Optical Illusion Puppet Generator

Vivian Yang
Department of Electrical Engineering, Stanford University
350 Serra Mall, Stanford, CA
viviany@stanford.edu

Abstract

We have designed a system that can automatically generate a real optical illusion puppet from any input individual frontal-view face photo. The basic function of the system is to capture the human face from the given photo and correctly transform it onto a fixed template. The final product is a 2-D layout that can be printed out for further assembling.

1. Introduction

Art is inherently visual. With the advances of computational imaging, one is able to explore the possibilities of allowing the public to interact with art in ways that would be otherwise impossible. For example, an engineered t-rex paper craft, as shown in Figure 1, can produce an optical illusion which makes the observer feels that the t-rex is always looking at him or her, as shown in Figure 2. Inspired by this optical illusion t-rex puppet, our project aims to design a system that can crop human faces from photos and map it on a template that can create an optical illusion effect which lets the image seem to turn its head and follow you as you move.

To achieve this goal, we proposed to design a system that can let the user input an individual frontal-view face photo, then auto customize a human-face optical illusion product, which seems to turn its head and follow you as you move. The pipeline of our method is: *1)* capture human faces from photos; *2)* apply affine transformation to modify the size and the position of the face; *3)* correct the color of the human face to match the template; *4)* merge the face and the skull template together.

2. Related work

2.1. Facial landmark detection

Facial landmark detection under generic settings is always an extremely challenging problem that gets compounded in presence of external factors such as illumination changes, pose variation and occlusion. Over the past 20 years, great amount of techniques have been proposed for computer landmark detection of human faces, such as CLM [1] and Dlib facial landmark detection [5]. OpenFace [2] can even do head pose tracking, eye gaze and facial Action Unit estimation. Also, these are all open source framework which can be implement in this project.

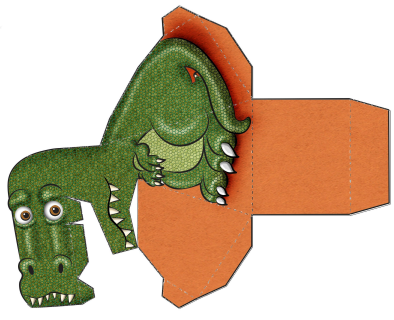


Figure 1: Printout of the t-rex template.

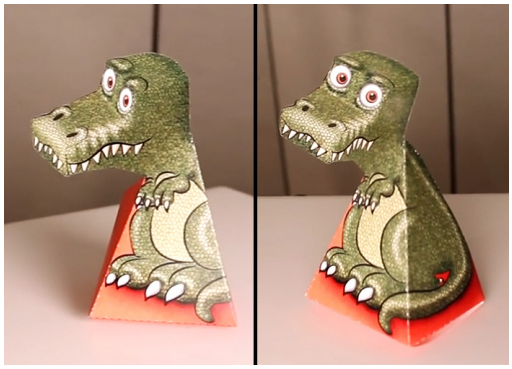


Figure 2: Example of an optical illusion t-rex.

2.2. Perspective transformation

There are a lot of transformation method being developed over years. For instance, the 3-point affine transformation, 4-point perspective transformation function in OpenCV [3] and the eight-point algorithm. There is even a more advanced image-based, facial reenactment system that replaces the face of an actor in an existing target video with the face of a user from a source video, while preserving the original target performance in [4].

3. Implementation

3.1. Environment Setting

To fulfill our design, we mainly need to use two kinds of library, which are described as follows.

3.1.1 OpenCV

OpenCV [3] is an open source computer vision and machine learning software library which has been widely utilized to provide a common infrastructure for computer vision applications.

3.1.2 Dlib

Dlib [5] is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.

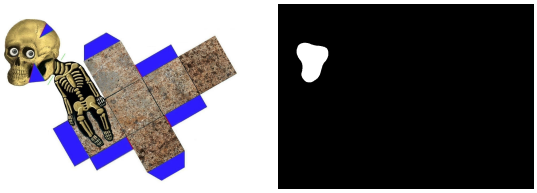
3.2. Hollow-Face illusion

The Hollow-Face illusion is an optical illusion in which the perception of a concave mask of a face appears as a normal convex face. The trick inside this illusion is that a hollow face can appear to move its eyes faster than the viewer: looking forward when the viewer is directly ahead, but looking at an extreme angle when the viewer is only at a moderate angle. According to this characteristic, it will create an illusion that the puppet's head seems to follow the viewer's eyes everywhere (even up or down), when lighting, perspective and/or stereoscopic cues are not strong enough to tell its face is actually hollow.

