Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

Streamlines for Illustrative Real-Time Rendering

K. Lawonn, T. Moench & B. Preim

Department of Simulation and Graphics, University of Magdeburg, Germany

Abstract

Line drawing techniques are important methods to illustrate shapes. Existing feature line methods, e.g., suggestive contours, apparent ridges, or photic extremum lines, solely determine salient regions and illustrate them with separate lines. Hatching methods convey the shape by drawing a wealth of lines on the whole surface. Both approaches are often not sufficient for a faithful visualization of organic surface models, e.g., in biology or medicine. In this paper, we present a novel object-space line drawing algorithm that conveys the shape of such surface models in real-time. Our approach employs ***con****tour- and* ***f****eature-based* ***i****llustrative* ***s****treamlines to convey surface shape (ConFIS). For every triangle, precise streamlines are calculated on the surface with a given curvature vector field. Salient regions are detected by determining maxima and minima of a scalar field. Compared with existing feature lines and hatching methods, ConFIS uses the advantages of both categories in an effective and flexible manner. We demonstrate this with different anatomical and artificial surface models. In addition, we conducted a qualitative evaluation of our technique to compare our results with exemplary feature line and hatching methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

The aim of an illustrative visualization method is to provide a simplified representation of a complex scene or object. Concave and convex regions are emphasized and the surface complexity is reduced by omitting unnecessary information. This abstraction is often preferred over fully illuminated scenes in a multitude of applications. Most anatomy atlases use non-photorealistic (NPR) techniques to illustrate anatomical structures. Repair manuals and pictograms employ NPR techniques as well. Common illustration techniques are feature lines and hatching. Feature lines attempt to depict only relevant surface features with separate lines. They are often used in scientific visualization [CSD*09, INC*06]. In contrast, hatching techniques convey a spatial impression on the surface. Illustrative visualization may also be applied to volume rendering [Vio05]. In this paper, however, we concentrate on surface meshes only.

The contribution of this paper is to present ConFIS, which stands for the keywords: **Con**tours, **F**eatures, **I**llustration, **S**treamlines (see Fig. 1). ConFIS combines the advantages of feature lines and hatching methods. Our methods are motivated by the peculiarities in medicine, but our concept is not restricted to that domain. We show that common feature line techniques can not convey the specific shape of several patient-specific anatomical surfaces, e.g., endoscopic views. On the one hand, hatching techniques allow for a better spatial perception in such endoscopic views. On the other hand, they usually draw hatching patterns allover the surface and miss to depict salient regions well. ConFIS is designed to remedy these issues. We evaluated our approach qualitatively with two physicians and three medical researchers. The goal of the evaluation was to assess, with which illustrative visualization technique the domain experts can better infer the surface shape. They had to compare the ConFIS-based illustrations to surface shading and the most commonly used feature line and hatching methods. We make the following contributions:

- A novel view-dependent illustrative visualization method.
- Explicit streamlines on the triangular surface mesh are employed and drawn in real-time.
- Feature regions are determined by resolving the minima and maxima of the mean curvature scalar field.
- Well illustrations on endoscopic anatomical surfaces.
- A qualitative evaluation with medical experts has been conducted to compare ConFIS with existing illustration techniques.

2. Related Work

For feature lines, image and object space methods have to be considered.

Image space methods use the image as an input. All calculations are performed on the rendered image where every pixel contains an RGB- or grey value. The image is usually convolved with different kernels, e.g., Roberts and Sobel edge detection. A comprehensive overview is given by Muthukrishnan et al. [MR11], Nadernejad et al. [NSH08], and Senthilkumaran et al. [SR09]. The resulting feature lines are not frame-coherent and are represented by pixels in the image space. Thus, there is only limited control over the resulting line attributes, e.g., rendering style.

Object-based methods use the connectivity and the 3D vertex position of the surface model as input. Additional information (camera, light position, curvature) may be used to detect features. The extracted lines are located on the surface and are represented as explicit 3D lines. Thus, they are usually frame-coherent and any stylistic rendering technique can be applied. For an extensive list of line drawings and their applications we refer to Rusinkiewicz et al. [RCDF08]. The most important object-space line are contours, which depict the strongest shape cues of the model. Unfortunately, contours can not express all features being relevant for shape perception. Interrante et al. [IFP95] proposed *ridges and valleys* as additional lines to convey the features. *Ridges and valleys* are defined as the loci of points at which the principle curvature reaches an extremum in the principle direction. DeCarlo et al. [DFRS03] introduced *suggestive contours*, which convey shape by extending the contours of the surface mesh. These lines are defined as the set of minima of the dot product between the surface normal and the tangential view vector. Unfortunately, objects without concave regions, e.g., most human organs have no suggestive contours. To resolve this problem, Judd et al. [JDA07] presented *apparent ridges*, which extend the definition of ridges by using view-dependent curvature and a view-dependent principle curvature direction. Thus, these lines occur where the derivative of the maximum view-dependent curvature in direction of the associated principle view-dependent curvature direction equals zero. Besides the depiction of lines, Ni et al. [NJLM06] proposed view-dependent feature lines. They reduce the details in distant regions where the perception of fine details would be difficult. Kolomenkin et al. [KST08] provide *demarcating curves* as the zero crossing of the normal curvature in the curvature gradient direction. Xie et al. [XHT*07] introduced *photic extremum lines* (*PELs*). These feature lines are as well view-dependent and light-dependent. *PELs* are defined as the set of points where the variation of illumination in its gradient direction reaches a local maximum. *PELs* generation has been optimized by Zhang et al. [ZHX*11] to reach real-time performance. They generalize the Laplacian of Gaussian edge detector to 3D surfaces. The *Laplacian lines* are defined as a set of points where the Laplacian of the illumination passes through zero.

