Eurographics Workshop on Visual Computing for Biology and Medicine (2016)
S. Bruckner, B. Preim, and A. Vilanova (Editors)

Sline:
Seamless Line Illustration for Interactive Biomedical Visualization

Nils Lichtenberg[1], Noeska Smit[2], Christian Hansen[3] and Kai Lawonn[1]

[1]Institute for Computational Visualistics, University of Koblenz, Germany
[2]Visualization group, University of Bergen, Norway
[3]Computer Assisted Surgery group, University of Magdeburg, Germany

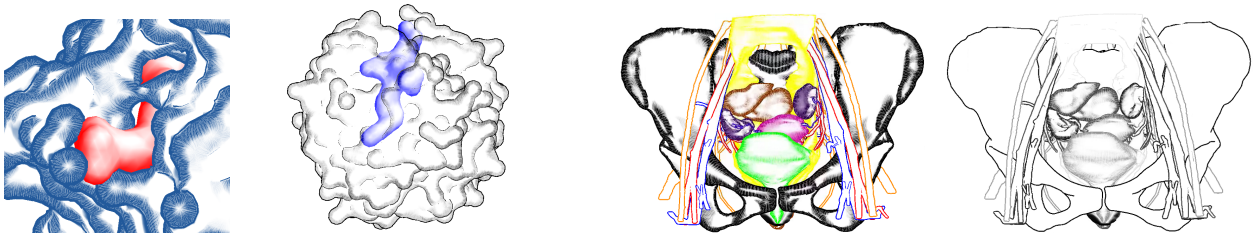


Figure 1: Examples of biomedical focus-and-context scenes generated with Sline.

Abstract

In medical visualization of surface information, problems often arise when visualizing several overlapping structures simultaneously. There is a trade-off between visualizing multiple structures in a detailed way and limiting visual clutter, in order to allow users to focus on the main structures. Illustrative visualization techniques can help alleviate these problems by defining a level of abstraction per structure. However, clinical uptake of these advanced visualization techniques so far has been limited due to the complex parameter settings required.

To bring advanced medical visualization closer to clinical application, we propose a novel illustrative technique that offers a seamless transition between various levels of abstraction and detail. Using a single comprehensive parameter, users are able to quickly define a visual representation per structure that fits the visualization requirements for focus and context structures. This technique can be applied to any biomedical context in which multiple surfaces are routinely visualized, such as neurosurgery, radiotherapy planning or drug design. Additionally, we introduce a novel hatching technique, that runs in real-time and does not require texture coordinates. An informal evaluation with experts from different biomedical domains reveals that our technique allows users to design focus-and-context visualizations in a fast and intuitive manner.

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

1. Introduction

Human anatomy consists of many closely arranged structures. In 3D visualization of anatomy, the proximity and number of structures can cause perception problems, hampering the focus on structures of interest. Especially for regions featuring complex anatomy, such as the pelvis, many organs are arranged in a confined region and visualization of these spatial arrangements is difficult. Furthermore, the number of different structures visualized can lead to visual clutter or distraction from the areas of interest. For many medical visualization tasks, a pathology or target structure needs to be visualized in anatomical context. Simply showing every structure is then not ideal, due to a potential visual overload.

Advanced visualization techniques, such as focus-and-context techniques, are able to emphasize structures of interest, and deemphasize context structures. Context representations are then abstracted and serve to provide an indication of the spatial extent, without presenting much shape detail. In this way, distraction from the focus structure(s) is prevented. The focus structures are then presented in a detailed way, so that shape and spatial perception is supported. Illustrative, or non-photorealistic rendering (NPR) tech-

DOI: 10.2312/vcbm.20161281

niques can be employed to provide the necessary level of abstraction.

While illustrative techniques can be successfully applied to medical visualization problems, e.g., to visualize multimodal data or for surgical treatment planning, clinical uptake of these novel techniques so far is limited. In clinical contexts where segmentation is routinely performed, such as for neurosurgery or radiotherapy planning, structures are often visualized using Phong shading combined with a simple opacity setting. Rendering structures in a similar shading style could lead to visual distraction from focus structures, or in case the structures are rendered in a transparent way, reduced shape perception. While applications such as these would benefit from more advanced visualization techniques, a limiting factor is the complex parameter specification required to generate effective visual representations. Furthermore, many illustrative techniques, such as silhouettes, suggestive contours, and hatching exist, but they are not presently available in a comprehensive implementation.

In this paper, we present the integration of several NPR techniques into a single application, entitled Sline (Seamless Line Illustration). With this, our main contributions are the following:

1. Our technique provides a seamless transition between surface rendering styles, from high levels of abstraction to less abstraction and finally to a smooth, non-illustrative appearance.
2. Using a single parameter setting per structure, users have the ability to quickly and intuitively visualize a complete 3D scene with several focus and context structures.
3. We introduce a novel real-time hatching technique, which does not require texture coordinates.

