FRIEND-BLEND

Rudrabha Mukhopadhyay [*], Sangeeth Reddy [*]

November 2018

1 Introduction

The main goal of this project was to implement [1]. Friend-Blend is an 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.

Figure 1: Inputs (Left and Center); Output (Right)

The major challenge of this project is to align both of these pictures properly with each other. It is also important to make sure that the background of both the images are similar to avoid undue artifacts. We combine both of these images using two different techniques. They are alpha blending [3] and grab-cut algorithm [4].

[*] Enrolled in CSE/ECE 478, Monsoon 2018 at IIIT Hyderabad

We discuss these methods and steps involved in greater detail in the Methodology section. One of the example results from our system is given in Figure 1.

2 Methodology

The pipeline of our application is shown in Figure 2. We describe each of the stages of our pipeline in detail in this section.

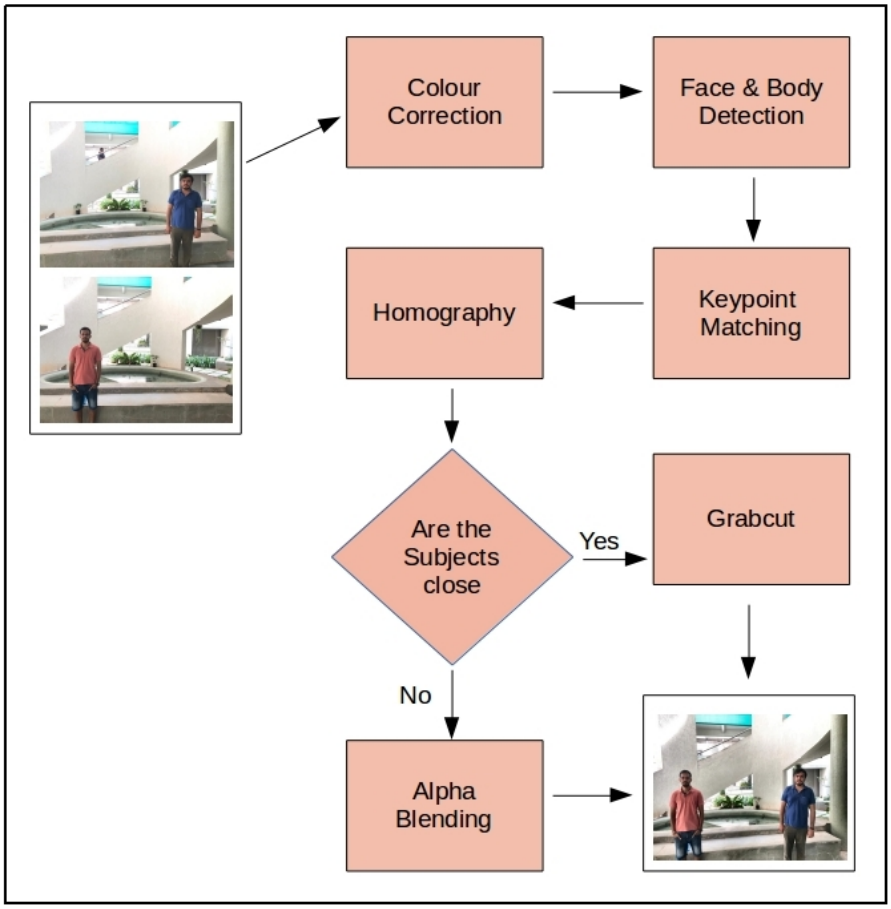

Figure 2: Pipeline of FriendBlend

2.1 Colour Correction

It is important to make sure that the background lighting of the two input images are similar. This ensures that blending these two images will not create additional artifacts due to colour differences. We perform contrast limited histogram equalization in the Lab color space on the lightness channel to preserve hue [**?**]. This ensures that both the images have similar lighting. An example of colour corrected output is given in Figure 3.

Figure 3: Original Image (Left), Colour Corrected Image (Right)

2.2 Face & Body Detection

We have to locate the positions of the human bodies present in the image. For detection of a human face we use a Haar feature based cascade classifier [2]. 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. We then choose the left or right most face and try to determine the bounding box for the body of the human. Bounding box for the body is determined by the following equations.

$$x_{left}^{body} = x_{left}^{face} - 1.5 * w \tag{1}$$

$$x_{right}^{body} = x_{left}^{face} + 2.2 * w \tag{2}$$

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

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

Here w and h are width and height of the detected face respectively. These equations were found empirically.

Figure 4: Detected face and body

2.3 Keypoint Detection and Matching

Detection: We use Oriented FAST and Rotated BRIEF (ORB) keypoint descriptors [6] in this project. 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. We detect a

total of 10,000 keypoints in each image.

