基于图形硬件的光滑线条绘制

赵汉理[1)] 金小刚[1)] 沈建冰[2)] 卫飞飞[1)] 冯结青[1)]

[1)](浙江大学计算机辅助设计与图形学国家重点实验室 杭州 310058)

[2)](北京理工大学计算机科学与工程学院 北京 100081)

摘 要 线条能够以相当少的可视信息来有效地表示一个三维模型的形状.利用图形硬件的高度并行处理能力,文中提出了一种新的基于物体空间的线条绘制方法.对于一个光滑物体形状,新型的 Suggestive 轮廓线能够结合侧影轮廓线来生成非常好的视点相关的线条效果.并且,最新被引入到 Direct3D 10 流水线的几何着色器能处理三角形图元数据及输出零个或多个线条图元.文中提出了一种并行化的线条绘制方法,能够直接在几何着色器中抽取三维的特征线条.该方法能根据视点与模型的距离自动地选取合适比例的线条进行绘制,绘制的线条平滑,绘制速度快.在文章最后给出的许多实验示例证实了该方法的效果.

关键词 线条绘制;侧影轮廓线;Suggestive 轮廓线;径向曲率;图形硬件

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2009.01582

Smooth Line Drawing on Graphics Hardware

ZHAO Han-Li[1)] JIN Xiao-Gang[1)] SHEN Jian-Bing[2)] WEI Fei-Fei[1)] FENG Jie-Qing[1)]

[1)](*State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou* 310058)

[2)](*School of Computer Science and Technology, Beijing Institute of Technology, Beijing* 100081)

Abstract This paper presents a new object-space based algorithm for rendering a 3D model as a line drawing using graphics hardware, based on the insight that a line drawing can effectively convey shape using remarkably minimal visual content. Suggestive contours, a new type of line can be combined with silhouettes to produce impressive view-dependent line drawing of smooth shapes. In addition, geometry shader, which is newly introduced into the Direct3D 10 pipeline, can process per-triangle primitive data and output zero or more line primitives. This paper proposes a parallelized line drawing approach that directly extracts 3D sparse linear features in geometry shader. The method selects lines at an appropriate scale automatically, and exhibits good frame-to-frame coherence at interactive rates. The implementation of the algorithm is straightforward, and several experimental examples are given at the end of the paper to demonstrate the effectiveness of our approach.

Keywords line drawing; silhouette; suggestive contour; radial curvature; graphics hardware

1 Introduction

Computer graphics boasts an amazing success story with regard to realistic rendering, while to make pictures photoreal is quite expensive. Fortunately, there is an alternative to communicate

收稿日期:2008-08-14;最终修改稿收到日期:2008-12-10.本课题得到国家自然科学基金(60533080,60833007)、国家"九七三"重点基础研究发展规划项目基金(2009CB320801)、国家科技支撑计划课题(2007BAH11B03)资助.**赵汉理**,男,1982 年生,博士研究生,主要研究方向为非真实感绘制、GPU 通用计算.**金小刚**,男,1968 年生,博士,研究员,主要研究领域为隐式曲面计算、特效模拟、网格融合、纹理合成、群组动画、布料动画、非真实感绘制.E-mail: jin@cad.zju.edu.cn.**沈建冰**,男,1979 年生,博士,副教授,主要研究方向为纹理合成、图像弥补、高动态范围成像及处理.**卫飞飞**,男,1982 年生,博士研究生,主要研究方向为实时绘制、GPU 通用计算.**冯结青**,男,1970 年生,博士,研究员,主要研究领域为几何造型、绘制、计算机动画.

visual information that avoids having to worry about a myriad of perhaps unnecessary details. Depicting information about shape by line drawing is clearly effective and natural, having been used for tens of thousands of years. When artists design imagery to portray a scene, they do not just render visual cues veridically. Instead, they select which visual cues to portray and adapt the information each cue carries. Line drawing styles can be found in many contexts, such as cartoon, storytelling, technical illustration, architectural design and medical atlases.

