Laplacian Lines for Real-Time Shape Illustration

Long Zhang
Institute of Graphics and Image,
Hangzhou Dianzi University,
lzhang@cad.zju.edu.cn

Ying He Xuexiang Xie
School of Computer Engineering
Nanyang Technological University,
{yhe|xuexiang}@ntu.edu.sg

Wei Chen
State Key Lab of CAD&CG,
Zhejiang University,
chenwei@cad.zju.edu.cn

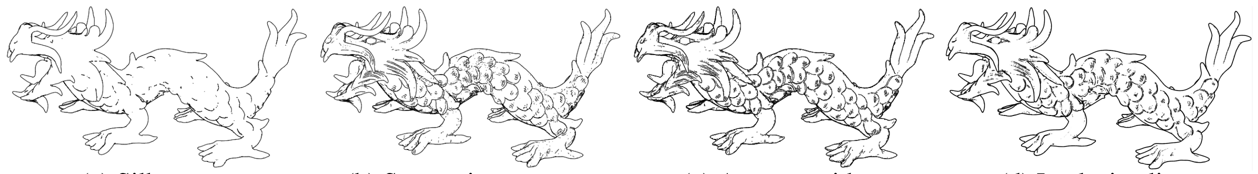


Figure 1: *Laplacian lines (LL) are a set of object-space, view dependent feature lines which generalize Laplacian-of-Gaussian (LoG) edge detector to 3D surfaces. Laplacian line is efficient in that most expensive computations can be pre-computed and the run-time complexity for LL extraction algorithm is similar to the silhouette extraction. Our un-optimized implementation can generate Laplacian lines at the speed of* $16.9fps$ *for the Stanford Dragon model with 1 million triangles. Testing platform: a PC with 3.0GHz CPU and 2.0G memory.*

Abstract

This paper presents a novel object-space line drawing algorithm that can depict shape 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 features, Laplacian lines are computationally efficient because most expensive computations can be pre-processed. Thus, Laplacian lines are very promising for interactively illustrating large-scale models.

Keywords: Laplacian lines, view dependent features, object-space line extraction, real-time line drawing, non-photorealistic rendering

1 Introduction

Line drawing is a widely used shape depiction technique due to its capability to express meaningful information in a relatively succinct manner by ignoring less important details [Rusinkiewicz et al. 2008]. For complicated models or large-scale scenes, line drawing can be used to eliminate unnecessary visual clutter and depict essential information. In the past decade, there has been a large amount of work on computer-generated line drawings, including suggestive contours [DeCarlo et al. 2003], ridge-valley lines [Ohtake et al. 2004], apparent ridges [Judd et al. 2007], principal and suggestive highlights [DeCarlo and Rusinkiewicz 2007], photic extremum lines (PELs) [Xie et al. 2007], demarcating curves [Kolomenkin et al. 2008], 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 using discrete differential geometry [Meyer et al. 2002; Rusinkiewicz 2004] which is known to be sensitive to noise and mesh tessellation.

Besides the object-space feature lines, image-space algorithms also draw much attention. For instance, Salisbury *et al.* [1996] proposed to represent pen-and-ink drawings with a grayscale image, a set of discontinuity segments, and associated stroke textures for the purposes of producing 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 [Raskar and Cohen 1999]. It also allows for an efficient way to map screen-space patterns onto 3D models [Breslav et al. 2007], and convey both the shape and material [Lee et al. 2007]. In addition, image-space line drawing does not require costly computation of surface information, and can be applied to animated models in real-time. However, image-space algorithms often suffer from pixel-level artifacts and are inefficient for shape stylization.

This paper presents an efficient line drawing algorithm for interactive graphics applications. Our method is inspired by an analogy with image edge detection. Loosely speaking we seek 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 a decent solution for image enhancement [Haralick and Shapiro 1992]. 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 pre-processing 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 which need to compute the derivatives on-the-fly, all the third order derivatives of Laplacian lines can be completely pre-computed. 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 pre-computed. As a result, the run-time Laplacian line extraction algorithm is just as simple as the conventional silhouette extraction algorithm, i.e., simply replacing the surface normal by Laplacian of normal. In addition, we choose the recently developed mesh Laplace operator [Belkin et al. 2008] and show that the extracted lines are much more robust than the conventional cotangent Laplace operator. These characteristics make Laplacian lines quite suitable for illustrating large-scale models in interactive graphics applications. Fig. 1 compares Laplacian lines with several famous object-space line drawing algorithms. It shows that our approach can generate nice results with comparable quality to previous approaches. Our current implementation are faster than the existing techniques.

The 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 pre-computed.
- By employing the recently developed mesh Laplace operator [Belkin et al. 2008], our Laplacian line extraction algorithm is robust to noise and irregular mesh tessellation.
- We reveal the relation between Laplacian lines and silhouettes and show that Laplacian lines are very close to silhouettes when the mean curvature does not change too much along the viewing direction.

