# Facial Expression Real-Time Animation Simulation Technology for General Mobile Platforms Based on OpenGL

Mingzhe Cao
Department of Tourism and Arts, Yangzhou University Guangling College, Yangzhou, 225000, China

***Abstract*—With the popularization and development of mobile devices, the demand for image processing continues to increase. However, the limited hardware resources of mobile devices make traditional CPU computing unable to meet the requirements of real-time image processing. In response to the limited rendering resources of mobile platforms, this study adopts OpenGL for graphic interface design and animation simulation of facial expressions to control the changes of facial expressions in real-time, achieve facial expression animation simulation, and develop effective expression fusion methods. By combining rich rendering effects such as particle effects, facial expressions can be expressed more realistically and interestingly in 3D models. The results indicated that the research method only required less than 50 MB of memory, and the average accuracy of facial expression recognition had significantly improved. The final normalized average error level was close to 4%, with higher accuracy. The processing speed of each image was around 19.4ms, which could achieve animation simulation of facial expressions and had strong universality and flexibility. This method optimizes the real-time performance, stability, and user experience of facial expression real-time animation simulation, which can meet the needs of different application scenarios.**

***Keywords*—OpenGL; mobile platform; facial expressions; animation simulation; rendering**

## I. Introduction

In computer graphics, due to the emergence of OpenGL technology, graphics has entered a new era. OpenGL is widely used in 3D animation, Computer-Aided Design (CAD), visual simulation, and other fields. Its high quality and high-performance characteristics enable developers to create high-quality 2D and 3D graphics. Whether for broadcasting, CAD/computer-aided manufacturing/computer-aided engineering, entertainment, medical imaging, or virtual reality development, OpenGL provides outstanding graphics quality and performance. OpenGL can make full use of the functions of modern graphics hardware, such as Graphic Processing Units (GPU) parallel computing, and texture mapping, to provide higher graphics rendering performance and effect. This hardware-accelerated support makes OpenGL excellent at handling complex graphic effects, such as shadow, reflection, anti-aliasing, etc. Therefore, OpenGL has become one of the ideal choices for graphics rendering due to its high-quality graphics rendering, cross-platform compatibility, rich development tools and resources, and hardware acceleration support. In traditional graphics, rendering is the process of texturing the surface of an object. OpenGL provides a new texture mapping method, which maps textures to the real world through a transformation matrix [1]. A transformation matrix can transform a 3D object into a 2D image, which can then be displayed on the screen. The transformation matrix can realize the 3D model view transformation, model transformation, clipping, projection transformation, and viewport transformation. OpenGL is a cross-language and cross-platform application programming interface for rendering 2D and 3D vector graphics. Whether it is 3D animation, CAD, or visual simulation, visual computation programs take advantage of OpenGL's high graphics quality and high-performance characteristics. OpenGL has a good structure, intuitive design, and logical commands. Compared to other graphics packages, OpenGL has very little code and therefore high execution speed. In addition, OpenGL encapsulates information about the basic hardware, so that developers do not need to design for specific hardware characteristics. OpenGL-based graphics applications can run on many systems, including a variety of user electronics, PCS, workstations, and supercomputers. As a result, OpenGL applications can be adapted to various target platforms chosen by the developer [2-3].

With the popularization and development of mobile devices, the demand for mobile image processing is increasing. However, due to the limited hardware resources of mobile devices, traditional CPU computing cannot meet the requirements of real-time image processing. A review of the relevant mainstream products on the market reveals that the majority of video special effects and facial expression simulations are based on the display of 2D animation. In contrast, the expression of 3D animation requires a significant amount of computing resources, making real-time performance difficult to achieve, and it is rarely promoted or applied in products. At present, most of the algorithms based on data structures and mathematical methods are used in research work, but such methods require computers to have a high computing speed, and the running speed on mobile platforms is slow [4-5]. The objective of this research is to develop a Facial Expression Animation Simulation (FEAS) system that can drive a virtual 3D model through facial expression unit parameters based on face key point detection through the front camera on a universal mobile platform. The system will then be encapsulated into operational applications that meet the requirements of real-time performance, low cost, practicality, stability, and maintainability. To adapt to the characteristics of the mobile platform and achieve the real-time goal, a universal real-time FEAS system based on OpenGL technology is developed.

OpenGL is used for graphic interface design, which can be run on the mobile platform, and the 3D Max model file format is adopted. The 3D model is drawn by OpenGL, and then the texture mapping is carried out by OpenGL to realize the real-time simulation of facial expression animation.

The article conducts research through six sections. Section II is a review of the research status of OpenGL in real-time FEAS applications. Section III is the research on 3D facial expression animation methods based on OpenGL. Section IV is performance validation of the proposed model. Results and discussion is given in Section V. Section VI is the conclusion.

## II. Related Works

Bossér L et al. developed an underwater target echo signal strength prediction method based on image processing technology. By injecting code into the segment shader stage of the OpenGL graphics pipeline, the light reflection problem was transformed into an acoustic reflection problem. Compared to the Kirchhoff method, this method had higher computational accuracy [6]. Calabuig B E et al. improved the 3D engine used for real-time CAD geometric representation by using OpenGL Shaders to make ray tracing and radiation rendering techniques more realistic. It improved computing power by solving visualization problems and optimizing data. After verification, the model provided correct results for both computer-optimized 3D engines and network environment 3D engines [7]. Yin J et al. proposed a new lattice Boltzmann method to reduce the computational time, memory allocation, and complexity of existing algorithms for high-precision graphics processing units. It used OpenGL-based Compute Shaders GPGPGPU technology to accelerate and avoid the storage of distribution function components to reduce memory allocation size. The spatial accuracy was tested through 2D and 3D lid drive cavity benchmark cases, with high accuracy [8]. Rémi F et al. proposed a new method based on the OpenGL4 framework to achieve GPU-based high-order mesh rendering and almost pixel-accurate rendering of high-order solutions. This method used OpenGL fragment shaders to calculate precise solutions for each pixel by transferring degrees of freedom and shape functions to GPUs with textures. Compared to standard techniques, it eliminated linear interpolation steps, reduced memory usage, and improved rendering accuracy and speed [9].

