Computer-Aided Design and Applications

http://www.cadanda.com

Haptic 3D Mesh Painting based on Dynamic Subdivision

Yongxiao Fu[1] and Yonghua Chen[2]

[1]The University of Hong Kong, kimi1984@hkusua.hku.hk
[2]The University of Hong Kong, yhchen@.hku.hk

ABSTRACT

In this paper, we present a haptic painting system which provides users an easy way to accurately add color on the mesh models. Unlike some of previously published methods which require texture mapping or point resampling, our system is totally mesh-based. Since a local dynamic subdivision algorithm has been developed in our method, the quality of painting can be less dependent on the original mesh density so that fine details can also be created in sparse models. Sample studies have shown that the visual painting results on mesh models are good enough for early industrial design of multi-materials (or multi-colors) parts where frequent and easy changes of painting are required. Our approach can also provide useful information in mesh editing like smoothing and cutting, which makes the system possible to be further developed into a 3D interactive design tool for parts made of multi-materials that are normally manufactured by multi-shot molding processes.

Keywords: haptics, painting system, dynamic subdivision.
DOI: 10.3722/cadaps.2008.131-141

1. INTRODUCTION

Virtual painting has been one of the major interests in computer graphics for many years. With the introduction of the haptic device, painting on 2D canvases has been extended to 3D object models. Haptics, or sense of touch is one of the most fundamental ways in which people perceive changes in the world around them. Compared with traditional painting methods which use mouse to control the brush, nowadays haptic devices can provide users with the sense of touch by sending reaction forces and also greater freedom in exploring the virtual space.

While other 3D painting systems were strived to simulate the real painting process, we see a larger potential of it in the product design field. Usually it is time consuming for the designer to complete a good painting model involving repeatedly modifications of the parameters in defining complex curves and surfaces using modeling software. However, by means of 3D haptic painting, any idea can be brought to the screen in a few minutes. Designer is free to try any combination of color and location on the model until the best one comes out. Therefore, our primary goal is to develop a haptic-based painting system which can be used as a preliminary design tool. In this paper, we also propose an efficient way to trace and smooth the boundary curves of the painted regions which can be exported for further operations like deformation or decomposition. Overall, the presented system has the following characteristics:

- Allow users to easily and accurately paint on the 3D surfaces in a natural style.
- Apply to CAD models as well as scanned objects with different mesh quality.
- Provide useful functions to extract and smooth the boundaries of painted area.

2. RELATED WORKS

2.1 Virtual Painting

Computer based virtual painting has been researched extensively in previous studies. Hanrahan and Haeberli [5] pioneered the work with a 3D painting system to achieve the WYSIWYG (What You See Is What You Get) fashion. The main drawback of it was the restriction of the brush controlled by 2D mouse and the user could only paint on the surfaces that are visible on the screen which may cause sudden discontinuity. In 1995, Agrawala et al [2] extended the above approach to a 3D input device, providing a natural force feedback to the user. This painting system could be applied to real objects through scanning but not to CAD files directly. A disadvantage of this system was that the physical object must be registered before painting. Registration error may occur and the registration process usually took several minutes. Since then, considerable efforts have been made to enhance the system and most of them chose the way to adopt texture mapping which can produce fine features but can cause distortion as well. The Chameleon painting system created by Igarashi and Cosgrove [6] used an adaptive unwrapping mechanism to automatically build a texture map and parameterization during interactive painting instead of traditional predefined UV-mapping. On the other side, in order to avoid the cumbersome mapping schemes, researches have also explored alternative approaches to increase the paint quality. Adams et al. [1] provide a solution by representing the object surface as collections of point samples. The well-known inTouch system developed by Gregory et al [4] used a subdivision framework to achieve multi-resolution mesh editing and solved the mapping problem by performing a standard scan-conversion in texture space. In 2001, Baxter et al [3] developed their painting system, DAB, aiming at creating painterly works. By using a physically-based brush, this system can generate different paint effects on the virtual canvas.

Commercial 3D painting toolkits were also available. The most famous one called FreeFrom was developed by Sensable Technologies [13] which was the developer of the haptic device PHANToM. Although FreeFrom integrated many useful functions and was easy to use, the volumetric based representation made it difficult to create fine details or features on the model. Furthermore, conversion from volume to surface may take time and cause problems. On the contrary, our painting system directly deals with meshes. Boundaries of the painted area and other useful surface parameters were available to provide functions in modeling and mesh editing after painting.