2 Previous Work

Extensive research has been done in computer-generated line drawings [Rusinkiewicz et al. 2008]. Roughly speaking current solutions can be classified into two categories: object-space and image-space approaches.

Object-space algorithms compute the surface derivatives to find the positions of expressive lines. Silhouettes show the strongest cues with model-to-background distinction [Hertzmann and Zorin 2000] [Gooch et al. 1999]. However, silhouettes alone are quite limited in conveying shape, since they can not capture the structure and complexity of the shape interior. DeCarlo *et al.* [2003] proposed suggestive contours which 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 [2007] introduced highlight lines which complement contours and suggestive contours. Burns *et al.* [2005] proposed a volumetric line drawing system that directly extracts silhouettes and suggestive contours using a temporally coherent seed-and-traverse framework. Ridge-valley lines are powerful shape descriptors but independent of the view point [Ohtake et al. 2004]. Judd *et al.* [2007] proposed apparent ridges which elegantly generalize ridge-valley lines with view dependent features. Xie *et al.* [2007] proposed photic extremum lines (PEL) which generalize Canny's edge detector to 3D surfaces. Similar to apparent ridge, PEL is a third-order feature, and thus computational expensive. Aiming at line drawing simplification, Ni *et al.* [2006] proposed view-dependently controllable feature lines using pre-constructed multi-resolution mesh. Recently, Kalogerakis *et al.* [2008] proposed a real-time and highly parallelizable method to compute curvatures and their derivatives for deforming objects. This method is ideal for deforming objects with temporal coherence by accurately predict the curvatures and their derivatives. However, for static objects, this property does not hold and the computation for surface curvatures is still expensive. Kolomenkin *et al.* [2008] proposed demarcating curves which are the loci of the strongest inflections on the surface and are capable of artifact illustration in archaeology.

Image-space approaches are usually easy to implement since the complicated surface derivatives computations can be avoided. Saito and Takahashi [1990] pioneered a method to extract feature lines using image processing techniques. Raskar *et al.* [2005] presented a novel NPR camera which detects depth edges using multi-flash images. Winnemöller *et al.* [2006] presented an automatic, real-time video and image abstraction framework using difference-of-Gaussian edges. Lee *et al.* [2007] proposed to automatically extract lines at appropriate scales from abstract shading.

3 Laplacian Line Extraction

3.1 Definition

The key question for computer-generated line algorithm is: where do you put the lines? According to the recent study on how artists made line drawings intended to convey 3D shapes, the image-space feature detector (which characterizes the significant changes in illumination) provides the strongest cues for artists to draw lines [Cole et al. 2008]. This observation leads a research direction to generalize the edge detection from 2D images to 3D surfaces.

Xie *et al.* [2007] presented a method to generalize Canny edge detector [Canny 1986] to 3D surface and proposed photic extremum lines (PELs) to characterize significant changes in the illumination. Given a 3D surface S, Xie *et al.* considered the illumination function $I : S \to \mathbb{R}$ defined on S and 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 [Xie et al. 2007], 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 $\mathbf{d}$ is the gradient of I.

Although PELs can produce nice line drawings for surfaces and volumetric models, 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. To develop an efficient and robust "edge" detector on 3D surfaces, this paper presents Laplacian lines, which generalizes the Laplacian of Gaussian (LoG) edge detector [Marr and Hildreth 1980] from 2D images to 3D surfaces. The Laplacian lines are defined 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 t, i.e.,

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

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

Let $\mathbf{e}$ denote the view point and $\mathbf{n}(\mathbf{p})$ the normal of a point $\mathbf{p} \in S$. Here 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 $\mathbf{e}$. Then, for any point $\mathbf{p} \in S$, the light vector $\mathbf{l}$ is identical to the viewing vector $\mathbf{l}(\mathbf{p}) = \mathbf{e} - \mathbf{p}$.
- The surface exhibits only diffuse reflection, namely, the illumination $I(\mathbf{p}) = \mathbf{n}(\mathbf{p}) \cdot \mathbf{l}(\mathbf{p})$.

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. Then, Laplacian of illumination $\triangle I$ can be simplified as follows:

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

where H is the mean curvature. Note that $\triangle \mathbf{n}$ is view-independent and can be pre-computed. As a result, Laplacian line extraction is much more efficient than that of PELs by avoiding on-the-fly time-consuming derivative computation.

3.2 Robust Laplace operator

The key component in Laplacian line extraction is to compute the Laplacian of surface normal $\triangle \mathbf{n}$. The popular cotangent Laplace operator depends on the mesh tessellation and tends to be sensitive to noise [Meyer et al. 2002].

