Face Merge Using Delaunay Triangulation

Arvin Aryanpour
Department of Computer Science
Toronto Metropolitan University
Toronto, Canada

Asher Andargachew
Department of Computer Science
Toronto Metropolitan University
Toronto, Canada

Daniel Tran
Department of Computer Science
Toronto Metropolitan University
Toronto, Canada

Peter Lee
Department of Computer Science
Toronto Metropolitan University
Toronto, Canada

***Abstract*—This paper explores the problem of merging facial images, a task crucial for applications in image processing and computer vision. The proposed approach employs Delaunay triangulation in combination with facial landmark detection to align and blend features from two input faces. The approach leverages geometric triangulation and pixel interpolation to achieve feature alignment and smooth transitions, however, challenges such as background ghosting and tonal mismatches persist. The results demonstrate the effectiveness of the technique in generating realistic merged faces while highlighting areas for improvement. Future refinements, such as incorporating generative models, are discussed to enhance the realism and applicability of the method.**

***Keywords*—Delaunay triangulation, face merging, facial landmark detection, pixel interpolation, background ghosting, tonal mismatch**

I. Introduction

Facial image merging has emerged as a significant area of research in computer vision, driven by its wide-ranging applications in entertainment, augmented reality, and synthetic data generation. The ability to seamlessly combine facial features from multiple images offers countless possibilities in generating realistic avatars, enhancing visual effects, and advancing biometric systems. However, this task is inherently challenging due to the geometric, lighting, and textural differences between input images. Variations in facial alignment, skin tone, and expressions further complicate the task, requiring more intensive algorithms to overcome these issues. The implications of such techniques extend beyond technical fields, making facial image merging a sought after tool.

Among various techniques explored, Delaunay triangulation has been noted for its effectiveness in geometric transformations. This approach partitions the facial area into a network of triangles based on detected landmarks, offering an exact alignment and blending of facial features. With affine transformations, Delaunay triangulation ensures that key landmarks such as the eyes, nose, and mouth are accurately aligned, providing a basis for smooth feature interpolation. The resulting alignment provides a more natural transition between the input faces. This approach addresses one of the most critical aspects of face merging: maintaining structural coherence along with aesthetic appeal.

This study addresses some of the challenges of facial image merging, such as feature alignment and the minimization of artifacts. By leveraging the strengths of Delaunay triangulation, we aim to assess the realism achievable through this method while also identifying areas for further improvement. The result of this analysis will add to the development of more sophisticated and efficient techniques for face merging, pushing the boundaries of what is possible in image processing and computer vision.

II. Material and Methods

The dataset used in this paper is a face dataset named " LFW - People (Face Recognition)" from Kaggle [1]. This dataset was only used for sampling different faces as inputs to the program. Therefore, a different dataset like ImageNet can be used. The pipeline used to create a merge between two images is as follows in figure 1:

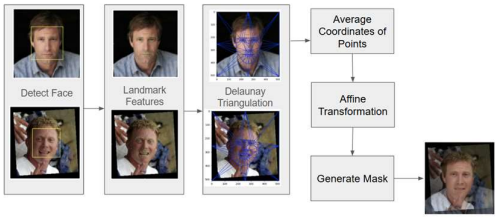

Fig. 1. The pipline needed to perform face merging using landmarked features and Delaunay Triangulation.

Firstly, the program makes sure that the image is a valid face by attempting to detect a face. This makes sure that the image contains a face that is capable of being merged in the program rather than a random object that is not considered a face. We used a model from the OpenCV GitHub named as " Stump-based 24x24 discrete(?) adaboost frontal face detector [6]". Next, landmarks were created to map unique landmark features of the two faces together. The model was outsourced from a GitHub repo named "facial landmarks recognition [4]" which uses the dlib library for the facial landmark detector. These landmarks are identified on the input face from 68 provided coordinates as shown in figure 2.

