Drawing Features of 3D Meshes in Pencil-drawing Style

Kyungha Min
Dept. of Digital Media, Sangmyung Univ., Seoul, Korea
minkyungha@gmail.com

Abstract

We present a feature-guided salient rendering of a 3D triangular mesh. For this, we compute directions on the vertices of the mesh and distribute noise on the faces of the mesh. After projecting the direction and noise into image space, we execute a convolution algorithm to render the mesh. We use three directions in this paper: a principal direction, the tangent direction of an isocurve of view-dependent features, and the tangent direction of an isocurve of luminance. By controlling the noise, we produce several artistic effects including pencil drawing and hatching.

Keywords: NPR, feature, convolution, 3D mesh

1. Introduction

Convolution on a 3D mesh or 3D volume data is an interesting method to visualize the shape. Some researchers used convolution to visualize 3D medical volume data [3, 4] or some researchers used convolution to produce painting effects on a 3D mesh [14]. In the image space, the line integral convolution (LIC) is used to produce pencil drawing or line illustration effects on a photograph [9].

In this paper, we present a novel method that renders the salient features of a 3D mesh using LIC algorithm. The LIC is an image space algorithm that visualizes the flow embedded in an image by integrating noise superimposed on the pixels of the image along the direction of the flow [2]. The convolution is applied for producing pencil drawing effects from photographs [9], and for conveying the shape of a 3D volume data [3, 4, 14], but has not been applied for conveying the features of a 3D mesh yet.

The overview of our algorithm, which is illustrated in Figure 1, is as follows:

Step 1. We compute the directions at the vertex of the mesh from three directions on a mesh: (i) the principal directions, (ii) tangent directions of isocurves of view-dependent features, and (iii) tangent directions of isophote curves. We smooth them to build the smooth direction required for the convolution.

Step 2. We extract features from triangular meshes and measure the strength of feature at each vertex of the meshes. We generate noise at each face of the meshes from the strength of feature.

Step 3. Using the directions and noise, we apply a convolution algorithm to produce salient rendering.

This paper is organized as follows. In Section 2, we briefly review the related work on 3D convolution and mesh rendering techniques. In Section 3 and 4, we explain how we compute the directions and generate noise on 3D mesh, respectively. In Section 5, we describe the
convolution technique. We suggest our results in Section 6 and draw conclusion and future work in Section 6.

2. Related Work

2.1. LIC on 3D Models

Interrante and Grosch present an LIC method working on 3D volume data. By rendering the 3D flow embedded in the volume data in various colors [3], they successfully convey the inside of a 3D volume data. Later, Interrante extends this work to visualize two overlapping bones by rendering hatching patterns of a transparent bone along principal directions [4]. Ebert and Rheingans presents a non-photorealistic rendering scheme for a 3D volume data by enhancing the salient features in the data set [5]. Their volume visualization scheme, therefore, enhances the structural perception of
volume model. Lu et al., [8] applied stippling method for rendering volume data. By combining the artistic and scientific illustration, they present enhanced feature illustration.

2.2. Pencil Drawing on 3D Models

Lake et al., [6] produced pencil textures of different tones by overlapping five types of pencil stroke to achieve different densities. Lee et al., [15] created a tonal map from a range of pencil textures in principal directions. They accelerated this process using a GPU. Kim et al., [16] extended Lee et al.’s scheme to generate line art illustrations of dynamic 3D objects with highly reflective surfaces. They introduced a real-time imagespace algorithm for estimating principal directions from objects. This approach allows them to create textures that correspond to line effects including pencil drawing. Paiva et al., [18] presented a fluid-based hatching scheme to render triangular meshes of arbitrary topology and complicated geometry. They used the SPH approximation method to determine the density of the fluid on the surface of the mesh and placed hatching textures that follow the directions of the fluid. Praun et al., [7] developed a tonal art map for hatching a 3D mesh in real time. They overlapped a simple hatching stroke with various densities to create a series of textures and applied these to a mesh. Webb et al., [10] extended this technique to achieve hatching effects giving finer control of tone. Zander et al., [13] introduced another hatching scheme for 3D triangular meshes, in which smooth and evenly distributed streamlines are derived from the mesh in the principal directions. The results of the hatching process are stored as vectors and are therefore independent of resolution.

3. Computing Directions

3.1. Computing Principal Directions

One of our directions used by the convolution algorithm is constructed from the principal direction computed at each vertex of the meshes. Among the various schemes for principal directions and curvatures, we use the scheme presented by Rusinkiewicz [12], which is known as the most stable and efficient scheme.

3.2. Computing Tangents of Isocurves

Another direction we use is a tangent direction of an isocurve computed on a 3D mesh. The isocurve is a set of piecewise line segments whose points have an identical value. In our paper, we compute two isocurves. One isocurve is computed by $n \cdot v$, the inner product between the normal vector ($n$) and the view vector ($v$) and the other isocurve is computed by $n \cdot L$, the inner product between the normal vector and the light source vector ($L$). The isocurve of $n \cdot v$ is for the directions along the view-dependent features and the isocurve of $n \cdot L$ is for the directions that emphasize the lighting effects.