To develop a robust Laplacian line extraction algorithm, we use the mesh Laplace operator proposed in [Belkin et al. 2008]: Given a function $f : M \to \mathbb{R}$ defined on the mesh M, the mesh Laplace operator L_M^h is defined as follows:

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

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 $\mathbf{p}$. In [Belkin et al. 2008], it has been shown that when mesh M is a sufficient approximation of a smooth underlying surface S, L_M^h is close to the Laplace-Beltrami operator.

The Gaussian kernel h in Eqn. (2) 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 and fewer the number of Laplacian lines we obtain. Fig. 3 shows the effect of different values of h.

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 much smoother Laplacian lines than the cotangent Laplacian. Furthermore, these computations are in the pre-processing step and our run-time line extraction algorithm remains simple yet efficient.

3.3 Laplacian line extraction algorithm

Given a triangle mesh M, our Laplacian line extraction algorithm takes two parameters, σ and τ, and consists of four consecutive steps:

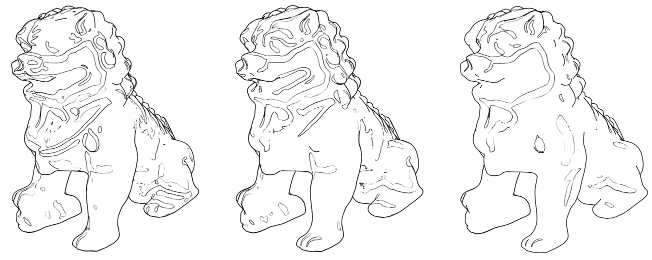

(a) $h = 0.01$ (b) $h = 0.03$ (c) $h = 0.05$

Figure 3: *The effects of the Gaussian kernel h. The larger the kernel size, the more smoothing effects obtained, thus, the less Laplacian lines extracted.*

1. (Pre-processing) For each vertex, smooth the normal $\mathbf{n}$ and then compute $\triangle \mathbf{n}$ using Eqn. 2.
2. For each point $\mathbf{p}$, compute the dot product of view 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 Gaussian or bilateral filter. In our experiments, we found that the illumination $I = \mathbf{n} \cdot \mathbf{l}$ is usually smooth for the clean data with regular triangulation. Note that the mesh Laplace operator also has smoothing effects, thus, smoothing normals for clean data is usually not necessary. However, for noisy meshes or meshes with 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 Eqn. 2 for each vertex. The parameter h is specified by the user.

Step 2. In the run-time step, we first compute the dot product of the view vector $\mathbf{v}$ and $\triangle \mathbf{n}$ for each vertex. To locate the zero-crossings, we follow the method in [Ohtake et al. 2004]. 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 exactly 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 clean meshes with good triangulation, we use the popular discrete differential geometry approach [Rusinkiewicz 2004; Meyer et al. 2002]. For the mesh with noise and/or irregular tessellation, we solve the following optimization problem:

$$E(\nabla f(\mathbf{p})) = \sum_{\mathbf{q} \in M \bigcap Ball(\mathbf{p}, h(\mathbf{p}))} |f(\mathbf{q}) - f(\mathbf{p}) - \nabla f(\mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})|^2, \tag{3}$$

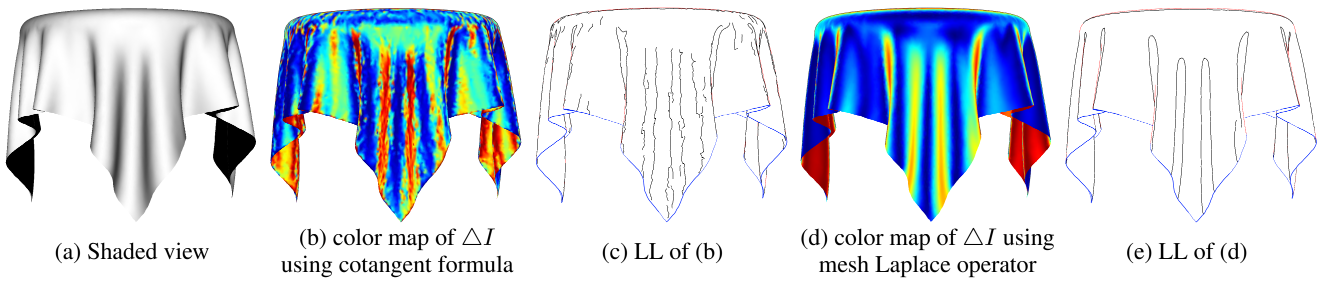


Figure 2: *Mesh Laplace operator vs cotangent Laplace operator. The popular cotangent Laplacian operator is sensitive to the mesh triangulation, and thus does not generate smooth scalar field $\triangle I$ and the Laplacian lines (see (b) and (c)). Mesh Laplace operator, on the contrary, is robust to irregular tessellation and generate smooth Laplacian of illumination and Laplacian lines (see (d) and (e)).*

where $Ball(\mathbf{p}, h(\mathbf{p}))$ is a solid sphere centered at $\mathbf{p}$ with radius $h(\mathbf{p})$. Note that the computation of gradient magnitude is only applied to the zero-crossing points.