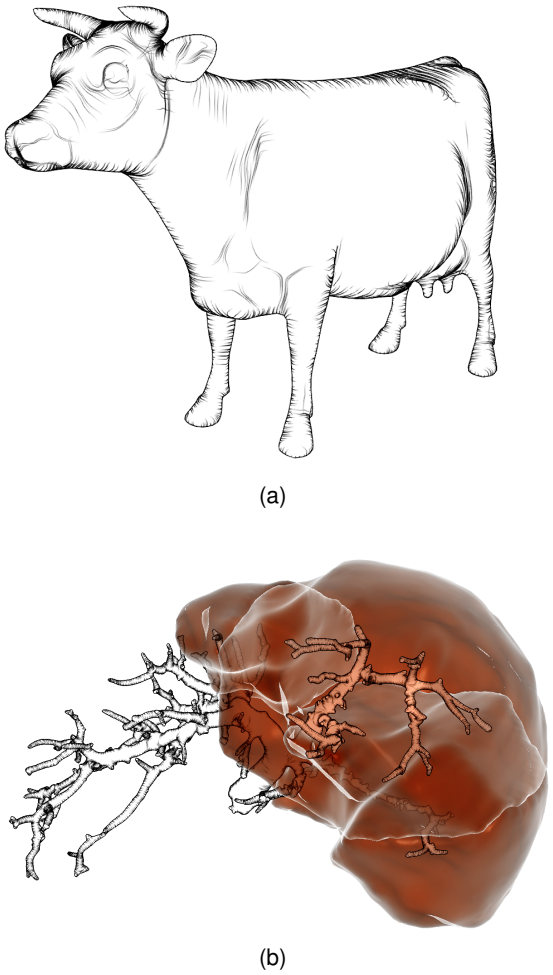


Figure 1: *ConFIS applied to two sample models: (a) cow and (b) the portal vein with three liver segments*

Furthermore, the gradient magnitude must be greater than a user-specified threshold.

Hatching is another category of common illustrative visualization techniques. They emphasize regions with high curvature by drawing lines along principal curvature directions. The lines drawn on different regions are oriented depending on the shading of the model. Hertzmann et al. [HZ00] describe a method to illustrate smooth surfaces. First, they extract the silhouette curves and separate two images of the area. These areas and the shading are used to define the hatching styles. Praun et al. [PHWF01] introduced a real-time hatching method which builds a lapped texture parametrization over the mesh which are aligned to a curvature-based direction field Gasteiger et al. [GTBP08] presented a texture-based method to hatch anatomical structures. Hummel et al. [HGH*10] examined

the use of transparancy and texturing techniques to illustrate vector fields. Buchin et al. [BW03] present a hatching scheme which can interactively change the stroke style. Zander et al. [ZISS04] use streamlines instead of textures to generate hatching lines based on curvature information. Gerl et al. [GI13] present a way to intuitively interact with the hatching illustration based on hand-drawn examples. They used a scanned-in hatching picture as input and machine learning methods to learn a model of the drawing style. A combination of different rendering techniques for medical applications was presented by Tietjen et al. [TIP05].
In summary, the available feature line methods are not sufficient to depict arbitrary shapes, especially in the domain of anatomical structures. They highlight features, but local shape properties, which relate to curvature changes, are often not satisfyingly represented. The latter is resolved by hatching methods, which have the drawbacks of being dependent on texturing or drawing too many hatching patterns. Thus, real-time performance can not always be guaranteed in combination with high visual quality.

3. Method

ConFIS is based on streamlines to illustrate the surface model. For this reason, we will first describe the regions where we want to seed the streamlines. Second, we explain how to calculate streamlines. For this, we divide this section in three parts:

1. **Contour margin:** We offer a definition for the contour. Furthermore, we explain the term *contour margin*.
2. **Feature regions:** We provide a curvature-based definition for a feature region.
3. **Streamlines:** We explain how to determine explicit streamlines.

3.1. Contour Margin

Definition 3.1 (Contour) The contour is defined as the loci of points where the surface normal and the view vector are mutually perpendicular ($\vartheta = 90°$).

For an extended overview about contours we refer to Isenberg et al. [IFH*03]. We highlight the common edge of two adjacent triangles if the signs of the dot products between the oriented face normals and the view vector change. Besides the contour line, we also draw streamlines at contour triangles. As we want to provide frame-coherent interaction, we seed streamlines at triangles which lie inside a *contour margin*. The *contour margin* is defined by the curvature-based method of Kindlmann et al. [KWTM03]. Triangles are included in the *contour margin* if the dot product of the normal and the view vector is less than $\sqrt{T\kappa_v(2-T\kappa_v)}$. Here, κ_v denotes the normal curvature in direction of the view vector and T denotes the thickness. This method provides a homgeneous *contour margin* in image space. Additionally, the opacity of the streamlines changes depending on the length to provide a convenient fade-off during the interaction.

3.2. Feature Regions

Definition 3.2 (Maxima and minima on a scalar field)

- Continuous Case: Maxima and minima can occur where the gradient of the scalar field vanishes.
- Discrete Case: We determine the gradient of the scalar field for each vertex of a triangle. The three dot products between the gradients are calculated. A maximum or a minimum occurs if two dot products are negative to identify zero-crossings.

We seed a feature streamline at a triangle t if the following properties for the mean curvature field (MCF) are fulfilled:

i. The MCF has a maximum or a minimum at t.
ii. The MCF at t exceeds a user-specified threshold.

3.3. Streamlines

Streamlines are seeded at every contour triangle and within a defined margin around the contour. Thus, the following issues have to be solved:

- Choose a suitable vector field.
- Select an appropriate starting point.
- Estimate the maximum streamline length.
- Calculate the streamline.
- Use an adaptive step size for the streamlines.

The vector field for streamline calculation: We choose a suitable curvature-based vector field for streamline calculation. In order to compute curvatures on a discrete triangle mesh, the algorithm from Rusinkiewicz [Rus04] is used, since it yields accurate and robust results even on irregularly tessellated surfaces. The curvature tensor of each triangle is determined by using finite differences of the vertex normals in direction of the edges. The vertex normals are obtained by averaging the area-weighted normals over adjacent faces. Afterwards, we compute the curvature tensor for every vertex by averaging the curvature tensors of the triangles adjacent to each vertex. The eigenvectors and eigenvalues provide the principal curvatures (PCs) and curvature directions (PCDs). At umbilic vertices, we set the PCDs to the zero vector and exclude the streamlines from seeding. We assign four vectors at the vertices and the triangle – namely two PCDs, each with two possible signs. For each triangle, only t_1 is used where t_1 is the PCD which corresponds to the maximum PC. We compare t_1 of each triangle with the four vectors of their adjacent vertices. We determine the dot product between t_1 and the four vectors of the first vertex. We select the vector which corresponds to the largest dot product. The resulting vector belongs to the final vector field to ensure that is smooth. We repeat this with the other two vertices, determine the dot products and use the vector which maximizes them. By doing so, we obtain a triple (e_1, e_2, e_3) of vectors for each triangle. We use the triple (e_1, e_2, e_3) and $(-e_1, -e_2, -e_3)$ to get two vector fields for each triangle by barycentric interpolation. With these two final vector fields,

