Report of Implementation on ODE-based C2 Continuous Surface Creation Technique

Junheng Fang
s5212191@bournemouth.ac.uk
Bournemouth University

ABSTRACT

Current storage methods of 3D models are mostly based on saving the data of all vertices, which spend massive storage space of hard drives. Besides, plenty of physics-based skin deformation algorithms have been provided to improve the realism of facial animation. Thus, the main purpose of this report is to (1) using ordinary differential equation (ODE)-based method to rebuild the 3D models, (2) blending the surface creation technique with different interpolation methods to calculate the keyframe models in between, and then compare the animations to demonstrate the feasibility to apply it in surface deformation.

1 INTRODUCTION

Recently, the two most significant and interlinked challenges, which limit the development of game production, are geometric modelling and computer animation. As the game industry focuses more and more on scene realism and performance, the game capacity has shown geometric multiplier growth especially after the 3D model techniques have been widely used. For instance, ***Super Mario Bros.***, released in 1985, spent only 64 KB to store the whole game. After over 10 years, ***Super Mario Bros. Deluxe***, released in platform GBC in 1999, took 342 KB, which is only approximately 6 times the size of the former game, remaining the 2D scene. However, when it comes to the 21st century, as the 3D models are introduced into the game industry, the size of ***New Super Mario Bros***, released in platform NDS in 2006, came to 10 MB, which is nearly 30 times compared with the 1999's one. Markedly, the size surged 250 times i.e. 2.5 GB in 2012 released ***New Super Mario Bros. U***, with both the scene and models using 3D techniques.

As is widely agreed, to achieve accurate and fine models and natural animation, rich details are required for good realism, which involves abundant design parameters, i.e. surface vertices of polygon models.

Nevertheless, problems are deservedly brought out when managing those massive data, causing a large increase of the time used in shape manipulation, animation disposing and rendering. Besides, transmitting is also an important issue of online games. As the large data of geometric models or rendered images transmitted among the game players over the Internet, the transmission time is noticeably increased and the real-time performance of game playing is reduced due to the limitation of the communication technology. Apart from the above problems, the size of the current games, with high storage cost, has placed a great burden on the hard drives, like video game ***Call of Duty: Infinite Warfare*** which uses over 130 GB of storage to maintain its game, when the average capacity of the mainstream SSD is only 120 GB. Sufficient example models with good realism and abundant details are required by data-driven techniques based advanced games, which spend heavy human involvement, high cost and cause inefficient game production.

As for animation, if it follows the underlying physics of object movements and deformations, the requirement of high computational resources and much computational time, which would directly affect the performance of real-time games, is inevitable. These are all challenges for techniques related to real-time animation, including modelling and skin deformation.

The widely used surface creation approaches are polygon, patch modellings like subdivision and NURBS. The polygon technique manipulates surface vertices of simple geometric primitives to generate complex models. The patch technique divides the complicated models into plenty of simple patches and after processing them separately, combines them together to fulfil the model. The subdivision technique separates the polygonal faces into smaller pieces by approximating or interpolating schemes to build a denser mesh of the model. However, these techniques require further improvements, for example how to reduce tedious and time-consuming manual operations when stitching separated patches together as well as considering continuity between adjacent ones, how to manipulate the global shape of complicated patches, and how to apply physics of object deformations in the rebuilt model.

Both the academia and industry practices in the animation field regard skin deformation techniques as the standardized approaches when forming real-time animation. Among all the different skin deformation methods, the major characteristics have not been changed, realism and efficiency. To achieve realism, more and more details are required to be stored, i.e. sufficient vertices, which causes plenty of redundancy that is not used often in computation. Physics-based techniques can not avoid numerical computation, which

costs high sources of CPU, and reduces efficiency. Therefore, how to solve the above challenges has become an urgent problem in developing advanced game techniques.

2 LITERATURE REVIEW

For commercial available graphics package, the widely recognized method to build models is polygon modelling and skin surface creation, which could exhibit details and branches, as well as assign UV texture coordinates (Russo 2006). Nevertheless, it is difficult to create curved surfaces. Instead, approximate curved surfaces are used to visualize them, which reduces the precision. Though polygon modelling is more easily to generate hard edges, NURBS modelling could create curved objects with smoother surfaces and more readily edit the patches with fewer control curves (Piegl and Tiller 2012). This advantage makes it welcome in building realistic models. However, stitching adjacent patches requires tedious manual operations to tackle the continuity problem. Another method to build models is subdivision modelling. It first creates a rough polygonal model, and then uses approximating or interpolation to subdivide each face into smaller ones to finally get a denser mesh. It could easily create complicated geometry with more efficiently rendering, whilst the insufficient underlying parametrization makes it hard to elaborate the accuracy. Besides, Várady *et al.* (2012) have declared a method based on in-curved network for further shape controls, which is particularly used to revise the inside of transfinite patches.

For generating more realistic models, physics-based modelling is provided with the physical laws underlying its surface deformation. Nealen *et al.* (2005) have made a review of multifarious physics-based modelling methods, including finite difference method, mass-spring system and reduced deformable models using modal analysis etc.

ODEs, as an important branch of modern mathematics and an effective tool for people to solve various practical problems, have been generally used to represent the underlying physics in the computer graphics field. Take free-fall motion of a bouncing ball as an instance, Newton's second law $F = dP/dt = d(mv)/dt = mdv/dt = ma$ actually uses the differential of velocity v with respect to time t to denote acceleration a, and then times mass m to describe force F. Its introduction could generate physically realistic facades as well as deformations of 3D models. According to the research from NCCA, ODE-based sweeping surfaces (You *et al.* 2007), ODE-based surface deformations(You *et al.* 2010) and ODE-based sweeping blending (Chaudhry *et al.* 2013) have already been developed. ODE-based methods provide users with a higher-level control to manipulate the surfaces via parameters and boundary condition, whilst the traditional surface modelling approaches use a vast number of control points. This achieves an easy implementation of interactive modelling package, whereas it is not easy to get the solution of the corresponding ODE efficiently, forming the most significant issue of these approaches.

