Image Registration Based on Control Points and GUI Implementation

Wangzhao Tangxue Jinwenli
School of Data Science, Fudan University
December 25, 2018

Abstract

In this report, we design local affine transformation, moving least squares transformation, image morphing algorithm, and interactive interface. This report realizes face deformation and optimization based oncontrol points, such as the transformation from human face to great ape face and fusion of other people's face.

Firstly, we design three different algorithms to achieve the goal of face deformation. These three algorithms continually optimize the visual effects of deformation. The first kind of local affine transformation can realize the transformation of human face and gorilla face; the second kind of moving least squares transformation can achieve smoother face transformation effect; the third image morphing algorithm can combine the characteristics of both sides and optimize the visual effect.

Secondly, we designed an interactive interface based on PySide. The program can generate fused photo results based on selected photos, making it easy for users to change photos. In addition, the user can choose the algorithm and parameter values autonomously.

1. Introduction

The affine transformation of an image mainly includes the following operations: flip, rotate, pan, zoom, and cut. The Affine Transform class describes the function of a two-dimensional affine transformation, which is a linear transformation from two-dimensional coordinates to two-dimensional coordinates, maintaining the "straightness" of the two-dimensional figure, that is, the transformation after the line is retained. Straight line, arc or arc; "parallelism", that is, the relative positional relationship between two-dimensional pixels is constant, parallel lines are parallel lines, and the intersection angle of the intersecting lines is constant. Our program implementation converts the face into a face similar to the great ape. The input data is two selected images.

2. Algorithm

2.1 Local Affine Transformation

1 Determine the corresponding area (point) of the local affine transformation.

The local affine transformation region has n points, which is the number of feature points. The manual positioning is used to determine the set of feature points of the orangutan face $U=\{U_1,U_2,\cdots,U_n\}$, and the set of facial feature points is located by using the API of company megvii Face++ $V=\{V_1,V_2,\cdots,V_n\}$.

2 Find the affine transformation function of the corresponding region (point).

For each corresponding regions (U_i,V_i), the affine transformation matrix can be found. The area of the local affine transformation is n points, so it is only necessary to find the translation vector a_i of each corresponding points (U_i,V_i) .Then, the affine transformation function of the i pair of corresponding point is $V_i=G(U_i)=U_i+a_i$.

3 Local affine transformation algorithm.

For each pixel of the great apes image X , we can calculate the coordinates $T(X)$ of the pixel corresponding to the face through the local affine transformation algorithm:

$$T(X)=\begin{cases} G_i(X), X\in U_i\ ,i=\ 1\ ,.2.,\ n \\ \sum_{i=1}^{n} w_i(X)G_i(X), X\notin \bigcup_{i=1}^{n} U_i \end{cases} \tag{2.1}$$

Where $w_i(x)$ is a normalized weighting factor related to the distance $d_i(x)$ between point x and region U_i.That is $G_i(X)=X+a_i$, $w_i(x)=\dfrac{1/d_i(x)^e}{\sum_{j=1}^{n}1/d_j(x)^e}$.

Where e controls the locality of affine transformations, we take e as 1.5, 2, 3 respectively.

4 Interpolation method for RGB values.

We use bilinear interpolation to find the RGB values of the face coordinates $T(X)$. The assumed coordinates of $T(X)$ are $(a+m,b+n)$, where a,b represent the integer part m,n represent the fractional part. Then, RGB value of $T(X)$ can be determined by the RGB values of (a,b)、$(a+1,b)$、$(a,b+1)$、$(a+1,b+1)$. Using $f(a,b)$ represent the

RGB value of (a,b), then we can get the bilinear interpolation algorithm is:

$$f(a+m,b+n)=(1-m)(1-n)\cdot f(a,b)+(1-m)n\cdot f(a,b+1)+m(1-n)\cdot f(a+1,b)+mn\cdot f(a+1,b+1)$$

Finally, we replace RGB values of all the pixels of the Teacher Zhuang image to get deformation image.

