Real-Time Shape Illustration Using Laplacian Lines

Long Zhang, Ying He, *Member*, *IEEE*, Jiazhi Xia, Xuexiang Xie, and Wei Chen

Abstract—This paper presents a novel object-space line drawing algorithm that can depict shapes with view-dependent feature lines in real time. Strongly inspired by the Laplacian-of-Gaussian (LoG) edge detector in image processing, we define Laplacian lines as the zero-crossing points of the Laplacian of the surface illumination. Compared to other view-dependent feature lines, Laplacian lines are computationally efficient because most expensive computations can be preprocessed. We further extend Laplacian lines to volumetric data and develop the algorithm to compute volumetric Laplacian lines without isosurface extraction. We apply the proposed Laplacian lines to a wide range of real-world models and demonstrate that Laplacian lines are more efficient than the existing computer generated feature lines, and can be used in interactive graphics applications.

Index Terms—Nonphotorealistic rendering, real-time line drawing, Laplacian lines, view-dependent feature line, object-space line extraction, volume illustration.

1 Introduction

Line drawings are an effective way to convey shapes in a relatively succinct manner by ignoring the less important or distracting details [2]. For complicated models or large-scale scenes, line drawing can be used to eliminate unnecessary visual clutter and depict essential information. A recent study shows that people interpret certain shapes almost as well from a line drawing as from a shaded image [3].

In the past decade, there has been a large amount of work on computer-generated line drawings, including suggestive contours [4], ridge-valley lines [5], apparent ridges [6], principal and suggestive highlights [7], photic extremum lines (PELs) [8] and demarcating curves [9], etc. These methods generate feature lines on 3D surfaces by computing either the second order (such as suggestive contours and highlights) or third order (such as ridge-valley lines, apparent ridges, PELs and demarcating curves) derivatives of certain surface-related properties. Typically, the derivatives have to be calculated on-the-fly by discrete differential geometry [10], [11]. The computations usually are expensive when the high order derivatives are involved. Thus, the existing third order feature lines can hardly be used for interactive graphics applications.

- *L. Zhang is with the Institute of Graphics and Image, Hangzhou Dianzi University, Hangzhou 310018, China. E-mail: lzhang@cad.zju.edu.cn.*
- *Y. He and J. Xia are with the School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, BLK N4, Singapore 639807. E-mail: {yhe, xiaj0002}@ntu.edu.sg.*
- *X. Xie was with the School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, BLK N4, Singapore 639807.*
- *W. Chen is with the State Key Laboratory of Computer Aided Design and Computer Graphics (CAD&CG), Zhejiang University, Hangzhou, China 310058. E-mail: chenwei@cad.zju.edu.cn.*

Manuscript received 7 Oct. 2009; revised 2 Apr. 2010; accepted 27 Apr. 2010; published online 8 Sept. 2010.
Recommended for acceptance by L. Kobbelt.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2009-10-0237.
Digital Object Identifier no. 10.1109/TVCG.2010.118.

Besides the object-space feature lines, image-space algorithms also draw much attention. For instance, Salisbury et al. [12] represented the pen-and-ink drawings with a gray-scale image, augmented by a set of discontinuity segments, along with a stroke texture to produce consistent drawings at any scale and resolution. One distinctive advantage of image-space algorithms is that the algorithm complexity is image-resolution dependent, making them amenable for hardware acceleration [13]. It also provides an efficient way to map screen-space patterns onto 3D models [14], and to convey both the geometry and material [15]. Furthermore, image-space line drawings usually are computationally efficient by avoiding the expensive computations of surface derivatives, as most object-space algorithms do. Thus, image-space algorithms can easily be used to render animated models in real time. However, image-space algorithms often suffer from pixel-level artifacts and are difficult for shape stylization.

This paper presents an efficient object space algorithm to generate view-dependent line drawings for interactive graphics applications. Our method is inspired by an analogy with image edge detection. Loosely speaking, we aim to simulate the Laplacian-of-Gaussian (LoG) edge detector in surface to extract view-dependent feature lines in object-space. The LoG operator can highlight regions of rapid intensity change while reducing the sensitivity to noise, and has proven to be an effective solution for image enhancement [16]. We define Laplacian lines as a set of points, where the Laplacian of the diffuse illumination vanishes, and the gradient magnitude is greater than certain user-specified threshold. Laplacian lines inherit the advantages of the LoG operator by employing a smoothing preprocessing to reduce the high frequency noise prior to the differentiation step. They provide considerably abstractive visual information and suppress distracting details.

Similar to apparent ridges, PELs and demarcating curves, Laplacian lines are third-order features that require third-order derivatives of the underlying surfaces. However, in sharp contrast to the other third-order features that

need to compute the derivatives on-the-fly, all the third-order derivatives of Laplacian lines can be completely precomputed. In particular, we show that the Laplacian of diffuse illumination is equivalent to the dot product of Laplacian of normals and the lighting vector. Note that Laplacian of surface normals is view-independent, and thus can be precomputed. As a result, the runtime Laplacian line extraction algorithm is just as simple as the conventional silhouette extraction algorithm, i.e., simply replacing the surface normal with Laplacian of normal.

The proposed Laplacian line is a general framework that can be applied to both surface and volumetric models. The key step in Laplacian line extraction algorithm is to compute the Laplacian of normals. For the surface case, we take advantage of the recently developed mesh Laplace operator [17], and show that it is much more robust than the conventional cotangent Laplacian operator. For the volume case, the underlying data are represented by the implicit function. Computing the Laplace-Beltrami operator is not as easy as the explicit representation. Using Weatherbrun's result [18], we derive a closed form formula to compute the Laplacian of normals in the volumetric data.

We conduct a wide range of experiments on real-world and synthetic data sets and demonstrate the capability of Laplacian lines to illustrate large-scale surface and volume models. Fig. 1 shows Laplacian lines and several other popular feature lines. Our approach not only generates visually pleasing results, but also is more efficient than the existing approaches.

The major contributions of this paper include:

- We generalize LoG edge detector to 3D surfaces, and define Laplacian lines as the zero-crossing points of Laplacian of illumination.
- We show that Laplacian of illumination is equal to the dot product of Laplacian of surface normals and the viewing vector, yielding an efficient Laplacian line extraction algorithm, since Laplacian of surface normals can be precomputed.
- By employing the recently developed mesh Laplace operator [17], our Laplacian line extraction algorithm is robust and insensitive to irregular tessellation. Furthermore, the users can globally control the number of extracted lines by specifying the Gaussian kernel size.
- We also define Laplacian lines for volumetric data sets and derive a closed form formula to compute the Laplacian of normals in implicit surface. Based on our theoretical results, we develop an algorithm to extract volumetric Laplacian lines without isosurface extraction. Following Burns et al.'s line tracing algorithm [19], our volumetric Laplacian lines can be drawn interactively with the user-specified isovalues.

The rest of this paper is organized as follows: We briefly review the related work in Section 2 and define the Laplacian lines in Section 3. We present the algorithms to extract Laplacian lines from surfaces and volumes in Section 4 and Section 5, respectively. Experimental results are presented in Section 6. We compare the proposed Laplacian lines with other popular feature lines in Section 7. Finally, we conclude our work and highlight the future work in Section 8.

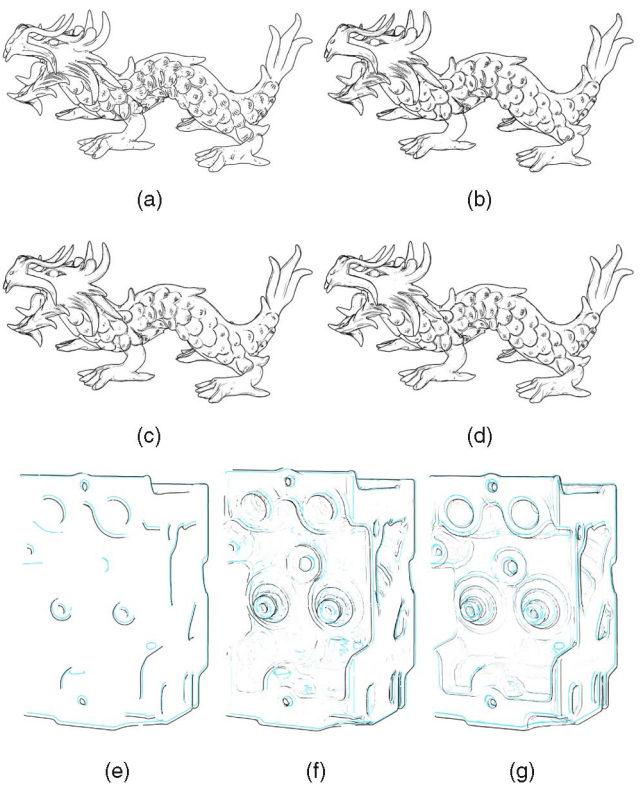


Fig. 1. Laplacian lines (LL) are a set of object-space, view-dependent feature lines, which generalize the Laplacian-of-Gaussian (LoG) edge detector to 3D surfaces and volumes. Laplacian line is efficient in that most expensive computations can be preprocessed and the runtime complexity for LL extraction algorithm is similar to the silhouette extraction. (a)-(d) View-dependent line drawings on 3D surface. (e)-(g) View-dependent line drawings from the volumetric data where the user specifies two isovalues (colored in black and blue). Note that silhouettes are rendered in all drawings in this paper, if not specially stated. Testing platform: a PC with 3.0 GHz CPU and 2.0 G memory. (a) Suggestive contours ($13.6fps$). (b) Apparent ridges ($2.6fps$). (c) PELs ($2.0fps$). (d) Laplacian lines ($16.9fps$). (e) Silhouettes ($4.3fps$). (f) Suggestive contours ($2.8fps$). (g) Laplacian lines ($3.9fps$).

2 Previous Work

There exists extensive literature in computer-generated line drawings and their applications. We refer the readers to Rusinkiewicz et al.'s SIGGRAPH course notes [2] for a comprehensive list.

