Smooth Force Rendering on Coarse Polygonal Meshes

Jun Wu, Yuen-Shan Leung, Charlie C.L. Wang[*], Dangxiao Wang, and Yuru Zhang

Abstract

Piecewise linear polygonal model has only G^0 continuity, thus users can easily feel the edges when using haptic device to touch a solid represented by coarse polygonal meshes. To produce an appealing haptic sensation for smooth solids, a large number of polygons are needed in conventional approaches. This however slows down computation and consumes much more memory. In this paper, we present a method to generate smooth feedback force in haptic interaction with coarse polygonal meshes. Our method calculates the interaction force based on Gregory patches, which are locally constructed from n-sided polygons and ensure G^1 continuity across boundaries of patches. During the real time haptic interaction, the contact point is continuously tracked on the locally constructed Gregory patches and thus generates smooth haptic forces to be rendered. Our method is validated on various models with comparison to conventional force rendering techniques.
Keywords: force rendering, Gregory patch, G^1 continuity, coarse mesh, n-sided polygon

1 Introduction

In computer haptics, the geometry model of an object is usually represented by polygonal meshes (i.e., G^0 piecewise linear surfaces). When applying the penetration-depth based force rendering method on coarse polygonal meshes, discontinuous feedback force will be produced if the contact point slips across the edge – this is described simply as *edge effect* below. Although force shading [1] (similar to Phong shading in visual rendering) and its variants [2, 3] can be applied to weaken this kind of edge effect, some researchers (ref. [4]) found that the force discontinuity caused by the G^0 shape of polygonal model can still be perceived by users. This will badly affect human performance in some specific applications (e.g., [3, 5]).

In conventional approaches, a large number of polygons are needed to produce an appealing haptic sensation. This consumes more memory and decreases computational efficiency. For example, a virtual environment of a whole set of teeth (see Fig.1) in medical simulation needs more than tens of thousands of triangles to make the force rendering feel smooth (i.e., realistic).

To eliminate edge effect completely and thus produce smooth feedback force, we present a penetration-depth based force rendering method on Gregory patches, which are constructed locally from the input polygonal meshes and have the property of G^1 continuity. The feedback force is directly computed on the patches. The main advantages of our method are as follows.

Smooth feedback force This is obvious as our method directly computes the feedback force based on G^1 continuous surfaces.

Compact representation For curved surfaces with high curvature, a massive number of piecewise linear meshes are needed to approximate the solid in a haptically appealing manner. However, as the Gregory patch is a high order polynomial, the number of basic meshes needed to represent a smooth object can be greatly reduced while still producing a smooth geometry, and

[*]Correspondence to: Charlie C. L. Wang, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong, China. E-mail: cwang@mae.cuhk.edu.hk

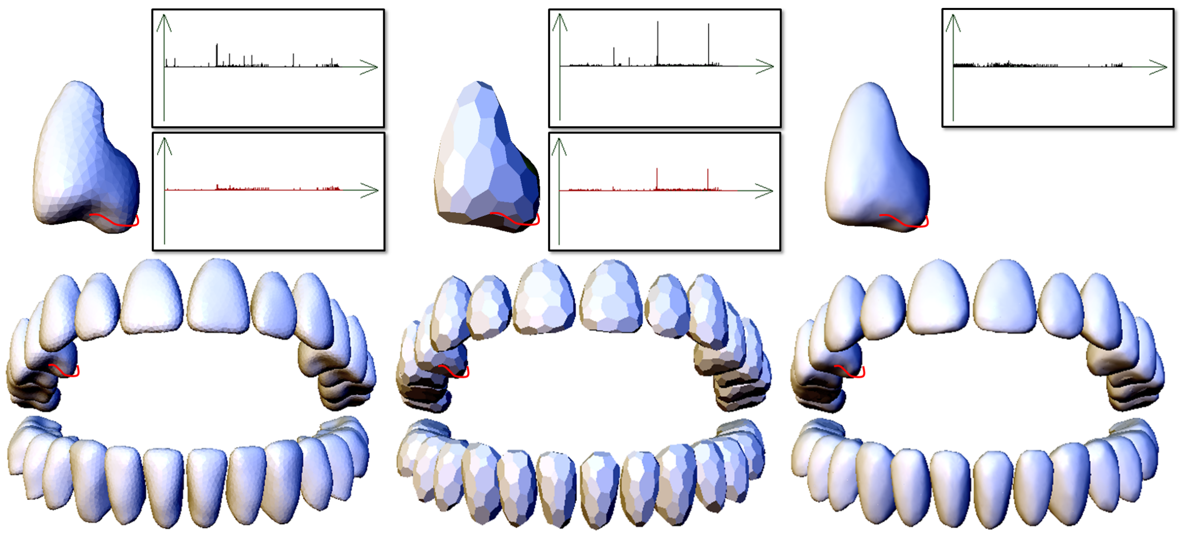


Figure 1: The example of teeth model in the dental contact simulation: (left) the polygonal mesh with 39,920 triangles, (middle) the coarse polygonal mesh with only 1,805 polygons, and (right) the patches generated by our method from the coarse polygonal mesh. The upper bar-graphs show the magnitude of difference vectors between consecutive forces during the simulation. Here, the contact point of haptic device is moved across the surface of a tooth along the red trajectory. The top row of bar-graphs (in black) are generated using direct rendering [6], and the second row of bar-graphs (in red) are measured on the forces generated by the force shading method [1]. Differently, our method can generate very smooth force on very coarse mesh surfaces.

thus resulting in smooth feedback force.

