FriendBlend

Jeff Han (CS231M), Kevin Chen (EE 368), David Zeng (EE 368)

Abstract

In this paper, we present an android mobile application that is capable of merging two images with similar backgrounds. The goal of the application is to enable two people to be present in an image without forcing either person to ask a stranger to take the picture for them and removing the necessity to take a "selfie". In normal device operation, users capture two images with the mobile devices camera, one image for each person. The two images are then passed through the FriendBlend algorithm and the merged image is returned to the users without any other interaction. The merging algorithm is able to, with a high degree of aesthetic confidence, deal with cases where the people in the images are non-overlapping or overlapping. The application is designed and optimized for the android operating system and was tested successfully on both a NVIDIA Shield tablet and a HTC One smartphone.

Section 1: Introduction

The idea of FriendBlend originated from the annoying experience of not being able everyone in a group be in a picture because at least one person needs to operate the camera. The commonplace solution to this problem is to ask a stranger to take the picture. But this approach is suboptimal because it is awkward and if the image does not turn out well, it is even more awkward to ask the stranger to retake. Another workaround to this problem is to take a group selfie. Such an image almost never turns out well and fails when the background is important to the context of the image (poor image field of view).

The solution is to alternate photographers and merge images together (Figure 1).

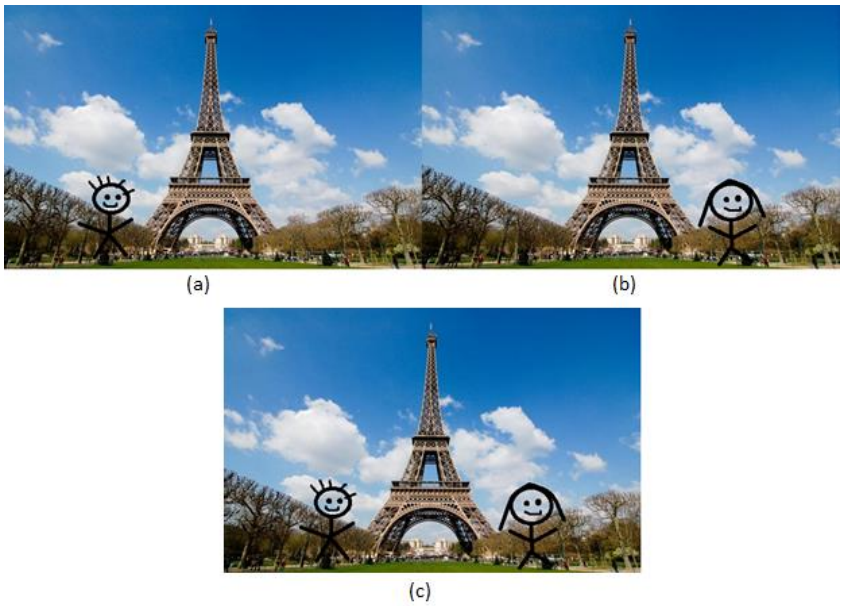

Figure 1: (a) Alice takes a picture of Bob. (b) Bob takes a picture of Alice. (c) FriendBlend does the rest.

In this project, we focus on merging pictures for exactly two people. However, this application can be further extended to make it possible to append any object into any picture.

Section 2: Background

Section 2.1: Previous Work

Many image blending algorithms have been developed that enable the merging of images with different subjects and environments. In the most manual approach, a picture editor with access to Adobe Photoshop can use a segmentation tool called "Intelligent Scissors" to extract image parts (or sprites) and merge multiple sprites together by placing them in different layers[1]. This method, however, does not take into consideration the relative lighting of the two images being blended together which leads to obvious edges in the output image. It also relies on a human to align images together which is a labor intensive process. Several methods have been established for the blending of images to reduce edge effects. These include alpha blending, seam carving and pyramid blending. These work well in areas where features are relatively constant in an image but lead to ghosting and produce other artifacts when images are not completely aligned. Poisson image editing[2] and image cloning[3] are more advanced image blending methods. However, the former requires human aided guidance and the latter is quite computationally intensive.

Section 2.2: Key Contributions

In this work, we focus on a completely autonomous image merging algorithm for mobile platforms. We focus on clean user experience because the user interface elements on mobile devices are much more primitive than those available to desktop software. The algorithm we present is able to provide aesthetically pleasing image blends without any user aid except for the initial image capture. The application is also designed to be as responsive as possible by minimizing image processing time requirements. All the image processing is done locally on the mobile device and the processes the image blending algorithm undergoes is optimized for speed.

