The Visual Computer (2021) 37:2741–2753
https://doi.org/10.1007/s00371-021-02169-4

ORIGINAL ARTICLE

Real-time surface manipulation with C^1 continuity through simple and efficient physics-based deformations

Shuangbu Wang[1] · Nan Xiang[1] · Yu Xia[1] · Lihua You[1] · Jianjun Zhang[1]

Accepted: 19 May 2021 / Published online: 17 June 2021

Abstract
We present a novel but simple physics-based method to interactively manipulate surface shapes of 3D models with C^1 continuity in real time. A fourth-order partial differential equation involving a sculpting force originating from elastic bending of thin plates is proposed to define physics-based deformations and achieve C^1 continuity at the boundary of deformation regions. In order to obtain real-time physics-based surface manipulation, we construct a mapping relationship between a deformation region in a 3D coordinate space and a unit circle on a 2D parametric plane, formulate corresponding C^1 continuous boundary conditions for the unit circle, and obtain a simple analytical solution to describe the physics-based deformation in the unit circle caused by a sculpting force. After that, the obtained physics-based deformation is mapped back to the 3D coordinate space, and added to the original surface to create a new surface shape with C^1 continuity at the boundary of the deformation region. We also develop an interactive user interface as a plug-in of the 3D modelling software package Maya to achieve real-time surface manipulation. The effectiveness, easiness, real-time performance, and better realism of our proposed method is demonstrated by testing surface deformations on several 3D models and comparing with other methods and ground-truth deformations.

Keywords Surface manipulation · Physics-based deformations · Partial differential equation · User interface · Mapping

1 Introduction

Surface manipulation, also known as surface or mesh editing, is the fundamental research topic in geometric modelling and computer-aided design. It is to manipulate and change the global or local surface shapes of 3D models by using manual deformation operations such as extending, extruding and twisting. Depending on whether physics of the model deformation is introduced or not, surface manipulation is roughly divided into two categories: purely geometric and physics-based [1]. Purely geometric surface manipulation achieves shape deformations by manually changing the positions of mesh vertices or control points, while physics-based surface manipulation generates different surface shapes by exerting virtual forces, torques or other physical quantities to deform the surfaces.

✉ Yu Xia
yxia@bournemouth.ac.uk

[1] National Centre for Computer Animation, Bournemouth University, Poole, UK

At an early stage, the commonly used purely geometric deformation method is to directly move mesh vertices of polygonal models or control points of NURBS models [2]. In order to improve the efficiency and capability of surface manipulation, free-from deformation (FFD) methods were developed, which embed an object within a 3D lattice and simulate the deformations by moving lattice control points [3,4]. Another widely used surface manipulation method is Laplacian coordinates [5,6], which describes the relation of each vertex to its local neighbourhood and modifies the derivatives of the surface to find the best-fit positions of vertices by solving a system involving the Laplacian. Since purely geometric surface manipulation methods do not follow any underlying physical laws, the quality of deformed shapes depends on the personal skills and perceptions of users, and different users may create somewhat different shapes for the same model. This issue may be resolved by introducing the underlying physics governing the deformation of deformable materials, and it also has a potential to create more realistic looking objects. A popular physics-based surface manipulation method is physics-based NURBS [7–9], which allows users to manipulate the surface shapes

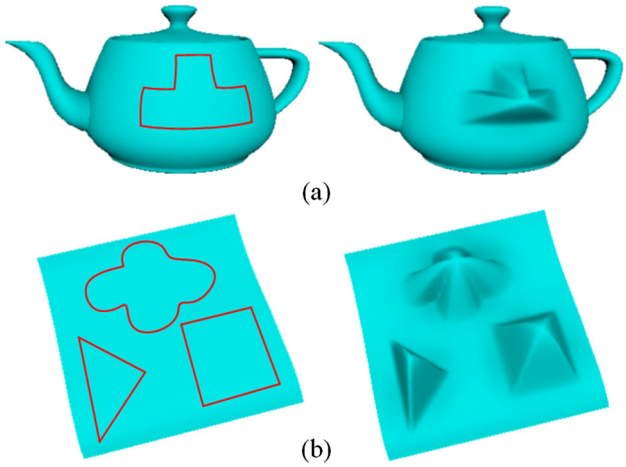


Fig. 1 Surface manipulation using the proposed method. **a** Single deformation on a teapot model, **b** Multiple deformations on a surface. (Red curves indicate the deformation regions)

of a model through not only adjusting control points and setting weights but also applying simulated forces and local and global shape constraints. However, physics-based methods usually involve heavy numerical calculations and slow responding time of real-time operations.

In this paper, we investigate real-time and physics-based surface manipulation and propose a novel but simple and efficient physics-based method to interactively manipulate surface shapes of 3D models in real time. As shown in Fig. 1, our method can deal with local deformations with an arbitrarily complicated boundary shape and C^1 continuity. We firstly select a deformation region on a surface in a 3D coordinate space, and map the boundary of the deformation region to a unit circle on a 2D parametric plane. Then, we define the circle as a thin elastic plate and simulate bending deformation of this plate by applying a sculpting force. To obtain the deformation, we derive an approximate analytical solution of a fourth-order partial differential equation (PDE) subjected to the sculpting force and the boundary conditions of the circle for x, y and z components. We apply a length-based method to determine the corresponding relationship between the vertices within the boundary of the deformation region and those within the circle. After obtaining the deformation of the unit circle, the coordinate values of all vertices within the deformation region are renewed to create a new surface shape according to the deformed values of corresponding points on the parametric plane.

Our method has five advantages. First, our method is physics-based and a surface is deformed by following underlying physical laws which make the created shapes more realistic, as shown in Fig. 12. Second, since the deformed shape is restricted by the boundary shape of a deformation region, a good method should create the deformation which keeps the features of the boundary shape. As shown in Fig. 11, the deformation created by our proposed method well keeps the features of the boundary shapes. Third, our method is more intuitive because we directly operate the vertices of a target region that need to be deformed. Fourth, our method has a low computational cost since we develop an approximate analytical solution of physics-based deformations rather than a computationally extensive finite element solution. Fifth, our method always maintains C^1 continuity at the boundary of a deformation region since the continuities of the position function and first derivatives with respect to the parametric variables u and v are satisfied as discussed in Sect. 3.3. The main contributions of our work are listed below:

- A novel and efficient physics-based surface manipulation method with C^1 continuity, which defines physics-based deformations with a fourth-order PDE involving a sculpting force to achieve physics-based deformation and C^1 continuity.
- A mapping method, which maps a deformation region from a 3D space to a 2D parametric plane to simplify the resolution of the PDE originating from the bending deformation of a thin elastic plate.
- An interactive user interface of our proposed surface manipulation method, which has been integrated into the software package Maya as a plug-in and can be used to achieve physics-based surface manipulation in deformation regions with an arbitrarily complicated boundary shape in real time.

The remaining parts of this paper are organized as follows. The related works on purely geometric and physics-based surface manipulation methods and PDE-based modelling methods are briefly reviewed in Sect. 2. The mathematical model and solution of our proposed method are described in Sect. 3. The user interface is developed in Sect. 4 and some experiment results of our method and several comparisons with other methods are presented in Sect. 5, and finally the conclusion is drawn in Sect. 6.

2 Related work

The work presented in this paper is related to purely geometric and physics-based surface manipulation methods and PDE-based modelling methods. In this section, we briefly review the most related work in these three fields.

2.1 Purely geometric surface manipulation

There is rich literature on the topic of purely geometric surface manipulation. Here, we briefly introduce some studies which have mature practical applications.