Local construction The construction of a Gregory patch for a mesh is totally based on local information (i.e., the position and normal of vertices on its corresponding polygon). Therefore, efficient computation and memory usage can be achieved.

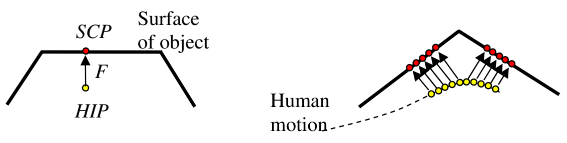
n-sided polygon input The patch construction is based on blending of the corner interpolation functions defined on polygons. It is unlike other Gregory patch construction methods that are restricted to quadrangles (e.g., [7]) or triangles-and-quadrangles (e.g., [8]). The input mesh model can have n-sided polygons for any $n > 2$, and we avoid the problem of inserting lines to split polygons to triangles and/or quadrangles.

2 Related Work

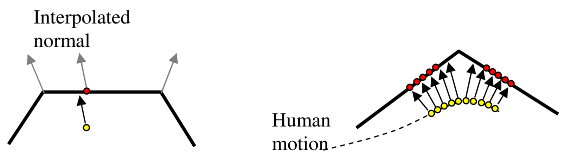
2.1 Force rendering

Polygonal mesh is the most prevalent geometry representations in computer graphics and receives the widest attention in the haptics research, ranging from point-object [6], line-object [9, 10], to object-object interaction [11, 12]. Zilles et al. [6] proposed the penetration-depth based implementation called god-object method for point-object interaction, where the user's position is represented by a single point. For convenience, the point representing actual human motion is denoted by haptic interface point (HIP) as it is obtained through the haptic interface. The god-object which stays on the surface of the solid is denoted by surface contact point (SCP). SCP is typically a local nearest point from the surface to the HIP. The feedback force is calculated based on the difference vector between these two points. As shown in Fig. 2(a), when the HIP slips across the edge, this method will generate discontinuous feedback force as the polygon model is only G^0 continuous.

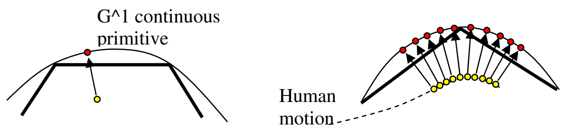
This problem of edge effect was first addressed by Morgenbesser et al. [1] using a method similar to the Phong shading in visual rendering (see Fig. 2(b)). The shading method is computationally fast and can partly eliminate



(a) The basic implementation of direct force rendering [6]. Force vector points from the haptic interface point to the nearest point on the surface.



(b) Force shading [1] is similar to Phong shading. The direction of force is altered to the smoothed normal at the nearest point.



(c) Our patch based force rendering. The feedback force is pointing to a nearest point on the locally constructed G^1 continuous patch.

Figure 2: Illustration of different force rendering approaches.

the edge effect. However, in this approach, the mapping from HIPs to the SCPs is still not continuous, thus modifying the force direction cannot fundamentally solve the force discontinuity problem, especially when the penetration depth is large and the corner is very sharp. Human evaluation results [4] show that users can still feel the edge effect even after using the force shading method. A variation of the force shading was proposed by Ho et al. in [2], and another improvement for convex solids was introduced by Ruspini et al. in [13]. Neither of these variations fundamentally solves the problem as they still rely on the G^0 continuous shape.

Volumetric model such as distance field [14] provides a straightforward way to perform collision detection and force calculation based on the penetration-depth/volume of two objects. Recently, Li et al. [15] adopted the C^1 continuous distance field for point-solid interaction. However, the computation of distance field (especially the one with C^1 continuity) is time-consuming. Moreover, the shape that can be presented by a distance field is also restricted by its resolution.

Force rendering based on point set surface was introduced by Lee et al. in [16]. Guo et al. [17] also proposed a modeling system based on a combination of point set surface and dynamic physics-based sculpting. The point set surface benefits from no explicit topology information but meanwhile makes it difficult to extend the method to more complicated interaction scenarios.

Parametric representation such as NURBS has also been applied in haptics (e.g., [18] by Thompson et al.). Converting existing prevalent polygon models to NURBS needs global parametrization which is not a trivial task. Our method locally constructs the parametric surface (i.e., Gregory patch) for every polygon efficiently and is suitable for parallelization and on-site construction.

The proposed approach addresses the problem of smooth force rendering from the geometry aspect. An expectation is that the generated Gregory patches model is more realistic than the base coarse polygon model with respect to the original object. Some other approaches, such as threshold filter, and virtual coupling [19] which acts as a kind of filter in some way, may also help to generate smoother force signal than directly on polygonal meshes. However, the realism after the filter needs further investigation. Moreover, the non-linear property of the filters makes the tuning of parameters herein not a trivial task.

2.2 Gregory patch

To construct a G^1 continuous surface for each polygon, Gregory corner interpolator function must be constructed for every corner of an n-sided polygon at first. Then, the final surface for this polygon is blended as a weighted sum of the n functions.