DeCarlo et al.[1] augmented the suite of available line types by introducing suggestive contours—Feature lines drawn along zero crossing of radial curvature. Suggestive contours blend visually with true contours and help convey 3D shape (see Fig. 1). In addition, graphics hardware is becoming more programmable and is increasingly used as a co-processor. The shader model version 4.0 fully supports 32-bit floating-point data format, which meets adequate precision requirement for general-purpose GPU computing(GPGPU). The programmable geometry shader, which is newly introduced into the Direct3D 10 pipeline, can process per-triangle primitive data and output zero or more line primitives.

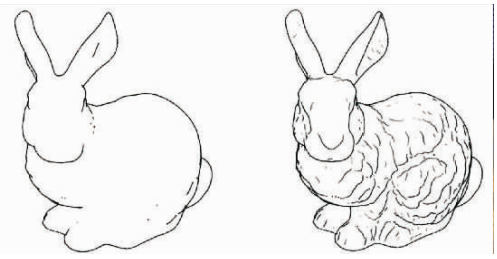

Fig. 1　An example showing the shape details depicted by suggestive contours. The left image is rendered using silhouette alone, whereas the right one is the result using both silhouettes and suggestive contours.

However, existing smooth line drawing algorithms using graphics hardware are image-space based and implemented in pixel shader, and thus cannot produce stylized effects for lines. In this paper, we propose an object-space based line drawing approach that directly extracts 3D sparse linear features in geometry shader. The method selects lines at an appropriate scale automatically, and exhibits good frame-to-frame coherence at interactive rates.

The rest of the paper is organized as follows. Section 2 give an overview of some of the previous work. Section 3 describe our new algorithm, while experimental results and related discussions are presented in section 4. Finally section 5 conclude the paper.

2　Related Work

A variety of surveys in computer-generated line drawings have been proposed for decades. In image-based approaches, the object is rendered to form an initial image, and then image processing methods yield an output drawing. While in object-based approaches, curves that have special properties in terms of the differential geometry of a surface are directly extracted. There are many types of feature lines: silhouettes (Hertzmann and Zorin[2]; Elber and Cohen[3]; Markosian et al.[4]; Cipolla and Giblin[5]; creases (Gooch et al.[6]; Markosian et al.[4]), ridges and valleys (Interrante et al.[7]; Ohtake et al.[8]), suggestive contours (Ni et al.[9]; DeCarlo et al.[1,10]; Burns et al.[11]), principal highlights and suggestive highlights (DeCarlo et al.[12]), apparent ridges (Judd et al.[13]). Other non-photorealistic techniques that generate lines from the rendered images are also widely developed (Iverson and Zucker[14]; Raskar and Cohen[15]; Lee et al.[16]; Zhao et al.[17]).

In Direct3D 10, a new GPU shading platform (Blythe[18]), the programmable geometry shader is firstly introduced into the GPU pipeline. It processes entire primitives (which is a single vertex for a point, two vertices for a line, or even three vertices for a triangle). Furthermore, the primitives generated by geometry shader can be streamed out to graphics memory, preparing for rendering next time. Now, more and more things can be transferred from CPUs into GPUs[19-20].

The paper only deals with silhouettes and suggestive contours, as suggestive contours blend visually with silhouettes and can produce a wide range of line drawings. However, our algorithm can also extract other types of lines, as long as they are defined on a triangle face and can be estimated and interpolated from the three vertices of triangle. Our method is different from the method proposed in [9], which draws lines via per-pixel operation on pixel shader and cannot produce stylized effects. Instead, we directly extract lines from 3D meshes in object-space, as the geometry shader has powerful ability to process triangle primitive and can emit line primitive. NVIDIA corporation① has presented an object-space based algorithm to extract silhou-

① http://developer.download.nvidia.com/SDK/10/direct3d/samples.html

ettes in geometry shader. However, they only take the edges of underlying triangles into account. Their lines are not smooth, while our approach can produce vivid smooth lines.

3 Smooth Line Drawing on Graphics Hardware

In this section, we begin with the definitions of silhouettes and suggestive contours. Next, the GPU implementation of feature line extraction is discussed in detail. Lastly, some rendering optimizations are introduced to improve the effect of the resulting image.

3.1 Silhouettes and Suggestive Contours

There are two elements, though not exhaustive, can produce a wide range of line drawings, and often contribute to more complex illustrations.

