Portable Facial Recognition Jukebox Using Fisherfaces (Frj)

Richard Mo
Department of Electrical and Computer Engineering
The University of Michigan - Dearborn
Dearborn, USA

Adnan Shaout
Department of Electrical and Computer Engineering
The University of Michigan - Dearborn
Dearborn, USA

***Abstract*—A portable real-time facial recognition system that is able to play personalized music based on the identified person's preferences was developed. The system is called Portable Facial Recognition Jukebox Using Fisherfaces (FRJ). Raspberry Pi was used as the hardware platform for its relatively low cost and ease of use. This system uses the OpenCV open source library to implement the computer vision Fisherfaces facial recognition algorithms, and uses the Simple DirectMedia Layer (SDL) library for playing the sound files. FRJ is cross-platform and can run on both Windows and Linux operating systems. The source code was written in C++. The accuracy of the recognition program can reach up to 90% under controlled lighting and distance conditions. The user is able to train up to 6 different people (as many as will fit in the GUI). When implemented on a Raspberry Pi, the system is able to go from image capture to facial recognition in an average time of 200ms.**

Keywords—Facial Recognition; Raspberry Pi; Computer Vision; GNU/Linux Operating System; OpenCV; C++

I. INTRODUCTION

Facial Recognition is a very difficult topic and computationally strenuous application. However with the advent of newer technology, computers are running faster and faster. New facial detection and recognition techniques are also being developed that are quicker and more reliable. For instance, a revolutionary change to object detection came in 2001 by Viola and Jones when they invented the Haar-based cascade classifier. The accuracy of the Haar can be up to 95% for detecting frontal faces. Still, an even faster technique was the LBP feature detector developed by Ahonen, Hadid and Pietikainen in 2006 [1] [2] [3]. LBP stand for local binary patterns and is potentially several times faster than Haar-based detectors albeit 10-20% less accurate.

Once an object (or face) is detected and an image is preprocessed, there still needs to be a way to train and recognize a new object when presented to the facial recognition system. In 1901 Karl Pearson proposed the technique of Principle Component Analysis, which transforms a set of possibly correlated variables (raw pixels of trained faces) into a smaller set of uncorrelated variables (eigenvectors and eigenvalues). The theory is that in a higher dimensional dataset, most of the information can be described by a few components. These components are called the principal components and are responsible for the most variance in the data [4].The Eigenfaces representation of faces uses this PCA method to train and store the model used for recognition. However the PCA method has a flaw in that it does not consider any of the classes (different people) and organizes the principle components purely based on the highest variance the component generates. In the case that an external source (such as light) is generating the variance, the principle components may not contain much discriminative information at all. To combat this issue, another class-specific reduction algorithm was developed by Sir R. A. Fisher and uses Linear Discriminant Analysis (LDA). The method was successfully used to classify flowers in his 1936 paper entitled "The use of multiple measurements in taxonomic problems". In this way, features are found to maximize the ratio of between-classes variation to within-classes variation as opposed to just maximizing overall variation [4]. Thus the recognition algorithm is more robust to external sources such as light. This method is called the Fisherfaces algorithm and is the same algorithm used by the FRJ system.

These advances make it more possible to design real-time portable facial recognition systems, such as the one implemented in this paper. While there are plenty of facial recognition software available in the market today, none provide the feature to play a person's favorite song upon recognition. Also, unlike some systems that use still images to detect faces, our system is a real time system that dynamically detects and recognizes a face depending on the mode of the system [5]. This paper presents a cost-effective solution that is cross-platform (Windows and Linux OS) to train faces of different people and upon recognition of a trained person can either play a song or personalized greeting. The system uses all off-the-shelf components that are easily accessible and does not require special infrared cameras as other systems do [6]. Playing a person's favorite song is useful in boosting a person's morale in much of the same way people use a jukebox to liven a place up. This Facial Recognition Jukebox can also be used to distinguish between different moods (emotion) of a person based on facial expression and play a different song accordingly. Through this emotional recognition it can enable the user or other people in the user's vicinity to become aware of the current user's state of emotion.