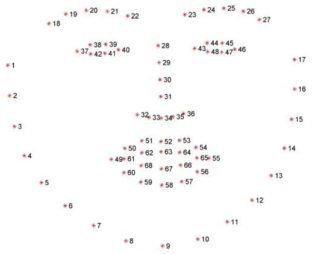

Fig. 2. Index for the 68 different coordinates for the face [3].

Each point in our program is a coordinate which has been displayed as a green dot overlaying the input face. Figure 3 demonstrates the location of these landmarks on the two input faces.

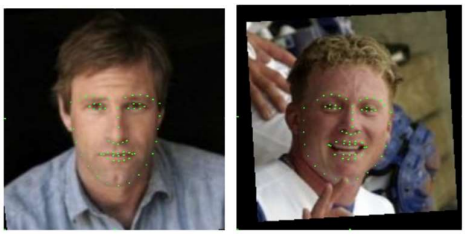

Fig. 3. The 68 different coordinates for feature landmarking on the face.

This paper utilizes the use of the Delaunay function from SciPy to create triangles from specific landmarks of the face. This next step of the pipeline, the Delaunay Triangulation method, is used to create triangles that connect two landmarks by an edge. Each edge has a distance value that shows how far one point is to another. By using this value and the relation between points, the program can understand how perform the affine transformations and generate the mask needed to merge the two faces.

There are many different way to compute the triangulation from the set of points or landmarks of the face. The Bowyer-Watson algorithm is one method to compute the triangulation. The algorithm first creates a super triangle which encapsulates all points. Figure 4 shows how three randomly plotted points are encapsulated by a super triangle.

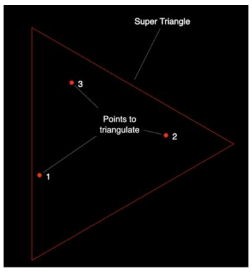

Fig. 4. Super triangle that encapsulates all points [2].

Next, an iterative step occurs where each point will be connected by an edge to the corners of the super triangle and to any points which has already been proceed by this step. Figure 5 has computed the vertices for point 1.

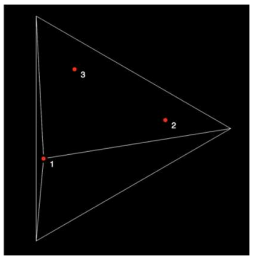

Fig. 5. Point 1 is connected to corners of super triangle and any previously processed points which are currently none [2].

Any vertices which are within the boundaries of a triangle, excluding the super triangle, will be removed. This process of going through every point and removing vertices is repeated until all points are processed. Figure 6 shows the result of all three points being processed.

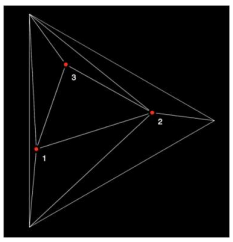

Fig. 6. Result after processing all points and removing certain vertices [2].

Finally, the super triangle and its connecting vertices are removed. This will result in the triangulation of all points. Figure 7 is the result of the complete triangulation of three points.

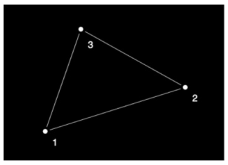

Fig. 7. The final result of computing for the triangulation of three points [2].

A unique feature of Delaunay Triangulation is how no vertex will be within the boundaries of any triangle in the set of landmarked points. This is the most important part of Delaunay Triangulation because of how there are no points that either is connected too much as it overlaps with other vertices or too little where certain points may miss critical distance values to other points. By minimizing the number of vertices while maximizing the retained information on distance values, the computational time is minimized and no redundant edges are created. Figure 8 shows what the edges of the triangles will look like for the two example input faces. Each edge is indicated as a blue line.