Matching: Keypoints in the two images are matched by Hamming distance. 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 the original implementation, the threshold was set to be 10 times the distance between the closest keypoint match. However, we find that the authors of the original project had used images with similar backgrounds without much variation. As we try our algorithm on images with varying background setting, we find that this threshold often results in false keypoint matches which affects homography badly. Thus, we change this, we set the threshold according to the following rule,

Algorithm 1: How to take keypoint matches

```
Remove all keypoint matches present within the bounding boxes of the
  persons present in the image.
if total no. of matches <= 20 then
    Take top 20 keypoints matches
else
    Take top 40 keypoint matches
```

Figure 5: Keypoints matched between two images

We plot all the top 40 keypoint matches for two images in Figure 5.

2.4 Homography

Using the top k keypoint matches we use RANSAC [5] algorithm to perform homography. We first find the homography matrix using the keypoint matches. We warp the image using the calculated homography matrix. This properly aligns both the images. With both of these images properly aligned we can then use either of Alpha Blending or Grabcut algorithm to merge them together to get the final picture.

Figure 6: Realigned using homography

Figure 6 shows us resultant image after homography.

2.5 Blending of Two Images

Once we have aligned the images properly, we blend both of them. We use two techniques depending on the position of the humans in the image. They are Alpha Blending [3] and Grabcut [4]. We discuss both of these algorithms one by one in the next section.

2.5.1 Alpha Blending

We use this technique when the humans relative positions are far apart from each others in the images. This gives us an opportunity to simply blend the area between the humans. We perform alpha blending on each of the RGB colour channel separately with the following algorithm.

Algorithm 2: An algorithm to perform alpha blending in FriendBlend

1 $colStart$ = Rightmost edge of the bounding box for the left subject.
2 $colEnd$ = Leftmost edge of the bounding box for the right subject.
3 $stepSize = \frac{1}{colEnd - colStart}$
4 **for** *i in range(colStart, colEnd)* **do**
5 $\quad stepCount = i - colStart$
6 $\quad resImg[i^{th}column] =$
$\quad (1 - stepCount * stepSize) * leftImg[i^{th}column] + (stepCount *$
$\quad stepSize) * rightImg[i^{th}column]$

In Algorithm 2, leftImg is considered to be the image with the subject on the left side while rightImg is considered to be the image with the subject on right side. This algorithm is applied separately to each channel of the image to get the desired output.

2.5.2 Segmentation using Grabcut

If both the subjects positions are very close to each other, then the region between the subjects is very limited to perform alpha blending, which does not not provide a visually appealing output. Thus we use a segmentation technique to segment out a human from one of the images and then blend it into the other. The technique we use for segmentation is Grabcut. We choose the larger of the two subjects for segmentation. 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.

Figure 7: Input Image (Left), Segmented Image using Grabcut (Right)

3 Experiments & Analysis

Several experiments have been conducted with various input images clicked under different background conditions. This enabled us to extensively test and analyze the robustness and limitations of the algorithm. Although the algorithm performs extremely well for images with well aligned background and slight deviations, It has limitations for huge variations or changed perspective camera angles for the input images. Though these limitations are highly dependent on the underlying techniques we try to over come them with possible alternates. Lastly, we also perform a user based study to prove the effectiveness of the algorithm. We discuss these one by one in this section. We implemented FriendBlend in Python. Our implementation extensively uses OpenCV for importing different functionalities.

3.1 Visual Results

We take pictures with varied backgrounds and at different camera angles to check the robustness of our algorithm.[1]. We pass these images one by one through our algorithm. Figure 8 and Figure 9 contains all the results that uses alpha blending to create the final image containing both the subjects. Figure 10 contains results from the grabcut algorithm.

[1] Our dataset can be found in https://drive.google.com/open?id=1lWpdcrhrv1DcAVf0-JwX0O5ze4V43g31

Figure 8: Some Results using Alpha Blending

Figure 9: Some Results using Alpha Blending

Figure 10: Some results using Grabcut

3.2 Limitations of the algorithm

Our algorithm is limited in certain ways. As both of the images are first aligned by performing homography hence it is dependent on how good the keypoints are matched. Our algorithm often fails when proper keypoints are not detected.

Figure 11: Failure case 1 due to inadequate keypoint matches

In Figure 11 we can see that both of these images fail to blend properly. To understand more, we plotted all the keypoint matches in Figure 12.

Figure 12: Top 40 keypoint matches

As we can see from Figure 12 all the top 40 keypoints are detected near the bottom of the image near the car number plate. Therefore when we perform homography we get a misaligned image. This problem can be mitigated by considering more number of keypoint matches.