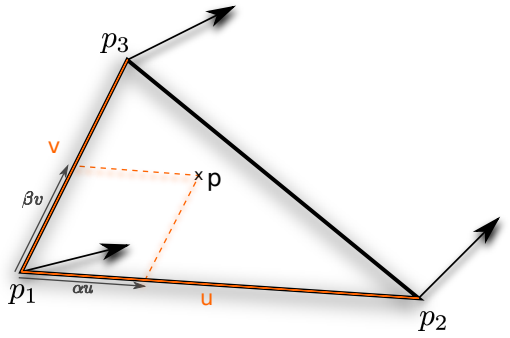


Figure 2: *Every point p in the triangle can be represented by the basis (u,v) with coefficients (α,β), $\alpha,\beta \geq 0$ and $\alpha+\beta \leq 1$. The corresponding resulting vector $e(p)$ can be obtained with the point $p=(\alpha,\beta)$ by: $e(p) = (e_2-e_1 \;\; e_3-e_1)\cdot p + e_1$.*

we can generate two different streamlines for each triangle.

Streamline starting points: Every streamline starts at the barycenter of the contour triangle. Furthermore, we are also seeding streamlines at the barycenter of adjacent triangles within the *contour margin*. As we want to achieve frame coherence, we have to start the streamlines at consistent start positions. We also generated streamlines at different randomly chosen start positions, but we could not see any visual difference. Thus, we stay at the choice of the barycenter as the start position for the streamlines.

Streamline length: We employ the principal curvatures to determine the length of the streamline. Every triangle obtains a minimum and maximum curvature κ_1,κ_2. The length L of a streamline is calculated by:

$$L = \frac{\pi}{3\cdot\max\{|\kappa_1|,|\kappa_2|\}}. \tag{1}$$

We want the streamline to have a length of one third of the the half perimeter of the osculating circle: $\frac{1}{3}\pi r$. If the streamline length L exceeds a user-defined threshold L_{max}, we set $L = L_{max}$. We suggest the median value of all calculated lengths per vertex for the streamline length threshold L_{max}.

Calculation of the streamline: Some authors use implicit or explicit iterative methods for the approximation of ordinary differential equations, e.g., [ZISS04]. Nielson et al. [NJ99] determined explicit streamlines over tetrahedral domains. Furthermore, they gave the solution for streamlines over triangle domains. We derive the explicit streamline solution for each triangle and describe this process in detail. Given a triangle $t=(p_1,p_2,p_3)$ with associated PCDs (e_1,e_2,e_3), the associated (non-normalized and non-orthogonal) basis (u,v) is built by: $u = p_2-p_1$ and $v = p_3-p_1$. Every position p and the associated vector $e(p)$ in the interior of the triangle can be calculated by the basis (u,v), see Figure 2. The parametrized streamline $c(t)$ fulfills the following condition:

$$\frac{\partial c(t)}{\partial t} = \underbrace{\begin{pmatrix} e_2-e_1 & e_3-e_1 \end{pmatrix}}_{=:A} c(t) + e_1. \tag{2}$$

Such an inhomogeneous ODE (iODE) $c' = Ac + e_1$ is solved by the method of separation of constants [Pag97, Har64]. First, we solve the homogeneous ODE (hODE) $\tilde{c}' = A\tilde{c}$. Afterwards, we determine the missing factor for the inhomogeneous part e_1. The solution of $\tilde{c}' = A\tilde{c}$ is a linear combination of a fundamental solution: $\tilde{c}(t) = \exp(A\cdot t) = \sum_{k=0}^{\infty}\frac{A^k\cdot t^k}{k!}$. In practice, we decompose the Matrix A in a Jordan-form: $A = DJD^{-1}$. Then we obtain $\tilde{c}(t) = D\cdot\exp(J\cdot t)\cdot D^{-1}$. As the inverse is not necessary for a fundamental solution system, we get $\tilde{c}(t) = D\cdot\exp(J\cdot t)$ as a fundamental system. The solution leads us to the general solution of the ODE: $c(t) = c_p(t) + \tilde{c}(t)$. We claim $c_p(t) = \tilde{c}(t)\cdot f(t)$ with an arbitrary 2D differentiable function $f(t)$ to determine a particular solution $c_p(t)$ for the iODE. The derivative of $c_p(t)$ yields:

$$\frac{\partial c_p(t)}{\partial t} = A\cdot c(t) + \tilde{c}(t)\cdot f'(t). \tag{3}$$

Comparing Equation 2 with Equation 3 leads to: $\tilde{c}(t)\cdot f'(t) = e_1$. Finally, the particular solution has the form:

$$c_p(t) = D\cdot\exp(J\cdot t)\cdot\left[\int_{t_0}^{t}\exp(-J\cdot x)\,\mathrm{d}x\right]\cdot D^{-1}\cdot e_1 + C. \tag{4}$$

For simplicity, we write $c_p(t)$ as an indefinite integral and leave out the constant C. Furthermore, we write

$$\mathcal{A}(t) := D\cdot\exp(J\cdot t)\cdot\left[\int\exp(-J\cdot t)\,\mathrm{d}t\right]\cdot D^{-1}. \tag{5}$$

We have got the final fundamental solution for the ODE. For the specific solution we need the parameters $x_1,x_2\in\mathbb{R}$ for the hODE. With the restriction $c(0)=p$, we obtain the condition: $p = \mathcal{A}(0)\cdot e_1 + D\cdot\binom{x_1}{x_2}$. Solving this equation with respect to x_1,x_2 yields:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = D^{-1}\cdot(p - \mathcal{A}(0)\cdot e_1). \tag{6}$$

Again, we refer to the appendix for a comprehensive form of x_1,x_2. Finally, we get the explicit streamline representation starting at position p with 4 and 6:

$$c(t) = c_p(t) + \tilde{c}(t)\cdot\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{7}$$

$$= \mathcal{A}(t)\cdot e_1 + D\cdot\exp(J\cdot t)\cdot D^{-1}\cdot(p - \mathcal{A}(0)\cdot e_1). \tag{8}$$