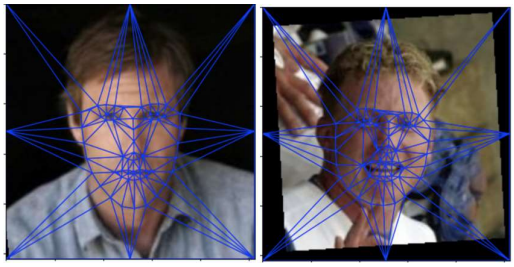

Fig. 8. Delaunay Triangulation applied on both input faces.

Finally, the two images with data from the Delaunay Triangulation method will map the landmarked features between the images and three processes are preformed. These processes are the average between the points, affine transformation to maintain a similar face position, size and orientation to one another, and finally, the generated mask which will output the concluding product of a merged face.

III. RESULTS

Using Delaunay Triangulation has helped with moving certain pixels to an average between the two faces. As noticed from the final product of the two merged faces, the result is a completely aligned face with the head where all landmarked features are placed according. Figure 9 shows the concluding output of the program after successfully merging two different faces together.

Fig. 9. Result of merging the two input faces.

However, the issue of ghosting is a problem where the background of the two images are still visible. But this ghosting effect was necessary in order to fix a different issue. For one thing, the merged face would not align with the head in terms of positioning. Notice how the edges of the merged face will not blend together with the rest of the head. The colouring between the merged face and the head is also different due to the averaging between pixels from the two input faces. Figure 10 shows the issue with a failure to blend between the face and the rest of the body.

Fig. 10. Issues with ghosting when two input faces are merged.

Note that the face itself has already been processed with affine transformation so that the two input faces have landmark features that are as closely related in position as possible. However, directly placing the merged face image on top of one of the input face images would not work because of the difference between the original input face and the merged face.

IV. DISCUSSION

We find that some potential solutions to fixing the background issue of the merge face image is to generate a body that can match the merged face. These generative methods would likely be from a GAN or diffusion model. Another option is having to blend the face to one of the input images with ghosting issues. This may help with a smooth transition between the edges of the merged face and the rest of the body. A benefit to blending is that this method is likely less computationally expensive compared to using a GAN or diffusion model. However, the difference in the skin tone may may affect the realism of the resulting merged face.

V. CONCLUSION

The results shown in this paper are rather effective face merging using Delaunay Triangulation, powered by facial landmark recognition and affine transformations. The methodology assures alignment and seamlessly integrated input faces by mapping unique facial features and averaging their positional information. Though highly effective in achieving the goal of an aligned face merge, this method faces setbacks such as disadvantages to skin tone matching and ghosting effects that negatively effect the realism of the final output. Results from the program show how important the balance between reduction of computational overhead and retaining facial features is.

Adopting GAN or diffusion models would potentially lead to more visually pleasant and practically feasible face merges by fine-tuning blending strategies. Overall, the research and practical implementation provide valuable insights and a solid foundation for further development in automated face-merging technologies, whose applications span much farther than the realm of computer vision.

VI. FUTURE WORK

To further improve the results from our paper, we believe that having to use blending tools like Poisson Image Editing can create realistic lighting and colours of an object in a specific environment. However, the issue with this method is that it is not perfect due to potential mismatch in the colour of certain pixels thus making the resulting post processing for blending seem artificial.

To combat this feeling of an artificial like result, another method in which this paper has not tried but will suggest is using an image generative approach such as using StyleGAN may be more preferable to get the best results for the merged face. This only issue is that the latest models being StyleGAN 3 released in 2021 and StyleGAN-T released in 2023, require good hardware which we cannot perform. The complexity of using GAN or diffusion models to generate a realistic face may provide a better result but the computation time is too much to perform for this paper.

The idea of using GAN to merge the two input faces is similar to the concept of deepfakes. GANs have two parts, the generator and discriminator. The generator is an autoencoder which consist of the encoder and decoder. The autoencoder will encode an input image to an encoder, thus translating it to latent space. This output of the latent space is inserted into the decoder where a reconstructed face will be outputted from the autoencoder. For deepfakes, the autoencoder process is performed on both the original image of a face and the input image of the face. The input image will replace the face from the original image. Figure 11 shows a visual demonstration of the autoencoder and its process.