Figure 13: Top 100 keypoint matches

In Figure 13 we consider top 100 keypoint matches instead of 40. As we can see, now we have keypoint matches near the top of the image as well. This leads to much stable homoography computation and thus the final blended result given in Figure 14 improves.

Figure 14: Blended result after considering top 100 keypoint matches

But we could not generalize the usage of top 100 keypoint matches over all im-

ages. We also found that in case the input images are aligned very differently then the algorithm fails to properly blend them. We show such cases in Figure 15.

Figure 15: Example of limitations of the algorithm to merge two highly misaligned images

As we use grabcut algorithm for segmenting out a human figure from the image we end up inheriting all its limitations. We report them in Figure 16. As we can see, grabcut algorithm often fails to differentiate between the foreground and background pixels and hence segment out unwanted parts. We tried to address this problem by tuning the parameters involved but we donot get any significant improvement.

Figure 16: Some failure cases of Grabcut algorithm

3.3 User Based Study

As the results from our system are generated for human consumption therefore the evaluation is subjective. Hence, we conducted a user-based study to evaluate the outputs of our method. This allowed us to get an average subjective evaluation. For doing this user based study we show 10 random results from our algorithm to 30 users and ask them to rate images based on how natural/real they look on a scale of 1 to 5 where 1 being poorest and 5 the best . We report the results from the user based study in Figure 17.

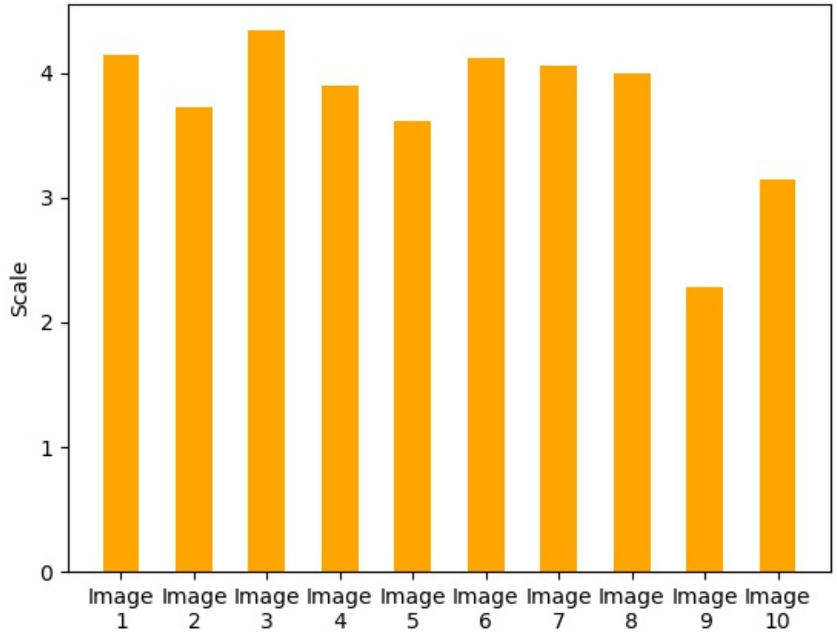

Figure 17: Mean Score for the 10 images used in the user based study

The mean score for all of these images is **3.73** with a standard deviation of **0.57**.

4 Scope for Future Work

There is major scope for improvement in FriendBlend. GrabCut algorithm often fails to properly segment out the human body. Hence, one obvious improvement would be to use some modern deep learning methods for human body segmentation. As our application heavily relies on keypoint matches thus using a better descriptor which gives better keypoints would be useful.

5 Work Division

Stage	**Rudrabha (%)**	**Sangeeth (%)**
Colour Correction	30	70
Face and Body Detection	70	30
Keypoint Matching	80	20
Homography Computation	80	20
Alpha Blending	20	80
GrabCut	20	80
Report	80	20
Presentation	20	80

Table 1: Work Division between Rudrabha and Sangeeth

Though at several stages of the project variety of different issues surfaced where the contributions were made equally by both the authors irrespective of assigned task.

The link for our project in github is *https://github.com/Rudrabha/FriendBlend*

6 Acknowledgements

We have used tutorials present in

https://docs.opencv.org/3.0-beta/doc/py_tutorials/

We have taken different tutorials present in this site and then modified them to fit

into our problem.

References

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

"FriendBlend".

https://web.stanford.edu/class/cs231m/projects/final-report-han.pdf

[2] P. Viola,

"Rapid Object Detection using a Boosted Cascade of Simple Features." IEEE CVPR, 2001.

[3] Porter Thomas, Tom Duff,

"Compositing Digital Images". Computer Graphics. 18 (3): 253–259. doi:10.1145/800031.808606. ISBN 0-89791-138-5.

[4] 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*

[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] E. Rublee.

“ORB: An efficient alternative to SIFT or SURF.” in ICCV 2011: 2564-2571.