Face and Photograph Augmentation Based on a Custom Theme

EE368 Project, Autmn 2015

Orly Liba
Electrical Engineering PhD candidate
Stanford University
orlyliba@stanford.edu

***Abstract*—This report describes a technique to automatically augment photographs by adding masks and hats (or ears) to people's faces and to additionally complement the scene by adding Instagram-like effects such as vignetting, gradient blurring and color filtering. Unlike commercially available applications, this approach can be used to make any photographed or animated face into a mask. The augmentation of the target photograph is fully automatic and takes 1-3 seconds, depending on the size of the mage. As next steps, this algorithm could be implemented on a mobile device in real time for video-chat applications.**

Keywords—Morphing; Photograph Augmentation;

I. INTRODUCTION AND MOTIVATION

Face and photograph augmentation are a form of expression, communication and entertainment. Applying a mask on one's face is surprising and fun and may also be considered artistic. This project presents a technique to make any photographed or animated face into a mask and an automated algorithm to apply masks on top of faces.

Augmenting photographs with masks is available commercially by companies such as Snapchat, Google and Baidu. However, they only provide a predefined selection of masks that one could use. This project presents a simple method to turn any image into a mask, thus providing the user additional freedom of expression. The creation of the mask from an image can also vary and provide an artistic aspect to this process. To complete the scene, one can also add additional Instagram-like filters to the resulting photograph, such as gradient blurring, vignetting and changing the color temperature.

Nine masks were created for this project, which were available for a live demonstration.

II. RELATED WORK

Face morphing was demonstrated more than 20 years ago, most notably in Michael Jackson's "Black Or White" music video (fig. 1a) [1]. Back then, morphing was assisted by a manual selection of feature points or lines in order to find correspondence between the merged images. More recently, companies such as Google and Baidu have created automated tools for automated photograph and video augmentation (fig. 1b-d). These are mainly used for live chats by continuously applying a mask on one's face. Automation was enabled in recent years owing to advanced machine learning algorithms which allow exact detection of dense facial landmarks. While these applications are quick and produce rather good results at placing the mask on the target face, they only provide a limited set of masks, which are usually animated, and do not merge naturally. This project shows a method to create a custom, natural-looking, augmented image. Other groups have worked on close projects, such as Face Substitution [2].

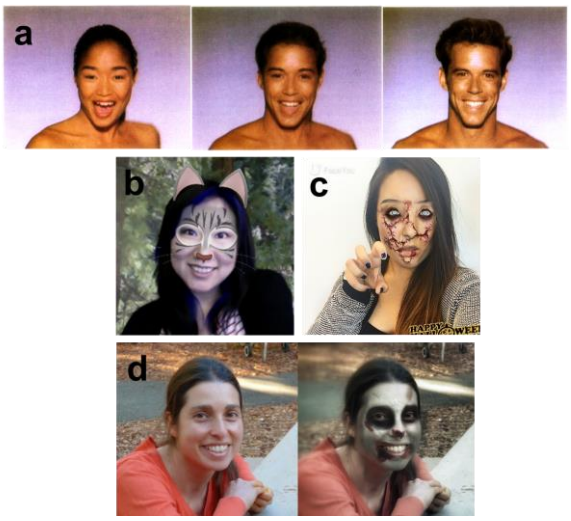

Figure 1 Related works. a) Images showing the morphing done in Michael Jackson's music video. Right and left are the originals and center is their mixture. b) Google hangout mask c) Baidu FaceYou morphing application d) Google's halloweenify (was available in 2014).

III. TECHNICAL APPROACH

A. Outline

The algorithm was implemented in Matlab with some of the functions running in C/C++ as mex files.