2.1 Object-Space Line Drawing

The object-space feature lines, usually, are defined by a differential equation on the 3D surfaces. Silhouettes show the strongest cues with model-to-background distinction [20] [21]. However, silhouettes alone are quite limited in conveying shape, since they cannot capture the structure and complexity of the shape interior. DeCarlo et al. [4] proposed suggestive contours that naturally extend contours and convey the shape effectively. To address the problem that suggestive contours do not appear in the convex regions, DeCarlo and Rusinkiewicz [7] introduced highlight lines which complement contours and suggestive contours. Ridge-valley lines are powerful shape descriptors but independent of the view point [5]. Judd et al. [6] proposed apparent ridges that elegantly generalize ridge-valley lines with view-dependent features. Aiming at line drawing

simplification, Ni et al. [22] proposed view-dependently controllable feature lines using preconstructed multiresolution mesh. Inspired by the Canny edge detector, Xie et al. [8] proposed photic extremum lines (PEL) on 3D surfaces. Similar to apparent ridge, PEL is also a third-order feature line, and thus computationally expensive. Kolomenkin et al. [9] defined demarcating curves as the zeros of the normal curvature in the curvature gradient direction. In their recent work [23], Kolomenkin et al. considered the surface as an unknown smooth manifold, on top of which is placed a local height image. Then the relief edge is defined as the edges of the local height image. They showed that relief edges are continuous and smooth, and very promising for artifact illustration in archeology [23]. Kalogerakis et al. [24] proposed a real-time and highly parallel method to compute curvatures and their derivatives for deforming objects. This method is ideal for deforming objects with temporal coherence by accurately predicting the curvatures and their derivatives. However, for static objects, this property does not hold, and the computation for surface curvatures and their derivatives is still expensive. Recently, Zhang et al. presented a GPU method to compute PELs in real time [25].

2.2 Image-Space Line Drawing

Image-space approaches, usually, are easy to implement, since the complicated surface derivatives computations can be avoided. Saito and Takahashi [26] pioneered a method to extract feature lines, using image processing techniques. Buchanan and Sousa [27] proposed the edge buffer data structure, which enables highlighting various feature lines on a polygonal model. Raskar et al. [28] presented a novel NPR camera, which detects depth edges using multiflash images. Winnemöller et al. [29] presented an automatic, real-time video and image abstraction framework, using difference-of-Gaussian edges. Lee et al. [15] proposed an algorithm to automatically extract lines at appropriate scales from abstract shading. Recently, Vergne et al. [30] proposed a novel image-space local shape descriptor to enhance surface depiction.

2.3 NPR from Volume Illustration

Besides surface rendering, nonphotorealistic rendering techniques have been widely used for volumetric data sets. Ebert and Rheingans [31] pioneered the volume illustration approach by combing the physics-based illumination model with NPR techniques to enhance important features. Later on, Svakhine and Ebert [32] improved the volume illustration by incorporating many feature enhancements at interactive speed. Lu et al. [33] presented a framework for interactive direct volume illustration system that simulates traditional stipple drawing. Nagy et al. [34] presented an approach that automatically generates line drawings and Toon shading to interactively enhance the important features in volumetric data sets. Bruckner and Gröller [35] presented a fully dynamic illustration environment for volumetric data sets. Nagy and Klein [36] proposed a high-quality algorithm to render silhouettes. Schein and Elber [37] developed an adaptive algorithm to model and visualize silhouettes from volumetric data with B spline functions. Burns et al. [19] presented an efficient algorithm to draw contours and suggestive contours from volumetric data without extracting the isosurfaces. Dong et al. [38] presented a surface hatching technique for medical volume data. Chen et al. [39] presented a shape-aware volume illustration framework by taking the shape variation into consideration. Inspired by traditional visual and line drawing techniques in medical illustration, Svakhine et al. [40] presented efficient algorithms to effective emphasis/de-emphasis of data and convey their spatial relationship.

3 Definition of Laplacian Lines

The key question for a computer-generated line drawing algorithm is: Where do we draw the lines? Cole et al. [41] conducted a comparative study on where artists made line drawings to convey 3D shapes. They registered a large number of artistic drawings with rendered 3D models and then quantitatively compared the artistic drawings with computer-generated lines, such as occluding contours, suggestive contours, apparent ridges, and Canny edges. The result of the study showed that the Canny edges [42], characterizing the significant changes in illumination, provide the strongest cues for artists to draw lines [41]. This observation motivates a research direction that generalizes the feature detection from 2D images to 3D surfaces.

Inspired by the Canny edge detector [42], Xie et al. [8] proposed photic extremum lines (PELs) that characterize significant changes in the illumination. Given a 3D surface S and the illumination function $I : S \rightarrow \mathbb{R}$ defined on S, they defined PELs as a set of points on the 3D surface, where the variation of illumination in the direction of its gradient reaches the local maximum [8], i.e.,

$$D_{\mathbf{d}}\|\nabla I(\mathbf{p})\| = 0 \quad \text{and} \quad D_{\mathbf{d}}D_{\mathbf{d}}\|\nabla I(\mathbf{p})\| < 0,$$

where ∇ denotes the surface gradient and $\mathbf{d} = \nabla\mathbf{I}/\|\nabla\mathbf{I}\|$. As shown in [8], PELs can produce high quality line drawings for surfaces and volumetric data sets. However, they are computationally expensive due to the involvement of the third and fourth order derivatives. Furthermore, the discrete differential geometry based mesh derivatives are sensitive to noise and irregular tessellation. As a result, a well tessellated and smooth mesh is required in [8].

To improve the performance of PELs and develop an efficient and robust edge detector on 3D surfaces, this paper presents Laplacian lines, which generalizes the Laplacian of Gaussian (LoG) edge detector [43] from 2D images to 3D surfaces. In the 2D image setting, the given image is first convoluted by a Gaussian function and then the zero-crossing of the Laplacian are defined as the edge point. Then, we define Laplacian lines as follows:

Definition. *The Laplacian lines are a set of points* $\mathbf{p}$ *on the 3D surface* S *(with at least* C^3 *continuity), where Laplacian of illumination* $\triangle I$ *passes through zero, and the gradient magnitude* $\|\nabla I\|$ *is greater than the user-specified threshold* τ*, i.e.,*

$$\triangle I(\mathbf{p}) = 0 \quad \text{and} \quad \|\nabla I(\mathbf{p})\| \geq \tau, \tag{1}$$

where $\triangle$ *is the Laplace-Beltrami operator on* S*.*

Note that the illumination I in (1) is a general illumination function, such as the popular Phong illumination and global illumination. However, to simplify the

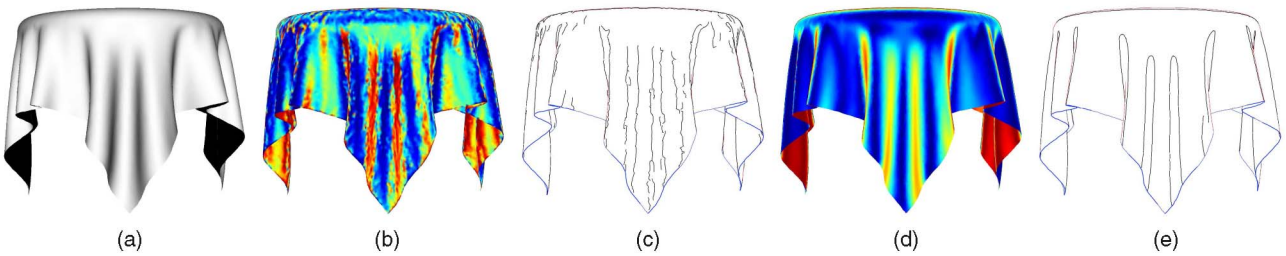


Fig. 2. Mesh Laplace operator versus cotangent Laplace operator. The popular cotangent Laplacian operator is sensitive to the mesh triangulation, and thus does not generate smooth scalar field $\triangle I$. The resulting Laplacian lines contain many short segments and are not smooth (see (b) and (c)). Mesh Laplace operator, on the contrary, is robust to irregular tessellation and generates smooth Laplacian of illumination and Laplacian lines (see (d) and (e)). (a) Shaded image. (b) The color map of $\triangle I$, using the cotangent Laplace operator. (c) LL of (b). (d) The color map of $\triangle I$, using the mesh Laplace operator. (e) LL of (d).

computation, we make two assumptions on the light source and the material of the underlying surface.

- The light source is a point light with unit intensity and locates at the view point $\mathbf{e}$. Then, for any point $\mathbf{p} \in S$, the light vector $\mathbf{l}$ is equal to the viewing vector $\mathbf{e} - \mathbf{p}$.
- The surface exhibits only diffuse reflection, namely, the illumination $I(\mathbf{p}) = \mathbf{n}(\mathbf{p}) \cdot \mathbf{l}(\mathbf{p})$, where $\mathbf{n}(\mathbf{p})$ is the normal of the point $\mathbf{p} \in S$.

We show that the above assumptions lead to a nice property to separate the view-independent component from the Laplacian of illumination. Given the local coordinate system x^i of S, let $\partial_i := \partial/\partial x^i$ be the partial derivative along x^i. Let g_{ij}, g^{ij}, b_{ij} denote the covariant metric tensor, contravariant metric tensor, and coefficient of second fundamental form, respectively. Let H be the mean curvature. Then, Laplacian of illumination $\triangle I$ can be simplified as follows:

$$\begin{aligned}\triangle I(\mathbf{p}) &= \triangle(\mathbf{n} \cdot (\mathbf{e} - \mathbf{p})) \\ &= (\triangle \mathbf{n}) \cdot (\mathbf{e} - \mathbf{p}) + \mathbf{n} \cdot \triangle(\mathbf{e} - \mathbf{p}) + g^{ij}\partial_i \mathbf{n} \cdot \partial_j(\mathbf{e} - \mathbf{p}) \\ &= (\triangle \mathbf{n}) \cdot (\mathbf{e} - \mathbf{p}) - \mathbf{n} \cdot \triangle \mathbf{p} + g^{ij} b_{ij} \\ &= (\triangle \mathbf{n}) \cdot (\mathbf{e} - \mathbf{p}) - 2H\mathbf{n} \cdot \mathbf{n} + 2H \\ &= (\triangle \mathbf{n}) \cdot (\mathbf{e} - \mathbf{p}) = (\triangle \mathbf{n}) \cdot \mathbf{l}.\end{aligned} \tag{2}$$

Note that $\triangle \mathbf{n}$ is view-independent and can be precomputed. Therefore, the runtime Laplacian line extraction is more efficient than that of PELs by avoiding on-the-fly time-consuming derivative computation. In particular, the complexity of Laplacian line extraction is similar to the silhouette extraction, where only the Laplacian of normal is replaced by the normal.

Remark. We use the viewing vector $\mathbf{l}$ rather than the normalized vector $\mathbf{l}/\|\mathbf{l}\|$ in (2) to simplify the derivation. As shown in Fig. 3, the Laplacian of the illuminations $\triangle(\mathbf{n} \cdot \mathbf{l})$ and $\triangle(\mathbf{n} \cdot \mathbf{l}/\|\mathbf{l}\|)$ are highly consistent after scaling both to the same range. Therefore, the corresponding Laplacian lines are also consistent.