One of the most representative purely geometric surface manipulation methods is FFD, the idea of which is first introduced by Barr [10] and then further developed by Sederberg and Parry [3]. This method embeds an object in a lattice and achieves the deformations of the object by deforming the lattice. By using the initial lattice points to define an arbitrary trivariate Bézier volume, and allowing the combination of many lattices to form arbitrarily shaped spaces, Coquillart [4] introduced extended free-form deformations. In order to provide a better control of the deformation and a more intuitive interface, Hsu [11] proposed a direct manipulation method of FFD. Based on FFD technique, several space deformation models such as rational FFD [12], NURBS-based FFD [13], volume-preserving FFD [14], T-spline FFD [15] and Proxy-driven FFD [16] have been developed. Although FFD is very popular and supported in many 3D modelling software packages such as 3DS Max and Maya, it has several drawbacks. For example, in order to achieve small and local deformation, the lattice grids need to be subdivided so that one lattice point can control the target deformation region without impact other regions. However, the subdivision will result in crowded or even messy lattice grids which not only block the view of the deformed shape but also cause inconvenience for interactive manipulation [16]. In addition, the FFD method is difficult to achieve exact shape deformation because the deformed shape does not follow the lattice points exactly, so that it is unclear which lattice points should be moved and how transformation will affect the deformation of the model [1].

The Laplacian coordinate is another successful surface deformation technique and a variant of [6]. It has been integrated into the software Blender. The potential of the Laplacian coordinate for local mesh morphing and deformation is introduced by Alexa [17]. By solving a linear least squares system, Lipman et al. [5] reconstructed the surface from discrete Laplacians of the mesh functions and spatial boundary conditions. In order to make Laplacian coordinates invariant to rotation and isotropic scaling, Sorkine et al. [6] proposed a Laplacian surface editing method which implicitly transforms the differential coordinates. Based on the idea of the Laplacian coordinate, Zhou et al. [18] used the volumetric graph Laplacian to solve the problem of large deformations. Since the Laplacian method needs to define anchor vertices first and then move some of them to achieve the deformation of non-anchor vertices, it is not intuitive and convenient to deform complex shapes, e. g., multiple deformations within a surface, as shown in Fig. 1b.

Unlike FFD and Laplacian methods that deform the surface by adding extra lattice points or anchors, the Delta Mush method [19] is to directly manipulate the polygonal meshes by moving mesh vertices. Its basic idea is to smooth the deformed shapes of the polygonal meshes. Delta Mush is also regarded as a surface deformation method and widely applied in 3D software packages such as Maya and Houdini.

As mentioned in last section, purely geometric surface manipulation methods do not consider any underlying physical laws so that the quality and aesthetics of deformed shapes mainly depend on the users' perception and skills.

2.2 Physics-based surface manipulation

Physics-based methods are to deform surface shapes by incorporating physical characteristics such as forces, torques and strain energies. They have been widely embraced by the computer graphics community [20]. Terzopoulos et al. [21,22] introduced dynamic differential equations for flexible materials such as rubber, cloth and paper by employing elasticity theory. After that, they extended their work from elasticity to viscoelasticity, plasticity and fracture [23]. By minimizing an energy function with user controlled geometric constraints and loads, Celniker and Gossard [24] developed a curve and surface finite element method for free-form shape design. Güdükbay and Özgüç [25] described a physically based modelling system based on a primal formulation and a hybrid formulation derived from elasticity theory. By using the theory of plate bending in elasticity, You et al. [26] developed a mathematical model of physically based elastic deformations. Considering non-homogeneous material properties and conducting finite element simulations of deformable objects in local frames, McDonnell and Qin [27] presented a modelling technique for physically based deformation.

Involving physics laws in traditional control point-based deformation methods is also a popular research topic. Terzopoulos and Qin [9] developed a dynamic NURBS to deal with mass distributions, internal deformation energies, and other physical quantities of the shape manipulation of NURBS. After that, they further investigated the surfaces with symmetries and topological variability and developed a dynamic NURBS swung surface [7]. By enhancing the power of triangular spline models and using Lagrangian mechanics, Qin and Terzopoulos [8] developed the dynamic triangular NURBS and manipulated the surfaces defined over arbitrary and nonrectangular domains through the finite element solution of its mathematical model. Applying sculpting forces on a surface and formulating and minimizing the energy functional of the surface, Vassilev [28] proposed a method to manipulate deformable B-spline surfaces.

Since physics-based methods are difficult to find analytical solutions, numerical methods such as the finite element method, are commonly used to resolve the mathematical models in the above studies. Therefore, physics-based methods usually require a high computational cost and long computing time, which do not meet the demand of real-time surface manipulation.

2.3 PDE-based modelling

PDEs have been widely used to describe various physical phenomena, and the shape deformations using PDEs can be regarded as physics-based technique [29]. PDE-based modelling is to describe surfaces by resolving a PDE subjected to a set of suitably defined boundary conditions, and it is a powerful technique to create and manipulate 3D models. It was first introduced by Bloor and Wilson [30]. They proposed a fourth-order PDE with a shape control parameter to create different surface shapes. After that, this fourth-order PDE was adopted to deal with a number of modelling problems such as interactive surface designs in real time [31], local surface shape controls [32] and dynamical manipulation of PDE surfaces [33]. In order to incorporate dynamic effects into a fourth-order PDE, You and Zhang [34] presented deformable moving surfaces, and they developed a sixth-order PDE with four shape control parameters which provides more degrees of freedom to manipulate surface shapes [35]. The methods of solving PDEs mainly include numerical and analytical ones. Since it is difficult to find analytical solutions of PDEs with high orders or complex boundary conditions, analytical solutions can only be used to solve simple surface modelling problems [36,37]. Complex surface modelling usually applies numerical methods such as the finite element method [38] and the finite difference method [39,40]. However, numerical methods involve expensive computing cost and are less ideal in real-time modelling applications.

In this paper, by introducing the PDE of plate bending deformation for manipulating surface shapes, we develop an approximate analytical solution for physics-based surface deformations with a low computational cost.

3 Our method

3.1 Theory of plate bending

The deformations of a surface can be simulated through those of elastic bending of a thin plate. When subjected to a lateral load q, the bending deformation of the plate in the xy plane can be described with the following fourth-order partial differential equation [41]

$$D\nabla^4 w = q \tag{1}$$

where w is the deflection of the plate in the z direction, and the symbol ∇^4 is a biharmonic differential operator defined by the following equation

$$\nabla^4 w = \frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} \tag{2}$$

and the symbol D is called the bending rigidity which is defined by

$$D = \frac{Eh^3}{12(1-\mu^2)} \tag{3}$$

In Eq. (3), E and μ are Young's modulus and Poisson's ratio of the plate, respectively, which are two material properties of the plate and reflect the capacity of the plate against bending deformations. The parameter h is the thickness of the plate.

Besides the applied load, material properties and geometric parameters of the plate, and the boundary constraints of the plate also affect its bending deformations. In surface modelling applications, usually, positional and tangential continuities are required. Therefore, in this paper, we will only consider the boundary constraints which maintain such continuities.

Assuming that the boundary of the plate is indicated by $\partial\Omega$, the fixed support boundary constraints in the plate bending can be written as

$$w = 0,\ \frac{\partial w}{\partial x} = 0,\ \frac{\partial w}{\partial y} = 0 \text{ on } \partial\Omega \tag{4}$$

Solving Eq. (1) subjected to boundary constraints (4), the bending deflection w for each point (x, y) within the plate is determined by resolving the function of the geometric position (x, y), i.e. $w = f(x, y)$.

3.2 Mathematical model

Based on the theory of plate bending, we can develop our mathematical model of the surface deformation. The two variables x, y and the deflection w in Eq. (1) form a three-dimensional space. When using a parametric representation to describe a three-dimensional surface, two parametric variables u and v and each component of the coordinate variables x, y and z also form a three-dimensional space. If we define the relationship between each component of coordinate variables x, y and z and the parametric variables u and v with the same function as that of the plate bending, we can determine the deformations of a parametric surface through Eqs. (1–4).