The streamline representation depends on the terms $\exp(J\cdot t)$ and $\mathcal{A}(t)$, which can be simplified, see Sec. 8. Within the triangle, streamlines have the following properties: $c(t)_x \geq 0, c(t)_y \geq 0$, and $c(t)_x + c(t)_y \leq 1$. Violating one condition leads to an intersection point of the streamline with one of the edges. Thus, we get the adjacent triangle of the edge. We can determine the underlying vector field of the new triangle according to Section 3 (The vector field for streamline calculation). Instead of using the triangle PCD as reference

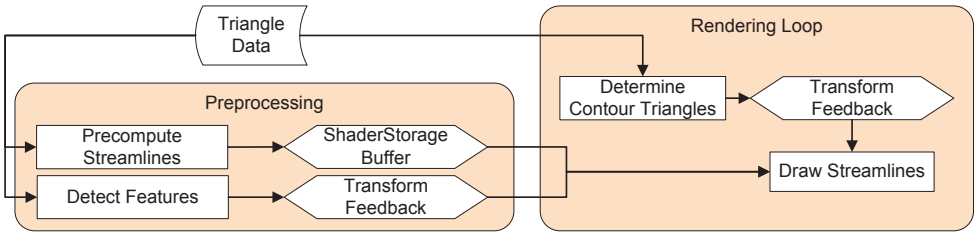


Figure 3: *The scheme of ConFIS which is divided in two parts: the preprocessing and the rendering part.*

direction, we employ the streamline direction vector.

Adaptive step size: For streamline propagation we use an adaptive step size. Whenever the length of a streamline segment exceeds the inradius of the triangle, we halve the step size. This ensures that the streamline will not immediately leave the triangle in most cases. Furthermore, the visual results of the streamlines seem to be smooth, although we do not have to calculate too much line segments in a triangle.

3.4. Advantage of explicit streamline calculation

First of all, determining the explicit streamline exhibits a lower error than iterative methods for the approximation of ODEs. Explicit streamlines do only produce an error when approximating the intersection point of one edge of the triangle with the streamline. Another advantage of explicit streamlines is the converging propagation behavior towards singularities. The explicit streamline will converge into this point. An iterative streamline can oscillate around the singularity. Moreover, iterative methods are based on a sequence of points. Thus, one point can only be approximated by using the previously calculated ones. In contrast, explicit streamlines can be calculated in parallel without knowing the previous points.

4. Algorithm and GPU Implementation

The algorithm for ConFIS on the mesh M is as follows:

1. (Optional) Subdivide and smooth M.
2. Compute the PCDs and the corresponding PCs.
3. Determine feature regions of M.
4. Compute two streamlines for all triangles.
5. Compute the contour and *contour margin*.
6. Draw the contour and feature streamlines.

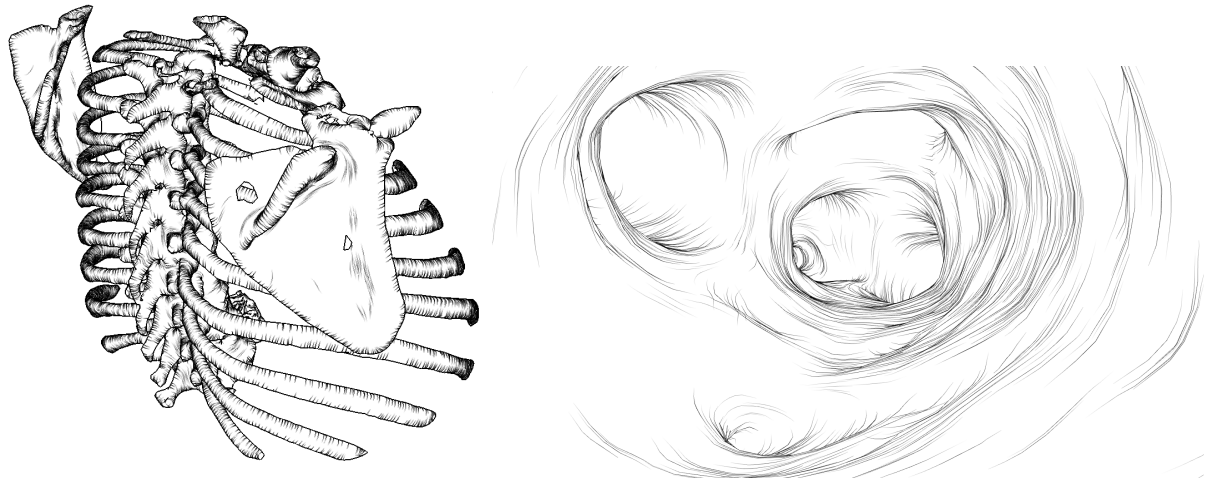
The algorithm is divided in two different parts. The first part (1.-4.) consists of the preprocessing steps and the second part (5.-6.) is executed during runtime (see Fig. 3). As shown earlier, the generation of a streamline for an arbitrary triangle requires to traverse M iteratively from one triangle to the next. For achieving a fast rendering, several APIs, such as CUDA, OpenCL, or DirectCompute are available. We chose to perform all computations with the OpenGL shader framework to be independent of graphics card vendors and to reduce any overhead by additional APIs. The shader concept is ideally suited for per-vertex and per-triangle operations, which are required by our streamline method. OpenGL shaders do natively not provide neighborhood information, such as the 1-ring of each vertex. Thus, we employ a data structure similar to the one which has been presented in [MKL*12]. Topological information, such as the location of neighboring vertices, is made available via vertex (VBO) and texture buffer objects (TBO). With this, we can seed and continue tracking a streamline at each triangle for an arbitrary number of triangles.

4.1. Preprocessing

First, we generate two streamlines at the barycenter for each triangle with given length (recall Sec. 3). These processing steps are entirely executed on the GPU via OpenGL shaders. Such a streamline buffer contains #Triangles $\times$ #LineSegments $\times$ 2 elements. For writing the streamline vertices to this buffer, we use shader storage buffer objects. Thus, the OpenGL extension ARB_SHADER_STORAGE_BUFFER_OBJECT, which is part of the OpenGL core since version 4.3, is required. With this, each triangle is able to write the vertices belonging to each of its two streamlines into the streamline buffer. Additionally, the preprocessing step allows for a detection of those triangles which shall later generate a feature streamline based on the curvature criterion (see Sec. 3.2). Until this point, nothing has been drawn – but all the streamlines have been precomputed for later rendering.

