Effective T-spline representation for VRML

Wenyu Chen [1], Yusha Li [1], Jianjiang Pan[1], Jianmin Zheng[1,3] and Yiyu Cai[2,3]

[1] School of Computer Engineering, Nanyang Technological University
[2] School of Mechanical & Aerospace Engineering, Nanyang Technological University
[3] Institute for Media Innovation, Nanyang Technological University

***Abstract*— Less control points are needed to represent a shape in T-Splines compared to NURBS and subsequently less time is spent in modeling. While getting more and more accepted by commercial software, T-splines, however, are yet part of VRML/X3D. The T-spline VRML is proposed in this work. An effective data structure is designed for T-splines to support online visualization. Compared to the NURBS and the polygonal representations, the proposed T-spline data structure representation can significantly reduce the VRML file size which is a central concern in online applications. As such, complex objects modeled in T-spline form have better chances for real-time transfer from servers to clients. Similar to other VRML nodes, T-spline VRML node can support geometry, color and texture. Users can interact with T-spline more effectively for LOD and animation applications.**

***Index Terms*—NURBS, T-splines, VRML/X3D, Virtual Reality**

I. INTRODUCTION

Virtual Reality Modeling Language (VRML) has been an ISO standard since 1997. After that, VRML97 has been widely used for online visualization. Extensible 3D (X3D) [1] is the enhanced successor to VRML. It also became an ISO standard in 2004. All specifications in VRML97 are fully adopted by X3D. Nowadays, VRML is still very popular.

Initially, VRML uses polygonal meshes for the online 3D objects. Although, Non-uniform Rational B-spline Surfaces (NURBS) [2] are popular for 3D freeform shapes, NURBS was too complex to be adapted in VRML97 due to the hardware limitations. As the hardware system becoming sufficient for in tessellation and rendering, NURBS was adopted in the VRML extensions [3]. Popular plug-ins such as BS Contact [4] and Cortona3D [5] have already supported the NURBS extensions. As the successor of VRML97, X3D also provides NURBS nodes in different forms.

The T-spline surface is an extension of NURBS by allowing T-junctions [6,7] . Compared to NURBS, a T-spline surface allows partial rows/columns of control points, thus reduces the number of control points.

Manuscript Received on October 1, 2011.
First author's e-mail address: wychen@ntu.edu.sg

Extending VRML to support T-splines can provide the following benefits:

- minimization of the VRML file size using the compact T-spline representation,
- faster modeling, as T-spline users can achieve the same shapes as NURBS users with less operations,
- supporting data communications, particularly due to the increasing use of T-splines in modeling and design industries (e.g., Rhino and Solidwork),
- better animation control, since T-splines support a better local modification ability compared to NURBS,
- automatic Level of Details (LOD) support.

The compact representation makes the T-spline surface a good candidate for online data transmission and visualization. Online data transmission in T-spline format can benefit the collaborative design. VRML has been used as a platform for the collaborative design systems [8,9]. However, complex models can be too huge to download in real-time. Many methods can be adopted to shorten the waiting time for a huge polygonal mesh. Progressive delivery [10] delivers only a coarse model which is gradually refined. Mesh compression [11] is performed with the model before the transmission. One drawback for polygonal models is that they may not be real-time updated for multi-user editing due to the huge file size. To achieve data transmission in the real-time, it is better to represent a 3D model using freeform surfaces rather than tessellated polygons. As such, the T-spline surface is ideal for data representation to reduce the size of transmitted data. Principally, only necessary data (e.g. control points, knot vectors) are transmitted for the T-spline represented model and tessellation can be done on the client sites.

A T-spline surface allows more freedom for its control mesh, which is different from NURBS. A VRML extension for T-splines can be useful for online users to learn and use T-splines. A T-spline VRML node can also be used in application like collaborative product design.

In the next section, we will first introduce VRML and T-splines. A new data structure for the T-mesh and the implementation of the T-spline VRML node is presented in Section III. Section IV shows the benefits using the T-spline VRML nodes in virtual worlds. The proposed T-spline node can work with other VRML nodes to achieve various effect. Finally, Section V concludes our research and discusses the future work.

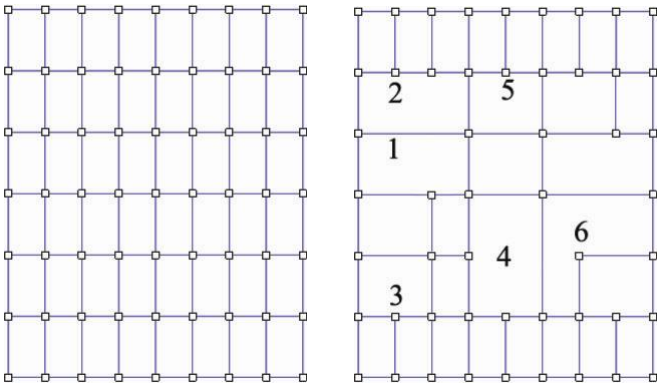


Fig. 1. The preimage (row for *s* direction and column for *t* direction): (a) for a NURBS surface, and (b) for a T-spline surface (No.1,No.2 and No.3 are on a same column; N-point No.4 locates on the same column as No.5 and the same row as No.6).

II. PRELIMINARIES

2.1 VRML

VRML may be used in a variety of application areas such as scientific visualization, entertainment and virtual worlds. It is a file format for describing interactive 3D objects. A VRML file is essentially a collection of **Nodes.** A **Node** can be a geometric shape such as a sphere, a box and a cylinder. It can also be a non-shape object like scripting, viewpoint, transformation, etc. One **Node** usually contains several **Fields**. The data for each **Node** are stored in **Fields**.

VRML is extendible. It provides the ability to add new object types. Users can do extensions to support any type of object's geometry, appearance and transformation. The extension can be done by prototyping mechanism, which uses ***PROTO*** and ***EXTERNPROTO*** keywords. A prototype consists of

- the ***PROTO*** keyword
- the name of the new node,
- the prototype declaration, and
- the prototype definition.

An external prototype with ***EXTERNPROTO*** keyword defines prototypes in external files by providing a URL.