Using the variable ξ to stand for each of x, y and z, the equations governing surface deformations become

$$D_\xi \nabla^4 \xi = q_\xi \ (\xi = x, y, z) \tag{5}$$

where the load q_ξ is called the sculpting force, and

$$\nabla^4 \xi = \frac{\partial^4 \xi}{\partial u^4} + 2\frac{\partial^4 \xi}{\partial u^2 \partial v^2} + \frac{\partial^4 \xi}{\partial v^4} \tag{6}$$

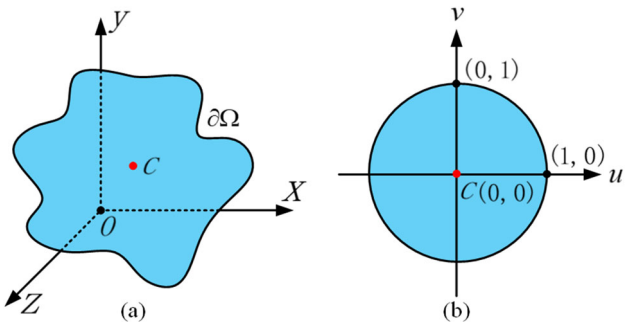


Fig. 2 Boundary of a deformation region. **a** the xyz geometric coordinate system. **b** the uv parametric coordinate system. (Point C is the centroid of the deformation region)

Accordingly, boundary constraints (4) are changed into

$$\xi = 0,\ \frac{\partial \xi}{\partial u} = 0,\ \frac{\partial \xi}{\partial v} = 0 \text{ on } \partial\Omega \tag{7}$$

From boundary constraints (7), we know both the displacements and the rotations of the deformed surface relative to the undeformed surface at the boundary are zero. Therefore, the deformed surface obtained from Eqs. (6) and (7) keeps both positional and tangential continuities at the boundary $\partial\Omega$. We call the continuity defined by Eq. (7) C^1 continuity which is more stringent than positional and tangential continuities on the boundary $\partial\Omega$. In the following subsection, we will discuss how to solve Eq. (5) subjected to boundary constraints (7).

3.3 Solution

It is known from the theory of plate bending that the analytical solution of Eq. (1) subjected to the constraints of an elliptic boundary is obtainable. Since parametric variables u and v are often defined within 0 and 1, i.e. $0 \le u \le 1$ and $0 \le v \le 1$, in the mathematical representation of parametric surfaces, we take the boundary defined by parametric variables u and v to be a unit circle, i.e.

$$u^2 + v^2 - 1 = 0 \tag{8}$$

It should be pointed out that although the boundary defined by parametric variables u and v is a circle, the corresponding boundary in the coordinate system defined by the coordinate variables x, y and z can be a very complicated shape including triangles, rectangles and three-dimensional curves, etc., as shown in Fig. 2.

For the deformation which has both positional and tangential continuity at boundary (8), we assume that the function relationships between each component of coordinate variables x, y and z and parametric variables u and v are

$$\xi = m_\xi (u^2 + v^2 - 1)^2 \ (\xi = x, y, z) \tag{9}$$

where m_ξ is an unknown constant.

On the boundary $u^2 + v^2 - 1 = 0$, the deformation disappears, i.e. $\xi = m_\xi (u^2 + v^2 - 1)^2 = 0$. Therefore, the first of Eq. (7) is satisfied and the positional continuity is guaranteed. In addition, we have

$$\begin{aligned} \frac{\partial \xi}{\partial u} &= 4 m_\xi u (u^2 + v^2 - 1) = 0 \\ \frac{\partial \xi}{\partial v} &= 4 m_\xi v (u^2 + v^2 - 1) = 0 \end{aligned} \tag{10}$$

Equation (10) indicates that the last two of Eq. (7) are also met and the tangential continuity is achieved on boundary $u^2 + v^2 - 1 = 0$.

Substituting Eq. (9) into (5) and solving for the unknown constant m_ξ, we obtain

$$m_\xi = \frac{q_\xi}{64 D_\xi} \tag{11}$$

Introducing Eq. (11) back into Eq. (9), the deformation of the surface within boundary $u^2 + v^2 - 1 = 0$ is found to be

$$\xi = \frac{q_\xi}{64 D_\xi} (u^2 + v^2 - 1)^2 \tag{12}$$

Equation (12) indicates that after applying a sculpting force q_ξ in the region within the boundary $u^2 + v^2 - 1 = 0$, the deformation in the region can be determined analytically.

4 Interaction design

In this section, we develop a user interface based on our proposed surface manipulation method so that users can directly apply our method to interactively manipulate surface shapes. The core algorithm of the user interface, i.e. the deformation algorithm, is introduced in Sect. 3, and the remaining algorithms behind the user interface include the projective transformation, boundary extraction and mapping relationship.

In order to use Eq. (12) to create surface deformations, we must map a deformation region on a 3D surface defined by coordinate variables x, y and z to a circle in a 2D coordinate system defined by parametric variables u and v. However, it is difficult to directly find the mapping relationship between the two spaces. Instead, we adopt three steps to achieve this goal. We first project all vertices within the deformation region to a local 2D plane, and then we extract the boundary curve of the projective deformation region. Finally, we apply a length-based method on the boundary curve to find the mapping

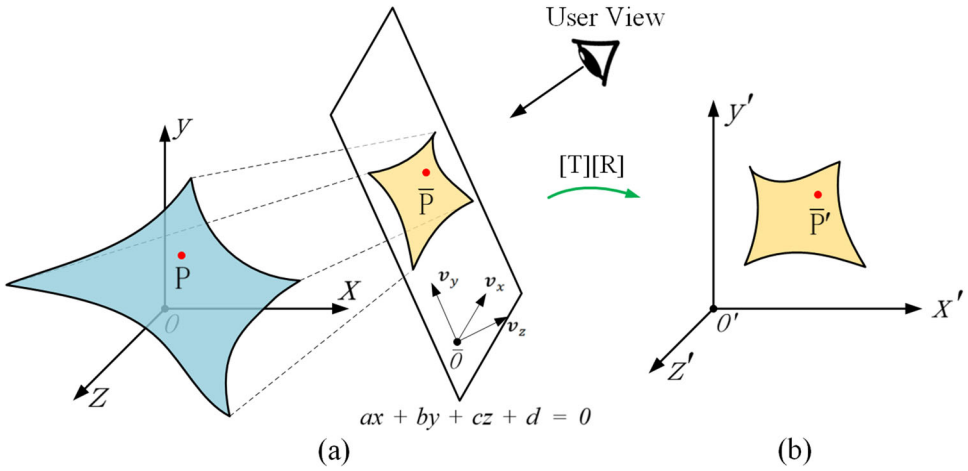


Fig. 3 Projective transformation of the deformation region. **a** the global xyz coordinate system. **b** the local $x'y'z'$ coordinate system

relationship between the projective deformation region and the parametric plane. The details are described in the following subsections.

4.1 Projective transformation

When users select the deformation region on the surface of a 3D model, we assume that the equation of users' view plane is $ax + by + cz + d = 0$, as shown in Fig. 3a. For each vertex $\mathbf{p} = [p_x \ p_y \ p_z]^T$ within the deformation region, we define that its projective vertex on the view plane is $\bar{\mathbf{p}}_1 = [\bar{p}_x \ \bar{p}_y \ \bar{p}_z]^T$ which can be obtained through following projective transformation:

$$\bar{\mathbf{p}}_1 = -\frac{ap_x + bp_x + cp_z + d}{a^2 + b^2 + c^2}\mathbf{n} + \mathbf{p} \tag{13}$$

where $\mathbf{n} = [a \ b \ c]^T$ is the normal vector of the view plane.