Pham H X et al. proposed using Recurrent Neural Networks (RNNs) to achieve time-varying contextual nonlinear mapping between audio streams and facial micro movements, to drive 3D mixed-shape facial models in real-time. It depicted different speech generation actions in the form of lip movements, automatically estimating the speaker's emotional intensity with high accuracy [10]. Berson et al. proposed a generative RNN to make facial animation editing faster and easier for non-experts. It generated realistic movements in designated parts of existing facial animations based on user-provided guidance constraints. The experiment showed that the system had strong usability in scenarios such as motion filling, expression correction, semantic content modification, and noise filtering during occlusion [11]. Pumarola et al. proposed a weakly supervised strategy to train the model to describe the anatomical facial movements that define human expressions using continuous manifolds. This strategy annotated images through activated action units and utilized a new self-learning attention mechanism to make the network robust to constantly changing backgrounds, lighting conditions, and occlusion, with high performance [12]. Paier W et al. developed an interactive animation engine using deep learning to achieve more realistic rendering in difficult areas such as the mouth and eyes of animated faces. It described an automatic pipeline for generating training sequences composed of dynamic textures and consistent 3D facial model sequences through simple and intuitive facial expression editing visualization. It also trained a variational auto-encoder to learn the low dimensional latent space of facial expressions for interactive facial animation [13]. Huang Z et al. proposed a real-time simulation method for humanoid robot facial expression imitation based on a smooth constrained inverse mechanical model by combining deep learning models with motion smoothing constraints to improve the spatio-temporal similarity and motion smoothness of facial expression imitation. This method sent the generated optimal position sequence to the hardware system, making it consistent with the performer's spatio-temporal characteristics, with a deviation of less than 8% [14].

In summary, although researchers have proposed many methods to improve real-time FEAS performance and achieved certain results, the optimization scheme still needs improvement. Therefore, this study aims to achieve facial animation simulation in different application scenarios through the OpenGL platform.

## III. A 3D Facial Expression Animation Method Based on OpenGL

This study is based on the OpenGL ES computational shader, which accelerates image processing algorithms through GPU parallel computing for different memory access and computational characteristics to achieve higher performance. Detailed implementation steps for face detection, facial keypoint localization, and tracking are introduced.

### *A. Design and Optimization of Facial Expression Capture Algorithm Module*

There are many types of facial expression capture devices, and traditional devices are mostly bulky and have larger machine sizes. Due to the popularity of mobile Internet and mobile devices, this study selects the video camera attached to mobile devices as the capture device. The facial expression capture module consists of various algorithms, including image-based facial detection and facial key point localization [15]. Facial detection is achieved by obtaining image data from videos or cameras, normalizing the image data, and tracking the position of key points. In addition, for some more complex expressions, specially trained classifiers such as blinking and sticking out the tongue are used to improve the richness of the expressions. Finally, the entire tracking result is filtered to generate and extract facial expression unit parameters.

Facial expression collection includes face detection, key point localization and tracking, and solving the motion parameters of expression primitives. The key point tracking adopts an image denoising method based on median filtering, which significantly reduces computational complexity while ensuring tracking accuracy. Kalman filtering is used for facial

key points to filter and process the key point information generated in each frame. By calculating the key point information, the parameters of facial expressions are obtained. The specific process is shown in Fig. 1.

In OpenGL ES computing shaders, it is necessary to fully tap into the hardware resources of the GPU to create a sufficient number of threads, ensuring that all processors on the GPU are in the operational state of data processing, that is, to partition tasks as finely as possible. Fine partitioning of GPUs can improve their computational efficiency and effectively suppress latency. In the OpenGL ES rendering program, different tasks have different requirements for the size of the thread workgroup. To maximize the bandwidth usage of memory access, the size of the thread team can be set to be a full multiple of the block size to ensure that each thread can continuously access memory and avoid access conflicts.

This study proposes an image denoising method based on median filtering. This method is a non-linear filtering technique based on statistics, which replaces noise with the median in the median filtering window. The median filtering window traverses the entire image and calculates the median of all values within the filtering window as new pixels. The median calculation formula of the median filtering algorithm is Eq. (1).

$$g(x,y)=median\left\{f(x-i,y-j),i,j\in H\times W\right\} \quad (1)$$

In Eq. (1), $f(x,y)$ and $g(x,y)$ are the substitution values of the original image and the output image, respectively. $H\times W$ is the size of the filtering window (usually $H=W$ and odd, such as 3×3, 5×5, 7×7, etc.). Due to the fact that median filtering belongs to nonlinear filtering, its mathematical analysis becomes more complex when dealing with images containing random noise. For a normally distributed image with zero mean filtering noise, the noise variance of the median filtering can be approximately expressed as Eq. (2).

$$\sigma_{med}^2=\frac{1}{4nf^2(\bar{n})}\approx\frac{\sigma_i^2}{n+\frac{\pi}{2}-1}*\frac{\pi}{2} \quad (2)$$

In Eq. (2), $\sigma_i^2$ represents the size of the incoming noise, $n$ represents the median, and $f(\bar{n})$ represents the noise density function. The variance of mean filtering is expressed as Eq. (3).

$$\sigma_0^2=\frac{1}{n}\sigma_i^2 \quad (3)$$

The performance of median filtering depends on one factor, which is the size of the filtering window. At the same time, due to the different data types of images, a single algorithm cannot solve the corresponding problem. The Image Histogram (IH) represents the image distribution by quantifying the pixel numbers in each brightness value. For instance, in grayscale images, the IH algorithm calculates the quantity of pixel values (from 0 to 255) in the image and produces a histogram array with 256 elements. This study uses IH equalization to change the original IH by dispersing pixels that were previously existed in specific pixel values. For the original histogram $H(i)$, the calculation of the equilibrium distribution $H'(i)$ is Eq. (4).

$$H'(i)=\sum_{0\le j<i}H(i) \quad (4)$$

Finally, the final output image is calculated using Eq. (4), as exhibited in Eq. (5).

$$equalized(x,y)=H'(src(x,y)) \quad (5)$$

To solve the representation problem of 3D face acquisition, this study introduces the grid sampling operator and defines the downsampling and upsampling of grid features. Fig. 2 shows the grid sampling operation. Vertices shrink during downsampling operations. The downsampled vertices are a subset of the original mesh vertices. Each weight represents whether the $Q$-th vertex is retained during downsampling. Due to the infeasibility of lossless downsampling and upsampling for general grid surfaces, the upsampling matrix is constructed during the downsampling period. The vertices retained during downsampling are convolved, while these vertices are retained during upsampling. During downsampling, discarded vertices are mapped onto the downsampled mesh surface using centroid coordinates.