2.2 T-splines

Like NURBS surfaces, T-spline surfaces are also tenor-product B-spline surfaces which are shaped by their control points. Similar to NURBS, a T-spline surface has a control grid, which is called a T-mesh. The difference is that, control points of NURBS surface traverse the entire control grid, whereas in T-splines, only partial control grids are associated with control points. That means that a T-spline surface allows T-junctions, L-junctions or even I-junctions on the T-mesh. Fig 1(a) shows a control grid of a NURBS surface. Each of its control points connects to four neighboring points. In Fig 1(b), the points No.2, No.3 and No.5 are T-junctions, each of which connects to three neighboring points. The point No.6 in Fig 1(b) is an L-junction, which connected to only two neighboring points. A T-spline surface degenerates to a NURBS surface if the T-mesh forms a rectangular grid. Transforming the T-mesh in Fig 1(b) into a rectangular grid gives the control mesh in Fig 1(a).

Each edge in a T-mesh is assigned with a knot interval, which should satisfy the following rules:

- **Rule 1:** The sum of the knot intervals on opposing edges of any face must be equal.
- **Rule 2:** If two T-junction on opposing edges of a face can be connected without violating the previous rule, that edge must be included in the T-mesh.

A knot coordinate system can be defined by designating a control point on the preimage to be the origin (0, 0). Every vertical or horizontal edge can be assigned with an *s* knot value or a *t* knot value, respectively. Suppose a control point P_i corresponds to the knot coordinates (s_i, t_i). A T-spline basis function $B_i(s,t)$ for P_i is defined as

$$B_i(s,t) = N[\mathbf{s}_i](s)N[\mathbf{t}_i](t)$$

where $N[\mathbf{s}_i](s), N[\mathbf{t}_i](t)$ are the cubic B-spline basis functions associated with the knot vectors $\mathbf{s}_i = [s_{i-2}, s_{i-1}, s_i, s_{i+1}, s_{i+2}]$ and $\mathbf{t}_i = [t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}]$ respectively. Here, the basis functions are those used for NURBS surfaces [2]. Different from NURBS, the knot vectors $\mathbf{s}_i$ and $\mathbf{t}_i$ for T-splines are extracted from the T-mesh as follow:

- (s_i, t_i) is the knot coordinates of P_i.
- $R(u)=(s_i+u, t_i)$ is a horizontal line. By casting a ray $R(u)$, $u > 0$, the two values s_{i+1} and s_{i+2} are the knot values of the first two vertical edges that intersect with the ray. By casting a ray $R(u)$, $u < 0$,, the two values s_{i-1} and s_{i-2} are the knot values of the first two vertical edges that intersect with the ray.
- Other values t_{i-2}, t_{i-1}, t_{i+1} and t_{i+2} are defined in a similar manner.

Combining control points with basis functions, we can define a T-spline surface with *n* control points as

$$P(s,t) = \frac{\sum_{i=0}^{n} P_i w_i B_i(s,t)}{\sum_{i=0}^{n} w_i B_i(s,t)}, \qquad (1)$$

where $B_i(s,t)$ are basis functions and (w_i, P_i) are weighted control points on the T-mesh.

Similar with NURBS surfaces, T-spline surfaces also support local refinements [7]. Knot removals for T-splines can be adopted for surface simplifications [12]. Surface fitting techniques can be used to obtain a T-spline surface from a polygonal mesh [13],[14]. A T-spline surface can also be transferred in to other surfaces such as NURBS and H-spline [15].

III. T-SPLINE VRML NODE

VRML is rich in features. However, polygonal meshes are insufficient for complex shapes [16]. For a scene with complex shapes, the VRML file may be huge. It will take a long time to download the VRML file before the user can view the scene. The NURBS VRML node can help to reduce the file size while producing a same shape as a polygonal mesh. The proposed T-spline VRML node offers a simplified solution of the NURBS node. It can produce smooth complex shapes with fewer points when compared with the polygonal shapes and the NURBS shapes.

VRML extensions can be done using the prototype mechanism which can be used to introduce new nodes. In this section, we build a new prototype for T-spline VRML nodes, which can be used with standard VRML nodes for complex world creations.

A T-spline node can be defined in a VRML file by using the ***EXTERNPROTO*** declaration, which will select the T-spline ***PROTO*** from the file TSplineSurface.wrl. As shown in Fig 2, when a T-spline VRML node is parsed by the browser, the browser will take the following steps to visualize the surface:

- **Step 1:** The T-spline prototype is loaded.
- **Step 2:** Necessary information of the T-spline surface is passed to the prototype, including knots, T-mesh, texture etc.
- **Step 3:** Tessellation is performed to obtain vertex coordinates and vertex indices. If texture is involved, the texture coordinates for each vertex is also calculated.
- **Step 4:** A VRML ***IndexedFaceSet*** node consisting of both polygons and textures is assembled and passed back to the browser for visualization.

We implemented the proposed prototype for T-spline VRML node using Java [17]. Thus, all information of a T-spline surface is passed to a Java applet, which will perform the tessellation and pass back a VRML ***IndexedFaceSet*** node (Fig 2). The visualization of the T-spline VRML node can be performed using InstantPlayer [18].

As shown in Fig 2, to implement a T-spline VRML node, we need to handle three issues

- How to organize the input data in a proper data structure?
- How to define the prototype?
- How to control and perform the tessellation?

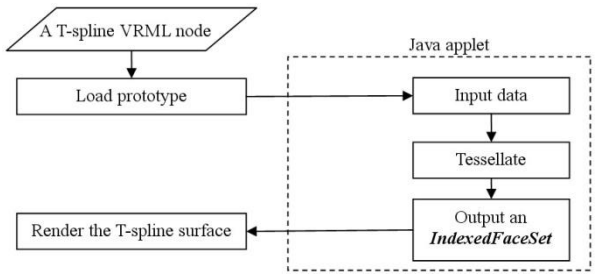


Fig. 2. The flowchart for T-spline visualization in VRML.

3.1 T-spline data structure

According to the initial definition [7], the information to store a T-mesh includes the T-points, the edges and the faces. Such data structure is quite different from NURBS, which uses a rectangular point grid to store its control points. The T-mesh data structure is not intuitive for implementation and editing. Removing or inserting a point into the T-mesh results in changing of the T-points data, the edge data and the face data. One advantage of VRML is that a VRML file can be directly edited using a text editor. However, with such T-spline data structure, user needs to keep in mind a correct T-mesh topology in order to edit the VRML file properly.