4 Computing Surface Laplacian Lines

4.1 Robust Laplace Operator

This section shows the detailed algorithm to compute the Laplacian lines on 3D surfaces. The key component in Laplacian line extraction is to compute the Laplacian of surface normal $\triangle \mathbf{n}$. Many discrete Laplace-Beltrami operators exist [44]. However, the popular cotangent Laplace operator depends highly on the mesh tessellation and tends to be sensitive to noise [10]. As shown in Fig. 2, the given table cloth model is smooth but with a irregular tessellation (as revealed by the color map of the $\triangle I$ shown in Fig. 2b). As a result, the extracted Laplacian lines contain many short segments and are not visually pleasing (Fig. 2c).

To develop a robust Laplacian line extraction algorithm, we use the mesh Laplace operator proposed in [17]: Given a function $f : M \to \mathbb{R}$ defined on the mesh M, the mesh Laplace operator L_M^h is defined as follows:

$$L_M^h f(\mathbf{p}) = \frac{1}{4\pi h^2(\mathbf{p})} \sum_{\triangle_i \in S} \frac{A(\triangle_i)}{3} \sum_{\mathbf{q} \in \triangle_i} e^{-\frac{\|\mathbf{q}-\mathbf{p}\|^2}{4h(\mathbf{p})}} (f(\mathbf{q}) - f(\mathbf{p})), \tag{3}$$

where $A(\triangle_i)$ denotes the area of triangle $\triangle_i$ and $h(\mathbf{p})$ is a positive quantity, which intuitively corresponds to the size of the neighborhood considered at point $\mathbf{p}$. When mesh M is a sufficient approximation of a smooth underlying surface S, L_M^h is close to the continuous Laplace-Beltrami operator [17].

The Gaussian kernel h in (3) is closely related to the number of extracted Laplacian lines. Intuitively speaking, h is a smoothing factor, the larger the value of h, the smoother

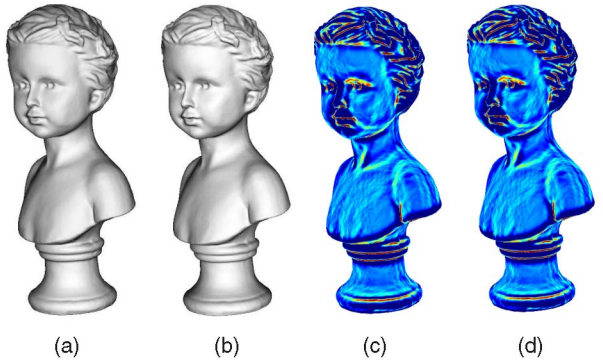


Fig. 3. Comparison of $\triangle(\mathbf{n} \cdot \mathbf{l})$ and $\triangle(\mathbf{n} \cdot \mathbf{l}/\|\mathbf{l}\|)$. (a) Illumination of $\frac{1}{d}\mathbf{n} \cdot \mathbf{l}$, where d is a user-specified *constant* to scale $\mathbf{n} \cdot \mathbf{l}$ into a reasonable range such that we can view it. We set d to be the distance from the view point to the center of the object. (b) Standard illumination $I = \mathbf{n} \cdot \mathbf{l}/\|\mathbf{l}\|$. Note that $\triangle(\mathbf{n} \cdot \mathbf{l}/\|\mathbf{l}\|)$ and $\triangle(\mathbf{n} \cdot \mathbf{l})$ have different ranges. Thus, we scale both into $[0, 1]$, and then use color map to visualize them as shown in (c) and (d). The scaled Laplacians are highly consistent.

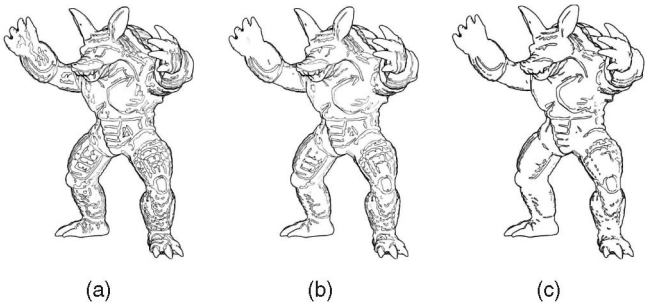


Fig. 4. The size of Gaussian kernel h in mesh Laplace operator is closely related to the number of extracted Laplacian lines. The larger the kernel size, the more smoothing effects obtained, thus, the fewer number of Laplacian lines extracted. The model is normalized to a unit cube. (a) $h = 0.01$. (b) $h = 0.03$. (c) $h = 0.05$.

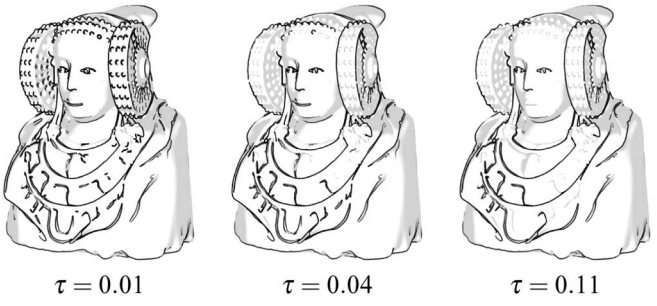


Fig. 5. Trimming the Laplacian lines with the user-specified threshold τ. Increasing the threshold results in less number of extracted Laplacian lines.

the Laplacian of illumination $\triangle I$ and fewer the number of Laplacian lines we obtain. Fig. 4 shows the smoothing effect of specifying h from small to large value.

We would like to emphasize the the analogy between the LoG edge detector and mesh Laplace operator. In edge detection, the image is first convoluted with a Gaussian filter to reduce the noise, and then followed by the discrete Laplace operator to detect the zero-crossing points. In mesh Laplace operator, see (3), the Gaussian filter with kernel h is also applied to the function value. In some sense, the mesh Laplace operator is a Laplace-of-Gaussian operator.

Compared to the cotangent Laplacian, the mesh Laplace operator is numerically stable due to the positivity and smoothing effects of the weights. As shown in Fig. 2, mesh Laplace operator produces smoother $\triangle I$ and Laplacian lines than the cotangent Laplacian. Though the mesh Laplace operator is more computationally expensive than cotangent Laplace operator, $\triangle \mathbf{n}$ can be computed in the preprocessing step and the runtime line extraction algorithm is still simple and efficient.

4.2 Laplacian Line Extraction Algorithm

Given a triangle mesh M, our Laplacian line extraction algorithm takes two parameters, h (see (3)) and τ (see (1)), and consists of four consecutive steps:

1. (Preprocessing) For each vertex, compute $\triangle \mathbf{n}$ using (3).
2. For each vertex $\mathbf{p}$, compute the dot product of the viewing vector $\mathbf{v} = \mathbf{e} - \mathbf{p}$ and $\triangle \mathbf{n}(\mathbf{p})$, and detect the zero-crossing of $\mathbf{v} \cdot \triangle \mathbf{n}(\mathbf{p})$.
3. For each zero-crossing point $\mathbf{p}$, compute $\|\nabla I(\mathbf{p})\|$ and filter out the ones whose magnitude of gradient is less than the user-specified threshold.
4. Trace the filtered zero-crossings to get the Laplacian lines.

Step 1. We compute the vertex normal, using the conventional discrete algorithm, i.e., weighted sum of the per-face normals. Similar to LoG edge detector, where a Gaussian filter is applied to reduce the image noise, we can also reduce the illumination noise by smoothing the vertex normals using the Gaussian or bilateral filter. In our experiments, we found that the illumination $I = \mathbf{n} \cdot \mathbf{l}$ is smooth for clean data. Note that the mesh Laplace operator also has smoothing effects; thus, smoothing normals for clean data is not necessary. However, for noisy meshes or meshes with very bad triangulation, the discrete normal algorithm leads to poor results, and this smoothing step is very helpful to reduce the noise of illumination. Then, we compute $\triangle \mathbf{n}$, using (3) for each vertex. The parameter h is specified by the user.

Step 2. In the runtime step, we first compute the dot product of the viewing vector $\mathbf{v}$ and $\triangle \mathbf{n}$ for each vertex. To locate the zero-crossings, we follow the method in [5]. For an edge $[\mathbf{v}_1, \mathbf{v}_2]$, if $\triangle I(\mathbf{v}_1) \triangle I(\mathbf{v}_2) < 0$, we use linear interpolation to locate a zero-crossing on edge $[\mathbf{v}_1, \mathbf{v}_2]$ with

$$\mathbf{p} = \frac{|\triangle I(\mathbf{v}_1)|\mathbf{v}_2 + |\triangle I(\mathbf{v}_2)|\mathbf{v}_1}{|\triangle I(\mathbf{v}_1)| + |\triangle I(\mathbf{v}_2)|},$$

and consider $\mathbf{p}$ a point on a Laplacian line. Note that this step is the same as the conventional silhouette extraction algorithm, except that the normal $\mathbf{n}$ is replaced by the Laplacian of normal $\triangle \mathbf{n}$.

Step 3. For each zero-crossing point $\mathbf{p}$, compute the magnitude of gradient. In our prototype system, we implement two methods to compute the gradient magnitude. For smooth meshes with good triangulation, we use the popular discrete differential geometry approach [10], [11]. For the noisy mesh, we solve the following optimization problem:

$$\min \sum_{\|\mathbf{p}-\mathbf{q}\| \le h} |f(\mathbf{q}) - f(\mathbf{p}) - \nabla f(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})|^2, \qquad (4)$$

where $\mathbf{q} \in \mathbf{M}$ is within the distance h of $\mathbf{p}$. Note that the computation of gradient magnitude is only applied to the zero-crossing points.

Step 4. We trace the detected zero-crossing to get the feature lines. We use the following integral to measure the strength of each line:

$$\int \|\nabla I\| ds \approx \sum_i \frac{\|\nabla I(\mathbf{p}_i)\| + \|\nabla I(\mathbf{p}_{i+1})\|}{2} \|\mathbf{p}_i - \mathbf{p}_{i+1}\|. \qquad (5)$$

Finally, we delete the feature lines whose strengths are less than the user-specified threshold τ as shown in Fig. 5.

5 Computing Volume Laplacian Lines

Given a volume data set and a user specified isovalue, volume Laplacian line is equivalent to the surface Laplacian lines on the corresponding isosurface. Thus, a naïve

solution would be to extract the isosurface and then apply the surface Laplacian line algorithm. However, this strategy cannot lead to a real-time line drawing system even for volume data sets of small size, since it is not practical to pre-extract all isosurfaces and then precompute the Laplacian of normals of the extracted isosurfaces. A better solution is to directly compute the volume Laplacian lines in the volume data sets without isosurface extraction.

