FriendBlend

Kevin Chen[1,3], David Zeng[1,3], Jeff Han[2,3]

***Abstract*— FriendBlend is a mobile application that merges two portraits of different people to create a single, multi-person photo. To do this, Person A takes a photo of Person B, and Person B takes a photo of Person A with the same background. Given these two input images, our goal is to create a third image with both Person A and Person B in the photo together. We first color correct the photos and then register the images by performing keypoint matching to estimate the homography. We then blend the images either by alpha blending or by GrabCut segmentation, depending upon the relative locations of the human subjects in the two images. Limitations in the underlying techniques can affect the robustness of the application, but in the majority of cases FriendBlend works quite well.**

I. INTRODUCTION

A self-portrait photograph ("selfie") is convenient since everyone can be included in the photo. However, the composition of the photo is wanting. For a properly composed photo, the photographer usually cannot be included. Our project intends to address both of these issues.

Our project requires two pictures. The first picture contains everyone except for the photographer and the second picture is the photographer alone. Our goal is to seamlessly add the photographer to the original image to achieve a properly composed image with everyone.

The core image processing relies primarily on registration and segmentation techniques. Our project works well for many cases but also is also limited by pathologies that it inherits from its underlying methods. Furthermore, our target is mobile devices and this also constrains the available techniques. We address some issues with our own heuristics but it is still apparent that for more robust operation, additional research or computing power is required.

II. ALGORITHM

A. Color Correction

It is crucial to make sure that the lighting from the two input images are approximately the same before the blending process. This ensures that there are no obvious artifacts in the resulting output image due to dramatic lighting differences between the two input photos. To address this issue, we perform contrast limited histogram equalization in the Lab color space on the lightness channel to preserve hue [1].

[1]Enrolled in EE 368/CS 232 Digital Image Processing

[2]Enrolled in CS 231M Mobile Computer Vision

[3]Department of Electrical Engineering, Stanford University, Stanford, California

B. Face and Body Detection

Locating the subjects in the pictures is important for determining which image merging process to use. To find where the humans are located in the images, we use face detection by Haar feature-based cascade classifiers [2], [3]. Using the location and size of the face, we can then determine a bounding box for the body of the human. We use a cascade of boosted classifiers based on Haar-like features to extract a bounding box for the face of each person. The bounding box for the body is determined from the following formulas:

$$x_{left}^{body} = x_{left}^{face} - w \tag{1}$$

$$x_{right}^{body} = x_{left}^{face} + 2 \cdot w \tag{2}$$

$$y_{top}^{body} = y_{top}^{face} - h \tag{3}$$

$$y_{bottom}^{body} = height(image) \tag{4}$$

In the equations above, w and h are the width and height of the face bounding box, and $height(image)$ is the height of the image. This size for the body bounding box was determined empirically and works well for most cases.

