FACE DETECTION & FACE RECOGNITION USING OPEN COMPUTER VISION CLASSIFIRES

LAHIRU DINALANKARA

August 4, 2017

Robotic Visual Perception and Autonomy
Faculty of Science and Engineering
Plymouth University

Abstract

Identifying a person with an image has been popularised through the mass media. However, it is less robust to fingerprint or retina scanning. This report describes the face detection and recognition mini-project undertaken for the visual perception and autonomy module at Plymouth university. It reports the technologies available in the Open-Computer-Vision (OpenCV) library and methodology to implement them using Python. For face detection, Haar-Cascades were used and for face recognition Eigenfaces, Fisherfaces and Local binary pattern histograms were used. The methodology is described including flow charts for each stage of the system. Next, the results are shown including plots and screen-shots followed by a discussion of encountered challenges. The report is concluded with the authors' opinion on the project and possible applications.

1 Introduction

The following document is a report on the mini project for Robotic visual perception and autonomy. It involved building a system for face detection and face recognition using several classifiers available in the open computer vision library(OpenCV). Face recognition is a non-invasive identification system and faster than other systems since multiple faces can be analysed at the same time. The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision "whose face is it ? ", using an image database. In this project both are accomplished using different techniques and are described below. The report begins with a brief history of face recognition. This is followed by the explanation of HAAR-cascades, Eigenface, Fisherface and Local binary pattern histogram (LBPH) algorithms. Next, the methodology and the results of the project are described. A discussion regarding the challenges and the resolutions are described. Finally, a conclusion is provided on the pros and cons of each algorithm and possible implementations.

2 The History of Face Recognition

Face recognition began as early as 1977 with the first automated system being introduced By Kanade using a feature vector of human faces [1]. In 1983, Sirovich and Kirby introduced the principal component analysis(PCA) for feature extraction [2]. Using PCA, Turk and Pentland Eigenface was developed in 1991 and is considered a major milestone in technology [3]. Local binary pattern analysis for texture recognition was introduced in 1994 and is improved upon for facial recognition later by incorporating Histograms(LBPH) [4], [5]. In 1996 Fisherface was developed using Linear discriminant analysis (LDA) for dimensional reduction and can identify faces in different illumination conditions, which was an issue in Eigenface method [6]. Viola and Jones introduced a face detection technique using HAAR cascades and ADABoost [7]. In 2007, A face recognition technique was developed by Naruniec and Skarbek using Gabor Jets that are similar to mammalian eyes [8], [9]. In This project, HAAR cascades are used for face detection and Eigenface, Fisherface and LBPH are used for face recognition.

3 Face Detection using Haar-Cascades

A Haar wavelet is a mathematical fiction that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognise signals with sudden transformations. An example is shown in figure 1. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are introduced [10] for object detection as shown in figure 1.

To analyse an image using Haar cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values pass a given threshold, it is considered a match. Face detection on a human face is performed by matching a combination of different Haar-like-features. For example, forehead, eyebrows and eyes contrast as well as the nose with eyes as shown below in figure A single classifier is not accurate enough. Several classifiers are combined as to provide an accurate face detection system as shown in the block diagram below in figure 3.

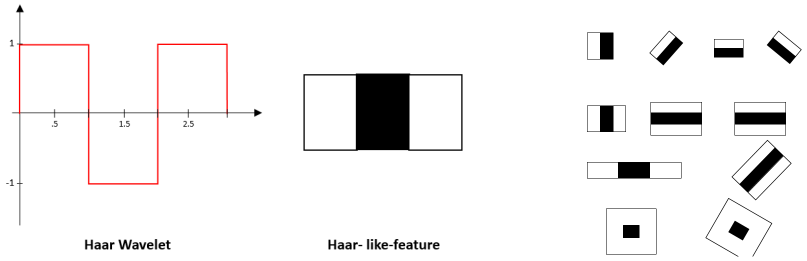


Figure 1: A Haar wavelet and resulting Haar-like features.

in this project, a similar method is used effectively to by identifying faces and eyes in combination resulting better face detection. Similarly, in viola Jones method [7], several classifies were combined to create stronger classifiers. ADA boost is a machine learning algorithm that tests out several week classifiers on a selected location and choose the most suitable [7]. It can also reverse the direction of the classifier and get better results if necessary [7]. 2. Furthermore, Weight-update-steps can be updated

Figure 2: Several Haar-like-features matched to the features of authors face.

only on misses to get better performance. The cascade is scaled by 1.25 and re-iterated in order to find different sized faces. Running the cascade on an image using conventional loops takes a large amount of computing power and time. Viola Jones [7] used a summed area table (an integral image) to compute the matches fast. First developed in 1984 [11], it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection. Using an integral image enables matching features with a single pass over the image.

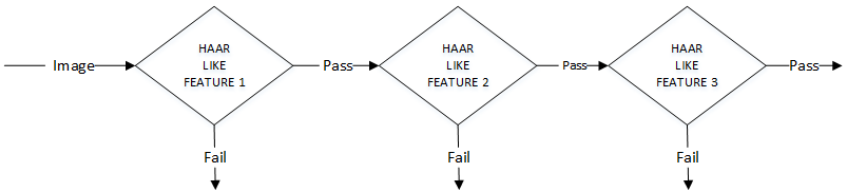


Figure 3: Haar-cascade flow chart

4 Face Recognition

The following sections describe the face recognition algorithms Eigenface, Fisherface, Local binary pattern histogram and how they are implemented in OpenCV.

4.1 Eigenface

Eigenface is based on PCA that classify images to extract features using a set of images. It is important that the images are in the same lighting condition and the eyes match in each image. Also, images used in this method must contain the same number of pixels and in grayscale. For this example, consider an image with n x n pixels as shown in figure 4. Each raw is concatenated to create a vector, resulting a $1 \times n^2$ matrix. All the images in the dataset are stored in a single matrix resulting a matrix with columns corresponding the number of images. The matrix is averaged (normalised) to get an average human face. By subtracting the average face from each image vector unique features to each face are computed. In the resulting matrix, each column is a representation of the difference each face has to the average human face. A simplified illustration can be seen in figure 4.

