Mesh Deformation Algorithms for Cosmetic Surgery Simulation

Mengyuan Zhu[1]

[1] Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill NC 27514, USA
gisellez@ad.unc.edu

Abstract. In the rapidly advancing field of medical simulations, cosmetic surgery simulations have become crucial tools for surgical planning, training, and enhancing patient communication. These simulations allow surgeons to visualize and predict post-operative outcomes, which aids in making informed decisions and strengthens the surgeon-patient relationship. Despite advancements in techniques like bone tissue cutting and prosthesis transplantation, challenges remain in accurately depicting postoperative appearances, largely due to the limitations of the underlying geometric models used in these simulations. The advancement of sophisticated computer-assisted technologies has significantly driven the evolution of mesh deformation algorithms. These algorithms support the iterative refinement of geometric designs through a process called mesh deformation. Typically, this involves maintaining a constant mesh topology while updating node locations to align with geometric changes, a requirement that can become restrictive when significant modifications are necessary. This review paper examines two advanced mesh deformation algorithms: Laplacian-based mesh deformation and grayscale texture-based mesh deformation, each serving distinct roles in the field of cosmetic surgery simulation. The aim is to juxtapose these techniques, analyzing their effectiveness in achieving detailed and realistic surgical simulations.

Keywords: Mesh Deformation, Differential Domain Methods, Volumetric Laplacian, Grayscale Texture.

1 Introduction

In the rapidly advancing field of medical simulations, cosmetic surgery simulations have become crucial tools for surgical planning, training, and enhancing patient communication. These simulations allow surgeons to visualize and predict post-operative outcomes, which aids in making informed decisions and strengthens the surgeon-patient relationship. Despite advancements in techniques like bone tissue cutting and prosthesis transplantation, challenges remain in accurately depicting postoperative appearances,

Y. Wang (ed.), *Proceedings of the 2024 International Conference on Artificial Intelligence and Communication (ICAIC 2024)*, Advances in Intelligent Systems Research 185,
https://doi.org/10.2991/978-94-6463-512-6_33

largely due to the limitations of the underlying geometric models used in these simulations. The advancement of sophisticated computer-assisted technologies has significantly driven the evolution of mesh deformation algorithms. These algorithms support the iterative refinement of geometric designs through a process called mesh deformation. Typically, this involves maintaining a constant mesh topology while updating node locations to align with geometric changes, a requirement that can become restrictive when significant modifications are necessary. This review paper examines two advanced mesh deformation algorithms: Laplacian-based mesh deformation and grayscale texture-based mesh deformation, each serving distinct roles in the field of cosmetic surgery simulation. The aim is to juxtapose these techniques, analyzing their effectiveness in achieving detailed and realistic surgical simulations.

Laplacian-based mesh deformation has traditionally been used on surface meshes, but has recently been extended to volumetric meshes, such as tetrahedral meshes [1, 2]. This method calculates Laplacian coordinates that capture the local curvature of the mesh, facilitating detailed edits. While effective for small rotations, its initial non-rotation invariant nature posed challenges for larger transformations [1], leading to advancements that adapt these methods to better handle the comprehensive three-dimensional requirements of volumetric meshes. Grayscale texture-based mesh deformation represents a more recent innovation that simplifies the deformation process significantly. By encoding deformation data directly onto a grayscale image, this method eliminates the need for complex preprocessing steps, allowing for more flexible and immediate adjustments to the mesh [3-5]. This approach is particularly valuable in dynamic surgical simulation environments where speed and adaptability are crucial.

2 Mesh Deformation Based on Laplacian Coordinates

2.1 Volumetric Mesh with Differential Domain Methods

Since 2004, Differential Domain Methods (DDMs) have significantly advanced the optimization of geometric model edits. These methods analyze the differential properties of deformation functions and optimize results under constraints. Among these, Lipman et al. [6] and Sorkine et al. [7] introduced the Laplacian editing method as a type of DDMs. The method's core concept, the Laplacian coordinate of a vertex, represents the difference between a vertex's coordinates and the weighted average of its adjacent vertices. This acts as a linear approximation of the Laplace operator on a smooth surface, indicating the direction and magnitude relative to average curvature.