Section 3: Technical Details

Section 3.1: Terminology

Below is a list of terms and definitions used throughout the paper:

Image Subject	The person in the image.
Seed Image	An image with one image subject, meant to be merged with another seed image.
Foreground Image	The seed image that will be blended in front of the other seed image.
Background Image	The seed image that will be blended behind the other seed image.
Merged Image	The image that is the result of merging the two seed images.

Section 3.2: Summary of Technical Solution

The main stages of this algorithm can be divided into image pre-processing, image merging, and image post-processing. In image pre-processing, the users take the two pictures used as input to the algorithm. The images are then color normalized to make sure that they will be similar in lighting when merged. The image merging stage, the image subjects in the two seed images are determined by facial recognition using Haar cascade classifiers[4]. Then, ORB keypoint matching and perspective warping is done to align one seed image with the other. If the image subjects overlap in position, GrabCut[5] is used to segment out the image subject from its respective seed image and blended with the second seed image with edge erosion. If the image subjects do not overlap in position, alpha blending is used without GrabCut. Lastly, in the image post-processing step, the output image is cropped and displayed to the user.

Figure 2 below is a schematic of the FriendBlend image merging algorithm.

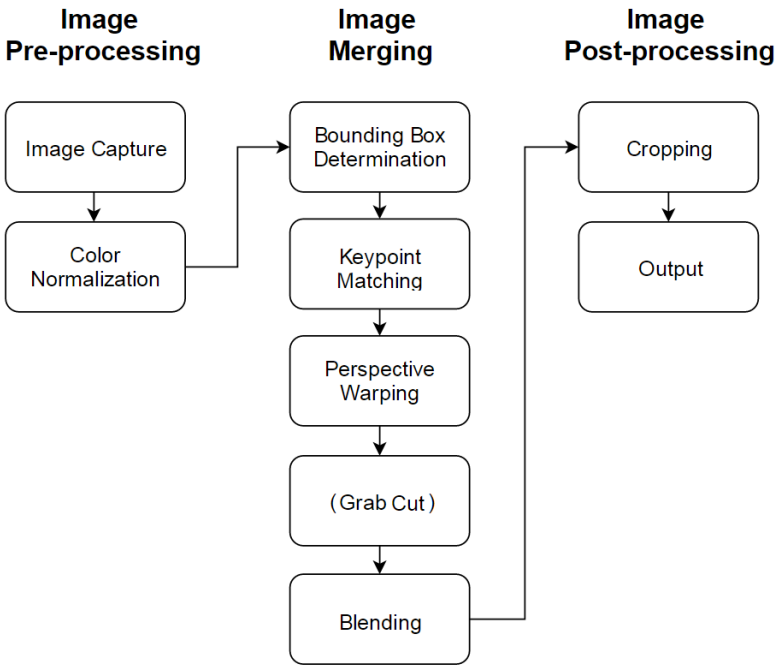

Figure 2: Schematic of FriendBlend Algorithm

Section 3.3: Technical Solution

Section 3.3.1: Image Pre-processing

Image Capture
The input images are captured by the target devices camera, a necessary hardware component for this application to function completely. These seed images are taken serially: one image subject stands in the camera view while the other takes the image. Then the subject who took the picture swaps places with the initial image subject. These two seed images are stored in the device and used for blending.

It is important to note, that for this algorithm to work well, the image subjects in the seed images should not have any occlusions to their face such as hats or large sunglasses. These types of accessories make facial detection less robust and thus threaten the algorithm flow. Furthermore, it is best if there are no other people prominently displayed in each of the seed images. If there are, the algorithm could assume another person is the intended image subject and blend them into the final result. Another consideration that must be taken into account is the size of the image subjects with respect to the frame. The larger the image subjects are, the less background area there will be for keypoint detection. This makes it difficult to find an accurate projection between the perspectives of the two seed images. On a similar vein, the background of the seed images should contain some structure or environment that contains edges or blobs, not just a homogenous surface. This provides a healthy number of keypoints for matching.

Color Normalization

After the input images are captures, they are color normalized to provide better blending results. If the lighting conditions are significantly different between the two seeds, the merging of one on top of the other produces a resulting image that looks like it was obviously edited. To color normalize, we used a contrast limited histogram equalization[6] (CLAHE) approach that normalized the lightness channel in each seed image. We first convert the input image from RGB to LAB color space and then, pixel by pixel, apply the adaptive histogram equalization on 4x4 pixel tiles, and finally convert the image back to RGB. The result of such normalization is shown in Figure 3.

