Cab Transaction Using Facial Recognition and Matching

Rohan Chavan[1], Kshitij Shukla[2], Saniket Patil[3]
Department of Computer Engineering[123]
St. John College of Engineering and Management, Palghar

Abstract*—After booking a Cab, at the end of the ride one can make the transactions with either cash, card, UPI, E-wallet etc. Sometimes these methods are not hassle free. Therefore, a new system has been proposed for transactions, where face scan method will be used. The system in the cab will calculate the fare based on the distance and then it will authenticate the riders face and transact via connected central database. Face recognition begins with extracting the coordinates of features such as width of mouth; width of eyes, pupil, and compare the result with the measurements stored in the database and return the closest record (facial metrics). The main purpose of this research is to investigate different types of face recognition algorithms like* Eigen *face and Fisherface. The open CV provides these recognition algorithms. This is done by comparing the receiver operating characteristics curve to implement in the given Transaction using Facial Recognition. In addition, it is noted that Eigen Face delivers better results than Fisherface algorithms; Eigen face delivers between 70 to 80% accuracy between faces. If the user's input image matched with the trained dataset image then the User Profile and Transaction details gets loaded, and the subsequent trip details gets stored in the User Profile database. The database is connected to frame web server. The accuracy of our system is 70%.*

1. INTRODUCTION

With the popularity in India of mobile payment platforms such as Paytm and PhonePe, QR codes can be found almost anytime, anywhere in Indian daily life. From luxury shopping centers to street vendors, consumers can make payments easily by scanning a QR code with their smartphones. The awkwardness of forgetting your wallets at home no longer exists. As long as you have a mobile payment set up on your phone, you can virtually always go cashless in India. But, things are changing as we speak. QR codes are just a step in the evolution of mobile payment technology and they may soon be a thing of the past. In fact, soon people in India may be able to forget about QR codes, and pay with virtually nothing but themselves. This new payment method we are talking about is facial recognition, which we are planning to implement in Cabs. The eigenface technique, another method based on linearly projecting the image space to a low dimensional subspace, has similar computational requirements which can be used for our project, Yet, extensive experimental results demonstrate that the proposed "Fisherface" method has error rates that are lower than those of the eigenface technique for tests on the Harvard and Yale face databases.[1]

2. LITERATURE SURVEY

In computer technology image based on identical twin, face recognition technology is challenging task. Traditional facial recognition system exhibit poor performance in differentiating identical twins and similar person under practical conditions. Traditionally lot of manual experiments were performed to identify twins and also to recognize their features with difference, and many more systems were existed to show differences in twins by using finger prints, voice and iris as part of pattern recognition. In existing methods, many techniques are used for twin's identification like finger print, voice and iris recognition. The process of finger print identification is used to identify unique person in industry or organizations. The method propose a scan image taken from the person and compare with database for identification. The iris recognition also similar method to finger prints identification. [2]

A proposed system is expected to provide higher level of authentication (multifactor authentication) which will bring unauthorized access to the barest minimum. Before access will be granted, the user will have to take a facial photograph to have access to his/her account, the geometry of the face, distance of the eyes and the nose is compared. This photograph will be compared with the photograph in the bank server and the NCC server for verification, if it passes the verification, access will be granted, otherwise it will denied. In the event of unauthorized access, a security alert message will be sent to the bank. [3]

Use of payment cards in various places such as shopping, restaurants, lodges and online payment for booking hotels, movie tickets, flight and train tickets etc are increasing day by day. Therefore, the problem is that a person has to carry payment cards along with him and keep the cards secure to use it all the time. This also lacked security. In the present work the biometric face recognition payments is used in all kinds of payments. Thus, it avoids the need to memorize different passwords. Face recognition payment system is safe, secure and even easy to use. It is reliable and more efficient compared to other payment technologies. A general design of online payment system using face recognition is proposed. The methods adopted for face recognition are by finding the Eigen faces and Euclidean distance.[4]

3. METHODOLOGY

The proposed system involves automated cashless payment system using facial recognition. This system will work as an API between the payment system and any other application in which face of the user can be used for authentication. Using this system, the user can directly pay fares for the cab rides or hotel stay etc. The initial step of the system is the registration process. During the time of registration, the system will collect the facial information of the user by asking user to upload a video of user's face. The system will then extract the frames from that video which will be used stored in the database for further use. Every user has specific user directory in which all collected frames will be saved.