We define an ordered set of rendering styles to represent different stages of abstraction. Sline does not require texture coordinates for its rendering styles, in order to make our technique suitable for a wide range of biomedical research data. Each rendering style depends on several parameters of which a subset will be used for the transition among the stages. By mapping these parameter subsets to a single comprehensive parameter, users can intuitively and quickly set the desired visual representation for structures or groups of structures. The resulting visualization can convey focus-and-context information through choice of rendering styles, even without the use of color (see Figure 1).

The rest of this paper is structured as follows. First, in Section 2, we describe work related to illustrative visualization techniques and their biomedical applications. In Section 3, we describe the methods that constitute Sline. Afterwards, we present the results of Sline in three case studies combined with an informal evaluation with domain experts in Section 4, followed by a discussion in Section 5 as well as a conclusion and outlook on future research directions in Section 6.

2. Related Work

The basis of our work focuses on a smooth transition from distinct illustrative visualization techniques at varying levels of abstraction to a smooth, non-illustrative rendering style. With this work, we mainly focus on line drawing techniques. Thus, in this section we will give an overview of the most commonly used line drawing techniques: *feature lines* and *hatching* approaches. Furthermore, we discuss applications of these techniques in the biomedical domain.

Feature Lines Feature lines aim to represent the strongest shape cues of a surface mesh with lines. When an artist is asked to draw a surface with only a few lines, the lines would be placed at the most significant regions. Although this is a very subjective task, some lines will be placed more often than others, depending on the geometric details of the underlying surface. Especially if the normals of the surface vary strongly, e.g., at a strong edge, a line would commonly be drawn there. Feature line techniques aim to automate this process, such that these lines are placed based on shading or surface information.

Shading approaches take the normals of the surface mesh and the light vector into account. Mostly, a headlight is used where the light vector coincides with the view direction of the camera. The most intuitive feature line in this category is the contour. The contour is defined as the loci of points where the normal and the view direction are mutually perpendicular. As the normal is only defined at the vertices of the mesh, the dot product of the view vector and the normal is determined at these points. Afterwards, the signs of the vertices per triangle are checked and if the sign changes, a line is constructed on the triangle that interpolates the negative and positive value such that a zero-crossing is obtained. This contour yields a reasonable impression of the surface, but is not sufficient for a detailed spatial impression. Another approach, which is based on shading, was presented by DeCarlo et al. [DFRS03]. Xie et al. [XHT*07] presented *photic extremum lines*. Their approach identifies regions of high variance in the shading. For this the maximum of the magnitude of the light gradient in direction of the light gradient is determined. The user has the possibility to add more light sources to influence the result. This feature can be useful when regions with a high amount of noise produce too many feature lines. Adding more light sources that point to this region reduces the number of lines. This approach was later improved by Zhang et al. [ZHS10] to improve the runtime performance. Zhang et al. [ZHX*11] also presented *Laplacian lines* as an extension of the Laplacian-of-Gaussian edge detector used in image processing to surface meshes. The Laplacian of the normal is determined component-wise, again yielding a vector. Afterwards, the dot product with the resulting vector and the view direction is determined, and the zero-crossings are used as potential candidates for the feature lines. Although this approach is fast in comparison to other approaches, the pre-processing step of determining the Laplacian is very time-consuming, as the authors suggested to use the Belkin weights [BSW08].

Regarding the feature line approaches that are based on geometric properties of the surface, Interrante et al. [IFP95] proposed *ridges and valleys*. Their approach was adapted to triangulated surface meshes by Ohtake et al. [OBS04]. The ridge and valley lines require principal curvature information. First, the zero-crossing of the directional derivative of the greatest curvature along the corresponding principal curvature directions is determined. Depending on the sign of the second directional derivative and on the sign of the curvature, the feature line is either a ridge or a valley. A more advanced approach was presented by Judd et al. [JDA07]. Their

apparent ridges are determined similar to the ridges and valleys, but in contrast they employ their own definition of curvature. They presented a view-dependent curvature such that contours are also determined.

All the presented feature line methods have advantages and disadvantages. An overview of feature line techniques, which summarizes these positive and negative aspects, can be found in the survey by Lawonn and Preim [LP15].

