Real-time Implementation of an Innovative Facial Database for Retail Customer Relationship Management (CRM) Systems

Xueqian Jiang Alex Li

Abstract

We built an automatic facial database in which a person's gender, facial characteristics and time of visit can be extracted and stored into the online database in real time. Such an application has the potential to increase in-store conversion rate in the retail sector. Based on this prototype system, further improvements could be made to develop a query system to push notifications to shop assistant's mobile devices, instructing them to make appropriate discounts on the right product, to the right customer, at the right time, based on the customers' visit and purchase history. The system is demonstrated to be a fast and storage efficient implementation, making use of a wide array of the popular methods in recognition, eigenface, fisherface and local binary pattern histogram. We compared, experimented, selected the best methods for the functions, and developed easy and effective mechanisms to achieve desirable system performance.

Section 1: Introduction

Recenlty, the offline retail sector has started to feel increasing pressure from online eCommerce. As the recent Forrester reports show, eCommerce will overtake the brick-and-mortar store by 2014 in revenue [1]. There is a need for more innovative ways for the traditional retailers to offer more and attractive products and promotions based on customers' preferences. The bottleneck is the low in-store conversion rate, and more, the inability to collect individual store visit – conversion data offline. We therefore propose a computer vision application that would extract the customers' biological information like gender, and store the facial features in the database in a compact way to identify, in real-time, the customer's visit pattern. The long-term goal would be (in future research) to use the database and real-time reporting, of an individual's purchasing behaviors to promote individualized purchases.

In our system, we would like to handle potentially tens and thousands of people's images to be stored as facial feature values and compared against in the image database. And therefore, the real-time operations would define the system to have fast retrieval.

<u>Pre-processing the images;</u>
The images were preprocessed to be 165*120 sizes and later reduced even further by Principal Component Analysis (PCA). Grayscale images were used to reduce 1 dimension from the original RGB images and also to make the images more invariant towards light;

<u>Tracking customers through reasonable estimate of speeds</u>
In our implementation, instead of using tracker algorithm to identify whether the person appeared in the video frame is the same one as before, we used reasonable estimation of shift in position, taking into account of the additional new faces into the view. With such a simple algorithm, the processing speed is fast, with tolerable level of error.

Pre-trained classifier like HaarCascade[2]
We decided to use the pre-trained classifier in Opencv for facial detection, to avoid wasting efforts / time on acquiring massive amount of positive and negative images for training. The exact implementation details utilize the following three things: convolutional kernel in which one subtracts the difference of the feature and non-feature windows, AdaBoost in which a combination of weak classifiers give strong results and finally a cascade of those classifiers to fast locate the face position.

Implementing the local binary pattern histogram in real-time facial recognition tasks[3]
The local binary histogram is produced by an algorithm, which takes the local features of an image and binarizes it by comparing one with its surrounding neighbors. The implementation is about 5 times faster than Eigenface or Fisherface implementations as it does not have to project into subspaces or build a model from there. Again, this does not require much storage and the extended version we implemented in our project allowed us to only take 8 points to represent the local features.

2.1 Review of previous work

Facial Detection
To detect faces serves as the first thing for facial recognition. Various ways to conduct facial detection within an image have been proposed. *Template-matching* [4-6] is used for face localization and detection by computing the correlations of an input image to a standard face pattern*, the feature invariant approaches* are used for feature detection [7-8]. The *appearance-based methods* are used for face detection with Eigenface [9-10] *neural network* [11-13] and *probabilistic Graph Matching* [14-17] that uses declarative representation of the information learned from the face and making decisions. Nevertheless, it is hard to find one overarching implementation good for all requirements.

The problem of facial detection and recognition has not been a new one since the 1980s. However, it is only until the 2000s that faster detection algorithms have emerged with Viola and Jones [18] invented the Haar-based cascade classifier for object detection and it was further improved in 2002 when Lienhard and Maydt invented extended Haar features and personally wrote the HaarCascade feature classifier we used in our project [19]. This largely sped up the facial detection processes. The recent systems with Haar-like features use the AdaBoost-based face detector by Viola and Jones demonstrated that faces can be fairly reliably detected in real-time i.e. more than 15 frames per second on 320 by 240 images with desktop computers) under partial occlusions [20]. Boosting is the method of combining the weak classifiers to form a strong classifier. In the later round of learning, AdaBoost finds the new weak classifier after re-weighing the training examples such that it emphasizes on the examples incorrectly classified by the previous weak classifier.