As illustrated in Fig.3, let $P(u)(0 \leq u \leq 1)$ and $Q(v)(0 \leq v \leq 1)$ be two regular curves in $\Re^3$ with $P(0) = Q(0)$, and $T_P(u)(0 \leq u \leq 1)$ and $T_Q(v)(0 \leq v \leq 1)$ be two C^1 vector functions in $\Re^3$ satisfying

$$T_P(0) = Q'(0),\ T_Q(0) = P'(0), \tag{1}$$

the Gregory corner interpolator of the four functions, $P(u)$, $Q(v)$, $T_P(u)$ and $T_Q(v)$, is a sur-

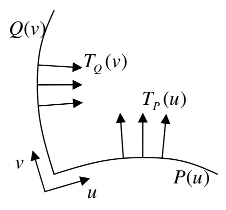


Figure 3: Gregory corner interpolator.

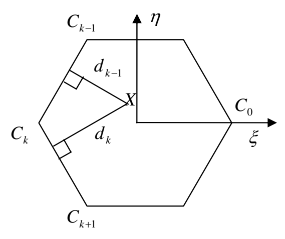


Figure 4: Parametric domain of a Gregory patch.

face in $\Re^3$ defined by

$$\begin{aligned} r(u,v) &= -P(0) - vT_P(0) - uT_Q(0) \\ &+P(u) + vT_P(u) + Q(v) + uT_Q(v) \\ &-uv(vT_P'(0) + uT_Q'(0))/(u+v). \end{aligned} \quad (2)$$

The Gregory corner interpolator function $r(u,v)$ agrees with $P(u)$ and $Q(v)$ along the two sides (i.e., $r(u,0) = P(u)$ and $r(0,v) = Q(v)$). Also, its partial derivatives with respect to u and v agree with $T_P(u)$ and $T_Q(v)$ along the respective sides as $r_u(u,0) = T_P(u)$ and $r_v(0,v) = T_Q(v)$.

After setting up the corner interpolator for every corner of a polygon, the next step is to blend them to form a parametric surface. The parametric domain of a Gregory patch, denoted as P_G, with n sides is defined as a regular n-gon in a unit circle in the $\xi - \eta$ domain (as shown in Fig. 4). The corners $C_k(k = 0, 1, ..., n-1)$ are placed in the anti-clockwise order. Given a point $X(\xi, \eta)$ in the parametric space, when computing its position defined by the k-th Gregory corner $r_k(u_k, v_k)$ the parameters (u_k, v_k) of the point are defined as

$$(u_k, v_k) = \left(\frac{d_{k-1}}{d_{k-1} + d_{k+1}}, \frac{d_k}{d_{k-2} + d_k} \right), \quad (3)$$

where d_k is the perpendicular distance from X to the edge $C_k C_{k+1}$. The corner C_k is defined by $C_k = (\cos \frac{2k\pi}{n}, \sin \frac{2k\pi}{n})$, where n is the number of corners and $k = 0, 1, ..., n-1$.

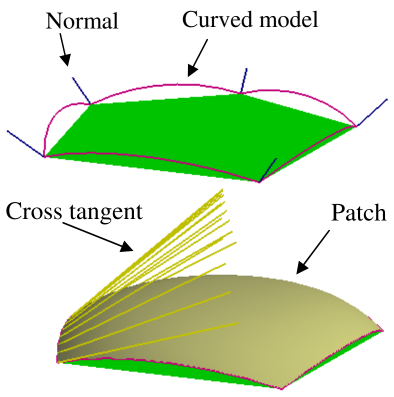


Figure 5: Gregory patch of a 5-sided polygon

The Gregory patch for a n-sided polygon is defined as a mapping from the parametric domain P_G to $\Re^3$ by

$$G(X) = \sum_{k=0}^{m-1} w_k(X) r_k(u_k(X), v_k(X)) \quad (4)$$

where the weight function is

$$w_k(X) = \frac{\prod_{j \neq k-1,k} d_j^2}{\sum_{l=0}^{j-1} \prod_{j \neq l-1,l} d_j^2}. \quad (5)$$

3 Local Construction of Gregory Patch

To construct the patches that are joined together with G^1 continuity, the derivatives of two neighboring patches sharing a common curve must be coplanar with the tangent of the common curve (i.e., G^1 continuity across the boundary). The tangent vectors of curves joining to a vertex on the given model must also be located on the same plane (i.e., G^1 continuity is preserved at the vertices of given coarse polygonal meshes). Moreover, the construction of boundary curves and cross tangent functions is expected to be localized. An example of 5-sided Gregory patch constructed from a polygon is shown in Fig.5.

3.1 Curve modeling

The curve on a polygonal edge with two vertices $\mathbf{v}_s\mathbf{v}_e$ is defined by a Hermite curve

$$\mathbf{C}(t) = (2t^3 - 3t^2 + 1)\mathbf{v}_s + (-2t^3 + 3t^2)\mathbf{v}_e + L(t^3 - 2t^2 + t)\mathbf{t}_s + L(t^3 - t^2)\mathbf{t}_e \tag{6}$$

where $t \in [0, 1]$, $L = \|\mathbf{v}_s\mathbf{v}_e\|$ is the length of edge $\mathbf{v}_s\mathbf{v}_e$, and $\mathbf{t}_s$ and $\mathbf{t}_e$ are tangent vectors of the Hermite curve presented at $\mathbf{v}_s$ and $\mathbf{v}_e$ respectively. To ensure the G^1 continuity at vertices, the tangent vectors $\mathbf{t}_s$ and $\mathbf{t}_e$ must satisfy that $\mathbf{t}_s \cdot \mathbf{n}_{v_s} = 0$ and $\mathbf{t}_e \cdot \mathbf{n}_{v_e} = 0$ with $\mathbf{n}_{v_s}$ and $\mathbf{n}_{v_e}$ being the vertex normals defined at $\mathbf{v}_s$ and $\mathbf{v}_e$. To satisfy these constraints, we define them by