To overcome this drawback, we propose a new data structure for T-splines. For T-spline VRML extension, we represent a T-mesh as a rectangular point grid of size $S_s \times S_t$, where S_s and S_t are the dimensions along the s and t directions, respectively. Different from NURBS, no all points on the T-mesh are control points (Fig 1). T-mesh points fall into two categories: N-points and T-points. A T-point is a control point on the T-mesh (square points in Fig 1(b), including points No.2, No.3, No.5, and No.6) while an N-point is a virtual point (point No.1 and No.4 in Fig 1(b)). By transferring all N-points into T-points, the T-mesh becomes a NURBS control mesh and the T-spline surface becomes a NURBS surface.

Each T-mesh point has a value c indicating its connectivity with its neighboring points. The value c is an accumulated value of

- 0x0001 if it connects to the next point along s^- direction,
- 0x0010 if it connects to the next point along s^+ direction,
- 0x0100 if it connects to the next point along t^- direction,
- 0x1000 if it connects to the next point along t^+ direction.

Thus, $c \in \{0,1,\cdots,15\}$. In Fig 1(b), $c = 3$ for N-point No.1, $c = 0$ for N-point No.4, $c = 11$ for T-point No.2, and $c = 6$ for T-point No.6.

The storage of a T-mesh is different from the control mesh of a NURBS. For N-points, no x, y, z coordinates are required. In our data structure, each T-point is represented by 5 values (w, c, x, y, z) with $w>0$ while an N-point only needs 2 values as (w, c) with $w = -1$ indicating that it is an N-point.

By assigning a connectivity value for each T-mesh point, we do not need to store the edges and the faces, which can be automatically derived from the connectivity values. Meanwhile, the proposed data structure is intuitive for implementation and editing. It is easy for user to directly manipulate the VRML file to edit and create a T-spline surface.

A T-point contains 5 values (w, c, x, y, z) with $w>0$ while an N-point only has 2 values as (w, c) with $w = -1$. Since the connectivity value c is an integer, we can encode w and c into a single value f as

$$f = \begin{cases} -c, & w \le 0, \\ 10c + n + w/10^n, & w > 0. \end{cases} \tag{2}$$

where n is the digit of w's integer part $\lfloor w \rfloor$. In this way, a T-point becomes (f, x, y, z) and an N-point is f. For decoding, given f, we can obtain w_1, c_1 as

$$\begin{aligned} f \le 0 \Rightarrow \quad & w_1 = -1, c_1 = -f, \\ f > 0 \Rightarrow \quad & c_1 = \lfloor f \rfloor / 10, m = f - 10c_1, n = \lfloor m \rfloor, \\ & w_1 = (m-n) \times 10^n. \end{aligned} \quad (3)$$

Theoretically, according to calculation, w_1=w and c_1=c hold for all N-points and all T-points. In applications, for a T-point, w_1 and w are of type SFFloat in VRML. Thus, $w_1 = (w/10^n) \cdot 10^n$ does not always lead to w_1=w. In fact, w_1=w only holds if the last n digits of w' decimal part are all zero. Otherwise, even if there is a tiny loss for the weight w_1<w, in most cases, w_1-w is too tiny to affect the visual effect of the surface.

For a NURBS control grid of size $S_s \times S_t$, each control point needs 4 values as (w, x, y, z). The number of values needed for the NURBS control grid in a VRML node is

$$L_1 = 4S_sS_t. \quad (4)$$

For T-mesh point grid of size $S_s \times S_t$, 4 values are required for a T-point while only one single value is needed for a N-point. The number of values needed for the T-mesh grid in a VRML node is

$$L_2 = 4S_sS_t - 3N, \quad (5)$$

where N is the number of N-points on the T-mesh grid.

Combining Eq.(4) and Eq.(5) gives

$$L_1 - L_2 = 3N \ge 0. \quad (6)$$

Thus, the VRML file size using T-spline nodes is smaller than the VRML file size using NURBS nodes if their control grids are of same dimensions. In the worst case N=0, a T-spline VRML node requires same number of values as a NURBS VRML node.

3.2 Prototype

The T-spline VRML prototype is defined in a file named "TSplineSurface.wrl".

PROTO TSplineSurface[
*exposedField SFInt*32 ***numRow***
*exposedField SFInt*32 ***numCol***
exposedField MFFloat ***sKnots***
exposedField MFFloat ***tKnots***
exposedField MFFloat ***controlPoints***
exposedField MFFloat ***texCoord***
*exposedField SFInt*32 ***sTess***
*exposedField SFInt*32 ***tTess***
eventIn SFBool ***rebuild***
field SFBool ***ccw***
field SFFloat ***creaseAngle***
field SFBool ***solid***]

numRow and ***numCol*** define the dimensions of the T-mesh on the s and t directions.

sKnots and ***tKnots*** are two knot vectors on the s and t directions. From a mathematical point of view, on each direction, the number of knots must be equal to the number of T-mesh points plus 2.

controlPoints defines a T-mesh of dimension ***numRow*** ×***numCol***. A positive weight value is assigned to each T-point. According to Eq.(1), a bigger weight w_j makes the surface closer to the T-point P_j. Suppose a T-mesh point P starts at f = ***controlPoints*[*i*]**.

- if $f \le 0$, it is an N-point without x, y, z coordinates, the connectivity $P.c$ is derived using Eq.(3), and the next T-mesh point starts at ***controlPoints*[*i*+1]**.
- If f >0, it is a T-point with
- $$\begin{aligned} P.x &= \textbf{controlPoints}[i+1] \\ P.y &= \textbf{controlPoints}[i+2] \\ P.z &= \textbf{controlPoints}[i+3] \end{aligned} \quad (7)$$

  the connectivity $P.c$ and the weight $P.w$ is derived using Eq.(3), and the next T-mesh point starts at ***controlPoints*[*i*+4]**.

texCoord provides texture coordinates for each T-point. If ***texCoord*** is empty, texture coordinates are automatically generated in the unit square.

sTess and ***tTess*** are tessellation parameters defining how to sample points on the domain. Five types of values are supported:

- ***sTess*** = ***tTess*** = 0 for T-mesh,
- ***sTess*** >0,***tTess*** > 0 for semi-uniform tessellations,
- ***sTess*** <0,***tTess*** < 0 for uniform tessellations,
- ***sTess*** >0,***tTess*** < 0 for semi-uniform dynamic tessellations,
- ***sTess*** <0,***tTess*** > 0 for uniform dynamic tessellations.