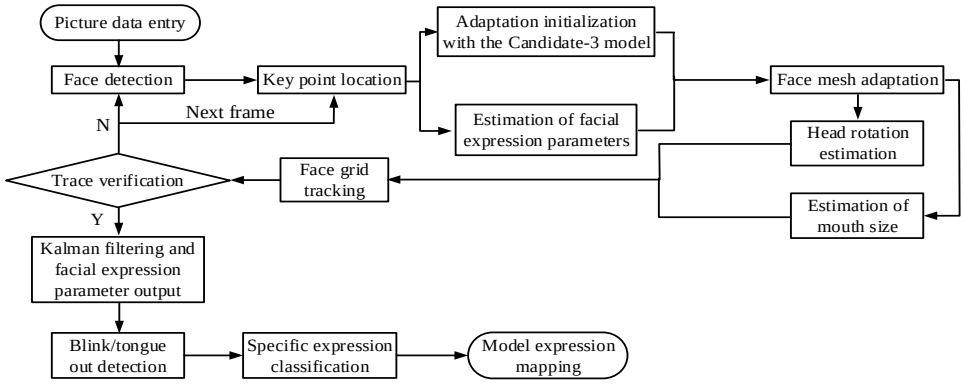



Fig. 1. Flow chart of facial expression capture.

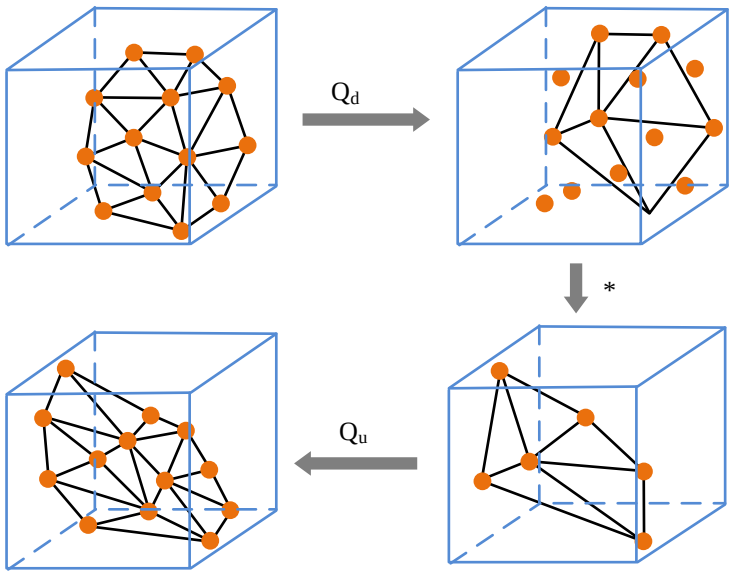



Fig. 2. 3D grid sampling operation.

In OpenGL ES computational shaders, data type optimization is an important optimization. This study aims to use low-precision data types as much as possible based on the actual needs of calculations. When precision is not high, use single-precision floating-point instead of double-precision floating-point to avoid using type conversion operations in shaders. Type conversion requires additional computational resources, especially for low-end devices. The advantage of integral filtering is that its computational complexity is independent of the filtering radius, so it can quickly process large-sized filtering templates and achieve filtering effects of different radii through multiple integral filtering. The localization and tracking of facial key points adopt a Supervised Descent Method (SDM)-based facial key point localization and tracking algorithm. Image information is extracted based on the current facial shape and position, and the shape is updated, as shown in Fig. 3.

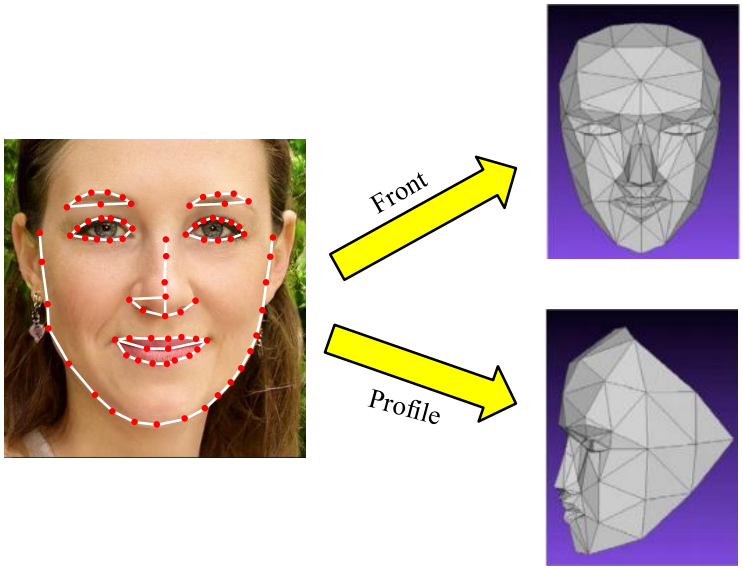



Fig. 3. Face key points.