Further improvements to facial recognition algorithms still need to be made. With today's technology and algorithms facial recognition is not reliable enough to be used for a true security system. However, they can still be used for other purposes that do not require as high reliability such as playing personalized music or generating a face cartoon [7]. Improvements can be made to make the training set less reliant on lighting conditions or angles in which the person is oriented.

The paper is organized as follows: Section II will present the new FRJ system design, Section III will present the experimental results, Section IV will describe how to use the system, and Section V will present the conclusion remarks.

II. System Overview

A. FRJ System Design

The design of the FRJ system can be split into the following major categories: 1. Face Detection 2. Face Preprocessing 3. Face Training 4. Face Recognition 5. Playing Music 6. Saving/Loading Data. A high level block diagram for the code can be seen in Figure 1. The green block illustrates the detection mode. This part of the code is always running by default. First, it acquires a new image and look for a face and eyes. Once the face and eyes are detected there is some preprocessing done to that face region of the image to be used in the next block. The purple Collect Face block code is triggered when a user clicks on the *Add Person* button and will add preprocessed faces for the corresponding person selected. Once the user indicates they are done collecting faces the code continues to the yellow Training Mode block. In this block the Fisherfaces algorithm model is trained using the preprocessed faces and associated face labels (essentially classes). Upon completion of training the code automatically transitions to the red Recognition mode block.

In recognition mode the program basically checks if the captured face matches any of the trained faces and upon matching will play the personalized music of the recognized person. In the event that no person is recognized the code goes back to acquiring the next image and no special action is taken. Side tasks include the ability to save faces so that the user does not need to train a new data set each time. This goes hand in hand with the loading faces functionality being the function to load the previously saved faces. The user can also delete all the faces if they wish to start over in their training. The side tasks are all mouse click based and do not follow the normal process flow as the main functions.

The setup of the overall system can be seen in Figure 2. More detail on the specific components in the setup will be explained in later sections of the paper.

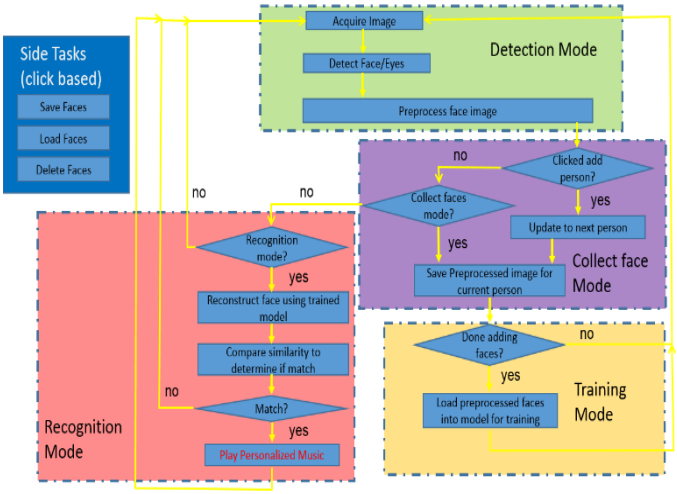

Fig. 1. High Level Block Diagram for Code

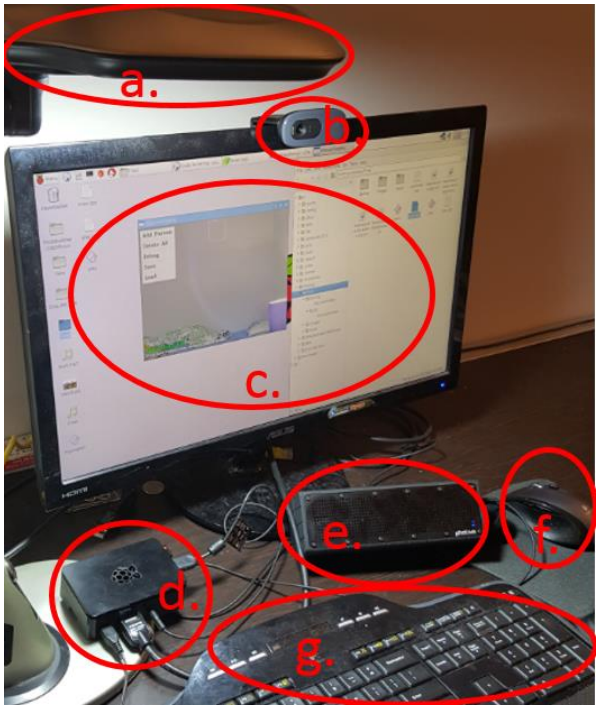

Fig. 2. a. Light fixture b. Webcam c. Display d. Raspberry Pi e. Speaker f. Mouse g. Keyboard

