Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 9, No. 3, 2024

A Fast Face Changing Method Adaptable to Various Postures

Huazheng Huang, Guihua Huang, Weijun Zhuo, Minghui You

School of Computer Science, Guangdong University of science and technology, Dongguan, Guangdong, China

Abstract: Image face-changing technology is influenced by skin color and posture. After changing faces, the contour information or skin color information of the user's face will be lost, resulting in low similarity between the changed image and the user's face in the source image. This paper introduces a rapid face-changing method that can adapt to various poses. Regardless of whether the facial poses in the source image and template image differ significantly, this method can retain the facial contours and skin tone in the user's source image without deformation, resulting in a high similarity between the transformed image and the face in the source image. Additionally, it offers low computational complexity, quick processing speed, and a small amount of computational effort.

Keywords: Triangular Affine Deformation Algorithm; Multi-pose Face Transformation; OpenCv.

1. Introduction

Face-changing technology, facilitated by facial fusion technology, is suitable for corporate marketing endeavors such as advertising, brand promotion, and event ticketing. It can also be utilized by companies in gaming and film production to develop fusion templates based on game and film characters, thus creating derivative merchandise. It can also be employed for personalized image customization for users, such as in online wig shops, clothing stores, hair salons, and cosmetic surgery clinics. Additionally, embedding facial fusion capabilities into camera apps enhances beauty features by making them more diverse and engaging.

The literature [1] investigated that most face fusion methods will cause the contour, skin color, and posture of the face after face swapping to be affected by the template image, resulting in the loss of the contour or skin color information of the user's face, resulting in low similarity between the face swapped image and the user's face in the source image. Even when there is a significant difference in facial posture between the source image and the template image, it can lead to a severely deformed face after changing faces. In some application scenarios, it is necessary to highly restore the face of the source image after changing the face. For example, in hairstyle design applications or devices, wig trial applications or devices, users need to display their face in the hairstyle model image (template-image) and try not to change the user's original facial contour and skin color to determine whether the user is suitable for a certain hairstyle or wig.

There are currently two mainstream methods for solving different pose recognition problems: one uses hand-drawn or learns features from different poses [2], making a trade-off between invariance and recognizability, often unable to effectively handle large angle poses. Another approach is to align a two-dimensional image model with a three-dimensional image model with a precise identity [3], using LBP to consider local changes and rendering a frontal facial image using three-dimensional geometric transformations. However, when this method encounters a large number of pose images with significant differences, the loss of texture details is severe, and the performance is not satisfactory [4].

2. Related Work

At present, the fusion of facial images with different skin tones or styles is generally carried out using Generative Adversarial Networks (GAN) [5]. For example, Isola et al. [6] proposed the conditional GAN model Pix2Pix, which achieved high synthesis quality. However, this model requires paired training data, and each pair of images must be strictly aligned to train image style transfer models on non-paired data [7].

In summary, facial fusion technology currently generally uses adversarial networks or convolutional neural networks for model training, model reconstruction, or searching for all Delaunay triangles [8] to perform a large number of affine transformations [9] and then performing Poisson fusion [10]. These methods can lead to problems such as high computational complexity, high computational complexity, slow speed, and complex implementation.

Therefore, this article proposes a rapid face-changing method that can adapt to various poses. Regardless of whether the facial poses in the source image and template image differ significantly, this method can retain the facial contours and skin tone in the user's source image without deformation, resulting in a high similarity between the transformed image and the face in the source image. Moreover, the computational complexity is low, and the speed is fast. This article also provides a system that can quickly display the transformed images generated by the server module on the front end of the webpage with less computational complexity and can match source images and template images of different poses. The transformed images maintain a high degree of similarity with the faces in the source images.

3. Method and Implementation

3.1. Implementation of Facial Keypoint Detection Methods

In this article, posture refers to the facial angle features of a person's head tilting and turning up and down in a flat image. The image coordinate systems used in this article are all planar Cartesian coordinate systems. Take the top left corner

of the image as the coordinate origin, the x-axis to the right, and the y-axis to the bottom.

To perform facial fusion, then the first step is to use OpenCV to recognize the source and template image separately.