The projective vertex $\bar{\mathbf{p}}_1$ is located in the global xyz coordinate system, we transform this coordinate system to a local $x'y'z'$ coordinate system by using a translation matrix $[\mathbf{T}]$ and a rotation matrix $[\mathbf{R}]$, as shown in Fig. 3b. We define that the origin of the local coordinate is $\bar{\mathbf{o}} = [\bar{o}_x \ \bar{o}_y \ \bar{o}_z]^T$ and its unit vectors along x', y' and z' axes are $\mathbf{v}_x = [v_{xx} \ v_{xy} \ v_{xz}]^T$, $\mathbf{v}_y = [v_{yx} \ v_{yy} \ v_{yz}]^T$ and $\mathbf{v}_z = [v_{zx} \ v_{zy} \ v_{zz}]^T$, respectively. The transformation equation can be written as

$$\bar{\mathbf{p}}' = \bar{\mathbf{p}}_2 \, [\mathbf{T}] \, [\mathbf{R}] \tag{14}$$

where $\bar{\mathbf{p}}' = [\bar{p}'_x \ \bar{p}'_y \ \bar{p}'_z \ 1]$ and $\bar{\mathbf{p}}_2 = [\bar{p}_x \ \bar{p}_y \ \bar{p}_z \ 1]$, and

$$[\mathbf{T}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\bar{o}_x & -\bar{o}_y & -\bar{o}_z & 1 \end{bmatrix}, \ [\mathbf{R}] = \begin{bmatrix} v_{xx} & v_{yx} & v_{zx} & 0 \\ v_{xy} & v_{yy} & v_{zy} & 0 \\ v_{xz} & v_{yz} & v_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since all transformed vertices are on the $x'y'$ plane, we obtain the final projective vertex $\bar{\mathbf{p}}' = [\bar{p}'_x \ \bar{p}'_y]$ within the deformation region by removing the z' component.

4.2 Boundary extraction

In order to define the mapping relationship between boundary curve on the $x'y'$ plane and the circle on the 2D parametric plane, the key step is to extract the boundary curve of the deformation region. We adopt the alpha shape (α-shape) method to extract the boundary curve, which is to generate the convex hull of a finite set of points [42]. The boundary extraction process is shown in Fig. 4. Given a set P of vertices $\bar{\mathbf{p}}'_i$ $(i = 1, 2, ..., N)$ from the projective deformation region, we first construct the Voronoi diagram as a set of cells (Fig. 4a), which is defined by

$$V_i = \left\{ \mathbf{p} \in \mathbb{R}^2 | \ \left\| \mathbf{p} - \bar{\mathbf{p}}'_i \right\| \leq \left\| \mathbf{p} - \bar{\mathbf{p}}'_j \right\|, \ \forall j \neq i \right\} \tag{15}$$

where V_i is the locus of the $\mathbf{p}$ closer to $\bar{\mathbf{p}}'_i$ than any other vertices.

Since the Delaunay triangulation is the dual shape of the Voronoi diagram, we can obtain the Delaunay triangulation by connecting all the vertices in P that share common Voronoi faces, as shown in Fig. 4b. Then, by giving the parameter $\alpha = d_P \alpha_0$ where d_P is the distance between two closest vertices in P and α_0 is the threshold value, if the length of any edge of a triangle is larger than 2α, this triangle is removed. After that, we construct the circles of radius α containing two end vertices of edges of the rest triangles. If a circle contains no vertices from P in its interior, this edge is regarded as a valid boundary edge. Finally, all valid boundary edges form the boundary curve, as shown in Fig. 4c.

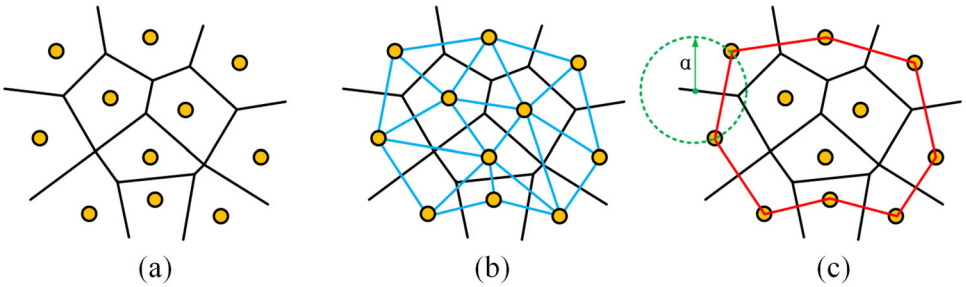


Fig. 4 Extraction of boundary curve. **a** Voronoi diagram for a set of vertices. **b** Delaunay triangulation. **c** The boundary curve of α-shape

4.3 Mapping relationship

The mapping, in computer graphics, is the key to achieve the parametrization [43]. Commonly used mapping methods include minimizing angle distortions (the conformal mapping) and area distortions (the equiareal mapping) [44]. However, in this paper, our goal is to make the surface deformation in the 3D space as similar as possible to the deformation in the 2D parametric space as well as keep the original shape information, rather than reducing the angle or area distortions. In order to achieve this goal, the one-to-one mapping between the projective vertices on the $x'y'$ plane and the parametric points on the parametric plane is necessary and the elastic deformation value of each parametric point needs to be exactly exerted on the corresponding original vertices. Through the one-to-one mapping, the physic-based deformation in the parametric space can be accurately and smoothly reconstructed in the 3D space.

In the parametric space, after generating physic-based deformation within the circle, the deformed shape becomes bigger when moving from the circle to the centre, as shown in Fig 8. For reconstructing the deformation in the 3D space, the distance from the point to the centre of the circle on the parametric plane should be consistent with the distance from the projective vertex to the centroid on the $x'y'$ plane. The radial mapping [44] has the similar idea which makes points only move along radial lines from the centre of the disc. However, this method can only deal with the mapping between a square and a disc, and we have not found any other effective mapping method which can achieve our goal. Here, we propose a simple and effective length-based mapping method to achieve our goal.

The deformation region on the $x'y'$ plane has N vertices. As shown in Fig. 5a, we define that the vertices on the boundary curve are A_i ($i = 1, 2, ... M$) and the rest vertices within the boundary curve are B_j ($j = 1, 2, ..., N - M$). The centroid of the deformation region is C, and the point D_j is the intersecting point between the line $\overline{A_iA_{i+1}}$ and the extended line of $\overline{CB_j}$.

The corresponding points $\tilde{A}_i$, $\tilde{B}_j$, $\tilde{C}$ and $\tilde{D}_j$ of A_i, B_j, C and D_j on uv plane are shown in Fig. 5b. All the points on the boundary curves can be mapped to the unit circle with

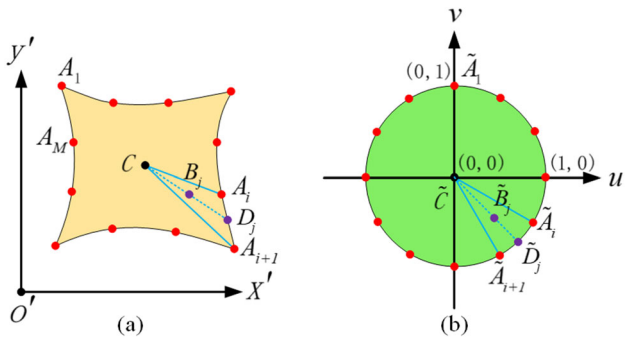


Fig. 5 Mapping relationship. **a** The $x'y'$ plane. **b** The uv plane

the method below. Here, we take the point $\tilde{D}_j$ as an example to discuss how to determine its position on the unit circle.

The total arc length of the boundary curve is $L_{sum} = L_{A_1A_2} + ... + L_{A_iA_{i+1}} + ... + L_{A_MA_1}$. The arc length from the starting point A_1 to the point D_j is $L_{A_1D_j} = L_{A_1A_2} + L_{A_2A_3} + ... + L_{A_iD_j}$. The perimeter of the unit circle is $L_{circle} = 2\pi$. And the arc length from the starting point $\tilde{A}_1$ to the point $\tilde{D}_j$ is assumed to be $L_{\tilde{A}_1\tilde{D}_j}$. Since $L_{A_1D_j}/L_{sum}$ should be equal to $L_{\tilde{A}_1\tilde{D}_j}/(2\pi)$, we have $L_{\tilde{A}_1\tilde{D}_j} = 2\pi L_{A_1D_j}/L_{sum}$.

