A Face Recognition Based Multiplayer Mobile Game Application

Ugur Demir, Esam Ghaleb, Hazim Kemal Ekenel

A Face Recognition Based Multiplayer Mobile Game Application

Ugur Demir, Esam Ghaleb, and Hazım Kemal Ekenel

Faculty of Computer and Informatics, Istanbul Technical University, Istanbul, Turkey
{ugurdemir,ghalebe,ekenel}@itu.edu.tr

Abstract. In this paper, we present a multiplayer mobile game application that aims at enabling individuals play paintball or laser tag style games using their smartphones. In the application, face detection and recognition technologies are utilised to detect and identify the individuals, respectively. In the game, first, one of the players starts the game and invites the others to join. Once everyone joins the game, they receive a notification for the training stage, at which they need to record another player's face for a short time. After the completion of the training stage, the players can start shooting each other, that is, direct the smartphone to another user and when the face is visible, press the shoot button on the screen. Both the shooter and the one who is shot are notified by the system after a successful hit. To realise this game in real-time, fast and robust face detection and face recognition algorithms have been employed. The face recognition performance of the system is benchmarked on the face data collected from the game, when it is played with up to ten players. It is found that the system is able to identify the players with a success rate of around or over 90% depending on the number of players in the game.

Keywords: face recognition, multiplayer mobile game, DCT, LBP, SVM

1 Introduction

Mobile applications have become more common and popular with the advances in smartphone technologies. They have also become one of the main segments of the digital entertainment. Many successful mobile gaming companies have emerged recently with very high market values. Most of these games require users' focus and attention, isolating them from the real world. However, mobile games can also be designed to provide entertaining human-human interaction experience. In this way, instead of isolating the users from the real world, mobile game applications can convert a smartphone to a tool that allows users to interact with the others in a fun way. With this motivation, in this study, we designed a mobile game application that enables individuals play paintball or laser tag style games using their smartphones[1]. The application benefits from computer vision and pattern recognition technologies, specifically, face detection and recognition to

[1] A demo video of the system can be found at http://web.itu.edu.tr/ekenel/facetag.avi

detect and identify the players. In order to have real-time processing capability, which is required for the game, we have employed fast and robust face detection and recognition algorithms.

The developed game consists of two main components, server and mobile application. The server application regulates the game rules and players' network stream. In addition, server application contains the face recognition module. The game application sends camera frames that contain players' faces to the server. Thus, mobile application contains only network and face detection module. At the beginning of the game, players join a game session, which is created by one of the players. After all the players are attended, training stage is started by the server. Mobile applications gather player images as training data and sends them to the server. Each player collects different player's face data. When server application receives all of the training data, it extracts the features and trains the classifiers. Afterwards, the game starts and players attempt to shoot each other. If camera detects a face, while a player is shooting, it sends the detected face image to the server application. The server application assigns an identity to the received face image by using trained classifiers, and then broadcasts the necessary information to the mobile devices. The flow diagram of the game is shown in Fig. 1.

The rest of the paper is organised as follows. In Section 2, the server application architecture is introduced. In Section 3, design of the mobile application is explained. In Section 4, proposed face recognition methods and classifiers are stated. Tests and experimental results are presented in Section 5. Finally, conclusions and future work are given in Section 6.

