B

2021 11th International Conference on Information Science and Technology (ICIST) | 978-1-6654-1266-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICIST52614.2021.9440622

Research on Simplified Mesh Deformation Based on Differentiable Computation

Zhuo Shi
School of Art and Design,
School of Computer Science and Information Security,
Guilin University of Electronic Science and Technology
Guilin, China
shzh@guet.edu.cn

Shuzhen Zeng
School of Computer Science and Information Security,
Guilin University of Electronic Science and Technology
Guilin, China
shuzhenzeng@foxmail.com

Xiaonan Luo
School of Computer Science and Information Security,
Guilin University of Electronic Science and Technology
Guilin, China
luoxn@guet.edu.cn

***Abstract*—In this paper, we propose a simplified mesh deformation method based on the differentiable calculation and uses the Pytorch3D library of deep learning. There are four stages, simplification, deformation, subdivision, re-deformation in this method. The simplification stage transforms the original target mesh into a simple mesh. The deformation stage uses the Pytorch3D tool to predict the simple mesh in the simplification result. The subdivision stage subdivides the resulting mesh of deformation, and the re-deformation stage uses the subdivision stage result mesh as the source mesh to predict the original target mesh. Our experiment shows that the number of iterations is similar or less in terms of shape and local features after simplifying the predicted target mesh. Our method is superior to the direct mesh deformation method in terms of mesh deformation speed and local mesh characteristics of deformation and has a better deformation effect.**

Keywords-component; 3D deformation; differentiable calculation; uniform sampling

I. INTRODUCTION

In computer graphics, mesh deformation has always occupied an important position, and it is a research hotspot in computer animation and geometric modeling. Currently, many technologies developed to provide natural-looking deformation. Mesh deformation refers to a geometric processing technology that modifies a geometric model's shape through user editing operations or constraints. It has a wide range of applications in industrial and artistic design and computer animation. For example, in 3D animation, the existing model needs to be deformed to generate keyframes or interpolated frames between keyframes.

In the early days, free deformation technology [1][2] was the main mesh deformation technology. Its algorithm was simple, and it was a point-by-point operation. A large amount of manual adjustment was required for meshes with rich local details. In local detail editing, multiresolution mesh editing technology [3][4][5] overcomes the shortcomings of free deformation technology, and it could edit in overall control and local detail. It has two shortcomings, and one is the need to construct a progressive representation of the geometric model. The need for semi-regular resampling limits the other. At the beginning of the 21st century, the mesh deformation technology in the differential domain [6][7][8][9][10], the advantage of the algorithm is that it does not need to decompose the original mesh, and it can maintain the local details like the grid-like multiresolution technology. The disadvantage is that the mesh deformation is only suitable for local deformation, limiting the overall mesh deformation. The early SSD technology [11] is simple and efficient. But its disadvantage that it requires professional manual pre-operation in the early stage. Therefore, Xu et al. proposed an improved SSD [12], which combines SSD with differential domain coordinates and means skeleton coordinates to maintain local geometric features and skeleton features. The algorithm can be operated simpler than SSD and can generate real visual effects. In mesh deformation, the average coordinate has been widely used in shape deformation [13]. The average coordinate can represent a point in the star polygon's kernel as a convex combination of vertices. It can use to calculate the good parameterization of the surface representing the triangle mesh. Sha et al. proposed a 3D mesh deformation technology based on average coordinates [14]. To reduce the amount of calculation, the method proposed by Sha reduces the number of vertices involved in the deformation. First, using an improved CPM algorithm to construct a simplified mesh, editing the simplified mesh through the mesh detail representation, which bases on the mean coordinates, and adding a series of details to the deformed simplified mesh to make the mesh deform natural.