B. Hardware Used

The computer used in the FRJ system is the Raspberry Pi as seen in Figure 3. It has a 900 MHz quad core ARM Cortex-A7 CPU and a Broadcom VideoCore IV @250 MHz GPU. It contains 1GB of RAM. The reason Raspberry Pi was used is it is a relatively cheap prototyping board and contains enough processing power for the computationally intensive algorithms used for the facial recognition.

Fig. 3. Raspberry Pi Model B: credit card sized mini-computer used to compile and execute C++ code

The Logitech HD Webcam C270 was used as the image capture device as it is UVC compatible, meaning that it is capable of streaming video and has a USB interface. Figure 4 shows the webcam. It captures images at 30 frames/sec. Through software the frame is set to 480 pixels height by 640 pixels width.

Fig. 4. Logitech HD Webcam C270

The rest of the components have some flexibility such as the mouse, keyboard, speaker, and light fixture. The mouse and keyboard are used to interface with the Raspberry Pi Raspbian Linux Operating system to click on buttons as well as type in commands on the command line. The speaker outputs the music file or personalized greeting. Finally, the light fixture is used to control the lighting such that there is strong uniform light on the face during training.

C. Software Used

Source code was written in C++ as it is a faster computer language for real time systems (as opposed to MATLAB). The OpenCV library was used for the computer vision functions and the Simple DirectMedia Layer (SDL) library was used for playing sound. Both libraries are cross-platform libraries and thus the system is cross-platform compatible over Linux and Windows Operating Systems.

D. Detection Mode

In detection mode, the program first tries to identify if a face exists within the captured frame. Many pre-trained models are available in the OpenCV official website for the frontal face and eyes. Our program uses the Local Binary Patterns (LBP) classifier for the face (lbpcascade_frontalface.xml) since it can be a few times quicker than the Haar classifier and thus better for real-time processes. LBP is a little less accurate than the Haar (10-20%) however we will then require detection of both eyes within the face for added reliability.

Once the face is detected, the program will then look for both eyes inside the face to be used in the preprocessing section of the code. Both actual eyes must be detected in order for the preprocessing to work correctly as will be evident in the preprocessing section discussed later. Problems will occur if the eye detector is simply used over the region of the whole face as can be seen in Figure 5. This is because some objects of the face can appear to be eye-like and will be misdetected as eyes, such as the nostril. The solution to the misdetection in eyes is to specify top left and top right regions of the face to search for the left eye and right eye respectively. These regions are based on geometric restrictions in which the majority of human eyes will be located in with respect to the face. Reduction in the search region for the eyes not only reduces the processing time but also increases the reliability of detection of the eyes. Correct detection of the face and eyes can be shown in Figure 6.

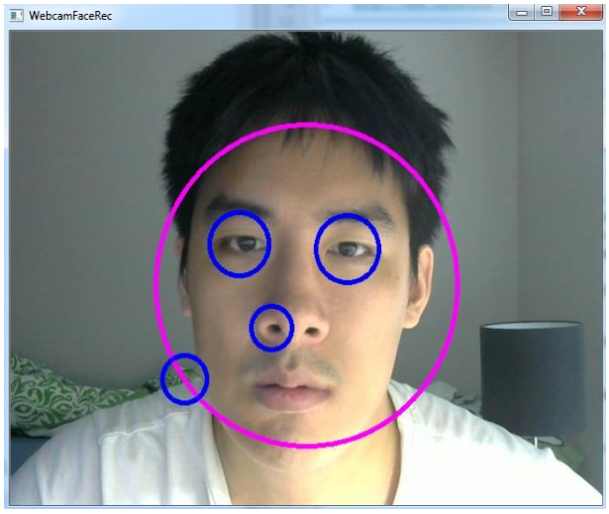

Fig. 5. More than two "Eyes" are detected if the eye classifier is used over the whole face