Facial Recognition
Facial Recognition is probably one of the most commonly used techniques in biometric applications. Though similar to the task requirements in facial detection, the recognition step uses more sophisticated algorithm and is largely dependent on light conditions. Several approaches have been proposed. For example,

- *Eigenfaces* takes the average of a mean face and then matches each and every face to the mean face before projecting all faces into a suitable subspaces [9-10];
- *neural network technologies[21]* extract features from the entire face as visual contrast units, and quantify and normalize them before they could be fed into a large network to be learned;
- *dynamic link architecture[22]* uses the methods to extend to the networks mentioned above, but it could memorize objects as sparse graphs and the edges gave information to the special location of the different features to be used;
- *automatic face processing[3]* makes use of the fact that faces could be marked for similarity score with the simple distance calculatedl
- and *Fisherface*, the last one and the most prevalent one[23], performs facial recognition task through maximizing the interclass distances through linear discriminant analysis (LDA) to achieve a good categorization result.

Fisherface projects well-separated classes in a low-dimensional subspace, even under severe variations in lighting and facial expressions. The Eigenface technique, however, is not fit for making predications based on large variances of inter-class differences. Eigenfaces have intrinsic limitation including high sensitivity to lighting conditions, expressions, camera angles and head poses. Moreover, extensive experimental results demonstrate that the proposed “Fisherfaces” method has error rates that are lower than those of the Eigenface technique for tests on the Harvard and Yale Face Database.[24]

The facial recognition tasks, in our project, are achieved using a combination of fisher, Eigen with a twist of local binary pattern histogram: recognize gender from a trained facial database through Eigen and Fisher implementations and then generate facial recognition by using compact histogram voting to represent the features within an image individually. In the first case, an efficient implementation with PCA+LDA [25] is used in this report and in the latter case, local binary pattern histogram with a tuned 8 bit (radius = 1,neighbor = 1 and grid space = 4)

2.2 Our Contribution

Our project focuses on innovation in practical application. The goal is to provide a fast, cheap and scalable system that can be used in real life. As a result, in the process, we researched, selected and developed various component solutions, while balancing the real world constraints on computing power, network traffic, cost and the need for accurate real time results. We tackled the following issues along the way:

Facial tracking:
The application project is intended for use in retail stores, where high foot traffic and constant unstructured human movements may be observed. We need to correctly identify a person’s initial visit time of one store visit, and differentiate that from the multiple appearance of the same person in consecutive frames. For example, hypothetically, person A walks into the store and is captured by the camera at 00:00:00, and walks out of the camera coverage range at 00:00:20, and walks back into the camera range at 00:00:22, and finally leaves the store at 00:00:24. The system should ideally record only 1 visit by person A at 00:00:00. To achieve this result, there could be two options:

- for each frame, we process the facial picture and report one visit event to the server. Server would check to see whether the same person’s multiple visit events are

within close timestamps. If two timestamps for the same person's two reported visits are apart by a small amount, below threshold, we delete the later record
- we track the face and only report a visit event when the face first appears in the series of consecutive frames. This solution would require comparing faces in two frames.

The first option would mean high frequency of data upload and backend server calculation. This could be costly, especially if we have hundreds of front end cameras uploading 20 visit records / second into the same database, and requesting comparison across all recent records to identify same person. The second option would mean less data traffic, however require more front end algorithm to track the same faces. A complicated facial comparison algorithm run every frame would significantly slow down the real time processing and would require computing power, implying higher cost. Thus, we need a simple, light, front-end algorithm to track faces.