Having determined the position of the point $\tilde{D}_j$ on the circle, we can calculate its parametric values $u_{\tilde{D}_j}$ and $v_{\tilde{D}_j}$. The parametric values of the point $\tilde{C}$ are $u_{\tilde{C}} = 0$ and $v_{\tilde{C}} = 0$. The coordinate values x'_{D_j} and y'_{D_j} of the point D_j on the boundary curve is determined by the intersection between the line $\overline{A_iA_{i+1}}$ and the extended line of $\overline{CB_j}$. The coordinate values x'_C and y'_C of the point C and the coordinate values x'_{B_j} and y'_{B_j} of the point B_j are known. Assuming the parametric values of the point $\tilde{B}_j$ are $u_{\tilde{B}_j}$ and $v_{\tilde{B}_j}$, we have $(u_{\tilde{B}_j} - u_{\tilde{C}})/(u_{\tilde{D}_j} - u_{\tilde{C}}) = (x'_{B_j} - x'_C)/(x'_{D_j} - x'_C)$ and $(v_{\tilde{B}_j} - v_{\tilde{C}})/(v_{\tilde{D}_j} - v_{\tilde{C}}) = (y'_{B_j} - y'_C)/(y'_{D_j} - y'_C)$, which give

$$\begin{aligned} u_{\tilde{B}_j} &= u_{\tilde{D}_j}\frac{x'_{B_j} - x'_C}{x'_{D_j} - x'_C} \\ v_{\tilde{B}_j} &= v_{\tilde{D}_j}\frac{y'_{B_j} - y'_C}{y'_{D_j} - y'_C} \end{aligned} \tag{16}$$

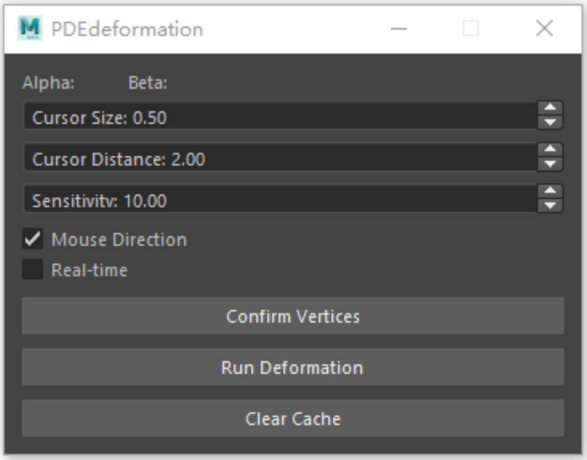


Fig. 6 User interface

Substituting Eq. (16) into (12), we can calculate the deformation values of the original vertex of B_j in the global xyz coordinate system, and finally obtain a new surface shape after solving all vertices within the boundary curve.

4.4 User interface

By integrating the above algorithms, we develop a user interface as a plug-in of the popular 3D modelling software Maya, as shown in Fig. 6. Users can select their desired deformation regions on the surface of a 3D model and generate new shapes using the interface in Maya.

Since directly inputting the value of the sculpting force to manipulate surface shapes is not intuitive for users, we apply a cursor (a visual cube) to achieve deformations, as shown in Fig. 7. In order to control the deformation direction, we only need to control the moving direction of the cursor. There are two ways to control the cursor. First, we can move the cursor by using the move tool in Maya. Once the move tool is activated, the cursor can be directly dragged on the current view plane of the user. Second, we can input moving values in x, y and z directions on the attributes editor of the cursor in Maya. In Fig. 6, there are three input parameters for the cursor on the interface, i.e. cursor size, cursor distance and sensitivity. Cursor size is to control the size of the cursor and avoid that the cursor is much larger or smaller than the target object. Cursor distance is a parameter to change the initial position of the cursor and sensitivity is a scale factor to control the magnitude of the sculpting force. The sculpting force in Eq. (12) can be obtained by $q_\xi = sd_\xi$ ($\xi = x,\ y,\ z$) where d_ξ is the moving vector of the cursor and s is the sensitivity. Figure 8 illustrates the influence of the moving direction and sensitivity of the cursor on the surface shape.

Except the three input parameters, we also provide two options, i.e. mouse direction and real-time, on the user interface. (1) Mouse Direction: its function is to make the deformation direction the same as the moving direction of the cursor. If it is turned off, the deformation direction is the normal of the selected region. (2) Real-time: since the user's view cannot be moved or rotated during moving the cursor, sometimes users have difficulties to estimate whether the cursor arrives at the desired position. Thus, we provide the option of real time to turn off the real-time function so that the shape cannot be deformed in real time unless clicking the Run Deformation button.

As shown in Fig. 7, the surface manipulation process through the user interface has six steps. (1) Users select the deformation region by using the lasso tool to draw freeform curves around target vertices. (2) The selected vertices are activated with yellow colour. (3) After inputting three parameters of the cursor, users click the Confirm Vertices button and generate the cursor. (4) Users can move the cursor to any position and the deformation will follow the moving direction if the option of Mouse Direction is ticked. Otherwise, the deformation direction is in the normal direction. (5) If the option of real time is ticked, the surface shape is automatically deformed in the fourth step once the cursor moves. Otherwise, users need to click the Run Deformation button to deform the surface shape. (6) After obtaining the final new surface shape or in any previous steps, users can click the Clear Cache button to stop this manipulation process.

5 Results and comparison

In the user interface, the basic parameters of Eq. (3) are taken to be: the material properties $E = 10^6$ and $v = 0.5$ according to the material rubber, and the geometric thickness $h = 0.1$. The threshold value α_0 is set to 0.86 because this value is applicable for accurately extracting the boundary curve of different deformation regions by trial and error. We test the user interface on four basic 3D models in Maya, i.e. the polygon sphere, cube, cylinder and cone, with different deformation regions, as shown in Fig. 9. We obtain different deformations on each model by moving the cursor with different directions and positions to generate various shapes. These results indicate that our method is effective and convenient to create various surface shapes.

The meshes of above basic 3D models are smooth and uniformly distributed. In order to further test our method, we use scanned 3D models such as Stanford Bunny, Dragon and Nefertiti as our deformation objects. These scanned 3D models are frequently used as test models in the computer graphics community and have non-uniformly distributed vertices. Figure 10 shows the test results. They indicate that our method is also suitable to manipulate the surface shape of scanned 3D models.

