Versatile 3D Texture Painting using Imaging Geometry

Xinyu Zhang, Young J. Kim*
Dept. of Computer Science and Engineering
Ewha Womans University, Korea
{*zhangxy,kimy*}*@ewha.ac.kr*

Xiuzi Ye
Dept. of Computer Science
Zhejiang University, China
yxz@cs.zju.edu.cn

Abstract

Texture map-based traditional painting systems are often limited by the model's parameterization from 3D model space to 2D texture space, since finding such a parameterization can be very difficult in practice (sometimes nearly impossible) for models with complex topology or detailed structure distribution. We present a novel data representation, imaging geometry, for 3D appearance modelling and painting. By generating 3D colored points for each triangle contained in a polyhedral surface model, the imaging geometry can support a great variety of painting operations similar to that of the conventional 2D image editor. The key ingredient of our approach is a novel, adaptive data representation for geometry, topology and color of an arbitrary surface, which allows users to treat each triangle as an image. In our experiments, we demonstrate how easily the imaging geometry is applicable to 3D painting for any arbitrary surfaces that includes a model consisting of only a single triangle, multi-part geometry, a non-orientable model, a non-manifold model, an open or a closed model.

Key words: 3D Painting, Texturing, Point geometry

1. Introduction

Traditional texture mapping techniques map a 2D image onto a 3D surface to describe visual features of the surface without increasing the complexity of the underlying geometry. Not only for augmenting visual characteristics to the surface, but texturing also can be employed for augmenting other surface attributes such as surface color, surface normal, peculiarity, transparency, illumination and surface displacement. Most of 3D painting systems use texture mapping by parameterizing the geometry and topology of a model in 2D texture space. As a result, 3D painting is often limited by the underlying parameterization between the model's geometry and 2D texture space, since all parameterization techniques introduce discontinuities, stretching, and other artifacts. Moreover, finding parameterizations can be very difficult in practice (sometimes nearly impossible) for models with complex topology or detailed structure distribution.

In our 3D painting system based on our earlier research work [21], we apply the 2D rasterization technique to 3D mesh geometry by point-sampling the surface of a triangular mesh with colored 3D points until the surface is completely covered by a suitable number of these points. We generate these sampled points with suitable resolutions and point sizes using the hardware-supported occlusion query [14]. Moreover, we also present a parameterization-free texture mapping method to eliminate the artifacts introduced by point-sampling.

The major advantages of our painting system compared to the earlier approaches are as follows:

- For 3D painting, the proposed surface representation can handle any types of polygonal mesh models and can represent both geometry and surface properties such as color in the model.
- No texture parameterization is required.
- Traditional texturing techniques can be intermixed with our representation. The transition between texture mapping and imaging geometry can be achieved easily.

*Corresponding author

- The geometry and topology of a given model are preserved for further mesh editing.

The remainder of the paper is organized as follows. After discussing the related work in Sec. 2, we show the data representation of imaging geometry in Sec. 3. We discuss the process of 3D painting in our system in Sec. 4. Finally, we show the examples and experimental results of our system in Sec. 5 and conclude the paper in Sec. 6.

2. Related Work

With the availability of a 3D painting system, designers are now able to paint textures directly on the surface of a model. The seminal work proposed by Hanrahan and Haeberli [8] enables direct painting on a 3D surface and now becomes available in various commercial systems. These tools let designers paint textures directly on 3D models by projecting screen space paint strokes to the 3D surface and then into texture space, where they are blended with other strokes.

Many commercial 3D shape modeling programs provide painting features, where designers can freely paint a 2D texture image in a 3D view [2, 10, 13, 19]. In these systems, designers set up a UV-mapping manually or automatically, and the system subsequently re-projects the designers' strokes in 3D to a corresponding position in a 2D image based on the pre-calculated UV-mapping. Manual mapping methods require repeated projection operations and unwrapping of the model into a planar domain. This process is tedious and challenging even for those who are familiar with such mathematic concepts.

A research progress has been made on autonomous UV-mapping [4, 7, 9, 11, 12, 20, 18]. A powerful texture painting system that supports a direct 3D interface via force-feedback devices was proposed by Foskey et al. [6]; Here, even though their technique supports automatic parameterization, texture distortion or stretching can be still caused by the parameterization itself.