The solution we ended up with includes two parts:

a) we reduced the # frames / second, to reduce # facial frames / visitor, so that we do not have too many near identical frames without any new information (20/sec to 1/sec, see next section)
b) we tested the following algorithm: if oldframe has no face and newframe has faces, we view these faces in the newframe as true new visits. If newframe has different number of faces than the oldframe, we try to find match among the faces based on relative positions of the centroids. Between two frames, if the centroids of the potential similar facial rectangles are close, we deem them as the same face. We observed satisfactory results from the algorithm, and thus adopted the solution. This would potentially yield errors and miss certain faces. However, the error would be reasonably tolerated, as long as the #frames/second is high enough.

Gender detection:
We have a few options to choose from for implementing gender detection. One decision is also on where to implement this – front-end at the camera, or back-end after the facial information has been extracted and uploaded. We reviewed the current gender detection methods / training sets available, and adopted the simple one that could be pre-trained. Given that we would like our system to be scalable (backend server handling hundreds / thousands of camera inputs at the same time), we tested the gender detection algorithm at the front end, and adopted it.

information representation of the facial images

We want to extract and store facial structures with the following criteria on the system:
- Fast and simple process of extracting data from images – the real time nature and front-end computing power limitation mean that we cannot use a complicated on-line training method locally at the front end
- Considering data storage cost, query cost and network traffic cost, we would like to minimize the amount of dimensions / size of data, representing each facial image. While the extreme case would be to store the entire image as a pixel-wise matrix / vector, we wanted to compress this as much as possible to use as few key values as possible.

We compared a few options, and eventually settled on the solution as described in the next section.

3.1: Technical part: Summary of the technical solution

Here is the layout of the system:

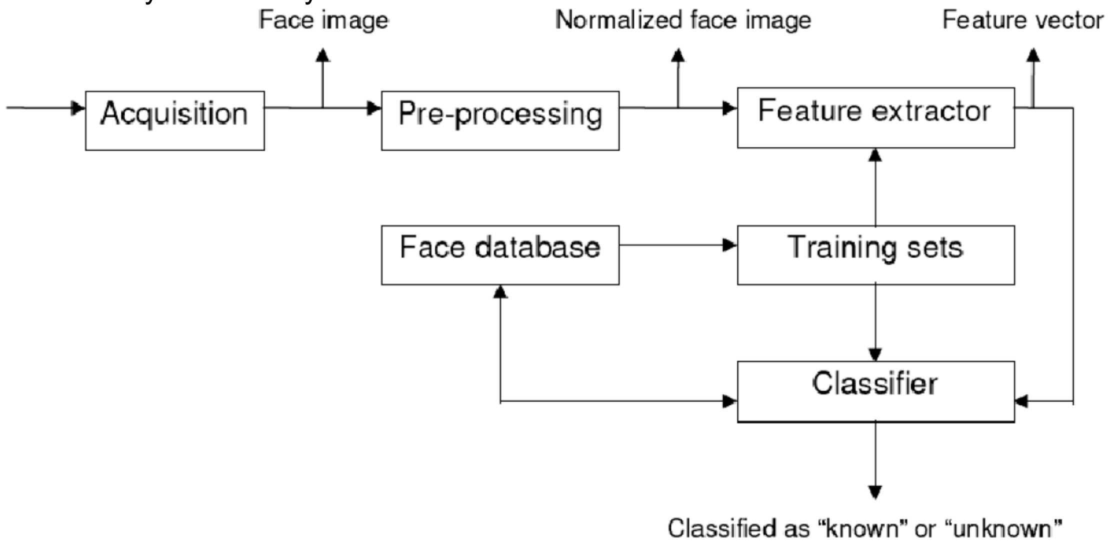


Figure 1: layout of the system

The Acquisition module
This part consists of a webcam (***C270 Logitech***) that is able to capture the videos in frames. We have tested that the camera is able to capture and detect faces within a parameter of 5m accurately.

The pre-processing module
Here many algorithms could be in place to improve the accuracy and the real-time performance of the system. In our current prototype, we have chosen to use *image size normalization function* by giving the images similar shades of gray level and similar sizes to the training set. Also video frames are coming in from the video stream with a speed of about 20 frames per second from the camera and down-sample it to be 1 frame per second to be displayed on the screen.