An outline of the approach is as follows (steps with * are optional):

1. For a chosen mask image (any photograph or animation):
   a. Create a blending map for the face. This gray level map describes the local weight of the mask in blending it with the target face. The blending map was created manually for this project, but could also be done automatically with segmentation. In order to achieve natural blending with the target the map can be smoothed with a Gaussian filter.
   b. * Create a blending map for a hat or ears. The process is similar to the previous step.
   c. Automatically or manually mark facial landmarks on the mask. This can be done automatically for people, using a landmark detector (see following steps) but has to be done manually in other cases.
   d. * Define parameters for changing the color temperature, vignetting and gradient blurring for the resulting image.
2. For the image to be augmented:
   a. Automatically detect all of the faces in the image using Haar Cascades with OpenCV.
3. For each face in the image:
   a. Detect the facial landmarks. In this project, landmarks were detected using Clandmark [3]–[5].
   b. Based on the correspondence of landmarks between the mask and the target face, create a transformation that warps the mask to fit onto the target face. The transformation was calculated by using locally weighted means [1].
   c. Warp the mask and the face-blending map according to the transformation.
   d. * Dynamically modify the face-blending map to include or exclude the regions of the eyes of the target face (in order to show or hide the eyes of the original face). This is done by adding or subtracting the convex hull of the eye-landmarks of the target face from the face-blending map.
   e. * Smooth the face-blending map, as defined by the mask parameters.
   f. Alpha-blend the warped mask and the target with the warped face-blending map used as the weight.
   g. * To automatically apply a hat or ears on the target face, scale, rotate and position the original mask (before warping) and the ear/hat-blending map according to the facial landmarks of the target face and the mask. Scale is determined by the width of the face. Position is determined by the landmark centered between the eyes. Rotation is determined by the angle of the landmarks of the nose.
   h. * Alpha-blend the mask and the target with the ear/hat-blending map used as the weight.
4. * Complete the scene by adding Instagram-like effects, such as:
   a. * Gradient blurring
   b. * Vignetting
   c. * Changing the color temperature

B. Facial landmark detection

Facial landmark detection is a key step to this algorithm and is crucial to obtaining good results. In this project, 68 landmarks were detected using Clandmark [3]–[5]. Clandmark is a multi-view facial landmark detector based on the Deformable Part Models (also called Pictorial Structures), which treats the problem of the simultaneous landmark detection and the viewing angle estimation within a structured output classification framework. A structured output SVM algorithm was used to simultaneously learn parameters of the shape model and the local detectors.

Clandmark is an open-source project available on github [3]. For this project, I had to compile the project and run it with the appropriate landmark model.

C. Calculating the mask transformation and warping

The mask-to-face transformation was calculated based on the facial landmarks, treating the landmarks of the mask as moving points and the landmarks of the face as static points. The transformation was calculated by using locally weighted means and implemented with Matlab's built-in function: `fitgeotrans (movingPoints, fixedPoints,'lwm',n)`. The local weighted mean transformation creates a mapping by inferring a second degree polynomial at each control point using neighboring control points. n is a parameter that chooses the number of closest points that are used to infer the polynomial transformation for each control point pair. The mapping at any location depends on a weighted average of these polynomials. The weighted average is calculated according to the distance of the pixel to the control point. A similar approach was used in [1]. Warping is performed with Matlab's function `imwarp`.

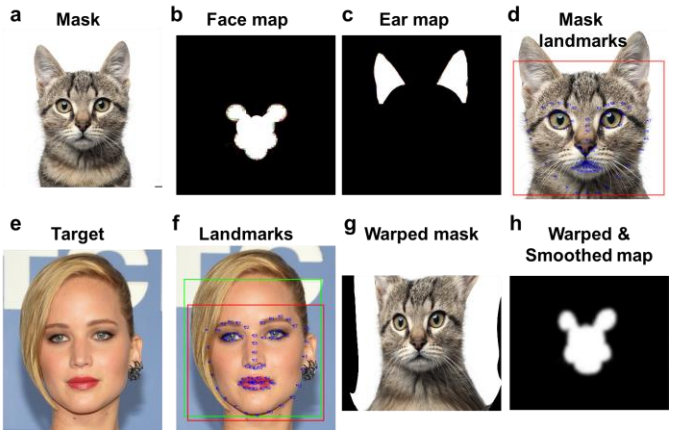

Figure 2 Outline of face augmentation. a) The chosen image for a mask. b) The face-blending mask. c) The ear-blending mask. d) 68 landmarks, in this case marked manually. e) The target image. f) Face and landmark automatic detection. g) The mask warped according to the landmarks. h) The face-blending map warped using a similar transformation and smoothed with a Gaussian filter.