4.2. Rendering loop

During runtime, the steps 5 and 6 are executed. Using geometry shaders, we can determine the object contour and a defined *contour margin*. Thus, we could access the previously generated streamline buffer and start drawing contour and feature streamlines. Unfortunately, this yields a bad load balancing, since we seed streamlines only at comparably few triangles. Most geometry shader invocations would not perform any streamline processing. In the worst case, all threads in one thread group have to wait until all streamline-generation-threads finish – even if only one thread is generating a streamline.



(a) Ribs (b) Inside view of the pulmonary artery

Figure 4: *The ConFIS method with two anatomical surface models.*

Thus, we split the rendering stage into two render passes. At first, all triangles, which shall draw a contour streamline (recall Sec. 3.3), are identified and marked. For each of these triangles and those which have been marked during preprocessing for seeding feature streamlines, we store triangle information (e.g., vertex IDs) using OpenGL's transform feedback mechanism. In the seconds pass, rendering is performed only for the marked triangles. As a result, the GPU processes only triangles which contribute to streamline rendering. Each of these triangles can access the complete streamline buffer via TBOs. The opacity of each streamline is reduced with increasing length.

5. Evaluation

We performed a qualitative evaluation for the five line drawing techniques: *suggestive contour* (SC), *apparent ridges* (AR), *photic extremum lines* (PEL, according to Zhang et al. [ZHX*11]), *high quality hatching* (HQ), and ConFIS. The goal was to assess their capabilities for expressing relevant surface characteristics. We wanted to figure out which of the line drawing methods yields the most expressive result for the participants. The evaluation was conducted with two physicians and three researchers with background in medical visualization. Four representative surface models were chosen: ribs (Fig. 4(a)), aneurysm 1 (Fig. 5, middle row), trachea (Fig. 5, bottom row), and femur (Fig. 6(d)). The models are derived from segmenting medical image data and preprocessed to ensure an appropriate and homogeneous degree of tessellation. For all compared methods, we employed the original implementations by the corresponding authors. The evaluation was conducted in three steps:

1. Each participant was shown the shaded surface models in different order. They had the possibility to explore the model interactively and gain a 3D impression.
2. The second task was to adjust the specific parameters of the illustration methods to obtain a subjectively satisfying and informative result. During this, we noted the participants' spoken comments and the parameter sets they were satisfied with.
3. The third part of the evaluation consisted of a visual comparison and a qualitative assessment between the feature line methods. Based on the recorded parameter sets, each participant should assess which method is considered more appropriate to express surface features and which limitations have been observed.

Aneurysm 1 model: The participants mentioned that the generated lines by the feature line methods were not appropriate to gain a comprehensive spatial impression. Most of the generated lines were considered distracting. On the other hand, these methods depicted parts of the bifurcation well. All participants agreed that the hatching method generates a reasonable 3D impression. For HQ, the evenly spread lines can not depict important features, e.g., the border between the vessel and aneurysm sac. Furthermore, several participants criticized the low performance of the HQ implementation (~13 fps). ConFIS fulfilled the demands to illustrate relevant features and to convey an appropriate 3D impression. All participants chose ConFIS as their preferred line drawing technique.

Trachea model: The inner view of the trachea has two features: the elongated structure and the bifurcation, where the carina tracheae splits into both branches. The participants confirmed that the feature line methods can depict the elongated structures but fail to enhance the carina tracheae. Apart from that, they explained that the hatching method as

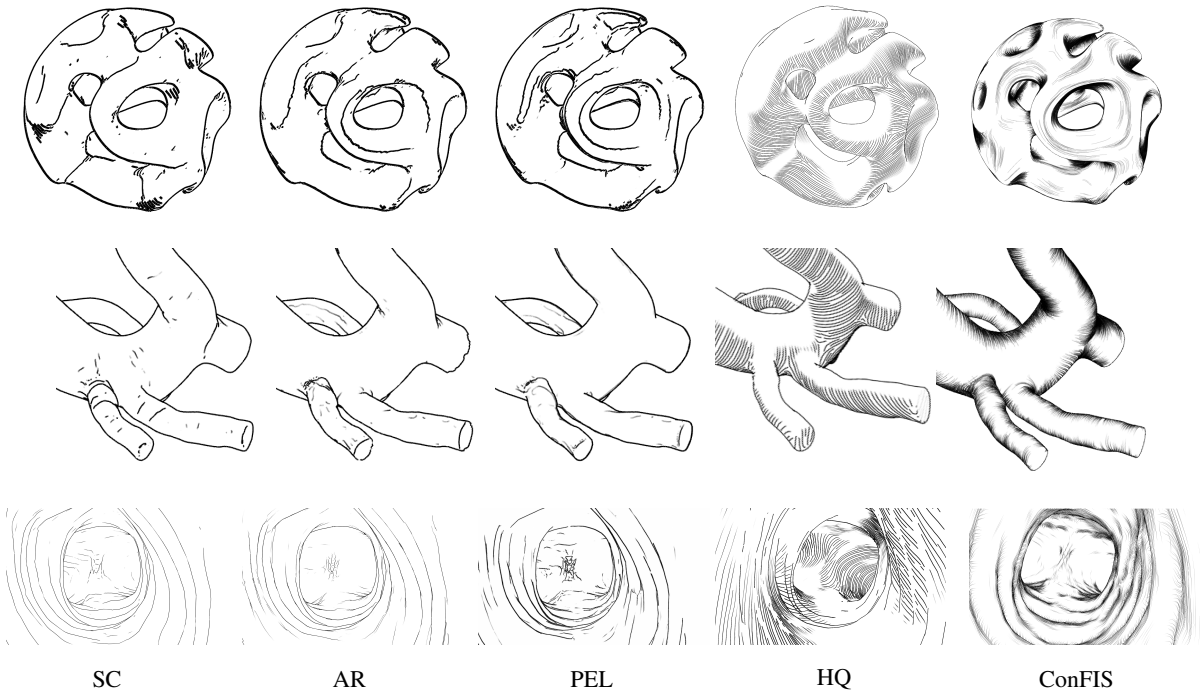


Figure 5: *Different surface models displayed with* suggestive contour *(SC),* apparent ridges *(AR),* photic extremum lines *(PEL),* high quality hatching *(HQ), and ConFIS. The models are from top to bottom: hyperthing, aneurysm 1, and endoscopic view of a trachea*

well as ConFIS depict both properties well. One participant found some streamlines slightly disturbing and unnecessary to gain a spatial impression. The hatching method could not highlight the bifurcation features. In contrast, it looked like a planar transition from one bronchus to the other. Finally, all participants preferred the ConFIS method.