Human Tracking
to identify whether or not an incoming customer's information has been stored in the database within the 1 frame per second capture time, we implemented a simple human tracking algorithm to decide: if the person's general location has shifted about ¼ of the overall bounding boxes, the object is a new person. Otherwise, he is not. We also take into consideration a completely new face and a disappeared face in the frame. This is to avoid comparing faces each time at the backend through the whole database system.

The feature extraction module
All the facial extraction includes the AdaBoost algorithm. The algorithm is used to select a specified number of weak classifier with lower error rate for each cascade and the

process is repeated until a set of optimization criteria (i.e. the number of stages, the number of features of each stage and the detection/false positive rates) is satisfied.

The feature extracted is through the implementation of Voila Jones and the improved version in Lienhard's paper. The information for the feature extraction is demonstrated below:

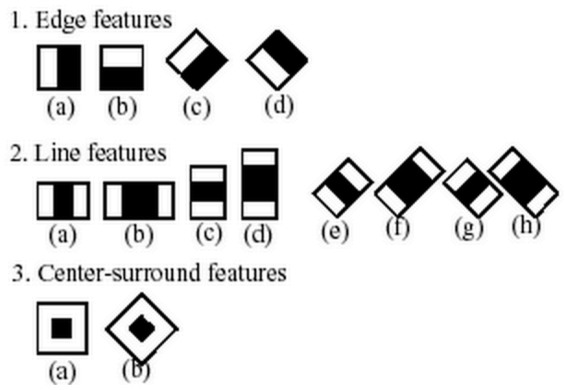


Figure 2: HaarCascade Feature Masks

Four edge features: 51664
Eight line features: 55292
Two Center-surrounded features: 9985

The classification module:
There are 2 classification tasks to be completed: gender recognition and identity recognition. For gender recognition, we used PCA+LDA, with the first 121 eigenvectors extracted and later on we conducted identity recognition with Local Binary Pattern Histogram(LBPH) with the distance function computed as the Chi-Squared distance and achieved high accuracy.

Feature computation could be achieved with the following method:

i. Compute a "distance" between the new one and each of the example faces;
ii. Select the example image that is closest to the new one as the most likely known person;
iii. If the distance to that face image is above a threshold, "recognize" the image as that person, otherwise, classify the face as an "unknown person".

Training set:
In our project, we specially requested the AR Face Database, which contains about 4000 faces categorized by female and male. Also the dataset has different conditions for each person to be tested, including frontal face, left light, right light, occlusion, moustache, etc. Also as the incoming picture is captured real-time, we are able to add the new faces into the database to be part of the training images.

Face database:
We implemented a very single backend with PHP and SQL. The database is hosted on HawkHost. The front-end algorithm would send each new facial / visit record to the server in json format. The PHP handler would store the visit info and update the facial profile database. For the next phase, we would like to have the backend conduct comparison among all profiles to determine whether the new visit record belongs to an

existing or new customer. We would also conduct test to eliminate duplicate visit records where timestamps are too close. Currently, the database conducts the basic functions by only storing the new visit / customer records (gender, timestamp, and facial representation). Also, theoretically, the database should store millions of facial features to be compared against in the future as mentioned above.

Section 4: Experiments:

Face Detection

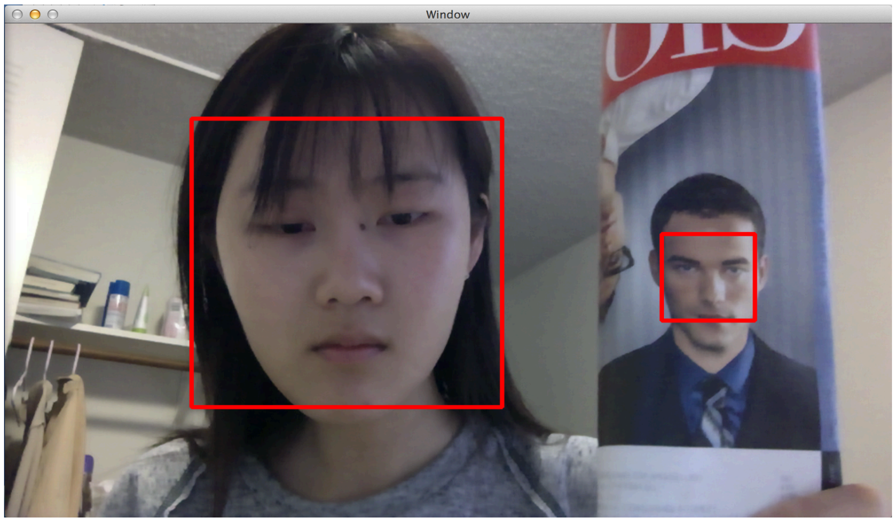
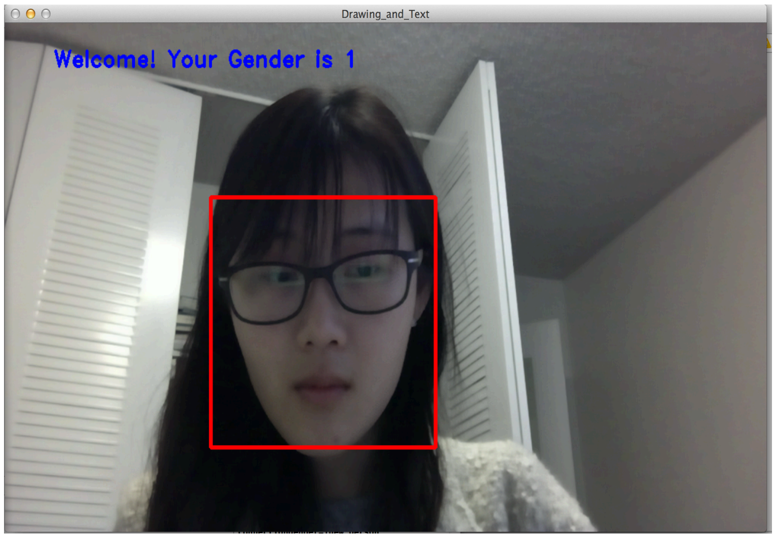


Figure 3: Facial Detection Unit at work

The Facial Detection Unit is able to detect many faces at the same time and store the faces in the temporary folder before batch-processing and sending to the backend.

Gender Recognition



Welcome! Your Gender is 1

Figure 4: Gender Recognition and Result (“1” represents female, whereas “0” will be male)

We used the AR Face Dataset[26] with 240 base images, half female and half male. And by running detection with PCA+LDA described above, we first passed the images through PCA and obtained the following Eigenfaces.

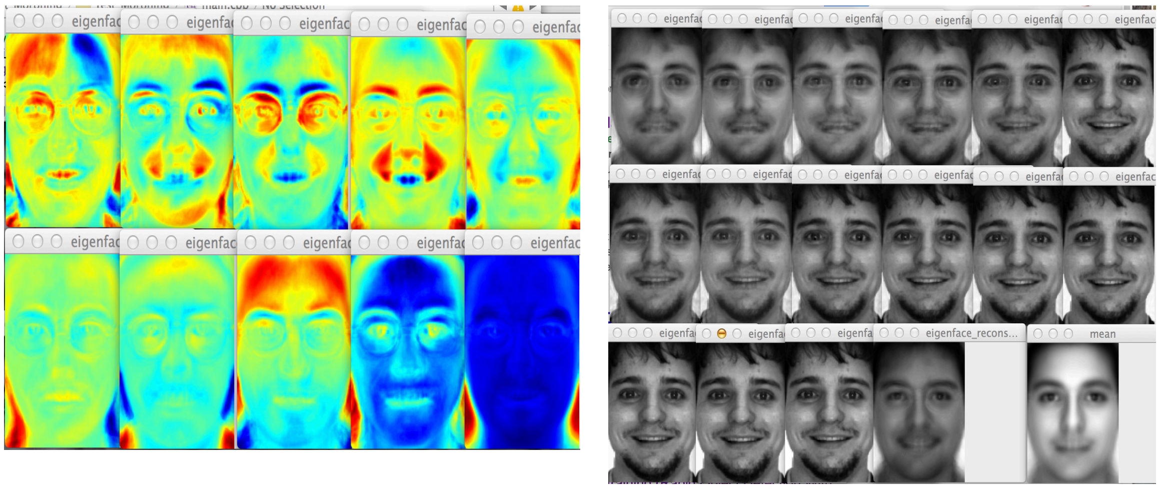


Figure 5: eigenface reconstruction and meanface.

