeight(); y++) {

pw.setArgb(x, y, bi.getRGB(x, y));

}

}

}

return wr;

}

// source:

// http://answers.opencv.org/question/10344/opencv-java-load-image-to-gui/

/\*\*

\* Converts an OpenCV Mat to buffered image

\* @param Mat m

\* @return The output can be assigned either to BufferedImage or to Image

\*/

protected BufferedImage Mat2BufferedImage(Mat m) {

int type = BufferedImage.TYPE\_BYTE\_GRAY;

if (m.channels() > 1) {

type = BufferedImage.TYPE\_3BYTE\_BGR;

}

int bufferSize = m.channels() \* m.cols() \* m.rows();

byte[] b = new byte[bufferSize];

m.get(0, 0, b); // get all the pixels

BufferedImage image = new BufferedImage(m.cols(), m.rows(), type);

final byte[] targetPixels = ((DataBufferByte) image.getRaster().getDataBuffer()).getData();

System.arraycopy(b, 0, targetPixels, 0, b.length);

return image;

}

/\*\*

\* Converts a buffered image to java Image

\* @param bufferdimage bi

\* @return Image newImage

\*/

protected WritableImage bufferedImg2Img(BufferedImage bi) {

WritableImage newImage = null;

if (bi != null) {

// create writable image with same width and height as buff image

newImage = new WritableImage(bi.getWidth(), bi.getHeight());

PixelWriter pixWrite = newImage.getPixelWriter();

for (int x = 0; x < bi.getWidth(); x++) {

for (int y = 0; y < bi.getHeight(); y++) {

// get pixel value at x and y co-ordinate

pixWrite.setArgb(x, y, bi.getRGB(x, y));

}

}

} else

System.out.println("buffered image is empty");

return newImage;

}

/\*\*

\* Converts an OpenCV Mat to java Image

\* @param Mat frame

\* @return new Image

\*/

protected Image Mat2Image(Mat frame) {

// temporary buffer

MatOfByte buffer = new MatOfByte();

// encode image frame into PNG format

Imgcodecs.imencode(".PNG", frame, buffer);

// build image from encoded buffered data

return new Image(new ByteArrayInputStream(buffer.toArray()));

}

protected double[][] squareNonMatrix(double[][] data) {

double[][] squaredNMResult = new double[data.length][data[0].length];

for (int i = 0; i < data.length; ++i)

for (int j = 0; j < data[i].length; ++j)

squaredNMResult[i][j] = data[i][j];

for (int i = 0; i < data.length; ++i) {

for (int j = 0; j < data[i].length; ++j) {

squaredNMResult[i][j] = squaredNMResult[i][j] \* squaredNMResult[i][j];

}

}

return squaredNMResult;

}

protected double[] matrixSubtract(double[] matrix1, double[] matrix2) {

double[] result = new double[matrix1.length];

for (int i = 0; i < result.length; ++i)

result[i] = matrix1[i] - matrix2[i];

return result;

}

protected double[][] subtractFromEachRow(double[][] data, double[] data2subtract) {

double[][] subtractMatResult = new double[data.length][data2subtract.length];

for (int i = 0; i < data.length; ++i)

for (int j = 0; j < data2subtract.length; ++j)

subtractMatResult[i][j] = data[i][j];

for (int i = 0; i < data.length; ++i) {

for (int j = 0; j < data2subtract.length; ++j) {

subtractMatResult[i][j] -= data2subtract[j];

}

}

return subtractMatResult;

}

protected String getMatrix2dFileString(double[][] input) {

List<List<Double>> list = new ArrayList<List<Double>>(input.length);

for (int i = 0; i < input.length; ++i) {

List<Double> list2 = new ArrayList<Double>(input[i].length);

for (int j = 0; j < input[i].length; ++j) {

list2.add(new Double(input[i][j]));

}

list.add(list2);

}

return getMatrix2dFileString(list);