2.2 Moving Least Squares Deformation

Let p be a set of control points and q be the deformed positions of the control points .For instance, p are the ape's deformation points and q are the man's deformation points .Given a point v in the image, we solve for an optimal affine transformation $l_v(x)$ to minimize

$$\sum_i w_i \left| l_v(p_i) - q_i \right|^2 \tag{2.2}$$

Where p_i and q_i are row vectors and the weights w_i have the form

$$w_i = \frac{1}{\left| p_i - v \right|^{2\alpha}} \tag{2.3}$$

Because the weights w_i in (2.2) are dependent on the point of v, we call this a Moving Least Squares minimization. For each point v, we can obtain a different transformation $l_v(x)$ .Finally, we define deformation function f to be $f(v)=l_v(v)$.

Since $l_v(x)$ is an affine transformation, it consists of two parts: a linear transformation matrix M and a translation T .

$$l_v(x) = xM + T \tag{2.4}$$

Minimizing the equation (2.2), we can get that

$$T = q_* - p_* M \tag{2.5}$$

Where $p_* = \frac{\sum_i w_i p_i}{\sum_i w_i}$ and $q_* = \frac{\sum_i w_i q_i}{\sum_i w_i}$.

Rewrite $l_v(x)$ in terms of the linear matrix M .

$$l_v(x) = (x - p_*)M + q_* \tag{2.6}$$

The least squares problem of equation (2.1) can be rewritten as

$$\sum_i w_i \left| \hat{p}_i M - \hat{q}_i \right|^2 \tag{2.7}$$

Where $\hat{p}_i = p_i - p_*$ and $\hat{q}_i = q_i - q_*$.

Finding an affine deformation that minimizes equation (2.7) is using the classic normal

equation solution.

$$M = \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right)^{-1} \sum_j w_j \hat{p}_j^T \hat{p}_j$$

With the closed-form solution for M . The deformation function $f(v)$ can be showed in the form

$$f(v) = \sum_j A_j \hat{q}_j + q_*$$

Where $A_j = (v - p_*) \left(\sum_i \hat{p}_i^{\,T} w_i \hat{p}_i \right)^{-1} w_j \hat{p}_j^{\,T}$.

There are some problems with affine transformation. The grid lattice in the original image is neatly arranged but no longer neatly arranged after transformed into the target image. Since it is a floating point operation, there are some points that will be transformed to the same point of the target image and there are some points in the target map that have not changed from the original image. It causes a white cutout in the transformed image .A simple way to solve this problem is to inversely transform the original image, which is equivalent to calculating the corresponding point in the original image on the grid lattice of the target image.(i.e. calculate v from $f(v)$).

$$v = \left(f(v) - q_* \right) \left(\sum_i \hat{p}_j^{\,T} w_j \hat{q}_j \right)^{-1} \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right) + p^*$$

2.3 Image Morphing

The third method is the Image Morphing. Image morphing was first used extensively in the movie and now is widely applies in the image transformation. In our project, we try to realize the deformation between two images, so morphing may be a proper method to achieve our goals.

The idea behind morphing is rather simple. Given two images I and J, we want to create an in-between image M by blending the original two images. The blending process is controlled by a parameter α which is between 0 and 1. You can get the image M using the following equation:

$$M(x, y) = (1 - \alpha) I(x, y) + \alpha J(x, y)$$

In the equation, (x, y) symbolizes every pixel in the image. However, if we use the equation directly, the results may be terrible because of double vision like Figure5 shows. So this method also need face feature detection and warping first to revise the

double vision problem.

Next, we will give more details about how to apply this method step by step:

1. Obtain the feature points in original image I and image J.

   In our project we use the face++ API interface to find the feature points and the points are one-to-one corresponding. The points are shown in Figure 6.

2. Delaunay Triangulation.

   From the previous step we have two sets of point——one set per image. We can calculate the average of corresponding points in the two sets and get a single set. On this set of average points we perform Delaunay Triangulation. The triangles in the two images can capture approximately similar regions and they are also corresponding. The triangulations are shown in Figure 7.

3. Warping images.

   (1) Find the location of feature points in morphed image: We can find the location of all the feature points using the following equation:

   $$x_m = (1-\alpha)x_i + \alpha x_j$$

   Where $(x_i, y_i), (x_j, y_j)$ are the feature points in the step 1.

   (2) Calculate affine transforms: Until now, we have three sets of points. Pick each triangle in image I from step2 and calculate the affine transform from image I to image M. Then, we repeat the process between image J and image M. This process can be done using get Affine Transform function.