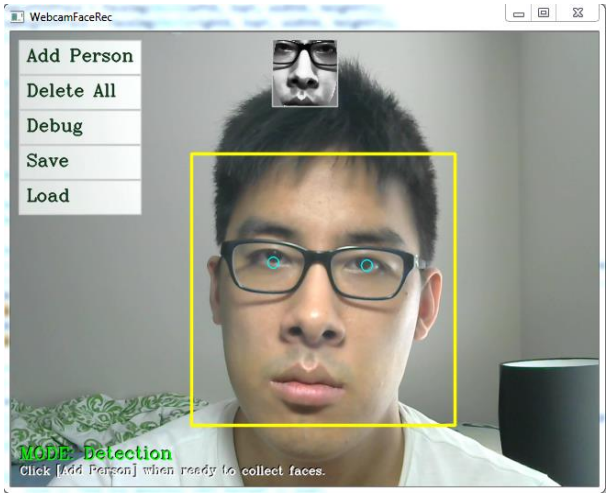

Fig. 6. Correct detection of the face and both eyes

E. Preprocessing

The preprocessing section is a very important section of the code that conditions the detected face images so that it makes it easier to train the facial recognition model and recognize faces.

First the image is converted to grayscale to reduce the data size as an RGB image would have 3 times the number of pixels. From the detection mode the program is able to detect both eyes within a detected face.

Figure 7 shows the two specified regions in which the program uses to search for both eyes. Once both eyes are detected the program calculates the distance between the eyes and then scales the whole face accordingly such that the distance between the eyes is always the same. The eyes are also adjusted so that they are horizontal and at a specified height.

Finally an elliptical mask is placed on the face to crop out any hair or shadows that may appear on the neck. The end result is a normalized 70 by 70 pixel of a preprocessed face as can be seen in Figure 7.

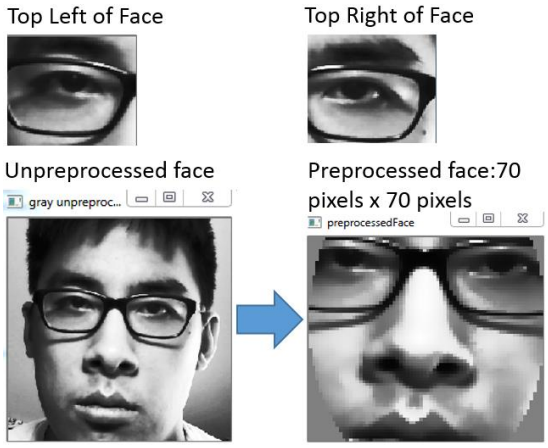

Fig. 7. Preprocessing the face: the position of the eyes are used to scale, rotate, and translate the face

F. Training

The user is able to initiate training after they are done collecting faces. The training set can be initiated with the adding of new faces per the *Add Person* button or can be loaded with the *Load* button to load faces that were stored previously. An example of a training set is shown in figure 8. The training set should contain different facial expressions, angles and lighting conditions for each person to be robust for recognition later. If the set of faces is trained under strong light on the right side of face and during the recognition mode the user as strong light on the left side of the face, the system will have do a poor job of recognizing the given person. The objective is to get more variation between the different faces so as more information is stored in the principle components. Thus the training set should contain more different conditions to get better results.

Fig. 8. Subset of training set

G. Recognition

Using the eigenvectors and eigenvalues trained in the model during the training phase, the program then takes the current preprocessed face and projects it into the PCA subspace. It then takes the projection and reconstructs the face into an image again.

If the query image is indeed part of the training set, then the reconstruction should be very good. The program then compares the reconstructed image to the preprocessed image using doing an L^2 relative error norm calculation. A threshold is set for this error. An error calculated that is below the threshold will indicate a match detected. Figure 9 shows an example of a match identified by the program between the face detected and the face of person 2 as can be seen by the green rectangle drawn on person 2's face.