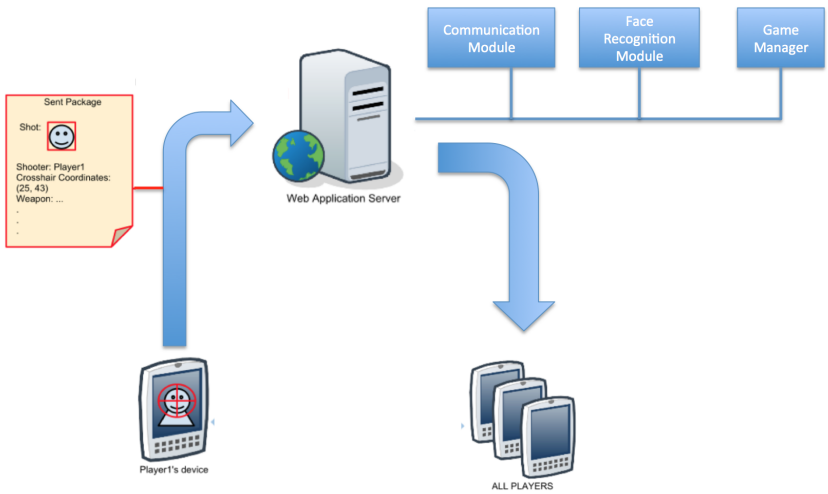


Fig. 1. Multiplayer Mobile Game Application Flow Diagram

2 Server Application

Server application that controls the game flow is the main part of the system and consists of three core modules, which are network module, game flow control module, and face recognition module. All of these core modules work collectively during a game session.

Network module's main purpose is receiving the commands and images from mobile devices, and then deciding to what information should be broadcasted to players. All of the mobile devices connect to the server over a TCP socket before starting to play. Server always listens to the TCP port for incoming client data, which is represented with JSON format. It creates and runs a new thread called network thread for each connected mobile device. These threads manage the network operations. Network module does not interfere the game flow logic. It is only responsible to transfer the received data to game flow module and transmit resulting data to relevant clients. This abstraction enables us to design a more robust multiplayer game server.

Game flow control module executes the client's orders delivered by network module, which are based on the game rules. This module decides to execute necessary function calls depending on the client's request. If an image is received, it is transferred to face recognition module and results are evaluated. Player scores are calculated by this module. If a player shoot another player, game flow control module tells network module to notify these players.

Face recognition module handles training and classification tasks in the server. It has two stages in its life cycle; training and classification. In the training part, face recognition module collects player images and generates a multi-class classifier for each mobile device. We assume that players do not shoot themselves. Thus, each classifier recognise all the players in a game session except the player, who owns the mobile device. After the training stage, flow control module starts the game. Then classification stage takes control in face recognition module. In this stage, player images are queried to determine, which player's face image it is. All modules and server application are implemented in Java programming language. In the face recognition module, OpenCV library Java port is used.

3 Mobile Application

Mobile game application contains user interface and communication module, which delivers user commands and listens for incoming server data. At first, devices send a connection request to the server, when a player starts the game application. If the connection is successfully established, a user menu, which shows the game options, appears on the screen. Then, player can join or create a game lobby via this menu to start the game.

After starting the mobile game application, a thread called communication thread is started to run by the application. Communication thread listens the incoming server data during the application life-cycle to broadcast the necessary data to the application screens, which player interacts with. We apply observer

pattern to achieve this scheme. In the observer pattern, objects that display the data is isolated from the object that stores the data and storage object is observed to detect changes on data [1]. Observer pattern is suitable for communication module of the game due to its architecture. Communication thread listens and stores the data, so it represents the observable object in the system. Application screens, which is called Activity in Android SDK [2], stand for the observer object in observer pattern. Server sends state changes via TCP sockets to communication thread and activities observe the communication thread for any data changes. Fig. 2 shows the network architecture.