For normal tessellations ***sTess*** · ***tTess***≥0, the tessellation is performed directly basing on the value ***sTess***, ***tTess***. For dynamic tessellations ***sTess*** · ***tTess***<0, ***sTess***, ***tTess*** values are not directly used for the tessellation. The final ***sTess***, ***tTess*** values will be dynamic updated for the tessellation according to the distance to the user.

rebuild is an eventIn function with a SFBool parameter. If the parameter value is true, the tessellation will be forced to perform once. This function can be call to apply tessellations after the change of surface parameters in exposedField such as ***sKnots***, ***tKnots***, and ***controlPoints***.

The definition for ***ccw***, ***creaseAngle*** and ***solid*** are the same as other VRML geometry nodes.

3.3 Tessellation

The tessellation parameters ***sTess*** and ***tTess*** determine how a polygonal mesh is derived from the T-spline surface. For

normal tessellation ***sTess*** · ***tTess***≥0, the parameters can be zero, positive and negative.

For zero parameters (***sTess*** = ***tTess*** = 0), the T-mesh is returned as an approximation to the surface (Fig 3(d)). On the output polygonal mesh, the vertex number is the same as the number of T-points, and a polygon is obtained from each face of the T-mesh.

Positive parameters (***sTess*** >0, ***tTess*** > 0) define a semi-uniform tessellation. A $(\boldsymbol{sTess}+1)\times(\boldsymbol{tTess}+1)$ uniform grid is sampled on each face $F_{ij}=[s_i,s_{i+1}]\times[t_j,t_{j+1}]$ with $s_{i+1}>s_i, t_{j+1}>t_j$ (Fig 3(e,f)). Each sampled point is taken at (u_m,v_n) with

$$u_m = s_i + m\cdot d_s, \quad m=0,1,\cdots,\boldsymbol{sTess},$$
$$v_n = t_j + n\cdot d_t, \quad n=0,1,\cdots,\boldsymbol{tTess},$$

where $d_s=\frac{s_{i+1}-s_i}{\boldsymbol{sTess}}, d_t=\frac{t_{j+1}-t_j}{\boldsymbol{tTess}}$. The grid sampled on the domain is of size

$$P_{ver}=\big(\boldsymbol{sTess}\cdot(\boldsymbol{numRow}-3)+1\big)\cdot\big(\boldsymbol{tTess}\cdot(\boldsymbol{numCol}-3)+1\big),$$

and the number of triangle is

$$P_{tri}=2\cdot\boldsymbol{sTess}\cdot\boldsymbol{tTess}\cdot(\boldsymbol{numRow}-3)\cdot(\boldsymbol{numCol}-3). \quad (8)$$

Negative parameters (***sTess*** <0, ***tTess*** < 0) lead to a uniform tessellation. A $(1-\boldsymbol{sTess})\times(1-\boldsymbol{tTess})$ uniform grid is sampled on the domain (Fig 3(a-c)). The number of vertices is

$$N_{ver}=(1-\boldsymbol{sTess})\cdot(1-\boldsymbol{tTess}),$$

and the number of triangles is

$$N_{tri}=2\cdot\boldsymbol{sTess}\cdot\boldsymbol{tTess}. \quad (9)$$

In Fig 3, the dimension for the T-mesh is $\boldsymbol{numRow}\times\boldsymbol{numCol}=56\times53$, and the tessellation parameters are taken as $\boldsymbol{sTess}=\boldsymbol{tTess}=\alpha$. Then the numbers of sampled points using negative and positive parameters become

$$N_{ver}=(1-\alpha)^2$$
$$P_{ver}=\big(53\alpha+1\big)\big(50\alpha+1\big). \quad (10)$$

Both uniform and semi-uniform tessellations can guarantee gap-free polygonal meshes. For the positive values, increasing α will sample more points on the surface. For the negative values, more points is sampled by decreasing α .

On the T-spline surface, the domain for local details usually contains several faces F_{ij} . As such, semi-uniform tessellation will sample points within each F_{ij} , thus preserve feature shapes. However, uniform tessellation may miss some faces, thus fail to sample the feature. On the other hand, for the semi-uniform tessellation, the number of vertices P_{ver} depends on the dimensions of a T-mesh. The number is easy to be huge by increasing α . Using negative values, N_{ver} is independent of the dimensions of a T-mesh. In VRML files, users can set a proper tessellation value according to different applications and the computer performances. For high-end PCs and for applications requiring more on surface features, positive values can be used. Otherwise, negative values are enough.

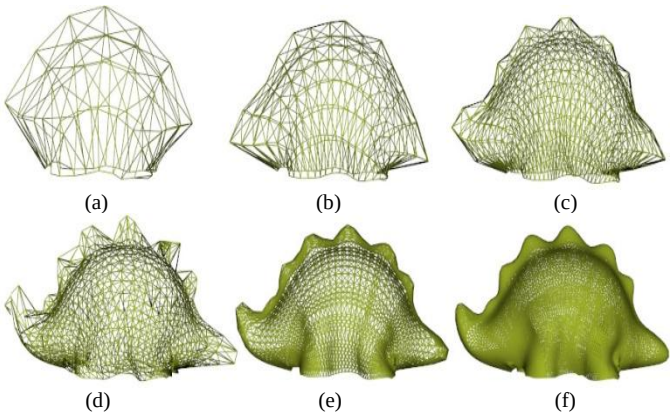


Fig. 3. The surface using different tessellation parameters: (a) $\alpha=-5$, (b) $\alpha=-15$, (c) $\alpha=-25$, (d) $\alpha=0$, (e) $\alpha=1$, and (f) $\alpha=2$.

3.4 Discussion

A complex surface can be represented as a T-spline surface, a NURBS surface or a polygonal mesh. To certain degree, different types of representations can provide a same shape. However, the sizes of the VRML file using different representations will differ from each others.

VRML ***IndexedFaceSet*** node adopts a polygonal mesh. One drawback of VRML ***IndexedFaceSet*** node is that the file size for a complex object may be many times larger than it needs [16]. Fig 4 shows an avatar surface of dimensions $\boldsymbol{numRow}\times\boldsymbol{numCol}=41\times45$. The surface is tessellated using $\boldsymbol{sTess}=\boldsymbol{tTess}=\alpha$, and saved as an ***IndexedFaceSet*** node in VRML. In order to preserve surface features, α should be smaller enough, i.e. more points should be sampled on the surface (Fig 4(e,f)). However, according the statistics in Table I, the more detail for the surface the larger file size required.