Consider a smooth, closed surface S that is viewed from a perspective camera centered at c. For any point p on the surface, the viewing direction is defined as $v(p)=c-p$. The silhouette is the boundary between the visible and hidden parts of the surface. It is generated by the set of points where the normal vector $n(p)$ is perpendicular to the viewing direction, that is:

$$v(p)\cdot n(p)=0 \tag{1}$$

As suggestive contours as concerned, of particular relevance is the notion of the curvature of a curve. The curvature $\kappa(p)$ at a point p on a curve is the reciprocal of the radius of the circle that best approximates the curve at p. Larger circles correspond to smaller curvatures, and any point on a straight line has curvature zero. Similarly, the normal curvature at a point p on a surface measures its curvature in a specific direction in the tangent plane. On a smooth surface, the normal curvature varies smoothly with the direction, and ranges between the principal curvatures $\kappa_1(p)$ and $\kappa_2(p)$ (the two values are calculated in their respective principal curvature directions, which are perpendicular to each other). If w is defined as the projection of the view vector v onto the tangent plane at p (see Fig. 2 (left)), the radial curvature $\kappa_r(p)$ is the normal curvature of the surface at p in the direction of w (Koenderink[21]). This defines the radial plane (see Fig. 2 (right)), which contains p, n, v, and w:

$$\kappa_r(p)=\kappa_1(p)\cos^2\phi+\kappa_2(p)\sin^2\phi,$$

where ϕ denotes the angle between $w(p)$ and the principal curvature direction corresponding to $\kappa_1(p)$. From its definition it is easy to see that κ_r is view-dependent, and that it is undefined wherever v and n are parallel.

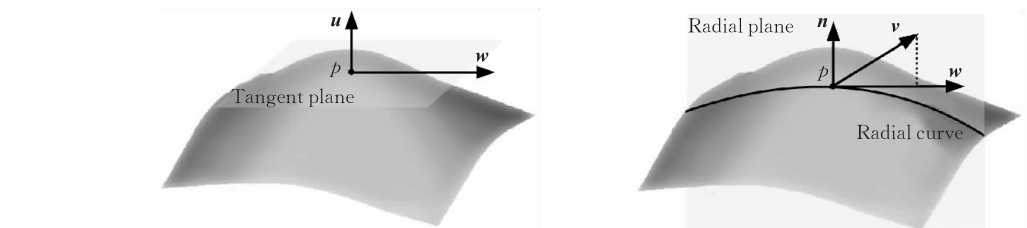

Fig. 2 (Left) w is obtained by projected the viewing vector $\boldsymbol{v}$ onto the tangent plane. (Right) The radial plane is formed by p, n, and w (also v) and slices the surface along the radial curve—the curvature of which is $\kappa_r p$.

Informally, suggestive contours, which are first introduced by DeCarlo et al., are contours in nearby viewpoints (see Fig. 3 (left)), or feature lines drawn along zero crossing of radial curvature. DeCarlo et al. offered three equivalent formal definitions for it. The first one is used in this paper, as follows:

$$\begin{cases}\kappa_r=0\\ D_w\kappa_r>0\end{cases} \tag{2}$$

Where the directional derivative $D_w\kappa_r$ stands for the differential of $\kappa_r(p)$ applied to w. In other words, Equ. (2) is equivalent to positive minima of

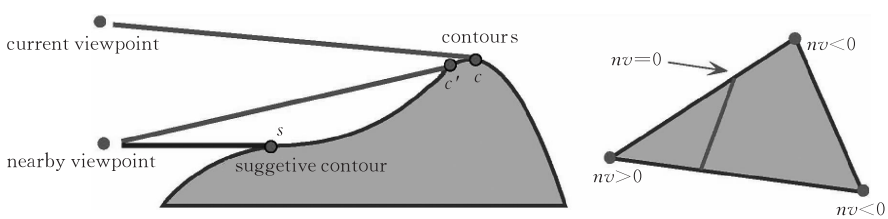

Fig. 3 (Left) c is contour as seen by the current viewpoint. As the viewpoint moves, the contour at c' slides along the surface from c, while a contour suddenly appears at s. Thus s is defined as suggestive contour. (Right) Zeros of "Phong"-interpolated $v\cdot n$: if signs at three corners not same, interpolate to find zero crossing within face.