Unparameterized painting methods exploiting an Octree have been presented recently. Debry et al. [5] present a solution for texture mapping an unparameterized model using an Octree to store texture data. Benson and Davis [3] have also done a similar work using Octree textures. Both of the systems remove 2D parameterization and store texture data as a sparse, adaptive Octree. The research trends based on Octree textures are expected to continue.

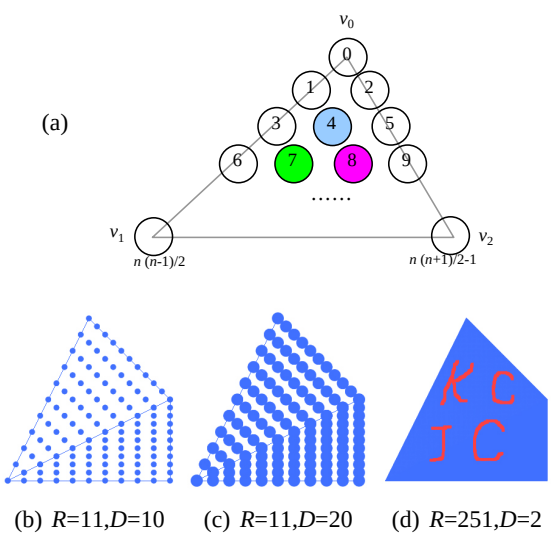
Another work similar to our approach is the research work by Zwiker et al. [22]. Their 3D painting system is also based on point geometry and uses irregular 3D points as 3D image primitives. However, they point-sample an input model irregularly and produce an irregularly point-sampled model as an output. As opposed to triangle primitives, point samples do not store connectivity information, so they must undergo an interactive parameterization step. Other painting systems using polynomials [2], triangles [1], implicit functions [16, 17] and images [15] have been presented for the past decade. Among them, Agrawala et al. [1] use a Cyberware scanner device to acquire the surface geometry of a given a physical object. Their system treats the sensor of the space tracker as a paintbrush and color the corresponding locations on the scanned mesh.

3. Data Representation

In this section we describe a flexible and adaptive data representation for 3D painting, imaging geometry. Our main goal is to get rid of parameterizations required for 3D painting. In order to accomplish this objective, we store the color information inside sampled points of the model, instead of in parameterized 2D textures.

3.1. Regular Sampling

Given an arbitrary triangle model M, we create a set of regular points for each triangle T in M by point-sampling T. Let T be a triangle in M with vertices v_0, v_1, v_2. Initially, we create n points for each edge in T and call n the resolution of the imaging triangle. Then, we connect these points and maintain virtual lines parallel to the edges. The intersections of these virtual lines indicate the locations of other sampled points to be created; See Fig. 1-(a). Fig. 1-(b), -(c) and -(d) demonstrate the examples of imaging triangles. In Fig. 1-(b), the resolution (R) is 11 and the rasterized diameter of points (D) is 10. We will explain what we mean by the rasterized diameter of points in Sec. 3.3. Likewise, in Fig.1-(c), R is 11 and D is 20 and in Fig. 1-(d), R is 251 and D is 2.



(b) R=11,D=10 (c) R=11,D=20 (d) R=251,D=2

Figure 1. Sampling a triangle. (a) Sampling pattern (b)-(d) Examples of imaging triangles.

This process will create $\frac{n(n+1)}{2}$ points for each triangle; n can be different for different triangles. If we use an index value representing the location of a point in the triangle, we need $\frac{n(n+1)}{2}$ index values in total. Moreover, we can treat this imaging triangle as a 1D array of indexed location values with color properties (r, g, b, a). These colored points are created for all triangles. We call these triangles as imaging geometries and we can get the new model M' from the original model M by imaging all the geometries.

In our system, we use 4 bytes to represent a RGBA color value. As a result, the total data size for each triangle is $2n(n+1)$ bytes. This size of data can overload the rendering time especially when the resolution is very high. In the following section, we explain techniques to speed up the performance.

3.2. Parameterization-Free Texture Map

In many commercial painting systems, parameterization-based texture mapping is normally used to perform painting operations. In these systems, a given mesh is parameterized in 2D image space. In this section, we present a parameterization-free texture mapping method to speed up the rendering performance.