To reduce the file size for a complex object, it better to used NURBS nodes or T-spline nodes. According to Eq.(5), to represent a same shape, the file size for a T-spline surface is smaller than the file size for a NURBS surface. By experiments, to represent a same complex shape using both T-spline surfaces and NURBS surfaces, we usually have

$$N_t=\beta N_n, \beta\in(1/3,2/3), \quad (13)$$

where N_t is the number of T-points for the T-spline surface, and $N_n=S_sS_t$ is the number of control points for the NURBS. To represent the shape in Fig 4, a T-spline surface needs 905 T-points, but a NURBS surface requires 1845 control points. As a result, compared with NURBS nodes, a T-spline node needs smaller file size.

T-spline node needs to store a weight value and a connectivity value for each point (T-point and N-point). To represent a non-rational surface, a NURBS VRML node does not need to store any weight in the file [3]. In the implementation, all the weights will be 1 by default. In such case, Eq.(4) becomes

$$L_1 = 3S_sS_t. \quad (14)$$

Combining Eq.(14) and Eq.(5) gives

$$L_1 - L_2 = 3N - S_sS_t. \quad (15)$$

Thus, for a non-rational surface, the file size for T-spline node is smaller than the file size for NURBS node if and only if $N > \frac{1}{3}S_sS_t$, that is N is bigger than one third of the number of control points for NURBS. From Eq.(13), we have

$$N = S_sS_t - N_t = (1-\beta)S_sS_t > \frac{1}{3}S_sS_t. \quad (16)$$

Therefore, in most applications, T-spline node can offer a smaller file size. This is the most important advantage of using the T-spline surface over NURBS surfaces and polygonal meshes.

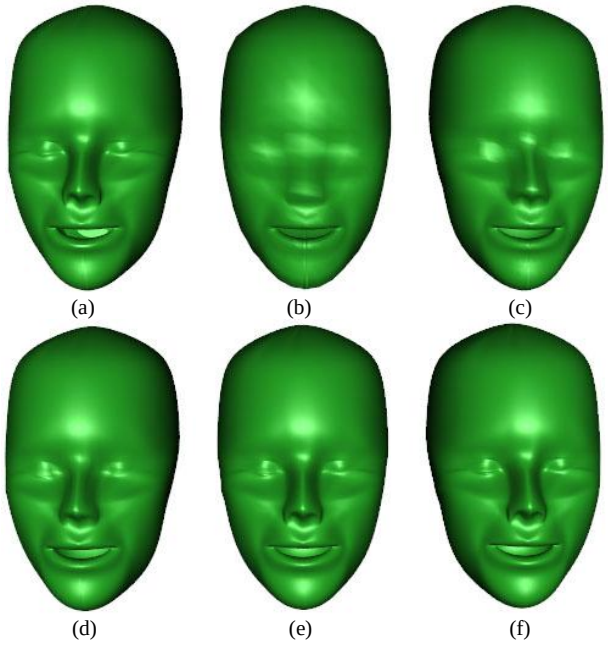


Fig. 4. The surface from different VRML files : (a) a T-spline surface, (b) $\alpha = -30$, (c) $\alpha = -60$, (d) $\alpha = -100$, (e) $\alpha = -200$, and (f) $\alpha = -300$. （Color Plate 5）

TABLE. I. STATISTICS FOR THE AVATAR IN FIG 4.

	File size(KB)	*Vertex Num*	*Triangle Num*
T-spline	37	905	-
NURBS	60	1845	-
Fig. 4 (b)	57	961	1800
Fig. 4 (c)	240	3721	7200
Fig. 4 (d)	675	10201	20000
Fig. 4 (e)	2876	40401	80000
Fig. 4 (f)	6541	90601	180000

IV. T-SPLINES FOR VIRTUAL REALITY

This section will show how the T-spline VRML node works with traditional VRML nodes to create various effects.

4.1. T-splines in VR

Using T-spline VRML node can benefit the virtual design. However, editing the VRML file by-hand is not an easy job, especially to create the animations and to adjust the texture coordinates. For NURBS VRML node, commercial tools such as 3D Studio MAX can be used to edit the surface and exported in a correct format. However, for T-splines, there is not existing tools that can be used directly to export the T-spline VRML node into a VRML file. We have been developing a new tool focusing on using T-splines for complex object modeling. Fig 5 shows a screen snap shot editing a bottle model using the tool. Textures and animations can be assigned to the model which finally will be exported into a VRML file. Fig 6(a) is the exported bottle model that consists of two T-spline surfaces with different textures.

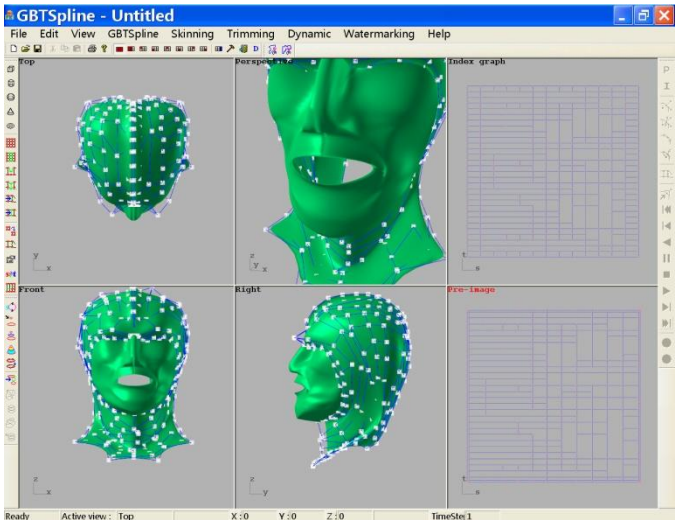


Fig. 5. The screen snap shot of the T-spline editing tool.