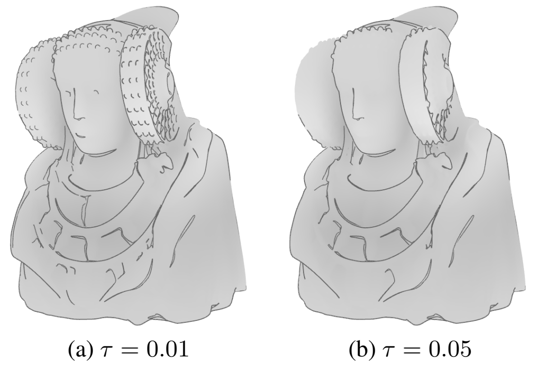


Figure 4: *Trimming the Laplacian lines with the user-specified threshold τ. Increasing the threshold results in less number of extracted Laplacian lines.*

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

$$\int \|\nabla I\| ds \approx \sum_i \frac{\|\nabla I(\mathbf{p}_i)\| + \|\nabla I(\mathbf{p}_{i+1})\|}{2} \|\mathbf{p}_i - \mathbf{p}_{i+1}\|. \quad (4)$$

Finally, we delete the feature lines whose strengths are less than the user-specified threshold τ as shown in Fig. 4.

4 Experimental Results

4.1 Performance

We have implemented a prototype system on a PC with 3.0GHz CPU and 2GB memory. Standard OpenGL library is used as the rendering engine. Our un-optimized implementation can generate Laplacian lines in interactive frame rates for all test models. Besides Laplacian lines, we measured the performance of silhouettes, suggestive contours, apparent ridges and PELs with the same hardware configuration. As shown in Tab. 1, our approach is slightly slower than silhouette, faster than suggestive contours, at least five times faster than apparent ridges and PELs.

Table 1: *Performance measurement. The statistics for silhouettes, suggestive contours and apparent ridges are obtained using the rtsc software from Princeton University. #$\triangle$: number of triangles; f_S: frame per second for silhouettes; f_{SC}: FPS for suggestive contours; f_{AR}: FPS for apparent ridges; f_{PEL}: FPS for PEL; f_{LL}: FPS for Laplacian lines.*

Models	#$\triangle$	f_S	f_{SC}	f_{AR}	f_{PEL}	f_{LL}
Armadillo	$330K$	64.1	38.6	8.5	6.3	50.1
Cathedral	$1180K$	22.2	12.1	2.5	1.4	13.6
Column	$200K$	94.3	71.5	12.5	8.6	80.1
Rosace	$250K$	76.4	51.2	9.3	7.9	67.4
Statute head	$300K$	85.5	55.3	9.8	7.6	56.3
Thai statute	$2000K$	13.1	6.9	1.6	1.0	7.4

4.2 Comparisons and Discussions

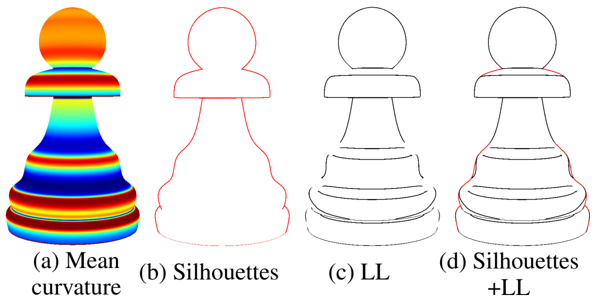


Figure 5: *Laplacian lines are very close to silhouettes if the mean curvature is constant or near constant along the viewing direction, i.e., $D_{\mathbf{w}}H = 0$, where $\mathbf{w}$ is the projection of viewing vector to the tangent plane.*

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. Given a parametric surface $\mathbf{s} : \Omega \subseteq \mathbb{R}^2 \rightarrow S \subseteq \mathbb{R}^3$, the Laplacian of normal is given by [Weatherbrun 1927]:

$$\triangle \mathbf{n} = (2K - H^2)\mathbf{n} - 2\nabla H,$$

where H and K are mean and Gaussian curvatures respectively. 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 $\mathbf{p}$. Laplacian lines satisfy

$$\begin{aligned}(\triangle\mathbf{n})\cdot\mathbf{l} &= (2K-H^2)\mathbf{n}\cdot\mathbf{l}-2(\nabla H)\cdot\mathbf{l}\\ &= (2K-H^2)\mathbf{n}\cdot\mathbf{v}-2(\nabla H)\cdot\mathbf{w}\\ &= (2K-H^2)\mathbf{n}\cdot\mathbf{v}-2D_{\mathbf{w}}H=0. \qquad (5)\end{aligned}$$

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. 5). From the computation 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 only take small amount of time. Therefore, Laplacian lines have similar performance as silhouettes.