D. *Gradient blur*

A gradient blur can be applied to emphasize the central region of the image. First, a normalized radius is calculated and the image is divided to eight blur levels. Next, the image is blurred eight times and then the blurred images are combined according to the local blur level. The standard deviation of the Gaussian blur kernel effectively increases with each blur level, as the distance from the center increases.

E. *Vingetting*

Vignetting, also known as "light fall-off" is common in optics and photography, which in simple terms means darkening of image corners when compared to the center. Vignetting is either caused by optics, or is purposefully added in post-processing in order to draw the viewer's eye away from the distractions in the corner, towards the center of the image. For this project, a vignetting effect was added by decreasing the pixel intensities by $\cos^4(\theta)$, similarly to natural lens vignetting. The angle, θ, is calculated as a linear function of the radius in the image: $\theta = r/d$, in which d is a parameter.

F. *Changing the color temperature*

The color temperature of a light source is the temperature of an ideal black-body radiator that radiates light of comparable hue to that of the light source. In In this project remapping of color values was done to simulate variations in ambient color temperature, for instance, to create more warmth or a spooky, cold atmosphere to the resulting image.

Figure 3 Temperature to RGB value from 1500 K to 5000 K [6]**.** High temperatures seem blue and low temperatures are orange-red.

Calculation of the color at given temperature was done based on [6]. The calculated color was averaged with the hue of the original image with a predefined weight (by alpha-blending), while maintaining the same image intensity. The resulting image seems warmer or cooler, as desired from the predefined temperature.

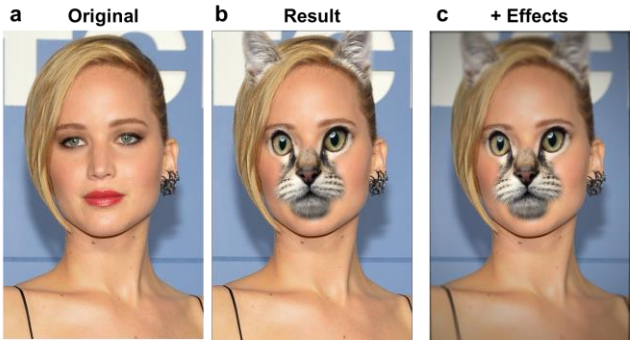

Figure 4 Result of the algorithm. a) The original image. b) The target image with mask and ears. c) The resulting image with vignetting and gradient blur (color temperature is unchanged).

IV. EXPERIMENTAL RESULTS

Nine themes were created for this projects. A description of the masks and the additional effects are listed below. Figure 5 shows examples of these masks, more examples can be found in the supplementary examples folder. The full process of applying the theme on a target image takes 1-3 seconds (on a laptop computer), depending on the size of the target image. The processing time increases with the number of faces in the image.

The masks are:

1. Santa-Claus: the face-blending map includes Santa's facial hair (eye brows and beard). An additional blending map was created for the hat. Landmarks were marked manually because the occlusion of the beard prevented the automated algorithm from detecting the landmarks. Blurring was applied to the blending maps for a gradual blend with the target image. Vignetting, blurring and a color temperature change to 3000K were added but are not shown (because without them the result looks more impressive).
2. Makeup: the face-blending map includes the eyes and lips of a woman wearing makeup. Blurring was applied to the blending maps for a gradual blend with the target image. Landmarks were detected automatically.
3. Obama: the face-blending map includes Barak Obama's face with a weight of 50%. In order to avoid seeing the eyes of the original face, dynamic masking was applied so that in the location of the original eyes the weight of the mask is 100%. Blurring was applied to the blending maps for a gradual blend with the target image. Landmarks were detected automatically.
4. Clown: the face-blending map includes the face of the clown. An additional blending map was created for the red hair. In order to see the eyes of the original face, dynamic masking was applied so that in the location of the original eyes the weight of the original image is 100%. Blurring was applied to the blending maps for a gradual blend with the target image. Landmarks were detected automatically and manually corrected for the eyebrows which were wrongfully detected due to white makeup.
5. Skull: the face-blending map includes the face of the skull. In order to see the eyes of the original face, dynamic masking was applied so that in the location of the original eyes the weight of the original image is 100%. Blurring was applied to the blending maps for a gradual blend with the target image. Landmarks were marked manually. To make the result seem spooky, the saturation of the image was decreased to a half of the original value and vignetting and blurring were added.
6. Anonymous: the face-blending map includes the face of the mask. In order to see the eyes of the original face, dynamic masking was applied so that in the location of the original eyes the weight of the original image is 100%. Only slight blurring was applied to the blending maps. Surprisingly, the landmark detection algorithm was able to detect facial landmarks automatically. Vignetting

and blurring were added to the image (but not shown in figure 5).

7. Cyborg: the face-blending map includes parts of the face of the terminator robot. Slight blurring was applied to the blending maps. Landmarks were marked manually. To make the result seem cool and metallic, the color temperature is set to 10,000K and vignetting and blurring were added.
8. Night's king: the face-blending map includes the full face of the night's king (from Game of thrones) with a weight of 100%. Facial landmarks were detected automatically. To make the resulting image appear as if it was captured in a cold foggy place, the color temperature is set to 10,000K and vignetting and blurring were added. In addition to that, the color balancing was transferred from the original image, meaning that the average of each channel (r, g and b) is equal to that of the original image.
9. Cat: the face-blending map includes the eyes, nose and mouth of a cat. An additional blending map was created for the ears. Blurring was applied to the blending maps for a gradual blend with the target image. Landmarks were detected manually. Vignetting and gradient blurring were added.

Figure 5 Examples of the nine masks.

V. Evaluation of System Parameters

One of the parameters of the algorithm is the number of closest points that are used to infer the polynomial transformation for each control point pair (the parameter n in `fitgeotrans`). A small value of n results in an unstable and incorrect transformation. Using all of the available points (68) results in a correct result. Processing time increase is negligible.

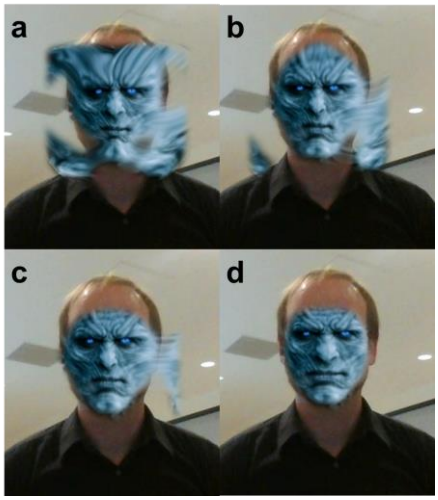

Figure 6 The effect of the number of points used for calculating the morphing transformation. a) n=20 b) n=30 c) n=40 d) n=68 (the image is after applying the mask and before adding additional effects).

Additional parameters include the size of the blur kernel, the chosen color temperature, the speed of signal decrease in the vignetting and other parameters which directly influence the appearance of the image. The effect of these parameters is intuitive therefore their evaluation is out of scope for this report.

VI. Comparison to Alternative Approaches

The two key components to the success of this algorithm are facial landmark detection and warping.

Warping needs to be continuous and natural. Using the local weighted means approach for the calculation of the transformation is a fairly robust and smooth method. An alternative method to calculate a per-pixel transformation would be a piecewise-linear approach. This transformation maps control points by breaking up the plane into local piecewise-linear regions in which a different affine transformation maps control points in each local region. This approach yields transformations which are not continuous or smooth and even broken (have holes). Therefore, the local weighted means approach was chosen for this project.