However, Laplacian coordinates are not rotation invariant, requiring local transformations for precise modifications. This involves using the Laplace equation to re-establish the three-dimensional vertex coordinates from the modified Laplacian coordinates, preserving mesh detail during deformation. Both Lipman et al. [6] and

Sorkine et al. [7] initially applied these methods to surface meshes, which are limited to the exterior dimensions and can distort significantly under large rotations. These methods are typically effective for rotations under 90 degrees. To overcome the limitations of non-rotation invariance in Laplacian coordinates, Lipman et al. [8] introduced a linear rotation-invariant coordinate system leveraging the first and second fundamental forms of discrete differential geometry. This enhancement transforms the Laplacian coordinates from the original world coordinate system to a local frame, involving the solution of two linear systems to preserve detailed accuracy in deformed meshes.

In cosmetic surgery simulations, where precision is crucial, volumetric meshes, particularly tetrahedral meshes, are preferred due to their detailed volumetric accuracy. Unlike surface meshes, which only depict exteriors, tetrahedral meshes consist of vertices, edges, faces, and tetrahedrons that fill the entire structure, providing a robust 3D representation. This comprehensive approach helps maintain structural integrity during deformations, making volumetric meshes ideal for tackling challenges related to distortion and volume preservation, thus ensuring more accurate and realistic outcomes.

2.2 Tetrahedral Mesh and Laplacian Coordinates

A tetrahedral mesh specifically refers to a type of volumetric mesh where the volume is divided into tetrahedra – a 3D simplex consisting of four vertices, six edges, and four triangular faces – each tetrahedron in the mesh fits together with others to fill a three-dimensional space [9]. The tetrahedral mesh is represented as a pair $M = (P, K)$, where P is a set of N points in three-dimensional Euclidean space, with each point corresponding to a vertex coordinate. The set K is an abstract simplicial complex that includes all adjacency information of the mesh and consists of subsets for vertices V, edges E, faces F, and tetrahedrons T. Similar to surface meshes, the Laplacian coordinates on tetrahedral mesh are computed for each vertex. For a vertex p_i, its Laplacian coordinate δ_i is defined as:

$$\delta_i = \Delta_V(p_i) = w_i \sum_{v_j \in N_1(v_i)} w_{ij} \left(f(v_j) - f(v_i) \right) \quad \#(1)$$

Here, w_{ij} is a variable weight for the edge between vertices v_i and v_j, and w_i is the normalization weight of the central vertex v_i, which relates to the volume of the dual Voronoi cell associated with v_i. The Laplacian operator can then be expressed in matrix form:

$$\left(\delta^{(x)}, \delta^{(y)}, \delta^{(z)}\right) = L\left(p^{(x)}, p^{(y)}, p^{(z)}\right) = M^{-1} L_s \left(p^{(x)}, p^{(y)}, p^{(z)}\right) \#(2)$$

Where L is the matrix representation of the Laplacian operator, constructed from the diagonal matrix M of volume weights and the symmetric sparse matrix L_s that contains the weights of the edges of the mesh. The terms $\left(p^{(x)}, p^{(y)}, p^{(z)}\right)$ are the X, Y, and Z

coordinates of the original mesh vertex coordinates and $\left(\delta^{(x)}, \delta^{(y)}, \delta^{(z)}\right)$ represent the Laplacian coordinates of the original mesh vertex coordinates. These special coordinates capture how a vertex is positioned relative to its neighbors.

The matrix M is diagonal, which means that only the entries going from the top left to the bottom right have values; the rest are zero. Each value on the diagonal M_{ij} corresponds to the volume weight of a vertex v_i, which is $1/w_i$. Here, w_i is determined by the volume of space around that vertex, given by the Voronoi cell volume.