There are a lot of optical illusion puppet templates online. The reason why we choose the skull template for our project is because there are more similarities between skull and human face. Therefore, the human face do not need to distort a lot to fit into the template.

3.3. Landmark detection

The system must be able to find the location of different facial features (e.g. centers of the eyes, nose, mouth), so that it can accurately crop the human face from the photo and put it on the template in correct size and position.



(a) (b)

Figure 3: (a) The chosen template. (b) The mask of the chosen template.

Therefore, we choose to use Dlib, which will detect 68 landmarks of the human face, as shown in Figure 5, because the algorithm it have chosen to implement is very fast and accurate. The detection function in dlib is as follow:

// We need a face detector. We will use this to get bounding boxes for each face in an image.
frontal_face_detector detector = get_frontal_face_detector()
// And we also need a shape_predictor. This is the tool that will predict face landmark positions given an image and face bounding box.
shape_predictor sp
deserialize(argv[1]) >> sp
det = detector(img)
full_object_detection shape = sp(img, det)

After applying the function, the detection result will be like Figure 5.

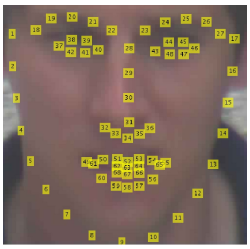


Figure 4: The positions of 68 landmarks.

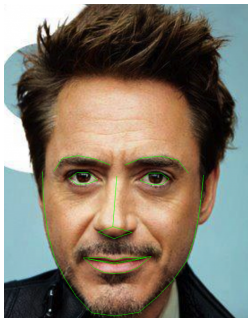


Figure 5: Actual facial landmark detection.

3.4. Affine transformation

Basically there are two different transform functions in OpenCv [3]:

getAffineTransform(src_points, dst_points),

which calculates an affine transform from three pairs of the corresponding points, and

getPerspectiveTransform (src_points, dst_points),

which calculates a perspective transform from four pairs of the corresponding points. Since there are three main features on both human and skull's face, which is right eye, left eye and mouth, we choose to use the first transform function *getAffineTransform*.

A typical way to represent an Affine Transform is by using a 2×3 matrix:

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}_{2\times 2} \quad B = \begin{bmatrix} b_{00} \\ b_{10} \end{bmatrix}_{2\times 1}$$

$$M = \begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & b_{00} \\ a_{10} & a_{11} & b_{10} \end{bmatrix}_{2\times 3}$$

Considering that we want to transform a 2D vector $X = \begin{bmatrix} x \\ y \end{bmatrix}$ by using A and B, we can do it equivalently with:

$$T = A \cdot \begin{bmatrix} x \\ y \end{bmatrix} + B$$

or

$$T = M \cdot [x, y, 1]^T$$

$$T = \begin{bmatrix} a_{00}x + a_{01}y + b_{00} \\ a_{10}x + a_{11}y + b_{10} \end{bmatrix}$$

Since we know both X and T's three points and we also know that they are related, our job is to find M. This is what the *getAffineTransform* do. Then we can apply this found relation to the whole pixels in the image, by using the following code:

warpAffine(mat_X, mat_T, mat_M, mat_T.size()).

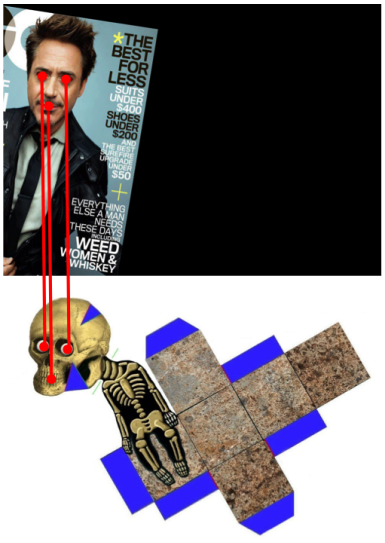


Figure 6: 3-point affine transformation.