Detection of facial landmarks has been a hot topic in the fields of computer vision and machine learning in the recent years. Facial landmarks can be used for example to identify people or estimate a person's mood, age, gender, etc. The first requirement of a landmark detector for this project is that it would detect dense landmarks, and not merely the edges of facial features. Second, since the project is implemented in Matlab, it is preferred that the detector have a Matlab

interface. I found three available online tools that meet these requirements:

1. Clandmark [3]–[5], that was described earlier and that was used for most of this project.
2. Face ++ [7]–[10], a commercial vision platform. Face++ provides a Matlab interface to upload images to the company's servers and returns the location of the face and facial landmarks. Their landmarks are very robust and more resistant to occlusions. However, I did not like the idea of uploading photos to unfamiliar servers therefore, I did not choose to use their product. A comparison between face++ and Clandmark is presented in figure 7.
3. Cambridge face tracker [11] had the potential to be useful for this project, however, I did not manage to compile it.

Other mentionable landmark detectors that were not tested because they did not have a Matlab interface are FaceTracker [12] and Active shape models [13].

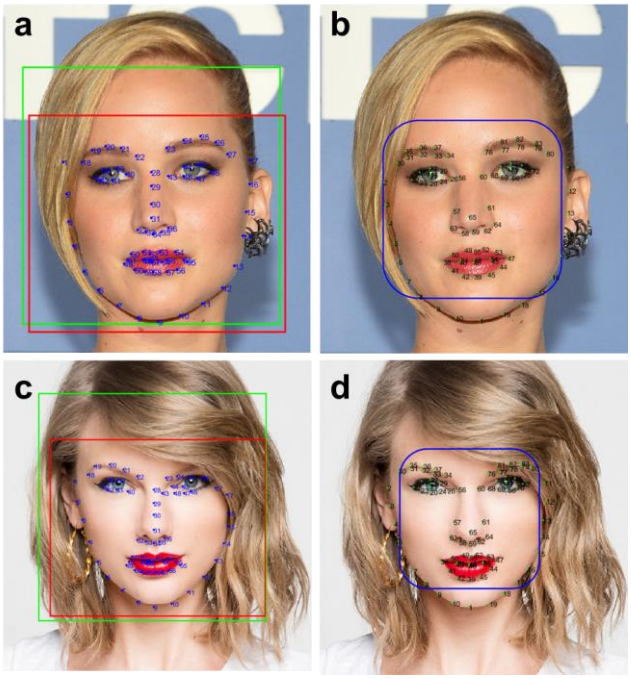

Figure 7 Comparison between Clandmark and face++. a, c) the 68 landmarks detected by Clandmark b, d) the 83 landmarks detected by face++. In c we observe that the landmarks are not detected correctly by Clandmark due to hair occluding the right eyebrow. In d we observe that face ++ is able to detect the landmarks correctly.

VII. Discussion, Limitations

Face and photograph augmentation were successfully demonstrated on a variety of images and masks. Owing to the exact identification of the landmarks, masks are nearly always placed perfectly at the right location and blend naturally with the target. The algorithm runs on Matlab and takes 1-3 seconds to complete.

Several limitations have been identified:

1. When the head is rotated at a large angle, OpenCV is not able to detect the face correctly, and therefore, facial landmarks are not detected or are incorrect. This may be resolved by using a more advanced face detector.
2. When the facial landmarks are detected incorrectly on the target image the algorithm yields suboptimal results. This happens when the main features of the face are obscured. This may be resolved by better training or a modified model.
3. Ear or hat placement supports rotation of the head but not yaw (rotation in phi). This is because the transformation of the ears/hat is rigid and only has rotation and scaling, not perspective.