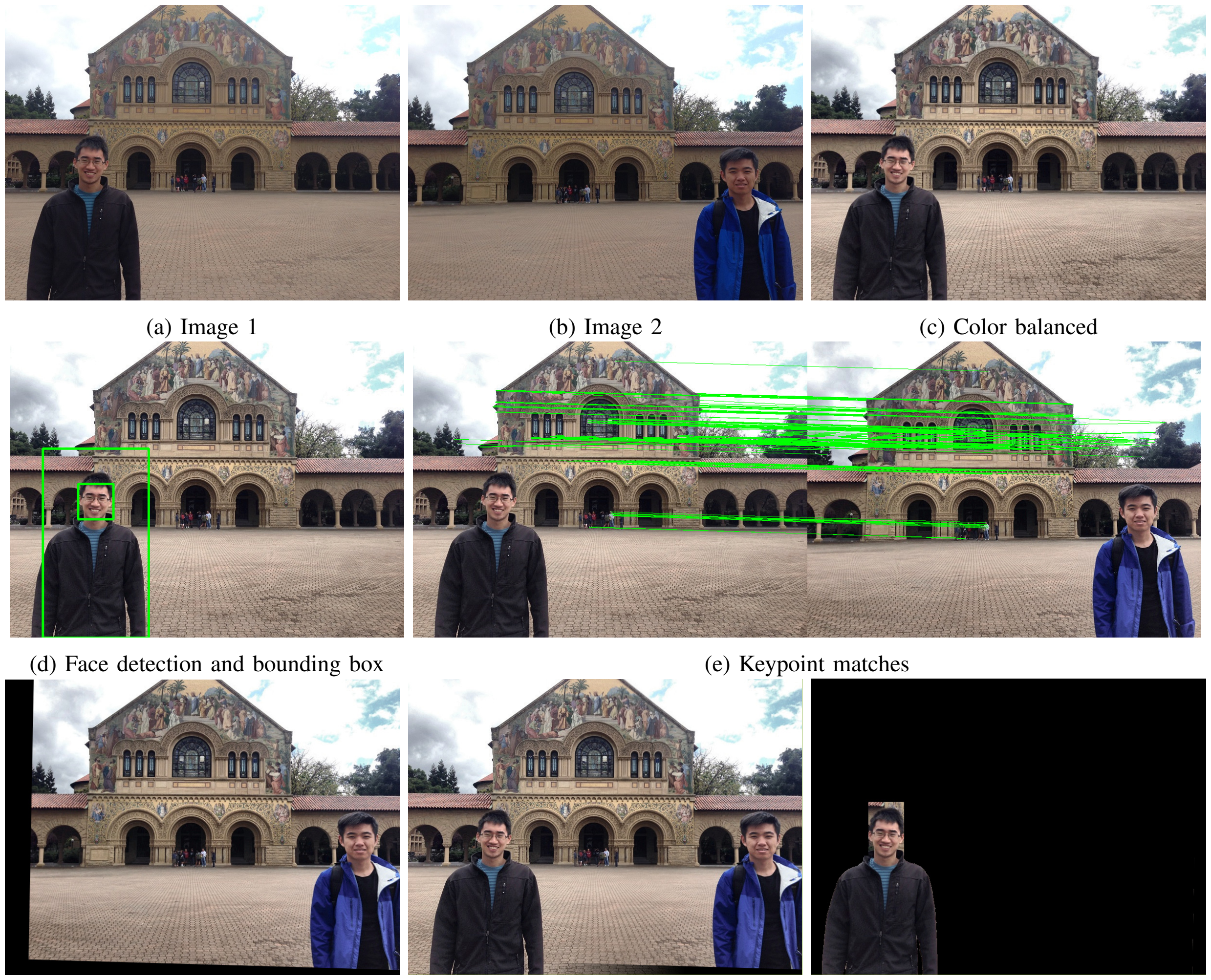
C. Homography Estimation

We compute a homography to warp one of the images so that the perspectives from the two photos are the same prior to merging. Although the scene is not planar, the background is assumed to be far away from the camera center so that a homography will be sufficient for our purposes. In practice, this turns out to work very well.

1) Keypoint Detection: FriendBlend uses Oriented FAST and Rotated BRIEF (ORB) keypoint detection [4] because of the balance in efficiency, performance, and monetary cost. ORB keypoint detection is similar to SURF but also includes a modification to account for rotation invariance, and the keypoint descriptors are essentially BRIEF descriptors for rotated patches around the keypoints.

2) Keypoint Matching: Keypoints in the two images are matched by Hamming distance, which is recommended for ORB descriptors [4]. A pair of keypoints, one from each image, are said to be matched if the keypoints are within a certain threshold Hamming distance from each other. In our implementation, we set the threshold to be 10 times the distance between the closest keypoint match.

Furthermore, we also prune our matches based off of keypoint locations. Since each person is in each image once, any keypoints on the person should be rejected. Keypoints from the face or body in one image should not be found in the other image. Therefore, we prune a keypoint match if any of the keypoints are within the bounding box of a person.


(a) Image 1 (b) Image 2 (c) Color balanced

(d) Face detection and bounding box (e) Keypoint matches

(f) Homography (g) Alpha blending (h) Grabcut result

Fig. 1: FriendBlend Processing Flow

3) Computing the Homography: Once we have found a good set of keypoint matches, we use RANSAC [5] to find the homography that best warps the images into the same perspective.

D. Image Blending

FriendBlend uses two techniques to merge the images depending on whether the subjects in the output photo should be close together or far apart. If the subjects are far apart, we run an alpha blending technique. If the subjects are close together, then we first find a rough segmentation of the human and then paste the cropped subject into the other image.

1) Subjects Far Apart: Since the subjects are far apart, we can just blend the area between the subjects together and then crop the merged image. To do this, we perform alpha blending in each color channel separately. The blending begins at the rightmost edge of the bounding box for the left subject (call this edge at $x = colStart$), and the blending ends at the leftmost edge of the bounding box for the right subject (call this edge at $x = colEnd$).

$$stepSize = \frac{1}{colEnd - colStart} \tag{5}$$

$$\begin{aligned} i^{result}(x,y) = (1 - stepCount * stepSize) * i^{left}(x,y) \\ + (stepCount * stepSize) * i^{right}(x,y) \end{aligned} \tag{6}$$

Here, $stepCount$ is an iterative variable defined by $stepCount = x - colStart$ when x is between $colStart$ and $colEnd$ (inclusive). Within a single column of the image, every pixel element is blended the same amount. This blended image is the final output image.