The algorithm of integral filtering is simple to implement, fast in calculation speed, and widely used in image processing fields, such as face recognition, object detection, etc. $I(x,y)$ is the integral image, which is the sum of the original $i(x,y)$ on the left and top of point $(x,y)$. The calculation formula is Eq. (6).

$$I(x,y)=\sum_{x'}^{x}\sum_{y'}^{y} i(x',y') \quad (x' \le x, y' \le y) \qquad (6)$$

Once the integral image is calculated, the pixel sum in any rectangular area composed of the upper-left point $(x1, y1)$ and the lower-right point $(x2, y2)$ can be found with the time

complexity of $O(1)$. Calculating the rectangular range in the left image requires eight summation operations, and the computational complexity increases with the expansion of the matrix area. The process is given in Eq. (7).

$$S(x_1, y_1, x_2, y_2) = \sum_{y=y_1}^{y_2} \sum_{x=x_1}^{x_2} i(x, y) \quad (7)$$

In facial detection, features are decomposed into integral filter algorithms based on row and column directions. The algorithm for row direction is an exclusive scanning operation, as shown in Eq. (8).

$$\lfloor a_0, a_1, \ldots, a_{n-1} \rfloor \to \left\lfloor 0, a_0, (a_0 + a_1), \ldots, \sum_{i=0}^{n-2} a_i \right\rfloor \quad (8)$$

### B. Design and Implementation of Animation Synthesis and Rendering Module

3D facial expression animation is mainly used to model models. Model modeling requires determining what kind of model to create and some basic facial expressions of the model. In the process of modeling and fusion, it is necessary to receive the basic expression parameter sizes transmitted by facial capture for linear fusion, to render the requirements of different models and enrich the expression of facial expressions [16]. Expression fusion $P_{expr}$ is Eq. (9).

$$P_{expr} = p_{neur} + \sum_i \beta_i \times \Delta p_i \quad (9)$$

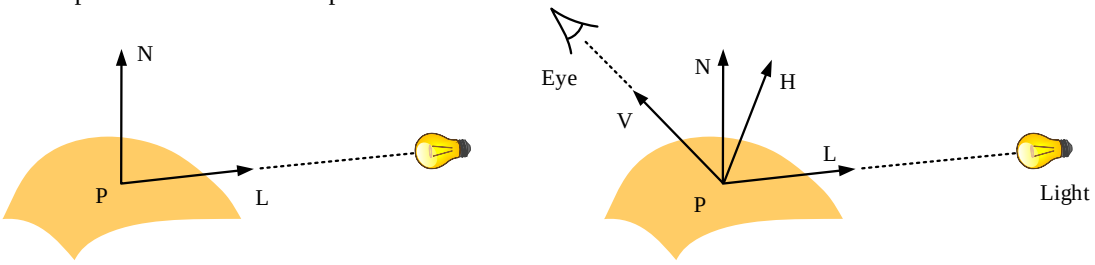
In Eq. (8), $\beta_i$ represents the expression parameter weight passed by the face capture module. $p_{neur}$ represents the vertex coordinates of the 3D model in its natural state, and $p_i$ represents the distance between the basic expression vertex of the 3D model and the natural expression vertex. By performing this linear calculation and overlaying all expression parameter weights, the final 3D model expression vertex position can be obtained, without the need for special processing in texture mapping. Based on considerations of computational complexity and effectiveness, this study chooses the Phong lighting model for calculation, as shown in Eq. (10).

$$light\ color = emissive + ambient + diffuse + specular \quad (10)$$

This model defines light as composed of self-illumination, diffuse reflection, ambient light, and mirror light, as shown in Figure 4. Self-luminous $emissive$ is used to simulate the light source information emitted by objects such as the sun. $ambient$ often simulates ambient light and low light on a global scale. In practical applications, diffuse reflection $diffuse$ is the most important, combined with the calculation of the incident angle of light rays, which has the strongest performance on the direction of light rays. The calculation method is Eq. (11).

$$diffuse = light\ color \times \max(N \cdot L, 0) \quad (11)$$

In Eq. (11), $N$ represents the normal and $L$ represents the inverse of the incident vector. $P$ represents the current vertex, while $N \cdot L$ is mainly used to determine whether the points on the face are facing or facing away from the light source.




(a) Diffuse reflection (b) Specular light

Fig. 4. Schematic diagram of diffuse and specular light.

Mirror light mainly simulates the reflection effect, calculated as Eq. (12).

$$specular = light\ color \times facing \times \left(\max(N \cdot H, 0)\right)^{shinness} \quad (12)$$

In Eq. (12), $H$ is the centerline of the line of sight reversal $V$ and L. $shininess$ is the light intensity parameter. $facing$ represents a Boolean value. If $N \cdot L$ is greater than 0 and takes a value of 1, it indicates that the light source vector $L$ is facing the surface. The final color value can be expressed as Eq. (13).

$$output_{color} = texture_{color} \times (material_k \times light_{color}) \quad (13)$$

In Eq. (13), $material_k$ can simulate different materials by adjusting the coefficient. Finally, the color output is obtained through Eq. (14).

$$light_{color} = K_e + K_a \times ambient + K_d \times diffuse + K_s \times specular \quad (14)$$

In Eq. (14), $K_e$, $K_a$, $K_d$, and $K_s$ represent vec4. If it is a model with fur, it needs to enable blending before rendering the transparency map. The mixing in OpenGL is achieved through Eq. (15).

$$C_{result} = C_s \times F_s + C_d \times F_d \quad (15)$$

In Eq. (15), $C_s$ is the color vector of the texture. $C_d$

represents the color vector currently stored in the color buffer. $F_s$ and $F_d$ are the source and target factor values, specifying the impact of alpha values on the $C_s$ color and target color.

This study also utilizes the MG2 algorithm in OpenCTM software to store and compress 3D model files. OpenCTM files are a simple format that can be well applied to modern 3D image rendering pipeline programs like OpenGL. In a 3D model, the minimum required information is the vertex coordinates of the model and the patch composed of coordinate points. OpenCTM simply includes these two prerequisites and also supports adding more options. An OpenCTM file only contains one Mesh, which is mainly divided into two parts: vertex information and patch information. Among all Vertex, Normal, UV, and Attrib, Vertex is mandatory. Table I shows the subscript $(0, 1, 2, \ldots, N)$ of the vertex to ensure the correspondence of the data in secret.