In this section, we derive the formula to directly compute the volume Laplacian lines. We use superscripts for vector components and subscripts for partial derivatives with respect to x^i, $i = 1, 2, 3$. Let $f : \Omega \subseteq \mathbb{R}^3 \rightarrow \mathbb{R}$ denote the volumetric data set. Given an isovalue c, define $F(x^1, x^2, x^3) = f(x^1, x^2, x^3) - c$ and denote $S = \{(x^1, x^2, x^3) | F(x^1, x^2, x^3) = 0\}$ the isosurface. To avoid ambiguity in this section, we use ∇_S and $\triangle_S$ to denote the gradient and Laplace-Beltrami operators defined on the isosurface S and ∇ and $\triangle$ the operators defined in euclidean space $\mathbb{R}^3$.

Let $\mathbf{g} = \nabla F = (F_1, F_2, F_3)$ be the gradient of F and $g = \|\mathbf{g}\|$ be the magnitude. Then the normal of S is given by $\mathbf{n} = -\mathbf{g}^T/g$. The Gaussian curvature K and mean curvature H of S are given by [45], [46]:

$$K = -\frac{\det\begin{vmatrix} \mathbf{H} & \mathbf{g}^T \\ \mathbf{g} & 0 \end{vmatrix}}{g^4}, \tag{6}$$

$$H = \frac{\mathbf{g}\mathbf{H}\mathbf{g}^T - g^2\mathtt{Trace}(\mathbf{H})}{2g^3}. \tag{7}$$

where $\mathbf{H} = (F_{ij})_{i,j=1,2,3}$ is the 3×3 Hessian matrix of F.

By the elegant formula of Weatherbrun [18], we compute the Laplacian of normal as

$$\triangle_S \mathbf{n} = (2K - H^2)\mathbf{n} - 2 \bigtriangledown_S H, \tag{8}$$

where $\bigtriangledown_S H = \bigtriangledown H - (\bigtriangledown H \cdot \mathbf{n})\mathbf{n}$ is on the tangent plane.

By straightforward computation, the gradient of mean curvature $\bigtriangledown H = (H_1, H_2, H_3)$ is

$$\begin{aligned} H_k = &\sum_{i=1}^{3}\sum_{j=1}^{3}\left(\frac{A_{ijk}}{2g^3} - \frac{3B_k}{2g^5}F_iF_jF_{ij}\right) \\ &- \sum_{i=1}^{3}\left(\frac{F_{iik}}{2g} - \frac{B_k}{2g^3}F_{ii}\right), \quad k = 1, 2, 3, \end{aligned} \tag{9}$$

where $A_{ijk} = F_iF_{jk}F_{ij} + F_jF_{ik}F_{ij} + F_iF_jF_{ijk}$ and $B_k = \sum_{i=1}^{3} F_iF_{ik}$.

In order to efficiently and robustly compute the derivatives F_i, F_{ij}, and F_{ijk}, we extend Vieville and Faugeras's algorithm [47] from 2D images to 3D volume data sets. Given a voxel (x_0^1, x_0^2, x_0^3), we can approximate the local function value by Taylor expansion

$$\begin{aligned} \tilde{f}(x^1, x^2, x^3) \approx &f + \sum_{i=1}^{3} f_i d^i \\ &+ \sum_{i=1}^{3}\sum_{j=1}^{3}\frac{f_{ij}}{2}d^id^j + \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\frac{f_{ijk}}{6}d^id^jd^k \end{aligned} \tag{10}$$

with $d^i = x^i - x_0^i$, and f, f_i, f_{ij}, and f_{ijk} denote the function and its derivatives at the voxel (x_0^1, x_0^2, x_0^3). Then the function value of a nearby voxel (x_1^1, x_1^2, x_1^3) is approximately equal to the integration of $\tilde{f}$ over the voxel space, i.e.,

$$\begin{aligned} f(x_1^1, x_1^2, x_1^3) \approx &\int_{-\frac{1}{2}}^{\frac{1}{2}}\int_{-\frac{1}{2}}^{\frac{1}{2}}\int_{-\frac{1}{2}}^{\frac{1}{2}} \tilde{f}(x_1^1 + x^1, x_1^2 + x^2, \\ &x_1^3 + x^3)dx^1dx^2dx^3. \end{aligned} \tag{11}$$

Let

$$P_n^i = \int_{x_1^i - x_0^i - \frac{1}{2}}^{x_1^i - x_0^i + \frac{1}{2}} \frac{(x^i)^n}{n!}dx^i.$$

Then, we have

$$\begin{aligned} f(x_1^1, x_1^2, x_1^3) \approx &f + \sum_{i=1}^{3}(f_iP_1^i + f_{ii}P_2^i + f_{iii}P_3^i) \\ &+ \sum_{i=1}^{3}\sum_{i\neq j}(f_{ij}P_1^iP_1^j + f_{iij}P_2^iP_1^j) \\ &+ f_{123}P_1^1P_1^2P_1^3. \end{aligned} \tag{12}$$

There are 20 unknowns in (12). Given sufficient number of neighboring voxels, we can convert the problem to an overconstrained linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ that can be solved by least-square as $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$. Note that $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the same for all voxels, and only $\mathbf{b}$ contains the neighboring voxel values. In this way, each derivative is actually solved by a simple filter operation. The filter size is related to the quality of the derivatives. Usually, the larger the filter size, the smoother and more robust the computed derivatives. In our experiments, we found that $5 \times 5 \times 5$ or $7 \times 7 \times 7$ filters lead to satisfying results for all tested models.

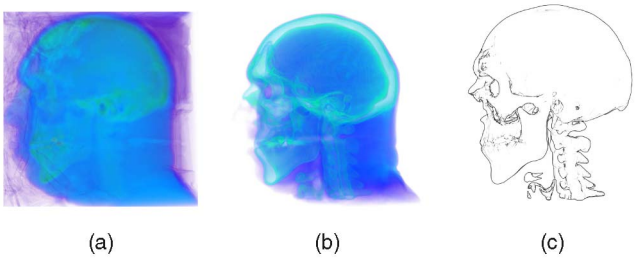


Fig. 6. Extracting volume Laplacian lines from the Head model. (a) The volume rendering result. (b) The volume rendering of $\triangle I$. (c) Volume Laplacian lines corresponding to (b).

The algorithm for extracting Laplacian lines is as follows:

Step 1. (Preprocessing) Compute for each voxel the derivatives F_i, F_{ij}, F_{ijk} by filtering the volumetric data set, and then compute $\triangle_S \mathbf{n}$, using (8).

Step 2. To trace volumetric Laplacian lines, we follow Burns et al.'s seed-and-traverse approach [19]. First, a set of random voxels are chosen as *seed*s. For each seed, we check whether it intersects with the isoline, i.e., the intersection between the isosurface and the surface satisfying $\triangle_S I = 0$. We adopt the *Marching Line* algorithm [48] to perform such test. Specifically, we check whether each face of the underlying voxel intersects the isoline. If yes, we trace the neighboring voxel with respect to that face and then we perform a similar test on that voxel. The tracing procedure continues until we get back to the seed voxel or reach the volume boundary. Note that this step is the same as that of

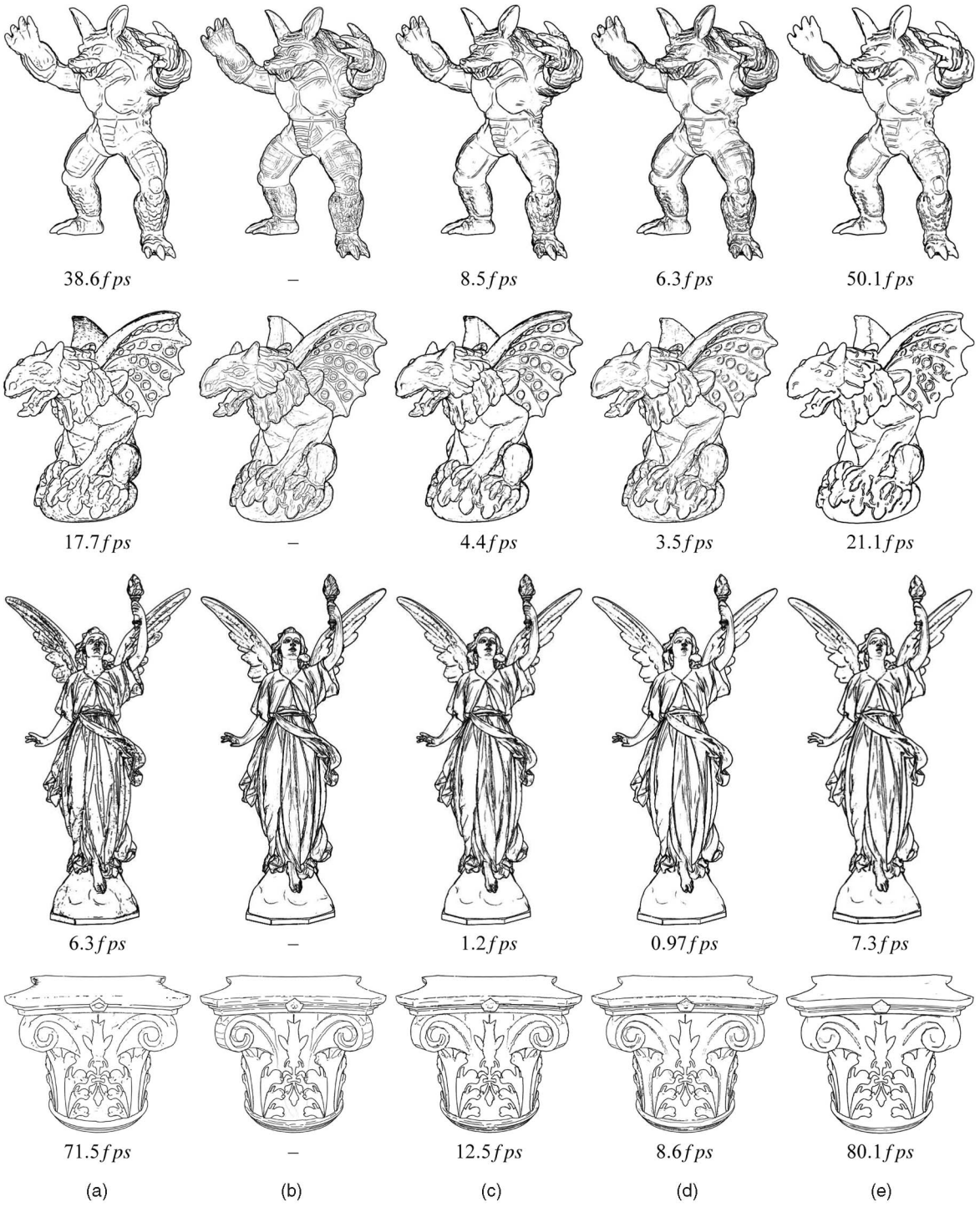


Fig. 7. Laplacian lines show comparable results with other object-space feature lines and are more efficient. Ridge-valley lines, suggestive contours and apparent ridges are rendered with the *rtsc* software [4]. (a) Suggestive contour. (b) Ridge-valley lines. (c) Apparent ridges. (d) PELs. (e) Laplacian lines.

silhouette extraction from volumetric data sets in [19], except that we replace normals $\mathbf{n}$ with Laplacian of normals $\triangle_S \mathbf{n}$.