Hatching In contrast to feature lines, where the most salient regions are depicted by single lines, hatching tries to convey a spatial impression of the surface, by covering the surface with a large number of line strokes. The stroke style depends on the underlying shading, such that the number of lines increases for regions with dark shading. The first approach of hatching lines on a surface was introduced by Hertzmann and Zorin [HZ00]. They determined the principal curvature directions and smoothed them to obtain less noisy results. Afterwards, they determined the integral lines that represent the lines over the surface. A texture-based approach was introduced by Praun et al. [PHWF01]. They used lapped textures that vary in hatching size and different sets of textures encoding the brightness of the shading. For dark regions a cross-hatched texture was used, and for bright regions only a few lines were drawn. Finally, these textures were projected on the surface. In contrast, Zander et al. [ZISS04] proposed to use geometrical lines as streamlines. The lines are then individually propagated on the surface along the principal curvature direction. Cylinders around the lines ensure a specific distance from one line to another. A dynamic approach was presented by Kim et al. [KYYL08]. They determined the principal curvature direction on the GPU and aligned hatching textures along this curvature direction. With their method, a hatching approach was introduced that can be applied on animated surfaces in a frame-coherent manner. A line drawing technique that combines the advantages of feature lines and hatching was presented by Lawonn et al. [LMP13]. They determined feature regions and the contour margin, which are the starting point for streamline propagation. Later, they improved their approach such that animated surfaces can be illustrated with hatching lines as well [LKEP14]. In contrast to our method, they used a noise texture to generate the hatching, which may result in blurry lines.

Biomedical Applications Illustrative techniques, including line drawings, have been successfully applied to biomedical data in various application contexts. In the biology domain, Tarini et al. [TCM06] presented edge cueing techniques for molecular visualization. Using depth-aware halos, depth-revealing contour lines, and intersection-revealing contour lines, they enhanced real-time visualization of molecules. Weber et al. [Web09] developed the ProteinShader application, which renders macromolecules using illustrative techniques. They employ real-time half-toning, bend textures and edge-line generation algorithms to visualize proteins, but in contrast to our approach, rely on texture mapping to attain their results. Parulek et al. [PRV13] utilize level-of-detail to enable the interactive rendering of large molecules. They apply a seamless transition from the display of the solvent-excluded surface to sphere rendering. More closely related to the work presented here, Zwan et al. [vdZLBI11] presented illustrative molecular visualization with continuous abstraction. The molecules were visualized using various levels from a space filling representation to abstract ribbons, specifically designed for molecular data.

In the medical domain, Interrante et al. [IFP95] enhanced transparent skin surfaces with the aforementioned *ridge and valley* lines for radiation therapy treatment planning. Dong et al. [DCLK03] presented NPR techniques for segmented volumetric medical data. They generate silhouette points and strokes in order to provide volumetric hatching, but only employ this to generate static images. Tietjen et al. [TIP05] combined silhouettes, surface, and volume rendering in a scene-graph-based application aiming at surgical education and planning. Their method successfully integrated strokes with surface and volume rendering, but did not include hatching and required complex interaction for adjustments. Ritter et al. [RHD*06] presented work on real-time illustration of vascular structures. They spatially accentuated vessels using hatching, distance-encoded surface and shadow illustrations. Based on this work, Lawonn et al. [LLPH15] proposed a combination of supporting lines, view-aligned quads, hatching and illustrative shadows to improve depth assessment of complex 3D vascular models.

Jainek et al. [JBB*08] combined volume and surface rendering to visualize anatomical and functional brain data using illustrative techniques. Gasteiger et al. [GTBP08] proposed a texture-based method to hatch anatomical structures derived from clinical volume datasets. They apply curvature-based hatching by incorporating model-based preferential directions of the underlying anatomy. Chu et al. [CCG*08] presented perception-aware depth cueing for illustrative vascular visualization, using depth-aware silhouettes, color-coded shading strokes and line-culling highlights. Svakhine et al. [SEA09] depth enhanced medical volume visualization with artistic styles for outlining features and conveying depth information. Hansen et al. [HWR*10] visualized 3D planning models for augmented reality in liver surgery using illustrative techniques. In their work, they combine distance-encoding silhouettes and surfaces with procedural textures for intra-operative visualization. Svetachov et al. [SEI10] illustrated brain fiber tracts from DTI data in an interactive application using interactive slice-based hatching. Born et al. [BMGS13] proposed illustrative visualization of cardiac and aortic blood flow from 4D MRI acquisitions.

In contrast to previous work, our method is the only method that supports a seamless interactive transitions between different levels of abstraction by integrating several illustrative rendering techniques, without relying on textures or assumptions about the underlying models.

3. Method

Our method, Sline, provides a smooth transition from silhouettes, to more detailed illustrative styles, to Phong shading for surface meshes. In this way, we achieve a transition from a very abstract representation along several levels of abstraction to a non-illustrative visualization. For every mesh $\mathcal{M}$ in the scene an individual rendering style can be chosen. To attain a step-wise decrease in illustrative levels of abstraction, we employ the following techniques in order:

1. (a) Silhouettes and (b) Contours (Section 3.2)
2. Suggestive Contours (Section 3.3)

3. Hatching (Section 3.4)
4. Phong Shading

An overview illustration of these stages is given in Figure 2.