2.2 Haptic Rendering

Haptic rendering is the process of applying forces through the user controlled device to provide haptic interaction in a virtual environment. Generally, three steps are essential in the rendering: sensing the position, locating the contact point and generating feedback force. In 1994, Salisbury and his colleagues developed the PHANToM haptic interface [10] which was used in our system. Two popular ways existed in solving the dynamics of colliding bodies in haptic force modeling. Mark et al proposed the penalty-based method [8] which may suffer from a strong force discontinuity. Zilles and Salisbury [14] introduced a constraint-based "god-object" method in which the movement of the god-object was constrained to the object's surface. A solution to the force discontinuity problem was then proposed by introducing an algorithm based on implicit surface representation [7]. Recently large progress has also been made on the research of the 6 DoF haptic rendering to reduce the large computation cost [11] [12].

3. SYSTEM OVERVIEW

Our painting system is used to assign different colors at arbitrary locations of a 3D mesh-based model. By using a haptic device in our proposed system, painting directly on the object model can be achieved with force feedback to the user. Compared with mouse, input device like PHANToM can provide user a natural feeling of painting with real time collision detection and dynamic force rendering, allow the artists and designers to freely express their creativity.

In our system, we choose to paint color directly on the triangle mesh and try to explore a way to improve the paint quality as well as increase the system efficiency. Unlike point-based method which may require another conversion step from polygon to point cloud, our system can be applied to simple CAD models directly. Although the point resampling strategy can obtain high display quality, point-based representation makes it difficult to catch the geometric information compared with traditional polygon mesh. Besides, modeling operation is not yet well developed for the point-based representation. By using dynamic triangle subdivision, we can make use of the existing techniques in handling the mesh model, which greatly facilitates the development of many useful modeling functions later in our system.

Direct mesh painting greatly facilitates the boundary tracing and smoothing process by eliminating the complicated surface parameterization process. Furthermore, due to the reduced computation cost, our system can easily handle complex model without delay in both visual display and force feedback. We use a sphere to represent the brush

controlled by users and the radius of the sphere tool is very important in determining the painting resolution. The block diagram in Fig. 1 depicts our haptic painting system configuration.

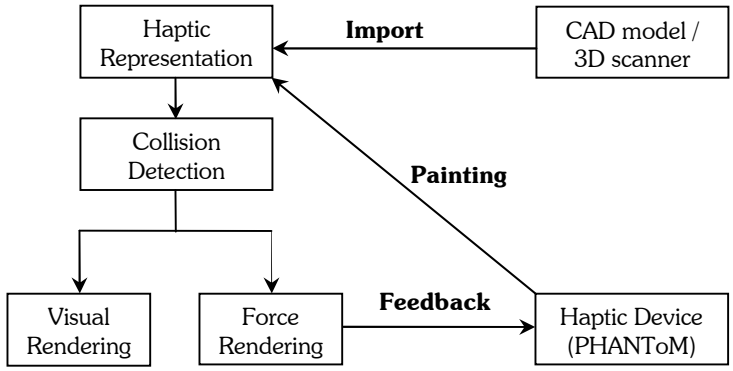
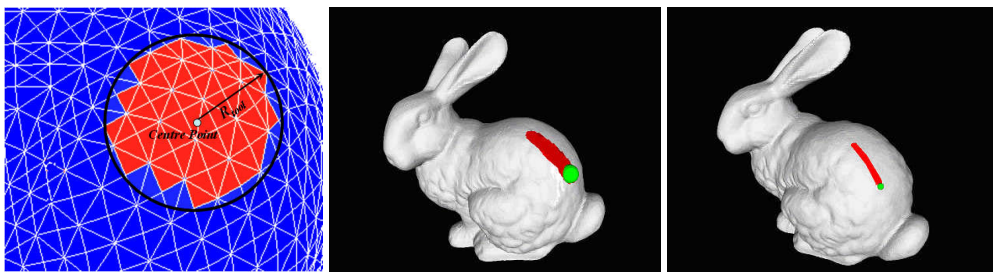


Fig. 1: Haptic painting configuration.

Models designed using commercial CAD software through triangulation as well as created by digital scanner can be imported to our painting system. The virtual brush used to paint is directly controlled using a three DoF input device PHANToM Desktop (Sensable Technologies) [13]. In order to display the color difference after painting, color information for every painted triangle is stored. To avoid any possible distortion while zooming and rotation, the relative orientations of the haptic device and the virtual brush are constrained to the same axis. This is accomplished by finding a transformation matrix between the two coordinate spaces.