L_s, it's a matrix that describes the connection weights between each vertex and its neighbors. It is defined such that if the row and column correspond to the same vertex, i.e., $i = j$, the entry is the negative sum of the weights W_{ik} for all neighboring vertices v_k. Conversely, If the row and column correspond to a pair of neighboring vertices, i.e., v_j is a neighbor of v_i, the entry is w_{ij}, the weight of the edge between them. Entries for vertex pairs that are not neighbors are set to zero. The matrix L_s is then given by the formula:

$$(L_s)_y = \begin{cases} -\sum_{v_k \in N_i(v_i)} W_{ik}, & if\ i = j \\ w_{ij}, & if v_j \in N_i(v_i) \\ 0, & otherwise \end{cases} \#(3)$$

In this equation, $N_i(v_i)$ denotes the set of vertices that are direct neighbors of vertex v_i. The sparse matrix L_s captures the edge weights and is used to build the full Laplacian operator matrix L, which is expressed by multiplying the inverse of M with L_s:

$$L = M^{-1}L_s \#(4)$$

While there isn't a direct equivalent of the mean curvature normal [10] for a tetrahedral mesh, the Laplacian coordinates effectively represent a vertex's relative position vector against its immediate neighbors. This provides a differential attribute that helps establish global relationships between vertex coordinates.

2.3 Mesh Deformation Based on Laplacian Coordinates

Tetrahedral mesh deformation using Laplacian coordinates retains translational invariance but not rotational or scaling invariance. During deformation, the original Laplacian coordinates δ_i are transformed to accommodate the modifications, resulting in the modified coordinates δ_i'. The coordinates of the deformed mesh p' can then be reconstructed by solving the following optimization problem – an objective function:

$$p' = \underset{p'}{\operatorname{argmin}} \sum_i V_i \parallel \Delta_V\left(p'_i\right) - \delta_i'^{\parallel 2} \#(5)$$

Here, the aim is to find the new vertex positions that minimize the squared differences between the modified Laplacian coordinates and the original ones: find the best new

positions for vertices p' such that, when the Laplacian operator is applied to these positions, the resulting difference from the original mesh is minimized [11]. V_i is a volume weight for each vertex related to its Voronoi cell in the mesh, and Δ_V is the Laplacian operator applied to the new positions. δ_i' are the target Laplacian coordinates after deformation. The solution to the least squares problem is thus can be computed by solving the normal equation [10]:

$$L^T M L p' = L^T M \delta' \#(6)$$

It's essentially a set of equations derived from the objective function that are solved to obtain new vertex positions p'. The matrix M has volume weights, and L is the Laplacian matrix, which relates vertex positions to each other. Substituted L with $M^{-1}L_s$ from equation (4) to make it clearer how L is constructed from M and L_s, this yields:

$$L_s M^{-1} L_s p' = L_s \delta' \#(7)$$

After canceling out M on both sides, this presents another way to express the problem being solved—adjusting the mesh while keeping its new shape as close as possible to the target shape defined by δ'. Multiply both sides by M^{-1} to get rid of M yields the bi-Laplacian operator equation:

$$L^2 p' = L\delta' \#(8)$$

Alternatively, the modified Laplacian coordinates δ' can be directly substituted into equation (2) to get the single Laplacian operator equation:

$$Lp' = \delta' \#(9)$$

Since the weight w_i for each vertex v_i is defined as the reciprocal of the volume of the Voronoi cell surrounding that vertex, $w_i = \frac{1}{(V_{dual})}$. The single Laplacian equation after normalization becomes:

$$L_s p' = \delta' \#(10)$$

Where L_s is now a normalized sparse matrix that incorporates the weights w_i for each vertex. Essentially, L_s has been scaled so that each row is weighted by the corresponding Voronoi cell volume. The bi-Laplacian equation after normalization becomes:

$$L^T L p' = L^T \delta' \#(11)$$

Where L is the Laplacian matrix with normalization, and L^T is its transpose. The normalization here means that both L and L^T reflect the weighting by Voronoi cell volumes. The bi-Laplacian, represented as $L^T L$, is often favored over the single Laplacian in mesh deformation tasks. This preference stems from its property of creating a positive-

semidefinite matrix. A positive-semidefinite matrix is advantageous because, when it is also of full rank, it ensures that there is a unique solution to the optimization problem, minimizing the potential for ambiguous results. Moreover, the multiplication of the Laplacian matrix by its transpose $L^T L$, isn't the same as applying the Laplacian operator twice in sequence. Instead, this product considers both the immediate and extended neighborhood around each vertex. Consequently, the resultant deformation benefits from a smoothing effect that gracefully balances local vertex adjustments with broader, more global mesh considerations.