   (3) Warp triangles: To correctly use the function warp Affine, we need to calculate a bounding box for the triangle first by using the function fill Conves Poly. After that, we need to use the affine transformation, calculated in the previous step, to transform all pixels inside the triangles in image J to the morphed image M and get a warped version of image I. Similarly, we can repeat the process to image J to get another warped version of image J. The warping process can be realized by the function warp Affine.

4. Alpha blending.

   After we get two warped version, the two version can be can be blended using the following equation:

   $$M(x_m, y_m) = (1-\alpha)I(x_i, y_i) + \alpha J(x_j, y_j)$$

   Under all the process above, we can get our final morphed image like Figure 8.

3. Experience

3.1 Results of Local Affine Transformation

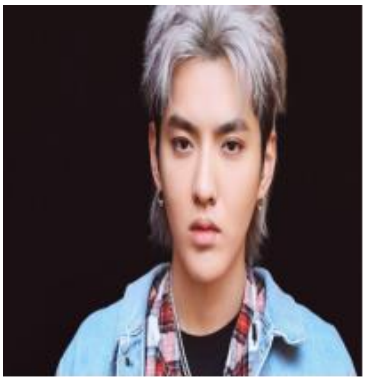
1. Origin picture: Kris Wu; Control points: 16

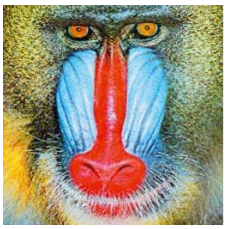


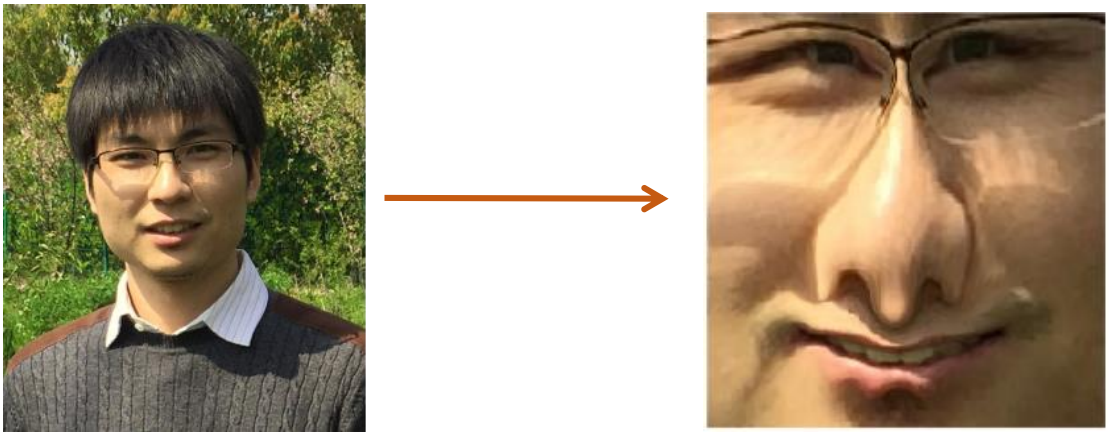
(a) e=1.5 (b) e=2 (c) e=3

From the results, we can find that when the parameter is 1.5, the part of the eye is obviously deformed and the lower lip is slightly curved. When the parameter is 3, the eyelids in the right eye will be slightly deformed, and the face shape will be similar to that of a gorilla. When the parameter is 2, the result is the best, and the face shape is thinner than the parameter is 3.

2. Replace the original picture



From the results, we can find that the difference of the initial picture on the result is mainly the nose part. Kris Wu's result is smoother than others. However, they all better reflect the characteristics of great ape.

3. Different control points

1)12 points (Eye 2*2 points; nose 3 points; mouth 5 points.