}

protected String getMatrix2dFileString(List<List<Double>> matrix) {

StringBuffer str = new StringBuffer();

for (int i = 0; i < matrix.size(); ++i) {

for (int j = 0; j < matrix.get(i).size(); ++j) {

boolean isEnd = j == matrix.get(i).size() - 1;

str.append(String.format("%.5f%s", matrix.get(i).get(j), isEnd ? "" : ", "));

}

str.append(System.lineSeparator());

}

return str.toString();

}

protected double[] multiply1DMatrix(double[] weights, Matrix subMatrix) {

double[] returnVal = new double[subMatrix.getColumnDimension()];

for (int i = 0; i < returnVal.length; ++i) {

double[] column = getColumn(subMatrix, i);

double sum = 0;

for (int j = 0; j < column.length; ++j)

sum += weights[j] \* column[j];

returnVal[i] = sum;

}

return returnVal;

}

protected double[] add1DMatrix(double[] m1, double[] m2) {

double[] returnVal = new double[m1.length];

for (int i = 0; i < returnVal.length; ++i)

returnVal[i] = m1[i] + m2[i];

return returnVal;

}

private double[] getColumn(Matrix m, int column) {

double[] col = new double[m.getRowDimension()];

for (int i = 0; i < m.getRowDimension(); ++i)

col[i] = m.get(i, column);

return col;

}

protected void print1dToFile(String fileName, double[] array) {

StringBuffer str = new StringBuffer();

for (int i = 0; i < array.length; ++i) {

boolean isEnd = i == array.length - 1;

str.append(String.format("%.5f%s", array[i], isEnd ? "" : ", "));

}

List<String> toWrite = new ArrayList<String>(1);

toWrite.add(str.toString());

try {

Files.write(Paths.get(fileName), toWrite, Charset.forName("UTF-8"));

} catch (IOException e) {

e.printStackTrace();

}

}

protected void print2dArrayToFile(String filename, double[][] matrix) {

String fileContents = getMatrix2dFileString(matrix);

List<String> toWrite = new ArrayList<String>(1);

toWrite.add(fileContents);

try {

Files.write(Paths.get(filename), toWrite, Charset.forName("UTF-8"));

} catch (IOException e) {

e.printStackTrace();

}

}

protected void print2dListToFile(String filename, List<List<Double>> matrix) {

String fileContents = getMatrix2dFileString(matrix);

List<String> toWrite = new ArrayList<String>(1);

toWrite.add(fileContents);

try {

Files.write(Paths.get(filename), toWrite, Charset.forName("UTF-8"));

} catch (IOException e) {

e.printStackTrace();

}

}

protected double[][] bufferedImageTo2DArray(BufferedImage bi) {

Raster raster = bi.getData();

int width = raster.getWidth();

int height = raster.getHeight();

double[][] returnArray = new double[width][height];

for (int i = 0; i < width; ++i) {

for (int j = 0; j < height; ++j) {

returnArray[i][j] = raster.getSample(i, j, 0);

}

}

return returnArray;

}

protected int getMinValue(int[] data2) {

int minVal = Integer.MAX\_VALUE;

for (int i = 0; i < data2.length; ++i) {

minVal = Math.min(minVal, data2[i]);

}

return minVal;

}

protected int getMaxValue(int[] data) {

int maxVal = Integer.MIN\_VALUE;

for (int i = 0; i < data.length; ++i)

maxVal = Math.max(maxVal, data[i]);

return maxVal;

}

protected double getMinValue(double[] data2) {

double minVal = Double.MAX\_VALUE;

for (int i = 0; i < data2.length; ++i) {

minVal = Math.min(minVal, data2[i]);

}

return minVal;

}

protected double getMaxValue(double[] data) {

double maxVal = Double.NEGATIVE\_INFINITY;

for (int i = 0; i < data.length; ++i)

maxVal = Math.max(maxVal, data[i]);

return maxVal;

}

}

1. A *product* in this context is something completed by the activity of the project. This is terminology of PRINCE2 project management. Some other PM regimes refer to *deliverables*. [↑](#footnote-ref-1)