2.4 Application

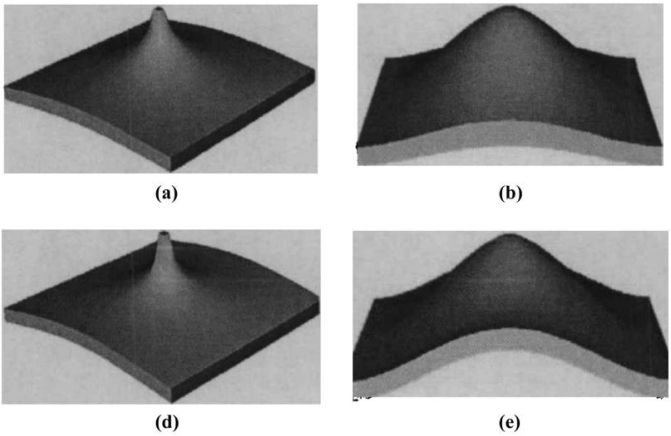

Fig. 1. Comparison of Mesh Manipulation on a Simplified Muscle Model [12]

As shown in Fig. 1, Research [12] has compared a simplified representation of a thin layer of muscle model, where the boundaries on both sides are fixed. A small manipulative region in the center is selected for upward translation operations, comparing (a) single Laplacian surface mesh, (b) double Laplacian surface mesh, (d) single Laplacian volumetric mesh, and (e) double Laplacian volumetric mesh. The results show:

Surface meshes experienced severe distortion, with the bottom boundary surfaces essentially remaining undeformed, while the volumetric meshes effectively preserved volume characteristics, and the deformation of the bottom boundary surfaces was very realistic. This is because some common operational methods require the use of volumetric meshes with internal grid structures and connections to properly diffuse local transformations and exert stronger deformation influences on distant areas. Additionally, since the translations themselves did not account for local rotational transformations, the deformations using a single Laplacian operator were not effective, whereas the double Laplacian operator still effectively maintained a smooth transition between the

constrained boundary areas and the free deformation areas, resulting in correct deformation outcomes.

3 Mesh Deformation Based on Grayscale Texture

Among various methods, differential domain mesh deformation algorithms stand out due to their ability to preserve the geometric details of meshes. However, these algorithms typically require the mesh model to undergo preprocessing, where the mesh is stored data structures, and differential domain coordinates of model vertices are calculated. Such preprocessing can be complex and time-consuming.

Addressing the need for simplification and efficiency, a novel approach has been proposed which utilizes grayscale texture for mesh deformation [3-5]. This approach decouples deformation data from geometric data, using grayscale textures to manage deformation, simplifying the deformation process by eliminating extensive preprocessing and allows for precise, direct manipulation of the mesh. Such advantages make grayscale texture-based deformation particularly suitable for the dynamic and detailed requirements of cosmetic surgery simulations.

3.1 Calculation of Texture Coordinates for Grayscale Images

The first step in mesh deformation operation is to construct a gray-scale texture that stores the deformation information. The steps are as follows:

1. Target Area Drawing: Choose an appropriate selection tool to draw the target area on the screen for the deformation operation, as shown in Fig. 2 a).
2. Grayscale Image Creation: Use the OpenCVSharp package to create a grayscale image that matches the size of the software window. Then, apply the flood-fill method to fill in the target area, resulting in the effect shown in Fig. 2 b).
3. Grayscale Texture Blurring: Blur the grayscale texture to ensure a smooth transition between the target and non-target deformation areas, as shown in Fig. 2 c).

a) b) c)

Fig. 2. Gray-scale Texture Generation [4]

The third step of blurring the texture uses common algorithms such as average blur, Gaussian blur, and median blur. The main difference between them lies in the convolution

kernel used for processing. The experiments conducted with these algorithms on the grayscale texture are shown in Table 1, with a convolution kernel size of 13×13.