TABLE I. MODEL DATA STORAGE ARRANGEMENT

| **Index** | 0 | 1 | 2 | 3 | 4 | … | N |
|---|---|---|---|---|---|---|---|
| **Vertex** | v0 | v1 | v2 | v3 | v4 | … | v0 |
| **Normal** | n0 | n1 | N2 | n3 | n4 | … | nN |
| **UVCoord1** | t10 | t11 | t12 | t13 | t14 | … | t1N |
| **UVCoord2** | t20 | t21 | t22 | t23 | t24 | … | t2N |
| **Attrib1** | a10 | a11 | a12 | a13 | a14 | … | a1N |
| **Attrib2** | a20 | a21 | a22 | a23 | a24 | … | a2N |

## IV. ANALYSIS OF REAL-TIME FEAS

This study compares various algorithms on a scale of 1024×1024 and selects several excellent algorithms in recent years for comparison. Except for the code not implemented in the OpenCV library, performance comparisons are made with the corresponding algorithms in the OpenCV library.

### A. Analysis of Facial Capture Performance

This study uses Xcode to compile static libraries, and then calls mobile platform camera data through the Object c interface to transfer image data to the C interface to test the performance of the proposed face capture method. This study performs timestamp operations on each specific module, and after running for 10 minutes, the statistical results are obtained as shown in Table II. In Table II, the entire process achieves ultra real-time processing speed on both iPhone 12 and Galaxy S8, requiring less than 50 MB of memory.

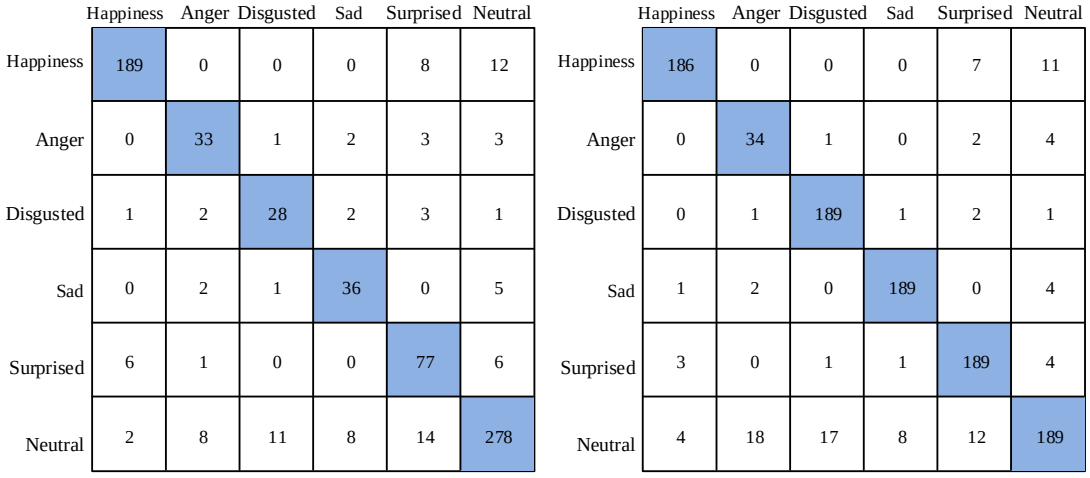
TABLE II. FACE CAPTURE PERFORMANCE STATISTICS

| **Face capture module** | **IPhone12(iOS 9.5.3)** | | | **Galaxy S8 (Android 8.0.1)** | | |
|---|---|---|---|---|---|---|
| | ***GPU (%)*** | ***Memory (MB)*** | ***Time (ms)*** | ***GPU (%)*** | ***Memory (MB)*** | ***Time (ms)*** |
| **Camera recording** | 12.2 | 48.6 | / | 13.6 | 51.3 | / |
| **Face detection and key location tracking** | 5.4 | 2.1 | 4.9 | 6.6 | 2.6 | 6.5 |
| **3D mesh adaptor** | 1.3 | 1.7 | 0.4 | 1.5 | 1.6 | 1.3 |
| **Eye classifier** | 1.2 | 0.4 | 0.7 | 1.4 | 1.4 | 3.2 |
| **Tongue sticking out classifier** | 1.0 | 1.2 | 0.4 | 1.1 | 1.3 | 3.2 |
| **Expression computing output** | 2.2 | 1.8 | 0.4 | 2.9 | 2.7 | 3.0 |
| **All functions** | 23.3 | 55.8 | 6.8 | 27.1 | 60.9 | 17.2 |

To objectively and accurately detect the recognition performance under different expressions, this study used the data from the expression fusion library as the basis to achieve data balance, and used the basic parameters in the model to adjust the corresponding penalty coefficients. The larger the parameter, the longer it took to identify the error. Therefore, it was necessary to determine a higher mean for this type of parameter. This study adjusted the parameters to predict the expressions to be tested for classification using the trained classifier, and compared the recognition rate with the pre-imbalanced expression data classifier. Fig. 5 shows the confusion matrix of six facial expressions. After data balancing, the recognition accuracy on the test sample increased from 87.69% before data balancing to 91.26%.

This study also conducted sample detection experiments on traditional texture features, Active Appearance Model (AAM), and proposed fusion feature extraction methods, and compared their results. Table III shows the average recognition rate of six types of expressions using three feature extraction methods. In Table III, compared to the other two methods, the research method had a better recognition rate on facial expressions than traditional and AAM algorithms. The recognition accuracy of Happy, Neutral, and Surprised facial expressions was relatively high, but the recognition accuracy of Anger, Sad, and Distorted facial expressions was not as good as the above three types. There were two main reasons for this situation. One reason is that each person has different ways of expressing sadness and disgust, and there are also significant differences. Therefore, the results of identifying such expressions are relatively scattered and may be recognized as various types of expressions. When

recognizing anger, it is easy to identify it as a surprise because when extracting features from the eye area, both types of expressions are prone to eye enlargement, which can confuse recognition. The second is that the three types of expressions, Anger, Distorted, and Sad, have certain similarities in themselves, and people may find it difficult to distinguish these three types of expressions in real life. Therefore, the recognition accuracy of these three types of expressions is lower than that of other expressions, but experiments have shown that the average accuracy of expression recognition in the research method still has a significant improvement.