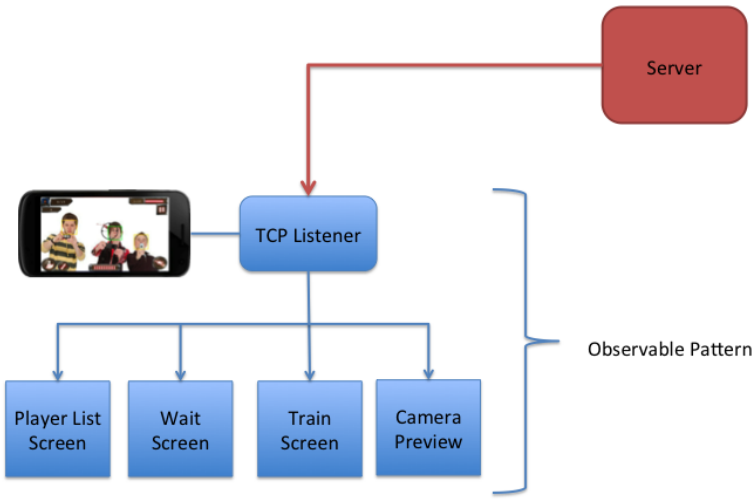


Fig. 2. Mobile Application Flow Diagram

Activities in the application register themselves to observe communication thread before starting and unregister before closing. Thus, communication thread transfer the data to only relevant activities and modules. Observer activities filter and parse the received data and display to user. Mobile application flow consists of three main step; joining the game, collecting the training data, and playing. One player, called creator, creates a game lobby and waits for the other players join the existing lobby. Creator player can see the entered players on the application screen and start, when there are at least two players in the lobby including creator.

After the creator starts the game, the game lobby is configured in the server and server sends a command, which tells training session must be started, to all players. In the training stage, players' face images are collected via smartphones' cameras. Collected images are sent to the server for training. When all players send the images and the training is completed, server starts the game and sends start game command to players' devices. Main game screen basically consists of

camera preview and an aim marker. If a player presses the shoot button and the aim marker overlays with a player's face at the same time, detected face image is sent to the server. Server analyses the face image and sends the information about who shoots which player to the corresponding players. Fig. 3 shows same screenshots from the mobile game screen.

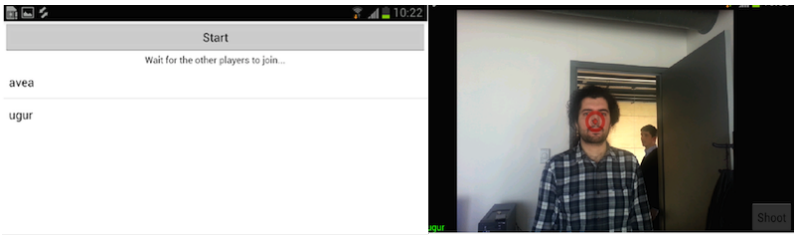


Fig. 3. Sample Screenshots from the Mobile Application

4 Face Recognition

Face recognition process consists of the following steps: face detection, feature vector extraction and classification as shown in Fig. 4. In face recognition, the first step is to detect a face in a given frame. In this study, Viola-Jones face detection method is used [3]. We benefit from the available OpenCV implementation of this approach [4]. Fig. 5 shows sample detected faces. Please note the illumination and view variations that pose challenges for the application.

For feature extraction from face images Discrete Cosine Transformation (DCT) and Local Binary Pattern (LBP) are implemented. These two representation methods are proven to be fast and robust in previous studies [5, 6]. DCT is popular for being a fast and efficient transformation, mainly used for data compression, due to its capability to represent data in a compact form. As a result, this makes DCT a suitable approach for mobile applications. In this paper, to extract a feature vector from a given image, DCT was applied as explained in reference [7]. The images are first resized to have 64×64 pixel resolution and then divided to nonoverlapping blocks. Each block has 8×8 pixel resolution. Then DCT is applied on these local blocks. Following this process, zig-zag scanning is used to order the DCT coefficients. For each block, M features are picked and normalised. These local features are then concatenated to form the overall feature vector of the image. This obtained feature vector represents the given image. In the implementation M is set to 5, making the resulting DCT feature vector of the length equal to $5 \times 8 \times 8 = 320$.