$$\mathbf{n}_{avg} = (\mathbf{n}_{v_s} + \mathbf{n}_{v_e})/2, \tag{7}$$

$$\mathbf{t}_s = \frac{\mathbf{n}_{v_s} \times ((\mathbf{v}_s\mathbf{v}_e) \times \mathbf{n}_{avg})}{\|\mathbf{n}_{v_s} \times ((\mathbf{v}_s\mathbf{v}_e) \times \mathbf{n}_{avg})\|}, \tag{8}$$

$$\mathbf{t}_e = \frac{\mathbf{n}_{v_e} \times ((\mathbf{v}_s\mathbf{v}_e) \times \mathbf{n}_{avg})}{\|\mathbf{n}_{v_e} \times ((\mathbf{v}_s\mathbf{v}_e) \times \mathbf{n}_{avg})\|}. \tag{9}$$

3.2 Cross tangent modeling

The cross tangent functions must be C^1 continuous and satisfy Eq.(1). To guarantee the G^1 continuity across the boundary curve defined above, the cross tangents on two sides of $\mathbf{C}(t)$ should be coplanar with $\mathbf{C}'(t)$. Therefore, we define a function to specify the normal vector of the reconstructed surface at any point of the common boundary curve. The defined normal must be perpendicular to $\mathbf{C}'(t)$ and interpolates $\mathbf{n}_{v_s}$ and $\mathbf{n}_{v_e}$ at $\mathbf{v}_s$ and $\mathbf{v}_e$.

The normal of the reconstructed surface on the Hermite curve $C(t)$ is defined by

$$\mathbf{n}_c(t) = \mathbf{C}'(t) \times (((1-t)\mathbf{n}_{v_s} + t\mathbf{n}_{v_e}) \times \mathbf{v}_s\mathbf{v}_e). \tag{10}$$

The direction of the cross tangent perpendicular to $\mathbf{n}_c(t)$ can then be expressed as

$$\mathbf{t}_c(t) = \mathbf{n}_c(t) \times (\mathbf{v}_{int}(t) \times \mathbf{n}_{int}(t)) \tag{11}$$

with

$$\mathbf{v}_{int}(t) = (1-t)\mathbf{v}_s^- + t\mathbf{v}_e^+ - \mathbf{C}(t), \tag{12}$$

$$\mathbf{n}_{int}(t) = \frac{1}{2}((1-t)\mathbf{n}_{v_s^-} + t\mathbf{n}_{v_e^+} + \mathbf{n}_c(t)). \tag{13}$$

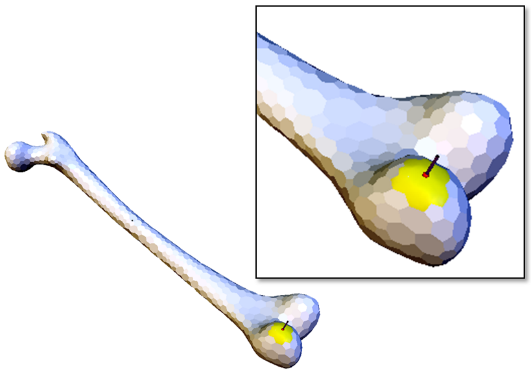


Figure 6: The Gregory patches are only constructed at the region near the contact point – the yellow region. Note that, the flat shading is employed to display (visually) the polygons, while in real applications the Phong shading will be used to improve the visual rendering.

Here $\mathbf{v}_s^-$ denotes the previous vertex of $\mathbf{v}_s$ and $\mathbf{v}_e^+$ represents the next vertex of $\mathbf{v}_e$ on the polygon. $\mathbf{n}_{v_s^-}$ and $\mathbf{n}_{v_e^+}$ are vertex normals defined on $\mathbf{v}_s^-$ and $\mathbf{v}_e^+$ respectively. Considered that the magnitude of the cross tangent should satisfy Eq.(1), we have the cross tangent function defined on $\mathbf{C}(t)$ as

$$\mathbf{T}_C(t) = ((1-t)L_{prev} + tL_{next})\frac{\mathbf{t}_c(t)}{\|\mathbf{t}_c(t)\|}, \tag{14}$$

where L_{prev} and L_{next} are the lengths of the edges $\mathbf{v}_s^-\mathbf{v}_s$ and $\mathbf{v}_e\mathbf{v}_e^+$. It is easy to prove that $\mathbf{T}_C(t)$ and $\mathbf{C}'(t)$ are perpendicular to $\mathbf{n}_c(t)$, and we have

$$\mathbf{T}_C(0) = L_{prev}\frac{\mathbf{t}_c(0)}{\|\mathbf{t}_c(0)\|} = \mathbf{C}'_{prev}(1)$$

and

$$\mathbf{T}_C(1) = L_{next}\frac{\mathbf{t}_c(1)}{\|\mathbf{t}_c(1)\|} = \mathbf{C}'_{next}(0)$$

so that the constraints in Eq.(1) are satisfied.