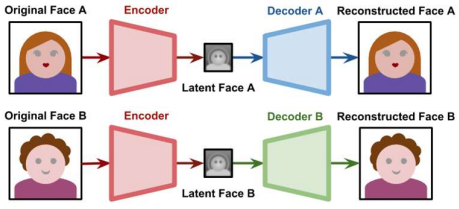

Fig. 11. Result of the two different faces using autoencoders for deepfakes [5].

This output of the latent space is inserted into the decoder where a reconstructed face will be outputted from the autoencoder. The deepfake utilizes the use of the latent space between the two autoencoders and swaps them in order to replace the face from the original image with a new face. Figure 12 shows this swapping process to replace the face.

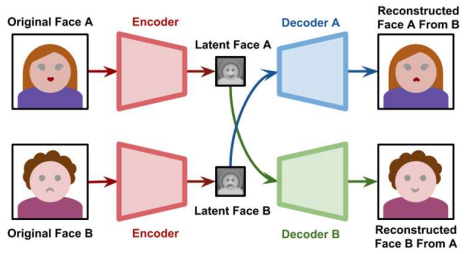

Fig. 12. The swapping of latent space between autoencoder to produce a deepfake [5].

When it comes to face merging, we can utilize the idea of swapping latent space in the deepfake process. But instead of swapping, the two latent space will be averaged to create a new latent space. Then a separate decoder will output the resulting merged face. Figure 13 shows the idea of create a new latent space using the two generated latent spaces. Latent face A and B and merged to create latent face C. This new latent face C is passed to a decoder to create the reconstructed face as the product of the two input face merged.

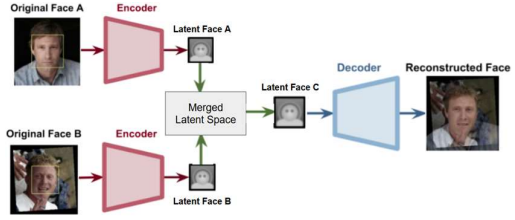

Fig. 13. Reconstructing a new face using merged latent space from separate encoders.

The benefit and unique feature of the GAN is the discriminator part of the model. The discriminator will determine if the reconstructed face is actually like a face. The if the output face is rejected, meaning the discriminator does not think that the output face is realistic, the generator will restart and try to create a better reconstructed merged face based on the feedback from the discriminator. Overall, the best result will be to have a merged face where a newly generated body including head can match the generated merged face.

ACKNOWLEDGMENT

The contributions of Dr. Omar Falou's teachings laid the foundation of computer vision knowledge that were applied throughout the conception and development of this work.

REFERENCES

[1] A. Anand, "LFW - People (Face Recognition)" 2019, kaggle dataset. [Online]. Available: https://www.kaggle.com/datasets/atulanandjha/lfwpeople

[2] A. Moussa, "Bowyer-Watson Algorithm for Delaunay Triangulation," Gorilla Sun, Jun. 12, 2022. https://www.gorillasun.de/blog/bowyer-watson-algorithm-for-delaunay-triangulation/

[3] A. Rosebrock, "Facial landmarks with dlib, OpenCV, and Python," pyimagesearch, Apr. 03, 2017. https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[4] I. José, "facial landmarks recognition" 2018, gitHub repository. [Online]. Available: https://github.com/italojs/facial-landmarks-recognition/tree/master

[5] S. Sarker, "DeepFake Creation Tutorial" 2023, kaggle code. [Online]. Available: https://www.kaggle.com/code/soumicksarker/deepfake-creation-tutorial

[6] R. Lienhart, "Stump-based 24x24 discrete(?) adaboost frontal face detector" OpenCV, 2024, gitHub repository. [Online]. Available: https://github.com/opencv/opencv/tree/master/data/haarcascades/haarcascade_frontalface_default.xml