2) Subjects Close Together: If the subjects are close together, then one person needs to be segmented on top of the other person so that it looks like one person is standing in front of the other. To determine which subject is in front, we compare the sizes of the face bounding boxes of each person and set the person corresponding to the larger face to be in the foreground.

Once we have determined which subject is in front, we need to crop the person in the foreground. We can obtain a fairly reasonable crop using GrabCut [6]. We set the body bounding box as input to GrabCut along with predictions of probable foreground pixels and definite foreground pixels. The pixels within the face bounding box are set to definite foreground pixels and the area below the head is set to probable foreground pixels since it likely contains the body.

Using the results of GrabCut, we extract a mask to crop out the person in the image. Next, we perform erosion on the cropped person using a 3x3 ellipse-shaped structuring element to get rid of any artifacts along the edge of the crop. Finally, we overlay these pixels on top of the background image to produce the final result.

III. Method

We implemented FriendBlend in both Python and C++. Both implementations rely on OpenCV [7], and face detection is based off of a pre-trained Haar feature-based cascade classifier from OpenCV. We also made the assumption that there is exactly one face in each input image. If multiple faces are detected, then FriendBlend will select the largest face.

In our implementation, we discard keypoint matches based off the Hamming distance between two descriptors and based off the keypoint location. To discard a keypoint match based off distance, we first calculate the smallest distance between any keypoint descriptor in the first image and any keypoint descriptor in the second image, which we call $dist_{smallest}$. Any keypoint match that has a distance greater than $dist_{cur} = 10 * dist_{smallest}$ is removed from the set of keypoint matches.

Furthermore, after we perform alpha blending for two far subjects, we make sure to crop the final image. The warping from the homography can cause the bounds of the image to expand. For example, a warped image can have a larger height than the original image. Therefore, we crop the image so there are no undesired artifacts in the final image.

When running GrabCut, we calculate a different-sized bounding box for the body. Rather than using equations (1) through (4), we use the following:

$$x_{left}^{body} = x_{left}^{face} - 2 \cdot w \tag{7}$$

$$x_{right}^{body} = x_{left}^{face} + 3 \cdot w \tag{8}$$

$$y_{top}^{body} = 0 \tag{9}$$

$$y_{bottom}^{body} = \text{height}(image) \tag{10}$$

Lastly, in order to reduce the runtime and save storage space (which is especially important for the mobile computing), we make sure to downsample the images so that the longest edge is at most 1000 pixels long.

Using a Java wrapper for our native C++ OpenCV code, we were able to run FriendBlend on Android which has a reasonable runtime even on a mobile device. We tested the Android application on an NVIDIA Tegra Tablet and an HTC One.

IV. Results and Discussion

We tested FriendBlend on an NVIDIA Tegra Tablet and an HTC One in many different environments, as shown in Figure 2. In Figure 2(a-c), the two subjects are on opposite sides of the image so alpha blending is utilized for the combining the images. We see that the processing works well for these input images and the output image convincingly appears that it is a single photo of two people. In figure 2(d-e), the two subjects overlap in the image so GrabCut segmentation is used. The results are again convincing and the seams have been blended well. Figure 2(f-h) also has overlapping subjects. There is a cropping issue present but the image seems reasonable otherwise.

From the results, we can see that when all of the methods work as intended, the image looks reasonably convincing that both people were originally in the image. We identified two significant pathologies that can cause our algorithm to fail.

The first pathology occurs when the Viola-Jones face detector fails to recognize any faces within either image. Our registration and segmentation techniques are based upon knowing the location of the subjects face and therefore body to properly identify keypoints, calculate the homography, and segment the subject. If we cannot identify a face in each image, then our heuristics and assumptions are not likely to hold, leading to the failure of the process. More robust face detection algorithms exist, such as [8], but they require significantly more computational capacity than the Viola-Jones detector.

Figure 2(j-l) has two subjects far away so alpha blending is used. The result is still reasonable but does not look as convincing. The homography is not quite right and the subject on the left looks like he is leaning forward. We can see that most of the keypoints are likely to come from the buildings in the background. However, the heads of the two subjects are occluding the buildings and thus we will discard those keypoints, significantly reducing the number of usable keypoints. With fewer quality keypoint matches, it makes sense that the homography would not perform as well.