Chaudhry *et al.* (2010) have made a review of significant skin deformation techniques. Surface-based techniques are widely used in computer animation field as they are much simpler with still achieving reasonable skin deformation, including purely geometric techniques, which ignore the underlying physics but only manipulate vertices to change the shape, and physics-based techniques, which have more natural performance but heavy computational expenses.

The simplest method of geometric deformation is the linear interpolation, which is always used to emphasis the better results of other methods. However, due to its bad performance, it is barely used in pipelines. Among the other geometric techniques, free form deformation (FFDs), provide by Sederberg and Parry (1986), is popular due to its simplicity and modelling speed, and then was developed into the Lattice and Wrap deformers in Maya. Other methods like joint-related approaches , where the skin is treated as using an explicit function of the skeleton to move the shell, skeleton subspace deformation (Magnenat-Thalmann *et al.* 1988), which introduces vertex weights for smooth transformation of bones, and example-based techniques, which use existing poses as the base to interpolate the keyframes (Allen *et al.* 2002).

The physics-based deformation techniques are based on the anatomy and biomechanics of skin deformation deriving from the action of muscles. It uses the mass-spring system, finite element method (FEM) and finite volume method to determine contractile muscle forces and muscle geometry transformation. Simulation of elastics, elastoplastic fracturing materials, skeleton-driven deformation and physics-based rigging will prefer to use FEM (Debunne *et al.* 2001), while the mass-spring system is more used for interactive animation of deformable models (Platt and Badler 1981). The finite volume method is more used to simulate the deformable movement of skeletal muscles. These methods are all based on partial differential equation (PDE), though they approximate PDEs by different equations. Bian *et al.* (2019) have also provided another PDE-based skin deformation model, which reduces the numerical calculations and creates physically realistic skin deformations with high efficiency by including vertex identification on iso-parametric curves, Fourier series conversion and the analytic solution to a formulated model with underlying physics.

3 ALGORITHM THEORY

Bian *et al.* (2020) have claimed an ODE-based C2 continuous surface creation technique to use the solution to a vector-valued ordinary differential equation with boundary constraints to generate a surface. As shown in Figure 1, it

uses only 6 curves to control the shape of surface patches: two boundary curves, representing the position of the patch, and four control curves, describing the tangent and curvature of each vertex on the boundary curves to calculate the iso-parametric lines.

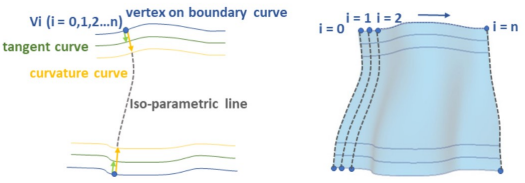


Figure 1: Algorithm Overview

This technique is used as the basic algorithm of this report, which will be discussed further in the following. It contributes mainly to four aspects: (1) Avoiding the manual operation when stitching the adjacent patches together, but achieves the continuities when they are generated. (2) Significantly reducing the data size of 3D models. (3) Much easier to control the global shape of each patch compared with the polygon method. (4) Based on physics, as differential equations characterizing natural physical process are used to control the representing curves of each patch.

This section will explain the mathematical model of surface reconstruction, how to get the closed-form complementation solution and how to achieve continuities between adjacent patches respectively to be used for rebuilding models.

Mathematical Model

The vector-valued mathematical equation $X = S(u, v)$ could represent any 3D parametric surface, where u and v, as coordinates parametric variables, are defined in the interval [0, 1], X is a vector-valued position function with three components in each direction x, y and z. When v is fixed as the constant v_i, the equation could be written as $c_i = S(u, v_i)$, with u controlling one single curve of the surface. Thus, a set of parametric curves c_0, c_1, c_2, ..., with v_i changing from 0 to 1 continuously, form the surface $X = S(u, v)$. That is to say, the modelling of any 3D parametric surface could be separated into a set of parametric curves to deal.

As mentioned above, the continuity problem is significant when creating contiguous patches. Hence, to make two adjacent surface patches achieve the C2 continuities, the same position function, and first and second partial differential with respect to u at their borders, i.e. $u = 0$ and $u = 1$, must be shared. If $c_j(v)(j = 1, 2, 3, 4, 5, 6)$ represents the six function at the joints, the boundary constraints satisfied by a C2 continuous surface patch could be defined as

$$\begin{aligned} u = 0 \quad & S(0, v) = c_1(v) \quad \frac{\partial S(0, v)}{\partial u} = c_2(v) \quad \frac{\partial^2 S(0, v)}{\partial u^2} = c_3(v) \\ u = 1 \quad & S(0, v) = c_4(v) \quad \frac{\partial S(0, v)}{\partial u} = c_5(v) \quad \frac{\partial^2 S(0, v)}{\partial u^2} = c_6(v) \end{aligned} \tag{1}$$

where $c_1(v)$ and $c_4(v)$ stand for position function, and $c_2(v)$, $c_5(v)$ and $c_3(v)$, $c_6(v)$ stand for the first and second partial differential at the boundaries respectively. Thus, determining the mathematical equation of these curves which satisfies Equation (1) at boundaries becomes the aim of surface modelling.

For the parametric curves, NURBS is used due to its smoothness and convenience. Since ODEs are usually applied to represent the underlying physics of curve-like objects deformation like beams and members, the realism of the model appearance could be improved with ODEs introduced. Besides, it is known that to meet the constraints of two position functions, the accurate solution of second order ODEs has two unknown constants, whilst that of fourth order ODEs has four unknown constants to meet the extra two first partial differentials. Fairly, the solution of sixth order ODEs owes six unknown constants as described by Equation (1). Thus, the vector-valued six order ODE is introduced as

$$\frac{\rho d^6 S(u, v_i)}{du^6} + \frac{\eta d^4 S(u, v_i)}{du^4} + \frac{\lambda d^2 S(u, v_i)}{du^2} = F(u) \tag{2}$$

where ρ, η and λ denote shape control parameters, and $F(u)$ stands for the virtual sculping force function also with three components in three directions $F_x(u)$, $F_y(u)$, $F_z(u)$.