Firstly, facial recognition is performed on the source image using OpenCV [11] to obtain a rectangular region of the face in the source image, which is stored in a rectangular-type structure (for convenience in the description, the name of the rectangular-type structure is marked later); Based on the mark, 68 key points of the user's face are identified [12], and these 68 key points are stored in an array of coordinate point types (for convenience in description, the array name is marked with markers later). Convex hull points are calculated for these 68 key points and stored in an array of coordinate point types (for convenience in the description, the array name is marked with markers later). These convex hull points store the point coordinates on the face contour in the user's source image.

Secondly, openCV is used for facial recognition on template images (such as hairstyle model images) to obtain the rectangular area of the face in the template image, which is stored in a rectangular type structure (for convenience in description, the name of the rectangular type structure is identified using templateRect in the following text); Based on templateRect, 68 key points of the user's face are identified and stored in an array of coordinate point types (for convenience in description, template points are used later to identify the array name). Convex hull points are calculated for these 68 key points and stored in a group of coordinate point types (for convenience in description, templateHull is used later to identify the array name). These convex hull points store the point coordinates on the face contour in the template image.

3.2. Implementation of a Method for Adapting to Multi-pose, Rapid Face Changing

On the basis of facial keypoint recognition in the previous text, the offset angle between the left and right eye corner coordinate points is stored in the source image in MarkPoints, and the X-axis is quickly calculated (for convenience in description, angleSrc is used later to indicate the offset angle). Quickly calculate the offset angle between the left and right eye corner coordinate points in the template graph stored in template points and the X-axis (for convenience in description, the offset angle is marked with an angle in the following text). Calculate the angle of rotation required for the user source image to be consistent with the facial pose in the template image, and rotate to indicate it, where rotate = angle - angleSrc.

Rotate the user source image as a whole to obtain the rotated source image. Re-perform facial recognition on the rotated source image to obtain a new rectangular area that replaces the area in the mark; Identify 68 key points of the user's face based on the mark, replace these 68 key points with the coordinate points in the marks, calculate convex hull points for these 68 key points, and replace the coordinate points in the markHull with the coordinates of these convex hull points.

Simply calculate the two triangles in the rotated user source image that need to undergo triangular affine transformation. The calculation method is as follows:

Obtain the bounding rectangle of markHull, mark it with markHullRect, and change the bottom right corner coordinate of mark to the bottom right corner coordinate of markHullRect. The two triangles that require triangular affine transformation are divided by a line connecting the top right and bottom left corners in the marked rectangle. The three vertices of the triangle near the top left corner are saved in array t1, and the three vertices of the triangle near the bottom right corner are saved in array t2.

Obtain the bounding rectangle of templateHull and mark it with templateHullRect. If the X-coordinate of the top left corner of templateRect is greater than the X-coordinate of the top left corner of templateHullRect, then change the X-coordinate of the top left corner of templateRect to the X-coordinate of the top left corner of templateHullRect. If the upper left corner Y coordinate of templateRect is greater than the upper left corner Y coordinate of templateHullRect, then change the upper left corner Y coordinate of templateRect to the upper left corner Y coordinate of templateHullRect. If the X-coordinate of the bottom right corner of templateRect is smaller than the X-coordinate of the bottom right corner of templateHullRect, then change the X-coordinate of the bottom right corner of templateRect to the X-coordinate of the bottom right corner of templateHullRect. If the bottom right Y coordinate of templateRect is smaller than the bottom right Y coordinate of templateHullRect, then change the bottom right Y coordinate of templateRect to the bottom right Y coordinate of templateHullRect.

In the template diagram, the two triangles that need to undergo triangular affine transformation are divided by a line connecting the top right and bottom left corners in the templateRect rectangle. The three vertices of the triangle near the top left corner are saved in array t3, and the three vertices of the triangle near the bottom right corner are saved in array t4.