We kept the first 121 EgienValues out from the 49500 eigenvalues to achieve a reduction of dimension of 97% and to show that PCA+LDA indeed performs better with out implementation, we compared the ROC curve for PCA, LDA methods with the full AR face dataset available with occlusion, different lighting conditions. In order to simulate extreme conditions in-store, we chose the training set to be a small 240 frontal faces with only mile facial feature changes. The test images are the full 2400 images comprising of 100 individuals under all conditions (50 male and 50 female) shown below.

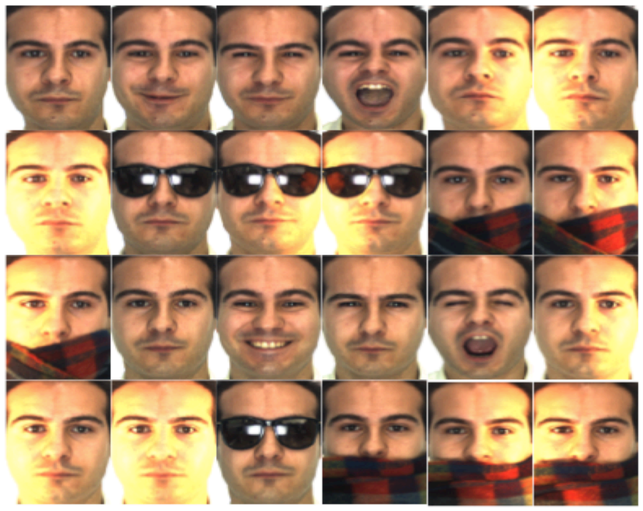


Figure 6: Test Images for One Individual Under Different Conditions

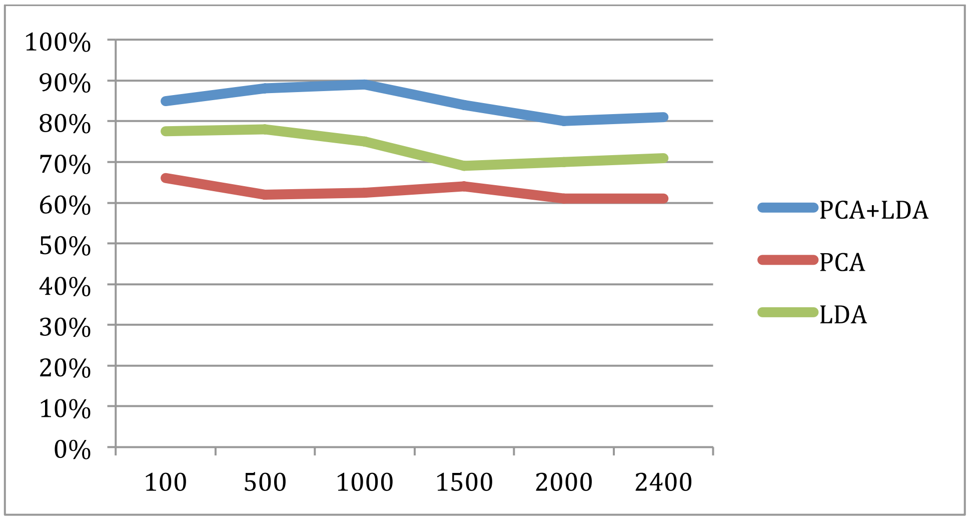


Figure 7: Test Accuracy Rate as a result of the varying dataset sizes

Identity Recognition

Here we used the LBPH method that is very much invariant to lighting conditions as it only computes the local features encoded with one max pixel unit in the binary format. The extended version of the Local Binary Pattern Histogram we have implemented has the following parameters in order to achieve the fastest computation possible.