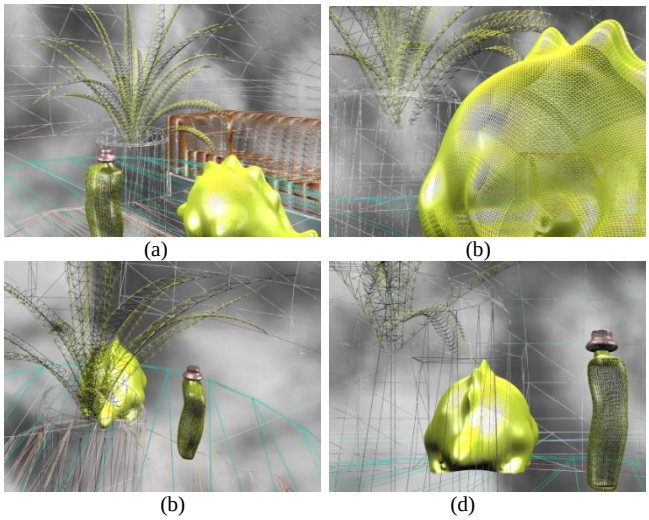
Fig. 6. Two models exported from our tool: (a) a bottle model, and (b) a superman model.

For a T-spline model consisting of several T-spline surfaces, the model can be presented by using a T-spline node for each T-spline surface. Thus, the complex shape in VRML may

contain several children T-spline nodes. The superman model in Fig 6(b) contains two T-spline surfaces.

To represent a virtual world, a VRML file usually contains a large number of geometry nodes, each of which can be a shape in the scene. Each shape can be stored in a single VRML file and be included into the main VRML file using a VRML ***Inline*** node. As the scene become complex, the file size may become huge and prevent the scene to be transmitted in real-time. This problem can be solved by using T-spline VRML nodes to reduce the file size while presenting a same virtual world. Fig 7 is a virtual room, where a bottle model, a toy model, and a plant model are stored in separated files. The bottle model on the table and the steg toy model on the floor use T-spline nodes. Tessellation parameters 5×5 are assigned to both models. The plant model on the floor is presented in a ***IndexedFaceSet*** node. In this room, the VRML file sizes for the toy model, the bottle model and the plant model are 46KB, 14KB and 533KB, respectively. The file size of the plant model is much bigger than the two T-spline models. In the close views with wire frames shown in Fig 8, the density of polygons for the plant model is lower than that of the tessellated T-spline surfaces. Thus, T-spline node can provide more details with smaller file size.

Fig. 7. A virtual room（Color Plate 6）



(a) (b)

(b) (d)

Fig. 8. Close views of the models in wire frames: (a) the first view, (b) zoom in to an area, (c) the second view, and (d) zoom in to an area.

4.2 Level of Detail

LOD techniques increase the rendering efficiency by reducing the visual quality of the model, which should be unnoticed under certain distance. There are two ways to support T-spline LOD: the first way is to use a VRML LOD node with normal tessellations and the second one is using the dynamic tessellation.

In normal tessellations, the two tessellation parameters, ***sTess*** and ***tTess*** can be used directly for LOD controls. A VRML LOD node can have multiple child nodes, each of which represents a same T-spline surface with different tessellation parameters (Fig 9). Only one appropriate child node will be chosen for rendering based on the distance from the user. Usually, one child T-spline node uses tessellation parameters for a fine representation (Fig 9(e)). Another child T-spline node will decrease the tessellation parameters for rough representation (Fig 9(c, d)). Thus, changing the tessellation parameters is enough to implement LOD. One advantage of this method is that users can specify the LOD node. The disadvantage is that all children with different parameters are tessellated as soon as they are loaded.

In Fig 9, the dimension for the bottle is $\boldsymbol{numRow}\times\boldsymbol{numCol}=14\times13$. The number of points on the T-mesh is 182. Only 144 of these points are T-points. With two tessellation parameters equal to α , different tessellated surfaces are obtains and the numbers of vertices and triangles are collected in Table II. For positive α , increasing α provides more details (Fig 9(d,e)). In another way, for uniform tessellation using negative α , more details will be provided only when α is decreased (Fig 9(a,b)). In applications, negative values are used when the object is far from the user. Positive values will be used for close views.

In case $\boldsymbol{sTess}\times\boldsymbol{tTess}<0$, a dynamic tessellation will be perform to support LOD. Like the NURBS LOD, a T-spline surface can be scaled down according to the viewer's distance from the object. We adopt the rendering quality as triangles per screen size used for NURBS nodes [3]:

$$Quality=\frac{triangles(sT,tT)}{bbox(dist)}, \tag{11}$$

where

sT,tT : tessellation parameters satisfying $sT\cdot tT>0$.

$triangles(sT,tT)$: the number of triangles using parameters (sT,tT), which can be derived from Eq.(8) and Eq.(9).

$bbox(dist)$: diameter of the bounding box in screen space.

$Quality$: the rendering quality of an object.

The dynamic tessellation is to select proper parameters such that

$$\frac{triangles(sT,tT)}{bbox(dist)} \geq |\,\boldsymbol{sTess}\,|. \qquad (12)$$

Eq.(12) can be achieved for both the uniform tessellation and the semi-uniform tessellation.

If $\boldsymbol{sTess} < 0$, semi-uniform tessellation is adopted and the parameters are initialized as $sT = tT = 1$. According to Eq.(8), increase sT or tT by 1 in turns, until

$$sT \cdot tT \geq \frac{bbox(dist)\cdot |\,\boldsymbol{sTess}\,|}{2\cdot(\boldsymbol{numRow}-3)\cdot(\boldsymbol{numCol}-3)}.$$

Similarly, if $\boldsymbol{sTess} > 0$, the parameters are initialized as $sT = tT = -2$ for a uniform tessellation. Based on Eq.(9), decrease sT or tT by 1 in turns, until