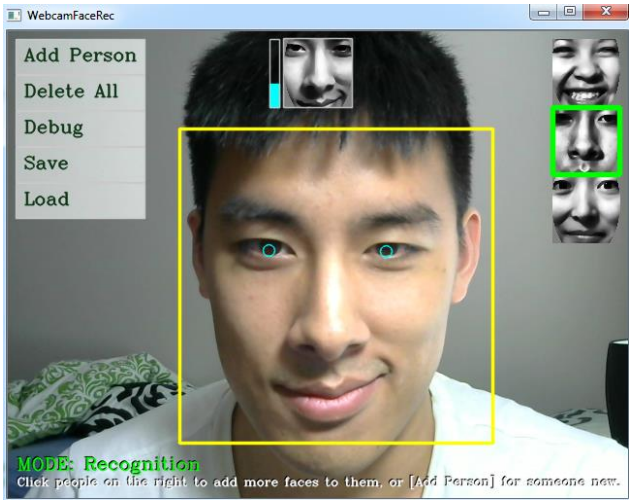

Fig. 9. Example of a recognized face

III. Experimental Results

Under controlled lighting and distance conditions, the FRJ system is able to achieve up to 90% accuracy in recognizing the correct person. The lamp used was an 1800 lumen lamp with color temperature of 6500 Kelvin. A set distance of 55 ± 5 cm from the webcam was defined. The program can accommodate up to 6 unique people within the GUI for training and recognition.

In terms of timing, two devices were tested for the detection rate and recognition rate, and the speeds can be seen in table 1. The first device was the Raspberry Pi in which our portable program will run. The second device was a benchmarking device and was a Toshiba Laptop computer. Comparing the two, the Toshiba Laptop computer was faster by only 50ms. The Raspberry Pi was still able to go from image capture to recognition within an average time of 200ms. Anything less than 1 second will not be perceivable to the user so using the Raspberry Pi is sufficient.

TABLE I. Detection and Recognition Timing

Device	Image capture to face detection average time, detection mode (ms)	Image capture to face recognition average time, recognition mode (ms)
Raspberry Pi: Model B	164.05	199.85
Toshiba Laptop: CPU@2.20 GHz, 6GB Insatlled Memory (RAM)	95.5	155.5

IV. USING THE FRJ SYSTEM

A. Detection mode

No special action is needed for detection mode. The program will automatically start detecting the face and put a yellow rectangle over it and put green circles over the detected eyes. Figure 10 shows an example of detection mode.

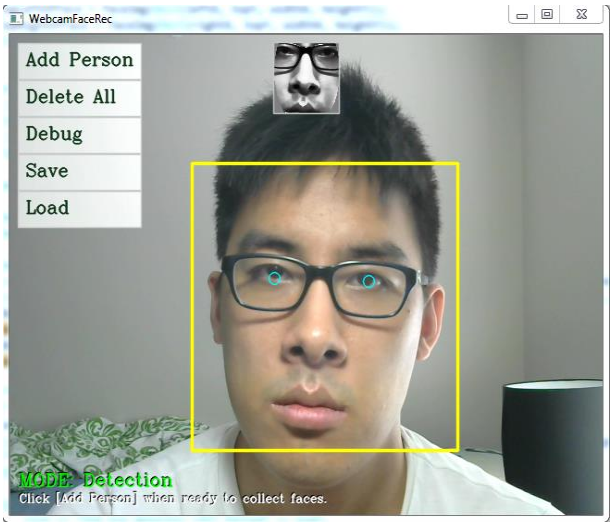

Fig. 10. Example of a recognized face

B. Training Faces for the program

To add people to train click on the "Add Person" button. This will add a person to the training set. Each time a picture is taken the frame within the yellow rectangle will flash white to alert the user. Notice that in the Collect Face mode, the person who is getting new faces trained will have a red rectangle over their face. The most recent captured face of each person will be shown on the right edge of the GUI. Figure 11 shows an example of a Collect Face mode face. If the user wants to add more faces to a person already on the screen, the user simply needs to click the most recent picture of said person and then the program will commence collecting faces for that person.