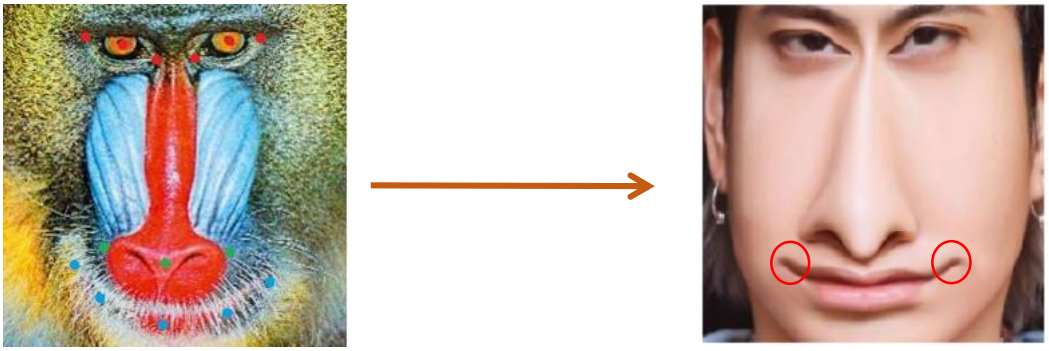
From the result, we can find that the shape of the eye is severely deformed, so we add two points in the center of the eye next.

2)14 points (Eye 3*2 points; nose 3 points; mouth 5 points)

Three feature points for each of the left and right eyes: two eye corners and the center of the eyeball (3x2=6)

Three characteristic points of the nose: left and right nosewing and tip of nose.

Five feature points of the mouth: divide the lower contour of the lips from the left corner to the right corner and take 5 points.



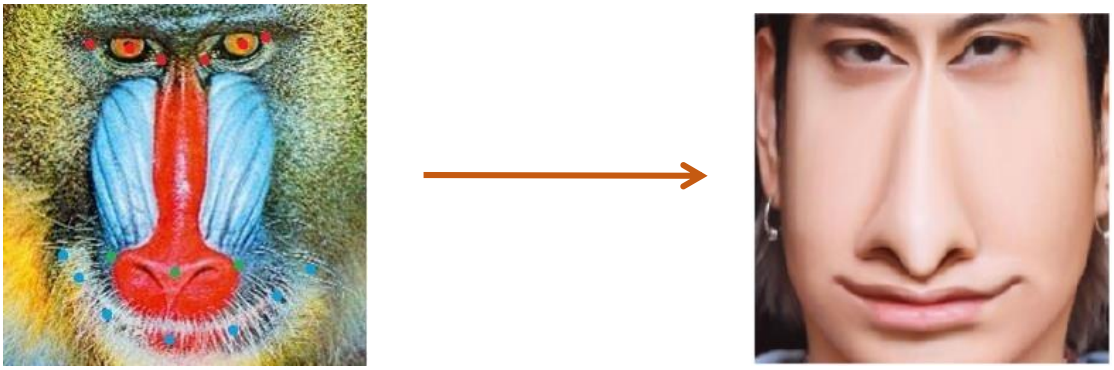
From the result, we can find that there are some coincidences on both sides of the mouth, which are circled in red circles. So next, we add two more points on the mouth.

3)16 points (Eye 3*2 points; nose 3 points; mouth 7 points)

Three feature points for each of the left and right eyes: two eye corners and the center of the eyeball (3x2=6)

Three characteristic points of the nose: left and right nosewing and tip of nose.

Seven feature points of the mouth: divide the lower contour of the lips from the left corner to the right corner and take 7 points.



From the result, we can find that this picture has better effect.

3.2 Results of Moving Least Squares Deformation

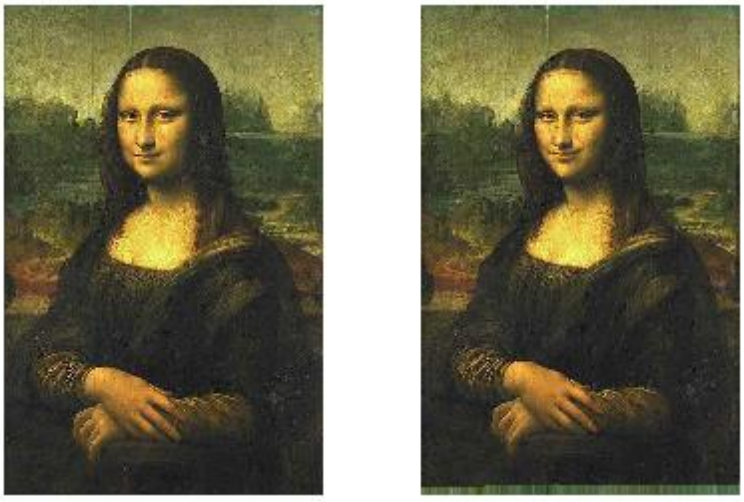


Figure1: Original image (left) and its deformation using the MLS Affine deformation method (right).