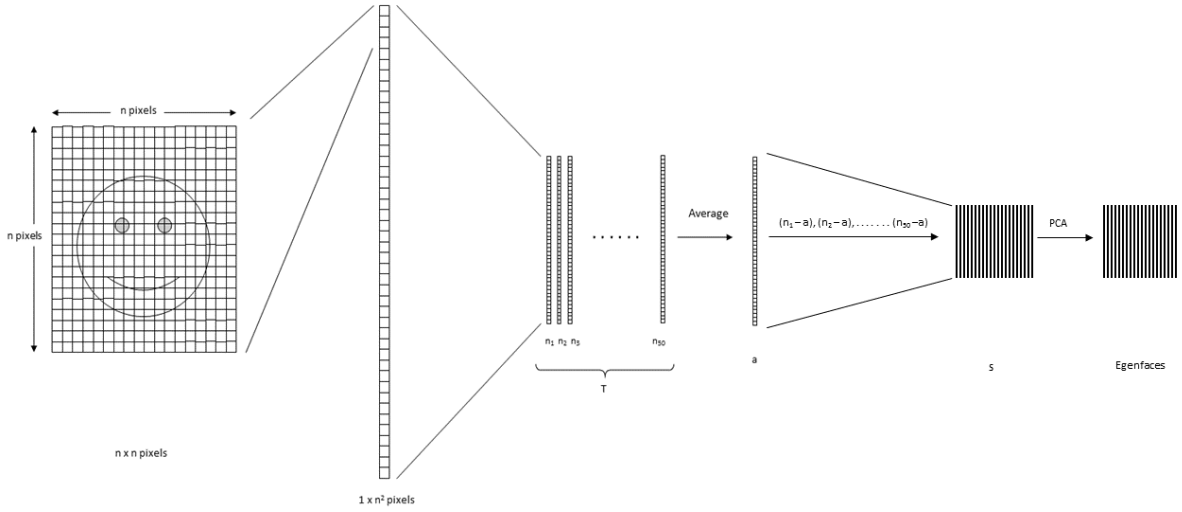


Figure 4: Pixels of the image are reordered to perform calculations for Eigenface

The next step is computing the covariance matrix from the result. To obtain the Eigen vectors from the data, Eigen analysis is performed using principal component analysis. From the result, where covariance matrix is diagonal, where it has the highest variance is considered the 1st Eigen vector. 2nd Eigen vector is the direction of the next highest variance, and it is in 90 degrees to the 1st vector. 3rd will be the next highest variation, and so on. Each column is considered an image and visualised, resembles a face and called Eigenfaces. When a face is required to be recognised, the image is imported, resized to match the same dimensions of the test data as mentioned above. By projecting extracted features on to each of the Eigenfaces, weights can be calculated. These weights correspond to the similarity of the features extracted from the different image sets in the dataset to the features extracted from the input image. The input image can be identified as a face by comparing with the whole dataset. By comparing with each subset, the image can be identified as to which person it belongs to. By applying a threshold detection and identification can be controlled to eliminate false detection and recognition. PCA is sensitive to large numbers and assumes that the subspace is linear. If the same face is analysed under different lighting conditions, it will mix the values when distribution is calculated and cannot be effectively classified. This makes to different lighting conditions poses a problem in matching the features as they can change dramatically.

4.2 Fisherface

Fisherface technique builds upon the Eigenface and is based on LDA derived from Ronald Fishers' linear discriminant technique used for pattern recognition. However, it uses labels for classes as well as data point information [6]. When reducing dimensions, PCA looks at the greatest variance, while LDA, using labels, looks at an interesting dimension such that, when you project to that dimension you maximise the difference between the mean of the classes normalised by their variance [6]. LDA maximises the ratio of the between-class scatter and within-class scatter matrices. Due to this, different lighting conditions in images has a limited effect on the classification process using LDA technique. Eigenface maximises the variations while Fisherface maximises the mean distance between and different classes and minimises variation within classes. This enables LDA to differentiate between feature classes better than PCA and can be observed in figure 5 [12]. Furthermore, it takes less amount of space and is the fastest algorithm in this project. Because of these PCA is more suitable for representation of a set of data while LDA is suitable for classification.

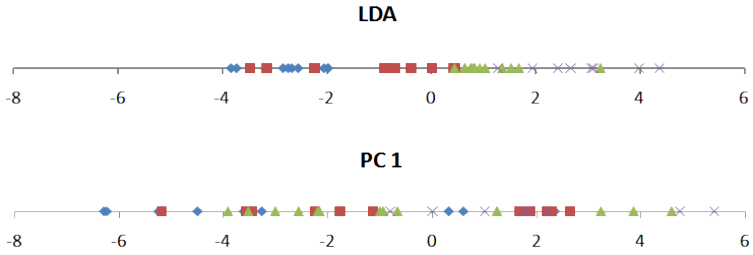


Figure 5: The first component of PCA and LDA. Classes in PCA looks more mixed than of LDA

4.3 Local Binary Pattern Histogram

Local binary patterns were proposed as classifiers in computer vision and in 1990 By Li Wang [4]. The combination of LBP with histogram oriented gradients was introduced in 2009 that increased its performance in certain datasets [5]. For feature encoding, the image is divided into cells (4 x 4 pixels). Using a clockwise or counter-clockwise direction surrounding pixel values are compared with the central as shown in figure 6. The value of intensity or luminosity of each neighbour is compared with the centre pixel. Depending if the difference is higher or lower than 0, a 1 or a 0 is assigned to the location. The result provides an 8-bit value to the cell. The advantage of this technique is even if the luminosity of the image

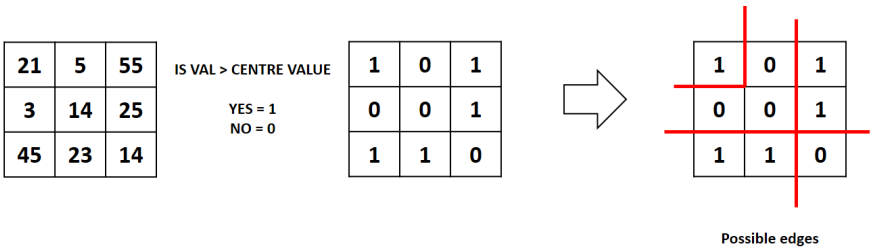


Figure 6: Local binary pattern histogram generating 8-bit number

is changed as in figure 7, the result is the same as before. Histograms are used in larger cells to find the frequency of occurrences of values making process faster. By analysing the results in the cell, edges can be detected as the values change. By computing the values of all cells and concatenating the histograms, feature vectors can be obtained. Images can be classified by processing with an ID attached. Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face. Eigenface and Fisherface compute the dominant features of the whole training set while LBPH analyse them individually.