4 Patch-based Force Rendering

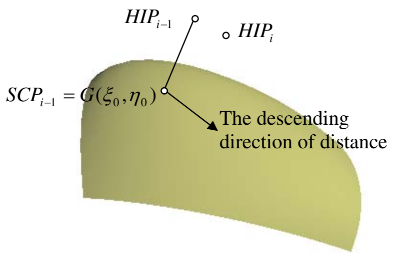
The avatar of the user (the end effector of haptic device) is expressed as a single point, and the object being touched is represented by Gregory patches that are locally constructed from polygonal meshes. The status of the relationship between the avatar point and the object is classified as contact status and non-contact status. The contact status starts when the avatar point penetrates into the object and the status holds until the avatar point leaves the object. Feedback force is calculated based on the penetration-depth. The simulation flow is given in the Algorithm 1. Note that, during the simulation, the Gregory patches are only constructed at the region near the avatar (see Fig.6 for an example).

Algorithm 1 Force rendering loop

```
1: if status == non-contact then
2:     Search the global nearest point
3:     Determine whether penetration
4:     if penetration then
5:         status = contact
6:         contact point = the global nearest point
7:         Calculate the feedback force
8:     else
9:         status = non-contact
10:    end if
11: else
12:    Track the local nearest point around contact point
13:    Determine whether still penetration
14:    if penetration then
15:        status = contact
16:        contact point = the local nearest point
17:        Calculate the feedback force by SCP-HIP
18:    else
19:        status = non-contact
20:    end if
21: end if
```

4.1 *Initialization*

Given a point P, and a surface S composed of patches, we need to find the nearest point on the surface to the point. Analytic method of nearest point query for a parametric surface is heavy in computation [20]. Our method determines the initial location of the nearest point using an approximate geometry model, and refines it numerically. For the approximate model for initial query, we simply employ the base polygons of the object. There are three steps in this initialization step. First, we need to find the global nearest point on the polygons. Many algorithms have been developed to solve this problem, such as PQP [21] we used. Second, we locally construct a Gregory patch for this polygon using the method described in last section. Third, we map the point on the polygon to a point on the patch. This point on the patch then serves as the initial global nearest point on the patches.



HIP_{i-1}
HIP_i
$SCP_{i-1} = G(\xi_0, \eta_0)$
The descending direction of distance

Figure 7: 3D illustration of the refinement step.

4.2 *Local nearest point refinement*

When the user slides along the surface, the simulator should try to find the local nearest point on the surface to the avatar and calculate the feedback force. The local nearest point, rather than the global nearest point, is adopted in haptic rendering because it prevents the undesired pop-through effect on thin objects. Therefore, given an initial surface contact point SCP_{i-1} on the patch and the current position of the avatar point HIP_i, we try to find a local nearest point SCP_i in the surface to the point HIP_i.

The problem is solved in an iterative manner. As our surface is a parametric one, we could search around the SCP_{i-1}'s parameter coordinate (ξ_0, η_0) locally. As illustrated in Fig. 7, the method is to project the HIP_i to the tangent plane of the point SCP_{i-1}, and find the descending direction of distance by numerical differences, (α, β), in the parametric domain. The parametric coordinate is moved forward for a certain step Δ in the descending direction to $(\xi_1, \eta_1) = (\xi_0 + \alpha\Delta, \eta_0 + \beta\Delta)$. Then, the new point is $\mathbf{G}(\xi_1, \eta_1)$, where $\mathbf{G}(\cdots)$ is the mapping from parametric domain to $\Re^3$. This process is iteratively performed until the angle between the vector from the point $\mathbf{G}(\xi_j, \eta_j)$ to HIP_i and the normal vector at $\mathbf{G}(\xi_j, \eta_j)$ is smaller than a given threshold (e.g., 10^{-5}), where j is the iterative times. $\mathbf{G}(\xi_j, \eta_j)$ will serve as SCP_i.

When smaller forward iteration step size is applied, more precise result can be obtained by taking the cost of more iteration time. Thus, it

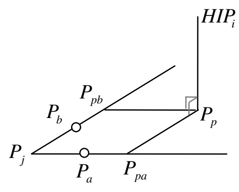


Figure 8: Adaptive forward gap for refining.

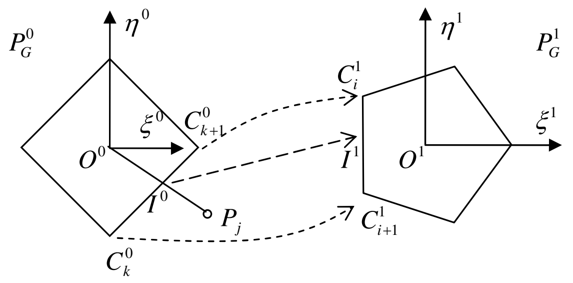


Figure 9: Transition between the parametric domains of two Gregory patches.

is better to choose the step size adaptively. Let $\mathbf{P}_j = \mathbf{G}(\xi_j, \eta_j)$ be the result after j times iteration, $\mathbf{P}_a = \mathbf{G}(\xi_j + \xi_a, \eta_j + \eta_a)$, $\mathbf{P}_b = \mathbf{G}(\xi_j + \xi_b, \eta_j + \eta_b)$ be other two points on the Gregory patch near $\mathbf{P}_j$, we can obtain the point $\mathbf{P}_p$ as the projection of HIP_i on the plane $\mathbf{P}_a\mathbf{P}_b\mathbf{P}_j$ (see the illustration in Fig.8). On the plane, we can find two points $\mathbf{P}_{pa}$ (on line $\mathbf{P}_j\mathbf{P}_a$) and $\mathbf{P}_{pb}$ (on line $\mathbf{P}_j\mathbf{P}_b$) to let $\mathbf{P}_a\mathbf{P}_{pa} \parallel \mathbf{P}_j\mathbf{P}_b$ and $\mathbf{P}_b\mathbf{P}_{pb} \parallel \mathbf{P}_j\mathbf{P}_a$. The step size can then be chosen as $(\alpha\xi_a + \beta\xi_b, \alpha\eta_a + \beta\eta_b)$, where $\alpha = \frac{\|\mathbf{P}_{pa}-\mathbf{P}_j\|}{\|\mathbf{P}_a-\mathbf{P}_j\|}$, $\beta = \frac{\|\mathbf{P}_{pb}-\mathbf{P}_j\|}{\|\mathbf{P}_b-\mathbf{P}_j\|}$. Also, the surface normal at $\mathbf{P}_j$ can be approximated by $\mathbf{P}_a\mathbf{P}_j \times \mathbf{P}_b\mathbf{P}_j$.