After deformation, the face is thinner and she is smiling.

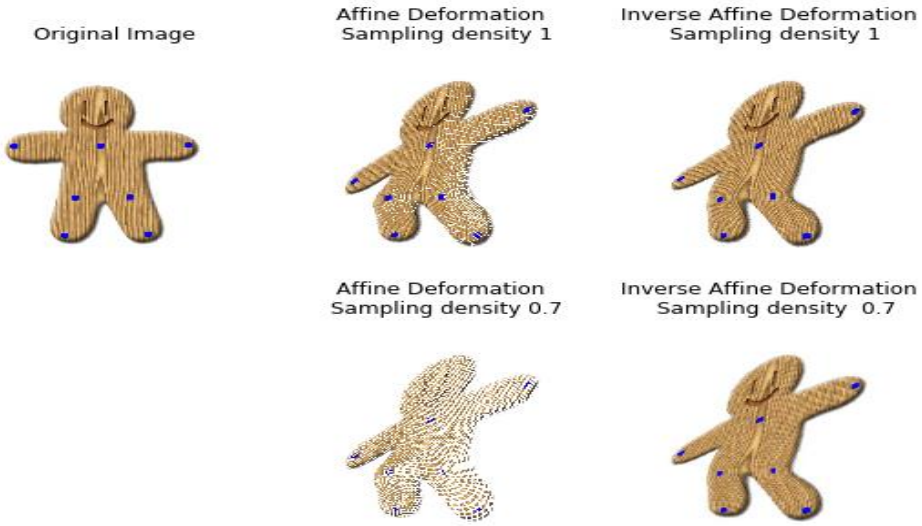


Figure2: Compare the Result between Affine Deformation and Inverse Affine Deformation using Moving Least Squares.

Figure3: Deformation using Moving Least Squares with four different α : 0.1, 0.5, 0.8,1.0.

It performs better when alpha is 0.1 or 0.5.

Change the initial man face image, the result is as follows:

Figure4 Deformation using Moving Least Squares with four different α : 0.1, 0.5, 0.8, 1.0. (image cr weibo)

3.3 Results of Image Morphing

1. Morphed the images directly

First, we do not detect the feather points and just use the equation 1 to get the morphed image with α=0.5， then we can get the Figure 5 which clearly shows the double vision.

Figure5: a simple morphed image

2. Add feature points

To verify the double vision problem, we use the API interface face++ to detect the face feather points, and we also add 8 margin points to get a better results. The feather points are shown in the Figure6.

The face++ can detect 83 face feature points for each image. In our assumption, we predict that the more feature points, the better the morphing image shows. So we try to change the number of feature points to show the effect of different points number.

