CS445 Final Project: Image Morphing: Comparison between Traditional and GAN-based Methodolgies

Seiyun Shin[1], Sohyun Park[1], and Sooha Ryu[1]

[1]University of Illinois at Urbana-Champaign

1 Background and Motivation

Morphing is a methodical approach that involves a transition from one object to another by a series of local transformations that result in the desired image. These digital transformations can be used to generate images, videos, GIFs, and other types of multimedia content. These techniques have a wide range of applications, including entertainment, art, scientific research, and more.

Traditional morphing techniques use correspondences between relevant features across participating instances to drive a warp and cross-dissolve operation [Beier and Neely, 1992]. Since these methods are mostly insensitive to the semantics of the underlying objects, they are prone to artifacts like ghosting and implausible intermediates. Correspondence points are typically provided by the user or computed automatically if sufficient similarity exists. Recently, an abundance of available data has led to its use as guidance proxies for extracting short or smooth paths between two endpoints [Averbuch-Elor et al., 2016], providing more plausible in-betweens.

Given the rising relevance of machine learning in the field of computer graphics, the data-driven morphing paradigm has been received a lot of attentions via leveraging the power of deep neural networks to learn a shape prior befitting a given source dataset for image morphing. Specifically, Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] have been used successfully in a variety of computer vision applications, including image-to-image translation and object detection, and texture synthesis, and video generation. Throughout training GANs, it is possible to learn the underlying structure of the dataset via learned embedding space. Whether or not the transformations under the learned embedding space yields smoother and realistic images compared to those directly in the pixel domain has not been fully explored.

We seek to gain a full understanding of these two modes of image transformation, specifically with the goal of distinguishing and comparing the traditional and conventional techniques employing Generative Adversarial Networks (GANs). With a simple experiment, we demonstrate the superiority of the GAN-based method that yields smoother and more realistic transformation from one image to another due to the transformation on the learned embedding space.

2 Traditional Image Morphing

2.1 Methodology and Experimental Setup

The traditional way of image morphing starts with the establishment of corresponding points between the source and target images, which is crucial for aligning the subsequent transformations. Following this, both images undergo a consistent triangulation process based on these points. The actual morphing takes place incrementally, represented by a

variable t ranging from 0 to 1. In each step, the algorithm computes an average shape using a weighted average of the corresponding points, setting the stage for the local transformations. Then the following involves affine mapping, where each triangle in the average shape is projected onto corresponding triangles in both the original and target images, ensuring precise pixel alignment. The method then calculates a weighted average value for each pixel within these triangles, optionally using interpolation to achieve smoother transitions. This process results in the generation of an image for each incremental step, representing a frame in the morphing sequence.

In this experiment, the implementation starts with preparing two distinct images, preprocessing to resize them to identical dimensions for proper alignment (explained further in 3.4). The images are then converted to grayscale to facilitate facial landmark detection, a crucial step in image processing. Using the Dlib library, the code employs a facial landmark detector and shape predictor to identify key points on each face, which are essential for aligning the images and ensuring an accurate morphing process.

Then we compute Delaunay triangulation on the landmarks of the first image. This creates a mesh of triangles over the face, forming the foundational structure for the morphing process. The core of this process is the 'morph' function, which performs affine transformations on each triangle in the source images. These transformations are warped and blended based on a specified ratio to match the corresponding triangles in the target image.

Lastly, 'generate_morph_sequence' function creates a series of images, each a frame in the morphing sequence. This function gradually adjusts the blending ratio across a specified number of frames, ensuring a smooth transition from the first to the second image. The frames are then compiled into a GIF, showcasing a fluid and visually appealing transformation.

2.2 Result

Figure 1: Sequential Frames of a Traditional Image Morphing Process: Output Video Link

As illustrated in Figure 1, the morphing project successfully demonstrates the capability to blend two facial images into a coherent sequence, generating intermediate frames where features transition smoothly from the first to the second image. Here we use Professor Derek Hoiem's two profile pictures. Given the sequential frames from Figure 1, we observe the existence of glitches in the intermediate frames, thus yielding a bit rough transformation. Due to the limitations on the affine transformation approach, this motivates the adoption of nonlinear transformations via deep learning approach.

2.3 Implementation Details

This method was implemented using Python. Key libraries utilized include NumPy for efficient array manipulation, Matplotlib for image display, OpenCV for image operations, Dlib for facial landmark detection, Scipy for Delaunay triangulation, and ImageIO for GIF creation. The Dlib library, with its pre-trained facial landmark detection model "shape_predictor_68_face_landmarks.dat," was instrumental in identifying key points for morphing. The Delaunay triangulation, a crucial step in the morphing process, was achieved using Scipy's computational geometry methods. The source images used, "source.jpeg" and "target.png," were processed to match size and orientation before being subjected to the morphing procedure. The final morph sequence was output as a GIF file "morph_sequence_.gif," which provides a visual summary of the morphing process from start to finish.

3 GAN-based Image Morphing

3.1 Methodology and Experimental Setup

GAN has a fundamental objective to complete: generate data from scratch that is good enough to trick even humans. This model, developed by Goodfellow et al. [2014], comprises of two neural networks (Generator and Discriminator) fighting with one another to generate some authentic material. The goal of two networks may be summarized as learning as much as possible about the underlying structure of the input database and then using that knowledge to create identical material that fits all of the parameters to fit in the same category. In the context of image morphing (i.e., transformation) the inputs are human faces, from which the model learns the embeddings (or features) constituting a human face, well, human.