$v \cdot n$ in the direction of w.

As θ, the angle between v and n, comes to zero, the suggestive contour generator is too unstable to give meaningful information about shape. Therefore, a threshold θ_c is enforced to place a tighter bound that the radial distance of the viewpoints considered is at most $\pi/2-\theta_c$ degrees:

$$\cos^{-1}\left(\frac{n(p)\cdot v(p)}{|v(p)|}\right)>\theta_c>0 \tag{3}$$

There are additional difficulties caused by errors in the curvature estimation. Again a small positive threshold t_d is applied to avoid those situations:

$$\frac{D_w\kappa_r}{|w|}>t_d>0 \tag{4}$$

3.2 Line Extraction in Geometry Shader

We have developed a new line extraction method from 3D triangular meshes entirely in geometry shader. The algorithm is highly parallelized and stable. Details follow in subsequent steps.

3.2.1 Preprocess Step

The normal, two principal curvatures, and the derivative-of-curvature tensor at each vertex on irregular triangle meshes need been estimated firstly. A finite-differences method for estimating curvatures (Rusinkiewicz[22]), which may be thought of as an extension of a common method for estimating per-vertex normals, is used here: first compute the per-face properties, and then estimate the value at each vertex as a weighted average over the immediately adjacent faces. These data accompanying with positions will be used as input for line extraction.

3.2.2 First Rendering Pass

The original mesh, as a canvas, is rendered with depth buffer writing enabled. As a result, the depth buffer can cull away the lines that lie on the back faces. Render target writing can be disabled to improve the efficiency.

3.2.3 Second Rendering Pass

The line extraction algorithm is performed in this pass. The silhouettes and suggestive contours are view-dependent, so we need render them every frame. The estimated normals and curvatures are bound as vertices buffer onto graphics hardware. Given a viewpoint, the values of $v \cdot n$, κ_r and D_w at each vertex are calculated using their definitions in vertex shader.

Next, we extract contours and suggestive contours in geometry shader. A naïve way to compute the contour is to check whether each edge has one adjacent front face and one adjacent back face. This can be done trivially easily in geometry shader, as each input triangle primitive can also include the vertex data for any edge-adjacent vertices. However, when viewed as a path along mesh edges, the contour may form loops in single triangle. Moreover, it has "pop-up" effect when viewpoint changes.

Instead, we first ask whether $v \cdot n$ has a different sign at some vertex for each face. If so, interpolate $v \cdot n$ along edges connecting positive-($v \cdot n$) and negative-($v \cdot n$) vertices to find zeros (as depicted in Fig. 3 (right)), then connect the two points with a segment. As for suggestive contour, the same idea is used to find zeros of κ_r, and check whether $D_w\kappa_r > 0$. Furthermore, we can solve the problems mentioned in naive method. If no contour or suggestive contour is on this triangle face, no line is emitted; otherwise, one or two lines are generated. At last, the color for line (usually black color) is directly output by pixel shader. The example lines are illustrated in Fig. 1 and Fig. 4.

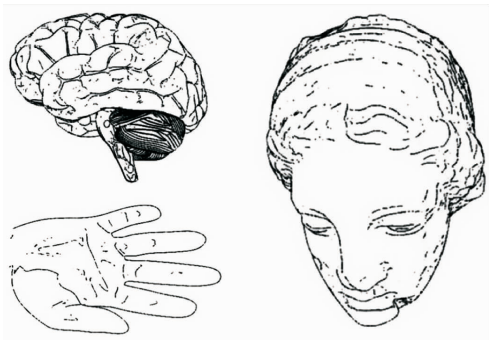

Fig. 4　Simple line rendering for (upper left) brain, (lower left) hand, and (right) Igea.