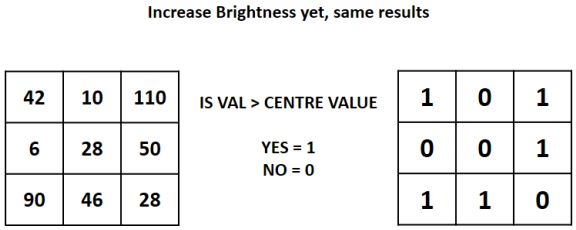


Figure 7: The results are same even if brightness is changed

5 Methodology

Below are the methodology and descriptions of the applications used for data gathering, face detection, training and face recognition. The project was coded in Python using a mixture of IDLE and PYCharm IDEs.

5.1 Face Detection

First stage was creating a face detection system using Haar-cascades. Although, training is required for creating new Haar-cascades, OpenCV has a robust set of Haar-cascades that was used for the project. Using face-cascades alone caused random objects to be identified and eye cascades were incorporated to obtain stable face detection. The flowchart of the detection system can be seen in figure 8. Face and eye

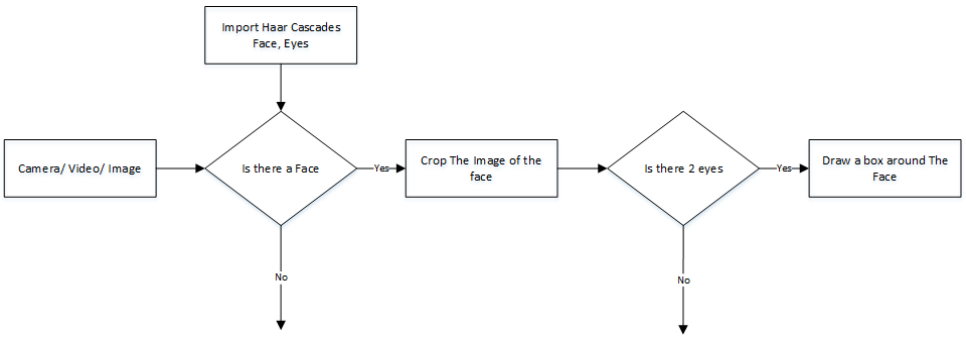


Figure 8: The Flow chart of the face detection application

classifier objects are created using classifier class in OpenCV through the **cv2.CascadeClassifier()** and loading the respective XML files. A camera object is created using the **cv2.VideoCapture()** to capture images. By using the **CascadeClassifier.detectMultiScale()** object of various sizes are matched and location is returned. Using the location data, the face is cropped for further verification. Eye cascade is used to verify there are two eyes in the cropped face. If satisfied a marker is placed around the face to illustrate a face is detected in the location.

5.2 Face Recognition Process

For this project three algorithms are implemented independently. These are Eigenface, Fisherface and Linear binary pattern histograms respectively. All three can be implemented using OpenCV libraries. There are three stages for the face recognition as follows:

1. Collecting images IDs
2. Extracting unique features, classifying them and storing in XML files
3. Matching features of an input image to the features in the saved XML files and predict identity

5.2.1 Collecting the image data

Collecting classification images is usually done manually using a photo editing software to crop and resize photos. Furthermore, PCA and LDA requires the same number of pixels in all the images for the correct operation. This time consuming and a laborious task is automated through an application to collect 50 images with different expressions. The application detects suitable expressions between 300ms, straightens any existing tilt and save them. The Flow chart for the application is shown in figure 9.

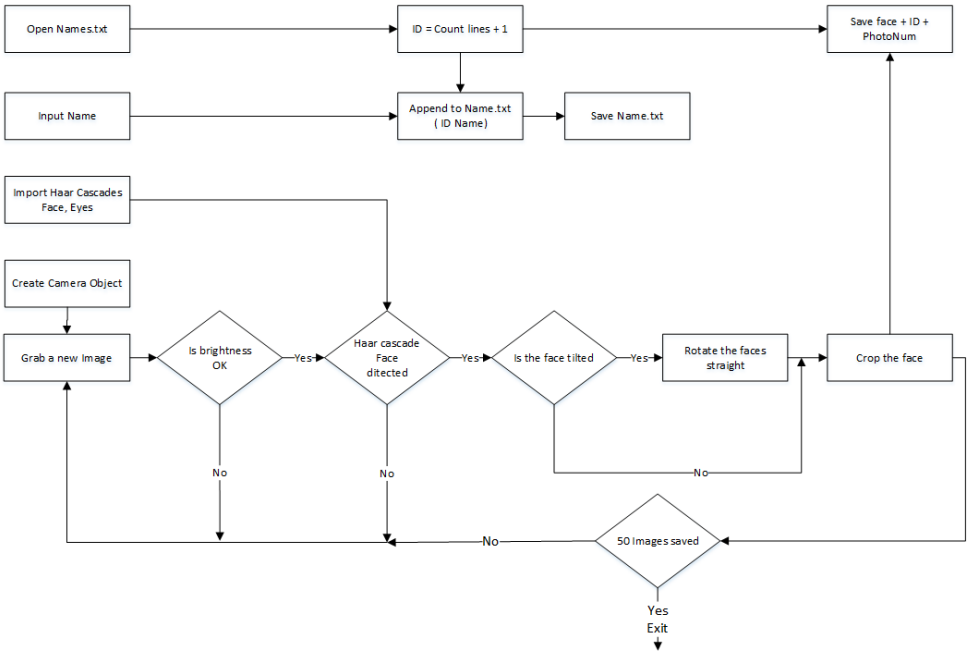


Figure 9: The Flowchart for the image collection

Application starts with a request for a name to be entered to be stored with the ID in a text file. The face detection system starts the first half. However, before the capturing begins, the application check for the brightness levels and will capture only if the face is well illuminated. Furthermore, after the face is detected, the position of the eyes are analysed. If the head is tilted, the application automatically corrects the orientation. These two additions were made considering the requirements for Eigenface algorithm. The Image is then cropped and saved using the ID as a filename to be identified later. A loop runs this program until 50 viable images are collected from the person. This application made data collection efficient.