LBP is widely used for face feature extraction and it has been proven to be very effective and successful approach for many tasks such as face recognition and face verification. In LBP, for every pixel in gray level image, it assigns a label

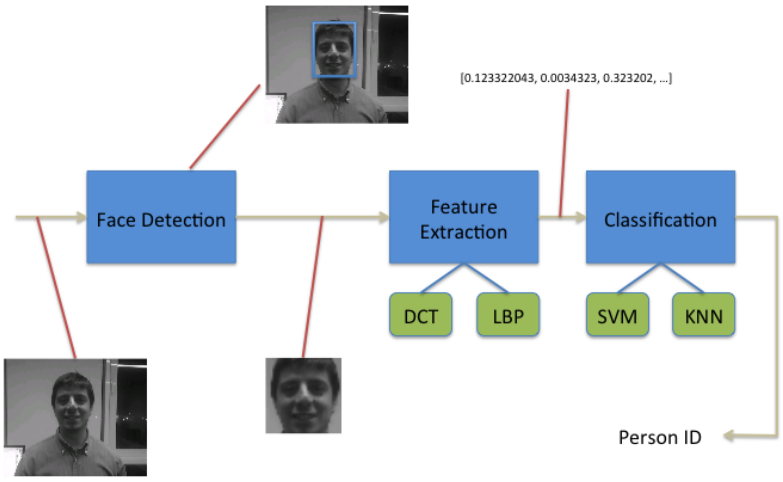


Fig. 4. Face Recognition Process

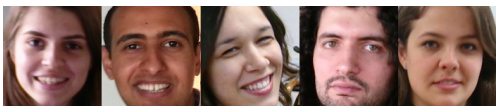


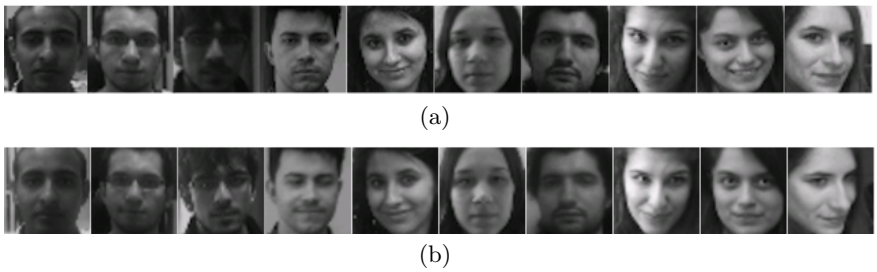
Fig. 5. Sample Detected Faces

to it, based on its neighborhood [5]. In this study, first the face image is resized to 66×66 pixel resolution -two additional pixels due to the operations at the image boundaries-, then divided to nonoverlapping blocks of 8×8 pixel resolution. For each pixel in a block, the pixel is compared to each of its 8 neighbours and its corresponding LBP code is assigned. Since the 8 neighborhood is considered, a 256-bin histogram of LBP codes is computed. To reduce the dimensionality, the subset of LBP codes, called the uniform patterns, are used [5, 8, 9]. A uniform pattern contains a two or less number of transitions from zero to one or vice versa. As a result, the patterns that has more than two transitions are aggregated into a single bin in the LBP histogram besides the other uniform patterns, reducing the histogram size to 59 bins. The obtained histograms from local blocks are concatenated into a single feature vector that represents the entire image. The resulting LBP feature vector's length equals to $8 \times 8 \times 59 = 3776$.

Following feature vector extraction from the face images, the application employs multi-class classifiers, either Support Vector Machine (SVM) or K-Nearest Neighbor (KNN) classifiers. In SVM the used kernel function is radial basis function and constraint parameter is set to 0.5 [10]. In KNN, K is set to 1 and the L1 and L2 distance metrics are used.

5 Experiments

In order to asses the face recognition system's performance, a face database is generated using the developed mobile game application. For these evaluations ten images for training and ten images for test are collected from twelve players. Samples from training and testing data sets are shown in Fig. 6. The variations in facial expression, head pose, and illumination results in a very challenging task. DCT and LBP are used for feature extraction, respectively, and the extracted features are then classified by SVM or KNN. In the tests, we change the number of players in the game from two to ten players and examine the performance for each number of players in the game. For each number of players in the game, we form ten different player combinations and the recognition rates are calculated as the average of these ten different combinations in order to have a reliable measure for the system performance.