Table 1. Experimental Results of Different Blur Algorithms [4]

Blur Algorithms	0 Iterations/n	20 Iterations/n	40 Iterations/n
Mean Blur			
Gaussian Blur			
Median Blur			

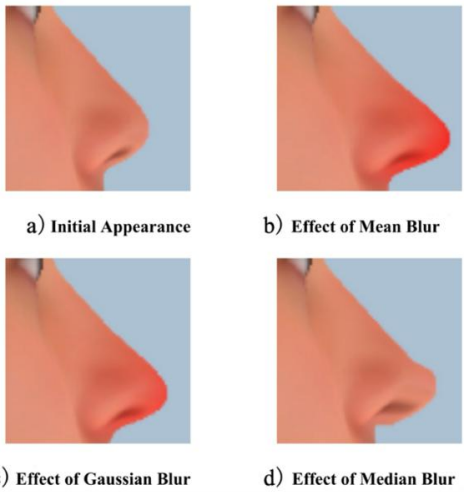

Fig. 3. Influence of Blur Algorithm on the Mesh Deformation [4]

A convolution kernel is a grid of weighted cells used to calculate the sum of products between each kernel element and the corresponding image pixel it covers. The result is the new pixel value at that position. For an $n \times n$ average blur kernel, the value of each element in the kernel is $\frac{1}{n \times n}$. The Gaussian blur convolution kernel is based on the Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \#(12)$$

Where σ represents the standard deviation, and x and y correspond to the distances from the current position to the center in integer distances. Fig. 3 illustrates the effects of different blur algorithms in practice, revealing that the median blur effect is the least smooth at the transition between target and non-target areas post-deformation. Both average and Gaussian blurs resolve this issue effectively. However, when controlling for a certain deformation range, the Gaussian blur did not achieve the expected amplitude at the most deformed vertex, while the average blur results were closer to the expected outcome. Considering all factors, the decision was made to use the average blur algorithm [4,5].

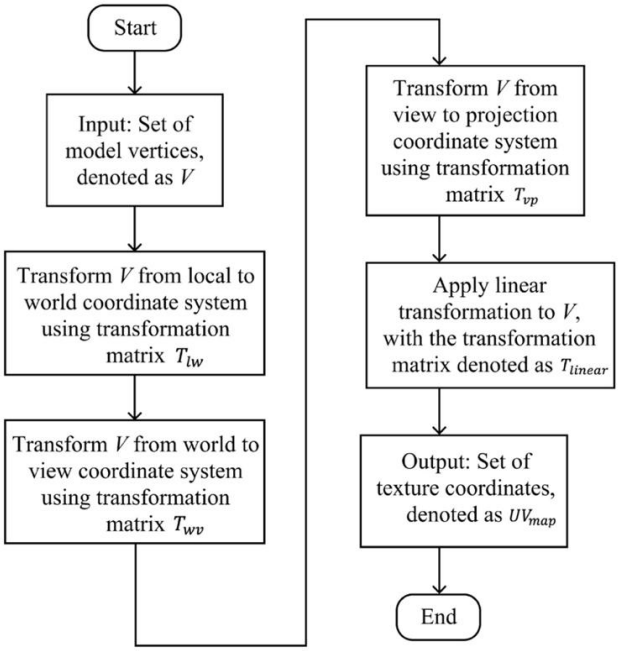

Fig. 4. Flow Chart of Texture Coordinate Calculation
(Picture credit: Original)

After generating the grayscale texture, it must be mapped to the mesh vertices by computing texture coordinates, denoted as UV_{map}, for the mesh model vertices. The method used involves line-of-sight-based linear interpolation to compute texture coordinates, transforming the model's three-dimensional coordinates from local to screen space. This normalizes the vertex coordinates, with the computational workflow outlined in Fig. 4.

Consider T as the aggregate transformation that is the product of all individual transformation, thus:

$$T = T_{linear}T_{vp}T_{wv}T_{lw} \#(13)$$

This composite transformation is mathematically represented as:

$$UV_{map} = T \cdot V \#(14)$$

3.2 Mesh Deformation Based on Grayscale Texture

To ensure a clear visual differentiation between the areas of a 3D model slated for deformation and those that are not—the target and non-target regions—the use of shader technology is instrumental [4,5]. By projecting a pre-constructed grayscale texture onto the model, with the lips selected by a shape-selection tool, shaders facilitate an intuitive presentation of the regions undergoing deformation (see Fig. 5).