Radius = 1,
Neighbors = 1,
Grid_X = 2,
Grid_Y = 2

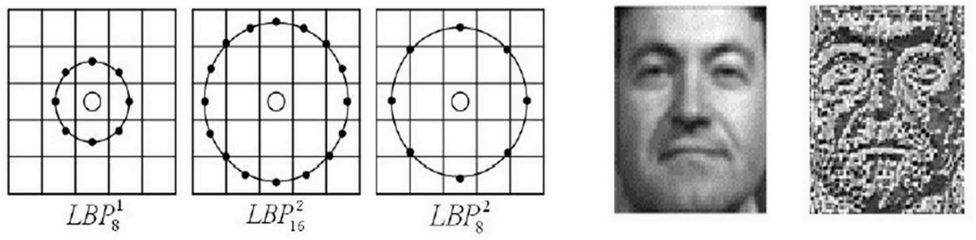


Figure 8: An Example from the test images.

So far, the overall accuracy rate from such a test when using the test set of the same 2400 faces in 100 category is rather high with the following accuracy level

100	500	1000	1500	2000	2400
100%	99.7%	99.7%	99.5%	99.3%	99.0%

Table 1: Accuracy rate of LBPH with respect to test database sizes
However, due to the different special requirement of having to test largely unseen faces, we separated the dataset into 2 parts, the training set will be a random 1200 samples from the dataset and another 1200 will be the test set to comprise 600 from the training

set(seen faces) and 600 from the other half(unseen faces). For the result to make sense, we defined a THRESHOLD value for the similarity marking for faces, and tuned from the same dataset above, here is the response graph for the THRESHOLD result.

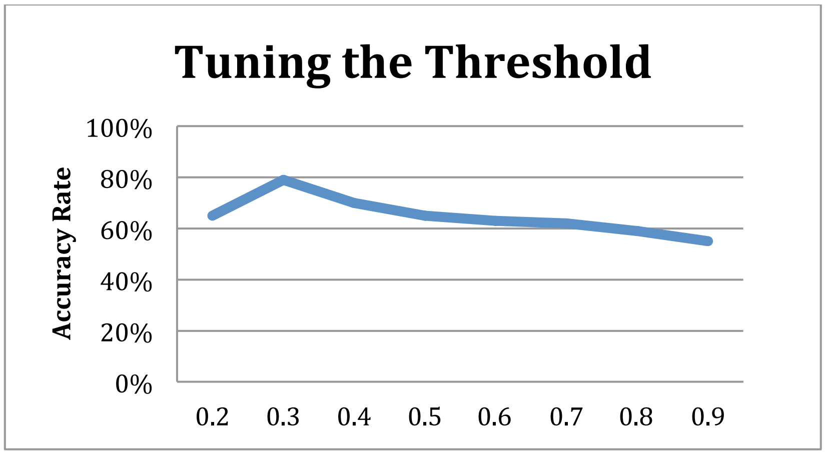


Figure 9: the threshold values used to tune.

```
Double getSimilarity(const Mat A, const Mat B){
Double errorL2 = norm(A, B, CV_L2);
Double similarity = errorL2/(double)(A.rows *A.cols);
Return similarity
}

similarity =getSimilarity(preprocessedFace, reconstructedFace);
if (similarity > THRESHOLD){
identity = -1;
}
```

Section 5: Conclusions:

Over the project, we have achieved system design, facial recognition algorithm comparison and selection, facial tracking algorithm development, gender detection algorithm selection and implementation, facial info representation structure selection / processing implementation, as well as basic data upload / storage implementation. We have successfully built up a prototype that meets the performance and deployment requirements set at the beginning, including real-time processing, computing power constraints, and desirable low network traffic / data storage. During the process, we surveyed the current algorithms available (refer to section 4), experimented with parameters (e.g. frame/second, and max distance for facial tracking classification), and developed an easy and effective facial tracking method. We believe that the current system is a robust prototype, and could be the basis for further development into a potentially commercializable facial-based Customer Relationship Management system.