(a)

(b)

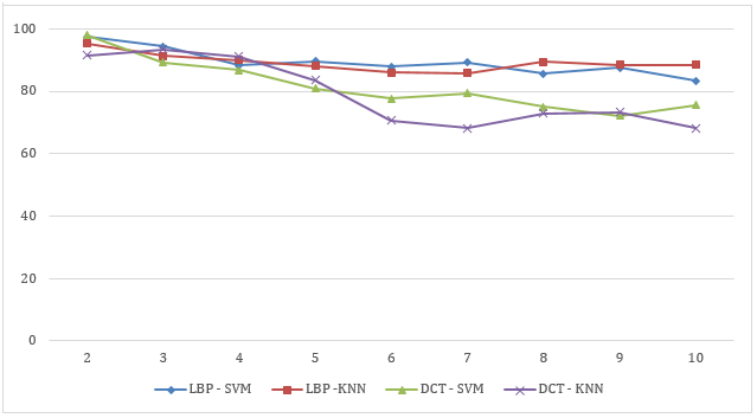
Fig. 6. Samples from (a) the Training Data Set, (b) the Testing Data Set

5.1 Experiments on the Standard Data Set

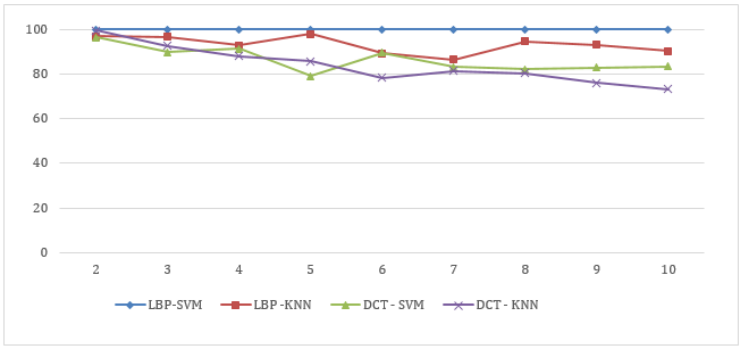
In the first experiment, the evaluation is conducted using the collected data set without doing any data manipulation for additional sample generation. The results of this experiment are plotted in Fig. 7(a). As shown in the figure, the horizontal axis represents the number of the players in the mobile game application, who must be classified, while the vertical axis shows the correct classification rate, which is the average over ten different player combinations. The performance curves give the results for different combination of classification and feature extraction methods. As can be observed from the figure, using LBP features, the best performance is achieved. There is no significant performance difference, when using SVM or KNN. As the number of players in the game increases, the obtained performance decreases, however this decline is slight and the system performance stays around 90% when using LBP features.

5.2 Experiments on the Extended Data Set

In the experiments on the extended data sets, two different scenarios are applied. In these scenarios, either training data set or test data set is extended. In the first scenario, the purpose is to enrich the training data set. As a result, the collected 10 image for training are manipulated to generate 900 images using mirroring, translation and rotation. This extended data set is used for training, while test data set remains as it is. The obtained results are shown in Fig. 7(b). As shown in the performance curve, enriching training data set has improved the recognition rate. Especially, the LBP+SVM combination consistently performs at 100%. However, since SVM's training complexity depends on the square of the amount of samples, that is, it has a complexity of $O(mn^2)$, where m is the dimension of the feature vector and n is the amount of training samples, it is not feasible to use it in this scenario.



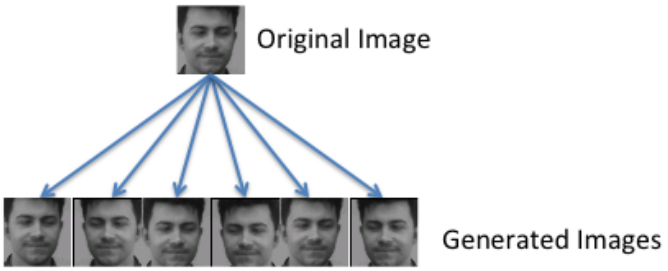
(a)