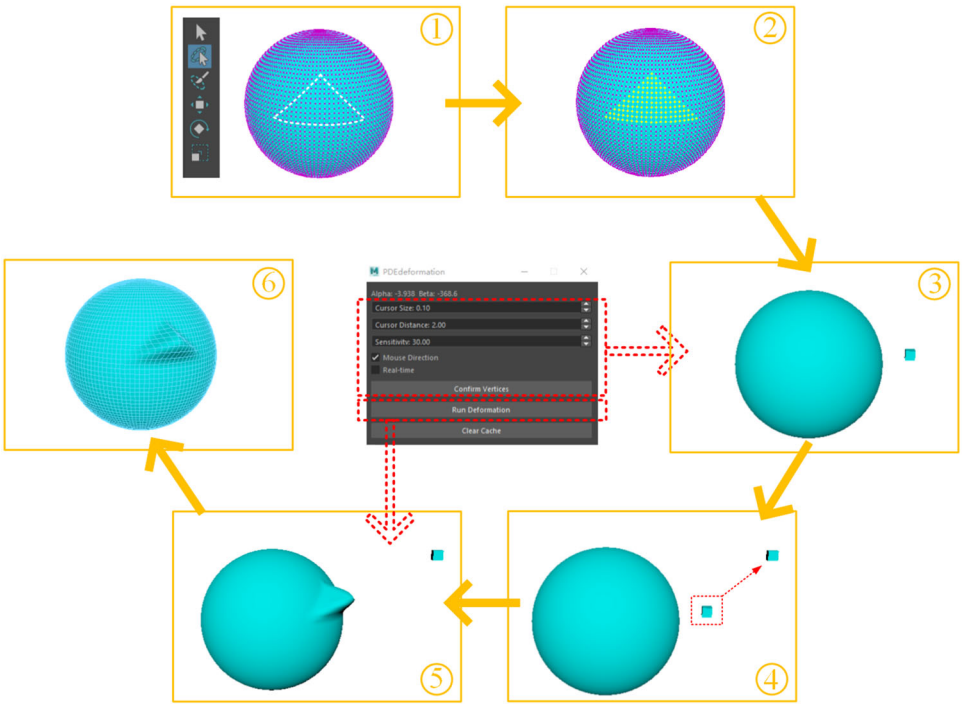


Fig. 7 Surface manipulation process of the user interface

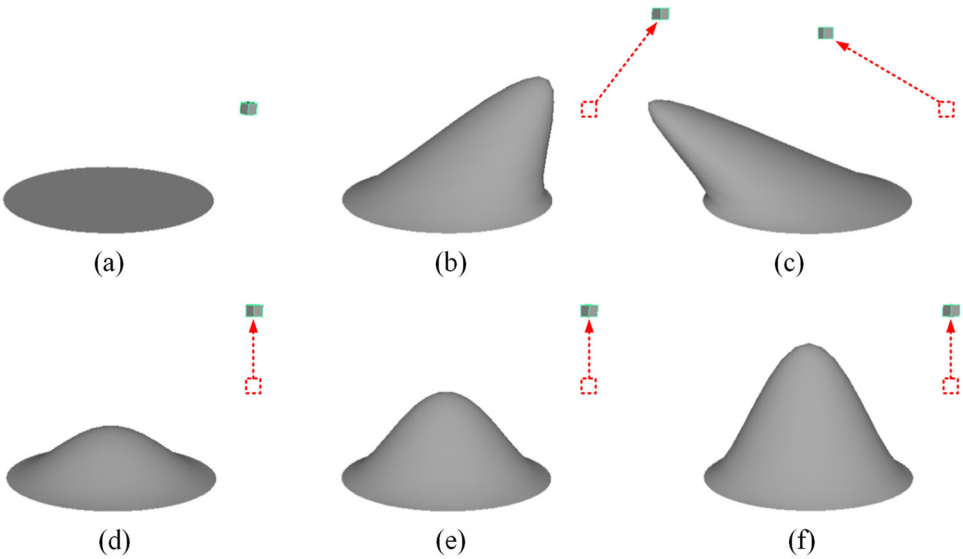


Fig. 8 Influence of the moving direction and sensitivity of the cursor on the surface shape. **a** The deformation region and the initial position of the cursor. **b**, **c** The deformed shapes with the same sensitivity and different moving directions (the red dotted arrow) of the cursor (the red dotted square is the initial position of the cursor). **d**–**f** The deformed shapes with the same moving direction and different sensitivities of the cursor. (The sensitivities are 40, 70 and 110 in **d**, **e** and **f**, respectively)

In order to demonstrate the effectiveness of our method, we compare our method with commonly used deformation methods, i.e. Delta Mush [19] and Laplacian Deformer [6]. Two examples of deforming the shapes of the square and the pentagram along the normal direction are given in Fig. 11. The first and third rows present deformed shapes and the second and fourth rows are contour maps of the deformed shapes in the first and third rows, respectively. As shown in Fig. 11, the deformed shape using our method is strictly constrained by the boundary shape compared with other methods. For example, the shapes of the contour lines with different heights of the square using our method stay the same as the boundary, while contour lines using Delta Mush and Laplacian Deformer gradually become a circle. Therefore, compared with Delta Mush and Laplacian Deformer, our method is better at creating the surface shape which well keeps the features of the boundary shape.

Since our method is physics-based, it can create more realistic shapes compared with purely geometric methods. The finite element analysis (FEA) is the most accurate and popular numerical method widely applied in scientific research and engineering calculations. In particular, it has been widely used to accurately predict elastic and inelastic deformations of various objects and structures in engineering fields. There-

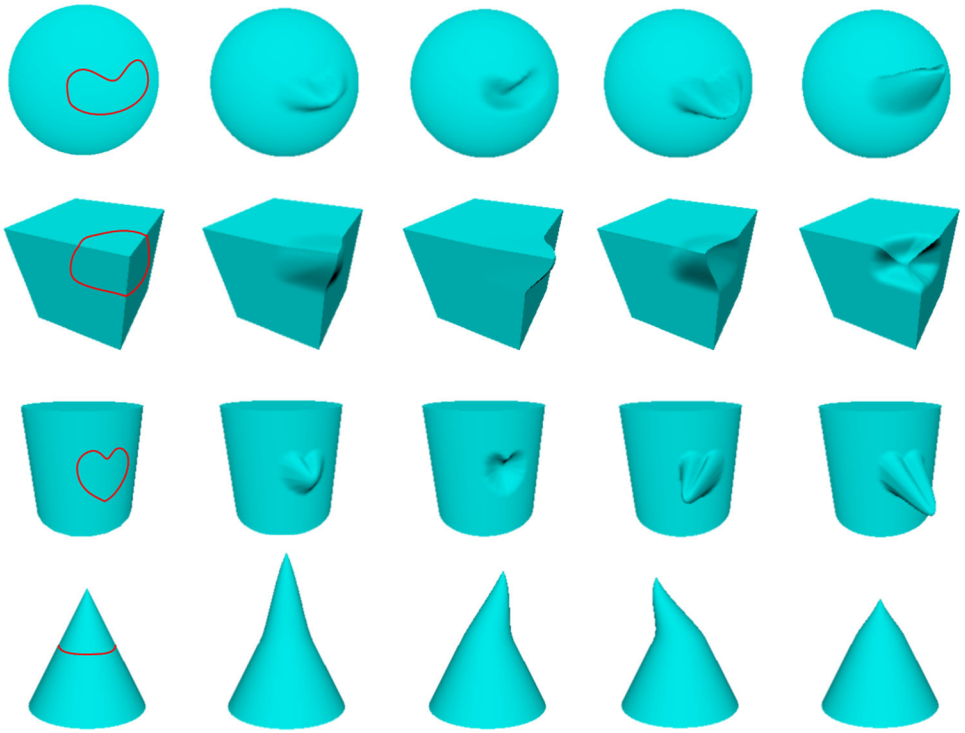


Fig. 9 Test results of four basic 3D models with different deformation regions surrounded by red curves

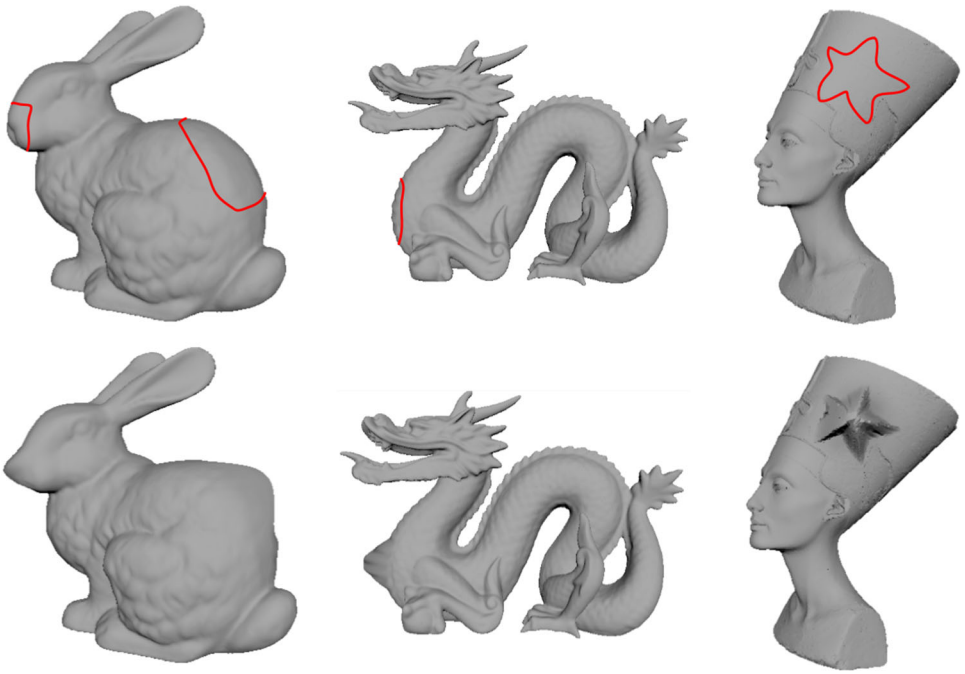


Fig. 10 Test results of three scanned 3D models with different deformation regions surrounded by red curves

fore, we apply FEA to obtain deformed shapes as ground truth and compared them with our method, Delta Mush and Laplacian Deformer in Fig. 12. The left and right columns in the figure present the side view of deformed shapes of a triangle and a rectangle, respectively. The results indicate that the deformed shapes using our method is the closest to the ground truth ones and thus most realistic among the three deformation methods used to obtain the deformed shapes shown in the figure. In addition, in order to manipulate the shape as similar as possible to the ground truth, Delta Mush needs to try different values of two parameters, i.e. the number of smoothing iterations and step lengths, which cause extra time, and Laplacian Deformer needs to find appropriate anchor vertices because different anchor vertices may lead to different shapes. In contrast, our method can directly