The task of generating the curves is then to compute the solution to Equation (2) meeting boundary constraints Equation (1). A particular solution and a complementary solution of the associated homogeneous equation of Equation (2) form the complete solution, whilst only the former is addressed in Bian's work.

First, to create the boundary constraints, the key is to generate two boundary curves $c_1(v)$ and $c_4(v)$ at $u = 0$ and $u = 1$, as well as the other four control curves at $u_2(0 < u_2 < u_3)$, at $u_3(u_2 < u_3 < u_4)$, at $u_5(\hat{u_4} < u_5 < u_6)$, where u_4 and u_4 are two different values, and $\hat{u_6}$ at $u_6(u_5 < u_6 < 1)$. Geometric transformations including translation, scaling and rotation could be applied in boundary curves to generate the control curves.

The next step is to compute the first differential at points $c_1(v_i)$ and for each point on the boundary curve $c_1(v)$ at position v_i by the following forward difference formula

$$c_2(v_i) = \frac{\bar{c}_2(v_i) - c_1(v_i)}{u_2}$$

$$\bar{c}_2'(v_i) = \frac{\bar{c}_3(v_i) - \bar{c}_2(v_i)}{u_3 - u_2}$$

And also, the second differential at the point $c_1(v_i)$,

$$c_3(v_i) = \frac{\bar{c}_2'(v_i) - c_2(v_i)}{u_2} = \frac{\frac{\bar{c}_3(v_i)-\bar{c}_2(v_i)}{u_3-u_2} - \frac{\bar{c}_2(v_i)-c_1(v_i)}{u_2}}{u_2} = \frac{u_2\bar{c}_3(v_i) - u_3\bar{c}_2(v_i) + (u_3-u_2)c_1(v_i)}{{u_2}^2(u_3-u_2)}$$

Then the same method could be used to get the first differential at points $c_4(v_i)$ and $\bar{c}_6(v_i)$ as well as the second differential at point $c_4(v_i)$,

$$c_5(v_i) = \frac{c_4(v_i) - \bar{c}_6(v_i)}{1-u_6}$$

$$\bar{c}_6'(v_i) = \frac{\bar{c}_6(v_i) - \bar{c}_5(v_i)}{u_6-u_5}$$

$$c_6(v_i) = \frac{c_5(v_i) - \bar{c}_6'(v_i)}{1-u_6} = \frac{\frac{c_4(v_i)-\bar{c}_6(v_i)}{1-u_6} - \frac{\bar{c}_6(v_i)-\bar{c}_5(v_i)}{u_6-u_5}}{1-u_6} = \frac{(u_6-u_5)c_4(v_i) + (1-u_6)\bar{c}_5(v_i) - (1-u_6)\bar{c}_6(v_i)}{(1-u_6)^2(u_6-u_5)}$$

As v_i monotonously changes in the interval [0, 1], the tangents and curvature of the boundary are continuous functions of the parametric variable v. Hence, $c_2(v)$, $c_3(v)$, $c_5(v)$ and $c_6(v)$ could be written as

$$c_2(v) = \frac{\bar{c}_2(v_i) - c_1(v_i)}{u_2}$$

$$c_3(v) = \frac{u_2\bar{c}_3(v_i) - u_3\bar{c}_2(v_i) + (u_3-u_2)c_1(v_i)}{{u_2}^2(u_3-u_2)}$$

$$c_5(v) = \frac{c_4(v_i) - \bar{c}_6(v_i)}{1-u_6}$$

$$c_6(v) = \frac{(u_6-u_5)c_4(v_i) + (1-u_6)\bar{c}_5(v_i) - (1-u_6)\bar{c}_6(v_i)}{(1-u_6)^2(u_6-u_5)}$$

With the above representations, surface with constraints Equation (1) could be generated.

Closed Form Complementary Solution

The shape control parameters of Equation (2), ρ, η, λ and the differentials of Equation (1) are used to manipulate the surface generated by the complementary solution of Equation (2). The particular solution allows further deformation control since it involves a sculpting force $F(u)$. However, it is not addressed in this report due to the time limitation. Therefore, Equation (2) could be rewritten as

$$\frac{\rho d^6 S(u,v_i)}{du^6} + \frac{\eta d^4 S(u,v_i)}{du^4} + \frac{\lambda d^2 S(u,v_i)}{du^2} = 0 \tag{3}$$

Introducing $\bar{S}(u,v_i) = d^2S(u,v_i)/du^2$ into Equation (3), the sixth order ODE equation changes into a fourth one,

$$\frac{\rho d^4 \bar{S}(u,v_i)}{du^4} + \frac{\eta d^2 \bar{S}(u,v_i)}{du^2} + \lambda \bar{S}(u,v_i) = 0 \tag{4}$$

Then considering $\bar{S}_\phi(u,v_i) = e^{ru}(\phi = x,y,z)$, Equation (4) could be transformed into an algebra equation,

$$\rho r^4 + \eta r^2 + \lambda = 0 \tag{5}$$

Substituting q with r^2, Equation(5) could be rewritten as a quadratic equation,

$$\rho q^2 + \eta q + \lambda = 0$$

whose roots are (without considering the condition $1 - 4\rho\lambda/\eta^2 < 0$)

$$q_{1,2} = \frac{-\eta(1 \pm \sqrt{1 - \frac{4\rho\lambda}{\eta^2}})}{2\rho}$$

Introducing $\xi_{1,2} = \sqrt{\eta(1 \pm \sqrt{1-4\rho\lambda/\eta^2})/(2\rho)}$ into equation $q = r^2$, the four roots of the quartic Equation (5) are gained,

$$r_{1,2} = \pm i\xi_1 \qquad r_{3,4} = \pm i\xi_2$$

Thus, the complementary solution to Equation (4) could be written as

$$S(u,v_i) = \bar{b}_1 cos\xi_1 u + \bar{b}_2 sin\xi_1 u + \bar{b}_3 cos\xi_2 u + \bar{b}_4 sin\xi_2 u \tag{6}$$

where $\bar{b}_k (k = 1,2,3,4)$ stands for vector-valued unknown constants.