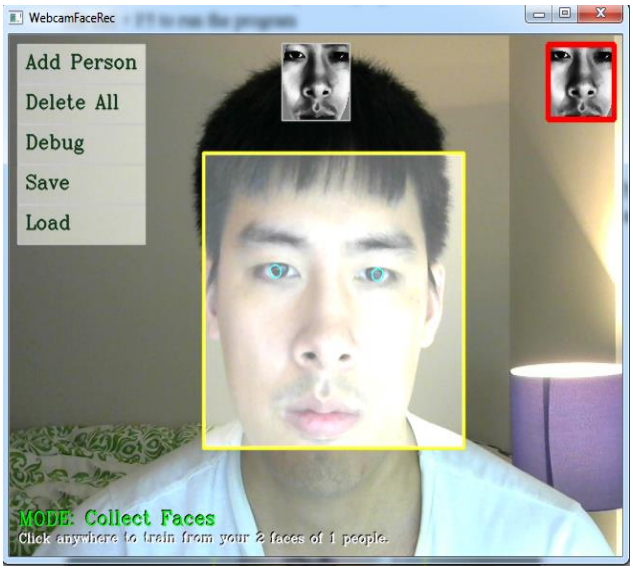

Fig. 11. Example of a face getting collected for the facial recognition model

C. Training Mode

After the user has finished collecting faces, the user must click on an area in the GUI not occupied by a face or button. This will commence the training of the preprocessed faces into the model. After the model is finished training the program will automatically switch to recognition mode. Figure 12 shows an example for the training mode.

Fig. 12. Faces that were previously collected are now being used to train the model

D. Recognition Mode

In recognition mode, once a captured face is recognized with one of the people in the training set a green rectangle will be shown over the recognized person's most recent face and the music file corresponding to that person will play. Figure 13 shows an example for the recognition mode.

The music folder contains the music files as shown in figure 14. The first person corresponds to music file 0.wav. The second person corresponds to music file 1.wav and so on.

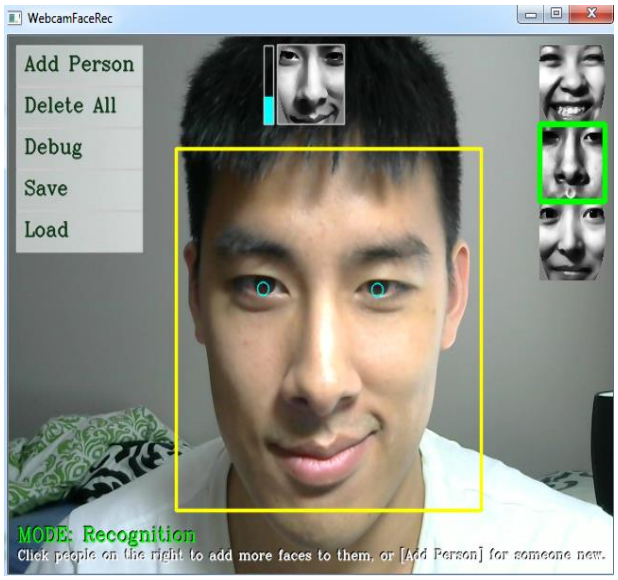

Fig. 13. The correct face is recognized

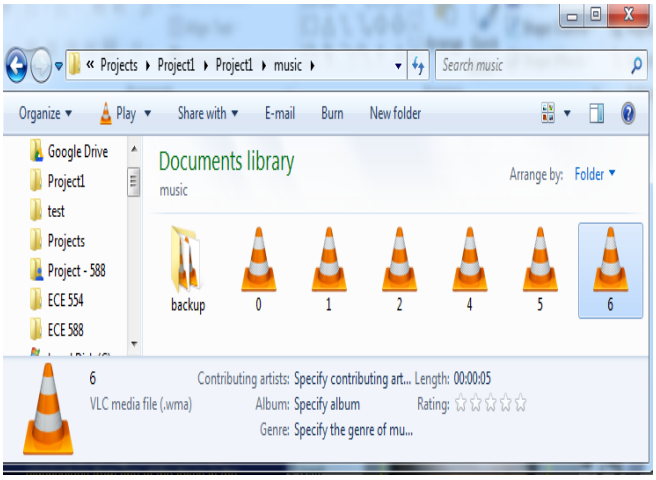

Fig. 14. The music folder

E. Saving Faces

The program will automatically save faces in the Collect Faces mode into the .bmp format. These images will be stored

at <working directory>/images/new_face/ as shown in figure 15.