Fig. 5. The Shape-Selection Tools, Gray-scale Texture, and Projection Rendering [4]

For the effective display of deformation regions, one might use a grayscale texture map G included in the shaders, the model texture coordinates UV_{map}, and the source color C_{source}. Consider a point on the model, represented by the coordinates (u_i, v_i), which corresponds to the pixel P_i on the grayscale texture map G. The gray-scale intensity I_i as this point can be determined through:

$$I_i = P_i \cdot V_{lum} \#(15)$$

Here, V_{lum} is a vector, commonly $(0.299, 0.587, 0.114)$, which represents the perceived luminosity of each color channel to the human eye, a standard in grayscale conversion [13]. The on-screen visual output, C_{render}, combines the diffuse color $C_{diffuse}$ from the graphics pipeline with the source color C_{source} modulated by the computed intensity:

$$C_{render} = C_{diffuse} \cdot (1 - I_i) + C_{source}I_i \#(16)$$

This equation illustrates a linear interpolation between the diffuse color and the source color, influenced by the intensity I_i. It creates a visual blending where the influence of the base color increases with the intensity of the grayscale texture, achieving a realistic rendering of the target deformation region.

The essence of this mesh deformation algorithm lies in utilizing the gray-scale intensity I from grayscale texture map G, mapped via UV_{map}, as weights for vertex transformation. Thus, the new vertex set V' is derived by adding the weighted displacement to the original coordinates, as formulated:

$$V'(D) = V(D) + W \cdot D \ \#(17)$$

This equation articulates how each point in the designated target area, originally at $V(D)$, shifts to a new position $V'(D)$, propelled by the transformation weight W, embodied in the deformation vector D. Fig. 6 below visualizes this transformation process:

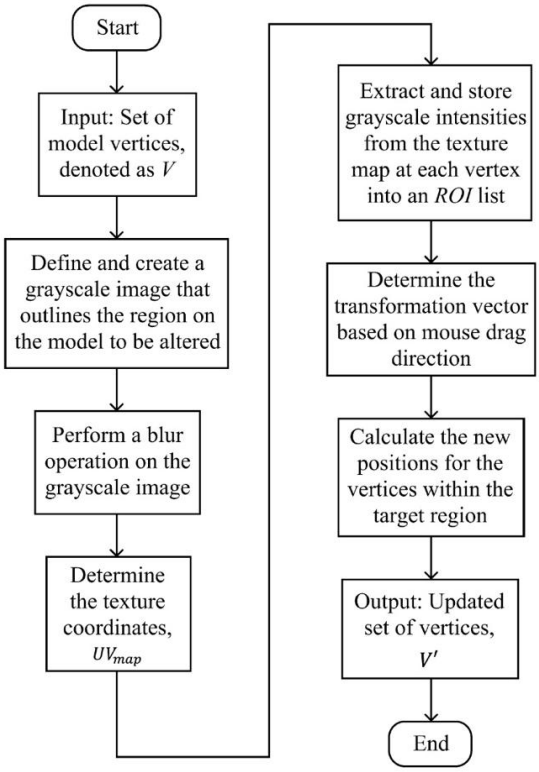

Fig. 6. Flow Chart of Mesh Deformation Algorithm
(Picture credit: Original)

3.3 Application

The 3D cosmetic surgery simulation system developed by the Guangdong Provincial Key Laboratory of Computer Integrated Manufacturing at the Guangdong University of Technology has applied this Mesh Deformation algorithm Based on Grayscale Texture. This system, which is crafted for offline medical aesthetic simulations on a PC-based platform, leverages this algorithm to provide realistic and precise simulations of various cosmetic procedures. This system comprises three main components: a PC running Windows 10 Pro. 64-bit, the cosmetic surgery simulation software, and the FaceGo180G 3D facial scanner by Revopoint [4]. This comprehensive system simulates over 20 different cosmetic procedures, including eyelid surgery, rhinoplasty, lip enhancement, and facial slimming, to provide realistic portrayals of potential post-operative outcomes. Developed with the Unity3D game engine and leveraging the OpenCVSharp package, the interface is strategically segmented into four principal functional modules, as shown in Fig. 7.