As illustrated in Fig. 2, for each triangle in a given mesh, we create a texture and map the triangle with this texture. In Fig. 2, the right lower triangle-shaped half-image is used to texture the

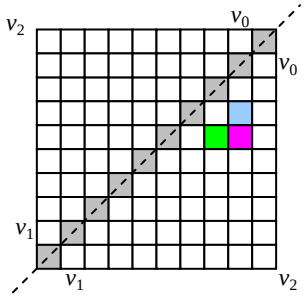


Figure 2. Texture mapping. The highlighted pixels are the corresponding points for 4, 7 and 8 in Fig 1-(a)

front of the triangle whereas the left upper is for the back of the triangle. Initially, the resolution of the prepared image is the same as a user-provided maximum resolution; however we assume that the resolutions of triangles can not be extremely high. From Fig.'s 1 and 2, we can see that there exists one-to-one mapping between the imaging geometry and a triangle-shaped half-image. Therefore, we can convert our imaging geometry to a regular texture without losing any surface details. Meanwhile, sampling and texturing both the front and back faces allows users to handle arbitrary surfaces, Möbius strips, for example.

To find the corresponding pixel for each colored point in imaging geometry, we use the following transformation between imaging geometry and 2D texture space: Given a point location index value id in a triangle, id specifies the location of a colored point. We need to compute the coefficients of a bi-linear interpolation. This is equivalent to acquiring a (u, v) coordinate value based on the index value id in a triangle. We can resolve this problem by solving the equation:

$$id = \frac{U(U+1)}{2} (id \geq 0).$$

We get

$$U = \frac{-1+\sqrt{4id+1}}{2}.$$

Then, we can get $u = \lceil U \rceil$ and $v = id - \frac{u(u-1)}{2}$.

For the front of a given triangle, the pixel position is $[R-(u-v)-1, R-u-1]$, and for the back of the given triangle, $[R-u-1, R-v-1]$

3.3. Multi-resolution

In order to reduce the amount of memory required in our system, we apply a multi-resolutional con-

cept to imaging geometry. There are two parameters to control the appearance of imaging geometry. The first one is the resolution of imaging geometry and the other one is the size of a rasterized point.

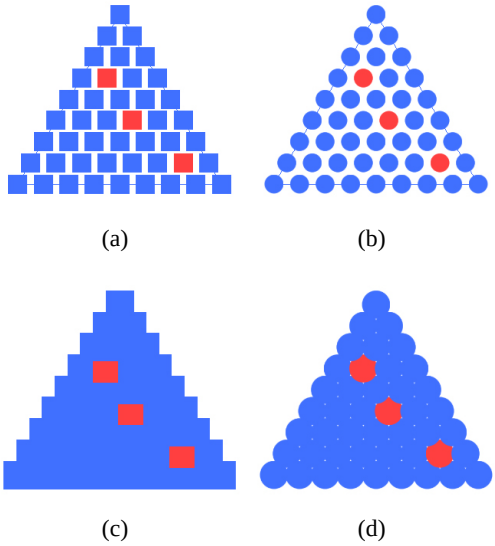


Figure 3. Multi-resolution and anti-aliasing. The rasterized diameter of a point is used to control the resolution of imaging geometry and the anti-aliasing should be followed to smooth out the rasterization.

Popular graphics library such as OpenGL can specify the rasterized diameter of a point. In our experiments, the maximum point size used is 63 and the minimum value is 1. If the rasterized size of a point is constant, we can change the resolution of a imaging triangle to obtain the appropriate appearance. If the resolution of imaging geometry is constant, we can change the rasterized size of a point. Moreover, when zooming in the model to see its details, we can also adjust the resolution and point size to get rid of apertures. We can approximate the resolution and the pixel size of a given triangles by solving the following equation:

$$N_{OQ} = \frac{(n)(n+1)}{2} \times \frac{\pi D^2}{4}$$

where N_{OQ} is the number of rasterized pixels of a triangle reported by the HW-supported occlusion query. We use the nVIDIA's occlusion query function to query the number of rasterized fragments [14].