5.2.2 Training the Classifiers

OpenCV enables the creation of XML files to store features extracted from datasets using the **FaceRecognizer** class. The stored images are imported, converted to grayscale and saved with IDs in two lists with same indexes. **FaceRecognizer** objects are created using face recogniser class. Each recogniser can take in parameters that are described below:

cv2.face.createEigenFaceRecognizer()

1. Takes in the number of components for the PCA for crating Eigenfaces. OpenCV documentation mentions 80 can provide satisfactory reconstruction capabilities.

2. Takes in the threshold in recognising faces. If the distance to the likeliest Eigenface is above this threshold, the function will return a -1, that can be used state the face is unrecognisable

cv2.face.createFisherfaceRecognizer()

1. The first argument is the number of components for the LDA for the creation of Fisherfaces. OpenCV mentions it to be kept 0 if uncertain.
2. Similar to Eigenface threshold. -1 if the threshold is passed.

cv2.face.createLBPHFaceRecognizer()

1. The radius from the centre pixel to build the local binary pattern.
2. The Number of sample points to build the pattern. Having a considerable number will slow down the computer.
3. The Number of Cells to be created in X axis.
4. The number of cells to be created in Y axis.
5. A threshold value similar to Eigenface and Fisherface. if the threshold is passed the object will return -1

Recogniser objects are created and images are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is gathered from splitting the file name, and stored in another vector. By using **FaceRecognizer.train(NumpyImage, ID)** all three of the objects are trained. It must be noted that resizing the images were required only for Eigenface and Fisherface, not for LBPH. Next, the configuration model is saved as a XML file using **FaceRecognizer.save(FileName)**. In this project, all three are trained and saved through one application for convenience. The flow chart for the trainer is shown in figure 10.

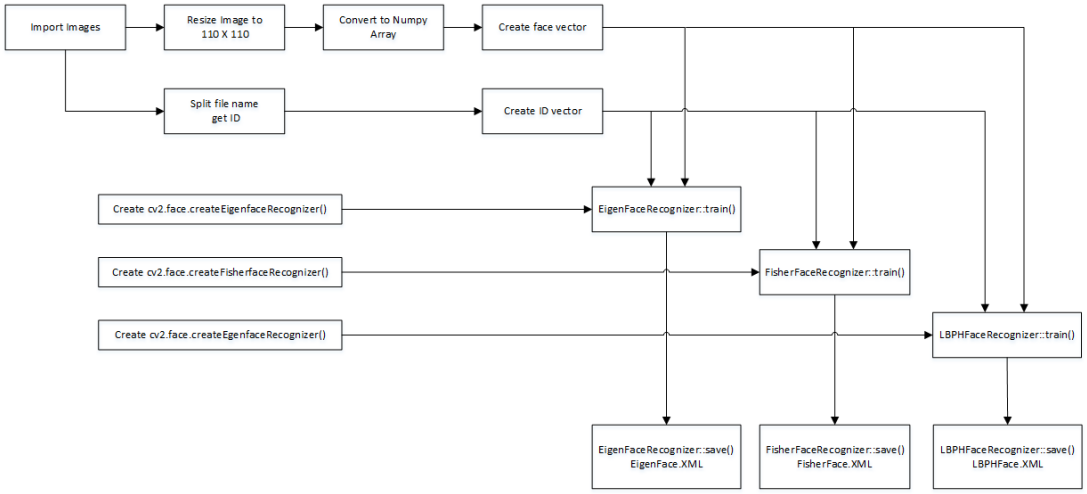


Figure 10: Flowchart of the training application

5.2.3 The Face Recognition

Face recogniser object is created using the desired parameters. Face detector is used to detect faces in the image, cropped and transferred to be recognised. This is done using the same technique used for the image capture application. For each face detected, a prediction is made using **FaceRecognizer.predict()** which return the ID of the class and confidence. The process is same for all algorithms and if the confidence his higher than the set threshold, ID is -1. Finally, names from the text file with IDs are used to display the name and confidence on the screen. If the ID is -1, the application will print unknown face without the confidence level. The flow chart for the application is shown in figure 11.