In recent years, the center of gravity coordinates' idea extended to arbitrary polygons and high-dimensional polyhedrons in the plane. That produced new solutions for grid parameterization and grid deformation applications. Krauth et al. proposed an interactive modeling framework for 3D shape and texture mapping [15], which combined the idea of geometric brushes based on differential deformation, and it achieves local scaling. The disadvantage is not considering the gradient field changes. Chen et al. proposed skeleton-driven surface deformation to get on real-time role animation [16]. The means is by constructing a cubic unit grid, which grid contains the input surface mesh, and automatically transmitting the weight of the smooth skin of the bone is to drive the surface to deform. The disadvantage is that it can only be positive numbers weighted calculation. Manson et al. proposed Gordon-Wixom coordinates [17], which introduces a new structure of the arbitrarily closed center of gravity coordinates in two dimensions. It uses the boundary curve distance tangent to define a positive and smooth weight function. But its disadvantage is that it uses polygons closed solution that can only approximate the smooth boundary. Sun et al. proposed a moving grid deformation method based on the center of gravity coordinates [18]. To calculate the displacement of the grid point, it according to the displacement of the center of gravity coordinates interpolating the boundary point. So it can calculate the center of gravity coordinate of the grid point that needs to move. The disadvantage is that the calculation amount is relatively large. Xian et al. use a full motion method to generate a thick cage surrounding the mesh model [19]. The automatically generated coarse boundary can maintain the topological structure and the original mesh model's main geometric characteristics. It is convenient to perform operations such as deformation, subdivision, and fusion. But the disadvantage is that it has higher requirements for the original mesh. Du et al. proposed an approximate rigid deformation algorithm for volume map control [20]. The means use the approximate rigidity of the volume map to constrain the deformation volume, and it overcomes the unreasonable volume change in the traditional deformation process. But the disadvantage is that the algorithm requires multiple iterations, and the parallel calculation is slow. Zhang et al. proposed a triangular mesh deformation method based on the generalized center of gravity coordinates [21]. It makes the original model enclosed in the control grid by operating the control grid's deformation to drive the deformation of the original model, which can generate a uniform control grid. The disadvantage is that no considering the simplification and smoothing of the model. Wu et al. proposed a Poisson grid editing algorithm combining the center of gravity coordinates [48]. First of all, determine the grid model's enclosing grid to be deformed, then it generates the control grid with the center of gravity coordinates determined by the enclosing grid. Secondly, arbitrarily manipulate the control grid, map the information corresponding to the center of gravity's change coordinate to the gradient field change of the grid model. Finally, reconstructing the grid by solving the coefficient matrix of Poisson's equation. Then the deformed model is obtained. This algorithm overcomes the defect of the traditional Poisson mesh deformation algorithm. And it effectively retains the detailed characteristics of the model deformation.

In the past ten years, deep learning has greatly improved AI systems' ability to process 2D image data. Now, we can classify objects [22][23][24][25][47] and scenes [26][27] as well as target detection [28], semantic [29] and instance [30]. As well as the construction of high-performance system posture estimation [31]. MADNet [46] is a dense and lightweight network. It can realize multi-scale feature expression and feature-related learning. These systems can run on complex image data and deploy in actual environments. Although these methods are successful, they have a common shortcoming: they deal with 2D snapshots and ignore the real 3D nature of the world. Thus, we input a single RGB image to predict the 3D object grid. According to [32], we can use a two-view training setting: each object image in the mini-batch processing includes its corresponding view under a random known transformation. In addition, some 3D reconstruction methods [33][34][35][36][37] have also produced good results. Inspired by [38], all models take a single image as input and directly predict the 3D object grid in camera coordinates.

Now the mesh deformation method uses a very complex model to achieve user-specified deformation, leading to an unstable deformation effect. Gao proposed a sparse mixing method [39]. This method can automatically select a small number of deformations. To approach the required deformation. But Wang proposed a deep learning framework [40]. It is an end-to-end depth learning framework for generating 3D shapes from monochromatic images. Using perceptual features extracted from input images, then generate the correct geometric shape. A strategy from coarse to fine is adopted to ensure stable deformation, and various losses related to the mesh/surface are defined. It ensures a visually attractive and physically accurate 3D geometric shape. The shape grid does not need to rely on voxels, point clouds, or other more information-rich data. Then Gkioxari combined two areas of progress: synthetic benchmarks and isolated targets. Also, they improved Mask R-CNN [30] and added grid prediction branches. They then proposed a Mesh R-CNN model [38], which can generate a 3D object and a complete three-dimensional triangular mesh of the 3D object. The generated triangular mesh keeps similar features to the 3D object.

There are many ways to simplify the grid. Among them, garland [41] and others proposed the quadratic error metrics (QEM). This method uses edge collapse and the square of the distance from a point to plane as the error measure. When QEM calculates, choose the edge with the smallest folding cost, shrink the pair of vertices in the mesh with the least cost of edge folding operations (that is, two points shrunk to one point), simplify the model step by step by continuous iteration. This algorithm simplifies the model step by step by iteration. In this paper, the simplification method's mesh adopts QEM, and the speed and quality of mesh simplification are relatively good.