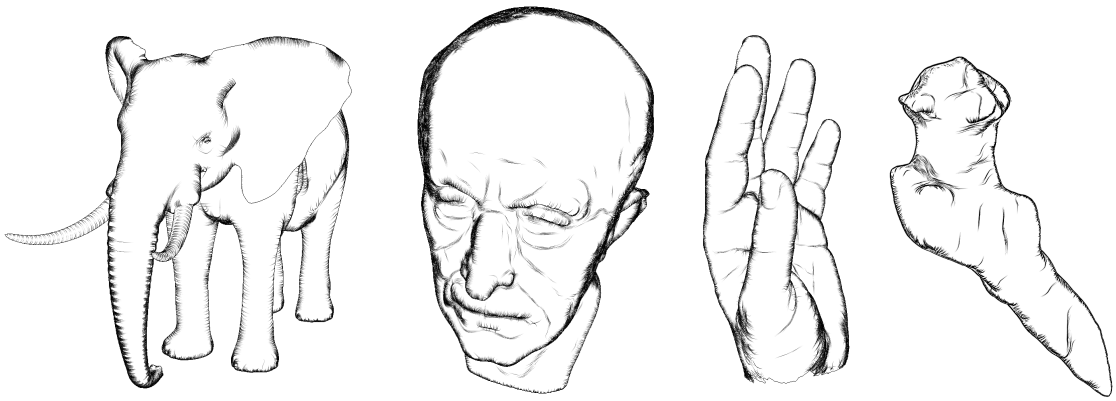
Ribs model: The ribs model was chosen to evaluate the 3D feeling of the mesh even if the surface has a lot of structures. The participants get a reasonable 3D impression by all line drawing methods. Some participants mentioned that there are only small differences between the feature line methods. One participant explained that the impression of the model is appropriate during the interaction but seeing only a screenshot would confuse. The participant could not set the ribs apart from the gaps. Furthermore, some lines which are produced by feature line methods are distracting and the hatching method can not illustrate the dents. ConFIS illustrates all ribs well and the participants can distinguish the ribs from the gaps and all dents are depicted as well.

Femur model: Some of the participants found fault with the view-dependent feature illustrations. The feature line methods only show some dents first if the camera position is chosen well. Again, two participants criticized the missing details using the hatching method. Some dents are missing and without interactive exploration some important regions are missed. Those regions have been highlighted with ConFIS, which was again preferred.

The results of our evaluation can be summarized as follows:

- Current line drawing techniques achieve satisfying results only if the models exhibit a smooth and regularly tessellated surface.
- The clutter of surfaces derived from measured image data, such as noise and staircase artifacts, are usually emphasized.
- For some cases, line drawing methods are not able to depict relevant features.
- The ConFIS method was the most expressive technique in our comparison.

However, the informal study does not allow a definitive statement and requires further evaluation. ConFIS is able to provide a sparse representation of a model's surface, since illustrative patterns are drawn along characteristic contours and only sparsely within the surface. Thus, ConFIS might also serve to depict the anatomical context as spatial reference in medical illustrations, e.g., flow visualization. Our method provides such spatial information and gives also hints on local shape properties. However, the discussion with



(a) Elephant (b) Max Planck (c) Hand (d) Femur

Figure 6: *The ConFIS method with different surface models.*

the physicians showed that such illustrations are not suitable for diagnostic purposes. In therapy planning, illustrative pictures are used for discussions. Especially in neck surgery physicians use abstract 3D illustrations as a printout to draw resection lines and access path planning.

6. Results

We performed different experiments in order to assess the performance of our approach. The experiments consider the frame rate achieved with ConFIS for different artificial and anatomical surface meshes (see also Fig. 6 and Tab. 1). Our approach is implemented on a mid-class desktop computer with an Intel Core i7 CPU (2.8GHz), 4GB RAM, and an NVidia GeForce GTX 660 Ti. For all employed surface models, rendering could be performed in real-time. Surface models with given curvature vector field with the corresponding number of triangles, averaged generated streamlines, initialization time, and average frame rates are summarized in Table 1. Streamline calculation and feature region detection is performed only once during the preprocessing step. The latter requires a lot of memory for storing the vertices of the precomputed streamlines. For a sample model with about 100k faces and, e.g., 50 vertices per streamline, such buffer may comprise ~153 MB. However, with recent graphics cards, at least 1024 MB are usually available.
During runtime, only the contour is determined and streamlines at contours and features are drawn. Our approach depends on two user-defined values, which our evaluation participants confirmed to be intuitive. We used different models and compared the results of different line drawing techniques, see Figure 5. In most experiments our approach could express the relevant surface features (see Fig. 6). Since our illustration technique seeds streamlines at the barycenter of triangles, it is tessellation-dependent. Thus, for low resolution meshes, the overall visual impression will be disturbed by only sparsely drawn lines. However, the user could always get an impression of the surface characteristics. During our tests with other feature line and hatching methods, we noticed that this seems to be a general problem.

Table 1: *Performance test of ConFIS for all shown models.*

Model	#Δ	# SL	Init per s	FPS
Aneurysm 2	16,778	5,134	2,735	440
Cow	23,216	7,788	0,559	260
Femur	40,978	11,734	0,697	160
Trachea	69,964	25,501	1,617	127
Portal Vein	80,062	23,067	2,849	155
Ribs	85,736	26,780	3,071	87
Hyperthing	88,756	41,023	2,270	57
Max Planck	98,260	30,407	1,992	90
Aneurysm 1	98,970	32,659	2,002	78
Pulmonary Artery	100,000	30,680	2,735	104
Hand	105,860	25,915	2,298	72
Elephant	157,160	48,875	2,665	57

7. Conclusion and Future Work

In this paper, we have presented ConFIS, a novel illustrative visualization technique for surface models based on streamlines. The streamlines have the advantage that the user gets a natural impression of the curvature of the model. Furthermore, the user gains an enhanced 3D impression. We made different comparisons to other feature line methods. Our experiments showed that ConFIS depicts most of the surface models well. ConFIS illustrates only salient regions which fulfills the conditions mentioned in Sec. 3.2. Obviously, convex regions will not be illustrated. We can modify the parameters in a way to emphasize sharp features of models.