Section 6: References

[1]http://justme-crazylifeofmarlene.blogspot.com/2013/12/while-will-probably-e-commerce-overtake.html
[2]http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

[3] Face Recognition with Local Binary Patterns, Timo Ahonen, Abdenour hadid, and matti Pietikainen, Computer Vision, ECCV 2004, 978-3-540-21984-2
[4] Face detection based on half face-template, Wei Chen, Tongfeng Sun, Xiaodong Yang, Li Wang, Electronic Measurement & Instruments, 2009. ICEMI '09. 9th International Conference on Digital Object Identifier: 10.1109/ICEMI.2009.5274642
[5] An Efficient Skin Illumination Compensation Model for Efficient Face Detection, Kumar, C.N.R; Bindu, A. IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on Digital Object Identifier: 10.1109/IECON.2006.348133
[6] *"Face description with local binary patterns: Application to face recognition."* Ahonen T, Hadid A. and Pietikäinen M. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12):2037-2041
[7]I. Craw, D. Tock, and A. Bennett, "Finding face features," Proc.of 2nd European Conf. Computer Vision. pp. 92-96, 1992.
[8] A. Lanitis, C. J. Taylor, and T. F. Cootes, "An automatic face identification system using flexible appearance models," Image and Vision Computing, vol.13, no.5, pp.393-401, 1995.
[9] I. T. Jolliffe, Principal component analysis, New York: Springer-Verlag, 1986.
[10] M. Turk and A. Pentland, "Eigenfaces for recognition," J. of Cognitive Neuroscience, vol.3, no. 1, pp. 71-86, 1991.
[11] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.12, no.1, pp. 103-108, Jan. 1990.
[12] T, Agui, Y. Kokubo, H. Nagashi, and T. Nagao, "Extraction of face recognition from monochromatic photographs using neural networks," Proc. 2nd Int'l Conf. Automation, Robotics, and Computer Vision, vol.1, pp. 18.81-18.8.5, 1992.
[13] O. Bernier, M. Collobert, R. Feraud, V. Lemaried, J. E. Viallet, and D. Collobert, "MULTRAK: A system for automatic multiperson localization and tracking in real-time," Proc, IEEE. Int'l Conf. Image Processing, pp. 136-140, 1998.
[14] T. K. Leung, M. C. Burl, and P. Perona, "Finding faces in cluttered scenes using random labeled graph matching," Proc. 5th IEEE int'l Conf. Computer Vision, pp. 637-644, 1995.
[15] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no.7. pp. 696-710, July, 1997.
[16] A. J. Colmenarez and T. S. Huang, "Face detection with information-based maximum discrimination," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 782-787, 1997.
[17] M. S. Lew, "Information theoretic view-based and modular face detection," Proc. 2nd Int'l Conf.Automatic Face and Gesture Recognition, pp. 198-203, 1996.
[18] P. Viola and M. Jones, "Robust real-time object detection," International Journal of Computer Vision, 2002.
[19] R. Lienhard and J. Maydt, "An extended set of haar-like features for rapid object detection," Proceedings of the IEEE International Conference on Image Processing, pp. 900–903, 2002.
[20] Mastering OpenCV with Practical Computer Vision Projects, Daniel Baggio, Shervin Emami, open source PACKT
[21] Face Recognition: A Convolutional Neural Network Approach, Steve Lawrence, C.Lee. Giles, Ah Chung Tsoi, Andrew D. Back, , IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition, 2012

[22] Face recognition by elastic bunch graph matching, L.Wiskott, etc, In Intelligent Biometric Techniques in Fingerprint and Face Recognition,eds. L.C. Jain et al., publ. CRC Press, ISBN 0-8493-2055-0, Chapter 11, pp. 355-396, (1999).
[23] Face recognition using a fuzzy fisherface classifier, Keun-Chang Kwek, Witold Pedrycz, Pattern Recognition, Volume 38, Issue 10, October 2005, Pages1717-1732.
[24] Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, Peter N. Belhumeur, Joao~ P. Hespanha, and David J. Kriegman, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 7, JULY 1997
[25] Revisiting Linear Discriminant Techniques in Gender Recognition, IEEE Transactions on Pattern Analysis and machine intelligence, Vol 33, nO.4 April 2011
[26] http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html