The parameters of the above $\mathbf{P}_a$ and $\mathbf{P}_b$ are chosen near that of $\mathbf{P}_j$, but are still in the parametric domain (i.e., inside the regular polygon). This is because a singular surface will be generated if the point in the $\xi - \eta$ domain is outside the regular polygon. We first determine the corner $\mathbf{C}_k$ of the parametric polygon which is closest to (ξ_j, η_j). If there are more than one nearest corner, we randomly select one among them. By choosing $(\xi_a, \eta_a) = \lambda(\mathbf{C}_{k+1} - \mathbf{C}_k)$ and $(\xi_b, \eta_b) = \lambda(\mathbf{C}_{k-1} - \mathbf{C}_k)$ with $\lambda = 0.1$, we can ensure that the new points $(\xi_j + \xi_a, \eta_j + \eta_a)$ and $(\xi_j + \xi_b, \eta_j + \eta_b)$ are in the polygon. This can be easily proved.

4.3 *Transition across boundary*

Although the above method can make the new points near P_j defined in the same parametric polygon, the new point position computed by iteration can still be moved into a new patch (i.e., when users slide across the boundary of two patches). As illustrated in Fig.9, when the computed $\mathbf{P}_j = \mathbf{G}(\xi_j, \eta_j)$ runs out of the parametric domain, the contact point should be transited to its neighboring patch. Suppose that the point $\mathbf{X} = (\xi_j, \eta_j)$ lies in the fan area between $\mathbf{O}^0\mathbf{C}_k^0$ and $\mathbf{O}^0\mathbf{C}_{k+1}^0$ where the superscript denotes the current parametric domain P_G^0, line $\mathbf{O}^0\mathbf{P}_j$ and line $\mathbf{C}_k^0\mathbf{C}_{k+1}^0$ intersect at a point $\mathbf{I}^0$ on the k-th edge of the parametric polygon. If this edge is shared by its neighboring patch whose parametric domain is P_G^1 and the edge is the i-th edge on P_G^1, we can easily map $\mathbf{I}^0$ to $\mathbf{I}^1$ by satisfying the condition $\frac{\mathbf{C}_k^0\mathbf{I}^0}{\mathbf{I}^0\mathbf{C}_{k+1}^0} = \frac{\mathbf{C}_{i+1}^1\mathbf{I}^1}{\mathbf{I}^1\mathbf{C}_i^1}$. Thus, we have

$$\mathbf{I}^1 = \mathbf{C}_i^1 + \frac{\mathbf{I}^0 - \mathbf{C}_{k+1}^0}{\mathbf{C}_k^0 - \mathbf{C}_{k+1}^0}\left(\mathbf{C}_{i+1}^1 - \mathbf{C}_i^1\right). \quad (15)$$

5 Results

We have implemented the proposed approach by C++. OpenGL is adopted for graphical rendering and OpenHaptics is employed as I/O for the haptic device – Phantom Omni from Senable Technology. Our test platform is a PC with an Intel Pentium4 3.00GHz CPU and 2GB main memory (see Fig.10). The PC is equipped with a NVIDIA GeForce 7600 GS graphics card, and runs Windows XP.

The first example shown in this paper is the model of human teeth (see Fig.1), where the fine mesh for a realistic force rendering by conventional force rendering technique has around 40k triangles. However, when using our approach, smooth forces can be generated on a model with only about 1.8k polygons. The second example shown in Fig.11 is a banana model with only about 409 polygons. Similar to Fig.1, the bargraphs of the magnitudes of difference vectors between consecutive forces during the movement are also shown in Fig.11. From the bargraphs, it is easy to find that our method generates the smoothest force when moving the contact point along the same trajectory – the red

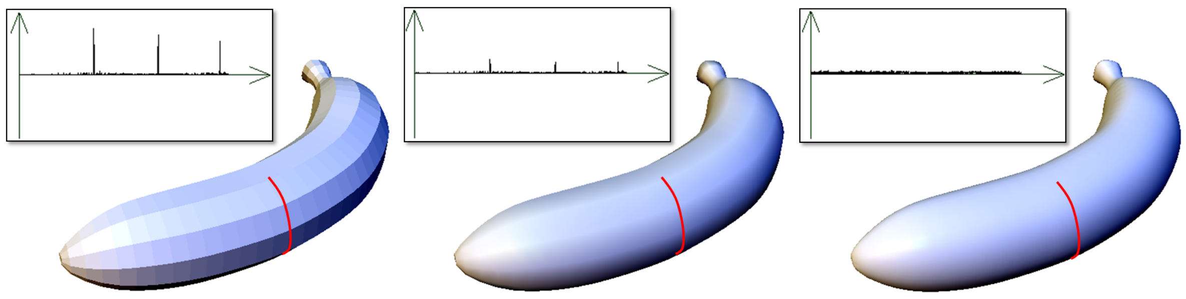


Figure 11: The force generated on a banana model with only 409 polygons using different methods: (left) the direct rendering [6], (middle) the force shading approach [1], and (right) our smooth force rendering by local Gregory patch construction. The models are visualized by (left) flat shading, (middle) Phong shading and (right) the Gregory patches. The red curves show the trajectories of contact point movement.