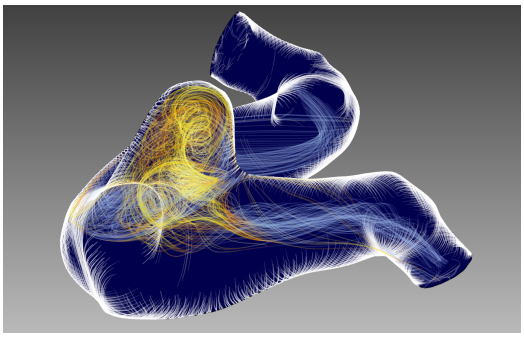


Figure 7: *Fresnel alternative with white streamlines (aneurysm 2).*

However, more parameters are required to define a lower and upper bound. We think about to determine the Hessian matrix of the mean curvature scalar field to obtain the minima and maxima of the surface mesh. We used the concept of the *contour margin* to provide a frame-coherent illustration.

An aspect of our future work is to parametrize the contour of the surface model in object space. First, the advantage is to uniformly distribute the seed points for the streamlines equidistant to each other. This provides the possibility to apply ConFIS even on low-tessellated surfaces. On the other hand, the streamlines are consistently drawn during the runtime. This would result in a frame-coherent approach which does not depend on the tessellation of the surface model. Another approach we would like to focus on is the simulation of light. As ConFIS seeds streamlines at the contour and at the *contour margin*, this conveys the feeling of a headlight. Therefore, the middle parts of cylindrical shapes are illuminated. Following the conclusions of Šoltészová et al. [vPV11], we think about a displacement of the view vector to simulate different illumination styles.

As an outlook, we consider to use ConFIS as an alternative rendering technique. ConFIS could be used in a similar manner as Fresnel shading to convey the impression of bending anatomical structures with streamlines, see Figure 7. We used Fresnel shading and ConFIS in combination to analyze the blood flow inside the vessel and to perceive the bending of the vessel. Furthermore, we consider extending our evaluation according to Kim et al. [KHSI04] and Blair et al. [BH07]. In their experiments, participants had to orient vectors on the model surface manually to fit the perceived surface normal. However, with respect to our current study we consider that ConFIS provides a good approach on filling the gap between feature line and hatching methods.

8. Appendix

The Jordan-matrix J can be represented by different forms. We have to consider all cases in order to simplify $\exp(J\cdot t)$ and $\int \exp(-J\cdot t)\,\mathrm{d}t$:

i. $J=\begin{pmatrix}\kappa & 1\\ 0 & \kappa\end{pmatrix}, \kappa\in\mathbb{R}$.

ii. $J=\begin{pmatrix}\kappa_1 & 0\\ 0 & \kappa_2\end{pmatrix}, \kappa_1,\kappa_2\in\mathbb{C}$.

Lemma 8.1 For one of the cases i. or ii. we have different simplified representations for $\exp(J\cdot t)$:

$$a)\qquad i.\ \ \&\ \kappa\neq 0:\ \exp(J\cdot t)=\exp(\kappa t)\begin{pmatrix}1 & t\\ 0 & 1\end{pmatrix},$$

$$b)\qquad i.\ \ \&\ \kappa=0:\ \exp(J\cdot t)=\begin{pmatrix}1 & t\\ 0 & 1\end{pmatrix},$$

$$c)\qquad ii.\qquad :\ \exp(J\cdot t)=\begin{pmatrix}\exp(t\cdot\kappa_1) & 0\\ 0 & \exp(t\cdot\kappa_2)\end{pmatrix}.$$

Proof **a)** First, using the definition of the matrix exponential: $\exp(J\cdot t)=\sum_{k=0}^{\infty}\frac{J^k t^k}{k!}$. Applying induction leads to $J^n=\begin{pmatrix}\kappa^n & n\cdot\kappa^{n-1}\\ 0 & \kappa^n\end{pmatrix}$. So we get $\exp(J\cdot t)=\exp(\kappa t)\begin{pmatrix}1 & t\\ 0 & 1\end{pmatrix}$.
b) J is a nilpotent matrix and we get $J^0=Id$, $J^1=J$, and $J^2=0$. So we get $\exp(J\cdot t)=\begin{pmatrix}1 & t\\ 0 & 1\end{pmatrix}$.
c) This case can be verified by knowing $J^n=\begin{pmatrix}\kappa_1^n & 0\\ 0 & \kappa_2^n\end{pmatrix}$ and this leads to $\exp(J\cdot t)=\begin{pmatrix}\exp(t\cdot\kappa_1) & 0\\ 0 & \exp(t\cdot\kappa_2)\end{pmatrix}$. □

Lemma 8.2 For the cases i. and ii. we get the following simplifications for $\mathcal{A}=D\begin{pmatrix}x & y\\ 0 & z\end{pmatrix}D^{-1}$:

a) $\underline{\kappa_1,\kappa_2\neq 0}$: $x=-\kappa_1^{-1}$, $z=-\kappa_2^{-1}$. Further, if $\kappa_1=\kappa_2$ then $y=\kappa_1^{-2}$ else $y=0$.
b) $\underline{\kappa_2=0}$: $z=t$. Further, if $\kappa_1=\kappa_2$ then $x=t$, $y=0.5t^2$ else $x=-\kappa_1^{-1}$, $y=0$.
c) $\underline{\kappa_1,\kappa_2\in\mathbb{C}}$:

$$\mathcal{A}(t)=\Re(\kappa^{-1})\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}+\frac{\Im(\kappa^{-1})}{bc-ad}\begin{pmatrix}ac+bd & -a^2-b^2\\ c^2+d^2 & -ac-bd\end{pmatrix},$$

with $D=\begin{pmatrix}a+bi & a-bi\\ c+di & c-di\end{pmatrix}$.