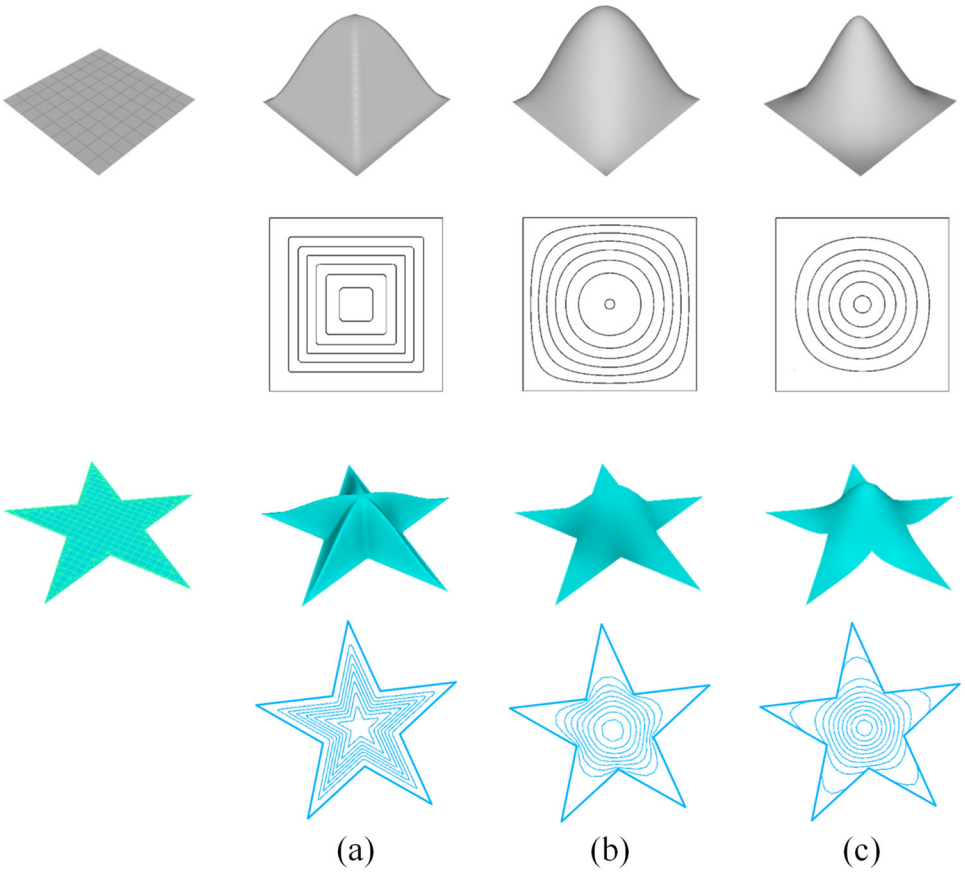


Fig. 11 Comparison between different deformation methods. **a** Our method. **b** Delta Mush [19]. **c** Laplacian Deformer [6]. (The first and third rows are deformed shapes, and the second and fourth rows are contour maps)

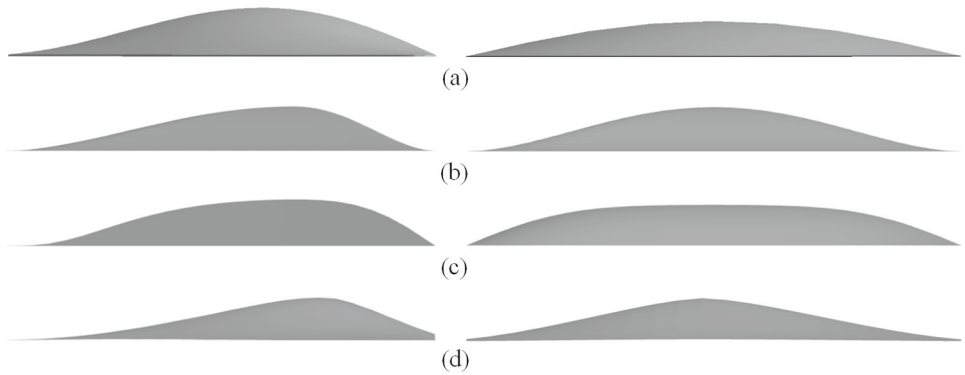


Fig. 12 Realism comparison of deformed shapes in the side view. **a** Ground truth. **b** Our method. **c** Delta Mush [19]. **d** Laplacian Deformer [6]. (The left and right columns are deformed shapes of a triangle and a rectangle, respectively)

create realistic deformed shapes once the deformation region is selected.

6 Conclusion

In this paper, we have obtained a simple approximate analytical solution of the PDE defining the underlying physics of surface deformations, presented a physics-based deformation method, and used it to develop a user interface as a plug-in of the 3D modelling software Maya to interactively manipulate surface shapes of 3D models with C^1 continuity in real time. We have demonstrated the validity of our method by testing different surface deformations on several 3D models. Compared with purely geometric methods, our method is stricter to constrain the deformed shape according to boundary shapes and can generate more realistic shapes.

Our method also has limitations. First, since the method needs to project the surface from a 3D space to a 2D plane, we can only manipulate the surfaces without overlapping parts. Simple overlapping parts within the deformation region can be avoided through changing the user view before confirming vertices. Second, our method cannot address the deformation region if its centroid is located outside of its boundaries.

One way to solve this limitation is to divide the deformation region into several sub-regions and then construct the mapping relationship of each sub-region independently, which requires further investigation.

Acknowledgements This research is supported by the PDE-GIR project which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 778035.

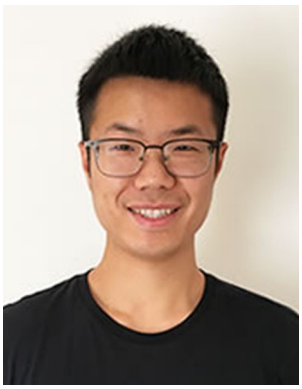
Declarations

Compliance with ethical standards The authors declare that they have no conflict of interest.