In order to get rid of such artifacts as sharp edges and corners in imaging geometry, it is necessary that anti-aliasing for points should be performed (see Fig. 3). Another possible aliasing effect can be caused by the resolution difference between adjacent triangles. We always set the resolution difference as one or two. Providing a user interface to control the resolution of adjacent triangles is an alternative solution.

4. Painting Method

Our system can easily handle the models that are normally used in other systems. For surface painting, we move the mouse in screen space while drawing strokes directly on the model surface. One way to find a 3D model position from the 2D mouse cursor is to find where the line of sight through the 2D mouse position intersects with the 3D surface. If multiple intersections are found, we choose the closest one. In OpenGL, we use the selection and feedback features to implement such an interactive operation. The selection operation provided as a current feature in OpenGL allows us to take a 2D mouse click as an input and determine which part of the geometry in the model is beneath it. With the OpenGL's selection feature, we can specify a viewing volume and determine which objects fall inside that viewing volume. Based on the aforementioned features, we can get the vertices of a triangle, v_i, v_j, v_k, as well as the location information of other colored points. As we mentioned in Sec. 3, for each triangle, we use an index value to specify the location of points. Here, we save the texture properties (i.e., the color information) in an one dimensional array.

5. Results

We have tested our system on a number of models with a simple or complicated connectivity information. In Fig. 4-(a) and -(b), we show two imaging triangles and their painting results. Fig.'s 4-(c) and -(d) show a torus with many holes. We paint funny patterns on the torus without requiring any additional parameterization just like we paint a 2D image using well-known Painter or Photoshop program. To produce fancy textures in Fig 4-(e) and -(f), we utilize the texture features available in the PointShop 3D [22]. Users can create such painted models very easily in ten or twenty

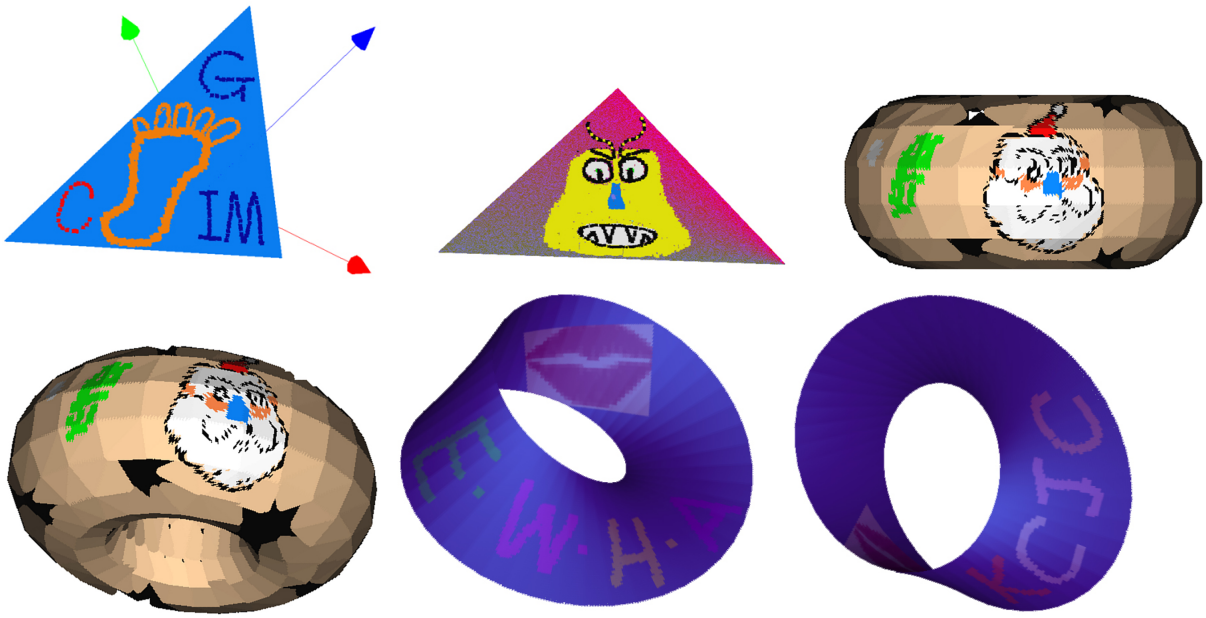


Figure 4. Painting examples

minutes. In Fig.'s 4-(e) and -(f), we show the results of painting on the Möbius strip.