1. Generator: As input, the generator receives a random noise vector. Following passage through the generator, which conducts numerous transposed convolutions to upsample the noise in order to generate the images.
2. Discriminator: It receives input at random from either the Real Word Sample (real sample) or the generated Images (fake sample).

As learners, we know whether the sample was real or fake, and we can use this knowledge to backpropagate a training loss in order for the discriminator to do a much better job. However, because the Generator is also a neural network, we can backpropagate all the way back to the random sample noise and thus help generate better images. As a result, the same loss function applies to both the discriminator and the generator.

The trick is to balance both networks during training. If done correctly, the discriminator will learn to distinguish even minor anomalies, while the generator will learn to produce the most realistic outputs. This can be thought of as the Generator and Discriminator are playing a mini-max game in which the goal is to solve the following optimization problem:

$$\min_G \max_D \mathbb{E}_{x\sim q_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))].$$

Here $D(x)$ denotes the probability of an image being a "real sample" image; and $G(z)$ denotes the Generator's output, with z being the random latent input. Essentially Discriminator calculates the probability that the generated image is from the "real sample" as $D(G(z))$. Basically, the generator is attempting to minimize the difference between the real and fake images in order to fool the discriminator. On the other hand, Discriminator is attempting to maximize understanding of real images in order to differentiate fake samples.

One benefit of using GAN is that it enables taking advantage of the underlying structure of the dataset which can be learned by the latent space of a Generator model after it has been trained. In particular, we use *StyleGAN* [Karras et al., 2019], a model developed by NVIDIA researchers, pre-trained on the FFHQ dataset, to learn the embedding space. The pre-trainined *StyleGAN*, capable of perturbing images in the direction of latent features, processses each image (two Professor Derek Hoiem's images in total), computing an alignment that allows a feature-based warp operation to map each input to the embedded image. Keeping track of difference between the direction of two perturbed images in the embedding space, image morphing can be accomplished by continuing to feed the warped inputs into the GAN and produces a series of frames, each corresponding to a specific point in time.

3.2 Result:

Throughout the experiment, as illustrated in Figure 2 we demonstrate that the GAN-based method yields smoother, more gradual and more realistic transformation from one image to another due to the transformation on the learned embedding space. On the other hand, it is often the case that manipulating images in the pixel domain directly can be far difficult, thus can cause rough transformations and inefficient. We refer to Figure 1 and Figure 2 for the qualitative comparison, where the traditional morphing yields some glitches on the image

during the transformations, whereas the GAN-based method yields smoother and gradual transformations.

Figure 2: Sequential Frames of a GAN-based Image Morphing Process: Output Video Link

3.3 Implementation Details

This method was implemented using Python. Key libraries utilized include NumPy, Matplotlib, OpenCV, Dlib, Glob, PIL, Torch, Legacy, and ImageIO. The Dlib library, with its pre-trained facial landmark detection model "shape_predictor_68_face_landmarks.dat," was instrumental in identifying key points for morphing. Pre-trained styleGAN by NVIDIA model was used (link for model download). The methods for pre-processing was used as the same way the data set has used for the training the styleGAN model we utilized (Github reference). Each of these libraries was chosen for its reliability and performance in handling specific aspects of the image morphing pipeline. More specific open-source code references are listed inside code files.

4 Challenge / Innovation

The project tackled a challenge of applying facial landmark detection and morphing techniques to create a seamless transition between images. Navigating the complexities of dlib's facial recognition tools required an in-depth understanding of the library's capabilities and limitations. The innovative aspect was in adapting these tools for the consistent detection of landmarks across facial structures. This morphing process, traditionally used in static images, was extended to create a dynamic sequence which represents a novel approach in our implementation.

One of the biggest challenges faced during the implementation of the GAN-based method was utilization of external GAN model. We used StyleGAN2 pre-trained model by NVIDIA for the implementation and the experiment for this project. During the utilization and some more research, we have found that there is a specific way to pre-process input images to work correctly with the model for best result. Different ways of pre-processing of images resulted in very distorted images due to the nature of this implementation of StyleGAN projection (influence of pre-processing for StyleGAN can be seen here). With the same pre-processing as for the FFHQ dataset, the dataset used for training of the GAN model, we were able to avoid poor results of projection using the model. To assure fairness, we used the same pre-processed images as input images for both the traditional and GAN-based methods. To assure fairness, we had to go back and use the same pre-processed images as source images for both the traditional and GAN-based methods.

Considering these challenges and the innovative solutions, the project merits a high score for its substantial contribution, highlighting the effort and technical expertise in translating theoretical models into a practical application.

5 Concluding Remarks

We have investigated the image morphing using deep learning (specifically GAN) and traditional methodologies. We demonstrated that the use of GAN facilitates the development of realistic in-betweens with smooth and gradual transitions, resulting in a solution that is resistant to inputs with variances in form and texture. This is due to the fact that the GAN learns an embedding space where the transformation under the embedding can be smoother than directly transforming an image of interest in the pixel (e.g., RGB) domain. One interesting direction for future work would be to perform quantitative comparisons using Fréchet inception distance (Wikipedia link).

References

Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *ACM SIGGRAPH Computer Graphics*, 26(2):35–42, 1992.

Hadar Averbuch-Elor, Daniel Cohen-Or, and Johannes Kopf. Smooth image sequences for data-driven morphing. In *computer graphics forum*, volume 35, pages 203–213. Wiley Online Library, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.