In any 3D painting system, it is crucial that the user be able to place color on the surface mesh easily and accurately. We provide the sphere tool of which the radius is adjustable to the user. Usually large radius is chosen to obtain a rough and fast color paint in the beginning, followed by small radius tool to handle the boundary part and small areas if necessary. During the actual painting, the sphere is directly controlled by the haptic device and its centre point ($\boldsymbol{x_0}$, $\boldsymbol{y_0}$, $\boldsymbol{z_0}$) is tracked. Triangles with all three vertices ($\boldsymbol{x_p}$, $\boldsymbol{y_p}$, $\boldsymbol{z_p}$) inside the sphere will be painted immediately if a collision is detected between the centre and the mesh surface (Fig. 2(a)). The radius of the sphere $\boldsymbol{R_{tool}}$ is changeable so that various painting efficiency can be met. (Fig. 2(b), 2(c))

$$\sqrt{(x_p - x_0)^2 + (y_p - y_0)^2 + (z_p - z_0)^2} < R_{tool} \tag{1}$$



(a) (b) (c)

Fig. 2: (a) Illustration of brush sphere tool, (b)-(c) painting on the Stanford Bunny using tool with large and small radius.

Virtual wall model [10] is applied to all polygons in order to prevent user from penetrating through the surface when painting. A large coefficient $\boldsymbol{K}$ is used to simulate the force $\boldsymbol{F}$ from the virtual wall, where $\boldsymbol{X_{wall}}$ represents the point position on the polygon surface corresponding to the real avatar position $\boldsymbol{X_p}$. (Fig. 3)

$$F = \begin{cases} 0 & if \quad X_p > X_{wall} \\ K(X_{wall} - X_p) & if \quad X_p \le X_{wall} \end{cases} \quad (2)$$

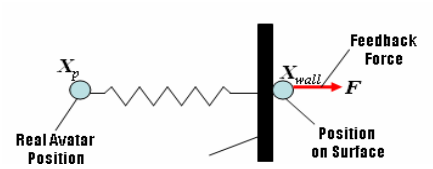


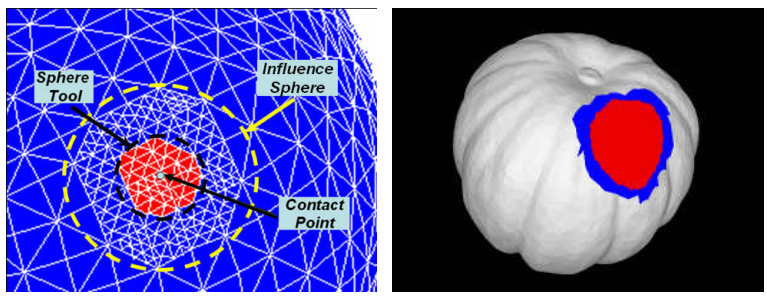
Fig. 3: Virtual Wall Model.

The force resulting from the dynamics is directly used for haptic feedback rendering. In addition, since each paint event only affects a local part of the surface, it is possible to maintain high frame rates by only locally updating the rendered image.

4. DYNAMIC MESH SUBDIVISION

Since the quality of the model greatly influences our painting results, a fine mesh model with more triangles will be better in producing more details. A prepossessing step to filter those long thin triangles is welcomed. Operations like edge-melting and inversion operations [9] are also helpful to avoid possible conflicts in the node connections. To make our system also works well when handling simple models, a local remesh process is integrated to provide dynamic subdivision functions. Users will be able to choose certain level of detail to paint by interactively selecting the tool size.

In our application we break the process into two stages. Firstly we define the region that needs to be subdivided and set a relation linking this active region with our paint tool. To make it simple to implement, an influence sphere with radius $\boldsymbol{R_{inf}}$ is introduced so that all the triangles within the influence sphere are subdivided before painting (Fig. 4(a)). It is easy to understand that the influence sphere should have the radius larger than our sphere tool to guarantee the paint quality. By default we set $\boldsymbol{R_{inf}} = \boldsymbol{3R_{tool}}$. But when a user wants to choose a small sphere tool, it is necessary to adjust the influence sphere to ensure that all the neighboring triangles is successfully remeshed. Fig. 4(b) shows an example where the influence region is in blue color and the actual painted area in red.



(a) (b)

Fig. 4: (a) Illustration of the dynamic subdivision process, (b) an example in our system.

When the influence region is determined, we begin to subdivide every triangle to match the resolution of the paint tool. We calculate lengths of the three edges and apply the following algorithms.

If all three edges exceed threshold ***d'***
  Construct 4 sub-triangles; (Figure 5(a))
  Subdivide each of the new sub-triangles;
Else If two edges exceed threshold ***d'***
  Construct 3 sub-triangles; (Figure 5(b))
  Subdivide each of the new sub-triangles;
Else If one edge exceeds threshold ***d'***
  Construct 2 sub-triangles; (Figure 5(c))
  Subdivide each of the new sub-triangles;
Else
  Render triangle;

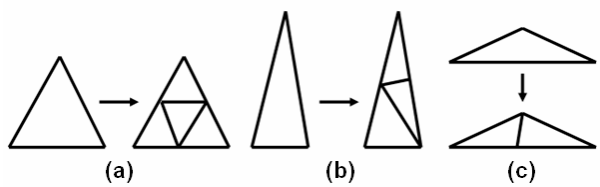


Fig. 5: Triangle subdivision algorithm.