Step 3. We test the visibility of the extracted Laplacian lines by the ray casting approach as described in [19]. We cast a ray from each point of extracted Laplacian lines to the view point and test whether the ray intersects with the isosurface of the current isovalue. Note that this approach does not need to explicitly construct the isosurface. Then the user can choose to either draw the hidden Laplacian lines with a different color from the visible lines or simply ignore them.

Step 4. Finally, we compute $\| \bigtriangledown I \|$ for each point of extracted Laplacian lines by means of finite differences, and filter out the false lines with small intensity change $\| \bigtriangledown I \| \leq \tau$, where τ is the user-specified threshold.

As stated earlier, the proposed volume Laplacian lines can be computed directly in the volumetric data sets without isosurface extraction. The user only needs to

TABLE 1
Performance Measurement of Surface Laplacian Lines

Models	#△	f_S	f_{SC}	f_{AR}	f_{PEL}	f_{LL}
Armadillo	330K	64.1	38.6	8.5	6.3	50.1
Carriage	600K	27.4	18.4	2.8	2.4	20.7
Cathedral	1.2M	22.2	12.1	2.5	1.9	13.6
Children	400K	37.5	27.7	6.6	4.8	29.3
Column	200K	94.3	71.5	12.5	8.6	80.1
Dragon	1M	22.5	13.6	2.6	2.0	16.9
Gargoyle	600K	28.5	17.7	4.4	3.5	21.1
Happy Buddha	260K	75.3	57.2	10.1	8.2	65.5
Lucy	2.1M	11.9	6.3	1.2	1.0	7.3
Monster head	3M	8.0	5.4	0.93	0.7	6.0
Pegaso	150K	115	80.4	14.9	10.9	82.8
Rosace	250K	76.4	61.2	9.9	7.9	67.4
Saint Mary	100K	165	105	21.3	18.2	116
Statute head	300K	70.5	55.3	9.8	7.6	56.3
Thai statute	2M	13.1	6.9	1.6	1.2	7.4

The statistics for silhouettes, suggestive contours and apparent ridges are obtained using the rtsc software [4]. #△: number of triangles; f_S, f_{SC}, f_{AR}, f_{PEL} and f_{LL} are the frame-per-second (fps) for silhouettes, suggestive contours, apparent ridges, PELs and Laplacian lines, respectively.

specify the desired isovalue(s) and the threshold τ to filter out the false Laplacian lines. Note that Xie et al. computed the PELs in volumetric data sets, using isosurface extraction that is time consuming and very difficult to control as the users cannot see the extracted lines in real-time [8]. Compared to PELs, the proposed volumetric Laplacian lines are a truly view-dependent line in volumetric data sets. Furthermore, it is flexible, efficient and easy to implement. Fig. 6 shows the volumetric Laplacian lines of the head model.

6 Experimental Results

We tested Laplacian lines for a wide range of surface and volumetric models on a PC with 3.0 GHz CPU and 2 GB memory.

6.1 Surface Laplacian Lines

For surface models, our current CPU-based implementation can generate Laplacian lines at interactive frame rates for all test models with up to three million faces. We compared the performance of Laplacian lines with other popular view-dependent feature lines, such as silhouettes, suggestive contours, apparent ridges and PELs with the same hardware configuration. To make the comparison fair, we tune up the parameters of each feature line (except the silhouette) to make the generated line draws have similar appearances (i.e., similar number of lines). Fig. 7 shows the visual comparison of various feature lines on 3D surfaces, and Table 1 shows the complexity of the test models and the detailed performance evaluation. As mentioned in Section 4, Laplacian line extraction algorithm is similar to the silhouette, except that a trimming process is required to remove short and false Laplacian lines. Also, Laplacian lines usually contain much more number of features than silhouettes. Therefore, rendering Laplacian line is slower than rendering silhouettes. Suggestive contour is a second-order feature that the radial curvature vanishes. In the implementation of suggestive contours, the viewing vector is first projected to the tangent plane and then the radial curvature is computed as the linear combination of principal curvatures which are also precomputed. Laplacian lines only compute the dot product of the viewing vector and the precomputed $\triangle\mathbf{n}$; thus, it is more efficient than suggestive contours. Apparent ridges and PELs are computationally expensive due to the third- and fourth-order derivatives are computed at runtime. Thus, Laplacian lines are much more efficient than apparent ridges and PELs.

Fig. 8 shows Laplacian lines for various large-scale 3D surfaces of complex geometry and fine details. Laplacian lines can also be naturally combined with NPR shading to convey the shapes more effectively. As shown in Fig. 9, most of the Laplacian lines coincide with the edges of the Toon shading that makes the rendering results natural and visually pleasing.

6.2 Volume Laplacian Lines

Since volumetric data sets usually contain much more number of data (i.e., voxels) than that of the surface (i.e., vertices); it requires great care to design the volumetric Laplacian line algorithm. Given the user-specified isovalue, the brute-force algorithm simply goes through every voxel that both satisfies (1) and intersects the isosurface. As pointed out in [19], the extracted lines in volumetric data set with N voxels is of order $\sqrt[3]{N}$, i.e., the majority of the voxels do not contain the lines. Thus, the brute-force algorithm is not efficient and can hardly lead to real-time line drawing even for small scale data sets. To improve the performance, we followed the seed-and-traversal algorithm proposed by Burns et al. [19]. Rather than checking every voxel, the seed-and-traversal approach starts from random seeds. If one seed happens to contain the point of Laplacian line, then we start to trace the Laplacian line from the current seed. The tracing will either come back to the seed or stop on the boundary of the volumetric data sets. We typically used 50,000 random seeds per frame, as suggested by Burns et al. [19]. And in our experiment, the number of random seeds does not affect the performance too much. What dominates the efficiency is the number of features in the underlying model. This approach works well in practice and does not generate the flickering artifacts, as demonstrated in the figures and accompanied video.

Table 2 shows the statistics of Laplacian lines, silhouettes and suggestive contours on volumetric data sets. Note that the performance of the seed-and-traversal approach depends highly on the number of extracted lines. Given the same view point, Laplacian lines, silhouettes and suggestive contours usually generate very different number of extracted lines. Our experiments show that Laplacian lines and suggestive contours usually contain much more lines than silhouettes. Thus, the widely-used fps (frames-per-second) is not enough to measure the true complexity of the above lines. In our experiments, we also compute the points-per-frame ppf to measure the total number of extracted feature points for each frame. Thus, the product $pps = ppf \times fps$ measures the total number of extracted feature points per second (points-per-second). As shown in Table 2, the inequality $pps_{LL} > pps_S > pps_{SC}$ holds for most of the test models. The reason $pps_{LL} > pps_{SC}$ is that

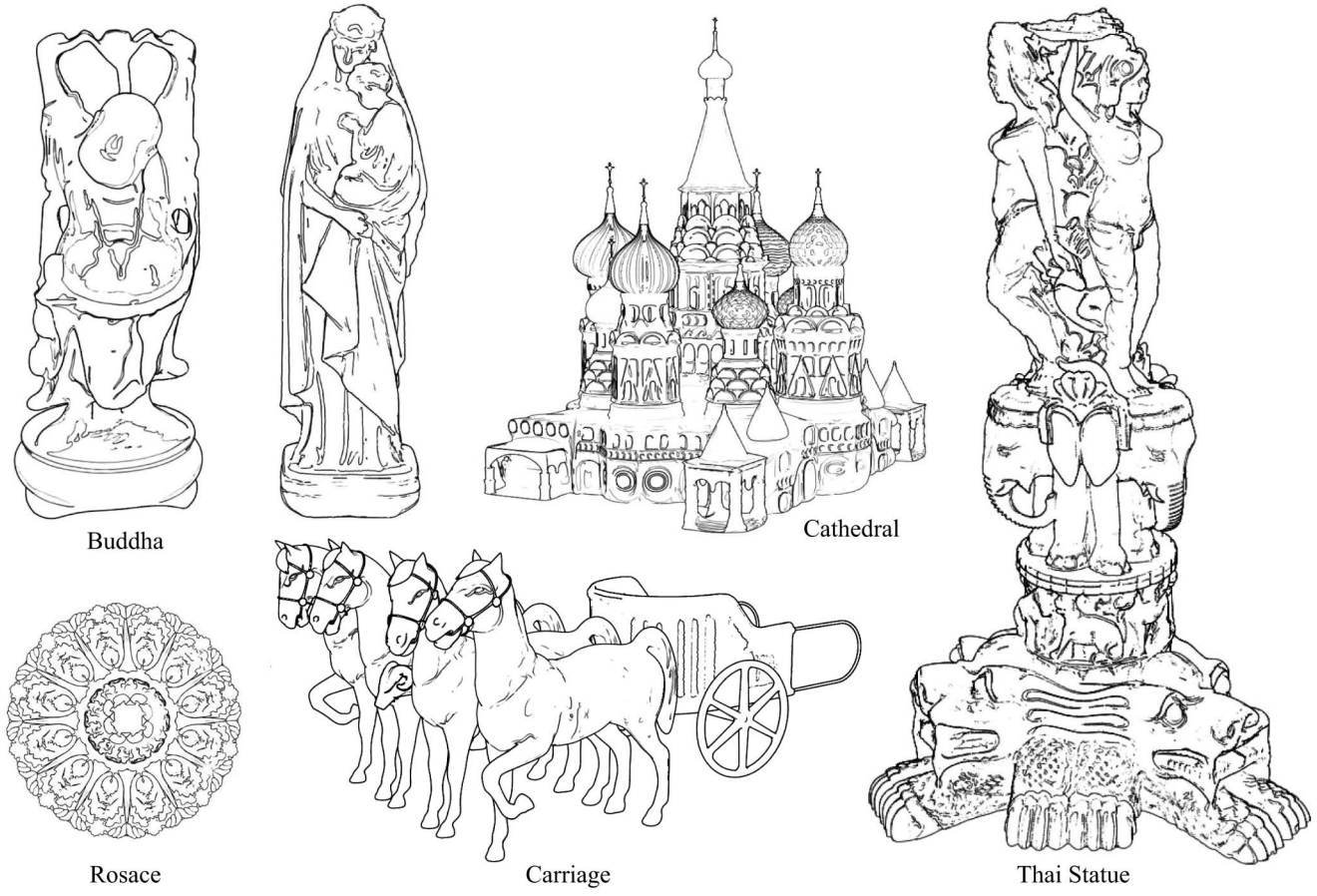


Fig. 8. Conveying 3D surfaces, using Laplacian lines.