The improvement of 3D understanding ability will be more conducive to the machine to better understand the real world. For example, robots can navigate in complex spaces. It can recognize occluded objects in 2D environments and even be used to improve the scene of various 3D-related AR/VR games. Therefore, the focus of this paper is to introduce 3D into computer vision. Unlike the mesh mentioned above deformation, in this paper, the mesh deformation uses the deep learning library Pytorch3D, which can transform from a 3D shape into another shape. The initial mesh can be any mesh. After the mesh subdivides, then obtain the source mesh. The source mesh was deforming by offsetting the vertices of its mesh. By randomly and uniformly sampling in the source grid and the target grid, it can generate the center of gravity coordinates and achieve grid prediction.

This paper proposes a simplified mesh deformation method for deep learning. The initial mesh is subdividing by the Loop method [42] to obtain the source mesh. Simultaneously, the simplified mesh of complex target meshes regards as the first deformation target mesh. By using the pytorch3d library [43], the source meshes transformed into the target mesh. Then, the first deformation result mesh is subdividing by the Loop. The result of the subdivision is close to the vertex number of the target mesh before simplification. Finally, the subdivision result mesh is used as the source mesh to form the original target mesh. The experimental results show that the deformation result is better than without simplification. The results show that mesh deformation's shape and details are better than fitted to the predicted target mesh.

II. ALGORITHM OVERVIEW

A. Loop subdivision method

The subdivision method used in this experiment is the Loop subdivision [42]. Loop subdivision is a classic triangular mesh subdivision algorithm. It was proposed by Loop of the University of Utah in 1987 in a master's thesis based on a triangular mesh. The subdivision mode uses a 1-4 triangle face splitting method, only generates new edge points and new vertex calculations. And then connects through topology rules, subdivision generates an approximate triangle mesh.

Set the two vertices of the inner side as (v_0, v_1), the v_2 and v_3 are the two vertices of two triangles, then the vertex calculations are as follows:

E-vertex: the two triangular faces sharing this side are (v_0, v_1, v_2) and (v_0, v_1, v_3), then the calculation of the E vertex is shown in (1).

$$V_E = \frac{3}{8}(v_0 + v_1) + \frac{1}{8}(v_2 + v_3) \tag{1}$$

V-vertex: If the neighboring points of the inner vertex V are v_0 , v_1 , … v_{n-1} , when n=|V|E, the calculation of the corresponding V-vertex is (2).

$$V_V = (1 - n\beta_n)v + \beta_n \sum_{i=0}^{n-1} V_i \tag{2}$$

That is the weighted sum of the vertex itself and all its neighboring vertices, its weight is the value of $1 - n\beta_n$, and the neighbor weight is (3).

$$\beta_n = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right) \tag{3}$$

When n=3, $\beta_3 = \frac{3}{16}$. And when n>3, $\beta_n = \frac{3}{8n}$.

E-vertices V_E on boundary edges (v_0, v_1) is (4). The two adjacent vertices of the boundary vertex V on the boundary are (v_0, v_1), then the V-vertex is (5).

$$V_E = \frac{1}{2}(v_0 + v_1) \tag{4}$$

$$V_V = \frac{1}{8}(v_0 + v_1) + \frac{3}{4}V \tag{5}$$

B. Simplified mesh deformation method

The existing grid processing methods were proposed by Kato [44] and Liu [45]. But there are two disadvantages to the existing methods. One is either batch not supported;the other is to assume that the mesh in a batch has the same number of vertices and faces. Only CUDA implementations were provided in the existing project, so they cannot use without a GPU. Inspired by Mesh R-CNN [38], Facebook created Pytorch3d [43]. Pytorch3D provides a set of commonly used 3D operators and fast and differentiable loss functions for 3D data. The Meshes format created by it can simplify the complexity of batch processing of 3D models. It can use for batch processing of heterogeneous mesh model data structures. Pytorch3d can handle a different number of vertices and faces, and it also supports GPU. That greatly simplifies the research of 3D mesh deep learning.