All the new triangles are obtained by connecting the correspondent midpoints. This process will loop until the maximum edge length of every triangle within the highlighted area is smaller than a threshold $\boldsymbol{d'}$. We preset $\boldsymbol{d' = R_{tool}/10}$ and this parameter is also adjustable by users. To guarantee that the final geometric detail will match the paint tool, adding a sufficient number of vertices is required.

Some mesh format (e.g. STL) may use very long and thin triangles to show fine details as well as reduce the number of triangles. In the presence of these extreme triangles, the above algorithm seems too expensive when user wants to add very small features on the model because it will generate many sub-triangles most of which will not be painted. In these cases, the remesh area is very small even compared with one triangle. Our solution is to minimize the remesh area by finding the influence triangle during subdivision process. We define the influence triangle as the smallest triangle containing the contact point with every edge $\boldsymbol{d > 10R_{tool}}$. Fig. 6 shows a typical example when user wants to paint with small size tool inside ΔABC.

1. ΔABC is firstly decomposed into three sub-triangles ΔADE, ΔDCE, ΔDBC.
2. Since the contact point is inside ΔDBC, ΔDBC is divided into four sub-triangles ΔDGH, ΔGBF, ΔHFC and ΔGFH.
3. Further check shows the contact point is inside ΔGFH, we repeat the process to find the influence triangle (in the example ΔGFH is the influence triangle).
4. Midpoint recursive subdivision is applied to ΔGFH to meet the radius of the paint tool.
5. Dynamic collision detection is performed to determine the user painted area 'S' inside ΔGFH.

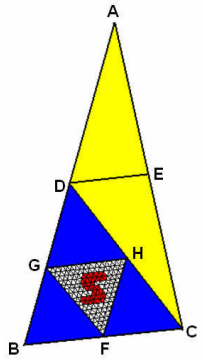
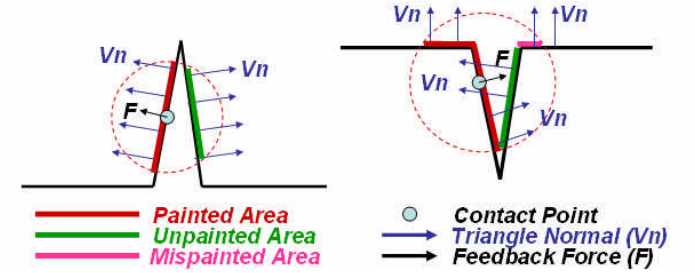


Fig. 6: Small paint in long triangle.

Using this approach, we can greatly reduce the number of newly created triangles to save memory space and increase the system efficiency. It is also proved to perform well in dealing with simple models having large triangles.

Finally we start to check all the triangles within the remesh area and assign colors to those located inside the volume of our paint tool. We dynamically delete information of new generated triangles if their parent is entirely painted to save memory as well as maintain a fast reaction speed. Basically, our painting system assumes that the target model mesh is relatively smooth. Some problems may happen when handling models having sharp corners.



(a) (b)

Fig. 7: Problems during painting. (a) green area in the sphere volume mispainted (b) pink area with $\boldsymbol{F \cdot V_n > 0}$ mispainted.

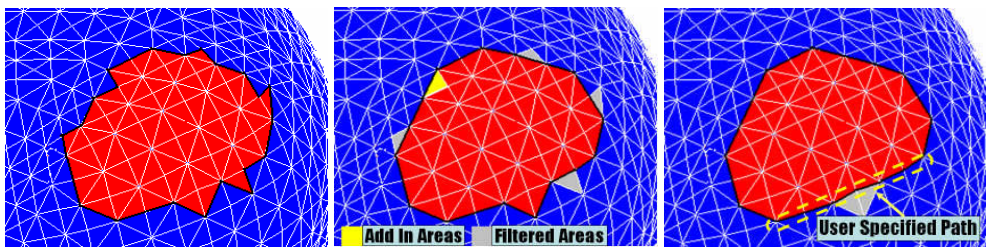
For example the green area in Fig. 7(a) will be automatically painted since the triangles there are all inside the sphere. In order to avoid the presence of these mispainted areas, we check the surface normal of every triangle $\boldsymbol{V_n}$ and the feedback force $\boldsymbol{F}$ and only label those triangles with $\boldsymbol{F \cdot V_n > 0}$ as painted, thus prevent adding color to the other side of the corner. However, we may see from Fig. 7(b) that the area in pink will also be colored because those triangles still pass our first check. One possible solution to overcome this would be adding a constraint that every triangle ready to be painted must be a neighbor of a painted triangle. As a result, our painting algorithm ensures that the painted region will be spread from the exact triangle containing the contact point to the peripheral area corresponding to the radius of the sphere tool. As a tradeoff, the processing time may be longer compared with the direct triangle labeling. Fig. 8 demonstrates the use of dynamic mesh subdivision, a letter 'S' is painted inside a single triangle of the original mesh.