Fig. 9. Conveying 3D surfaces, using Laplacian lines and Toon shading.

TABLE 2
Performance of Volumetric Laplacian Lines, Suggestive Contours, and Silhouettes Using the Seed-and-Traversal Approach [19]

Models	Resolution	ppf_S	ppf_{SC}	ppf_{LL}	fps_S	fps_{SC}	fps_{LL}	pps_S	pps_{SC}	pps_{LL}
Angio	$256 \times 320 \times 128$	10K	15K	23K	16.4	8.0	7.3	161K	120K	165K
Carp	$256 \times 256 \times 512$	52K	103K	124K	5.9	2.8	2.5	305K	257K	350K
Chest	$384 \times 384 \times 240$	30K	91K	250K	7.7	2.2	1.2	231K	207K	288K
Engine	$256 \times 256 \times 110$	9K	44K	61K	16.5	4.0	3.7	150K	149K	247K
Knee	$379 \times 229 \times 305$	43K	168K	218K	10.4	1.9	2.1	291K	253K	283K
Male	$128 \times 256 \times 256$	16K	50K	67K	14.2	4.0	4.3	222K	209K	268K
Teapot	$256 \times 256 \times 178$	7K	60K	48K	17.8	3.9	4.1	129K	196K	254K
Tooth	$256 \times 256 \times 161$	2K	10K	8K	22.4	9.0	9.0	44K	70K	93K
Ventricles	$256 \times 256 \times 124$	21K	55K	63K	12.0	4.1	4.6	243K	213K	253K

The fps is proportional to the number of processed voxels. ppf is the average number of processed points per frame and $pps = ppf \times fps$ measures the total number of extracted points per second.

volumetric Laplacian lines have a smaller per-voxel computation cost than suggestive contours. On the other hand, $pps_{LL} > pps_S$ because Laplacian lines are much more denser than silhouettes. Consequently, most randomly selected seed voxels do not intersect the underlying isoline. Such computation costs are not included in the pps metric. Fig. 10 shows the performance of suggestive contours and volumetric Laplacian lines.

To make a more fair comparison, we also tried a simple *bucket* based approach. Given a volumetric model, we uniformly divide its voxel value range into a set of intervals. And for each interval, we use a bucket to store all voxels whose values are in the interval. When the user specifies an isovalue at runtime, only the voxels in the corresponding bucket are processed. In this approach, the same number of voxels is processed for all feature lines. As shown in Table 3, volumetric Laplacian lines have very similar efficiency with silhouettes, and are faster than suggestive contours.

Fig. 11 shows volumetric Laplacian lines on various volumetric data sets. We use color to distinguish Laplacian lines of different isovalues. For more interactive demos, please see the accompanying video.

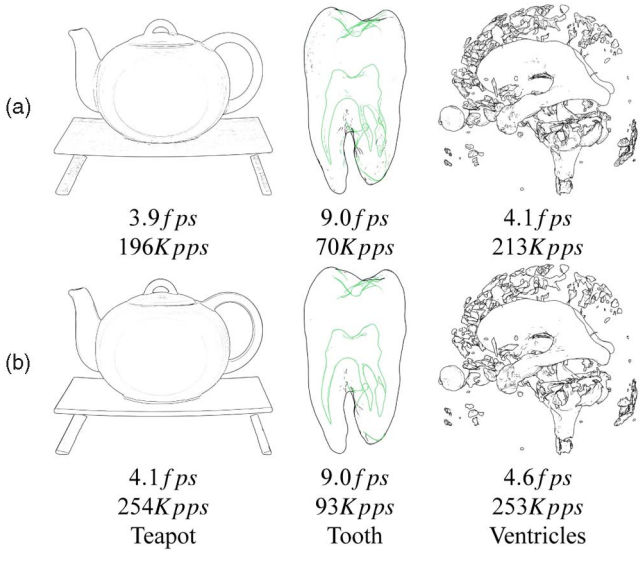


Fig. 10. Comparison of the performance of volumetric Laplacian lines with suggestive contours, using the seed-and-traverse approach [19]. (a) Suggestive contours. (b) Laplacian lines. pps, the number of extracted feature points per second, roughly measures the complexity of the per-voxel computation. The computation of Laplacian lines is less expensive than suggestive contours.

7 Comparisons and Discussions

7.1 Comparison to Silhouettes

Silhouettes are a set of first-order feature lines that show strongest cues with model-to-background distinction. Laplacian lines are different from silhouettes in general in that they are third-order features. However, Laplacian lines coincide with silhouettes in certain areas.

In case of the point light, the lighting vector $\mathbf{l}$ is equivalent to the viewing vector $\mathbf{v}$, i.e., $\mathbf{l} = \mathbf{v} = \mathbf{e} - \mathbf{s}$. Note that ∇H is on the tangent plane of $\mathbf{p}$. Let $\mathbf{w}$ be the projection of $\mathbf{l}$ onto the tangent plane of $\mathbf{p}$. By Equation (8), Laplacian lines satisfy

$$\begin{aligned}(\triangle \mathbf{n}) \cdot \mathbf{l} &= (2K - H^2)\mathbf{n} \cdot \mathbf{l} - 2(\nabla H) \cdot \mathbf{l} \\ &= (2K - H^2)\mathbf{n} \cdot \mathbf{v} - 2(\nabla H) \cdot \mathbf{w} \\ &= (2K - H^2)\mathbf{n} \cdot \mathbf{v} - 2D_{\mathbf{w}}H = 0.\end{aligned} \tag{13}$$

Clearly, the Laplacian lines coincide with the silhouettes if and only if $D_{\mathbf{w}}H = 0$, i.e., the mean curvature is constant or near constant along the viewing direction. This explains that the silhouettes usually coincide with the Laplacian lines for the local neighborhood of points, where the mean curvature does not change too much along the viewing direction (see Fig. 12).

From the implementation point of view, Laplacian line extraction is very similar to the silhouette extraction except that the normal $\mathbf{n}$ is replaced by Laplacian of normal $\triangle \mathbf{n}$, and filtering and trimming are applied to the zero-crossing points. Note that the number of zero-crossing points is much less than the number of the vertices in the given model, and these post-processing steps take only a small

TABLE 3
Performance of Volumetric Laplacian Lines, Suggestive Contours, and Silhouettes Using the Bucket-Based Approach

Models	Resolution	f_S	f_{LL}	f_{SC}
Angio	$256 \times 320 \times 128$	13.6	13.4	10.1
Carp	$256 \times 256 \times 512$	2.8	2.7	1.9
Chest	$384 \times 384 \times 240$	2.0	1.7	1.5
Engine	$256 \times 256 \times 110$	4.3	3.9	2.8
Knee	$379 \times 229 \times 305$	1.7	1.4	1.0
Male	$128 \times 256 \times 256$	4.1	3.6	2.7
Teapot	$256 \times 256 \times 178$	3.6	3.4	2.4
Tooth	$256 \times 256 \times 161$	18.9	17.2	12.2
Ventricles	$256 \times 256 \times 163$	6.3	6.2	4.6

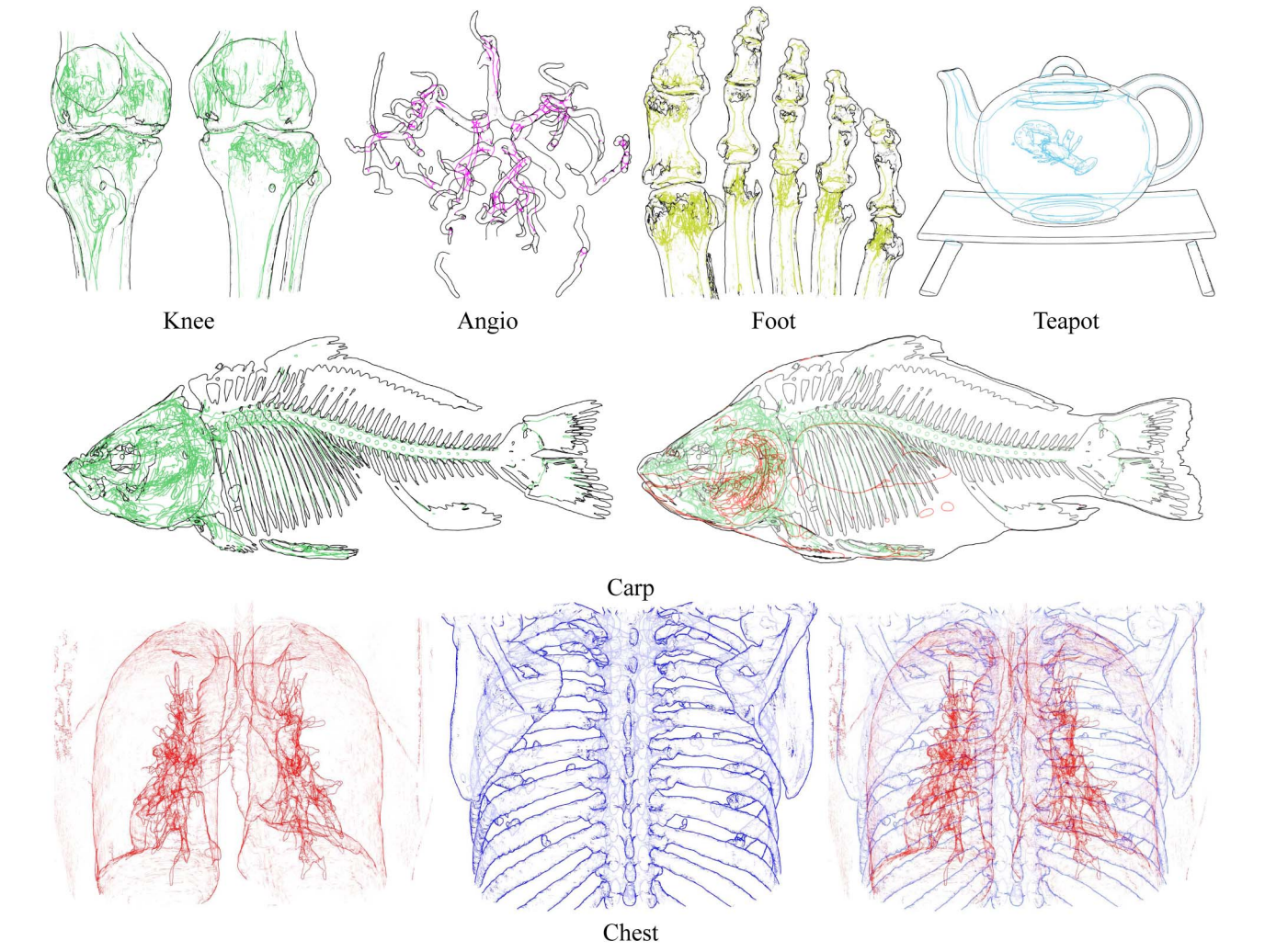


Fig. 11. Laplacian lines for volumetric data sets. Laplacian lines of different isovalues are rendered in different colors.

amount of time. Therefore, rendering Laplacian lines is slightly slower than that of silhouettes. Our experimental results show that the performance of rendering silhouettes (in terms of fps) is 1.2 times more than the Laplacian lines (see Table 1).