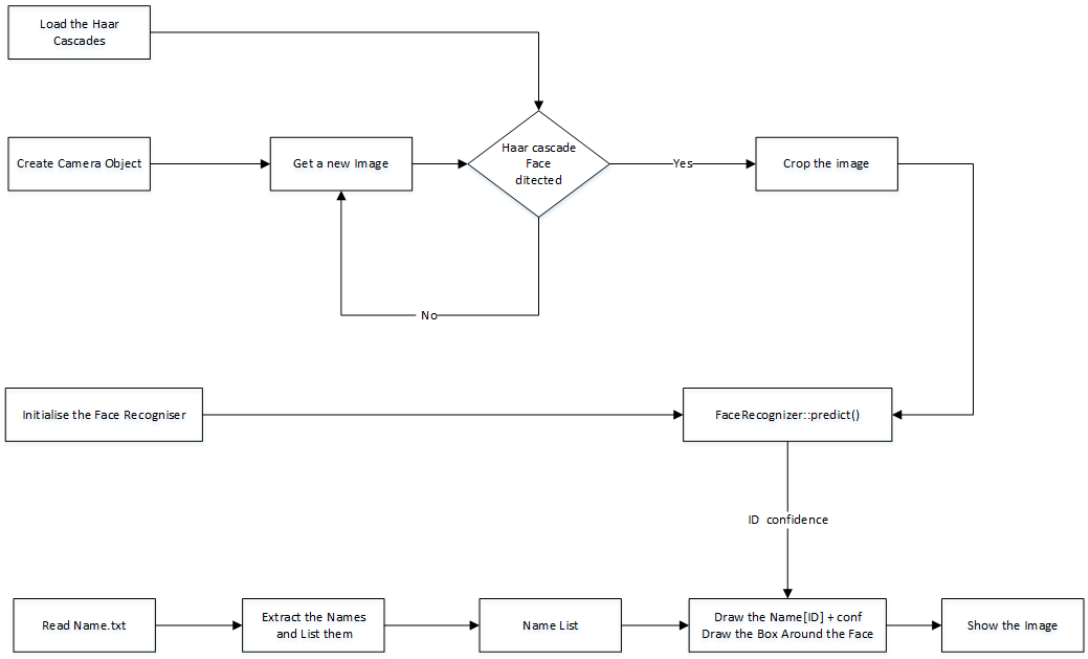


Figure 11: Flowchart of the face recognition application

6 Results

The collected images are shown below. Each face has 50 images. Three applications were written to iterate through the parameters of each algorithm. On each iteration, the algorithm is trained using different parameters and tested against a photo. The resulting data is plotted at the after finishing the tests. The applications are :

TestDataCollector_EigenFace.py

TestDataCollector_FisherFace.py

TestDataCollector_LBPH.py.

ID: 2 ID: 3 ID: 13 ID: 14 ID: 15 ID: 17 ID: 20 ID: 21

The first test image is shown in figure 12 and the plots are analysed below.
The resulting ID change is plotted below in figure 13. Note when components were 1, it identified

Figure 12: Image used for this test

the face as ID-17 and the rest are between ID-20 and ID-21, which is the same person. The change of

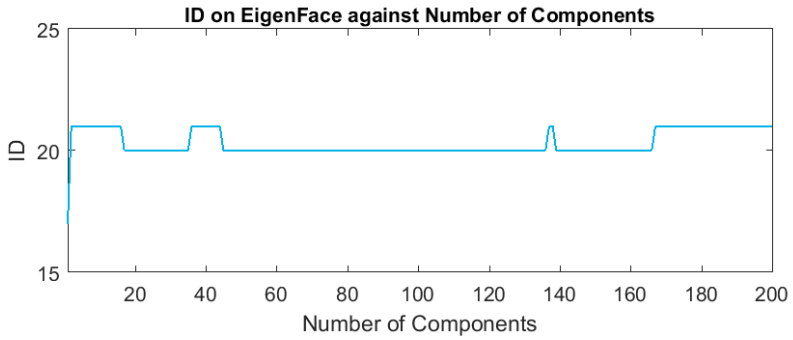


Figure 13: The ID from the face recogniser changes between two classes of the same person

Confidence is plotted in figure 14, increasing with components. From this plot it appears the best is when components are below 20.

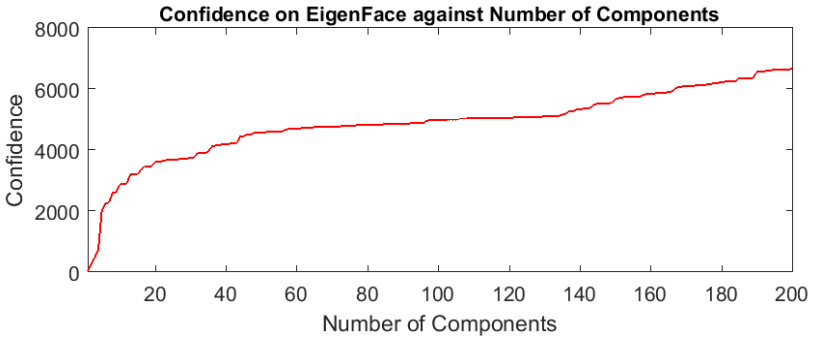


Figure 14: The Confidence increasing with No. of components

The ID results from Fisherface are more stable than Eigenface and is on ID-21 as seen in figure 15.

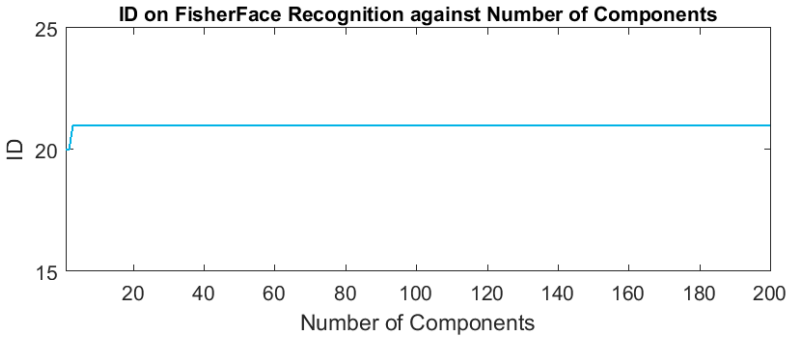


Figure 15: Fisherface ID is stable

Fisherface confidence increase in figure 16 until the number of components is 10 and will be used as the ideal value. LBPH has more than one parameter to change. All are incremented to the maximum

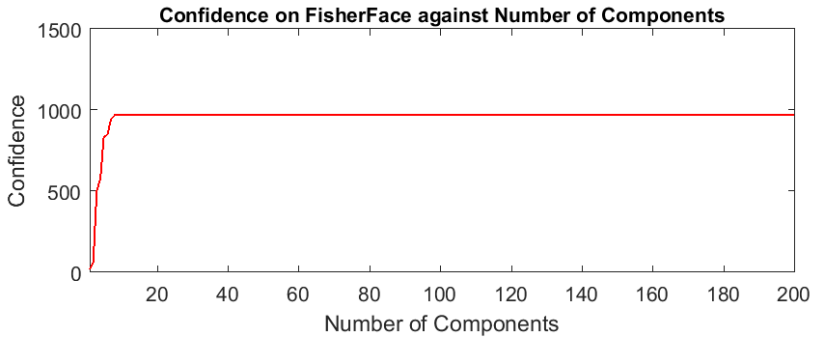


Figure 16: Stable confidence after 10 components

limit and the results are shown below. The first is the radius from the centre pixel and since the image size is 110 X 110, maximum radius is 54. The ID is steady all the way to 50 as can be seen in figure 17.