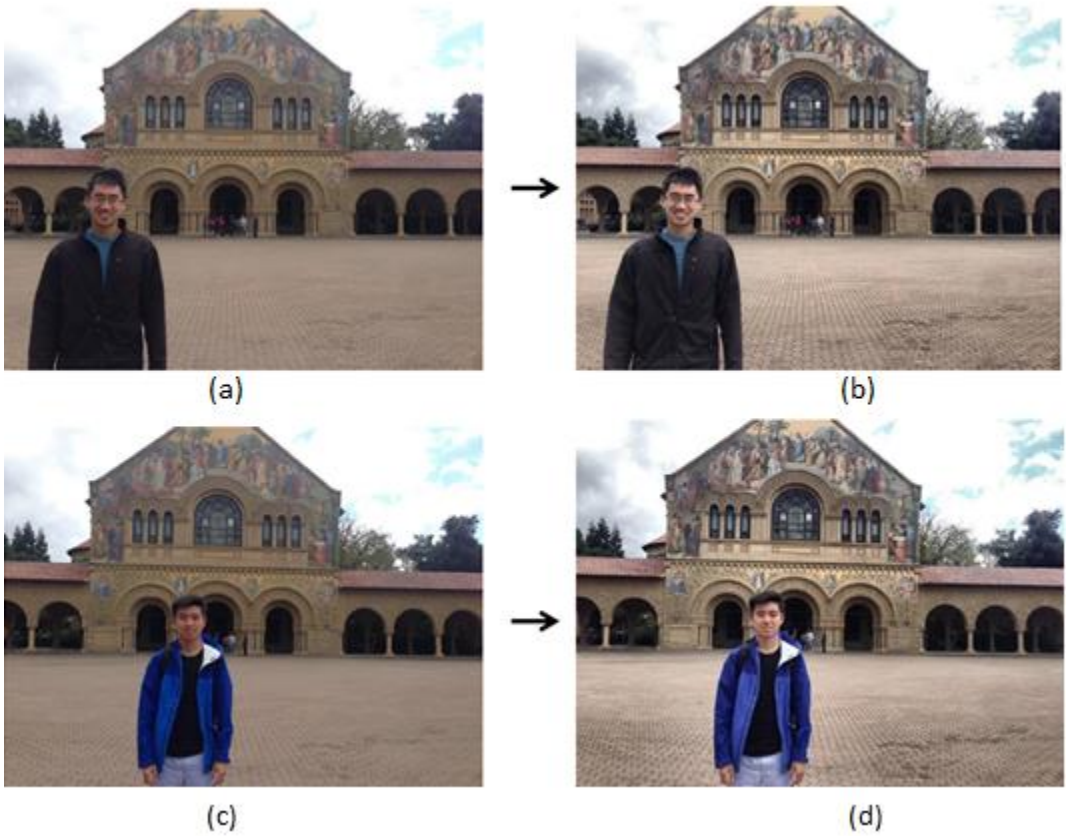

Figure 3: (a), (c) Original seed images. (b), (d) Seed images after color normalization.

As can be clearly seen in the clouds on the images above, contrast limiting has been applied to both seeds. The color in the seed images is a bit flattened by the limiting but we see the result as a feature, something of an “artsy filter” on the original picture.

It should be noted that CLAHE is not a true color normalization in that it does not take the color distribution of one image into account when blending in the second. However, we believe it is quite robust for the purpose of this application under the assumption that both seed images are taken in a similar time period. We also believe that CLAHE is much faster than implementing a membrane cloning approach.

Section 3.3.2: Image Merging

Bounding Box Determination

In order to accurately blend the two image subjects into the same image, it is important to obtain a precise bounding box for the bodies of the two subjects in the seed images. To do this, we start by facial recognition using pre-trained set of Haar cascaded classifiers for frontal faces. This classifier set offers robust facial detection and returns a bounding box for the face of the image subject. From this facial bounding box, we estimate the bounding box of the image subject by taking fixed ratios of the height and width of the face. In particular, we take the body to be three times the width of the facial bounding box and the height to extend from the a facial height above the top of the head to the bottom of the image.

This approximation is quite accurate in containing the full body of the image subjects and can be visualized in Figure 4.

Figure 4: Facial and body bounding boxes of image subjects.