$$sT \cdot tT \geq \frac{bbox(dist)\cdot |\,\boldsymbol{sTess}\,|}{2}.$$

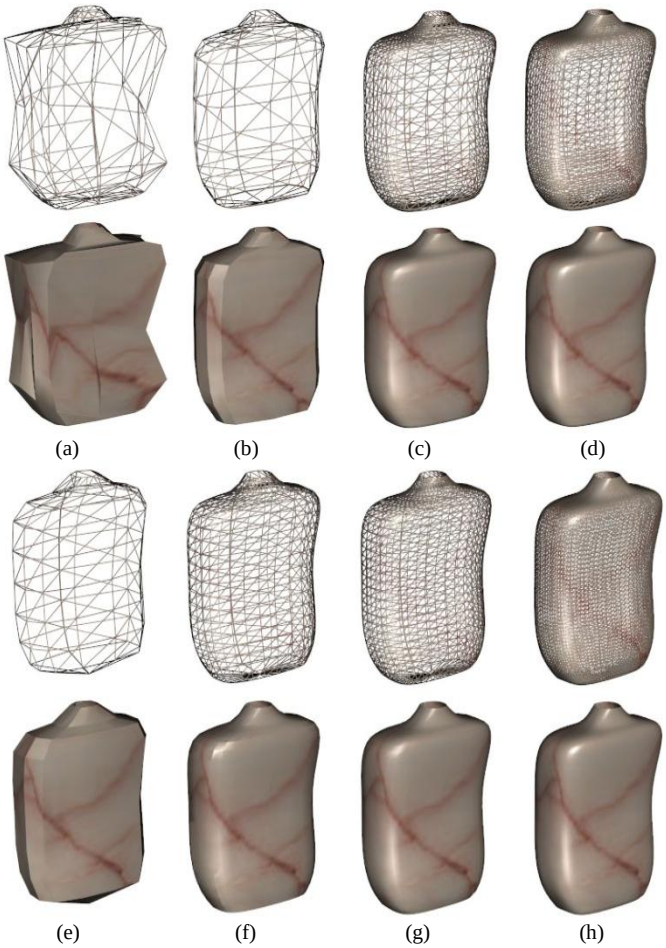


Fig. 9. Different versions of a bottle used in a LOD node: (a) $\alpha = -10$, (b) $\alpha = -22$ (c) $\alpha = -31$, (d) $\alpha = -63$ (e) $\alpha = 0$, (f) $\alpha = 2$, (g) $\alpha = 4$ and (h) $\alpha = 6$.

TABLE. II. STATISTICS FOR THE T-SPLINE SURFACE IN FIG 9.

	Number of vertices	*Number of triangles*
Fig 9 (a)	4096	7938
Fig 9 (b)	1024	1922
Fig 9 (c)	484	882
Fig 9 (d)	100	162
Fig 9 (e)	144	236
Fig 9 (f)	483	880
Fig 9 (g)	1054	1980
Fig 9 (h)	4087	7920

4.3 Texture

The texture coordinates (u_i, v_i) for each T-point P_i are provided in ***texCoord***. Different coordinates lead to different textured surfaces. In Fig 10(a), ***texCoord*** is empty. By default, texture coordinates are automatically generated. The eye of the texture is not correctly mapped to the geometric shape. In Fig 10(b), the texture coordinates for those T-points near the head region is modified. The eye texture is moved to a proper position. Due to the T-spline local editing property, such modification will only affect the texture mapping on the head region. Texture mapping of the body region and tail region remains unchanged in Fig 10(a) and Fig 10(b).

Fig. 10. T-spline surfaces with textures: (a) a T-spline node with empty ***texCoord***, and (b) a T-spline node with customized ***texCoord***.

Currently, the texture mapping is supported by using ***texCoord***. Texture coordinates are assigned for each T-point. This means that a texture surface is defined using the same preimage as the 3D T-spline surface. However, for a complex surface with a huge number of T-points, it is not easy to assign texture coordinates for each T-point and at the same time match the texture with geometry. One solution is to define the texture surface as a separated 2D T-spline surface. This can help to reduce the number of texture coordinates.

4.4 Animation

T-splines can be simply animated by alteration of one or more T-points. Modification of T-points can be done from a javascript. For geometry animation, we only need to modify the coordinates (x, y, z, w) of the selected T-points. Fig 11(b-d) show a geometry animation by modifying three T-points. On the other hand, texture animation can be achieved by adjusting

the texture coordinates (u,v) for the selected T-points. Fig 11(e-h) are the top view and a side view of the tessellated surface in Fig 11(a) by assigning a texture. Using the same texture, without changing any geometry coordinates; a texture animation is performed by changing texture coordinates of only three T-points. Fig 11(e-h) are different frames of the texture animation. From the side view of the surfaces, we can see that the geometry shape does not change during the texture animation.

Animations with T-splines are simple and effective. One benefit using T-splines for animation is that the animated T-spline surface will automatically keep its geometric smoothness. During the animation, if the knot vectors, ***sKnots*** and ***tKnots***, and the tessellation parameters, ***sTess*** and ***tTess***, keep fixed, the sampling grid on the domain and the values of the basis functions at the sampling grid will also keep fixed thus could be cached for reuse during an animation.

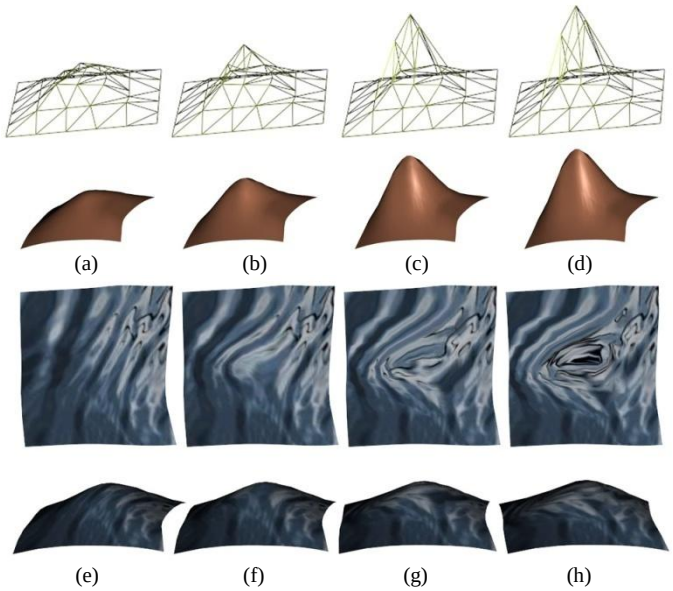


Fig. 11. Animations on T-spline nodes : (a) a T-spline surface and the T-mesh with 45 T-points, (b-d) the T-mesh and the surface during an animation, (e) a T-spline surface with a texture, and (f-h) the textured surface after alteration of texture coordinates u,v.

4.5 Interaction

Users can interact with a T-spline surface in the same way interacting with standard VRML nodes. Besides, a special interaction with a T-spline node is to show the relationship between the surface and its T-mesh. This can also be useful to show the difference between a T-mesh and the control mesh of a NURBS. Fig 12(a) is a T-spline surface which is grouped with a VRML TouchSensor node. When the sensor receives a isOver event, the T-spline surface will switch to show the T-mesh (Fig 12(b)). Our T-spline node can also represent a NURBS surface. The T-spline surface can be transformed into a NURBS surface (Fig 12(d)). In this example, the control mesh of the NURBS surface seems to be more regular than the T-mesh. However, a T-spline surface requires less control points. The NURBS surface contains 774 control points, but the T-mesh consists of only 598 T-points.