The following solution to the sixth ordered ODE Equation (3) could be obtained by inserting Equation (6) into the second ordered ODE Equation $\bar{S}(u,v(i)) = d^2S(u,v_i)/du^2$,

$$\begin{aligned} S(u,v_i) =& b_1 cos\xi_1 u + b_2 sin\xi_1 u + b_3 cos\xi_2 u \\ &+ b_4 sin\xi_2 u + b_5 u + b_6 \end{aligned} \tag{7}$$

where $b_k (k = 1,2,...,6)$ stands for different constants.

Substituting Equation (7) into Equation (1) and putting the results back into Equation (7), a function of a 3D parametric surface with boundary constraints are achieved,

$$\begin{aligned} S(u,v) =& g_1(u)c_1(v) + g_2(u)c_2(v) + g_3(u)c_3(v) \\ &+ g_4(u)c_4(v) + g_5(u)c_5(v) + g_6(u)c_6(v) \end{aligned} \tag{8}$$

where

$$
\begin{aligned}
g_1(u) =& -d_1 cos\xi_1 - d_4 sin\xi_1 u + d_1(e_{11}+1)cos\xi_2 u \\
&+ e_9 sin\xi_2 u - (\xi_2 e_9 - \xi_1 d_4)u - (e_{11}d_1 - 1) \\
g_2(u) =& -(d_1+d_2)cos\xi_1 u - (d_3+d_4)sin\xi_1 u \\
&+ (e_9+e_{10})sin\xi_2 u - e_{11}(d_1+d_2) \\
&- [\xi_2(e_9+e_{10}) - \xi_1(d_3+d_4) - 1]u \\
g_3(u) =& -d_5 cos\xi_1 u - d_7 sin\xi_1 u + d_{10} cos\xi_2 u \\
&+ (e_5 e_9 + e_6 e_{10} + e_{12}/\xi_2)sin\xi_2 u \\
&- [\xi_2(e_5 e_9 + e_6 e_{10}) - \xi_1 d_7 + e_{12}]u \\
&- (e_{11}d_5 - 1/{\xi_2}^2) \\
g_4(u) =& d_1 cos\xi_1 u + d_4 sin\xi_1 u - d_1(e_{11}+1)cos\xi_2 u \\
&- c_9 sin\xi_2 u + (\xi_2 e_9 - \xi_1 d_4)u + e_{11}d_1 \\
g_5(u) =& d_2 cos\xi_1 u + d_3 sin\xi_1 u - d_2(e_{11}+1)cos\xi_2 u \\
&- e_{10} sin\xi_2 u + (\xi_2 e_{10} - \xi_1 d_3)u + e_{11}d_2 \\
g_6(u) =& -d_6 cos\xi_1 u - d_8 sin\xi_1 u + d_6(e_{11}+1)cos\xi_2 u \\
&+ (e_7 e_9 + e_8 e_{10} - e_{13}/\xi_2)sin\xi_2 u \\
&- [\xi_2(e_7 e_9 + e_8 e_{10}) - \xi_1 d_8 - e_{13}]u - e_{11}d_6
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
d_1 =& e_1/(e_1 e_3 - e_2 e_4) \\
d_2 =& -e_2/(e_1 e_3 - e_2 e_4) \\
d_3 =& e_3/(e_1 e_3 - e_2 e_4) \\
d_4 =& -e_4/(e_1 e_3 - e_2 e_4) \\
d_5 =& d_1 e_5 + d_2 e_6 \\
d_6 =& d_1 e_7 + d_2 e_8 \\
d_7 =& d_4 e_5 + d_3 e_6 \\
d_8 =& d_4 e_7 + d_3 e_8 \\
d_9 =& (e_{11}+1)(d_1+d_2) \\
d_{10} =& -1/{\xi_2}^2 + (e_{11}+1)d_5
\end{aligned}
\tag{10}
$$

and

$$
\begin{aligned}
e_1 =& \xi_1(cos\xi_1 - 1) + {\xi_1}^2 sin\xi_1(1 - cos\xi_2)/(\xi_2 sin\xi_2) \\
e_2 =& sin\xi_1 - \xi_1 + {\xi_1}^2 sin\xi_1(1/sin\xi_2 - 1/\xi_2)/\xi_2 \\
e_3 =& cos\xi_1 - 1 + {\xi_1}^2(1 - cos\xi_2)/{\xi_2}^2 \\
&+ {\xi_1}^2(cos\xi_2 - cos\xi_1)(1/\xi_2 - 1)/sin\xi_2)/\xi_2 \\
e_4 =& \xi_1(-sin\xi_1 + \xi_1 sin\xi_2/\xi_2) \\
&+ {\xi_1}^2(cos\xi_2 - cos\xi_1)(cos\xi_2 - 1)/(\xi_2 sin\xi_2) \\
e_5 =& (1/\xi_2 - cos\xi_2/sin\xi_2)/\xi_2 \\
e_6 =& (sin\xi_2 + cos^2\xi_2/sin\xi_2 - cos\xi_2/sin\xi_2)/\xi_2 \\
e_7 =& (1/sin\xi_2 - 1/\xi_2)/\xi_2 \\
e_8 =& (1 - cos\xi_2)/(\xi_2 - sin\xi_2) \\
e_9 =& {\xi_1}^2[d_4 sin\xi_1 - d_1(cos\xi_2 - cos\xi_1)]/({\xi_2}^2 sin\xi_2) \\
e_{10} =& {\xi_1}^2[d_3 sin\xi_1 - d_2(cos\xi_2 - cos\xi_1)]/({\xi_2}^2 sin\xi_2) \\
e_{11} =& {\xi_1}^2/{\xi_2}^2 - 1 \\
e_{12} =& cos\xi_2/(\xi_2 sin\xi 2) \\
e_{13} =& 1/(\xi_2 sin\xi_2)
\end{aligned}
\tag{11}
$$