There are, however, clear limitations to this approach. The most obvious is that it assumes that the image subjects are standing straight and with their arms to their sides. It also assumes that the image subjects are not holding something that extends outside their body bounding box that is important to the context of the picture. We take these as system limitations. This also assumes that the body is cut off by the image (we extend the body to down to the image height). We do this as a measure of redundancy to prevent really tall people from being prematurely truncated.

Keypoint Matching

ORB keypoints (we didn't want to use SIFT because it is patented) are used to determine a mapping between the two seed images. This is needed because we cannot assume that the perspective of each image will be the same since the image subject of one image is the photographer for the next. After keypoints are determined on each image independently, keypoint matching is done with Hamming distance measure and initially pruned based on a distance threshold of three times the minimum match distance. We also added an additional pruning step that removes keypoint matches from the bounding boxes of each image subject. This is done because it does not make sense for a keypoint on either image

subject from matching with a keypoint from the other seed image since that image subject was not in the other image. Figure 5 is an example of the keypoint matches generated from this process.

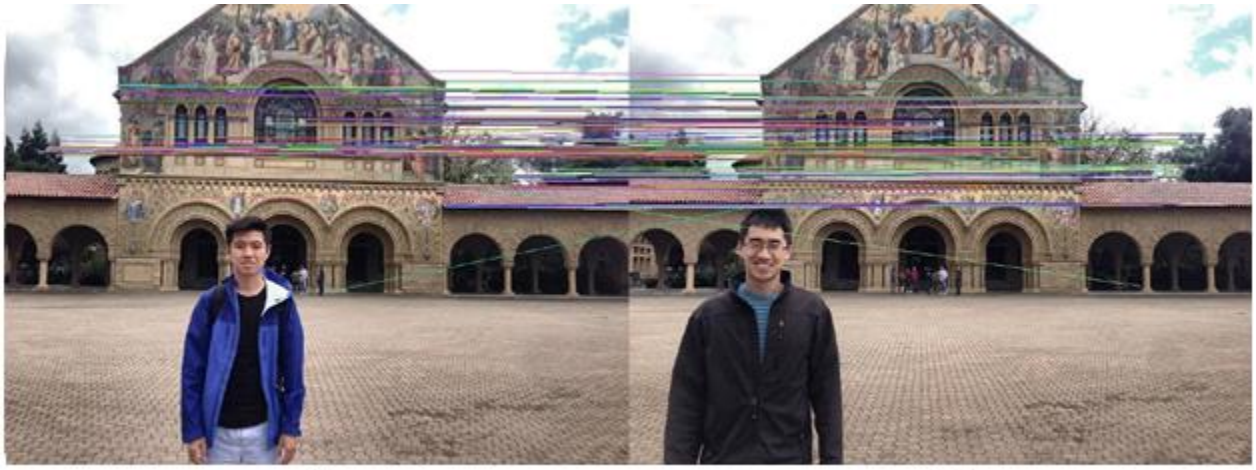

Figure 5: Keypoint matching after pruning on seed images.

Perspective Warping

A homography is determined using the pruned keypoint matches and RANSAC is used to single out the best transformation. The homography is then applied to foreground image: the image that will be blended in front of the other, background, seed image. The result of this step is shown in Figure 6.

Figure 6: Foreground image (b) is warped by homography.

The determination of the foreground and background images is important when image subjects overlap but does not need to be done if the body bounding boxes are distinct. We choose to use the size of the facial bounding box to distinguish the foreground image because we assume the subject with the larger face is most likely in front of the subject with the smaller one.

Grab Cut for Overlapping Image Subjects

If the image subjects have overlapping bounding boxes, then we need to segment out the image subject from the foreground image before pasting onto the background image. If this is not done, sections of the foreground image will occlude the image subject in the background image.

Image subject segmentation is done using an interactive tool called GrabCut. With a picture and a mask, GrabCut is able to distinguish foreground and background in the image and return a well segmented output. The mask is a matrix that associates each pixel in the image with a tag (definite background, probable background, definite foreground, probable foreground) which is usually supplied by the user manually. To fully automate the process, we automatically generate this mask with our knowledge of the facial and body bounding box. The face labeled foreground, the rest of the body bounding box is labeled probable foreground, and the rest of the image is labeled probably background. A sample result of GrabCut is shown in Figure 7.

Figure 7: (a) Mask determined by bounding boxes fed into GrabCut to produce (b).