V. CONCLUSION AND FUTURE WORKS

This paper proposes and implement the T-spline VRML node. Users can design a complex model with relatively small number of control points and visualize on the web. A new data structure using point grid has been proposed to represent T-spline surface in VRML nodes. Similar with the NURBS based and polygon based VRML nodes, T-spline VRML node can be easily edited and exported using our editing tools. The proposed T-spline node can work together with traditional VRML nodes to create a complex scene. Fundamental nodes such as texture, LOD and animations have been supported by using T-spline node. Compared with NURBS based and polygonal based VRML nodes, T-spline VRML node can minimize the size of the VRML file at the same time represent a same shape. The implementation details and the application examples are discussed.

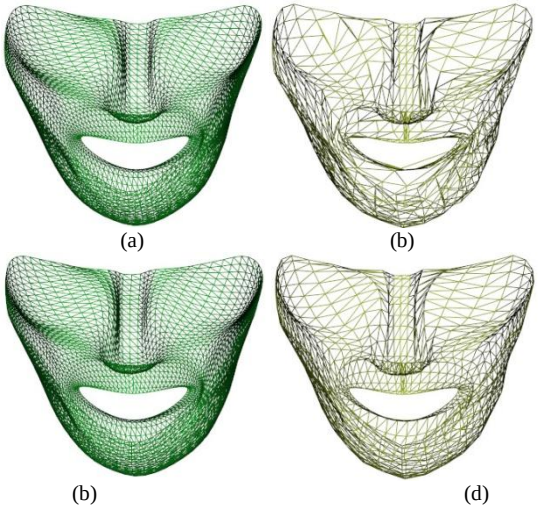


Fig. 12. Interaction on T-spline nodes : (a) a tessellated T-spline surface with 15276 vertices, (b) the T-mesh with 598 T-points, (c) a tessellated NURBS surface with 15276 vertices, and (d) the NURBS control mesh with 774 control points.

In the future, we will provide the ***texCoord*** as a new 2D T-spline, which can have a different structure (including T-mesh and knots) from the corresponding 3D T-spline surface. A new VRML node to support T-spline based freeform deformation (FFD) will be developed. X3D will also be extended to support T-splines.

ACKNOWLEDGEMENT

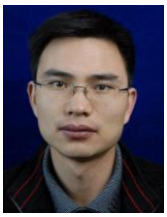
This work is supported by the ARC 9/09 Grant (MOE2008-T2-1-075) of Singapore.

REFERENCES

[1] L. Daly and D. Brutzman, "X3D: extensible 3D graphics standard," in *ACM SIGGRAPH ASIA 2008 courses*, ed, 2008, pp. 1-6.
[2] L. A. Piegl and W. Tiller, *The NURBS book*: Springer Verlag, 1997.
[3] H. Grahn, *et al.*, "Nurbs in VRML," in *Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML)*, ed, 2000, pp. 35-43.
[4] Bs. *BS Contact VRML*, *http://www.bitmanagement.com*.
[5] Cv. *Cortona3D Viewer*, *http://www.cortona3d.com*.
[6] T. W. Sederberg, *et al.*, "T-spline simplification and local refinement," *ACM Transactions on Graphics*, vol. 23, pp. 276-283, 2004.
[7] T. W. Sederberg, *et al.*, "T-splines and T-NURCCs," *ACM Transactions on Graphics*, vol. 22, pp. 477-484, July 2003.
[8] C. Y. Yu, *et al.*, "Combining Java with VRML worlds for Web-based collaborative virtual environment," in *IEEE Networking, Sensing and Control*, ed, 2005, pp. 299-304.
[9] Q. Liu and A. Sourin, "Analytically-defined collaborative shape modeling in VRML," in *IEEE International Conference on Cyberworlds*, ed, 2004, pp. 70-77.
[10] E. Fogel, *et al.*, "A web architecture for progressive delivery of 3D content," in *Proceedings of the sixth international conference on 3D Web technology*, ed, 2001, pp. 35-41.
[11] M. Isenburg and J. Snoeyink, "Coding polygon meshes as compressable ASCII," in *Proceedings of the seventh international conference on 3D Web technology*, ed, 2002, pp. 1-10.
[12] Y. Wang and J. Zheng, "Control point removal algorithm for T-spline surfaces," *Geometric Modeling and Processing*, pp. 385-396, 2006.
[13] J. Zheng, *et al.*, "Adaptive T-spline surface fitting to z-map models," in *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ed, 2005, pp. 405-411.
[14] Y. Wang and J. Zheng, "Adaptive T-spline surface approximation of triangular meshes," in *IEEE 6th International Conference on Information, Communications & Signal Processing*, ed, 2007, pp. 1-5.
[15] Y. Wang, *et al.*, "Conversion between T-splines and hierarchical B-splines," in *Proceedings of computer graphics and imaging*, ed, 2005, pp. 8-13.
[16] B. Roehl and others, *Late night VRML 2.0 with Java*: Ziff-Davis Press, Emeryville, Calif, 1997.
[17] D. Brutzman, "The virtual reality modeling language and Java," *Communications of the ACM*, vol. 41, pp. 57-64, 1998.
[18] Ip. *Instant Player*, *http://www.instantreality.org/*.

Wenyu Chen is a research fellow in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include Geometric Modeling, Complex Surface Modeling and Interactive Virtual Reality.

Yusha Li is a PhD student in the School of Computer Engineering at Nanyang Technological University, Singapore. Her research mainly focuses on geometric modelling using T-spline.



Jianjiang Pan is an associate professor in the School of Science at Hangzhou Dianzi University, China. He was a research fellow in the School of Computer Engineering at Nanyang Technological University, Singapore from 2011 to 2012. His research interests include image processing and computer aided geometric design.

Jianmin Zheng is with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interest includes computer aided geometric design, CAD/CAM, computer graphics, animation, digital imaging and visualization.

Yiyu Cai is associate professor with the School of Mechanical & Aerospace Engineering, Nanyang Technological University, Singapore. His research interests include VR, Computational Biomedical Sciences, and Interactive & Digital Media.