In order to save storage space before predicting the target grid, this paper simplifies the target grid. In the process of prediction, the number of vertices of the grid does not change. Besides, in order to speed up the deformation, the neural network optimizer SGD is used. The grid predictor used in this paper is similar to the grid predictor of Mesh R-CNN [38]; the loss of shape is similar to the loss of a grid. It is challenging to define the loss of local operations on a triangular grid. This paper uses a loss function defined on a finite set of points by sampling a point cloud from the grid's upper surface to represent a grid. This paper adopts the differentiable network sampler to sample points evenly from the grid surface—point cloud from the source grid and point cloud from the target mesh.

In this paper, the calculation process of mesh loss is as follows: set two sets of point cloud P, Q and normal vector, let $\Lambda_{P,Q} = \{(p, \arg\min_q \| p - q \|) : p \in P\}$,$(p, q)$ become a set of data pair (p, q), where q is the nearest neighbor of$p$ in Q, and u_p is the normal vector of point p, and u_q is the normal vector of point q. The chamfer distance between point cloud P and Q $L_{\text{cham}}(P, Q)$ is (6):

$$L_{\text{cham}}(P,Q) = |P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} \| p - q \|^2 + |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} \| q - p \|^2 \tag{6}$$

The normal distance is (7):

$$L_{\text{norm}}(P,Q) = -|P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} | u_p \cdot u_q | - |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} | u_q \cdot u_p | \tag{7}$$

Chamfer and normal distance measure the mismatch position and normal between two point clouds. However, only minimizing these two distances will result in mesh degradation. High-quality mesh prediction requires additional shape adjusters, so edge loss $L_{\text{edge}}(V, E)$ is (8):

$$L_{\text{edge}}(V,E) = \frac{1}{|E|} \sum_{(v,v')\in E} \| v - v' \|^2 \tag{8}$$

$E \subseteq V \times V$ is the edge of the prediction mesh. Besides, adding Laplace loss, which using it to impose smoothing constraints. The grid loss of the iteration number is the

weighted sum of $L_{\text{Laplacian}}$, $L_{\text{norm}}(P^i, Q^i)$, $L_{\text{edge}}(V, E)$, and Laplace loss.

The experimental steps of the simplified mesh deformation method are as follows:

Input: initial mesh, target mesh.

Output: deformation results mesh.

Step1. Read in the initial mesh and use the Loop subdivision to subdivide it and then obtain the first deformation's source grid.

Step2. Simplifying the target mesh by QEM.

Step3. Use the source grid obtained in step 1 to predicts the simplified target grid obtained in step 2. Then we can get the deformed result grid of the first prediction.

Step4. The deformed result grid obtained in Step 3, we use to subdivided by Loop subdivision. The subdivision result grid predicts the original target grid to obtain the second deformed result grid.

III. EXPERIMENT

The existing mesh deformation method only edits the source mesh and does not change the mesh's essential shape. For example, Fig. 1 shows the 3D grid deformation method based on average coordinates [14]. And Fig. 2 shows a Poisson grid editing algorithm combined with the center of gravity coordinates [48]. This method uses the Poisson grid editing method combined with the center of gravity coordinates to enclose. It was using the grid to determine the center of gravity coordinates to obtain the control grid. Then re-set the effect of the differential coordinate direction and adjust the direction of the gradient field whole. Finally, reconstructing the deformed model.

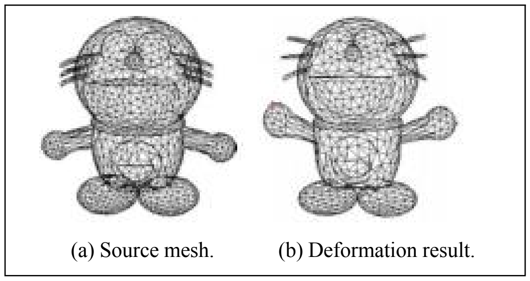


Figure 1. Deformation of the Doraemon model [14].

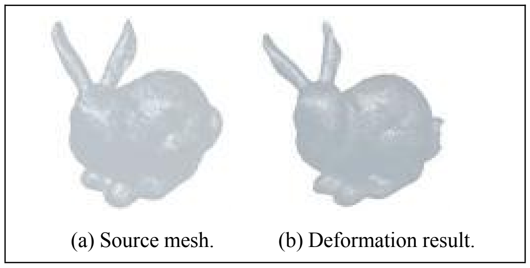


Figure 2. Deformation of the bunny model [48].