This dataset of images will be used to train our facial recognition model powered by OPENCV, Local Binary Patterns Histograms Face Recognizer (LBPH) algorithm. After the complete registration process, the user can login into the system by using facial recognition, and selects the destination. The system will then allocate a nearby cab to the user, and ensure the pickup for that user. Based on the pickup point and the destination, an invoice will be generated and the amount will be shown to the user. After the completion of the ride, the user will be redirected to the payment engine, where user will be prompted to enter his special pin to complete the transaction. [5]

4. ARCHITECTURE

Our Main Web-App will be responsible for User Registration. Here 60 user photos will be taken and stored into the User Image Database.

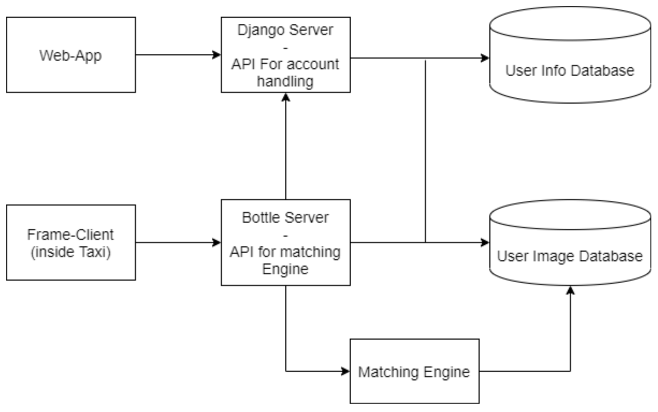

Web-App
Django Server - API For account handling
User Info Database
Frame-Client (inside Taxi)
Bottle Server - API for matching Engine
User Image Database
Matching Engine

Figure 1: Architecture Diagram

The Django Server will be used for account handling, such as user registration, user login, password updating, data updating, etc. The entire project is based on microservice architecture. Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight. Every module will be independent. The bottle server will help the modules communicate with each other. The client will hit the API endpoint of bottle server to access the matching engine inside the cab. The web app will also use bottle server api to store images in the database.

5. ALGORITHM

We will be using LBPH (Local Binary Patterns Histogram) Algorithm. Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. [6] It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with a simple data vector. [7]

5.1. PSEUDOCODE

```
// Check for valid command line arguments, print usage if no
arguments were given.
    if (sys.argv < 0){
        exit
    }
    // Get the path to your CSV, which hold the images and
    corresponding labels.
    if data != valid
    try {
        read_csv(fn_csv, images, labels);
    } catch (Exception) {
        print(exception)
        exit(1);
    }
    // Quit if there are not enough images for this demo.
    if(images.size() <= 1) {
        string error_message = "This demo needs at least 2
        images to work. Please add more images to your data
        set!";
        CV_Error(CV_StsError, error_message);
    }
    // Get the height from the first image.
    int height = images[0].rows;
    Mat testSample = images[images.size() - 1];
    int testLabel = labels[labels.size() - 1];
    // The following lines create an LBPH model for
    // face recognition and train it with the images and
    // labels read from the given CSV file.
    Ptr<FaceRecognizer> model = createLBPHFaceRecognizer();
    model->train(images, labels);
    // The following line predicts the label of a given
    // test image:
    int predictedLabel = model->predict(testSample);
    // To get the confidence of a prediction call the model
    with:
    string result_message = format("Predicted class = %d /
    Actual class = %d.", predictedLabel, testLabel);
    cout << result_message << endl;
    // Set threshold
    model->set("threshold", 0.0);
    predictedLabel = model->predict(testSample);
    cout << "Predicted class = " << predictedLabel << endl;
    cout << "Model Information:" << endl;
    string model_info = format("\tLBPH(radius=%i, neighbors=%i,
    grid_x=%i, grid_y=%i, threshold=%.2f)",
        model->getInt("radius"),
        model->getInt("neighbors"),
        model->getInt("grid_x"),
        model->getInt("grid_y"),
        model->getDouble("threshold"));
    cout << model_info << endl;
    // We could get the histograms for example:
    vector<Mat> histograms = model->getMatVector("histograms");
    // Change the length of histogram
    cout << "Size of the histograms: " << histograms[0].total()
    << endl;
    return 0;
}
```

The algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.
So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a 'confidence' measurement. Note: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.

