Face Image Morphing

Armin Gholampoor
York University
arminir@yorku.ca

December 15, 2023

1 Introduction

"Morphing is an image processing technique used for the metamorphosis from one image to another. The idea is to get a sequence of intermediate images which, when put together with the original images, would represent the change from one image to the other." [8]. This process is used to create animations of smooth transitions from one image to another. When this transition is created over face images, it is called face image morphing. This technique can also be used for identity fraud and possibly other criminal activities [4].

There are two major steps in performing image morphing: Image Alignment and Transition Control. "Image alignment (also known as image registration) is the technique of warping one image (or sometimes both images) so that the features in the two images line up perfectly" [6]. This step helps the transition look as smooth as possible. Transition Control specifies the technique used to generate the intermediate images.

The simplest way of achieving Transition Control is to linearly interpolate images in the spatial domain, which does not produce good-quality transitions. Another approach is to transform both images and interpolate them to get the frequency domain representation of the intermediate images and finally perform inverse transform over them to get spatial depictions of them. There also exist Machine Learning methods to perform the task, but we do not explore them in this article.

In this work, we build on the work of [3], which presents a simple image morphing technique based on frequency domain representations of images. They manually align images and use DCT as their transformation technique. We aim to achieve two goals in this project: First, to automate the aligning procedure. Second, we will use DFT as our transformation technique.

We have to mention that the baseline paper [3] did not use any metrics to evaluate their method. However, due to the importance of metrics, we use two evaluation metrics: Peak signal-to-noise ratio (PSNR) [2] and structural similarity index measure (SSIM) [7]. To be able to compare our results with the baseline paper, we also implemented the paper's method and performed a visual evaluation of it.

The code for this project can be found at:
GitHub Link

2 Related Work

The baseline for our method is built on [3]. As mentioned before, image morphing consists of two steps: Image Alignment and Transition Control. In this section we will explain the details of methods used in each step of the paper.

2.1 Image Alignemnt

They propose two ways to perform this step: Point Based Feature Alignment (PBFA) and Line Segment Based Feature Alignment (LSBFA).

In PBFA method, they perform the alignment using projective transformation. In this method they have to specify the feature correspondence points in each image at first. Suppose each correspondence has two points $(x, y), (x', y')$ with (x, y) being in the source image and (x', y') being in the target image. Now the projective transformation is given by:

$$x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}$$

$$y' = \frac{a_3 x + a_4 y + b_2}{c_1 x + c_2 y + 1}$$

This equation can also be written in the matrix form:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -yx' \\ 0 & 0 & 0 & x & y & 1 & -xy' & -yy' \end{bmatrix} \begin{bmatrix} a1 \\ a2 \\ b1 \\ a3 \\ a4 \\ b2 \\ c1 \\ c2 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

As we can see, there are eight unknown parameters in this equation. Hence, we need at least four

points to calculate these parameters and, as a result, get the transformation.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 \end{bmatrix} \begin{bmatrix} a1 \\ a2 \\ b1 \\ a3 \\ a4 \\ b2 \\ c1 \\ c2 \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$$

Now, by manually selecting four or more points, one can solve this equation using the inverse or pseudo-inverse of the matrix on the left to calculate the parameters.

In LSBFA method, they manually mark lines across a feature in two images. Suppose this correspondence is shown by lines $AB = (x_1, y_1), (x_2, y_2)$ in the source image and $A'B' = (x'_1, y'_1), (x'_2, y'_2)$ in the target image. Let $P = (x, y)$ be a pixel in the target image and $P' = (x', y')$ be the new coordinates of pixel P in the source image. If we show the position of point P on line AB by u we have:

$$u = \frac{(P - A).(B - A)}{|B - A|^2}$$

This value is interpreted as shown below:

$$u = \begin{cases} [0, 1] & if A \leq P \leq B \\ < 0 & if A > P \\ > 1 & if B < P \end{cases}$$

If we show the perpendicular distance of P from line AB by v we have:

$$v = \frac{(P - A).(B - A)^{\perp}}{|B - A|}$$

Hence the new position, P', of point P is calculated as:

$$P' = A' + u(B' - A') + \frac{v(B' - A')^{\perp}}{|B' - A'|}$$

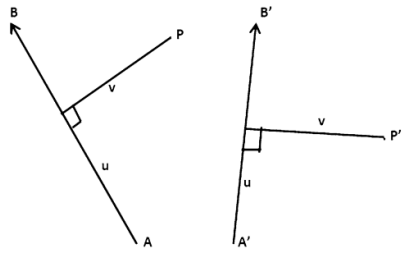

Figure 1: Illustration of LSBFA method

2.2 Transition Control

In this step, they calculate the DCT of both source and target images at first. They use the Gaussian Low Pass Frequency Filter as a means to gradually decrease the low frequencies of the source image and increase the high frequencies of the target image so that, eventually, the resulting image looks like the target image.