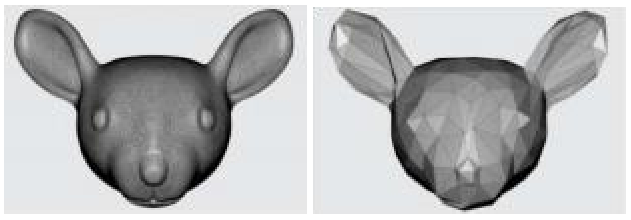
The above mesh deformation methods are only mesh editing in the source mesh. The above mesh methods are only mesh editing in the source mesh. And they were only changing the local characteristics of the source mesh. In this paper, the algorithm introduces a deep learning method that aims to change the source grid's global characteristics. The purpose is to transform the mesh model into a target mesh.

In the experiment, both the initial grid and the target grid are files in OBJ format. In the mesh deformation, the initial mesh's vertices and the target mesh keep consistent. The experiment compares the deformation result mesh after the source mesh predicts the target mesh. The simplified mesh deformation method proposed is to simplify the target mesh to deformation. Then Respectively predict the target mesh before and after the simplification. The simplified result of the target grid is showing in Fig 3. Otherwise, adding the experiment of directly predicting the original target grid to comparison. They are, namely, subdividing the source grid into the number of vertices close to the target grid for direct prediction. In this paper, the mesh deformation performers twice. And the final deformation result mesh is much better than the mesh deformation result of the direct deformation method, especially in the details. In this paper, the initial grids are dodecahedron and cube and select the target grids as mouse heads and fish for the experiment.

IV. ANALYSIS OF EXPERIMENTAL RESULTS

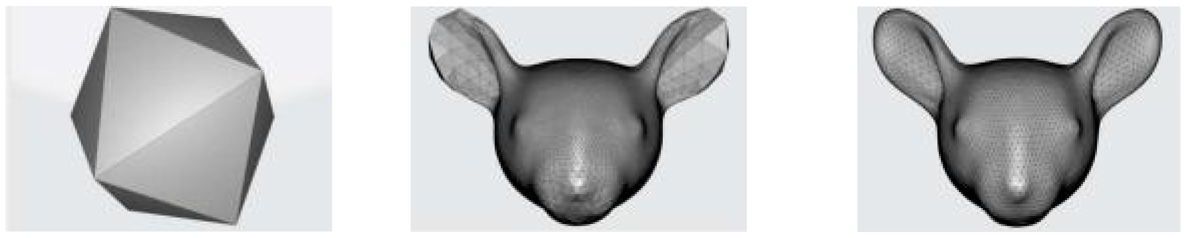
In this paper, to implement the algorithm, the processor used is Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz and 8GB of memory. And the GPU is NVIDIA Geforce 930M. The simplified mesh environment is in Meshlab. Pytorch3D configures under Ubuntu.

It can be seen from Fig. 4 that the initial mesh is in a regular dodecahedron. In the direct mesh deformation method and the simplified mesh deformation method, deformation results are roughly the same as the target mesh. But at the ears, the direct mesh deformation method is in the deformed ear during the process, there are missing corners, and the mesh is too sparse. That is not the case with the simplified mesh deformation method in this paper. The predicted result is roughly the same as the target mesh. In the nose, the deformation result of the direct mesh deformation algorithm is too rough. However, the deformation results of the simplified mesh deformation algorithm are dense. And its result is also closer to the nose of the target mesh. Obviously, in the ear and nose of the mouse's head, the deformation result of the simplified mesh deformation method is better than that of the direct mesh deformation method.



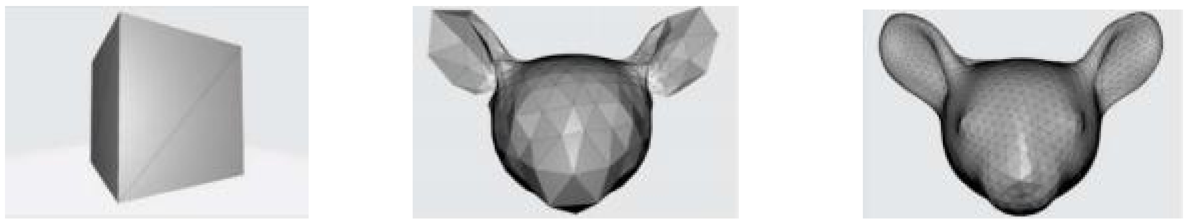
(a) Original target mesh. (b) Simplified target mesh.