Continuity At The Joint Between Adjacent Patches

To make the appearance of the 3D model smooth, the continuities in both u and v parametric directions have to be maintained where the adjacent patches are connected. This part will explain how the continuities are achieved through this algorithm.

Continuity in parametric direction u. First assume two neighbour patches $S(u,v)$ and $\hat{S}(u,v)$, which are all defined by Equation (1) and Equation (2), with different shape control parameters ρ, η, λ and $\hat{\rho}, \hat{\eta}, \hat{\lambda}$ respectively. The surface patch $\hat{S}(u,v)$ could be generated by these six curves:

$$
\begin{aligned}
&\hat{S}(0,v) = c_4(v) \quad \frac{\partial \hat{S}(0,v)}{\partial u} = c_5(v) \quad \frac{\partial^2 \hat{S}(0,v)}{\partial u^2} = c_6(v) \\
&\hat{S}(1,v) = \hat{c}_4(v) \quad \frac{\partial \hat{S}(1,v)}{\partial u} = \hat{c}_5(v) \quad \frac{\partial^2 \hat{S}(1,v)}{\partial u^2} = \hat{c}_6(v)
\end{aligned}
\tag{12}
$$

As mentioned above, the two patches must share the same boundary curve as well as the first and second differential for fulfilling the continuities at the joint, i.e. satisfying the constraints $S(1,v) = \hat{S}(0,v)$, $\partial S(1,v)/\partial u = \partial \hat{S}(0,v)/\partial u$ and $\partial^2 S(1,v)/\partial u^2 = \partial^2 \hat{S}(0,v)/\partial u^2$.

Continuity in parametric direction v. Like creating surfaces in u direction according to Equation (8), the similar method could be used when considering the continuity in direction v. The first and second partial differential of $S(u,v)$ with

respect to the parametric variable v could be unified as

$$\begin{aligned} S^{(i)}(u,v) =& g_1(u)c_1^{(i)}(v) + g_2(u)c_2^{(i)}(v) \\ &+ g_3(u)c_3^{(i)}(v) + g_4(u)c_4^{(i)}(v) \\ &+ g_5(u)c_5^{(i)}(v) + g_6(u)c_6^{(i)}(v) \end{aligned} \tag{13}$$

where $i = 0, 1, 2$ stands for the surface, its first and second partial differential in direction v respectively.

Thus, to achieve continuities, the condition $S^{(i)}(u,1) = \hat{S}^{(i)}(u,0)$ $(i = 0,1,2)$ must be satisfying $g_k^{(i)}(u) = \hat{g}_k^{(i)}(u)$ $(i = 0,1,2; k = 1,2,...,6)$ and $c_k^{(i)}(1) = \hat{c}_k^{(i)}(0)$.

To guarantee $g_k^{(i)}(u) = \hat{g}_k^{(i)}(u)$ $(i = 0,1,2; k = 1,2,...,6)$, according to Equation (9), $d_l = \hat{d}_l (l = 1,2,...,10)$ could be achieved, and then it is satisfied when $e_p = \hat{e}_p (p = 1,2,...,13)$ depending on Equation (10).

Finally, from Equation (11), the task has changed into ensuring $\xi_n = \hat{\xi}_n (n = 1,2,3,4)$. In another word, when $\xi_n = \hat{\xi}_n (n = 1,2,3,4)$ and $c_k^{(i)}(1) = \hat{c}_k^{(i)}(0)$, the two patches $S(u,v)$ and $\hat{S}(u,v)$ also satisfy the continuity with respect to the parametric variable v.

Patch Separation

After elaborating how to generate signal surfaces as well as how to achieve the C2 continuities among different patches to make the whole model smooth, it could not be ignored how to create complex objects with the above approach.

There are two different ways to be addressed to use. The first is to decompose the objects into small parts. It is convenient for creating objects whose each surface patch shares the same boundary curves and same control curves with its neighbour at the edges. As shown in Figure 2, the dog is decomposed into parts including rear legs, front legs, tail, torso, neck and head, with some parts further divided into patches. Every two adjacent patches share the same boundary and control curves at the joint.

Figure 2: Model Patches Separation

The second way is more used for detailed models like human face: separating the patches according to the sketched curves which define 4-sided and 3-sided patches. This approach will be explicated in the Implementation part.

Interpolation

Two disparate approaches are used in this project for interpolation, as comparison. One is linear interpolation, which is purely geometric without considering underlying physics. The other is a method stemmed from Newton's second law, written as

$$ma = f \tag{14}$$

where m stands for mass, a denotes the acceleration which has three components $a^{(i)}(i = 1,2,3)$ with $a^1 = a_x$, $a^2 = a_y$ and $a^3 = a_z$, and f stands for the external force also with three components in each direction $f^{(i)}(i = 1,2,3)$.

The acceleration a is related to the position change x as

$$a = \frac{d^2x}{dt^2} \tag{15}$$

where the position change x has three components $x^{(i)}(i = 1,2,3)$ with $x^{(1)} = x$, $x^{(2)} = y$, and $x^{(3)} = z$.