Fig. 8: Paint "S" inside one triangle using dynamic mesh subdivision.

5. BOUNDARY SMOOTHING

The result of haptic painting not only provides designers a high quality visual evaluation, but also provides important geometric information for design purpose. In our system, we prepare the function to automatically extract and smooth the boundaries of the painted area.

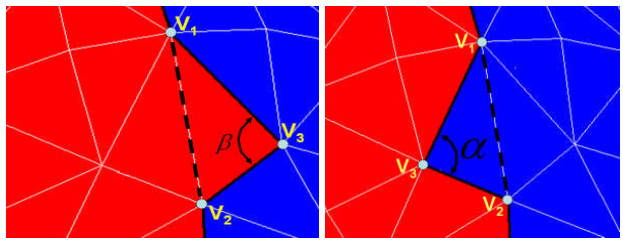
Because colors are painted directly on the triangle mesh, the boundary tracing process turns into a search and sort problem. By iteratively searching and checking every vertex inside a painted region, we can get all the boundary points and link them together according to the sequence to obtain a preliminary boundary polyline (Fig. 9(a)).



(a) (b) (c)

Fig. 9: (a) Boundary tracing of the original painted area, (b) automatic and (c) semi-automatic detection and remove of the unwanted painted region.

Note that due to the complexity and randomness of the mesh and painting, there may be some sharp edges on the boundary which may affect the smoothness of the overall boundary curve. To solve the problem, an optional check is enabled for the designer to automatically or semi-automatically remove these areas. By choosing the automatic approach, our tracing function will quickly filter those single sharp corners by adding or removing area boundary triangle (Fig. 9(b)).



(a) (b)

Fig. 10: Illustrations of automatic filtering single sharp corners: (a) removing the user painted area, (b) adding in the new paint area.

In automatic boundary smoothing, a global test is carried out to mark the situations shown in Fig. 10(a) where all the three vertices $\boldsymbol{V_1}$, $\boldsymbol{V_2}$, $\boldsymbol{V_3}$ form a painted triangle and lie on the traced boundary. To deal with convex corners, angle β is firstly calculated. If β $<\boldsymbol{135^0}$, the corner is treated as a sharp corner and should be removed. We now link $\boldsymbol{V_1}$, $\boldsymbol{V_2}$ together to construct the new boundary segment and $\boldsymbol{V3}$ is removed from the boundary. The color of triangle $\boldsymbol{V_1}$, $\boldsymbol{V_2}$, $\boldsymbol{V_3}$ is reverted to its previous color. If β $\geq\boldsymbol{135^0}$, the corner is treated as a smooth corner and kept unchanged. The

actual selection of a threshold value for angle β is up to the user. Consider the case in Fig. 10(b) where three consecutive boundary points $\boldsymbol{V_1}$, $\boldsymbol{V_3}$, $\boldsymbol{V_2}$ form an unpainted triangle. If α $\boldsymbol{<135^0}$, the triangle $\boldsymbol{V_1}$, $\boldsymbol{V_3}$, $\boldsymbol{V_2}$ is painted and point $\boldsymbol{V_3}$ is removed from the boundary point list. Otherwise, the corner is treated as a smooth corner.

It is noticed that the above boundary smooth operation is restricted to handle single triangle only. As a result, it may not work efficiently on larger corners especially when the model is in high resolution with thousands of small triangles. Thus, a semi-automatic filtering operation is also available. In this operation, the user can mark up a portion of the boundary that he wishes to smooth, boundary points outside the marked line are quickly removed as in Fig. 9(c).

Because of the good performance and high efficiency in producing smooth curves, Hermite curve is used to smoothly interpolate between key-points which are the boundary points in our applications. To calculate the Hermite curve (Fig. (11)), we need the following vectors:

$\boldsymbol{P_1}$: the start point of the curve
$\boldsymbol{T_1}$: the tangent value how the curve leaves the start point
$\boldsymbol{P_2}$: the end point of the curve
$\boldsymbol{T_2}$: the tangent value how the curve meets the end point

$$S = \begin{bmatrix} s^3 \\ s^2 \\ s \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} P_1 \\ P_2 \\ T_1 \\ T_2 \end{bmatrix} \quad M = \begin{bmatrix} 2 & -2 & 1 & -1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

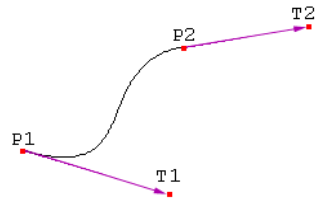


Fig. 11: A Hermite curve.