The other significant pathology occurs when we cannot maintain enough keypoints after pruning. It is not unusual for a person to occupy one-third of an image. When we consider two images, potentially two-thirds of the image must be ignored for keypoints. This leaves only a third of the image for keypoints which are then pruned. Sometimes no keypoints survive RANSAC or the keypoints that do survive are poor matches which lead to a poor homography.

Our segmentation method is moderately fast but does not produce an exact segmentation of the person. There exist better segmentation methods such as [9] and [10]. However, these are even more computationally expensive than GrabCut.

Figure 2(m-o) has two subjects close together so GrabCut is used. In the result, the artifact of the right boundary of the display case is most noticeable. This is from the failure of the assumption that the background can be treated as planar in the homography. If the segmentation were more precise however, only segmenting the subject, then the failure of the homography would not have been as apparent.

Mobile devices are known for their computational limits and these constraints lead to challenges in developing a robust application. A feasible solution is to offload the computation to a dedicated server and transfer only the initial and final images. In the context of a mobile application, it remains to be seen how responsive this computational flow would be for user experience.

A future extension of this project would be to handle more than two images and continuously composite the new image to the existing images to create images of arbitrary size and shape. Another extension would also be to generalize this algorithm for multiple faces rather than just two.

ACKNOWLEDGMENT

We would like to acknowledge Professor Bernd Girod, Professor Gordon Wetzstein, and Professor Silvio Savarese for their instruction.

WORK DIVISION

- Kevin Chen: Developed some image algorithms in Python. Python and C++ debugging. Data collection. Worked on the poster and written report for EE 368.
- David Zeng: Developed some image algorithms in Python. C++ implementation. Data collection. Worked on the poster and written report for EE 368.
- Jeff Han: Developed some image algorithms in Python. Developed Java wrapper and Android interface. Data collection. Worked on the report for CS231M.

REFERENCES

[1] S. Naik. Hue-preserving color image enhancement without gamut problem, IEEE Trans. on Image Process., vol. 12, no. 12, pp. 1591-1598, 2003.

[2] P. Viola. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.

[3] R. Lienhart. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[4] E. Rublee. ORB: An efficient alternative to SIFT or SURF. ICCV 2011: 2564-2571.

[5] M. A. Fischler and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM, vol. 24, no. 6, pp. 381395, 1981.

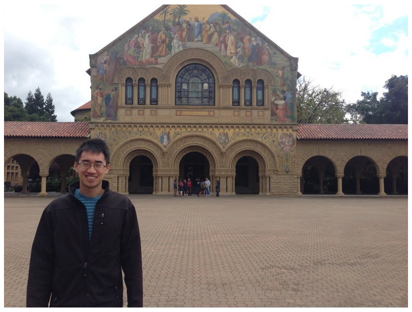
[6] C. Rother, V. Kolmogorov, and A. Blake, GrabCutInteractive Foreground Extraction Using Iterated Graph Cuts, ACM Trans. Graphics, vol. 23, no. 3, pp. 309-314, 2004

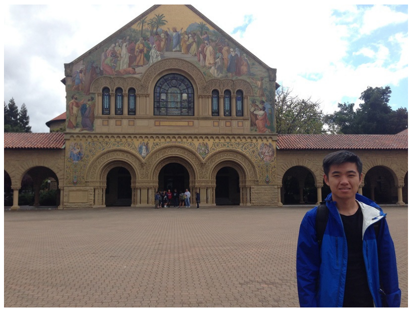
[7] G. Bradski, OpenCV, Dr. Dobb's J. of Software Tools, 2000

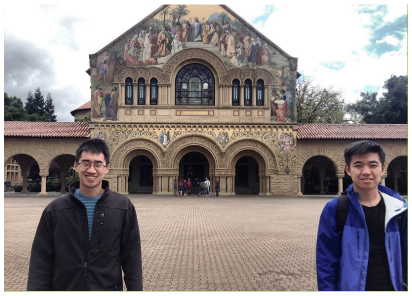
[8] S. Farfade, M. Saberian and L. Li, 'Multi-view Face Detection Using Deep Convolutional Neural Networks', ICMR 2015, 2015.

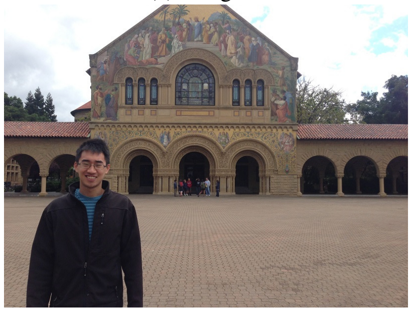
[9] P. Duygulu, 'Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary', in 7th European Conference on Computer Vision, Copenhagen, 2015.