Substituting Equation (15) into Equation (14), the following formula could be obtained in the form of components,

$$m\frac{d^2x^{(i)}}{dt^2} = f^{(i)} \qquad (i = 1,2,3) \tag{16}$$

The next step is to compute the interpolation values as the model varies from the first model to the second one with time t changing. Assume the first model is defined by the polygon vertices $\bar{x}_n (n = 1,2,3,...,N)$, and the second one is defined by $\bar{\bar{x}}_n (n = 1,2,3,...,N)$. It is easy to get that, the position change x at the $t = 0$ is zero, i.e. $x_n = 0 (n = 1,2,3,...,N)$ and is $x_n = \bar{\bar{x}}_n - \bar{x}_n$ at $t = 1$.

If we assume that the rate of the position change at $t = 0$ is zero, i.e. $dx_n/dt = 0$, the boundary conditions could be gained

$$\begin{aligned} t = 0 \qquad & x_n = 0 \\ & \frac{dx_n}{dt} = 0 \\ t = 1 \qquad & x_n = \bar{\bar{x}}_n - \bar{x}_n \\ & (n = 1,2,3,...,N) \end{aligned} \tag{17}$$

In the form of components, Equation (17) could change into

$$\begin{aligned} t = 0 \qquad & {x_n}^{(i)} = 0 \\ & \frac{d{x_n}^{(i)}}{dt} = 0 \\ t = 1 \qquad & {x_n}^{(i)} = {\bar{\bar{x}}_n}^{(i)} - \bar{x}_n^{(i)} \\ & (i = 1,2,3; \quad n = 1,2,3,...,N) \end{aligned} \tag{18}$$

Replacing $x^{(i)}$ with ${x_n}^{(i)}$ in Equation (16), we can get

$$m\frac{d^2{x_n}^{(i)}}{dt^2} = {f_n}^{(i)} \quad (i = 1, 2, 3; \quad n = 1, 2, 3, ..., N) \tag{19}$$

Dividing both sides of Equation (19) by m, we obtain

$$\frac{d^2{x_n}^{(i)}}{dt^2} = \frac{{f_n}^{(i)}}{m} \quad (i = 1, 2, 3; \quad n = 1, 2, 3, ..., N) \tag{20}$$

Integrating Equation (20) with respect to time t twice, we can gain

$${x_n}^{(i)} = \frac{{f_n}^{(i)}t^2}{2m} + c_0 t + c_1 \quad (i = 1, 2, 3; \quad n = 1, 2, 3, ..., N) \tag{21}$$

where c_0 and c_1 are unknown constants.

Then introducing Euqation (21) into Equation (18), and solving for the two unknown constants c_0 and c_1 as well as the external force ${f_n}^{(i)}$, we have

$$c_0 = c_1 = 0$$
$$f_n^{(i)} = 2m[\bar{\bar{x}}_n{}^{(i)} - \bar{x}_n^{(i)}] \quad (i = 1, 2, 3; \quad n = 1, 2, 3, ..., N) \tag{22}$$

Substituting Equation (22) into Equation (21), we reach the following position change formula

$${x_n}^{(i)} = [\bar{\bar{x}}_n{}^{(i)} - \bar{x}_n^{(i)}]t^2 \quad (i = 1, 2, 3; \quad n = 1, 2, 3, ..., N) \tag{23}$$

Finally, by adding ${x_n}^{(i)}$ to the position values $\bar{x}_n^{(i)}$ of each point of the first model, we could get the interpolated position values $\hat{x}_n^{(i)}$ at any poses between the first and second model as,

$$\hat{x}_n^{(i)} = \bar{x}_n^{(i)} + [\bar{\bar{x}}_n{}^{(i)} - \bar{x}_n^{(i)}]t^2 \quad (i = 1, 2, 3; \quad n = 1, 2, 3, ..., N) \tag{24}$$

With time t changes in the interval [0,1], we obtain a set of different position values of the model, and then they are used to form a deformation animation.

4 IMPLEMENTATION

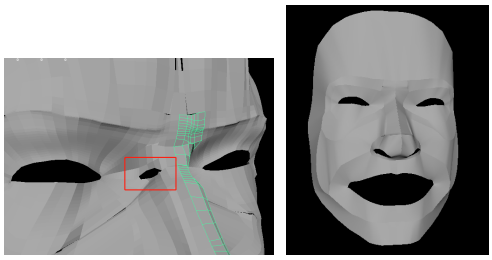
Compared with other motion animation, face deformation is more crucial since it requires higher precision to be regarded as realistic. Thus, in order to demonstrate the result of the implementation, two face models in the same topological structure with different expressions are chosen as Figure 3. I first use the ODE-based approach to recreate the two different models and then use linear interpolation to calculate the keyframes in between. Finally, form an animation between the two different reconstructed expression models to demonstrate the algorithm implementation as well as how it could be applied in face deforming animation.

(a)laugh (b)reference

Figure 3: Original Models

Patch Separation

The first step of dealing the model is separating the face into small 4-sided or 3-sided patches according to the theory part. In fact, it is the most time-consuming step in the whole project since I have to manually record the coordinates of every point on the sketched curves to extract the wireframe of the face model. Besides, I have met problems like surface distortion, details missing and soon on, which are shown in Figure 4.



(c)distortion (d)details missing

Figure 4: Mistakes When Rebuilding Models

However, after becoming more and more familiar with the face topological structure, I finally extracted a set of sketched curves which are relatively good as shown in Figure 5.

Paul *et al.* (2002) have provided the Facial Action Coding System (FACS) to use corresponding parameters to control disparate face shapes. Hamm *et al.* (2011) claimed that, the facial appearance could be used to encode the action of each facial muscles with marginally imperative changes. Furthermore, any anatomically possible facial expression could be encoded by FACS through deconstructing it into temporal segments and action units (Freitas-Magalhães 2013), which could be used for further interpolation to adapt to any intelligent environment.