In the frequency domain, the Gaussian Low Pass Frequency is written as:

$$H_{lp}(u, v) = e^{-(D^2(u,v)/2(D_0^2))},$$

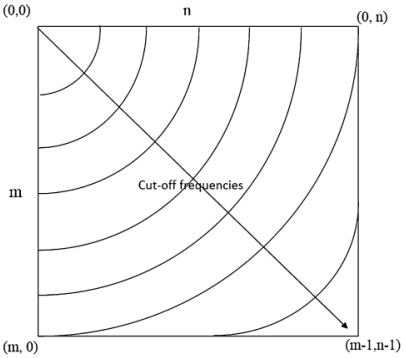

Figure 2: Cut off frequency demonstration

where D_0 is the cut off frequency and $D(u, v)$ is the distance of the point at position (u, v) from the origin $(0, 0)$:

$$D(u, v) = \sqrt{u^2 + v^2}$$

They set the cut off frequency value in ith step as:

$$D_0^i = \frac{i}{N}\sqrt{(m - 1)^2 + (n - 1)^2},$$

where m, n are the dimensions of the images, N is the total number of images of the whole morph sequence, including source and target images, and i is the number of the current step.

As a result, the frequency domain representation of the intermediate image is calculated as:

$$F_i(u, v) = S(u, v)H_{lp}^i(u, v) + T(u, v)H_{hp}^i(u, v),$$

where $S(u, v)$ is the frequency domain representation of the source image at position (u, v), $T(u, v)$ is the same for the target image, and $H_{hp}^i(u, v)$ is the Gaussian Gigh Pass Filter that is calculated as:

$$H_{hp}^i(u, v) = 1 - H_{lp}^i(u, v)$$

With this method, they generate the intermediate images, and they only have to perform IDCT to get the spatial representation of the images.

3 Proposed Solution

In this section, we go over our method in the same way we did for our baseline. We discuss the procedure of the two major steps.

3.1 Image Alignment

We use the OpenCV Python module [1], an optimized computer vision library in Python, to perform our alignment. We also use facial landmarks recognition (FLR) asset [5] to extract the features of a face, such as an eye position, nose position, and mouth position.

First, we use FLR to calculate the position of the eyes and rotate the image so that the eyes are parallel to the ground.

Suppose the position of the corner of the right eye is (x_r, y_r) and the position of the left eye is (x_l, y_l). Now, we can calculate the angle of the line connecting the eyes with this formula:

$$\theta = \arctan((y_r - y_l)/(x_r - x_l))$$

We can also calculate the center of the eyes:

$$(x_c, y_c) = ((x_r + x_l)/2, (y_r + y_l)/2)$$

Now, we can rotate the source and target images around the center of the eyes with the calculated angle so that the resulting image is aligned with the ground.

The next step is to transform the target image so that the eye and mouth positions are aligned with the source image using the affine transformation. Suppose each correspondence has two points $(x, y), (x', y')$ with (x, y) being in the source image and (x', y') being in the target image. The affine transformation is given by:

$$x' = a_1 x + a_2 y + b_1$$

$$y' = a_3 x + a_4 y + b_2$$

This equation can also be written in the matrix form:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a1 \\ a2 \\ b1 \\ a3 \\ a4 \\ b2 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

In an affine transformation, there are six unknown parameters. Therefore, we need at least three points to calculate these parameters and obtain the transformation.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a1 \\ a2 \\ b1 \\ a3 \\ a4 \\ b2 \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

Similar to perspective transformation, by selecting three or more points, one can solve this equation using the inverse or pseudo-inverse of the matrix on the left to find the parameters of the affine transformation.

We use FLR to get the position of the eyes and mouth and use the three points to perform an affine transformation to align the target image with the source image.

The resulting target image is aligned with the source image on eyes and mouth positions, which makes the transition look smoother.

3.2 Transition Control

We calculate the DFT of both source and target images to get their magnitudes and phases of their frequencies. Let α be the weighting parameter used to interpolate between source and target images:

$$\alpha_i = \frac{1 - \cos\frac{\pi \times i}{N}}{2},$$

where N is the total number of images of the whole morph sequence, including source and target images, and i is the number of the current step.

For interpolating the source and target images, we separately interpolate magnitude and phase. Instead of linearly weighting source and target images, we use polar interpolation. Suppose m_s, ϕ_s is the magnitude and phase of the source image and m_t, ϕ_t is the magnitude and phase of the target image. Let m_i, ϕ_i be the magnitude and phase of the ith intermediate image. Then we have:

$$m_i = \text{magnitude}((1 - \alpha_i)(m_s e^{j\phi_s}) + \alpha_i(m_t e^{j\phi_t})),$$

$$\phi_i = \text{angle}((1 - \alpha_i)(m_s e^{j\phi_s}) + \alpha_i(m_t e^{j\phi_t})),$$

where:

$$\text{magnitude}(z) = |z|$$

$$\text{angle}(z) = \arg(z)$$

With this information, we can construct the frequency representation of the intermediate image:

$$\text{DFT of intermediate image} = m_i * e^{j\phi_i}$$

By taking the IDFT, we can get the spatial representation of the intermediate image.

4 Results

In this section, we will compare our method with the baseline paper using three methods: demonstration, SSIM, and PSNR.