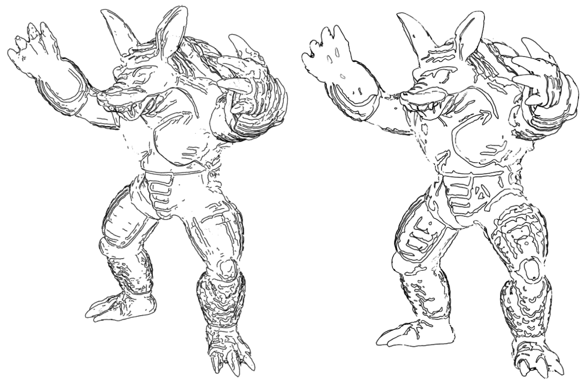


Figure 7: *LLs can lead to similar results of PELs, since both are generalized from 2D edge detectors. No silhouettes are drawn in order to provide better views of PELs and LLs.*

Comparison to suggestive contours Suggestive contours are second-order features which naturally extend the contours in concave regions and convey the shape in a succinct and elegant way. Note that suggestive contours algorithm can also pre-compute the curvature, so the performance of suggestive contours is similar to Laplacian lines. However, suggestive contours must be used together with silhouettes since they can not illustrate salient features in convex regions. Laplacian lines works well on both convex and concave regions. In many cases, Laplacian lines without silhouettes can produce satisfactory rendering results.

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 completely determined by the geometry and independent of the view point, thus, they do not make a natural looking line drawing in an animation sequence. By introducing view dependent curvatures, apparent ridges elegantly extends ridge-valley lines, yielding more perceptually pertinent results. PELs generalize the Canny edge detector to 3D surface and characterizes the significant changes of illumination. Note that the proposed Laplacian line is also an extension of image edge detection. As expected, Laplacian lines can generate similar results as PELs (see Fig. 7). From the computation point of view, apparent ridges and PELs compute the third-order derivatives on-the-fly using discrete differential geometry. Laplacian lines can pre-compute the Laplacian of normals and the line extraction is much faster than apparent ridges and PELs.

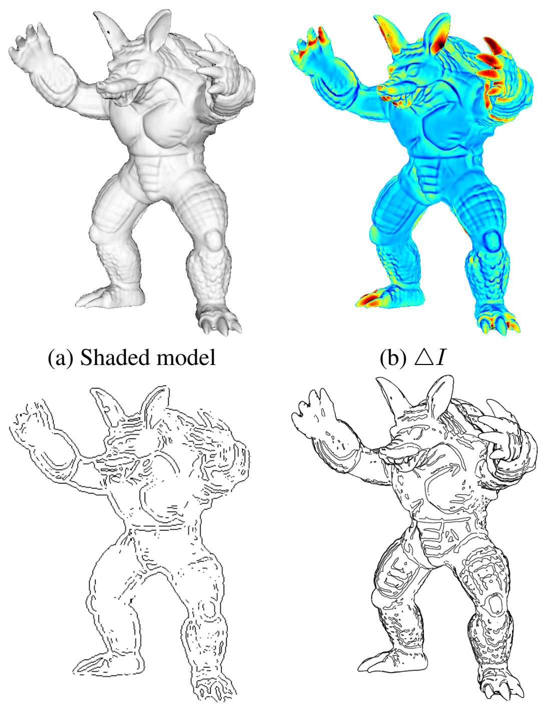


Figure 10: *Comparison of Laplacian lines to LoG edge detector. Applying LoG edge detector directly to the rendered image does not generate satisfactory results. Note the short lines in (c). Clearly, the object space algorithm is much more robust.*

Comparison to edge detection. Although originated from edge detection, Laplacian lines are different from its 2D counterpart. Compared to edge detection, Laplacian line is numerically stable and do not suffer from the precision issues as shown in Fig. 10. Furthermore, Laplacian lines can be easily combined with other NPR techniques, such as Toon shading (see Fig. 9), stylization, hatching, diffusion curves.

Limitations Laplacian lines generalize the LoG edge detector, and inherit its limitations. For instance, Laplacian lines can not convey the sharp corners effectively. Fig. 11 shows a failure case of Laplacian lines for a simple cube model with rounded edges and corners. As indicated by [DeCarlo et al. 2003], the ridge and valley lines are most effective for such cases.