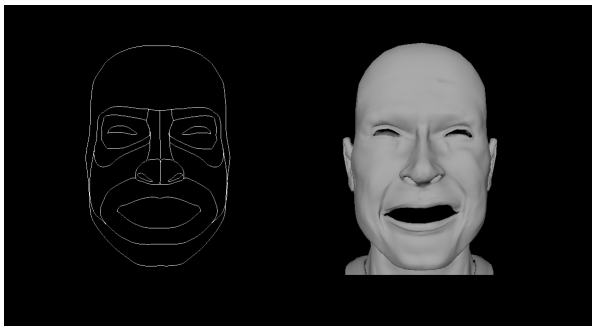


Figure 5: Sketched Curves

Therefore, based on this system, I separate the face model into seven different patches in Figure 6: eye socket, eye bone, nose bridge, nostril, cheek, mouth and outer face frame.

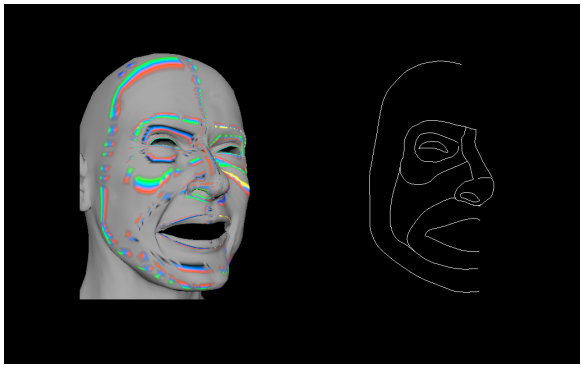


Figure 6: Patche Separation

Surface Recreation

After adding the relative control curves for each boundary curve to calculate the tangent and curvature of points on it, each surface is computed according to Bian's C2 continuity ODE-based method as Figure 7.

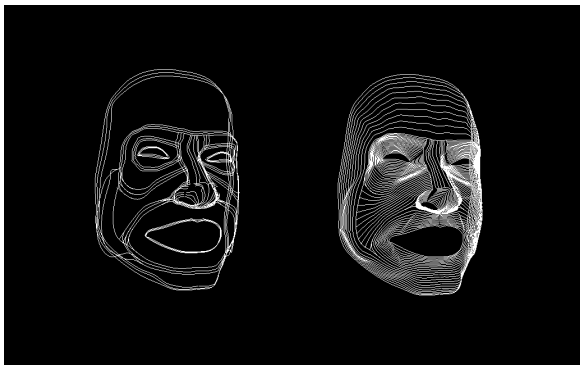


Figure 7: Wireframe

Figure 8 and 9 are the comparison between the original model and the recreated model of laugh pose and reference pose respectively.

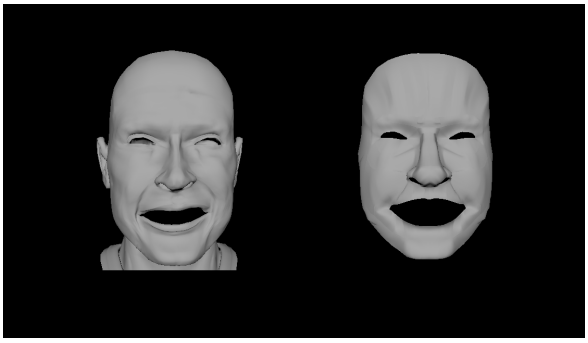


Figure 8: Comparison of Laugh Model

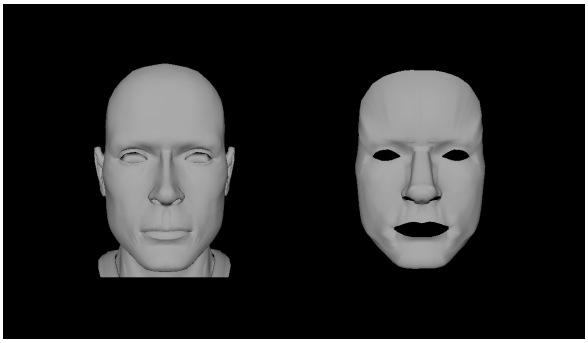


Figure 9: Comparison of Ref Model

It has to be noticed that, the width of the nose bridge has some difference. That is caused by the asymmetric face of the laugh model as the left face in Figure 8 whose axle wire is not unified in the direction x. My implementation is just to demonstrate the feasibility of the ODE-based algorithm and it takes double work to recreate the other sides without making too much sense. Therefore, I only recreate the left-side face with the axle wire unified to a proper value in the y-z coordinate plane and mirror it to the other side. Since the two models share the same topology, the reference model is changed relatively. Moreover, if more specific details require to be added, like the nasolabial folds, further subdivision on the cheek patch could achieve that, though I did not finish that due to the time limitation.

Anyway, as is easy to see that, the faces on the right hand do not lose too many details compared with the left-hand ones, whilst the former only use six curves to generate each patch. Furthermore, the original model takes 15378 polygons to store whilst the recreated model has only 4236 polygons, which is only 28% of the original one. That is to say, the ODE-based recreated model compresses 72% of the data size. Thus,

compared with the polygon modelling method, this approach could greatly save the data storage space with remaining the quality of the original model. This approach also provides a convenient way to quickly manipulate the face shapes and expressions through modifying the boundary curves and control curves.

In addition, by observing the generated models, the surface remains smooth since this technique achieves up to curvature continuities naturally. There is no extra manual operation to stitch different patches, which saves massive work.

Face Deforming Animation

To better demonstrate the advantage of this algorithm in further face deforming operation, I use the linear interpolation and Newton second law-based interpolation to generate the 30 keyframe models in between. The course is to calculate the six curves of each patch for keyframes. Then the computed boundary curves and control curves of each frame are used to recreate the interpolated models with the same surface recreation algorithm. After getting all the frames, I put them into Houdini to generate simple animations. Finally, the two animations with different interpolation methods are compared to show the advantage of the physics-based approach.