Proof Using the property $(\exp(A))^{-1}=\exp(-A)$ and integrating the result, we get the simplification. If the eigenvalues κ_1,κ_2 of J are complex, then it follows: $\kappa_1=\bar{\kappa_2}$, so we write $\kappa:=\kappa_1$. Additionally, if v is an eigenvector, then its conjugate is also an eigenvector. □

Acknowledgments

Tobias Moench is funded by the German Federal Ministry of Education and Research (BMBF) within the ViERforES-II project (no.01IM10002B). The author wish to thank Long Zhang, S. Rusinkiewicz, D. DeCarlo, T. Judd, and T. Isenberg who made PEL, suggestive contours, apparent ridges and high quality hatching available. The models are courtesy of Max Planck Institute, Inria, Espona, and O. Beuing.

References

[BH07] Bair A., House D. H.: Grid with a view: Optimal texturing for perception of layered surface shape. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1656–1663. 9

[BW03] Buchin K., Walther M.: *Hatching, Stroke Styles & Pointillism*, vol. ShaderX2 - Shader Tips and Tricks. Wordware Publishing, September 2003, ch. Rendering Techniques, pp. 340–347. 3

[CSD*09] Cole F., Sanik K., DeCarlo D., Finkelstein A., Funkhouser T., Rusinkiewicz S., Singh M.: How well do line drawings depict shape? In *In Proceedings of ACM SIGGRAPH 2009* (2009), pp. 28:1–28:9. 1

[DFRS03] DeCarlo D., Finkelstein A., Rusinkiewicz S., Santella A.: Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2003), 848–855. 2

[GI13] Gerl M., Isenberg T.: Interactive example-based hatching. *Computers & Graphics* (2013). 3

[GTBP08] Gasteiger R., Tietjen C., Baer A., Preim B.: Curvature- and model-based surface hatching of anatomical structures derived from clinical volume datasets. In *Smart Graphics* (2008), pp. 255–262. 2

[Har64] Hartman P.: *Ordinary Differential Equations*. John Wiley & Sons, 1964. 4

[HGH*10] Hummel M., Garth C., Hamann B., Hagen H., Joy K.: Iris: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1319–1328. 2

[HZ00] Hertzmann A., Zorin D.: Illustrating smooth surfaces. In *Proceedings of ACM SIGRAPH 2000* (2000), pp. 517–526. 2

[IFH*03] Isenberg T., Freudenberg B., Halper N., Schlechtweg S., Strothotte T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications 23*, 4 (July 2003), 28–37. 3

[IFP95] Interrante V., Fuchs H., Pizer S.: Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of the 6th conference on Visualization 1995* (1995), pp. 52–59. 2

[INC*06] Isenberg T., Neumann P., Carpendale S., Sousa M., Jorge J.: Non-photorealistic rendering in context: an observational study. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2006), pp. 115–126. 1

[JDA07] Judd T., Durand F., Adelson E.: Apparent ridges for line drawing. In *ACM SIGGRAPH 2007 papers* (2007), p. 19. 2

[KHSI04] Kim S., Hagh-Shenas H., Interrante V.: Conveying shape with texture: Experimental investigations of texture's effects on shape categorization judgments. *IEEE Transactions on Visualization and Computer Graphics 10*, 4 (2004), 471–483. 9

[KST08] Kolomenkin M., Shimshoni I., Tal A.: Demarcating curves for shape illustration. In *ACM SIGGRAPH Asia 2008 papers* (2008), pp. 157:1–157:9. 2

[KWTM03] Kindlmann G., Whitaker R., Tasdizen T., Moller T.: Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (2003), VIS '03, IEEE Computer Society, pp. 67–. 3

[MKL*12] Mönch T., Kubisch C., Lawonn K., Westermann R., Preim B.: Visually Guided Mesh Smoothing for Medical Applications. In *VCBM 2012 - Eurographics Workshop on Visual Computing for Biology and Medicine* (September 2012), pp. 91–98. 5

[MR11] Muthukrishnan R., Radha M.: Edge detection techniques for image segmentation. *International Journal of Computer Science and Information Technology (IJCSIT) Vol 3, No 6* (Dec 2011). 2

[NJ99] Nielson G. M., Jung I.-H.: Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (Oct. 1999), 360–372. 4

[NJLM06] Ni A., Jeong K., Lee S., Markosian L.: Multi-scale line drawings from 3d meshes. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (2006), pp. 133–137. 2

[NSH08] Nadernejad E., Sharifzadeh S., Hassanpour H.: Edge detection techniques: Evaluations and comparisons. *Applied Mathematical Sciences Vol. 2*, no. 31 (2008), 1507 – 1520. 2

[Pag97] Page J. M.: *Ordinary Differential Equations - An Elementary Text-Book With An Introduction To Lies's Theory Of The Group Of One Parameter*. MacMillian and CO., 1897. 4

[PHWF01] Praun E., Hoppe H., Webb M., Finkelstein A.: Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 579–584. 2

[RCDF08] Rusinkiewicz S., Cole F., DeCarlo D., Finkelstein A.: Line drawings from 3d models. In *ACM SIGGRAPH 2008 classes* (July 2008), pp. 39:1–39:356. 2

[Rus04] Rusinkiewicz S.: Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission* (Sept. 2004). 3

[SR09] Senthilkumaran N., Rajesh R.: Edge detection techniques for image segmentation – a survey of soft computing approaches. *International Journal of Recent Trends in Engineering, Vol. 1, No. 2* (May 2009). 2

[TIP05] Tietjen C., Isenberg T., Preim B.: Combining silhouettes, surface, and volume rendering for surgery education and planning. In *EuroVis* (2005), pp. 303–310. 3

[Vio05] Viola I.: *Importance-Driven Expressive Visualization*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, June 2005. 1

[vPV11] Šoltészová V., Patel D., Viola I.: Chromatic shadows for improved perception. In *Proceedings of the ACM SIGGRAPH 2011* (2011), pp. 105–116. 9

[XHT*07] Xie X., He Y., Tian F., Seah H.-S., Gu X., Qin H.: An effective illustrative visualization framework based on photic extremum lines (pels). *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1328–1335. 2

[ZHX*11] Zhang L., He Y., Xia J., Xie X., Chen W.: Real-time shape illustration using laplacian lines. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 993–1006. 2, 6

[ZISS04] Zander J., Isenberg T., Schlechtweg S., Strothotte T.: High quality hatching. *Comput. Graph. Forum 23*, 3 (2004), 421–430. 3, 4