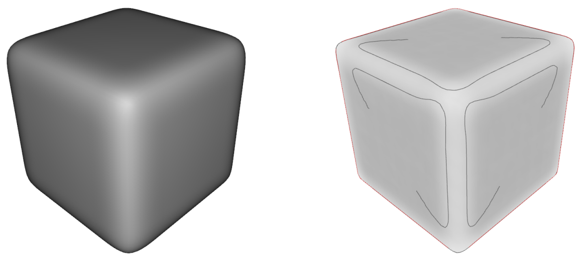
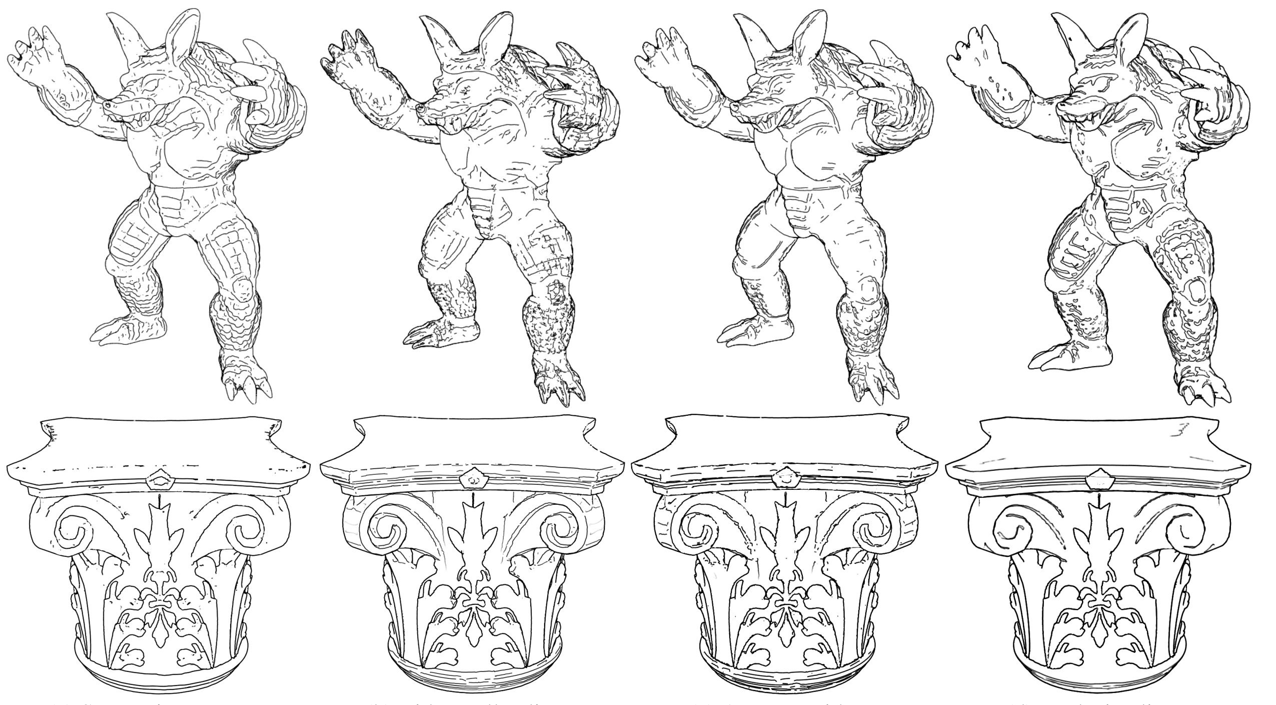


Figure 11: *Similar to LoG edge detector, Laplacian lines do not convey the corners effectively.*



(a) Suggestive contour (b) Ridge-valley lines (c) Apparent ridges (d) Laplacian lines

Figure 6: *Comparison of Laplacian lines with ridge-valley lines, suggestive contours and apparent ridges.*

5 Conclusions and Future Work

In this paper, we have introduced line drawing technique based on the LoG edgedetector. We showed that Laplacian of illumination can be decomposed so that the time consuming computation of Laplacian of normals can be completely pre-computed. As a result, the run-time Laplacian line extraction is as simple and efficient as the conventional silhouette extraction. In addition, by taking advantage of the recently-developed mesh Laplace operator [Belkin et al. 2008], we can produce robust and smooth Laplacian lines on complicated models. We also revealed the relation 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 promising to convey large scale models for interactive graphics applications.

Acknowledgement

Ying He is supported by NTU-SUG 19/06, AcRF 69/07 and NRF2008IDM-IDM-004-006 grant. Long Zhang is supported by an Open Project Program of the State Key Lab of CAD&CG (Grant No. A0905). Wei Chen is supported by NSFC 60873123. The authors would like to thank Princeton and Rutgers Computer Graphics Groups for providing the rtsc software. The models are courtesy of Stanford University and artist3d.com.

References

BELKIN, M., SUN, J., AND WANG, Y. 2008. Discrete laplace operator on meshed surfaces. In *Proceedings of Symposium on Computational Geometry*.

BRESLAV, S., SZERSZEN, K., MARKOSIAN, L., BARLA, P., AND THOLLOT, J. 2007. Dynamic 2D patterns for shading 3D scenes. *ACM Transactions on Graphics 26*, 3, 20–28.

BURNS, M., KLAWE, J., RUSINKIEWICZ, S., FINKELSTEIN, A., AND DECARLO, D. 2005. Line drawings from volume data. In *Proceedings of ACM SIGGRAPH*, 512–518.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 679–698.