6. Conclusions

We present a novel data representation to paint directly on any 3D triangle surface. Imaging geometry allows us to treat each triangle as a 2D image and paint points on a 3D triangle surface without requiring parameterization nor texture mapping process. The output of our paint system is still a triangular mesh model with colored points. This information can be further used in any rendering program by texturing the color pointed back to the surface geometry.

For future work, we would like to provide an intuitive man/machine interface to our painting system based on haptic interfaces such as PHANToM Premium 1.5.

Acknowledgement

This work is sponsored in part by the grant R08-2004-000-10406-0 of the KRF funded by the Korean government, the Ewha SMBA consortium, the ITRC program, China National Science Foundation (60273060; 60073026), China Ministry of Science and Technology software special project (2003AA4Z1020).

References

[1] Maneesh Agrawala, Andrew C. Beers, and Marc Levoy. 3D painting on scanned surfaces. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 145–150, 1995.

[2] Alias Wavefront. Maya. *http://www.aliaswavefront.com*, 2002.

[3] David Benson and Joel Davis. Octree textures. In *Proceedings of SIGGRAPH 2002*, pages 785–790, 2002.

[4] Nathan A. Carr and John C. Hart. Painting detail. In *Proceedings of SIGGRAPH 2004*, pages 845–852, 2004.

[5] David Debry, Jonathan Bibbs, Devorah D. Petty, and Nate Robins. Painting and rendering textures on unparameterized models. In *Proceedings of SIGGRAPH 2002*, pages 763–768, 2002.

[6] Mark Foskey, Miguel A. Otaduy, and Ming C. Lin. Artnova:touch-enable 3d model design. In *Proceedings of IEEE Virtual Reality Conference 2002*, pages 119–126, 2002.

[7] Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro,

and Michael Halle. Conformal surface parameterization for texture mapping. In *Transactions of Visualization and Computer Graphics*, volume 6, pages 181–189, 2000.

[8] Pat Hanrahan and Paul Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, volume 24, pages 215–223, 1990.

[9] Takeo Igarashi and Dennis Cosgrove. Adaptive unwrapping for interactive texture painting. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 209–216, 2001.

[10] Interactive Effects,Inc. AMAZON 3D Paint. *http://www.ifx.com*, 2002.

[11] Bruno Lévy. Constrained texture mapping for polygonal meshes. In *Proceedings of SIGGRAPH 2001*, pages 417–424, 2001.

[12] Jerome Maillot, Hussein Yahia, and Anne Verroust. Constrained texture mapping for polygonal meshes. In *Proceedings of SIGGRAPH 93*, pages 27–34, 1993.

[13] Metacreations,Inc. Paint 3D. *http://www.metacreations.com*, 2002.

[14] NVIDIA. http://oss.sgi.com/projects/ogl-sample/registry/NV/occlusion-query.txt.

[15] Byong Mok Oh, Max Chen, Julie Dorsey, and Frdo Durand. Image-based modeling and photo editing. In *Proceedings of SIGGRAPH 2001*, pages 433–442, 2001.

[16] Hans Kohling Pedersen. Decorating implicit surfaces. In *Proceedings of SIGGRAPH 95*, pages 291–300, 1995.

[17] Ronald N. Perry and Sarah F. Frisken. Kizamu: A system for sculpting digital characters. In *Proceedings of SIGGRAPH 2001*, pages 47–56, 2001.

[18] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of SIGGRAPH 2000*, pages 465–470, 2000.

[19] Right Hemisphere Ltd. Deep paint 3D (Deep UV, Deep Paint). *http://www.righthemisphere.com*, 2002.

[20] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001*, pages 409–416, 2001.

[21] Xinyu Zhang, Xiuzi Ye, and Sanyuan Zhang. 3D painting based on imaged geometry. *Journal of Software*, 15(3):461–467, 2004.

[22] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3D: An interactive system for point-based surface editing. In *Proceedings of SIGGRAPH 2002*, pages 322–329, 2002.