At a vertex $v$ whose value of the isocurve is $t$, we locate two points $p_1$ and $p_2$ that have $t$ in the surrounding edges of $v$. Therefore, tangent direction of the isocurve at $v$ is computed as $(p_2 - p_1) / \| p_2 - p_1 \|$. This process is illustrated in Figure 2, and three directions we compute are illustrated in Figure 3.
Figure 2. The Process of Computing Tangent Directions of an Isocurve on a 3D Mesh: (a) a Value is Estimated at each Vertex, (b) we Compute two Points $p_1$ and $p_2$ whose Value is Identical to the Target Vertex, (c) we Compute a Isocurve Connecting the Points, (d) the Tangent at the Vertex is Computed

Figure 3. Three Directions: (a) Input 3D Mesh, (b) the Tangent of the Isophote Curve, (c) the Tangent of the Isocurve of $n\cdot v$, (d) the Principal Direction

4. Generating Noise

In generating noise at each face of a 3D mesh, we determine the value of each noise and the number of noise generated on a face.
The value of noise is determined from the strength of feature estimated at the vertex of the mesh. We extract two different types of features: view-independent feature such as ridge or valley and view-dependent feature such as silhouette and contour. The strength of view-independent feature is estimated from the principal curvature and the strength of view-dependent one is from $n \cdot v$. The principal curvature is computed by the scheme used for the principal direction. For a stable computation, we normalize the principal curvature to $(0, 1)$.

The value of a noise, which is binary value, is determined from the strength as follows. We assign a characteristic value $\tau$ to each noise randomly from $(0, 1)$. The value of a noise is 1, if $\tau$ is smaller than the strength, and 0, otherwise.

The number of noise on each face depends on the projected area of the face. Since the projected area changes by the projection matrix that zooms in or zooms out the meshes, we generate noise on each face according to the maximum zoom-in of the meshes. For a face $f$ whose area is $a(f)$, we assume the maximum zoom-in as $k$ times and generate $k \cdot a(f)$ noise on $f$. The noise is generated using the dart throwing algorithm [1] which guarantees the Poisson distribution of any prefix of the generated noise. The size of the prefix is determined by the projection matrix.

![Figure 4. The Result of Noise Generation: (a) The Red Noise comes from the View-dependent Features and Blue Noise from the View-independent Features, (b) The Rendering Image from the Generated Noise](image)

5. Convolution

The directions and noise computed on a 3D mesh are projected into an image space where the convolution is executed. The projection is executed by applying the same projection matrix to the directions and noise. Note that the direction at each vertex is projected into a 2D vector whose origin is a pixel in an image space. The noise of a face is also projected into a pixel. When more than two noises are projected into a pixel, we discard the newly projected noise.

We use the LIC to integrate the noise embedded in the pixels of an image space. The LIC scheme is developed to visualize the smooth flow field in an image space [2]. By substituting the intensity of the pixels with noise of proper value, the LIC can produce hatching images and pencil drawings [9]. The LIC formula we use is as follows:
where $y = x + t \, d$. $(-1, 1)$ denotes the range of integration along direction $d$. $N$ is the number pixels in the integration direction and $I(y)$ is the value of noise at a pixel $y$.

6. Implementation and Results

We implemented our scheme on a PC with a Pentium i7 processor, 4 GByte main memory and nVidia GTX460 graphic processor. We have accelerated our scheme using CUDA, the GPU programming environment supported by nVidia. We tested our scheme on several 3D mesh models and produce results in Figure 5. In producing the results, we can control the style of the results. We blend the three directions and produce various hatching patterns on the mesh.

Most of the existing schemes that produce pencil drawing, line illustration or hatching patterns on a 3D mesh concentrate to mimic the tone of the mesh. They have limitations in presenting the salient features of the mesh. DeCarlo et al., have presented a scheme that conveys the salient features of a 3D mesh, but they represented the shape only lines and curves [11]. Our scheme can depict the salient of shape of a 3D mesh using pencil drawing and hatching patterns.

7. Conclusion and Future Work

We present a feature-guided convolution scheme on a 3D mesh to render the mesh in pencil drawing or hatching style. The directions that guide the convolution are the principal directions and tangent vectors of isocurves whose value is either $n \cdot v$ or $n \cdot L$. The noise and directions generated on a mesh are projected into an image space where convolution is executed to produce the result images. We successfully produce pencil drawing or hatching effects that convey the salient shape of a 3D mesh.

We will further extend this work to produce animation techniques. Another plan is to build a stereoscopic rendering that will be used in 3D theater.

Acknowledgement

This research was supported by a 2013 Research Grant from Sangmyung University.

Figure 5. Results
References


Author

Kyungha Min, got his Ph.D at POSTECH at 2000. His research interest is computer graphics and image processing. Currently he is at Dept. of Digital Media, Sangmyung Univ., Seoul, Korea.