To obtain a point P on the curve we build the Vector $\boldsymbol{S}$, multiply it with the matrix $\boldsymbol{M}$ and Vector $\boldsymbol{C}$.

$$P = S \cdot M \cdot C \tag{4}$$

Since we do not have the information for the exact tangent values $\boldsymbol{T_1}$, $\boldsymbol{T_2}$ of the curve, the above algorithm may not be directly applied to our system. Instead we turn to the Catmull-Rom splines, where the tangent values can be calculated from the control points. Given $\boldsymbol{n+1}$ points $\boldsymbol{P_0}$, ..., $\boldsymbol{P_n}$, to be interpolated with $\boldsymbol{n}$ Catmull-Rom spline segments, the tangent value $\boldsymbol{T_i}$ can be obtained by

$$T_i = \frac{P_{i+1} - P_{i-1}}{2} \tag{5}$$

Although we may lose some of the flexibility of the interpolated curves, but as a tradeoff, the curves are much easier to calculate with satisfying results.

Notice that after applying the boundary smoothing operations, the resulting boundary curves stay in the 3D space rather than being constrained onto the original mesh surface (Fig. 12). To solve this problem an extra projection step is added. Given a target plane $\boldsymbol{ax + by + cz + d = 0}$ with the normal vector $\boldsymbol{V}$ = ($\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$), projecting a point ($\boldsymbol{x_0}$, $\boldsymbol{y_0}$, $\boldsymbol{z_0}$) onto the target plane gives the distance as

$$D = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}} \tag{6}$$

The corresponding point on the plane ($\boldsymbol{x_p}$, $\boldsymbol{y_p}$, $\boldsymbol{z_p}$) can be calculated by

$$\begin{cases} x_p = -ax_0 + D \\ y_p = -bx_0 + D \\ z_p = -cx_0 + D \end{cases} \tag{7}$$

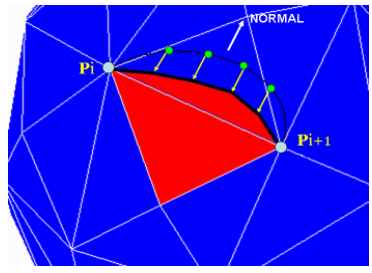
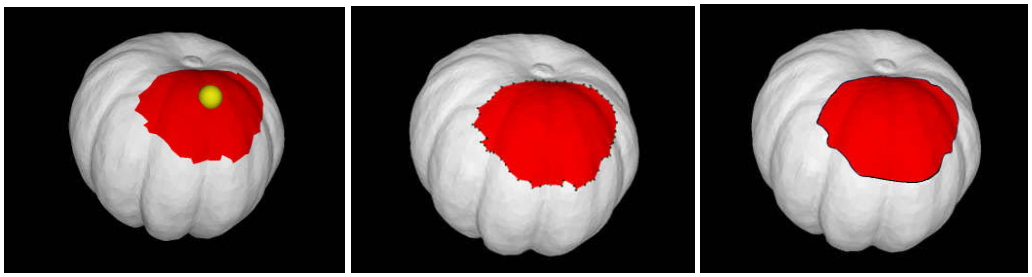


Fig. 12: Boundary curves after projection.

With this method, boundary curves can be exactly constrained onto the model surface. Fig. 13(a)-(c) shows the generation of 3D boundary curves from the painting on a pumpkin.



(a) painting (b) boundary tracing (c) boundary curve fitting

Fig. 13: Virtual painting (a) on the pumpkin with boundary tracing (b) and smoothing (c).

6. HAPTICS RENDERING

Our haptic drawing system can provide user virtual interaction with the 3D model. However, compared with mouse control which constrains user's movement on the 2D plane, painting in the 3D space may lead to low accuracy control due to the extra degree of freedom. As a result, user may feel it difficult to paint the color to the exact location without deflection. In this section we discuss some of the approaches to overcome the drawback.

Friction and Damping Force

Adding friction and damping force is a practical way to prevent the occurrence of oscillation and unwanted sliding that may affect the painting result. According to the classic model we break the friction force into static and kinetic stages. For easy implementation we preset a force $\boldsymbol{F_s}$ with small magnitude as a threshold to replace the maximum static friction force. If user applies a force larger than $\boldsymbol{F_s}$ the sphere tool will begin to move along the surface and the total force $\boldsymbol{F_t}$ in the opposite direction is calculated as

$$F_t = f_k + F_d = \mu_k \cdot F_N + c \cdot x_p' \tag{8}$$

Where $\boldsymbol{f_k}$ and $\boldsymbol{F_d}$ indicate the kinetic friction force and the damping force respectively, $\boldsymbol{F_N}$ is the normal force obtained the from the Virtual Wall model, $\boldsymbol{c}$ and $\boldsymbol{\mu_k}$ are the coefficients of viscous damping and kinetic friction, $\boldsymbol{x_p'}$ is the velocity of the sphere tool controlled by user.