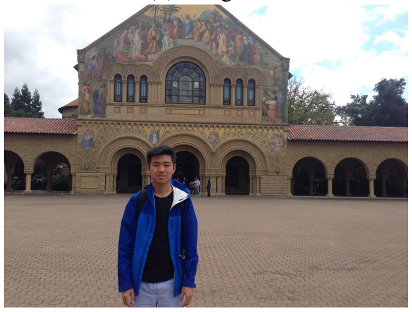
[10] C. Juang, 'Computer Vision-Based Human Body Segmentation and Posture Estimation', in BTAS, Crystal City, 2007.

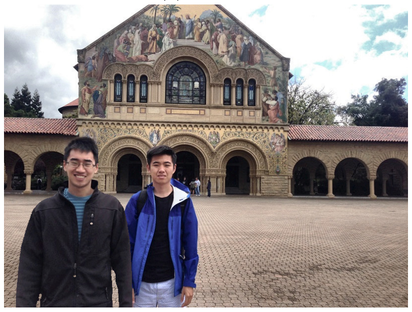

(a) Image 1

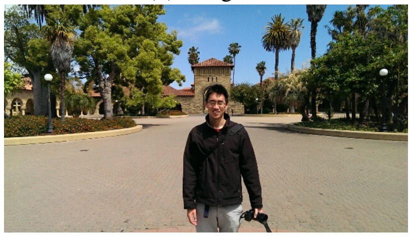

(b) Image 2

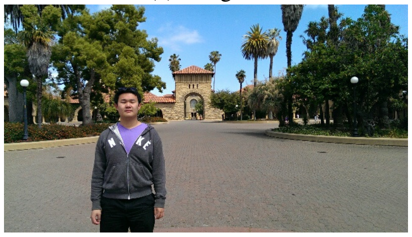

(c) Result

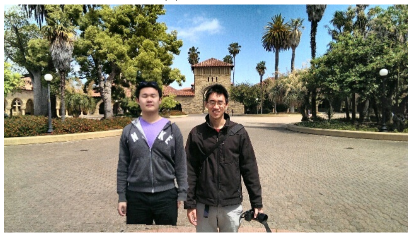

(d) Image 1

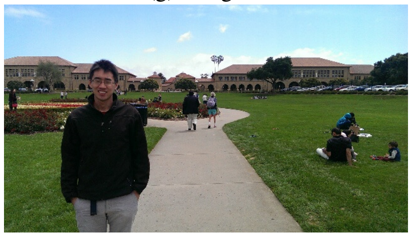

(e) Image 2

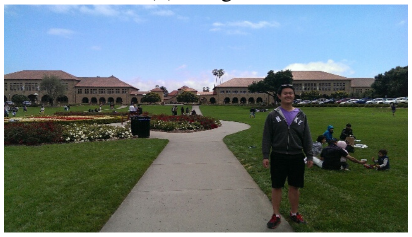

(f) Result

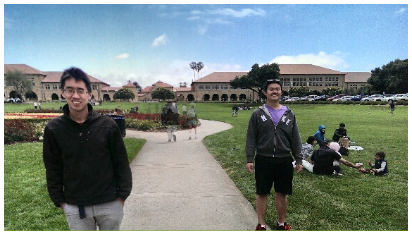

(g) Image 1

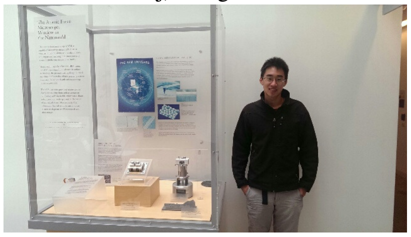

(h) Image 2

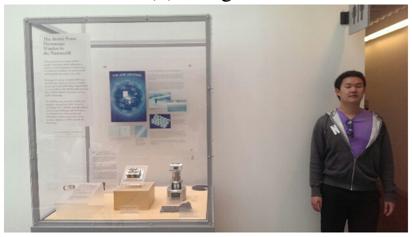

(i) Result


(j) Image 1


(k) Image 2


(l) Result


(m) Image 1


(n) Image 2

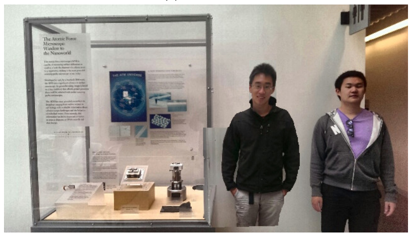

(o) Result

Fig. 2: FriendBlend Results