F. Loading Faces

Pressing the "Load" button will load the faces that stored in <working directory>/images/

This feature makes it convenient in that the user does not have to train the model each time with each person at startup as these faces were previously saved.

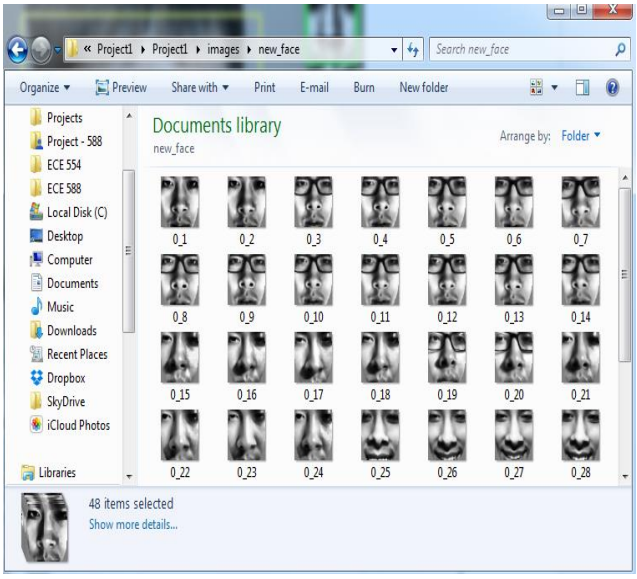

Fig. 15. Collected face directory

G. Deleting Faces

The user can choose to delete faces and essentially start all over in the training process. To delete all the faces the user must click on the "Delete All" button.

V. Conclusion

A system called the *Portable Facial Recognition Jukebox Using Fisherfaces* (FRJ) was developed and provides a convenient portable machine that plays a person's favorite song or a personalized greeting upon recognition of a person's face. The system is implemented on the Raspberry Pi Model B hardware platform and uses OpenCV and Simple DirectMedia Layer libraries for computer vision and media play respectively.

Source code was written in C++. Additional functionality was added to be able to save and load faces trained previously. Currently the system is able to achieve up to 90% accuracy in recognizing the correct trained person under controlled lighting and distance conditions. Up to 6 unique people can be trained as dictated by the size of the GUI. As implemented on the Raspberry Pi, the image capture to facial recognition average time is within 200 ms.

Future work for this project would be to further increase the accuracy of the system by improving upon the Fisherfaces algorithm or by introducing better training mechanisms to be more robust to different environments and people. Also improvements to the HMI/GUI can be made so that it is easier for the user to save and load faces.

References

[1] D. L. Baggio, "Face Recognition using Eigenfaces or Fisherfaces," in *Mastering OpenCV with Practical Computer Vision Projects*. Birmingham, UK: Packt Publishing, 2012, ch. 8, pp. 231–268.

[2] D. Lee et al., "A Face Detection and Recognition System based on Rectangular Feature Orientation," *International Conference on System Science and Engineering*, 2010.

[3] Face Description with Local Binary Patterns: Application to Face Recognition, T. Ahonen, A. Hadid and M. Pietikäinen, Proceedings of the IEEE Transactions on PAMI 2006, Vol. 28, Issue 12

[4] OpenCV Development Team (2015, Feb 25), Face Recognition with OpenCV [Online]. Available: http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html, [Accessed October 25 2015]

[5] P. Laytner et al., "Robust Face Detection from Still Images," *IEEE Symposium*, 2014.

[6] M. Weng et al., "Remote Surveillance System for Driver Drowsiness in Real-time Using Low-cost Embedded Platform," *IEEE International Conference on Vehicular Electronics and Safety*, 2008.

[7] K. Wang, "Implementation of Face Cartoon Maker System Based on Android," *Fourth International Conference on Intelligent Control and Information Processing*, 2013.