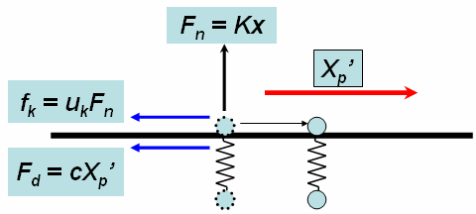


Fig. 14: The kinetic friction and damping force model.

Note that the feedback force is related to the penetration depth (kinetic friction force) and also the velocity of the user controlled sphere tool (damping force). In the application, the threshold force $\boldsymbol{F_s}$ is helpful to avoid unwanted small movements of the painting tool while kinetic friction force and damping effect are used to limit the painting speed so that we can have enough time for the procession of dynamic mesh subdivision and collision detection.

Normal Interpolation

In our original application, when a collision is detected our haptic loop will calculate a feedback force in the normal direction with magnitude proportional to the penetration distance from the virtual wall model. However in real painting we still found some problems near concave and convex regions where sudden change of the surface normal could possibly lead to discontinuity in the force feedback.

To solve the problem, we use the similar way as in the basic graphics shading to assign a unique normal vector to every vertex. The new normal is obtained as an average vector of the all the normals of the vertex's adjacent triangle. We find the exact triangle containing the contact point or proxy position. Barycentric coordinate $\boldsymbol{V_p(\lambda_1, \lambda_2, \lambda_3)}$ with respect to the three vertices $\boldsymbol{V_1}$, $\boldsymbol{V_2}$, $\boldsymbol{V_3}$ of the triangle (Fig. 15) is solved using Eqn. 9

$$\begin{cases} V_p = \lambda_1 V_1 + \lambda_2 V_2 + \lambda_3 V_3 \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{cases} \tag{9}$$

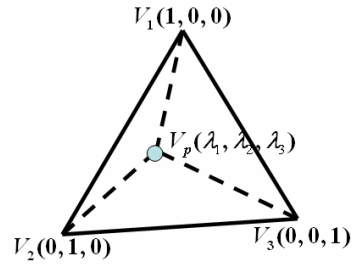


Fig. 15: Barycentric coordinate.

An interpolated normal of the contact point $\boldsymbol{N_p}$ can be calculated and used in our force rendering as a weighted average of the normals of the three vertices.

$$N_p = \lambda_1 N_1 + \lambda_2 N_2 + \lambda_3 N_3 \tag{10}$$

Basically user will choose a large sphere in the actual painting rather than a single point. Then the final force direction $\boldsymbol{N'}$ is obtained as an average of all the vertices' normals $\boldsymbol{N_i}$ inside the sphere tool.

$$N' = \left(\sum_{i=1}^{n} N_i\right) \Big/ \left\| \left(\sum_{i=1}^{n} N_i\right) \right\| \tag{11}$$

As shown in Fig.16, the above modification is effective in removing sudden changes in the direction of the feedback force and providing user a virtual smooth surface although the output graphics and painting result still go with the original model geometry.

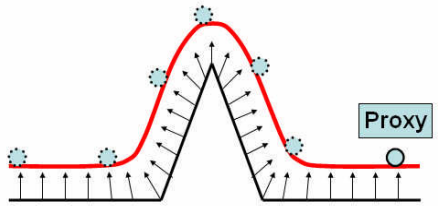


Fig. 16: Continuous force feedback after normal interpolation.

7. IMPLEMENTATION AND RESULTS

Our experimental system is implemented using Visual C++ together with a commercial PHANToM desktop haptic device. The painting program consists of two main loops. The haptic toolkit OpenHaptics is used in the haptic loop for force rendering while OpenGL is employed in the painting loop handling the visual display and user interface. Fig. 17(a)-(c) demonstrate some results using our application.

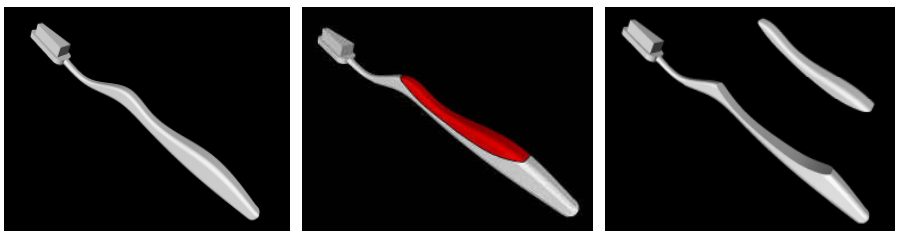
(a) (b) (c)

Fig. 17: Painting on (a) OpenGL teapot, (b) pumpkin model, (c) computer mouse.