(a) Confusion matrix before data equalization (b) Confusion matrix after data equalization

Fig. 5. Confusion matrix of six facial expressions.

TABLE III. COMPARISON OF FACIAL EXPRESSION RECOGNITION RESULTS

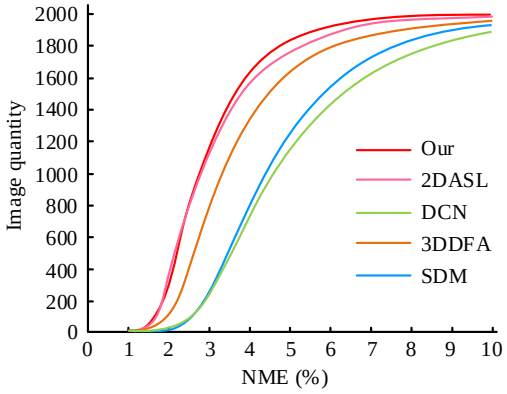
| Method | Traditional texture feature | AAM feature | Research proposed fusion features |
|---|---|---|---|
| **Happiness** | 72.13% | 83.62% | 92.43% |
| **Anger** | 75.32% | 81.75% | 91.65% |
| **Sad** | 76.32% | 80.69% | 93.47% |
| **Surprised** | 74.19% | 79.42% | 89.62% |
| **Disgusted** | 73.58% | 83.66% | 86.78% |
| **Neutral** | 71.96% | 84.51% | 90.33% |
| **Average** | 74.37% | 80.42% | 91.76% |

### *B. Analysis of Facial Animation Rendering Effects*

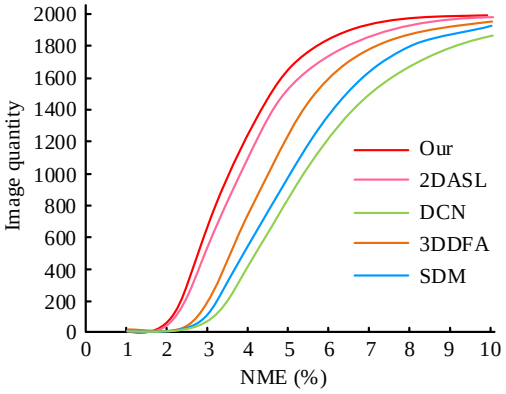
To comprehensively evaluate the facial rendering performance of research methods, multiple excellent algorithms in recent years were selected for comparison, including 2DASSL [17], DCN [18], 3DDFA [19], SDM [20] and other algorithms. The Normalized Mean Error (NME) was selected as the evaluation metric, which represents the average error value of normalized feature points. The normalized size corresponds to the square value of the product of the length and width of the face border to ensure that face alignment was not affected by external factors. Fig. 6 (a), (b), and (c) show the cumulative alignment errors of 68 2D face key points (2D-KP), all 2D face key points (A2D-KP), and all 3D face key points (A3D-KP) for face alignment. Among the 68 cumulative errors of 2D-KP, 2DASSL and research methods performed similarly, demonstrating excellent performance. 3DDFA is stable and efficient, and also performs well. The research method achieved optimal NME on 68 2D-KP, A2D-KP, and A3D-KP datasets.

This study evaluated the impact of loss functions on alignment using three commonly used loss functions: PDC, VDC, and WPDC, based on MATLAB. The performance of models trained with different loss functions on the dataset is Fig. 7. PDC could not fit the model well, and the convergence of errors was poor. It basically converged when the NME reached around 8%. VDC performed better than PDC, but was affected by pathological curvature during the convergence process, so fitting errors might occur during the fitting process, resulting in poor visualization results. Compared to other methods, WPDC had better error performance and could quickly reduce NME. However, overall, the loss function of the research method combined key point loss, resulting in better performance and the fastest reduction of NME. The final error level was close to 4%, and the accuracy was significantly higher.

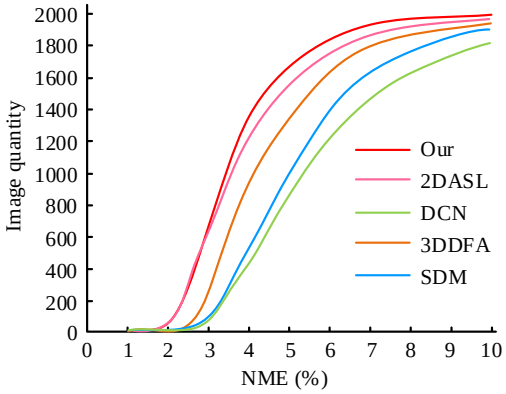
The proposed facial animation rendering method consumes a time proportional to the image size, as shown in Fig. 8. In a size of 200×300, the processing speed of each image was around 19.4 milliseconds, which was extremely fast. After testing, it could reach about nine milliseconds at a size of 120×120, but due to the small window, it was not suitable for real-time input of facial images. If face detection was performed by scaling images, sometimes the detection efficiency might be affected due to the small size of the face.




(a) NME for 2D key points




(b) NME for all 2D key points




(c) NME of all 3D key points

Fig. 6. Comparison of NME results.

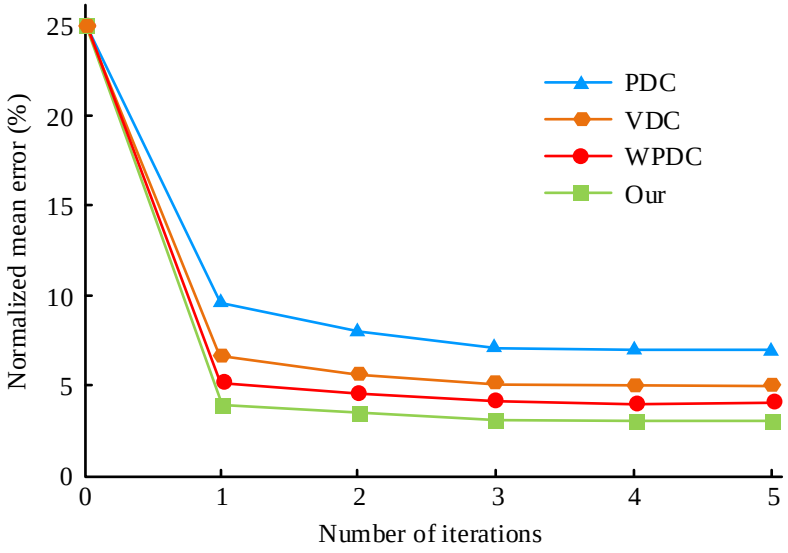



Fig. 7. Error function results in face alignment.