7.2 Comparison to Suggestive Contours

Suggestive contours are second-order features, which naturally extend the contours in concave regions. Note that the suggestive contours algorithm can also precompute the curvature, and thus the performance of suggestive contours is similar to Laplacian lines. However, suggestive contours must be used together with silhouettes, since they cannot illustrate salient features in convex regions. In contrast, Laplacian lines work well on both convex and concave regions. In many cases, Laplacian lines, without silhouettes, can produce satisfactory rendering results.

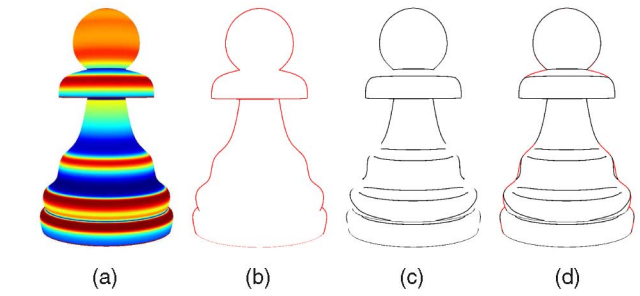


Fig. 12. Laplacian lines are very close to silhouettes, if the mean curvature is constant or near constant along the viewing direction, i.e., $D_{\mathbf{w}}H = 0$, where $\mathbf{w}$ is the projection of viewing vector to the tangent plane. (a) Mean curvature. (b) Silhouettes. (c) LL. (d) Silhouettes $+$LL.

7.3 Comparison to Ridge-Valley Lines, Apparent Ridges, and PELs

Ridge-valley lines, apparent ridges, PELs and Laplacian lines are third-order feature lines. Ridges and valleys are curves where the surface bends sharply. Ridge-valley lines are determined by the geometry and independent of the view point, thus, they can hardly make a natural looking line drawing in an animation sequence. By introducing view-dependent curvatures, apparent ridges elegantly extend ridge-valley lines, yielding more perceptually pertinent results. PELs generalize the Canny edge detector to 3D surface and characterize the significant changes of illumination. Note that the proposed Laplacian line approach is also an extension of image edge detection. As expected, Laplacian lines can generate similar results as PELs (see Figs. 13c

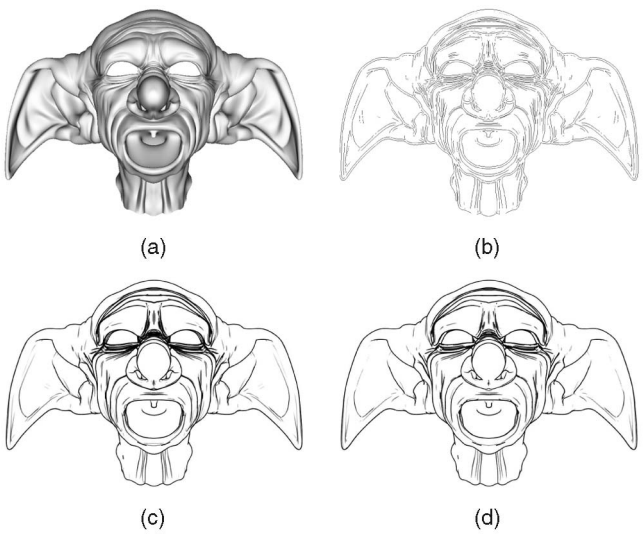


Fig. 13. Comparison of Laplacian lines with LoG edge detector and PELs. Applying LoG edge detector directly to the rendered image does not generate satisfactory results. Note the short lines in (b). Clearly, the object space algorithm (d) is much more robust. In addition, PELs and Laplacian lines are both generalized from image edge detectors. As shown in (c) and (d), the results are very similar. However, Laplacian lines are much more efficient than PELs. (a) Shaded model. (b) LoG edges. (c) PELs ($0.7fps$). (d) Laplacian lines ($6.0fps$).

and 13d). From the computation point of view, apparent ridges and PELs compute the third-order derivatives on-the-fly, using discrete differential geometry. As the Laplacian of normals can be precomputed, extracting Laplacian lines is much faster than apparent ridges and PELs.

7.4 Comparison to Edge Detection

The effect of Laplacian lines is different from its 2D counterpart. Compared to conventional image edge detection, our Laplacian lines approach is numerically stable and does not suffer from the precision issues, as shown in Fig. 13b. In terms of 3D shape stylization, Laplacian lines can be easily combined with other NPR techniques, such as Toon shading (see Fig. 9), stylization, hatching, diffusion curves, etc.

7.5 Limitations

Laplacian lines generalize the LoG edge detector, and thus, inherit its limitations as well. For instance, Laplacian lines cannot convey the corners effectively. Fig. 14 shows Laplacian lines and other feature lines for a cube model with sharp and round corners/edges, respectively. Laplacian line works for the sharp corners; however, it fails for the round corners. For such a case, ridge-valley line and apparent ridge seem to be the best choice. Furthermore, Laplacian line is not suitable for time-variant or deformable models, where the normals and their Laplacians cannot be precomputed.

8 Conclusions and Future Work

In this paper, we have introduced a new line drawing technique, based on the LoG edge detector. We showed that Laplacian of illumination can be decomposed so that the time-consuming computation of Laplacian of normals can be precomputed. As a result, the runtime Laplacian line extraction is as simple as the conventional silhouette extraction. As Laplacian lines contain much more information than silhouettes, rendering Laplacian lines is slightly slower than that of silhouettes. In addition, by taking advantage of the recently-developed mesh Laplace operator [17], we can produce robust and smooth Laplacian lines on 3D complicated models. Then, we generalized Laplacian lines to volumetric data sets and develop the algorithm to extract volumetric Laplacian lines without isosurface extraction. We also revealed the relationship between Laplacian lines and silhouettes and showed that Laplacian lines coincide with silhouettes, if the mean curvature does not change too much along the viewing direction. Our experimental results demonstrate that Laplacian lines are an effective view-dependent feature and are promising to convey large scale models for interactive graphics applications.

We will pursue several research directions in the future. First, our current implementation can be easily improved by employing a GPU-based acceleration scheme due to the parallel nature of Laplacian line extraction. Second, each kind of computer-generated line has its own advantages and disadvantages. Thus, there is a need to investigate

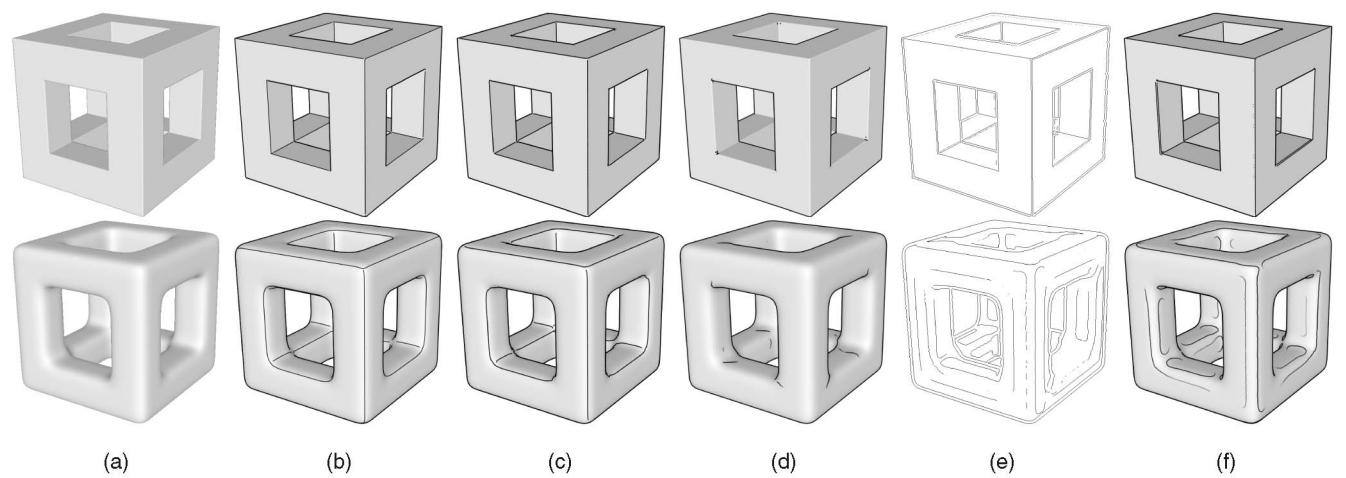


Fig. 14. Laplacian lines work for the sharp corners, however, they do not convey the rounded corners effectively. (a) Shaded image. (b) Ridge-valley lines. (c) Apparent ridges. (d) Suggestive contours. (e) LoG edge detector. (f) Laplacian lines.

effective techniques to combine various feature lines to better convey the shape.

ACKNOWLEDGMENTS

This work was supported by the Singapore National Research Foundation Interactive Digital Media R&D Program, under research Grant NRF2008IDM-IDM004-006. Long Zhang was supported by NSFC 60903085 and an Open Project Program of the State Key Lab of CAD&CG (A0905). Wei Chen was supported by 973 program of China (2010CB732504) and NSFC 60873123. Part of the work has been presented in ACM Symposium on Interactive 3D Graphics and Games (I3D '09) [1]. The authors thank the anonymous reviewers for their careful reviews and constructive comments, and Princeton and Rutgers Computer Graphics Groups for providing the rtsc software. The surface models are courtesy of Stanford University, Aim@Shape, and artist3d.com. The volume models are courtesy of www.volvis.org and Stefan Roettger. Xuexiang Xie was with the School of Computer Engineering, Nanyang Technological University at the time of this work. He is now deceased.

REFERENCES

[1] L. Zhang, Y. He, X. Xie, and W. Chen, "Laplacian Lines for Real-Time Shape Illustration," *Proc. ACM Symp. Interactive 3D Graphics and Games (I3D '09),* pp. 129-136, 2009.

[2] S. Rusinkiewicz, F. Cole, D. DeCarlo, and A. Finkelstein, "Line Drawings from 3D Models," *Proc. ACM SIGGRAPH Course Notes,* 2008.

[3] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh, "How Well Do Line Drawings Depict Shape?" *Proc. ACM SIGGRAPH '09,* pp. 1-9, 2009.

[4] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive Contours for Conveying Shape," *Proc. ACM SIGGRAPH,* pp. 848-855, 2003.