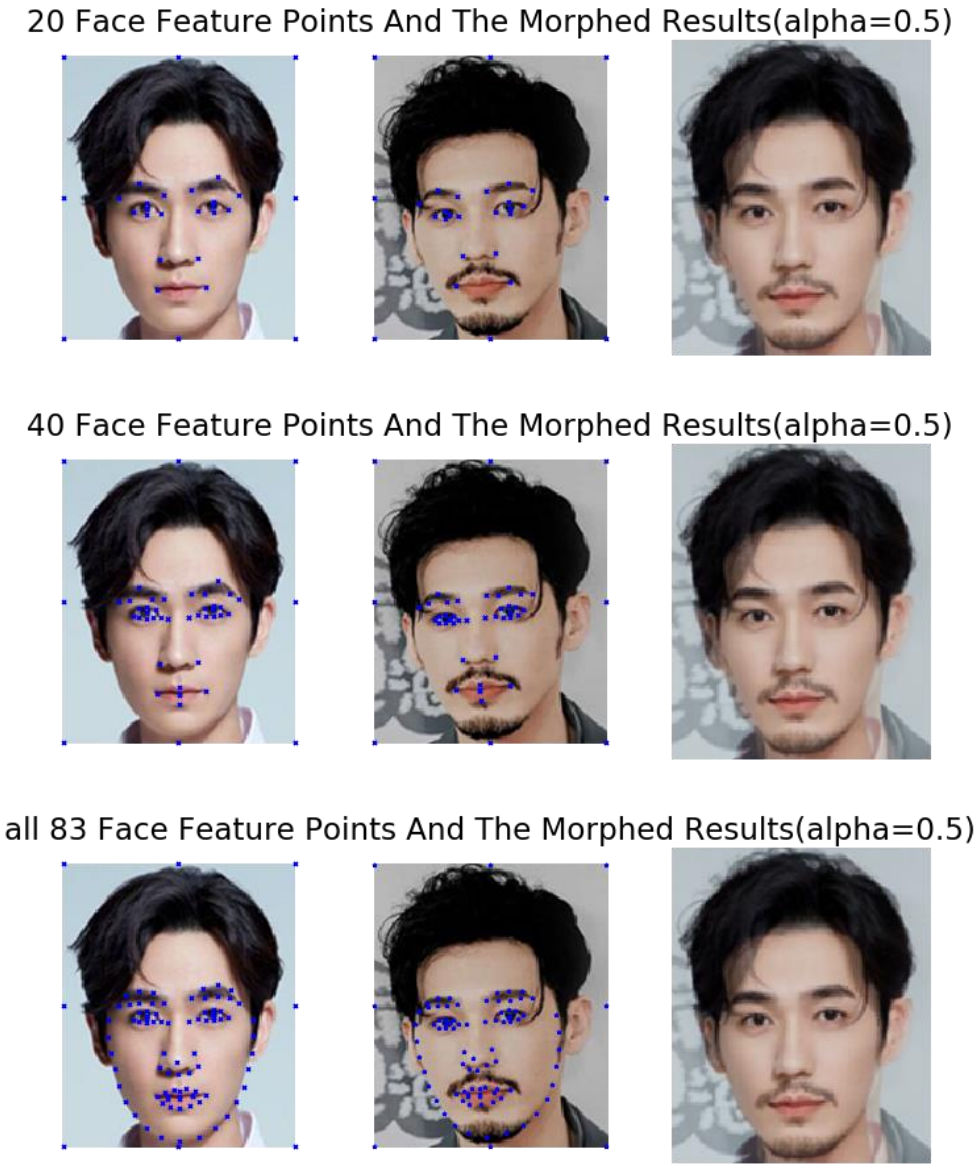


Figure 6: the feather points

Figure 6 shows that our prediction is right, the double vision becomes less obvious as we add more feature points. So in the following steps, we just use the 83 feature points that the face++ detect and 8 margin points that we add.

3. Triangulation

Now, we try to plot our delaunay triangulations for the Image I and Image J like Figure 7.

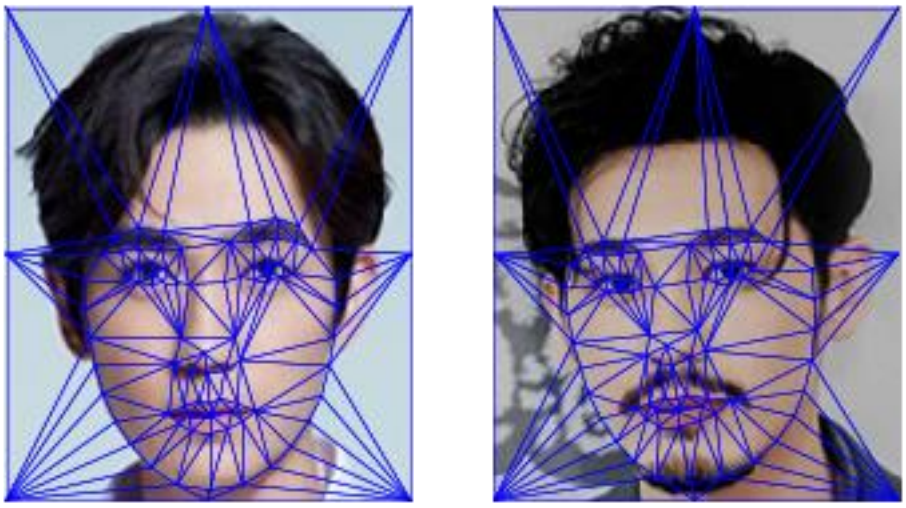


Figure 7: The Delaunay Trangulation of Image I and Image J.