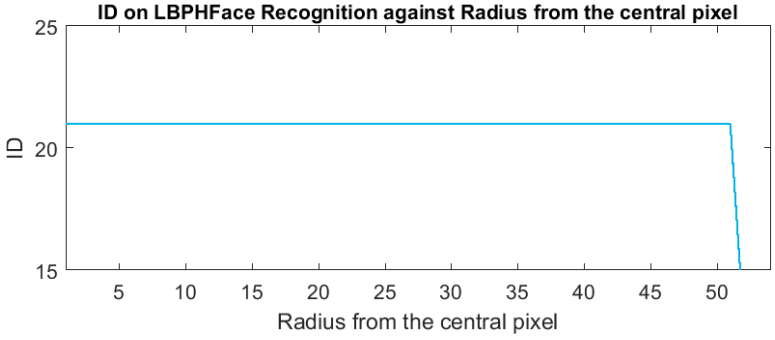


Figure 17: The ID returned from LBPH

Confidence level is graphed against the radius in figure 18. The confidence is fluctuating after 40. The lowest confidence level is at 2.

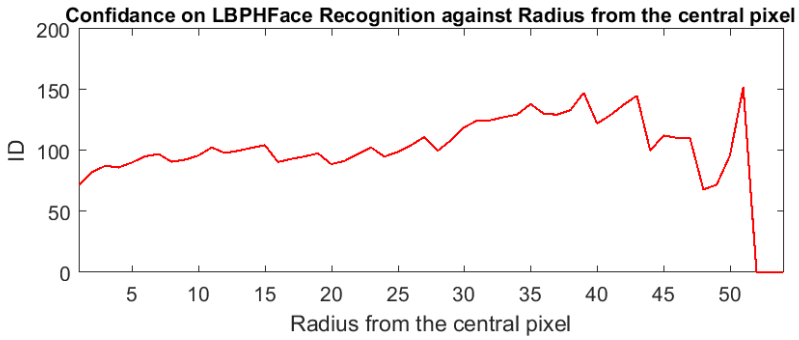


Figure 18: The confidence returned from LBPH

The number of neighbours are changed from 1 - 13. Further increase caused the computer to stall. The returned ID is plotted below in figure 19. ID steady until 9 neighbours and changed to ID-20.

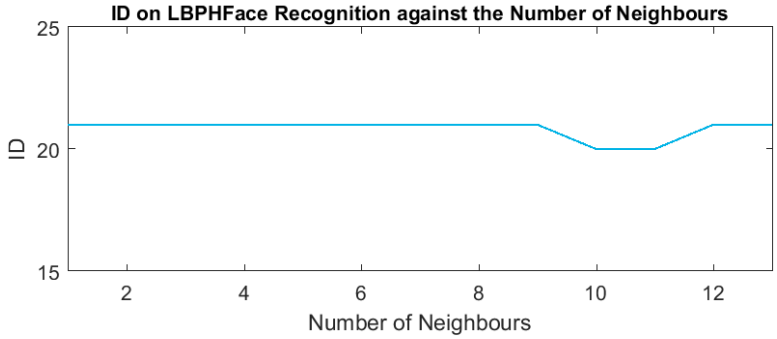


Figure 19: The ID returned from LBPH changing neighbours

The confidence continuously increased as can be seen in figure 20 and 1 neighbour will be included to the next test.

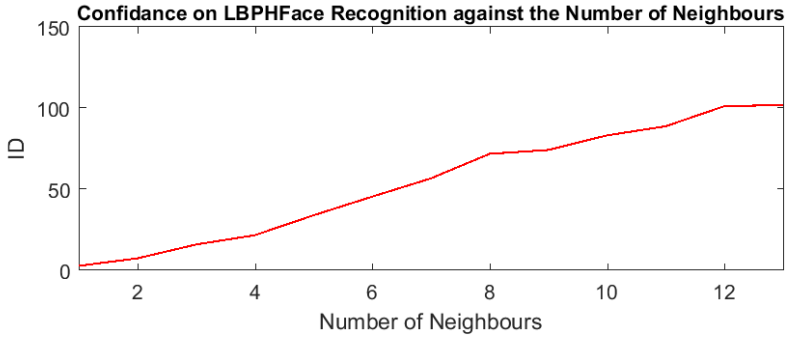


Figure 20: The Confidence returned from LBPH changing neighbours

The number of cells in X and Y directions are changed simultaneously . ID return is plotted below in figure 21. The ID changed from ID-20 to ID -21 and stay steady.

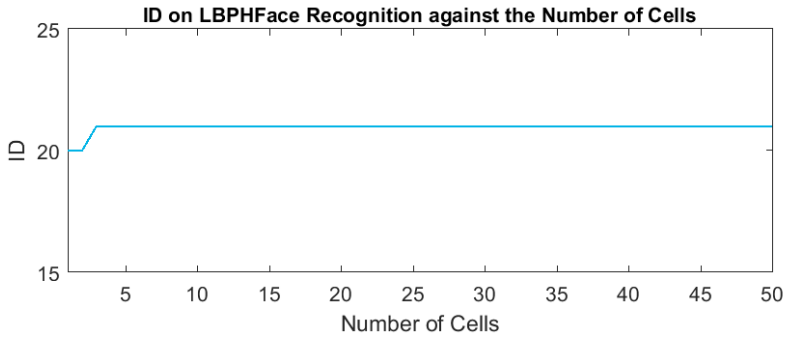


Figure 21: The ID returned from LBPH changing the number of cells

The returned confidence is plotted in figure 22. The confidence was low when cells were less than 8 per side and increased rapidly after 10 ending up more than 1700 at 50.