[5] Y. Ohtake, A. Belyaev, and H. Seidel, "Ridge-Valley Lines on Meshes via Implicit Surface Fitting," *Proc. ACM SIGGRAPH,* pp. 609-612, 2004.

[6] T. Judd, F. Durand, and E. Adelson, "Apparent Ridges for Line Drawing," *Proc. ACM SIGGRAPH,* 2007.

[7] D. DeCarlo and S. Rusinkiewicz, "Highlight Lines for Conveying Shape," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 63-70, 2007.

[8] X. Xie, Y. He, F. Tian, H.S. Seah, X. Gu, and H. Qin, "An Effective Illustrative Visualization Framework Based on Photic Extremum Lines (PELs)," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1328-1335, Nov. 2007.

[9] M. Kolomenkin, I. Shimshoni, and A. Tal, "Demarcating Curves for Shape Illustration," *ACM Trans. Graphics,* vol. 27, no. 5, Dec. 2008.

[10] M. Meyer, M. Desbrun, P. Schr, and A. Barr, "Discrete Differential Geometry Operators for Triangulated 2-Manifolds," *J. Visualization and Math.,* 2002.

[11] S. Rusinkiewicz, "Estimating Curvatures and Their Derivatives on Triangle Meshes," *Proc. Second Int'l Symp. 3D Data Processing, Visualization and Transmission (3DPVT '04),* pp. 486-493, 2004.

[12] M. Salisbury, C. Anderson, D. Lischinski, and D.H. Salesin, "Scale-Dependent Reproduction of Pen-and-Ink Illustrations," *Proc. 23rd Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 461-468, 1996.

[13] R. Raskar and M. Cohen, "Image Precision Silhouette Edges," *Proc. Symp. Interactive 3D Graphics,* pp. 135-140, 1999.

[14] S. Breslav, K. Szerszen, L. Markosian, P. Barla, and J. Thollot, "Dynamic 2D Patterns for Shading 3D Scenes," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 20-28, 2007.

[15] Y. Lee, L. Markosian, S. Lee, and J.F. Hughes, "Line Drawings via Abstracted Shading," *ACM Trans. Graphics,* vol. 26, no. 3, p. 18, 2007.

[16] R. Haralick and L. Shapiro, *Computer and Robot Vision.* Addison-Wesley Publishing Company, 1992.

[17] M. Belkin, J. Sun, and Y. Wang, "Discrete Laplace Operator on Meshed Surfaces," *Proc. Symp. Computational Geometry,* 2008.

[18] C. Weatherbrun, *Differential Geometry of Three Dimensions,* vol. 1. Cambridge Univ. Press, 1927.

[19] M. Burns, J. Klawe, S. Rusinkiewicz, A. Finkelstein, and D. DeCarlo, "Line Drawings from Volume Data," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 512-518, 2005.

[20] A. Hertzmann and D. Zorin, "Illustrating Smooth Surfaces," *Proc. 27th Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 517-526, 2000.

[21] B. Gooch, P.J. Sloan, A. Gooch, P. Shirley, and R.F. Riesenfeld, "Interactive Technical Illustration," *Proc. ACM Symp. Interactive 3D Graphics,* pp. 31-38, 1999.

[22] A. Ni, K. Jeong, S. Lee, and L. Markosian, "Multi-Scale Line Drawings from 3D Meshes," *Proc. ACM Symp. Interactive 3D Graphics and Games (I3D),* pp. 133-137, 2006.

[23] M. Kolomenkin, I. Shimshoni, and A. Tal, "On Edge Detection on Surfaces," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '09),* pp. 2767-2774, 2009.

[24] E. Kalogerakis, D. Nowrouzezahrai, P. Simari, J. McCrae, A. Hertzmann, and K. Singh, "Data-Driven Curvature for Real-Time Line Drawing of Dynamic Scene," *ACM Trans. Graphics,* vol. 28, no. 1, pp. 1-24, Jan. 2009.

[25] L. Zhang, Y. He, and H.-S. Seah, "Real-Time Computation of Photic Extremum Lines (PELs)," *The Visual Computer,* vol. 26, no. 6, pp. 399-407, 2010.

[26] T. Saito and T. Takahashi, "Comprehensible Rendering of 3D Shapes," *Proc. ACM SIGGRAPH,* vol. 24, no. 4, pp. 197-206, 1990.

[27] J.W. Buchanan and M.C. Sousa, "The Edge Buffer: A Data Structure for Easy Silhouette Rendering," *Proc. First Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '00),* pp. 39-42, 2000.

[28] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk, "Non-Photorealistic Camera: Depth Edge Detection and Stylized Rendering Using Multi-Flash Imaging," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 679-688, Aug. 2005.

[29] H. Winnemöller, S.C. Olsen, and B. Gooch, "Real-Time Video Abstraction," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 1221-1226, 2006.

[30] R. Vergne, R. Pacanowski, P. Barla, X. Granier, and C. Schlick, "Light Warping for Enhanced Surface Depiction," *ACM Trans. Graphics,* vol. 28, no. 3, 2009.

[31] D. Ebert and P. Rheingans, "Volume Illustration: Non-Photorealistic Rendering of Volume Models," *Proc. IEEE Conf. Visualization,* pp. 195-202, 2000.

[32] N.A. Svakhine and D.S. Ebert, "Interactive Volume Illustration and Feature Halos," *Proc. Pacific Conf. Computer Graphics and Applications,* pp. 347-354, 2003.

[33] A. Lu, C.J. Morris, D.S. Ebert, P. Rheingans, and C.D. Hansen, "Non-Photorealistic Volume Rendering Using Stippling Techniques," *Proc. IEEE Conf. Visualization,* pp. 211-218, 2002.

[34] Z. Nagy, J. Schneider, and R. Westermann, "Interactive Volume Illustration," *Proc. Vision, Modeling, and Visualization Workshop (VMV '09),* pp. 497-504, 2002.

[35] S. Bruckner and M.E. Gröller, "Volumeshop: An Interactive System for Direct Volume Illustration," *Proc. IEEE Conf. Visualization,* pp. 671-678, 2005.

[36] Z. Nagy and R. Klein, "High-Quality Silhouette Illustration for Texture-Based Volume Rendering," *Proc. Workshop Computer Graphics (WSCG '04),* pp. 301-308, 2004.

[37] S. Schein and G. Elber, "Adaptive Extraction and Visualization of Silhouette Curves from Volumetric Datasets," *The Visual Computer: Int'l J. Computer Graphics,* vol. 20, no. 4, pp. 243-252, 2004.

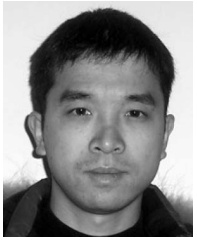
[38] F. Dong, G.J. Clapworthy, H. Lin, and M.A. Krokos, "Nonphotorealistic Rendering of Medical Volume Data," *IEEE Computer Graphics and Applications,* vol. 23, no. 4, pp. 44-52, July 2003.

[39] W. Chen, A. Lu, and D.S. Ebert, "Shape-Aware Volume Illustration," *J. Computer Graphics Forum,* vol. 26, no. 3, pp. 705-714, 2007.

[40] N.A. Svakhine, D.S. Ebert, and W.M. Andrews, "Illustration-Inspired Depth Enhanced Volumetric Medical Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 15, no. 1, pp. 77-86, Jan./Feb. 2009.

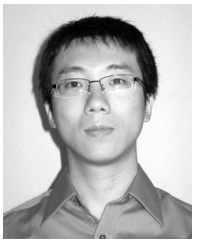
[41] F. Cole, A. Golovinskiy, A. Limpaecher, H.S. Barros, A. Finkelstein, T. Funkhouser, and S. Rusinkiewicz, "Where Do People Draw Lines?," *ACM Trans. Graphics,* vol. 27, no. 3, 2008.

[42] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, pp. 679-698, Nov. 1986.
[43] D. Marr and E. Hildreth, "Theory of Edge Detection," *Proc. Royal Soc. London,* vol. 207, pp. 187-217, 1980.
[44] G. Xu, "Discrete Laplace-Beltrami Operators and Their Convergence," *Computer Aided Geometric Design,* vol. 21, no. 6, pp. 767-784, Oct. 2004.
[45] A.G. Belyaev, A.A. Pasko, and T.L. Kunii, "Ridges and Ravines on Implicit Surfaces," *Proc. Int'l Conf. Computer Graphics,* pp. 530-535, 1998.
[46] R. Goldman, "Curvature Formulas for Implicit Curves and Surfaces," *Computer Aided Geometric Design,* vol. 22, no. 7, pp. 632-658, 2005.
[47] T. Vieville and O.D. Faugeras, "Robust and Fast Computation of Unbiased Intensity Derivatives in Images," *Proc. Second European Conf. Computer Vision (ECCV '92),* pp. 203-211, 1992.
[48] J.-P. Thirion and A. Gourdon, "The 3D Marching Lines Algorithm," *Graphical Models and Image Processing,* vol. 58, no. 6, pp. 503-509, 1996.



Long Zhang received the PhD degree in 2008 from the Department of Mathematics, Zhejiang University. He is an assistant professor in the School of Computer Science and Technology, Hangzhou Dianzi University. His current research interests include NPR, expressive rendering, and visualization.

Ying He received the BS and MS degrees in electrical engineering from Tsinghua University, China, and the PhD degree in computer science from the State University of New York (SUNY), Stony Brook. He is currently an assistant professor at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include computer graphics, computer-aided design, and scientific visualization. His major works include manifold splines, polycube parameterization, volumetric mapping, and computer-generated line drawings. More information about his research can be found at http://www.ntu.edu.sg/home/yhe.



Jiazhi Xia received the BS and MS degrees from Zhejiang University, in 2005 and 2008, respectively. He is currently working toward the PhD degree at the School of Computer Engineering, Nanyang Technological University, under the supervision of Dr. Ying He. His research interests include computer graphics and human-computer interaction.

Xuexiang Xie received the BS degree in computer science from Zhejiang University in 1996 and the PhD degree in computer engineering from Nanyang Technological University in 2009. His research interests include computer graphics, scientific visualization, and computer vision.

Wei Chen received the PhD degree in July 2002 from the Fraunhofer Institute for Graphics, Darmstadt, Germany, as a joint PhD student, under the supervision of Prof. Qunsheng Peng and Prof. Georgios Sakas. From July 2006 to September 2008, he was a visiting scholar at Purdue University, working in PURPL with Prof. David S. Ebert. Since December 2009, he is working as a full professor in the State Key Laboratory of Computer-Aided Design and Computer Graphics at Zhejiang University, P.R. China. He has performed research in computer graphics and visualization, and has published more than 60 peer-reviewed journal and conference papers in the last five years. His current research interests include scientific visualization, visual analytics, and biomedical image computing.