Create a black image of the same size as the template image (marked as maskRect), transform the triangle area formed by t1 in the rotated source image into a triangular affine shape, and copy it to the t3 area in maskRect. Transform the triangle area formed by t2 in the rotated source image into a triangular affine shape and copy it to the t4 area in maskRect. Finally, in maskRect, the complete affine deformed image of the face region in the source image is presented, which is consistent in position and size with the face region in the template image. All areas outside the face region in maskRect are black.

By using the seamless fusion function SeamlessClone of OpenCV, the maskRect face area is covered within the area contained in the template image's convex hull point templateHull, and the effect image after face replacement is displayed to the front-end user.

The method for calculating the offset angle between the line connecting the coordinate points of the left and right eye corners and the X-axis is as follows:

Step1: Calculate the distance w1 between the line connecting the left eye corner coordinate point PL and the right eye corner coordinate point PR:

$$w1 = \sqrt{(PR.Y - PL.Y)^2 + (PR.X - PL.X)^2} \quad (1)$$

Step 2: Calculate the sine value sin of the angle between the line connecting the left eye corner coordinate point PL and the right eye corner coordinate point PR and the X-axis. Note that to ensure that the subsequent rotation direction is correct, subtract the right eye Y coordinate from the left eye Y coordinate:

$$sin = (1.0f * (PL.Y - PR.Y)) / w1 \quad (2)$$

Step 3: Calculate the offset angle yaw using the anti-sine

function using the sine value sin:

$$yaw = 180*Math.Asin(t2)/Math.PI \quad (3)$$

Step 4: Correct the degree of yaw offset angle. If the absolute value of yaw is less than n (for example, n=3), make the value of yaw 0. If the offset angle is within a negligible range, it is considered that there is no offset.

4. Experiment and Result Analysis

4.1. Implementation of the System

Figure 1 is a schematic diagram of the system structure implemented in this article, and Figure 2 is a flowchart of the face-changing method steps provided in this article. As shown in Figure 1, the server-side face-changing module includes a loading and storage submodule 106 for image files, a face recognition submodule 107, a pose alignment submodule 108, an affine transformation submodule 109, and a seamless image fusion submodule 110.

The front-end display module includes login submodule 101, user image upload submodule 102, template image display and selection submodule 104, and rendering display submodule 105.

The user image upload submodule provides a user interface for users to upload or capture user source images, save them to the specified disk directory on the server, and save the path (URL) of the user source image in the server to the database, and store the source image URL in the Session buffer.

After the user logs in correctly, the login submodule retrieves the previously uploaded user source image URL from the database and stores it in the Session buffer.

The template image display and selection submodule retrieves template image URLs that match skin color, hair length, and facial shape from the database and displays the template image on the page based on these URLs. After selecting a template image, save the URL of the template image to the Session buffer and automatically redirect to the rendering display submodule. The rendering display submodule displays images on the page based on the user source image URL and template image URL in the Session buffer, providing the user with a button to generate the rendering. After the user clicks the button, they pass the source image URL, template image URL, and the upcoming rendering URL to the server-side face-changing module for processing. They wait for the server-side face-changing module to complete the operation and save the rendering to the server disk directory specified by the rendering URL, and the page will automatically refresh to display the rendering.