3.3 Rendering Optimizations

We have successfully extract contours and suggestive contours. When the viewpoint starts to approach a head-on view of a surface location (here, θ, the angle between v and n, comes to zero), the suggestive contour generator is too unstable to give meaningful information about shape. Moreover, there are additional difficulties caused by errors in the curvature estimation. DeCarlo et al.[10] applied a single positive threshold to avoid those situations:

$$\sin^2\theta\frac{D_w\kappa_r}{|w|}>t_d>0 \tag{5}$$

As we can see that suggestive contours will be trimmed continually as the value of t_d increases. The parameter can be used for Level-Of-Detail (LOD[23]) representation of these lines. When the viewpoint is near the model, decrease the threshold

value; otherwise, increase it. The method selects lines at an appropriate scale automatically according to the distance between viewpoint and mesh. The scalar fields $v \cdot n$ and κ_r are continuous on mesh surface in our method, the trimmed lines will occur at the "end" positions of the original lines. Thus, it guarantees good temporal coherence frame-to-frame. It does not need any additional pre-computation, while the LOD method in [9] needs complex progressive meshes[24] generation. As shown in Fig. 5, the underlying shape of dragon can be still recognized with reduced detail in distant view.

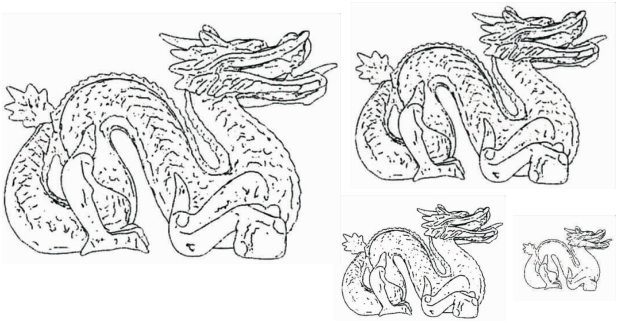

Fig. 5 A dragon model rendered from four viewpoints with automatic controlled detail. When viewed very far away, little detail is displayed, while the whole shape is still preserved.

Lastly, because the vertices of primitives generated in geometry shader can be streamed out into a vertices buffer, any non-photorealistic effect can be added as wish (toon shading, line width, etc., see Fig. 6).

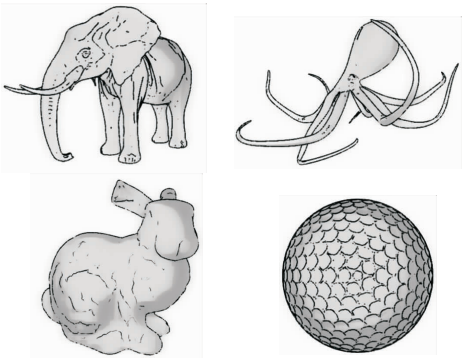

Fig. 6 Line rendering with toon shading, resulting vivid visual effects. Details conveyed by suggestive contours play an important role in these images.

4 Experimental Results and Discussions

We have incorporated our new method into a Direct3D 10 based scene graph library and tested it on some scenes in order to evaluate its efficiency for different scene types. All tests were conducted on a PC with an Intel Core 2 Due 6320 CPU, 2GB main memory, an NVIDIA GeForce 8800 GTS GPU, 320MB graphics memory, and Windows Vista Operating System.

The eight test scenes comprise of a dragon model (0.19MB polygons); a elephant model (0.64MB polygons); a octopus(49KB polygons); a bunny(69KB polygons); a golfball (0.98MB polygons); a brain (2.35MB polygons); a hand (43KB polygons); and an Igea model (0.26MB polygons); all are in resolution of 1440×900 pixels. Table 1 shows the model sizes and frame rates for our test models. As we can see, our method can achieve interactive frame rates even for models with millions of faces. Note that the method in this paper is object-space based, and the bottleneck is the feature extraction, not the rendering. As a result, the processing time of each frame is little relevant to the resolution of the rendering window.

Table 1 Performance statistics
(The rendering window is in 1440×900 all cases)

Model Name	Triangles	Vertices	Frames Per Second
Dragon	197664	98832	86.9
Elephant	640000	320118	23.5
Octopus	49920	24968	236.0
Bunny	69473	34835	113.0
Golfball	983040	491522	16.5
Brain	2353128	1179752	6.68

However, because the original mesh is used as a canvas for lines rendering, some lines will be clipped to the mesh's boundary in 2D. Moreover, as the faces number increases, the data cannot all fit in the memory of the GPU, resulting in extra overhead each frame.