We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined. [8]

6. WORKING

Login Screen is made in ReactJs, Consists of Simple Username and Password combination interface. The other links are Registration links and also links for Terms of Use and Privacy policy.

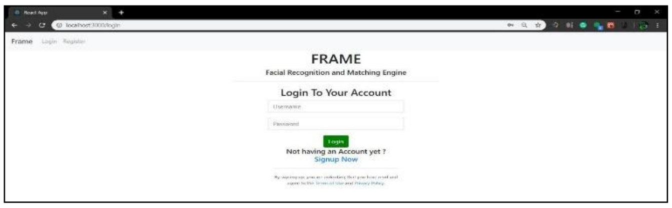


Figure 2: Login Screen

Registration will be done on the following screen. The registration page will accept username, email id, password combination.

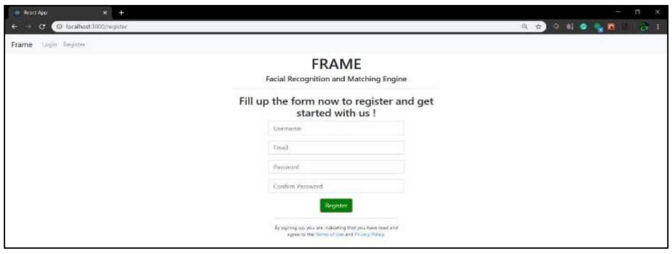


Figure 3: Registration Screen

Upon successful registration user will be redirected to the login page which he can login.

The login screen will be as shown before.

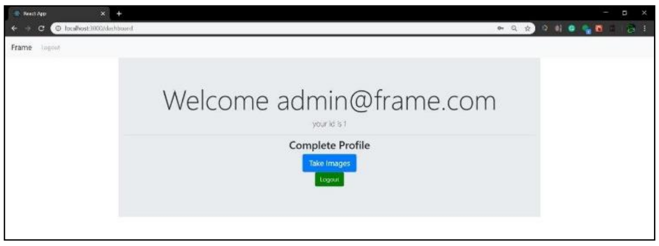


Figure 4: Logged In Screen

Upon login the user will be greeted with the above screen from which he can take images for the database.

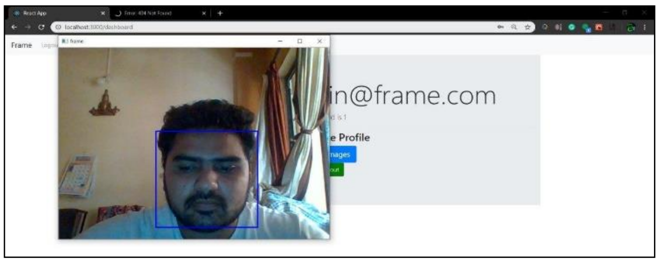


Figure 5: Dataset Collection

The system will open the webcam and take up to 60 images of the user and save it to the user image database, where as the textual details like username and password will be stored in user info database as show in the architecture diagram.

In the Client application, our Face is the login Feature, Once the client enters a taxi, the Camera will scan the face on the passenger seat, if the person has signed up, and his profile exists, then it'll log in the user automatically, otherwise the client will give an error message such as follows.

The Home screen of the client side application will be as follows:

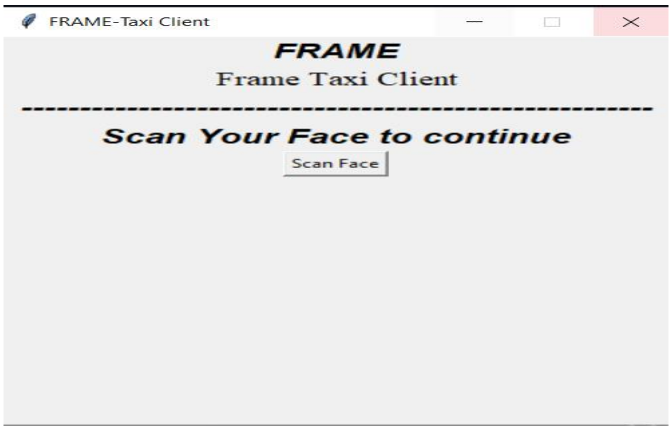
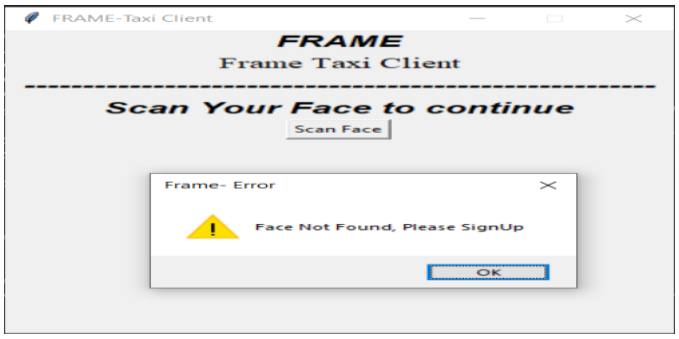