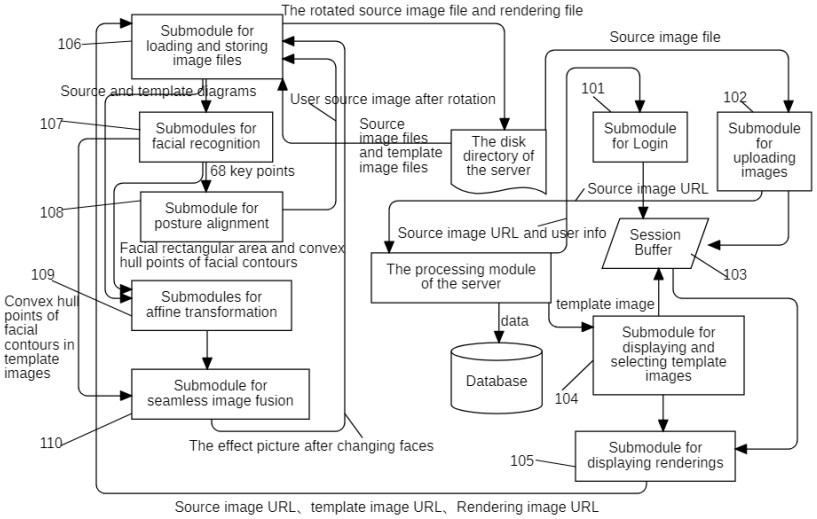

Fig 1. Schematic diagram of face-changing system structure

The loading and storage submodule 106 of image files first obtains the source image URL and template image URL passed by the front-end display module, converts the two URLs into corresponding server disk paths, and uses OpenCV's ImRead function to load the source image and template image into memory. The ImWrite function of OpenCV is used to save the rotated user source image and the transformed effect image from memory to the server disk path.

The facial recognition submodule will perform facial recognition on the user source image and template image in the memory and save the rectangular area of the face to the memory structure variable, 68 key points to the memory array, and the convex hull points of the face contour to the memory array.

The pose alignment submodule will calculate the facial rotation angles based on 68 key point groups of the user source image and template image in the memory and rotate the user source image in memory to match the facial pose and template image. Then, the rotated user source image will be saved to the specified disk directory on the server.

The affine transformation submodule will calculate the vertices of the affine triangle based on the variable of the face rectangle region of the user source image and template image in memory, as well as the array of convex hull points of the face contour. It will also perform the triangular affine transformation on the face region of the user source image in memory to obtain the complete face image of the source image after the triangular affine transformation and save it to memory.

The image seamless fusion submodule covers the complete facial image after triangular affine transformation in memory to the area contained in the convex hull points of the facial contour in the template image and obtains the effect image after face replacement, as shown in Figure 3.

4.2. Result Analysis

Compared with existing technologies, this article has the following advantages: when transforming the source image face into a face with the same pose and region as the template image, only the left and right eye corner lines and the X-axis offset angle are used for rotation. At the same time, only two effective regions are selected to complete the triangular affine

transformation without the need to calculate numerous Delaunay triangles for triangular affine transformation.