GrabCut is a fairly slow process that can be iterated several times to increase the tightness of the segmentation. However, the time complexity of further iterations is linear and thus we limit the number of iterations to one to provide fast performance. There are also spill-over effects that are characteristic to GrabCut. Areas of above and to the left of the image subjects head in Figure 7b are included erroneously in the segmentation. It is also worth noting that GrabCut does not always perform well, especially in cases where the clothing or the hair of the image subject matches with the color of the background. These limitations to the current algorithm can best be solved with tighter body bounding boxes.

Blending

If the image subjects are well separated in the seed images, alpha blending is used in the region between the bounding boxes to form the merged image. Otherwise, if GrabCut segmentation was necessary, the pixels in the segmented foreground image are directly copied onto the background. To remove edge effects and artifacts from the merged image, we apply a erosion using a 3x3 ellipse-shape element on the edges of the segmented image (Figure 8).

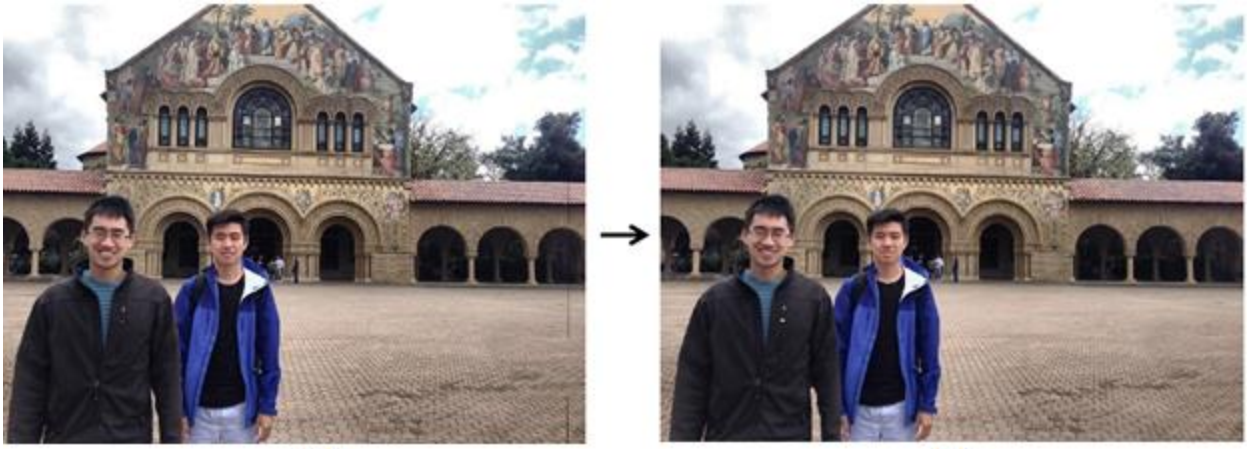

Figure 8: (a) Image before erosion blending. (b) Image after blending.

Section 3.3.3: Image Post-Processing

Cropping
The merged image is cropped based on the corners of the perspective warped image or the warped bounding box of the image body.

Output
The output of the merged image is saved on the device and displayed to the user.

Section 3.3.4: User Interface
The user interface (Figure 9) for this application reuses much of the code from the Panorama/HDR lab. The user taps the screen to take a seed image and after two seed images have been taken, the application proceeds to run the algorithm. There is also a preset available on the top navigation panel that allows static testing of two predefined seed images.