5 Conclusions and Future Work

In this paper, graphics hardware is made useful to our new object-space based algorithm for smooth line drawing. We propose a parallelized line drawing approach that directly extracts 3D sparse linear features in geometry shader. The method selects lines at an appropriate scale automatically, and exhibits good temporal coherence at interactive rates. In addition, the experimental results demonstrate both the feasibility and efficiency of our proposed algorithm.

This paper gets good performance for generating contours and suggestive contours. However, silhouettes and suggestive contours cannot convey all the shape information about a 3D model, more types of feature lines need be included in future. Moreover, stylized strokes and other non-photore-

alistic rendering techniques will be introduced to improve the resulting image.

Acknowledgments We would like to thank the anonymous reviewers for their substantial feedback to improve the paper. Many thanks also to Luo Xiaoyan, Zhou Chuan, and Yang Wenwu for their dedicated help in completing the paper!

References

[1] DeCarlo D, Finkelstein A, Rusinkiewicz S, Santella A. Suggestive contours for conveying shape. ACM Transactions on Graphics, 2003, 22(3): 848-855

[2] Hertzmann A, Zorin D. Illustrating smooth surfaces//Proceedings of the ACM SIGGRAPH. New York, 2000. New York: ACM, 2003: 517-526

[3] Elber G, Cohen E. Hidden curve removal for free form surfaces//Proceedings of the ACM SIGGRAPH. Dallas, USA, 1990: 95-104

[4] Markosian L, Kowalski M A, Trychin S J, Bourdev L D, Goldstein D, Hughes J F. Real-time nonphotorealistic rendering//Proceedings of the ACM SIGGRAPH. New York, 1997. New York: ACM, 2003: 415-420

[5] Cipolla R, Giblin P. Visual Motion of Curves and Surfaces. New York: Cambridge University Press, 2000

[6] Gooch B, Sloan P-P J, Gooch A, Shirley P, Riesenfeld R. Interactive technical illustration//Proceedings of the ACM Symposium on Interactive 3D Graphics. Atlanta, USA, 1999: 31-38

[7] Interrante V, Fuchs H, Pizer S. Enhancing transparent skin surfaces with ridges and valley lines//Proceedings of the 6th Conference on Visualization. Washington, 1995: 52-59

[8] Ohtake Y, Belyaev A, Seidel H P. Ridge-valley lines on meshes via implicit surface fitting//Proceedings of the ACM SIGGRAPH. Los Angeles, 2004: 839-846

[9] Ni A, Jeong K, Lee S, Markosian L. Multi-scale line drawing from 3D meshes//Proceedings of the ACM Symposium on Interactive 3D Graphics and Games. Redwood City, 2006: 133-137

[10] DeCarlo D, Finkelstein A, Rusinkiewicz S. Interactive rendering of suggestive contours with temporal coherence//Proceedings of the ACM Symposium on Non-Photorealistic Animation and Rendering. Annecy, France, 2004: 15-24

[11] Burns M, Klawe J, Rusinkiewicz S, Finkelstein A, DeCarlo D. Line drawings from volume data//Proceedings of the ACM SIGGRAPH. Los Angeles, 2005: 512-518

[12] DeCarlo D, Rusinkiewicz S. Highlight lines for conveying shape//Proceedings of the ACM Symposium on Non-Photorealistic Animation and Rendering. San Diego, 2007: 63-70

[13] Judd T, Durand F, Adelson E. Apparent ridges for line drawing//Proceedings of the ACM SIGGRAPH. San Diego, 2007: 19

[14] Iverson L A, Zucker S W. Logical/linear operators for image curves. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995, 17(10): 982-996

[15] Raskar R, Cohen M. Image precision silhouette edges//Proceedings of the ACM Symposium on Interactive 3D Graphics. Atlanta, USA, 1999: 135-140

[16] Lee Y, Markosian L, Lee S, Hughes J F. Line drawings via abstracted shading//Proceedings of the ACM SIGGRAPH. San Diego, 2007: 18

[17] Zhao H, Jin X, Shen J, Mao X, Feng J. Real-time feature-aware video abstraction. The Visual Computer, 2008, 24(7-9): 727-734