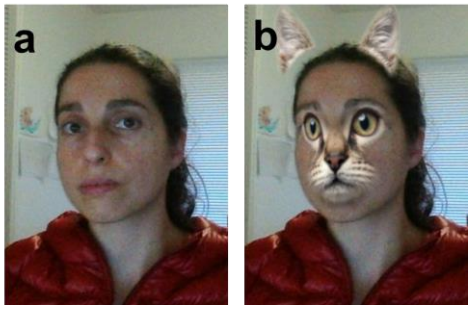

Figure 8 The effect of yaw rotation on ear placement. a) Original b) Augmented image, one ear is in place but the other is floating above the head.

4. There are no landmarks on the forehead and at the top of the head. This makes masks unstable above the eyebrows. Training the landmark detector to detect the forehead may resolve this, but perhaps a deferent model may be needed. From the examples in figure 1d, it seems that Google has managed to solve this issue.

VIII. Future Work

As next steps:

1. This technique could be implemented on a mobile device and used as a fun application for augmenting photographs.
2. Increasing the speed of the processing would allow real-time augmentation for video and chat. Landmark tracking instead of continuous detection would assist in reaching this goal.
3. While facial landmark detection is usually very exact, additional improvement should be done to make it more robust and detect landmarks on rotated and partially obscured faces, such as by a beard, glasses or bangs.
4. Find or design a facial landmark detector that is able to identify landmarks at the edge of the forehead (hairline) and the contour of the top of the head. These landmarks would enable "stretching" a mask fully on the entire face and allow for a more precise placement of ears or a hat.

Acknowledgment

I would like to acknowledge Michal Uřičář for making Clandmark an open source project. In addition, I would like to thank Professor Gordon Wetzstein and the EE368 course staff.

References

[1] T. Beier and S. Neely, "Feature-based image metamorphosis," *ACM SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 35–42, Jul. 1992.

[2] A. Castro and K. McDonald, "FaceSubstitution," 2011. [Online].

Available: https://github.com/arturoc/FaceSubstitution.

[3] M. Uřičář, "clandmark," 2015. [Online]. Available: https://github.com/uricamic/clandmark.

[4] M. Uřičář, V. Franc, D. Thomas, A. Sugimoto, and V. Hlaváč, "Real-time Multi-view Facial Landmark Detector Learned by the Structured Output SVM," in *BWILD'15: Biometrics in the Wild*, 2015.

[5] M. Uřičář, V. Franc, and V. Hlaváč, "Facial Landmark Tracking by Tree-based Deformable Part Model Based Detector," in *Proceedings of IEEE International Conference on Computer Vision, 300 Videos in the Wild (300-VW): Facial Landmark Tracking in-the-Wild Challenge & Workshop (ICCVW'15). Santiago, Chile*, 2015.

[6] T. Helland, "Temperature to RGB." [Online]. Available: http://www.tannerhelland.com/4435/convert-temperature-rgb-algorithm-code/.

[7] Megvii Inc., "face++ Research Toolkit," 2013. [Online]. Available: www.faceplusplus.com.

[8] H. Fan, M. Yang, Z. Cao, Y. Jiang, and Q. Yin, "Learning Compact Face Representation," in *Proceedings of the ACM International Conference on Multimedia - MM '14*, 2014, pp. 933–936.

[9] H. Fan, Z. Cao, Y. Jiang, Q. Yin, and C. Doudou, "Learning Deep Face Representation," Mar. 2014.

[10] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin, "Extensive Facial Landmark Localization with Coarse-to-Fine Convolutional Network Cascade," in *2013 IEEE International Conference on Computer Vision Workshops*, 2013, pp. 386–391.

[11] T. Baltrusaitis, "Cambridge Face Tracker," 2014. [Online]. Available: https://github.com/TadasBaltrusaitis/CLM-framework.

[12] J. Saragih and K. McDonald, "FaceTracker." [Online]. Available: http://facetracker.net/.

[13] S. Milborrow and F. Nicolls, "Active Shape Models with SIFT Descriptors and MARS," in *International Conference on Computer Vision Theory and Applications*, 2014, pp. 380–387.