4.1 Demonstration

Here we can see the performance of our algorithm and the baseline paper

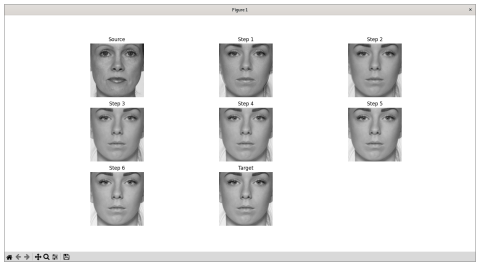

Figure 3: Baseline 1

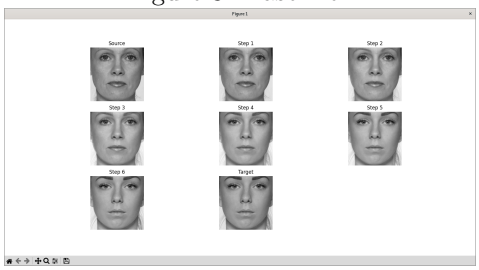

Figure 4: Ours 1

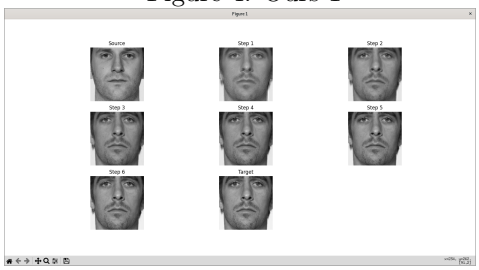

Figure 5: Baseline 2

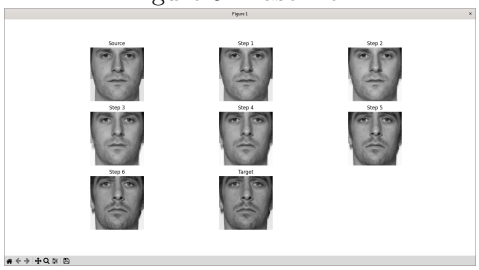

Figure 6: Ours 2

As we can see, the baseline paper quickly gets to the target image. This is because of their Gaussian Low Pass radius assignment that is increasing at a high rate. This shows their method works only for images with their used image size.

In our method, we can see the gradual transition of the source image to the target image. We can also see ghost effects that should be removed, and this is a good continuing point for future work on this topic.

Since the baseline algorithm moves quickly to the target image and the radius parameter should get fixed, we will not evaluate their methods with our metrics in the next two sections. For the next two sections, we randomly selected 2000 image pairs from all of the image pairs in our dataset (14878 image pairs in total) and calculated the SSIM/PSNR of each generated image with both source and target images. We used polar weights (using α parameter calculated above) to weight source SSIM/PSNR and target SSIM/PSNR values and calculated the weighted average for each intermediate step. Finally, we average these values among the 2000 calculated values.

4.2 SSIM

The SSIM value for each of the generated steps of our algorithm is shown in table **??**. This Value is always between 0 and 1 with 1 being most similar to the original image and 0 being least similar. This parameter shows how intermediate images related to both source and target images. The intermediate images must have values close to 1 for the morphing process to be as real as possible.

Step	1	2	3	4	5	6
SSIM	0.97	0.90	0.83	0.83	0.90	0.97

Table 1: Average of weighted SSIM values for each step

As we can see our algorithm holds high SSIM values during the generation process which is desired.

4.3 PSNR

The PSNR value for each of the generated steps of our algorithm is shown in table 2. The higher this value is, the more signal exists in the image compared to noise.

Step	1	2	3	4	5	6
PSNR	42.03	29.43	24.04	24.04	29.43	42.03

Table 2: Average of weighted PSNR values for each step

References

[1] Gary Bradski, Adrian Kaehler, et al. *OpenCV (Open Source Computer Vision Library)*. [Online; accessed 14-Dec-2023]. 2023. URL: https://opencv.org/.

[2] Fernando A. Fardo et al. *A Formal Evaluation of PSNR as Quality Measurement Parameter for Image Segmentation Algorithms*. 2016. arXiv: 1605.07116 [cs.CV].

[3] M. Shahid Farid and Arif Mahmood. "Image Morphing in Frequency Domain". In: *Journal of Mathematical Imaging and Vision* 42.1 (Jan. 2012), pp. 50–63. ISSN: 1573-7683. DOI: 10.1007/s10851-011-0273-3. URL: https://doi.org/10.1007/s10851-011-0273-3.

[4] Regula Forensics. *Facial Morphing*. https://regulaforensics.com/blog/facial-morphing/. 2023.

[5] Italo José. *Facial Landmarks Recognition*. [Online; accessed 14-Dec-2023]. 2023. URL: https://github.com/italojs/facial-landmarks-recognition/tree/master.

[6] LearnOpenCV. *Image Alignment (Feature Based) using OpenCV (C++/Python)*. 2023. URL: https://learnopencv.com/image-alignment-feature-based-using-opencv-c-python/.

[7] Jim Nilsson and Tomas Akenine-Möller. *Understanding SSIM*. 2020. arXiv: 2006.13846 [eess.IV].

[8] Rice University. *Morphological Image Processing*. 1997. URL: https://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/morph.html.