After using the function, the photo will be resized and moved to let the face correctly fit perfectly into a specific frame.

3.5. Color correction

Differences in skin-tone and lighting between the two images will cause a discontinuity around the edges of the overlaid region. Therefore, we need to correct the color of the warped human face image, using the following function:

correct_color(template_img, skull_img){
GaussianBlur(face_img, kernel_size, standard_deviation)
GaussianBlur(skull_img, kernel_size, standard_deviation)
return (face_img . skull_img_blur ./ face_img_blur)*
}

This function is to change the coloring of human face image to match that of the template image. The idea is that of a RGB scaling color-correction, but instead of a constant scale factor across all of the image, each pixel has its own localized scale factor.

Figure 7: Color correcting the facial image.

3.6. Seamless Cloning

In this part, we are going to blur the mask and use this blurred mask to select which parts of the face image and which parts of the template image should be shown in the final image. In the blurred mask, regions with value 1 (colored as white) correspond with areas where the face image should show, and regions with colour 0 (colored as black) correspond with areas where the template image should show. Value in between 0 and 1 correspond with a mixture of the face image and the template image. The result will be like the image being shown in Figure 8

4. Experimental results

To evaluate the performance of the proposed optical illusion puppet generator, we printed out the image we generated, as shown in Figure 9, and assembled it. From the

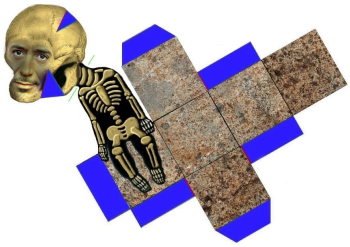


Figure 8: Merge the morph facial image onto the template.

Figure 10 we can see that, comparing to the t-rex and ironman puppet, although our puppet does not perform as well as the t-rex puppet, it outperforms the ironman puppet, because our puppet has more details.

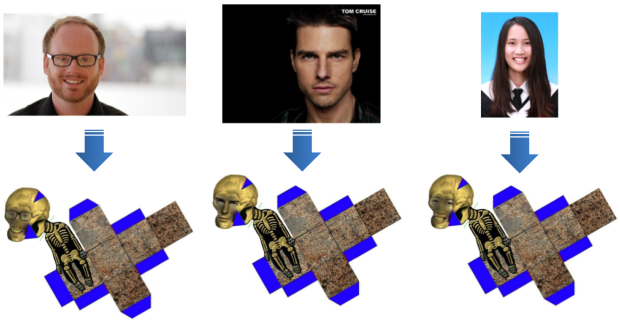


Figure 9: Output result.

(a) (b)

Figure 10: (a) Look from left-hand side. (b) Look from right-hand side.

5. Discussion

From the experiment above we can see that this system only works for single person, frontal-view face photo. Also, in the affine transformation we only use three point to transform the face image, which is less accurate than the eight-point algorithm. Therefore, to better the system, we can try to implement the eight-point algorithm in the affine transformation part. Furthermore, we can try to let the system can generate puppets from a photo with a group of people at once.

Figure 11: The back of the puppet.

6. Conclusion

In this project, we have proposed a system that can generate a real optical illusion product from a photo. To fulfill this goal, we need to detect the landmarks of the input photo first. Then, we have to geometrically transform it into the right size, right position. Next, we modify the face to let its skin color be more similar to the color of the skull. Last, we paste the color corrected face on the skull template that tricks our brains to interpret faces as convex. From the result we can see that the final product is quite delicate. Therefore, the performance of our generator is still competitive.

References

[1] T. Baltrušaitis, P. Robinson, and L.-P. Morency. 3d constrained local model for rigid and non-rigid facial tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Openface: an open source facial behavior analysis toolkit. *IEEE Winter Conference on Applications of Computer Vision*, 2016.

[3] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] P. Garrido, L. Valgaerts, O. Rehmsen, T. Thormäehlen, P. Perez, and C. Theobalt. Automatic face reenactment. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4217–4224, 2014.

[5] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.