4 Different Alpha

After we got the feather points, we can then got our final morphed pictures with α=0.2, 0.4, 0.6, 0.8, respectively. The final morphed pictures are shown as Figure 8.

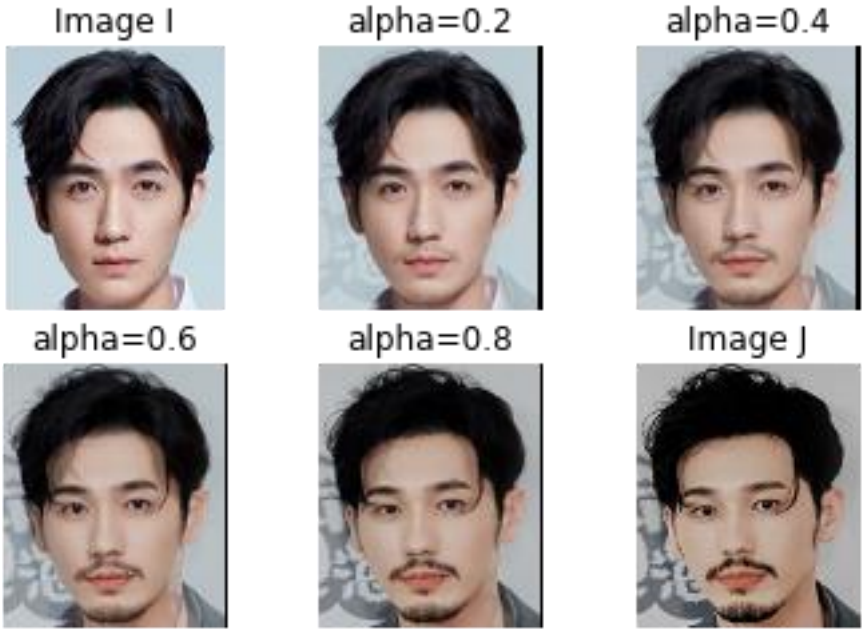


Figure 8: The final morphed images

From Figure 8, we can find that different alpha will have various morphed effect.

Based on our original images I and J, the best alpha may be 0.4.

5 Different Initial Image

From the following steps, we know that 83 feature points and alpha=0.4 will get the

best results of the morphing. But if we want to get a good results of morphing, the initial image should obey following rules:

1, The human characteristics except face between image I and image J, such as the cloth, motion, hair style and so on, should be similar, because we do not add other body feature points and the points will be morphed directly which may easily lead to the problem of double vision.

2, The pixels of Image I and Image J can be different, but their gap cannot be too large, as no-matching pixel will just be black point which will destroy aesthetic feeling.

Now, we try to use different original image to show the importance of matching.

Figure 9: The effect of different original image.

From Figure 8, we can find that the “New Image J” and the “Image I” will produce a bad “New Image M”, because the backgrounds and the face angles of the two images are not so matched.

4. GUI

For details on how to design and use the GUI, see the documentation GUI.pdf.

5. Conclusion & Suggestion

In this report, we design three algorithms to achieve image registration based on control points and feature points, and all these methods obtain good results. We also design the GUI to visualize the results after registration. At the same time, we put forward some suggestions for improvement. For example, the local affine deformation has the

problem of local unevenness after registration, which can be improved by moving least squares transformation. Image morphing requires matching of the original image.

References

[1] Xiahai Zhuang, Kawal S. Rhode, Reza S. Razavi, David J. Hawkes, and Sebastien Ourselin. A registration-based propagation framework for automatic whole heart segmentation of cardiac MRI. IEEE Transactions on Medical Imaging, 29(9):1612-1625, 2010.

[2] J. A. Little, D. L. G. Hill, and D. J. Hawkes. Deformations Incorporating Rigid Structures. Computer Vision and Image Understanding, 66(2):223-232, 1997.

[3] Scott Schaefer, Travis McPhail, and Joe Warren. Image Deformation Using Moving Least Squares. In ACM transactions on graphics (TOG), volume 25, pages 533-540. ACM, 2006.