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines? *ACM Transactions on Graphics 27*, 3 (Aug.).

DECARLO, D., AND RUSINKIEWICZ, S. 2007. Highlight lines for conveying shape. In *Proceedings of NPAR*, 63–70.

DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. In *Proceedings of ACM SIGGRAPH*, 848–855.

GOOCH, B., SLOAN, P. J., GOOCH, A., SHIRLEY, P., AND RIESENFELD, R. F. 1999. Interactive technical illustration. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, 31–38.

HARALICK, R., AND SHAPIRO, L. 1992. *Computer and Robot Vision*. Addison-Wesley Publishing Company.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH*, 517–526.

JUDD, T., DURAND, F., AND ADELSON, E. 2007. Apparent ridges for line drawing. In *Proceedings of ACM SIGGRAPH*.

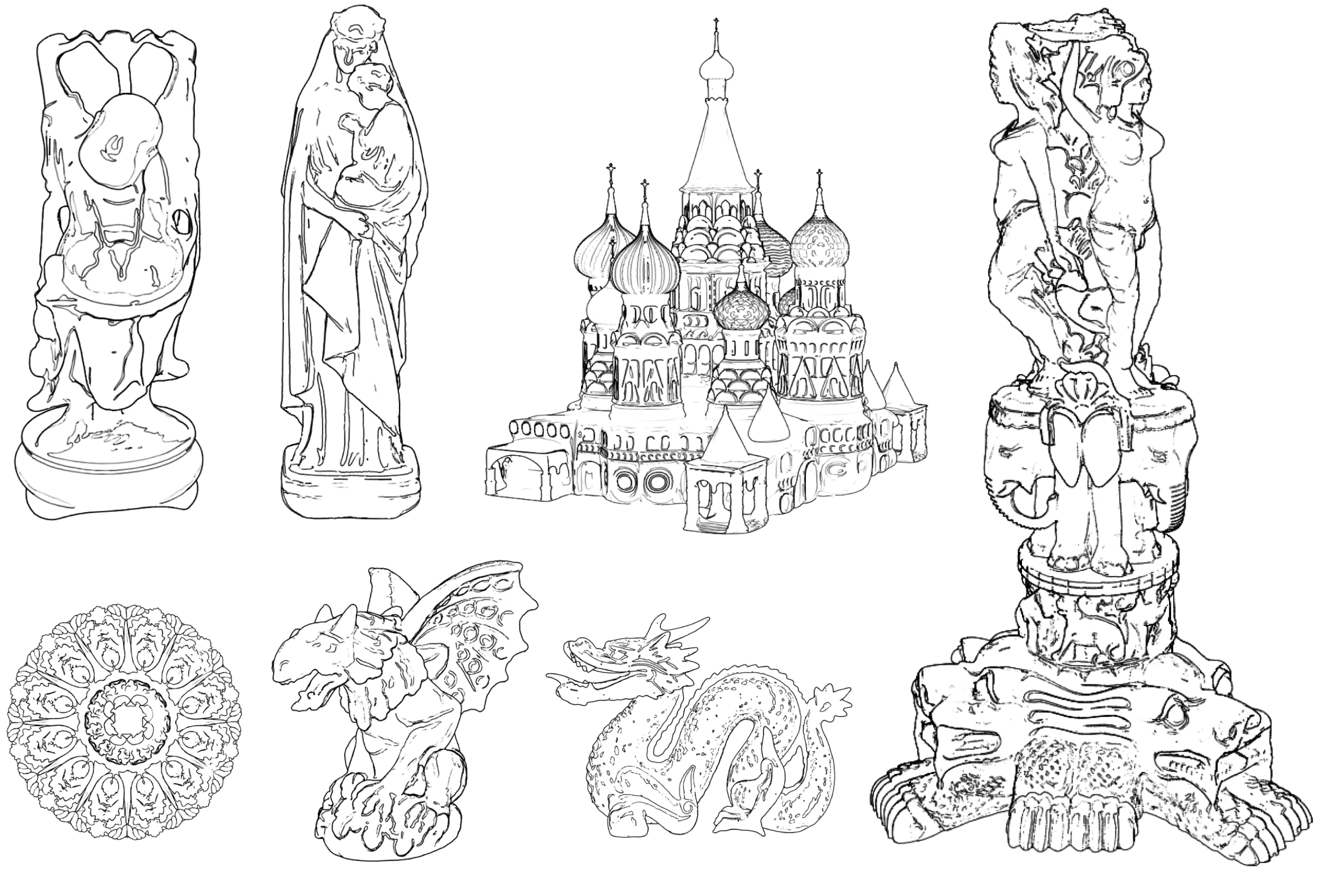


Figure 8: *Conveying shapes using Laplacian lines. Silhouettes are not drawn in the image.*