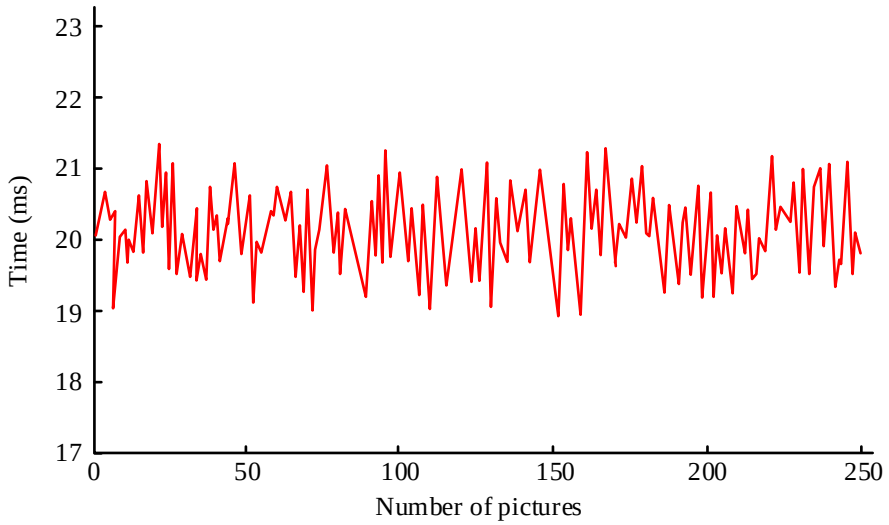



Fig. 8. Runtime of the image enhancement method on 200 images.

To better explore the impact of video frame preprocessing algorithms, speech features of speech driven networks, and lip animation results on expression and posture networks, this study divides each part and obtained six variants by considering one or more component parts. Method 1 is the benchmark method. Method 2 is to crop each frame separately. Method 3 is to add speech features from the speech network as inputs to the expression and posture network for feature fusion. Method 4 uses speech driven network prediction of lip animation as the basis. Method 5 adds the speech features of the speech network as input to the expression and posture network for feature fusion, and also uses the lip animation predicted by the speech driven network as the basis. Method 6 uses the proposed rendering method.

Table IV shows the objective evaluation results of various situations on the dataset. It can be found that the proposed rendering method can bring certain results in improving the expression posture. Among them, video frame preprocessing has the best effect on VER reduction, surpassing the combination of speech features and lip animation. In addition, it also has a good decrease in WER and VWER, surpassing individual speech features or lip animation.

TABLE IV. RESULTS OF ABLATION EXPERIMENT

| Serial number | Video frame preprocessing | Ocular features | Lip animation | CER | WER | VER | VWER |
|---|---|---|---|---|---|---|---|
| 1 | | | | 68.7 | 98.1 | 61.0 | 97.6 |
| 2 | √ | | | 67.5 | 96.2 | 59.1 | 94.4 |
| 3 | | √ | | 67.0 | 96.4 | 60.0 | 94.8 |
| 4 | | | √ | 67.8 | 97.0 | 60.3 | 95.7 |
| 5 | | √ | √ | 66.4 | 95.0 | 59.4 | 93.7 |
| 6 | | √ | √ | 65.0 | 92.7 | 57.5 | 91.0 |

## V. RESULTS AND DISCUSSION

The process of face detection and face feature point detection is conducted on the collected ordinary images with the objective of determining the face region. Subsequently, facial expression recognition and face reconstruction are carried out on the cropped face region aiming to obtain the 3D model data of the corresponding 2D face image. Finally, 3D animation can be realized by using the renderer. However, when face image 3D modeling is applied in mobile devices, higher requirements are put forward for computing time. In regard to graphical rendering, OpenGL employs hardware acceleration technology, thereby enabling direct access to the GPU to accelerate the graphical rendering process, thus markedly enhancing the efficiency of the rendering process [21]. Furthermore, OpenGL employs a dual-cache system to process the computing scene, generated image, and display image separately. This approach significantly enhances the computing power and display speed of the computer, thereby improving the user experience and greatly reducing processing time. The proposed method was validated on iPhone 12 and Galaxy S8, and it was found that the entire process achieved ultra real time processing speed with less than 50 MB of memory required. In addition, the processing speed of each image in the 200×300 size was about 19.4 milliseconds, which is extremely fast. Wang B et al. proposed a face feature extraction algorithm based on deep learning, which adopted C++ and OpenGL for rendering simulation, and has good performance in accuracy and speed [22]. The results are consistent with the results of this study, indicating that the mobile platform FEAS technology based on OpenGL can meet the real-time requirements.

Face images contain some very complex information and content. Typical facial features include the eyes, mouth,

eyebrows, and the face as a whole. Additionally, there is a great deal of information present within the face, composed of these parts [23]. Based on face detection, the key facial feature points are automatically located according to the input face image, such as the eyes, nose tip, corners of the mouth, eyebrows, and contour points of the face parts, etc. The input is the face appearance image and the output is the set of face feature points. Due to the combination of key point loss, the proposed method can rapidly reduce the normalized average error, and the final error level is close to 4%. The error performance on 68 2D key points, all 2D face key points, and all 3D face key points is optimal. 3D DFA and 2D ASL are messy in the 3D spatial error distribution. It indicates that there are some defects in the face estimation, which reduces the accuracy and visibility of model fitting. With the help of OpenGL, 3D model can be quickly mapped to 2D space and 3D effect can be displayed in 2D space. The 3D face data acquisition based on OpenGL is extremely efficient, which can ensure a high frame rate, achieve continuous animation effect, and effectively improve the adaptability of face reconstruction to the scene.