Fig. 7. Structural Mockup of Original System Interface
(Picture credit: Original)

The Cosmetic Vault module uses XML files to store patient data and manages a database for facial cosmetic knowledge, facilitating actions like registration, deletion, or modification of patient information and managing 3D facial models. The Facial Analysis module employs facial recognition technology to identify key facial landmarks and generates detailed reports on facial metrics. The Intelligent Cosmetic Procedures module also uses facial recognition to automatically pinpoint areas for treatment, minimizing

manual input and enhancing user interaction. Lastly, the Precision Sculpting Tools module provides manual tools for users to select and adjust any facial area, allowing greater control and customization in the simulation process.

The system has been preliminarily launched in the market and is in partnership with The General Hospital of the Southern Theater Command of the Chinese People's Liberation Army for further clinical application. By employing this simulation system, Fig. 8 and Fig. 9 demonstrate the before-and-aftereffects of specific cosmetic procedures.

Fig. 8. Facial Slimming Simulation [5]

Fig.9. Rhinoplasty Simulation [5]

4 Conclusion

This review evaluates two advanced mesh deformation techniques in cosmetic surgery simulations: volumetric mesh deformation using Laplacian coordinates and grayscale texture-based mesh deformation. The Laplacian-based method, demonstrated on a simplified muscle model, effectively preserves volume and achieves realistic deformations within fixed boundary conditions, making it ideal for precise volumetric representations. Enhancements could focus on improving rotation invariance and introducing adaptive weighting to increase accuracy and realism. Integrating machine

learning could also optimize deformation parameters, enhancing computational efficiency and accuracy in real-time applications. The grayscale texture-based method reduces preprocessing by encoding deformation data directly into grayscale images. Applied in the Guangdong Provincial Key Laboratory's 3D cosmetic surgery simulation system, this method facilitates rapid and flexible adjustments, proving particularly effective in dynamic surgical environments. Future advancements could improve texture mapping techniques to automatically adjust granularity based on deformation complexity, thus boosting performance and accuracy.

References

1. Sumner, R. W., Zwicker, M., Gotsman, C., Popović, J.: Mesh-based Inverse Kinematics. ACM Transaction on Graphics 24(3), 488–495 (2005).
2. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H. Y.: Large Mesh Deformation Using the Volumetric Graph Laplacian. ACM Transaction on Graphics 24(3), 496–503 (2005).
3. Kim, G., Baek, N.: A Height-Map Based Terrain Rendering with Tessellation Hardware. In: 2014 International Conference on IT Convergence and Security (ICITCS), pp. 1-4 (2014).
4. Li, R.: Geometric Information Processing in Facial Plastic Surgery Simulation Systems. Guangdong University of Technology (2018).
5. Li, J., Li, R., He, H., Xu, J., Liu, J., Li, H.: Mesh Morphing Process for Facial Plastic Surgery Simulation. Journal of System Simulation 30(7), 2453-2458 (2018).
6. Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rossi, C., Seidel, H.P.: Differential coordinates for interactive mesh editing. In: Proceedings of Shape Modeling Applications, Genova, Italy, pp. 181-190 (2004).
7. Sorkine, O., Cohen, O. D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H. P.: Laplacian Surface Editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '04), Association for Computing Machinery, New York, NY, USA, pp. 175–184 (2004).
8. Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D.: Linear Rotation-Invariant Coordinates for Meshes. ACM Transaction on Graphics 24(3), 479–487 (2005).
9. Bloch, I., Pescatore, J., Garnero, L.: A new characterization of simple elements in a tetrahedral mesh. Graphical Models 67(4), 260-284 (2005).
10. Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and Mathematics III, Springer, Berlin, Heidelberg, pp. 35-57 (2003).
11. Au, O. K. C., Tai, C. L., Liu, L., Fu, H.: Dual Laplacian editing for meshes. IEEE transactions on visualization and computer graphics 12(3), 386-395 (2006).
12. Liao, S.: Research on Medical Model Reconstruction and Volumetric Mesh Generation and Deformation. PhD Dissertation, Zhejiang University (2008).
13. Series, B.T.: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. International Telecommunication Union, Radiocommunication Sector (2011).