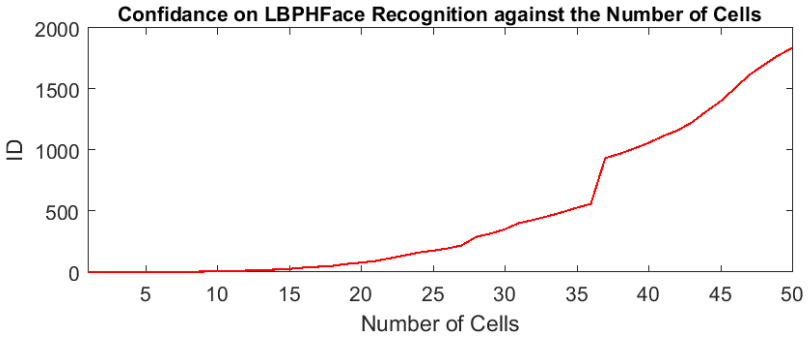


Figure 22: The confidence returned from LBPH changing the number of cells

6.1 Recognising a different subject

For a secondary test, authors' colleague provided a photo which was 7 years old. Although the photo has two people, plots only contain data on Mr. Westlake for clarity. The photo is shown below in figure 23. The idea was to observe if the program can identify a younger face from the data-set of an older face.

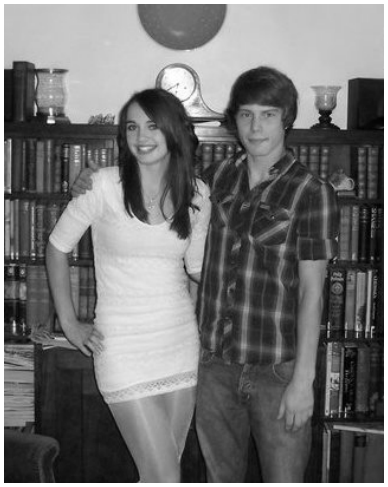


Figure 23: A photo of Samuel Westlake and his friend 7 years ago

The same program is used as above and plots for Eigenface, Fisherface and three separate pairs for the LBPH are below in figures 24 25 26 27 28.
Eigenface and Fisherface IDs and confidence was as expected. Note that Mr. Westlakes' ID is 17. The LBPH ID fluctuated when the pixels were above 38. Same can be noticed when cells are increased and low cells are unstable. The calibration details are detailed after the plots.

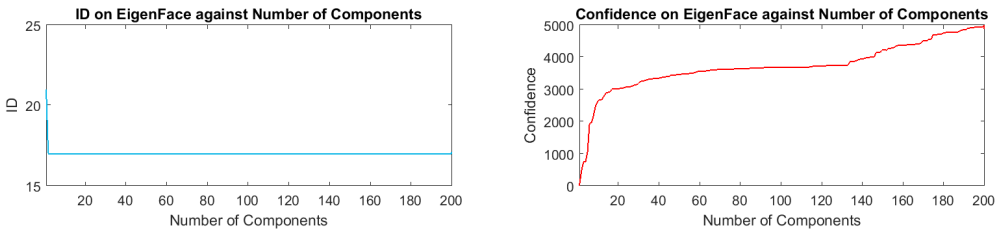


Figure 24: ID and Confidence for Eigenface. Note the ID is at 17 which is correct

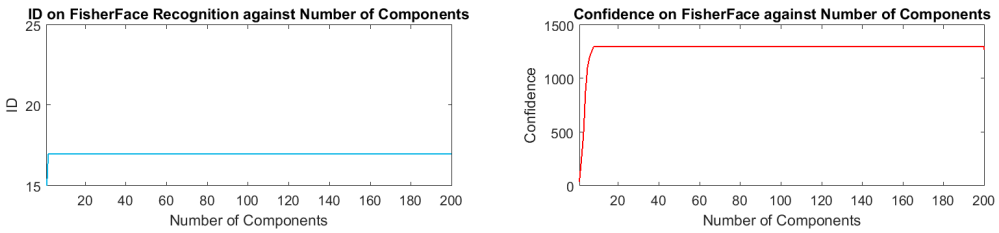


Figure 25: ID and Confidence for Fisherface. Note the ID is at 17 which is correct

The following plots show the data collected for LBPH parameters.

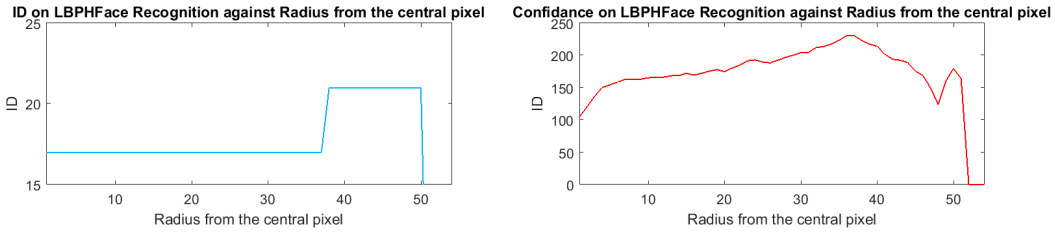


Figure 26: ID and Confidence for LBPA neighbours. Note the ID fluctuating at 37 pixels

