Abstract—In this paper we consider the problem of creating the best group picture by capturing multiple group pictures. We discuss two different ways to go back and forth between different frames of a sequence of images to let the user choose the best face for each person and thus make the picture perfect. The first algorithm corrects for camera motion using homographic transformation and then blends the faces using pyramid blending. The second algorithm uses template matching to find the best location for each desired face on the reference frame. We provide an Android application as well as a desktop implementation of our algorithms.

Keywords—Android; Homography; Pyramid blending; ORB feature matching; Template matching

I. INTRODUCTION

Imagine it’s your child’s graduation. After the ceremony when taking a photo with all their friends, it often doesn’t turn out to be as good as you had hoped. For example, if one of their friend’s mom is taking the photo at the same time, their attention might be divided. Similar situations happen during many important moments of our lives. So to get the perfect picture, we need to capture multiple images, and then select the best faces of each person from amongst them.

II. RELATED WORK

Face detection is the first task in our algorithm. There is a lot of literature available in this field. We used the HAAR Cascade technique for face detection. OpenCV has its own face detection library [1][2]. This paper uses integral image and Haar like basis functions that have very high computational performance. It uses AdaBoost learning algorithm [11] and a cascade structure which greatly improves efficiency.

Next we deal with proper blending of cropped faces with the rest of the image. In existing literature, the image stitching techniques are of two main types. In the first type a curve in found in the overlapped region where the difference between the two images is zero or minimum [13][14]. This seam does not necessarily always exist. The other category of image blending algorithm is based on smoothing transition between the two images to be blend. Example of such an algorithm is feathering where weighted combination of the two image are carried out [12]. Pyramid blending is a similar blending technique, where this weighted combination is carried out at different image scales that takes into account different size objects in the boundary [4]. There are several techniques of finding the weight maps [3][15]. A simple technique is linear transition of weight maps in a direction normal to the edge of the transition boundary. The weight maps are normalized so that the sum of weight maps at each position in the two images is equal to 1.

One assumption made while trying out the above mentioned image blending techniques is that the images are aligned. Image alignment is carried out through calculation of the rotation and translation vectors between the two images giving rise to homographic transformation. To do this we need feature extraction and matching between the two images. Robust features like SIFT [8], SURF [9], ORB [5] are used. Techniques like RANSAC [7] are used to improve the calculation of homography transform parameters [6].

III. ALGORITHM OVERVIEW

Fig. 1. shows the various steps of the algorithm. Our algorithm takes multiple images of a group of people and detects faces using OpenCV’s Haar Cascade frontal face library. These faces are cropped and concatenated, with each row (or column) containing the faces of one person and different rows (or columns) containing different faces. Now the user selects one face for each person in the group which he considers the best to be in the final frame. These faces are stored and blended into the reference image which is the first frame captured.

Two different techniques were used to stitch the images together. First is homographic transformation with pyramid blending where the homography is for aligning the face with the rest of the reference image and the pyramid blending is for proper blending of the aligned images without artifacts. The other technique that was tried was template matching where we found the location of the cropped face in the reference image and just pasted it without any blending. Both these techniques will be described in greater detail later.

IV. METHODOLOGY

We implemented two methods for creating the perfect moments. One approach was using homography and pyramid blending, and the other, template matching. We describe these methods in details in the following sections.

A. Homography and Pyramid Blending

Just pasting the required face on the reference face would work only if there was absolutely no movement of the camera and all the images captured were with same illumination. Apart from these constraints, only the face of any person can move. Nothing else should move, or else it would be seen in the output. We try to avoid all these constraints by first correcting for movement of the camera using homographic transformation and then blending the required face into the reference image using pyramid blending.
Homography transform is generally used to transform one plane to another, and so is used to align images. However, it works much better outdoors than indoors. This is because, for outdoors, almost everything can be approximated to be at infinity and so can be thought of as contained in the plane of infinity. This is not valid for indoors. We cannot assume everything in our indoor photo to be in the same plane and so, ideally homography should not be applied to align the images. However, a person’s body can be approximated to be one single plane and so, for each face, we can align the reference image and the image containing the required face by finding the homography transform between the body of the person in the two images.

Thus, the method works as follows. Given the images, we first detect the faces in the reference frame using OpenCV’s Haar Cascade frontal face detector. For each face detected in the reference frame, we find the same face in all the other frames by narrowing down the ROI in which the Haar Cascade searches for faces. Once this is done, all the faces in all the images are shown together to the user and the user is asked to select the faces he/she wants in the final image.

Upon getting the user input, for each face, we align the frame containing the required face to the reference frame by detecting ORB feature points on the narrowed down ROI of the person’s body in both frames, and match them to get the homography matrix H. This transformation is then applied to each pixel of the frame containing the required face to align it to the reference frame. Fig. 3 shows an example of two frames captured and feature correspondence applied to align the third face.