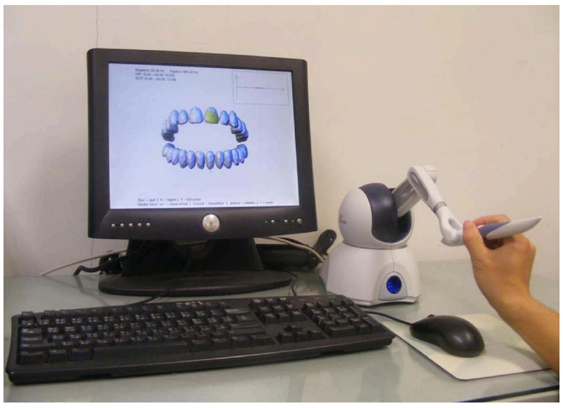


Figure 10: Our test platform with a PC and a Phantom Omni.

curves in Fig.11. To mimic the force rendering results, the banana model is displayed in three ways: flat shading, Phong shading and the Gregory patches. Another interesting test conducted in Fig.12 is to study the force continuity with different penetration depths by using different approaches. A torus knot model with 1,440 quadrangles is adopted for the test. The model is offset inward by two different values: 2mm and 4mm. Points on the offset surface are then used as *HIP* for computing the forces. The results are graphically displayed by point rendering with the normalized force directions serving as normals of points. It is obvious that our method produces the smoothest feedback forces. Moreover, we can easily find that when increasing penetration depth, the force discontinuity is more significant on the results of the direct rendering [6] and the force shading [1] methods.

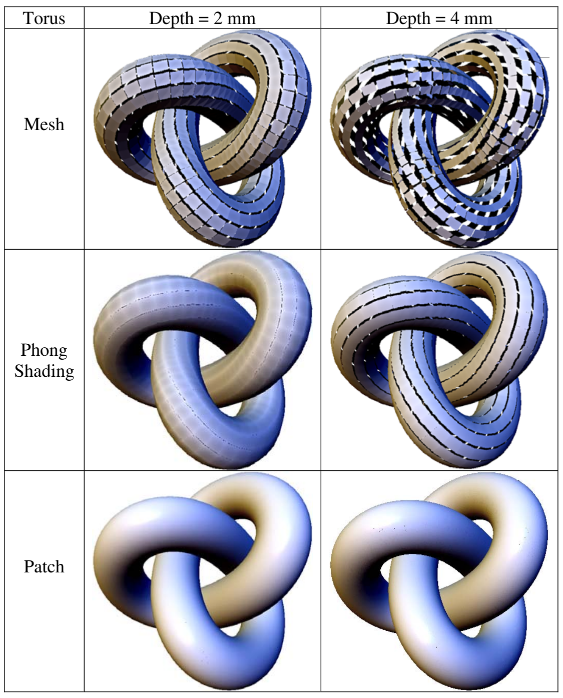


Figure 12: Study of force discontinuity on the torus knot model with 1440 quadrangles using different methods and with different penetration depths. (Top row) direct rendering [6], (middle row) force shading based rendering [1], and (bottom row) our Gregory patch based force rendering. When increasing the penetration depth, more significant force discontinuity can be found on the results of the direct rendering and the force shading. Such discontinuity does not appear in the results of our approach.

Figure 13 shows another similar study on a hand model.

Our approach is very efficient in both computing speed and memory consumption. For the teeth example shown in Fig.1, similar smooth feedback force can be generated on the dense mesh by force shading and the coarse mesh by our patch-based force rendering; however, the memory cost for the dense mesh is around 10 times of that of the coarse mesh. The size of model must be well controlled in the applications which need to send the model through some channels (e.g., network or from main memory to the graphics memory), where data communication is still a bottleneck. Our Gregory patch based force rendering approach is fast. This is because G^1 continuity preserved Gregory patches can be constructed in a local data-access manner from the given polygons and the normal vectors on vertices. The average construction time of a Gregory patch on our PC is around $0.02 - 0.03ms$. The Gregory patches are only constructed near the region of contact point, and only a few patches need to be constructed and stored during the movement of contact point. The update frequency is higher than $1kHZ$, so it satisfies the requirement of continuous force generation on the Haptic device [22]. Moreover, the construction of Gregory patches can be easily parallelized on the multi-core CPUs or GPUs to further speed up the computation.

6 Conclusion

We present a method for smooth force rendering on coarse polygonal meshes in this paper. Our method is based on the local construction of G^1 continuous Gregory patches from the n-sided polygons. During the real time haptic interaction, the contact point is continuously tracked on the locally constructed Gregory patches and thus generates smooth haptic force to be rendered. Our method has been compared with conventional force rendering techniques. Experimental results prove that the force rendered by our method is much smoother without adding a heavy cost on computing time and memory usage.