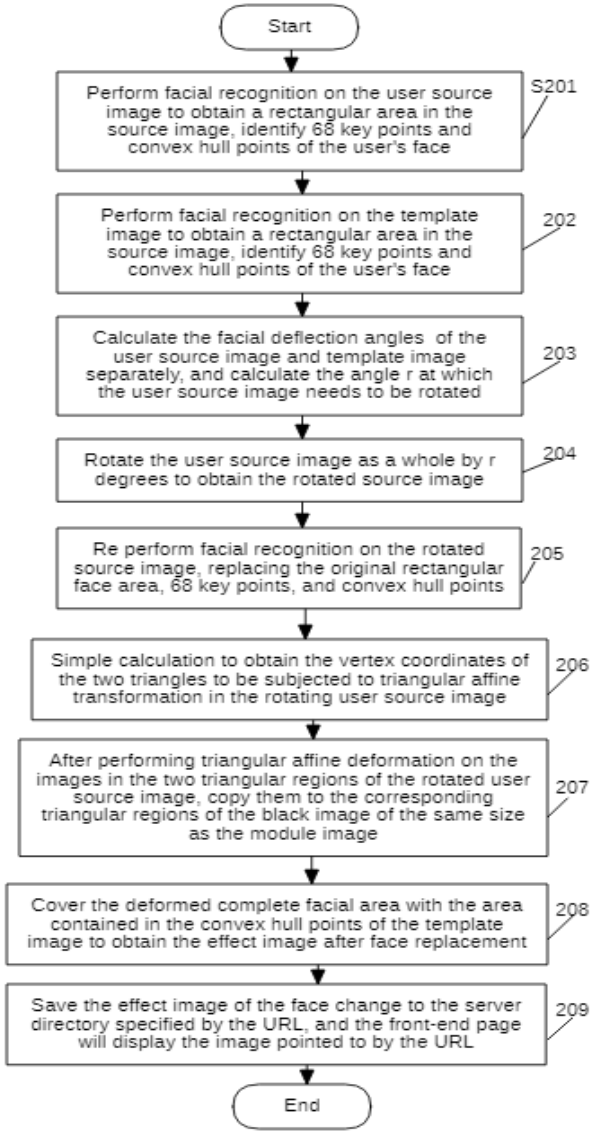

Fig 2. Process flow diagram of face-changing method steps

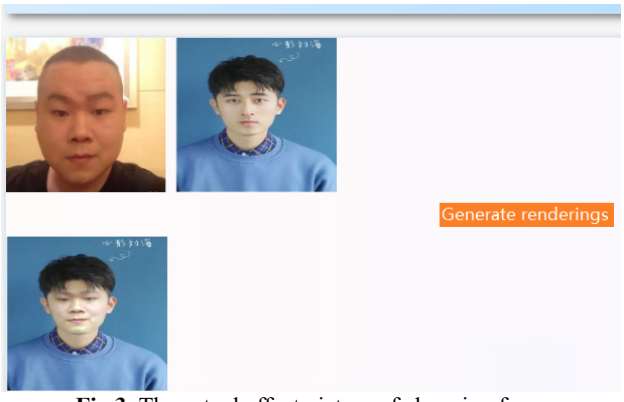

Fig 3. The actual effect picture of changing faces

Therefore, the computational complexity is small, the implementation is simple, and the effect image after face transformation can be generated and displayed quickly to front-end users. It can also ensure that user images with different poses and template images maintain high similarity with the user's face after face transformation without deformation.

Acknowledgments

This work is supported by the Guangdong University of Science and Technology (GKY-2023HX-078, GKY-2022K YYBK-19).

References

[1] WANG Chang-bo and WANG L, "Method of HumanFaces Synthesis for Virtual Display," Journal of Computer and Modernization, no. 10, pp. 68-71, May 2010.

[2] CHEN D, CAO X, WEN F,and SUN J, "Blessing of dimensionality: High- dimensional feature and its efficient compression for face verification," CVPR, vol. 389, pp.3025-3032, 2013.

[3] TAIGMAN Y, YANG M,RANZATO M,et al, "Deepface: Closing the gap to human-level performance in face verification," CVPR, vol.220, pp.1701-1708,2014.

[4] YU Si-hong, YU Yang, WANG Er-bo, and WANG Cun-rui, "Improved Algorithm of Face Fusion Based on Region Transformation, "Journal of Dalian Minzu University, vol. 23,no.1, pp.81-84,2021.

[5] Goodfellow I., Pouget-Abadie J., Mirza M., et al, "Generative Adversarial Nets, "Advances in Neural Information Processing Systems, pp.2672-80,2014.

[6] Isola P.,ZHU J.Y.,ZHOU T, et al, "Image-to-Image Translation with Conditional Adversarial Networks, "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1125-34,2017.

[7] WANG Tong-ping, "An Improved Heterogeneous Face Image Synthesis Algorithm Based on CycleGAN," Journal of Modern Computer, vol.5, pp.53-56,2020.

[8] Ke Yu-ding, "Research on Local Face Registration Methods, "Shenyang: Shenyang University of Technology,2019.

[9] ZHOU Da-nian, MA Guo-jun, DING An, WANG Biao, and LIU Wei, "Side-Scan Sonar Image Mosaic Based on Affine Transformation Method of Triangulation, "Electronics Optics & Control, vol.26, no.10, pp.17-21,2019.

[10] XIONG Xhi-Fei and SHEN Jiang-hai, "A Multi-focus Image Fusion Algorithm Based on Multidimensional Adaptive Filtering and Poission Fusion, "Science Technology and Engineering, vol.23, no.24, pp.10427-10436,2023.

[11] Long Hui, Zhu Meng-chun,Deng Ya-qian,Yao Xiao-yin, and Liu Wei, "Intelligent Campus Face Access Control System Based on OpenCV, "Journal of Industrial Control Computers, vol.36, no.9, pp.73-75,2023.

[12] GAO Qiang, PAN Jun, and LIU Wei, "A Fatigue Detection System for Flight Dispatcher Based on Face Feature Points, "SOFTWARE, vol.45, no.3, pp.37-41,2024.