Upon aligning the two frames, a mask is generated. This mask is a set to 1 in a circular region of the detected face and 0 everywhere else. The two frames are then blend together by first constructing the Laplacian pyramids of the frames and Gaussian pyramid of the mask and alpha blending the two frames at each level with the mask at that level. This process is repeated for all faces in the group to get the final perfect picture.

B. Template Matching

In this technique we use template matching to find the best location of the cropped face and replace the person’s face in the reference image with the new face. As in the previous technique, we use OpenCV’s Haar Cascade frontal face detector to detect the faces of all the people in the images. Then we crop the face regions in each of these images and concatenate them into one matrix so that they can be displayed to the user. Each row has the same person’s face from different images whereas each column has the faces of all people in the same image. The user clicks on the faces he wants on the final image.

Upon getting the user input, the image numbers corresponding to each face are stored and called one by one to find their best location for replacement in the reference frame.

To explain template matching, let’s take an example as shown in Fig. 3. Let there be 3 images captured of a group of 2 people. Let the user select the first person’s face from image 3 and the second person’s face from image 2 respectively. Both these faces have to replace the corresponding faces in image 1 which is the reference image. For the first person let her face center be at location (x,y) in the 3rd image. In the reference frame, we move the cropped face block to be centered at location ±40 around (x,y) location in the horizontal and vertical direction. At each of these locations we check where is the minimum difference of the boundary region of the face block and the boundary region of the specific portion of the background reference image with the same size as the cropped face.
block and with center location determined by the process described earlier. The best location at which to paste the replacement face is given by the location at which these differences between the boundaries of the two blocks are the minimum. This process is repeated for all faces in the group to get the final perfect picture.

V. ANDROID IMPLEMENTATION

We made an android application for capturing the perfect moments using either of the methods discussed in the previous section. The code for the application and its video demonstration are available in the supplementary information. The Android application allows the user to capture two group photos, or use the preloaded images, to run either of the two methods. Upon capturing the images, the application returns all the faces across all the images and the user is asked to select which face of each person he/she wants in the final image. Upon tapping on the desired faces, the user clicks on done in the options menu and the application returns the final result image with all the desired faces.

Apart from this, we have a desktop implementation of each of the two methods, which works for any number of images. They too have user interfaces for the user to select which face he/she wants of each person in the final image. These codes are also available in the supplementary information.

VI. RESULTS

Our results using both the methods are shown in Fig.5,6. The first two columns show the two frames captured, the third column shows the results of the Homography and pyramid blending method and the last column shows the Template matching results.

For comparison between the two methods, A/B test was performed. Around 100 random people were asked to choose between the outputs of the two algorithms. The results of A/B are shown in Fig.4. The results show that homography and pyramid blending is better than template matching.

Fig.7 shows the results using the desktop implementation of the two methods. The desktop version can take any number of images as input and perform the same process to get the final perfect pictures.

VII. DISCUSSIONS

Both the methods depend upon correct faces being detected by the Haar cascade classifier. Sometimes, the classifier does not detect a face or detects false faces. In such cases, either of the two methods might not produce correct results, or worse, crash. The first method normally produces better results than the second, as the A/B test results suggest. The use of pyramid blending helps avoid artifacts observed in the second method’s output due to change in illumination between the two images captured (Fig.5, rows 7, 9). However, the success of this method largely depends upon the correct homography being calculated and so might give random results if calculated homographic transform is wrong (Fig.5, row 10). The technique of template matching works better in such cases.

The first method tries to align each person’s body in the two images and then blend the two images. This results in a more natural looking foreground (the person), while in the background we could observe some artifacts. The second method tries to fit the face rectangle by minimizing the error at the edges. The background contributes about 75% to this and so, if the illumination is same, no artifacts are observed in the background, while foreground might have some artifacts. Fig.5, rows 4,6 help explain this better.

Also, for template matching we need a 40 pixel margin surrounding all detected face bounding box, failing in which the system crashes. This is usually not a major issue as people are generally centered in photo frames.

ACKNOWLEDGMENT

We would like to thank Dr. Peter Vajda for his guidance and constant encouragement, and Saumitro Dasgupta for providing a starting point in our Android code. We would also like to thank Prof. Girod, and David and Huizhong Chen for everything.
Fig. 5. Results: The first two columns show the input images captured, the third column shows the results of homography and pyramid blending, and the last column show the results of template matching.
Fig. 6. Fig. 5. contd.

Fig. 7. Desktop implementation of the two methods using five images. The left result is using Homography and pyramid blending, and the right using the template matching.
REFERENCES


APPENDIX: CONTRIBUTIONS

- Ayesha: Android implementation, Homography and Pyramid Blending code
- Pallabi: Template matching code
- Joint efforts: Report, poster, proposal, video demo, brainstorming