An interesting problem with the proposed

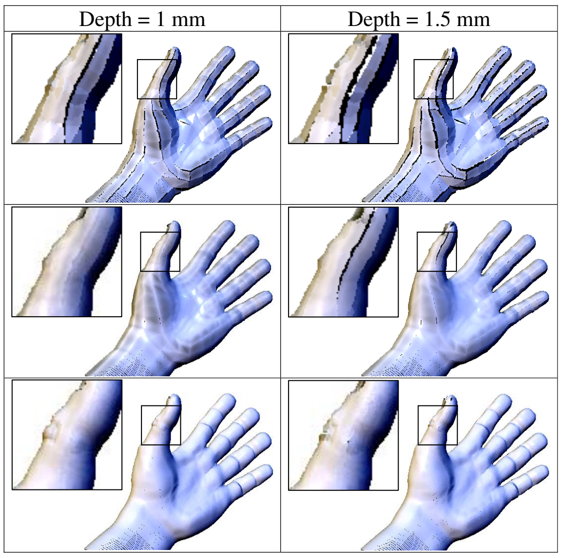


Figure 13: Study of force discontinuity on a hand model with 1272 quadrangles using different methods and with different penetration depths. (Top row) direct rendering [6], (middle row) force shading based rendering [1], and (bottom row) our Gregory patch based force rendering.

method is how to preserve the feels of sharp features. Method of feature preserving patch generation will be investigated in the near future. Then it will be integrated into a uniform force rendering approach based on patches.

Acknowledgements

This work is supported by the National Science Foundation of China under grant $No.60605027$ and $No.50575011$, the National Hi-tech Research and Development Program of China under grant $No.2007AA01Z310$, the Hong Kong RGC grant CUHK/417508, and the open project – *Research of GPU-Accelerated Force Rendering on Volumetric Model* – of State Key Lab of Virtual Reality Technology and Systems of China.

References

[1] H.B. Morgenbesser and M.A. Srinivasan. Force shading for haptic shape perception. In *Proceedings of the ASME Dynamics Systems and Control Division*, pages 407–412, 1996.

[2] C.-H. Ho, C. Basdogan, and M.A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence: Teleoperators and Virtual Environments*, 8(5):477–491, 1999.

[3] Y. Shon and S. McMains. Evaluation of drawing on 3d surfaces with haptics. *Computer Graphics and Applications, IEEE*, 24(6):40–50, Nov.-Dec. 2004.

[4] Y. Shon and S. McMains. Haptic force shading parameter effects on path tracing accuracy. In *14th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 517–523, March 2006.

[5] D. Wang, Y. Zhang, Y. Wang, Y.-S. Lee, P. Lu, and Y. Wang. Cutting on triangle mesh: Local model-based haptic display for dental preparation surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):671–683, 2005.

[6] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. In *IROS '95: Proceedings of the International Conference on Intelligent Robots and Systems*, volume 3, page 3146, 1995.

[7] M.A. Puso and T.A. Laursen. A 3d contact smoothing method using gregory patches. *International Journal for Numerical Methods in Engineering*, 54(8):1161–1194, 2002.

[8] C. Loop, S. Schaefer, T. Ni, and I. Castaño. Approximating subdivision surfaces with gregory patches for hardware tessellation. *ACM Transactions on Graphics (SIGGRAPH Asia 2009)*, 28(5):1–9, 2009.

[9] C.-H. Ho, C. Basdogan, and M.A. Srinivasan. Ray-Based Haptic Rendering: Force and Torque Interactions between a Line Probe and 3D Objects in Virtual Environments. *The International Journal of Robotics Research*, 19(7):668–683, 2000.

[10] A. Maciel and Suvranu De. A new line-based algorithm for real time haptic interactions with virtual environments. In *HAPTICS '08: Proceedings of the 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 217–223, 2008.

[11] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin, and D. Manocha. Six degree-of-freedom haptic display of polygonal models. In *VIS '00: Proceedings of the conference on Visualization*, pages 139–146, 2000.

[12] D.E. Johnson, P. Willemsen, and E. Cohen. Six degree-of-freedom haptic rendering using spatialized normal cone search. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):661–670, 2005.

[13] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 345–352, 1997.

[14] J. Barbic and D.L. James. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *Haptics, IEEE Transactions on*, 1(1):39–52, 2008.

[15] W. Li, Y. Shon, and S. McMains. Haptic rendering using c1 continuous reconstructed distance fields. In *IEEE International Conference on Shape Modeling and Applications*, pages 163–170, June 2009.

[16] J.-K. Lee and Young J. Kim. Haptic rendering of point set surfaces. In *WHC '07: Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 513–518, 2007.

[17] X. Guo, J. Hua, and H. Qin. Touch-based haptics for interactive editing on point set surfaces. *IEEE Computer Graphics and Applications*, 24(6):31–39, 2004.

[18] T.V. Thompson II, D.E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *SI3D '97: Proceedings of the symposium on Interactive 3D graphics*, pages 167–176, 1997.

[19] Lin M.C. Otaduy, M.A. A modular haptic rendering algorithm for stable and transparent 6-dof manipulation. *Robotics, IEEE Transactions on*, 22(4):751 –762, Aug. 2006.

[20] Y.L. Ma and W. T. Hewitt. Point inversion and projection for nurbs curve and surface: control polygon approach. *Computer Aided Geometric Design*, 20(2):79–99, 2003.

[21] E. Larsen, S. Gottschalk, M.C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. In *Proc. of Int. Conf. on Robotics and Automation*, pages 3719–3726, 2000.

[22] K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004.