Figure 3. Mesh simplification results of the mouse head model.



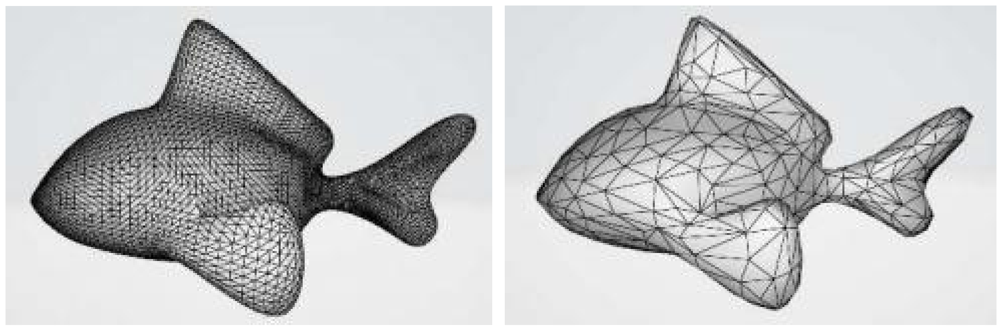
(A)Initial mesh. (b)Deformation result of direct deformation method. (c)Deformation result of the algorithm in this paper.

Figure 4. Deformation result of mouse head model. (the initial model is a regular dodecahedron).



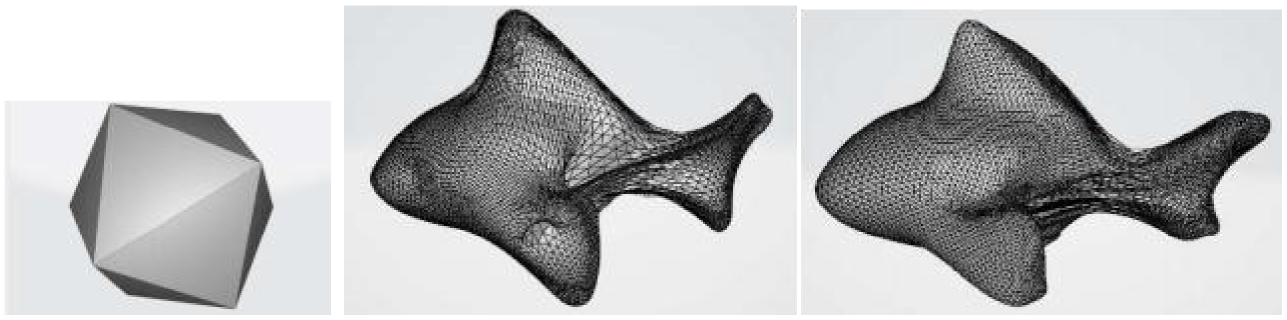
(A)Initial mesh. (b)Deformation result of direct deformation method. (c)Deformation result of the algorithm in this paper.

Figure 5. Deformation result of mouse head model. (the initial model is a cube).



(a)Original target mesh. (b)Simplified target mesh

Figure 6. Deformation result of the fish model.



(A)Initial mesh. (b)Deformation result of direct deformation method. (c)Deformation result of the algorithm in this paper.

Figure 7. The comparison results of the two algorithms of the fish model.

It can be seen from Fig. 5 that the original mesh is a cube. The resulting mesh of the direct mesh deformation method does not fit the target mesh's local features. There is only a rough outline, there are missing corners at the ears, and the eyes and nose are not to predict. It appears that the effect of mesh deformation is extremely poor. However, the simplified mesh deformation method does not have such problems. It predicts the ears and nose successfully. It can seem that the simplified mesh deformation method is better than the direct mesh deformation method at the mouse's ears and nose. This kind of problem occurs due to uniform sampling on the source grid's surface and the target grid during the prediction process.

It can be seen from Fig. 7 that the original mesh is a regular dodecahedron. As a result of the direct mesh deformation method, the effect of local feature deformation is not very good. It's rough. But the fin position fitted by the simplified mesh deformation algorithm is better. The results are close to the characteristics of the fin. It can seem that in the fin of fish, the deformation result of the simplified mesh deformation method is better than that of the direct mesh deformation method.