Figure 10 and 11 are the keyframe models of linear and Newton second law-based interpolation results respectively.

Figure 10: Linear Interpolation Keyframes

As could be seen, the face deformation is natural and smooth, which indicates that with the same topology, the ODE-based recreated models could adapt the deformation operations. Moreover, compared between the two figures, the

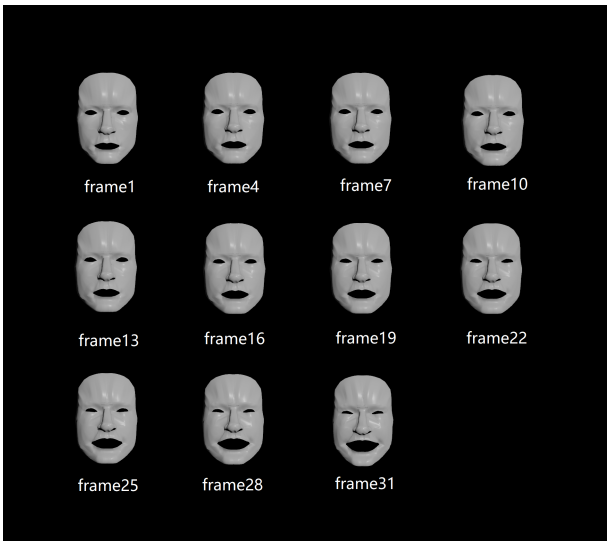


Figure 11: Physics-based Interpolation Keyframes

change of expression of Figure 11 starts later than Figure 10, with a smoother change, and the amplitude of the variation is also smaller. That is to say, the physics-based approach could better describe the natural variation of the facial expressions.

5 CONCLUSION

In this master project, I implement the algorithm, which is based on the mathematical model using C2 continuous boundary constraints and a vector-valued sixth order ODE, to recreate the surfaces of two face models in the same topology with different expressions. The smooth appearance demonstrates that, when the adjacent patches satisfy sharing the same constraints of the position, and the first and second differentials, the boundary continuities could be achieved. Besides, this method is fairly convenient to control the shapes of each patch by only manipulating six key curves, rather than modifying all vertices through polygon modelling, so that the work of stitching patches could be left. Most significantly, this method greatly saves the storage space without losing the quality of the original model. Furthermore, I blend it with skin deformation techniques to form facial variation animations. From the comparison between two animations, it is easy to figure out that physics-based techniques could create more natural and smooth interpolation keyframes.

However, there could be some developments like how to reduce the work in extracting the wireframes, which spent the most time of this master project, whether I could finish this procedure procedurally, and how to blend the surface creation approach with more advanced skin deformation methods like Bian *et al.* (2019)'s PDE-based algorithm. I wish I could make more progress in my further study.

Anyway, I am grateful for this period of study since it let me have a better understanding of face modelling techniques, face deformation techniques and ODEs as well as its powerful function in 3D model sculpting.

REFERENCES

[1] Allen B., Curless B. and Popović Z., 2002. Articulated body deformation from range scan data. *ACM Transactions on Graphics (TOG)*, **21**(3), 612–619.

[2] Bian S., Deng Z., Chaudhry E., You L., Yang X., Guo L., Ugail H., Jin X., Xiao Z. and Zhang J. J., 2019. Efficient and realistic character animation through analytical physics-based skin deformation. *Graphical Models*, **104**, 101035.

[3] Bian S., Maguire G., Kokke W., You L. and Zhang J. J., 2020. Efficient c2 continuous surface creation technique based on ordinary differential equation. *Symmetry*, **12**(1), 38.

[4] Chaudhry E., You L. and Zhang J. J., 2010. Character skin deformation: A survey. In *2010 Seventh International Conference on Computer Graphics, Imaging and Visualization*. IEEE, 41–48.

[5] Chaudhry E., You L., Jin X., Yang X. and Zhang J. J., 2013. Shape modeling for animated characters using ordinary differential equations. *Computers & graphics*, **37**(6), 638–644.

[6] Debunne G., Desbrun M., Cani M.-P. and Barr A. H., 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 31–36.

[7] Freitas-Magalhães A., 2013. *The face of lies*. Leya.

[8] Hamm J., Kohler C. G., Gur R. C. and Verma R., 2011. Automated facial action coding system for dynamic analysis of facial expressions in neuropsychiatric disorders. *Journal of neuroscience methods*, **200**(2), 237–256.

[9] Magnenat-Thalmann N., Laperrire R. and Thalmann D., 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer.

[10] Nealen A., Müller M., Keiser R., Boxerman E. and Carlson M., 2005. Physically based deformable models in computer graphics. In *EUROGRAPHICS 2005 STAR–STATE OF THE ART REPORT*. Citeseer.

[11] Paul E., Joseph C H. and Wallace V F., 2002. *Facial action coding system*. Salt Lake City : A Human Face.

[12] Piegl L. and Tiller W., 2012. *The NURBS book*. Springer Science & Business Media.

[13] Platt S. M. and Badler N. I., 1981. Animating facial expressions. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, 245–252.

[14] Russo M., 2006. *Polygonal modeling: basic and advanced techniques*. Jones & Bartlett Learning.

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

[16] Várady T., Salvi P. and Rockwood A., 2012. Transfinite surface interpolation with interior control. *Graphical models*, **74**(6), 311–320.

[17] You L., Yang X., You X. Y., Jin X. and Zhang J. J., 2010. Shape manipulation using physically based wire deformations. *Computer Animation and Virtual Worlds*, **21**(3-4), 297–309.

[18] You L., Yang X., Pachulski M. and Zhang J. J., 2007. Boundary constrained swept surfaces for modelling and animation. In *Computer Graphics Forum*, volume 26. Wiley Online Library, 313–322.