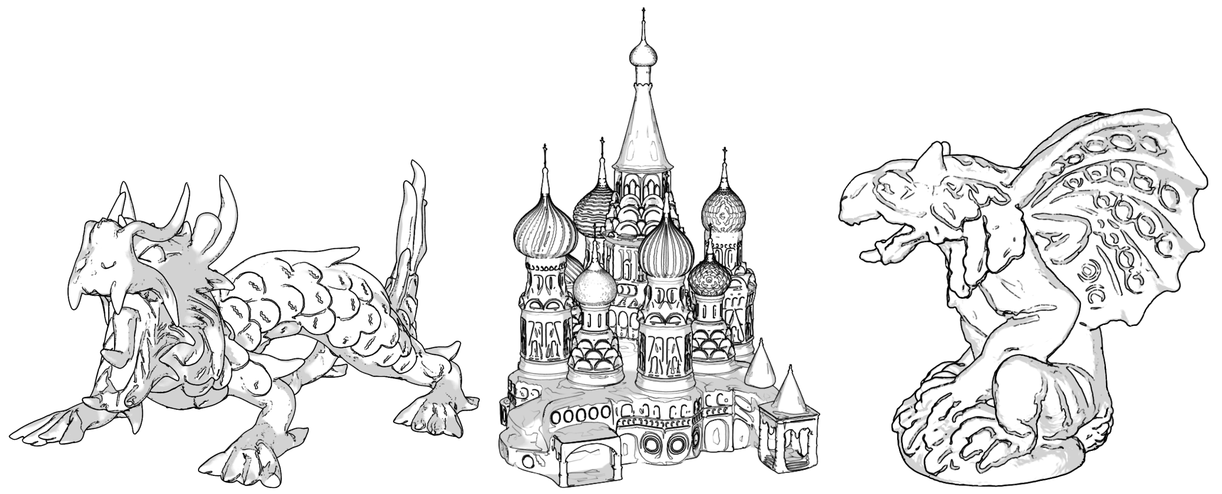


Figure 9: *Conveying shapes using Laplacian lines and Toon shading.*

KALOGERAKIS, E., NOWROUZEZAHRAI, D., SIMARI, P., MCCRAE, J., HERTZMANN, A., AND SINGH, K. 2008. Real-time line drawing for animated surfaces. *ACM Transactions on Graphics 27*, 3 (Aug.).

KOLOMENKIN, M., SHIMSHONI, I., AND TAL, A. 2008. Demarcating curves for shape illustration. *ACM Transactions on Graphics 27*, 5 (December).

LEE, Y., MARKOSIAN, L., LEE, S., AND HUGHES, J. F. 2007. Line drawings via abstracted shading. *ACM Transactions on Graphics 26*, 3, 18.

MARR, D., AND HILDRETH, E. 1980. Theory of edge detection. In *Proceedings of Royal Society, London*, vol. 207, 187–217.

MEYER, M., DESBRUN, M., SCHR, P., AND BARR, A. 2002. Discrete differential geometry operators for triangulated 2-manifolds. *Journal of Visualization and Mathematics*.

NI, A., JEONG, K., LEE, S., AND MARKOSIAN, L. 2006. Multi-scale line drawings from 3D meshes. In *Proceedings of ACM Symposium on I3D*, 133–137.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H. 2004. Ridge-valley lines on meshes via implicit surface fitting. In *Proceedings of ACM SIGGRAPH*, 609–612.

RASKAR, R., AND COHEN, M. 1999. Image precision silhouette edges. In *Proceedings of ACM SIGGRAPH*, 135–140.

RASKAR, R., TAN, K.-H., FERIS, R., YU, J., AND TURK, M. 2005. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. In *Courses, Proceedings of ACM SIGGRAPH*, 2.

RUSINKIEWICZ, S., COLE, F., DECARLO, D., AND FINKELSTEIN, A. 2008. Class notes: Line drawings from 3D models. In *Proceedings of ACM SIGGRAPH*.

RUSINKIEWICZ, S. 2004. Estimating curvatures and their derivatives on triangle meshes. In *Proceedings of 3D PVT*, 486–493.

SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3D shapes. *Proceedings of ACM SIGGRAPH 24*, 4, 197–206.

SALISBURY, M., ANDERSON, C., LISCHINSKI, D., AND SALESIN, D. H. 1996. Scale-dependent reproduction of pen-and-ink illustrations. In *Proceedings of ACM SIGGRAPH*, 461–468.

WEATHERBRUN, C. 1927. *Differential geometry of three dimensions*, vol. 1. Cambridge University Press.

WINNEMÖLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Realtime video abstraction. *ACM Transactions on Graphics 25*, 3, 1221–1226.

XIE, X., HE, Y., TIAN, F., SEAH, H. S., GU, X., AND QIN, H. 2007. An effective illustrative visualization framework based on photic extremum lines (PELs). *IEEE Transactions on Visualization and Computer Graphics 13*, 6, 1328–1335.