According to the above analysis,even if change the original model, from the perspective of the local characteristics of the resulting mesh, the simplified mesh deformation method in this paper is better than the direct mesh deformation method in transforming the target mesh.

V. CONCLUSION

This paper proposes a simplified mesh deformation method based on deep learning—it through four stages, simplification, deformation, subdivision, re-deformation. The final deformation effect is better than the direct mesh deformation method in detail and local features, and it can better fit the target mesh. In the experiment, using the neural network optimizer for training, grid deformation speed can be accelerated. In the future,we will continue to improve the mesh deformation algorithm. We hope to achieve a better deformation effect. That has a major role in computer vision, games, film, television, medicine, virtual reality, etc.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (Nos. 61862018, 61772149, 6202780103, 2018AAA0100305, and 62033001), Guangxi Science and Technology Project (Nos. 2018GXNSFAA138084, 61762028, AA18118039, AD18216004), Innovation Project of GUET Graduate Education (Nos. 2021YCXS056).

REFERENCES

[1] T. W. Sederberg, S. R. Parry, "Free-form deformation of solid geometric models," Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 151-160, 1986.

[2] R. MacCracken, K I. Joy, "Free-form deformations with lattices of arbitrary topology," Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 181-188, 1996.

[3] D. Zorin, P. Schröder, W. Sweldens, "Interactive multiresolution mesh editing," Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 259-268, 1997.

[4] L. Kobbelt, S. Campagna, J. Vorsatz, H. Seidel, "Interactive multi-resolution modeling on arbitrary meshes," Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 105-114, 1998.

[5] I. Guskov, W. Sweldens, P. Schröder, "Multiresolution signal processing for meshes," Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 325-334, 1999.

[6] M. Alexa, "Differential coordinates for local mesh morphing and deformation," The Visual Computer, vol. 19, no. 2, pp. 105-114, 2003.

[7] O. Sorkine, D. Cohen-Or, Y. Lipman, et al., "Laplacian surface editing," Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry processing, pp. 175-184, 2004.

[8] Y. Yu, K. Zhou, D. Xu, et al., "Mesh Editing with Poisson-Based Gradient Field Manipulation," ACM SIGGRAPH 2004 Papers, pp.644-651, 2004.

[9] J. Huang, X. Shi, X. Liu, et al., "Subspace Gradient Domain Mesh Deformation," ACM SIGGRAPH 2006 Papers, pp. 1126-1134, 2006.

[10] X. Shi, K. Zhou, Y. Tong, et al., "Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics," ACM SIGGRAPH 2007 papers, pp.81-es, 2007.

[11] N. Magnenat-Thalmann, R. Laperriere, D. Thalmann, "Joint-dependent local deformations for hand animation and object grasping," In Proceedings on Graphics interface'88, pp. 26-33, 1988.

[12] Q. Xu, G. Tan, S. Zhang, Y. Zhang, "Deformation of the subspace grid of the mean skeleton maintaining geometric features," Journal of

Computer-Aided Design and Graphics, vol. 21, no. 03, pp. 289-294, 2009.

[13] T. Ju, S. Schaefer, J. Warren, "Mean value coordinates for closed triangular meshes," ACM Siggraph 2005 Papers, pp. 561-566, 2005.

[14] C. Sha, X. Zhang, Y. Yue. "3D meshes deformation based on mean value coordinates," International Conference on Software Intelligence Technologies and Applications & International Conference on Frontiers of Internet of Things 2014, pp.288-291, 2014.

[15] N. Krauth, M. Nieser, K. Polthier, "Differential-based geometry and texture editing with brushes," Journal of mathematical imaging and vision, vol. 48, no. 2, pp. 359-368, 2014.

[16] C.-H. Chen, M.-H. Tsai, I.-C. Lin, Pin-Hua Lu, "Skeleton-driven surface deformation through lattices for real-time character animation," The Visual Computer, vol. 29, no. 4, pp. 241-251, 2013.

[17] J. Manson, K. Li, S. Schaefer, "Positive Gordon-Wixom coordinates," Computer-Aided Design, vol. 43, no. 11, pp. 1422-1426, 2013.