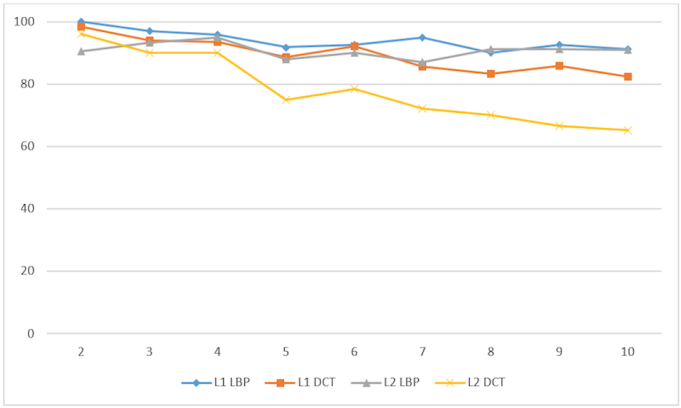
(b)

Fig. 7. (a) Results when ten images are used for training and test stages, (b) results using extended training data set

In the second scenario, the training data set is used as it is, however, this time the test data set is extended and enriched by operations, such as translation, rotation and mirroring. As a result, the collected ten images for test are extended to 60. Since the testing has to be done in real time, this number is far less than the number of additional samples generated for training in the first scenario. Sample generated test images are shown in Fig. 8(a). The performance curve for this experiment is shown in Fig.8(b). Only the KNN classifier is used for this experiment due to its ease of use in decision fusion among different test samples. Similar to the case in the first scenario, the performance improves when using the enriched test data set.



(a)



(b)

Fig. 8. (a) sample generated test images, (b) results using extended testing data set

6 Conclusion

The developed mobile game application is a proof of concept for a successful application of computer vision and pattern recognition technologies to convert a

smartphone to a tool that enables human-human interaction in an entertaining way. The game allows users to play paintball or laser tag style games with their smartphones. Face detection and recognition technologies have been employed to detect and identify the players in the game. The implemented practical and real-time face recognition system is benchmarked within the scope of the application. When using LBP features with ten players in the game, around 90% performance is obtained. SVM and KNN provided similar results. The application is found to be entertaining by the users. It also inspires users about how these technologies can be used in the future. However, we experienced a problem about the usability of the system. It was observed that players keep their smartphones at the face level in order to see the screen, which causes occlusion on the face. To solve this problem, we plan to work on occlusion robust face detection and recognition. In addition, we also plan to design a physical mechanism that can be used in the game to plug the smartphone to, so that the players can have a sufficient distance from their smartphones, while playing.

7 Acknowledgement

This work was supported by the Avea Labs Research Grant, TUBITAK project no. 113E121; and a Marie Curie FP7 Integration Grant within the 7th EU Framework Programme.

References

[1] J.W. Cooper, "Java Design Patterns: A Tutorial", Addison-Wesley, 2000.
[2] Android SDK Activity, http://developer.android.com/reference/android/app/Activity.html
[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", Proc. of CVPR, pages 511-518, 2001.
[4] OpenCV Android SDK, http://opencv.org, May 2014.
[5] T. Ahonen, A. Hadid, and M. Pietikinen, "Face Description with Local Binary Patterns: Application to Face Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (12) (2006) 2037-2041
[6] H.K.Ekenel, "A robust face recognition algorithm for real-world applications", PhD dissertation, University of Karlsruhe (TH) (2009)
[7] H.K. Ekenel, R. Stiefelhagen, "Analysis of local appearance based face recognition: Effects of feature selection and feature normalization", CVPR Biometrics Workshop, New York, USA. (2006)
[8] C. Shan, G. Shaogang, and P.W. McOwan. "Facial Expression Recognition Based on Local Binary Patterns: A Comprehensive Study." Image and Vision Computing 27 (6) (2009): 803-816.
[9] S.E. Choi, Y.J. Lee, S.J. Lee, K.R. Park, J. Kim , "Age estimation using a hierarchical classifier based on global and local facial features", Pattern Recognition 44 (6) (2011) 1262-1281
[10] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", Data Mining and Knowledge Discovery, 121-167 (1998)