Figure 6: Client Homescreen

Upon Unsuccessful scan in case of missing profile the User shall get the following screen.



Face Not Found, Please SignUp

Figure 7: Unsuccessful Login Attempt

And the successful login shall forward user to the logged in screen.

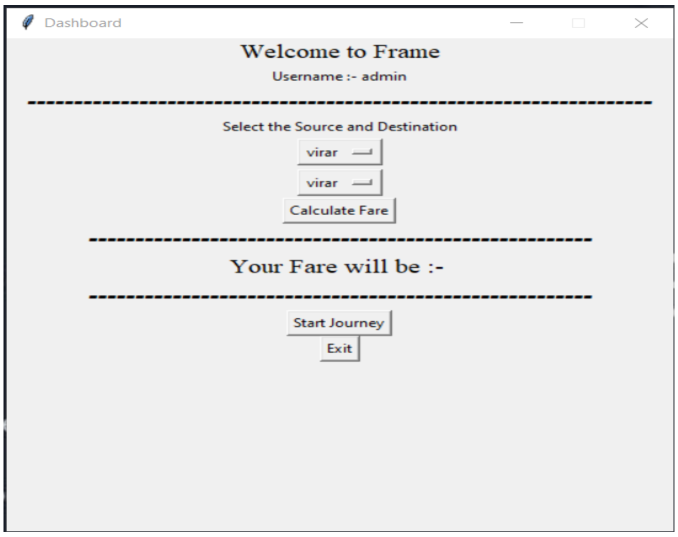


Figure 8: Successful Login

Here the user has options to select source and destination of journey, get fare details, and make transactions.

7. CONCLUSION

The project is an advanced system which follows microservice architecture based on server client as well as web infrastructure. All the services are interconnected and use REST API for inter-process request/response communication.

Account management is done using an API built with Django Rest framework, Bootle server acts as an communication unit with the matching engine which uses openCV LBHP (local binary histogram pattern) to recognize the images which are stored in the user image database.

8. REFERENCES

[1] 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, vol. 19, no. 7, pp. 711–720, 1997. [Online]. Available: 10.1109/34.598228;https: //dx.doi.org/10.1109/34.598228

[2] R. Prema and D. P. Shanmugapriya, “A Face Recognition Techniques for Differentiate Similar Faces and Twin Faces,” International conference on Energy, Communication, Data Analytics and Soft Computing, 2017.

[3] F. A. Adesuyi, “Secure Authentication for Mobile Banking Using Facial Recognition,” IOSR Journal of Computer Engineering, vol. 10, no. 3, pp. 51–59, 2013. [Online]. Available: 10.9790/0661-01035159;https: //dx.doi.org/10.9790/0661-01035159

[4] . R. Surekha, Gondkar, D. B. C. S. Saurab, and Mala, “Biometric Face Recognition Payment System”,” International Journal of Engineering Research & Technology (IJERT). Special Issue, vol. 6, pp. 2278–0181, 2018.

[5] N. Stekas, D. V. Den, and Heuvel, “Face recognition using local binary patterns histograms (lbph) on an fpga-based system on chip (soc),” IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2016.

[6] Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. “Face description with local binary patterns: Application to face recognition.” IEEE transactions on pattern analysis and machine intelligence 28.12 (2006): 2037–2041.

[7] Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. “Face recognition with local binary patterns.” Computer vision-eccv 2004 (2004): 469–481.

[8] Ojala, Timo, Matti Pietikainen, and Topi Maenpaa. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns.” IEEE Transactions on pattern analysis and machine intelligence 24.7 (2002): 971–987.