The most important aspect in developing a 3D painting system is maintaining an intuitive, precise and responsive interface. To guarantee the required 1 kHz update frequency of the haptic device to provide stable real time interaction, we decouple the force computation from the rest of the applications. Only computations related to force

feedback like collision detection and force rendering are performed in the haptic loop. The other operations are done in the painting loop which runs at the 30 Hz frequency for visual display.

Fig. 18(a)–(c) show a simple application of our painting system in the design process. Fig. 18(a) is a toothbrush CAD model created by commercial software. A satisfactory color assignment is done by designer in Fig. 18(b) with the boundary of the colored region traced and smoothed. Based on the smooth boundary curve, a cutting surface is generated and cut the volume into two as shown in Fig. 18(c).



(a) (b) (c)

Fig. 18: Design of a multi-material toothbrush. (a) the original model, (b) boundary smoothing of the painted area, (c) volume decomposition using the cutting surface.

8. CONCLUSION AND FUTURE WORKS

We have presented a haptic-based 3D painting system which supported direct painting on the CAD model as well as scanned physical objects. The painting system provides virtual brushes based on a 3 DoF input haptic device. In order to simplify the boundary tracing algorithm and increase computation efficiency, we choose the direct mesh based method with dynamic subdivision to achieve multi-resolution paint. Our painting system is able to provide stable feedback in handling complex model without distortions. A simple and efficient algorithm for building smooth boundary curves on triangulated mesh has been introduced.

There are several limitations in our current application. In the painting simulation, we lack a good brush model which is capable of torque output together with force feedback. Integrating a 6-DOF haptic device would be helpful to provide various brush effects. In the modeling part, with the advent of the haptic device, it will be useful to the designer if local deformation is also enabled after painting in the conceptual design process. Also, we are still in the process of improving the system by modifications of in the boundary smoothing function to avoid possible failure in the presence of complex surface topology.

9. REFERENCES

[1] Adams, B.; Wicke, M.; Dutre, P.; Gross, M.; Pauly, M.; Teschner, M.: Interactive 3D Painting on Point-Sampled Objects, Eurographics Symposium on Point-Based Graphics, 2004, 59-66.

[2] Agrawala, M.; Beers, A. C.; Levoy, M.: 3D painting on scanned surfaces, Proceedings of ACM SIGGRAPH 1995, 145-150.

[3] Baxter, W.; Scheib, V.; Lin. M. C.; Manocha, D.: DAB: Interactive Haptic Painting with 3D Virtual Brushes, Proceedings of ACM SIGGRAPH 2001, 461-468.

[4] Gregory, A. D.; Ehmann, S. A.; Lin. M. C.: inTouch: Interactive multi-resolution modeling and 3d painting with a haptic interface, Proceedings of IEEE Conference on Virtual Reality, 2002, 45-52.

[5] Hanrahan, P.; Haeberli, P.: Direct WYSIWYG painting and texturing on 3D shapes, Proceedings of ACM SIGGRAPH 1990, 215-223.

[6] Igarashi, T.; Cosgrove, D.: Adaptive unwrapping for interactive texture painting, 2001 ACM Symposium on Interactive 3D Graphics, 2001, 209-216.

[7] Kim, L.; Kyrikou, A.; Sukhatme, G. S.; Desbrun, M.: An Implicit-based Haptic Rendering Technique, IEEE IROS 2002, Switzerland

[8] Mark, W.; Randolph, S.; Finch, M.; Verth, J. V.; Taylor II, R. M.: Adding Force Feedback to Graphics Systems: Issues and Solutions, Proceedings of ACM SIGGRAPH 1996, 447-452.

[9] Markosian, L.; Cohen, J. M.; Crulli, T.; Hughes, J.: Skin: a constructive approach to modeling freeform shapes, Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, New York, 1999, 393–400.

[10] Massie, T. H.; Salisbury, J. K.: The Phantom Haptic Interface: A Device for Probing Virtual Objects, ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems, 1994.
[11] McNeely, W. A.; Puterbaugh, K. D.; Troy, J. J.: 6 Degree of Freedom Haptic Rendering Using Voxel Sampling, Proceedings of SIGGRAPH 1999, 401-408.
[12] Otaduy, A.; Lin, M. C.: Stable and Responsive 6 Degree of Freedom Haptic Manipulation Using Implicit Integration, World Haptics Conference 2005.
[13] SensAble PHANToM™ available at http: //www.sensable.com
[14] Zilles, C.; Salisbury, J. K.: A Constraint-based God-object Method For Haptic Display, ASME Haptic Interfaces for Virtual Environment and Teleoperator System, 1994.