## VI. Conclusion

This study was based on the OpenGL platform and utilized the OpenGL ES computational shader to develop a GPU version of a high-performance image algorithm, providing a real-time facial animation processing solution for mobile devices. By combining a series of architecture and resource optimization strategies, the cross-platform application of Android and iOS systems on mobile devices has been implemented, and the stable performance of the entire solution on mobile platforms has been guaranteed. The results indicated that the research method achieved ultra real-time processing speed on two different operating systems and only required less than 50 MB of memory. After data balancing, the recognition accuracy increased from 87.69% before data balancing to 91.26%, and there was a significant improvement in the average accuracy of facial expression recognition. The loss function of the research method combined key point loss, resulting in better performance and the fastest reduction of NME. The final error level was close to 4%, and the accuracy was significantly higher. In a size of 200×300, the processing speed of each image was around 19.4 modes. This method has the characteristics of simple operation, powerful functionality, high realism, and strong real-time performance. However, this study only optimized high-performance image processing algorithms for a single architecture and efficient computation cannot be performed when the algorithms are ported to other platforms. In the future, high-performance facial animation simulation processing algorithms for other architecture processors can be optimized.

## References

[1] Usman A M, Abdullah M K. An assessment of building energy consumption characteristics using analytical energy and carbon footprint assessment Model. Green and Low-Carbon Economy, 2023, 1(1): 28-40.

[2] Aryavalli S N G, Kumar G H. Futuristic vigilance: Empowering chipko movement with cyber-savvy IoT to safeguard forests. Archives of Advanced Engineering Science, 2023, 1(8): 1-16.

[3] Xu L. Fast Modelling Algorithm for Realistic Three-Dimensional Human Face for Film and Television Animation. Complexity, 2021, 20(2):1-10.

[4] Liu K, Sun Y, Yang D. The administrative center or economic center: Which dominates the regional green development pattern. A case study of shandong peninsula urban agglomeration, china. Green and Low-Carbon Economy, 2023, 1(3), 110-120.

[5] Pham H X, Wang Y, Pavlovic V. Learning Continuous Facial Actions from Speech for Real-time Animation. IEEE Transactions on Affective Computing, 2020, 13(3):1567-1580.

[6] Bossér L, Bossér J D. Target echo calculations using the OpenGL graphics pipeline. Applied Acoustics, 2021, 181(9):108133.

[7] Calabuig B E, Davia-Aracil M, Mora-Mora H F. Computational model for hyper-realistic image generation using uniform shaders in 3D environments. Computers in Industry, 2020, 123(1):1-13.

[8] Yin J, Yang J H. Virtual Reconstruction Method of Regional 3D Image Based on Visual Transmission Effect. Complexity, 2021, 92(12):1778-1797.

[9] Rémi F, Maunoury M, Loseille A. On pixel-exact rendering for high-order mesh and solution. Journal of Computational Physics, 2020, 424(13):1098-1112.

[10] Pham H X, Wang Y, Pavlovic V. Learning Continuous Facial Actions from Speech for Real-time Animation. IEEE Transactions on Affective Computing, 2020, 13(3):1567-1580.

[11] Berson E, Catherine S, Stoiber N. Intuitive Facial Animation Editing Based on A Generative RNN Framework. Computer Graphics Forum, 2020, 39(8):241-251.

[12] Pumarola A, Agudo A, Martinez A M, Sanfeliu A, Moreno N F. GANimation: One-Shot Anatomically Consistent Facial Animation. International Journal of Computer Vision, 2020, 128(3):698-713.

[13] Paier W, Hilsmann A, Eisert P. Interactive facial animation with deep neural networks. IET Computer Vision, 2020, 14(6):359-369.

[14] Huang Z, Ren F, Hu M, Chen S. Facial Expression Imitation Method for Humanoid Robot Based on Smooth-Constraint Reversed Mechanical Model (SRMM). IEEE transactions on human-machine systems, 2020, 50(6):538-549.

[15] Xiao R, Zhang J, Chai P, Ju C, Lin W C, He R. Dual interpolation boundary face method for 3-D acoustic problems based on binary tree grids. Engineering analysis with boundary elements, 2023, 150(6):7-19.

[16] Cao H, Wu Y, Bao Y, Feng X, Wan S, Qian C. UTrans-Net: A model for short-term precipitation prediction. Artificial Intelligence and Applications. 2023, 1(2): 106-113.

[17] Feng Y, Feng H, Black M J, Bolkart T. Learning an animatable detailed 3D face model from in-the-wild images. ACM Transactions on Graphics, 2021, 40(4):1-13.

[18] Alboqami F, Van Oudenhoven V C O, Ahmed U, Zahid U, Emwas A, Sarathy S M, Jameel A G. A Methodology for Designing Octane Number of Fuels Using Genetic Algorithms and Artificial Neural Networks. Energy & Fuels, 2022, 36(7):3876-3880.

[19] Ly A, El-Sayegh Z. Tire wear and pollutants: An overview of research. Archives of Advanced Engineering Science, 2023, 1(1): 2-10.

[20] Garai S, Paul R K, Kumar M. Intra-annual national statistical accounts based on machine learning algorithm. Journal of Data Science and Intelligent Systems, 2023, 2(2): 12-15.

[21] Kwolek B, Rymut B. Reconstruction of 3D human motion in real-time using particle swarm optimization with GPU-accelerated fitness function. Journal of Real-Time Image Processing, 2020, 17(4): 821-838.

[22] Wang B, Shi Y. Expression dynamic capture and 3D animation generation method based on deep learning. Neural Computing and Applications, 2023, 35(12): 8797-8808.

[23] Yali L, Huijie Z. Three-dimensional campus 360-degree video encoding VR technology based on OpenGL. Multimedia Tools and Applications, 2020, 79(7): 5099-5107.