[18] Blythe D. The Direct3D 10 system//Proceedings of the ACM SIGGRAPH. Boston, 2006: 724-734

[19] Bærentzen A, Nielsen S L, Gjøl M, Larsen B D, Christensen N J. Single-pass wireframe rendering//Proceedings of the Sketches of ACM SIGGRAPH. Boston, 2006: 149

[20] Microsoft Corp. Direct3D 10 samples. DirectX Software Development Kit, New York: Microsoft Corp., 2007

[21] Koenderink J. J. What does the occluding contour tell us about solid shape? Perception, 1984, 13(3): 321-330

[22] Rusinkiewicz S. Estimating curvatures and their derivatives on triangle meshes//Proceedings of the Symposium on 3D Data Processing, Visualization, and Transmission. Washington, 2004: 486-493

[23] Astheimer P, Pöche M-L. Level-of-detail generation and its applications in virtual reality//Proceedings of the Conference on Virtual Reality Software and Technology. River Edge, NJ, USA, 1994: 299-309

[24] Hoppe H. Progressive meshes//Proceedings of the ACM SIGGRAPH. New Orleans, 1996: 99-108

ZHAO Han-Li, born in 1982, Ph. D. candidate. His research interests include non-photorealistic rendering, texture synthesis, and general-purpose GPU computing.

JIN Xiao-Gang, born in 1968, Ph. D., professor. His research interests include implicit surface computing, special effect simulation, mesh fusion, texture synthesis, crowd animation, cloth animation, and non-photorealistic render.

SHEN Jian-Bing, born in 1979, Ph. D., associate professor. His research interests include texture synthesis, image completion, high-dynamic-range imaging and process.

WEI Fei-Fei, born in 1982, Ph. D. candidate. His research interests include real-time rendering and general-purpose GPU computing.

FENG Jie-Qing, born in 1970, Ph. D., professor. His research interests include geometric modeling, rendering, and computer animation.

Background

Computer graphics boasts an amazing success story with regard to realistic rendering, while to make pictures photoreal is quite expensive. Fortunately, there is an alternative to communicate visual information that avoids having to worry about a myriad of perhaps unnecessary details. Depicting information about shape by line drawing is clearly effective and natural, having been used for tens of thousands of years. When artists design imagery to portray a scene, they do not just render visual cues veridically. Instead, they select which visual cues to portray and adapt the information each cue carries. Line drawing styles can be found in many contexts, such as cartoon, storytelling, technical illustration, architectural design and medical atlases.

DeCarlo et al. recently augmented the suite of available line types by introducing suggestive contours — Feature lines drawn along zero crossing of radial curvature. Suggestive contours blend visually with true contours and help convey 3D shape. In addition, graphics hardware is becoming more programmable and is increasingly used as a co-processor. The shader model version 4.0 fully supports 32-bit floating-point data format, which meets adequate precision requirement for general-purpose GPU computing (GPGPU). The programmable geometry shader, which is newly introduced into the Direct3D 10 pipeline, can process per-triangle primitive data and output zero or more line primitives.

However, existing smooth line drawing algorithms using graphics hardware are image-space based and implemented in pixel shader, and thus cannot produce stylized effects for lines. This paper proposes an object-space based line drawing approach that directly extracts 3D sparse linear features in geometry shader. The method selects lines at an appropriate scale automatically, and exhibits good frame-to-frame coherence from at interactive rates.

The paper only deals with silhouettes and suggestive contours, as suggestive contours blend visually with silhouettes and can produce a wide range of line drawings. However, our algorithm can also extract other types of lines, as long as they are defined on a triangle face and can be estimated and interpolated from the three vertices of triangle. Our method is different from the method proposed by Ni et al., which draws lines via per-pixel operation in pixel shader and cannot produce stylized effects. Instead, we directly extract lines from 3D meshes in object-space, as the geometry shader has powerful ability to process triangle primitive and can emit line primitive. Nvidia corporation[①] has presented an object-space based algorithm to extract silhouettes in geometry shader. However, they only take the edges of underlying triangles into account. Their lines are not smooth, while our approach can produce vivid smooth lines.