[18] S. Sun, B. Chen, "An Algebraic Deformation Approach for Moving Grid Based on Barycentric Coordinates," 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, 2010, pp. 1-4.

[19] C. Xian, H. Lin, S. Gao, "Automatic generation of coarse bounding cages from dense meshes," 2009 IEEE International Conference on Shape Modeling and Applications, Beijing, China, 2009, pp. 21-27.

[20] Z. Du, H. Zhang, "Nearly rigid mesh deformation controlled by the volumetric diagram," Journal of Computer-Aided Design and Graphics, vol. 28, no. 2, pp. 218-227, 2016.

[21] H. Zhang, "Research and application of triangular mesh model deformation method based on the generalized center of gravity coordinates," Jilin University, 2015.

[22] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409, p.1556, 2014.

[23] O. Russakovsky, J. Deng, H. Su, et al., "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, 2015.

[24] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.

[25] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[26] F. Yu, A. Seff, Y. Zhang, et al., "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," arXiv preprint arXiv:1506.03365, 2015.

[27] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, A. Oliva, "Places: An image database for deep scene understanding," arXiv preprint arXiv:1610.02055, 2016.

[28] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017.

[29] J. Long, E. Shelhamer, T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440.

[30] K. He, G. Gkioxari, P. Dollar, R. Girshick, "Mask R-CNN," Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969.

[31] R. A. Güler, N. Neverova, I. Kokkinos, "Densepose: Dense human pose estimation in the wild," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7297-7306.

[32] S. Liu, W. Chen, T. Li, H. Li, "Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction," arXiv preprint arXiv:1901.05567, 2019.

[33] C. B. Choy, D. Xu, J. Y. Gwak, K. Chen, S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction, "European conference on computer vision. Springer, Cham, pp. 628-644, 2016.

[34] H. Fan, Ha. Su, L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 605-613.

[35] W. Chen, J. Gao, H. Ling, et al., "Learning to predict 3d objects with an interpolation-based differentiable renderer," arXiv preprint arXiv:1908.01210, 2019.

[36] C. Nash, Y. Ganin, S. M. A. Eslami, P. Battaglia, "Polygen: An autoregressive generative model of 3d meshes," International Conference on Machine Learning. PMLR, pp. 7220-7229, 2020.

[37] O. Wiles, G. Gkioxari, R. Szeliski, J. Johnson, "Synsin: End-to-end view synthesis from a single image," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7467-7477.

[38] G. Gkioxari, J. Malik, J. Johnson, "Mesh R-CNN," Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 9785-9795.

[39] L. Gao, Y. K. Lai, J. Yang, et al., "Sparse Data Driven Mesh Deformation," in IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 3, pp. 2085-2100, 1 March 2021.

[40] N. Wang, Y. Zhang, Z. Li, et al., "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," IEEE transactions on pattern analysis and machine intelligence, 2020.

[41] M. Garland, P. S. Heckbert, "Surface simplification using quadric error metrics," Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 209-216, 1997.

[42] C. Loop, "Smooth subdivision surfaces based on triangles," Master's thesis, University of Utah, Department of Mathematics, 1987.

[43] N. Ravi, J. Reizenstein, D. Novotny, et al., "Accelerating 3d deep learning with pytorch3d," arXiv preprint arXiv:2007.08501, 2020.

[44] H. Kato, Y. Ushiku, T. Harada, "Neural 3d mesh renderer," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3907-3916, 2018.

[45] S. Liu, W. Chen, T. Li, H. Li, "Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), pp. 7707-7716, 2019.

[46] R. Lan, L. Sun, Z. Liu, H. Lu, et al., "MADNet: A Fast and Lightweight Network for Single-Image Super Resolution," IEEE Transactions on Cybernetics, vol. 51, no. 3, pp. 1443-1453, March 2021.

[47] R. Lan, Y. Zhou, Z. Liu, X. Luo, "Prior Knowledge-Based Probabilistic Collaborative Representation for Visual Recognition," IEEE Transactions on Cybernetics, vol. 50, no. 4, pp. 1498-1508, April 2020.

[48] S. Wu, Q. Huang, H. Li, A. Wang, "Poisson grid editing algorithm combined with the center of gravity coordinates," Journal of Taiyuan University of Science and Technology, vol. 40, no. 2, pp. 111-115, 2019.