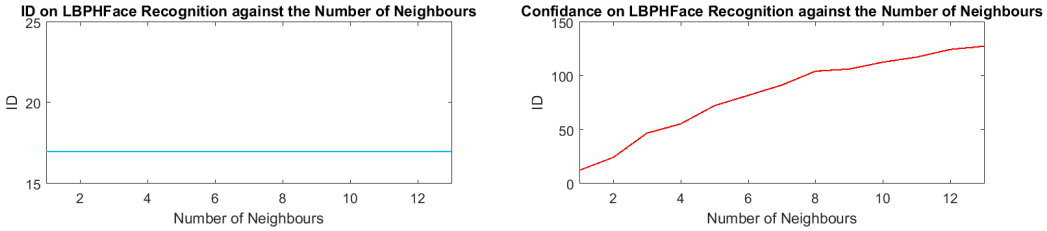


Figure 27: ID and Confidence for LBPA neighbours. Note the confidence increasing

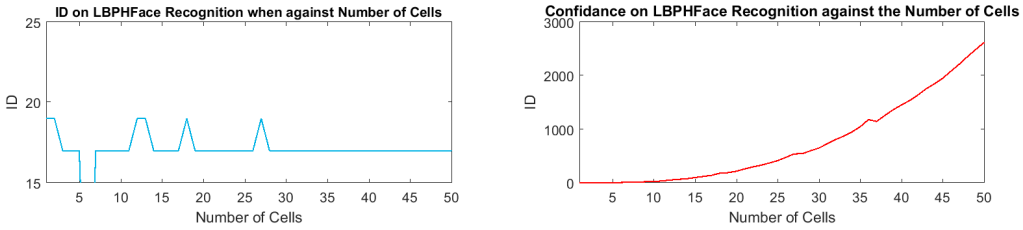
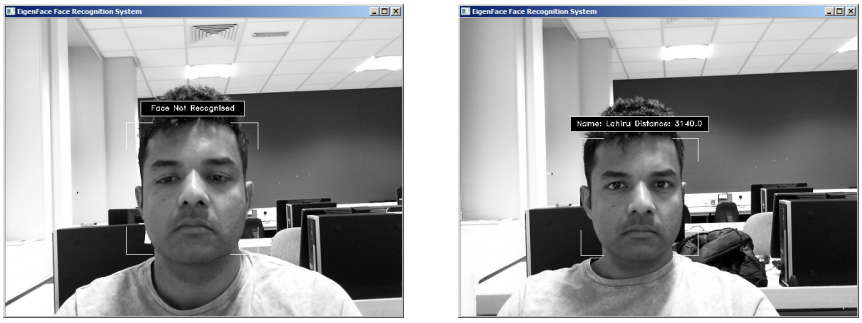


Figure 28: ID and Confidence for LBPA cells. Note the ID-17 is steady.

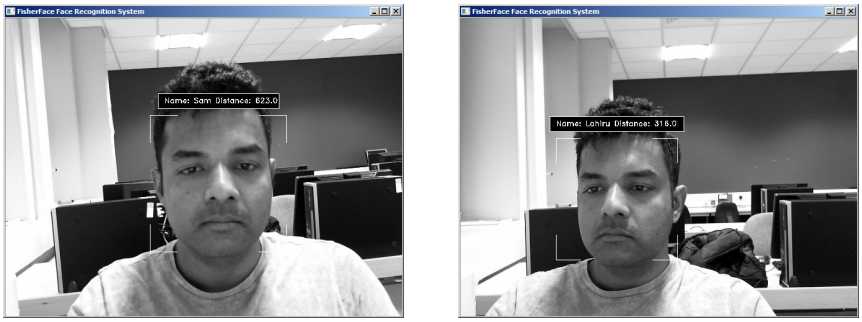
6.2 Calibrating the Trainer and Testing on Video

The Trainer (Trainer-All.py) application is used to train and recogniser (Recogniser-All.py) is used for testing the results. The image pairs shown below are with and without calibration of the trainer. All images are taken through video via Microsoft Lifecam camera. The values used are as followed:
Eigenface = number of components 15, Threshold 4000
Fisherface = number of components 5, Threshold 400
LBPH = Radius 2, Neighbours 2, Cells 7, Threshold 15



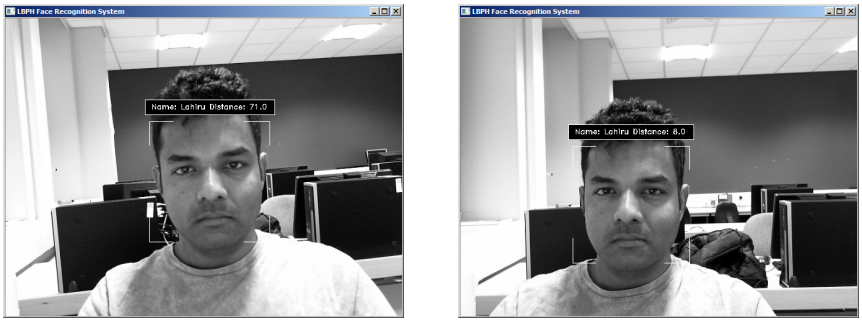
Before Calibration After Calibration

Figure 29: Before and after Eigenface Calibration threshold: 4000



Before Calibration After Calibration

Figure 30: Before and after Fisherface Calibration threshold: 400



Before Calibration After Calibration

Figure 31: Before and after LBPH face Calibration threshold: 15

Figure 32: After calibration of Eigenface the author and Mr. Westlake. Note both faces are detected successfully

The following image shows the system stating that the face is not recognised. Mr. Wallkoetters' face was not included at the time.

Figure 33: A face that is not in the data-set not recognised.

6.3 Discussion

Early attempts on Eigenface and Fisherface was disappointing since the LBPH alone recognised a face. By designing an application to test the algorithms, and after calibration with new data, both algorithms performed well. The tester applications also allowed accurate threshold settings. Another problem was people tilting their head when images taken for the data. This was fixed with an application that identifies the locations of the eyes and rotate the image to correct the off-set. It was noticed that some early-stage images in the data-set different brightness's. To resolve this, before taking an image, the brightness was averaged to prevent dark images. These changes to the system improved the performance noticeably.

7 Conclusion

This paper describes the mini-project for visual perception and autonomy module. Next, it explains the technologies used in the project and the methodology used. Finally, it shows the results, discuss the challenges and how they were resolved followed by a discussion. Using Haar-cascades for face detection worked extremely well even when subjects wore spectacles. Real time video speed was satisfactory as well devoid of noticeable frame lag. Considering all factors, LBPH combined with Haar-cascades can be implemented as a cost effective face recognition platform. An example is a system to identify known troublemakers in a mall or a supermarket to provide the owner a warning to keep him alert or for automatic attendance taking in a class.

References

[1] Takeo Kanade. *Computer recognition of human faces*, volume 47. Birkhäuser Basel, 1977.

[2] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

[3] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991.

[4] Dong chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):509–512, Jul 1990.

[5] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39, Sept 2009.

[6] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul 1997.

[7] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518 vol.1, 2001.

[8] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

[9] S Marĉelja. Mathematical description of the responses of simple cortical cells. *JOSA*, 70(11):1297–1300, 1980.

[10] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–I. IEEE, 2002.

[11] John P Lewis. Fast template matching. In *Vision interface*, volume 95, pages 15–19, 1995.

[12] Meng Xiao He and Helen Yang. Microarray dimension reduction, 2009.

View publication stats