References

1. Zheng, J.M., Chan, K.W., Gibson, I.: Constrained deformation of freeform surfaces using surface features for interactive design. Int. J. Adv. Manuf. Technol. **22**(1), 54–67 (2003)
2. Piegl, L., Tiller, W.: The NURBS Book. Springer, Berlin (2012)
3. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, pp. 151–160 (1986)
4. Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, pp. 187–196 (1990)
5. Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rossi, C., Seidel, Hans-Peter.: Differential coordinates for interactive mesh editing. In Proceedings Shape Modeling Applications, pp. 181–190. IEEE (2004)
6. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H-P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 175–184 (2004)
7. Qin, H., Terzopoulos, D.: Dynamic nurbs swung surfaces for physics-based shape design. Comput. Aided Des. **27**(2), 111–127 (1995)
8. Qin, H., Terzopoulos, D.: Triangular nurbs and their dynamic generalizations. Comput. Aided Geom. Des. **14**(4), 325–347 (1997)
9. Terzopoulos, D., Qin, H.: Dynamic nurbs with geometric constraints for interactive sculpting. ACM Trans. Graph. **13**(2), 103–136 (1994)
10. Barr, Alan H.: Global and local deformations of solid primitives. In: Readings in Computer Vision. Elsevier, pp. 661–670 (1987)
11. Hsu, W.M., Hughes, J.F., Kaufman, H.: Direct manipulation of free-form deformations. ACM Siggraph Comput. Graph. **26**(2), 177–184 (1992)
12. Kalra, P., Mangili, A., Thalmann, N. M., Thalmann, D.: Simulation of facial muscle actions based on rational free form deformations. In: Computer Graphics Forum, vol. 11, pp. 59–69. Wiley Online Library (1992)
13. Lamousin, H.J., Waggenspack, N.N.: Nurbs-based free-form deformations. IEEE Comput. Graph. Appl. **14**(6), 59–65 (1994)
14. Hirota, G., Maheshwari, R., Lin, M.C.: Fast volume-preserving free-form deformation using multi-level optimization. Comput. Aided Des. **32**(8), 499–512 (2000)
15. Song, W., Yang, X.: Free-form deformation with weighted t-spline. Vis. Comput. **21**(3), 139–151 (2005)
16. Zhang, Y., Zheng, J., Cai, Y.: Proxy-driven free-form deformation by topology-adjustable control lattice. Comput. Graph. **89**, 167–177 (2020)
17. Alexa, M.: Differential coordinates for local mesh morphing and deformation. Visual Comput. **19**(2), 105–114 (2003)
18. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., and Shum, H-Y.: Large mesh deformation using the volumetric graph laplacian. In: ACM SIGGRAPH 2005 Papers, pp 496–503 (2005)
19. Mancewicz, J., Derksen, M. L. , Rijpkema, H., Wilson, C. A.: Delta mush: smoothing deformations while preserving detail. In: Proceedings of the Fourth Symposium on Digital Production, pp. 7–11 (2014)
20. Nealen, A., Müller, M., Keiser, R., Boxerman, E., Carlson, M.: Physically based deformable models in computer graphics. In: Computer graphics forum, vol. 25, pp. 809–836. Wiley Online Library (2006)
21. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 205–214 (1987)
22. Terzopoulos, D., Fleischer, K.: Deformable models. Vis. Comput. **4**(6), 306–331 (1988)
23. Terzopoulos, Demetri., and Fleischer, Kurt.: Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, pp. 269–278 (1988)
24. Celniker, G., Gossard, D.: Deformable curve and surface finite-elements for free-form shape design. In: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, pp. 257–266 (1991)
25. Güdükbay U., Özgüç, B.: Animation of deformable models. Comput.-Aided Des. **26**(12), 868–875 (1994)
26. You, L.H., Rodriguez, Javier Romero., and Zhang, Jian J.: Manipulation of elastically deformable surfaces through maya plug-in. In: Geometric Modeling and Imaging–New Trends (GMAI'06), pp 15–21. IEEE (2006)
27. McDonnell, K.T., Qin, H.: A novel framework for physically based sculpting and animation of free-form solids. Vis. Comput. **23**(4), 285–296 (2007)
28. Vassilev, T.I.: Interactive sculpting with deformable nonuniform b-splines. Comput. Graph. Forum **16**, 191–199 (1997)
29. You, L., Yang, X., Pan, J., Lee, T.-Y., Bian, S., Qian, K., Habib, Z., Sargano, A.B., Kazmi, I., Zhang, J.: Fast character modeling with sketch-based PDE surfaces. J. Multimedia Tools Appl. **79**, 23161–23187 (2020)
30. Bloor, M.I.G., Wilson, M.J.: Using partial differential equations to generate free-form surfaces. Comput. Aided Des. **22**(4), 202–212 (1990)

31. Ugail, H., Bloor, M.I.G., Wilson, M.J.: Techniques for interactive design using the PDE method. ACM Trans. Graph. **18**(2), 195–212 (1999)
32. Bloor, M.I.G., Wilson, M.J.: Local control of surfaces generated using partial differential equations. Comput. Graph. **18**(2), 161–169 (1994)
33. Du, H., Qin, H.: Dynamic PDE surfaces with flexible and general geometric constraints. In: Proceedings the Eighth Pacific Conference on Computer Graphics and Applications. IEEE, pp. 213–447 (2000)
34. You, L., Zhang, J.J.: Fast generation of 3-d deformable moving surfaces. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **33**(4), 616–625 (2003)
35. Zhang, J.J., You., L.H.: Fast surface modelling using a 6th order PDE. In: Computer Graphics Forum, vol. 23, pp. 311–320 (2004)
36. Zhang, J.J., You, L.: PDE based surface representation-vase design. Comput. Graph **26**(1), 89–98 (2002)
37. Lowe, T.W., Bloor, M.I.G., Wilson, M.J.: Functionality in blend design. Comput.-Aided Des. **22**(10), 655–665 (1990)
38. Brown, J.M., Bloor, M.I.G., Bloor, M.S., Wilson, M.J.: The accuracy of b-spline finite element approximations to PDE surfaces. Comput. Methods Appl. Mech. Eng. **158**(3–4), 221–234 (1998)
39. Du, H., Qin, H.: Dynamic PDE-based surface design using geometric and physical constraints. Gr. Models **67**(1), 43–71 (2005)
40. Wang, S., Xia, Y., Wang, R., You, L., Zhang, J.: Optimal nurbs conversion of PDE surface-represented high-speed train heads. Optim. Eng. **20**(3), 907–928 (2019)
41. Timoshenko, S.P., Woinowsky-Krieger, S.: Theory of Plates and Shells. McGraw-hill, New York (1959)
42. Edelsbrunner, H., Kirkpatrick, D., Seidel, R.: On the shape of a set of points in the plane. IEEE Trans. Inf. Theory **29**(4), 551–559 (1983)
43. Sheffer, A., Praun, E., Rose, K.: Mesh parameterization methods and their applications. Found. Trends® Comput. Graph. Vis. **2**(2), 105–171 (2006)
44. Fong, C.: Analytical methods for squaring the disc. arXiv preprint arXiv:1509.06344 (2015)



Shuangbu Wang is currently a PhD candidate at National Centre for Computer Animation, Bournemouth University, UK. He received his BS degree in Mechanical Design and Manufacturing and Automation and MS degree in Mechanical Engineering from Chongqing University (China) in 2014 and 2017, respectively. His research interests include computer-aided design, computer graphics, geometric modelling, image processing and deep learning.

Nan Xiang received his BS degree from the School of Software, Nanchang University, China, in 2012. MA degree from School of Animation and Digital Arts, Communication University of China, in 2017. He is currently a PhD student at the National Centre for Computer Animation, Bournemouth University, UK. His research interests include virtual reality, 3D reconstruction and deep learning.

Yu Xia is currently a PhD candidate at National Centre for Computer Animation, Bournemouth University, UK. She received her BS degree in Mechatronics Engineering and MS degree in Mechanical Engineering from Chongqing University (China) in 2014 and 2017, respectively. Her research interests include sketch-based image and 3D shape retrieval, computer-aided design and deep learning.

Lihua You is Professor at the National Centre for Computer Animation, Bournemouth University, UK. He received his Ph.D. degree from Chongqing University, China, and another Ph.D. degree from the National Centre for Computer Animation, Bournemouth University, UK. His current research interests are in computer graphics, computer animation, geometric modelling, and 3D printing.

Jianjun Zhang is currently a Professor of Computer Graphics at the National Centre for Computer Animation, Bournemouth University, and leads the Computer Animation Research Centre. His research focuses on a number of topics relating to 3D computer animation, including virtual human modelling and simulation, geometric modelling, motion synthesis, deformation and physics-based animation. He is also interested in virtual reality and medical visualization and simulation. Prof. Zhang has published over 200 peer-reviewed journal and conference publications. He has chaired over 30 international conferences and symposia and serves on a number of editorial boards.