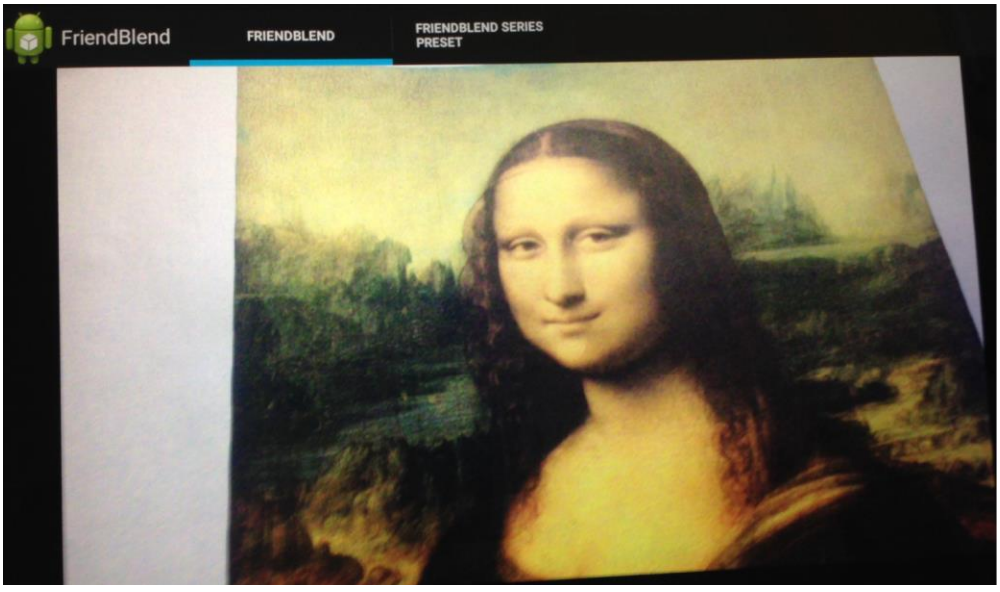

Figure 9: User interface of FriendBlend application.

Section 4: Experiments

Section 4.1: Sample Inputs and Outputs

Below, in Figure 10, are some sample outputs of the application.

Figure 10: Sample outputs of Friendblend at Stanford and in Hawaii.

As shown above, the merging application works fairly well, especially in cases where there are well-defined structures in the background that can be used for keypoint matching. When alpha blending is used (Figure 10 a, b, d), we do see some ghosting in the image area between the image subjects, which is due to an imperfect homography. In the case where GrabCut is used (Figure 10 c) there is a coloration mismatch around the shoe of the foreground subject. This is due to shadows on the ground, an artifact that the program in its current form does not address.

Section 4.2: Timing

If image subjects do not have intersecting bounding boxes, the computation time required for image merging is 3.5 seconds. If bodies do intersect, then the computation time is 7.2 seconds. Both of these times were recorded on the NVIDIA Shield platform. The extra time required for the intersection case is

mainly due to GrabCut as the difference in computation time between alpha blending and edge erosion is comparable. Increasing the iteration count of GrabCut increases the runtime by a bit more than three seconds per iteration.

Section 5: Conclusions

In conclusion, we have demonstrated FriendBlend, a simple and efficient native Android application that enables image merging of two pictures. The application is able to robustly determine a tight bounding box in each seed image of the image subject and based on the bounding box, merge images together to make it seem like an original photograph. The process is robust and can tolerate seed images in which image subjects are overlapping in position. A mobile user interface was implemented and tested on both an NVIDIA Shield tablet and HTC One smartphone with positive outcomes. We hope that FriendBlend provides photographers with more independence and reduces the prevalence of pesky "selfies" from filling the feeds of social networks.

Section 6: Work Division

The division of work was as follows:

Jeff Han: Initial project idea. Algorithm design and implementation of bounding box determination, keypoint matching, perspective warping, GrabCut and alpha blending flows in Python. Front end design and Java to C++ interface code. Data collection and testing. Final presentation and paper for CS 231m.

Kevin Chen: Algorithm design for edge erosion blending and cropping in Python. C++ debugging. Data collection and testing. Final poster and paper for EE 368.

David Zeng: Algorithm design for color normalization in Python. Full Python to C++ translation. C++ debugging. Data collection and testing. Final poster and paper for EE 368.

Section 7: References.

1) Eric N. Mortensen , William A. Barrett, Intelligent scissors for image composition, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, p.191-198, September 1995
2) PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. TOG (SIGGRAPH '03) 22 (2003), 313–318.
3) FARBMAN, Z., HOFFER, G., LIPMAN, Y., COHEN-OR, D., AND LISCHINSKI, D. 2009. Coordinates for instant image cloning. ACM Transactions on Graphics 28, 3 (Aug.), 67.
4) WILSON, P., FERNANDEZ, J. 2006. Facial Feature Detection Using Haar Classifiers. JCSC 21, 4 (April), 127-133.
5) ROTHER, C., KOLMOGOROV, V., BLAKE, A. 2004. GrabCut – Interactive Foreground Extraction using Iterated Graph Cuts. ACM Transactions on Graphics.
6) S. M. Pizer, E. P. Amburn, J. D. Austin, et al.: Adaptive Histogram Equalization and Its Variations. Computer Vision, Graphics, and Image Processing 39 (1987) 355-368.
7) SCHWARTZ W., KEMBHAVI A., HARWOOD D., DAVID L.: Human Detection Using Partial Least Squares Analysis. Proc. IEEE Int'l Conf. Computer Vision, 2009.