The next sections describe the preprocessing, implementation of the algorithms, and the parameterization of the transition between consecutive rendering steps respectively.

3.1. Preprocessing

Silhouettes and contours do not require any preprocessing. The suggestive contours as in [DFRS03] and our hatching approach require the computation of the radial surface curvature $\kappa_{\mathbf{w}}(\mathbf{p})$ and its directional derivative $D_{\mathbf{w}}\kappa_{\mathbf{w}}$. Here, $\mathbf{p} \in \mathcal{M}$ is a point on the mesh $\mathcal{M}$ and $\mathbf{w}$ is the projection of the view vector $\mathbf{v}$ on the tangent plane at $\mathbf{p}$. Note that $\mathbf{v}$ points from the camera towards $\mathcal{M}$. The scalar field $\kappa_{\mathbf{w}}$ is computed based on the light gradient vector field $\mathbf{k}$. We compute $\mathbf{k}$ and $\kappa_{\mathbf{w}}$ as in the technique proposed by Lawonn et al. [LKEP14] and use the method described by Judd et al. [JDA07] to compute $D_w\kappa_{\mathbf{w}}$. Finally, we define a feature size $\mathcal{F}$, that will be used to adjust parameters that depend on the world size of a mesh. For example, the curvature on the surface of a large sphere is smaller than that on a small sphere. Thus, we set $\mathcal{F}$ to the bounding sphere radius of $\mathcal{M}$.

3.2. Silhouettes and Contours

Silhouettes describe the outline of an object, while contours are found where the surface normal of an object is orthogonal to the view vector. Our silhouettes and contours use the same algorithm, which is similar to the object space algorithm described by Hertzmann [Her99]: We find zero crossings on the triangle edges where $\langle \hat{\mathbf{n}}, \mathbf{v} \rangle = 0$ holds, with $\langle .,. \rangle$ defining the dot product. Here, $\hat{\mathbf{n}}$ denotes the interpolated normal along the edge between two vertices. If we find zero crossings on two edges of a triangle, we create a contour patch with four vertices - two for each of the two edges, given by:

$$\hat{\mathbf{p}}_0; \;\; \hat{\mathbf{p}}_1 = \hat{\mathbf{p}}_0 + a \cdot \hat{\mathbf{n}} \cdot \mathcal{F}, \tag{1}$$

where $\hat{\mathbf{p}}_0$ is the location of the zero crossing on the edge with corresponding $\hat{\mathbf{n}}$ and a controls the width of the contour. If only the silhouette is required, the inner contour patches are discarded using a stencil test. We render $\mathcal{M}$ into the stencil buffer and then render the contours with the stencil test enabled, allowing the contours to be drawn only where $\mathcal{M}$ has not been drawn into the stencil buffer. To avoid terminology confusion between contours and suggestive contours, we will refer to the contours described in this section as *actual contours*.

3.3. Suggestive Contours

Suggestive contours, proposed by De Carlo et al. [DFRS03], can be understood as an elongation of actual contours to support shape perception. Suggestive contours extend actual contours, as these represent contours in nearby camera locations. If we look at $\mathcal{M}$ from location A and see a suggestive contour, this contour might be an actual contour seen from location B, where B is close to A. In order to determine the suggestive contours, $\kappa_{\mathbf{w}}$ and $D_{\mathbf{w}}\kappa_{\mathbf{w}}$ (recall Section 3.1) are required. The algorithm checks for the following constraints, based on the work by De Carlo et al., to find suggestive contour patches:

1. Search for a zero crossing of $\kappa_{\mathbf{w}}$ at point $\hat{\mathbf{p}}$ on each triangle edge.
2. Check for $0 < t_d < \frac{D_{\mathbf{w}}\kappa_{\mathbf{w}}(\hat{\mathbf{p}})}{\|w\|}$.
3. Check for $0 < \theta_c < acos\left(\frac{-\langle \mathbf{n}(\hat{\mathbf{p}}), \mathbf{v}(\hat{\mathbf{p}}) \rangle}{\|\mathbf{v}(\hat{\mathbf{p}})\|}\right)$.

Condition 2 uses t_d to ensure that $D_{\mathbf{w}}\kappa_{\mathbf{w}}$ is positive and to suppress regions with frequent changes of the sign of $D_{\mathbf{w}}\kappa_{\mathbf{w}}$ that might result from noise. The purpose of condition 3 is to discard suggestive contour lines that would appear in regions where the angle between $\mathbf{v}$ and $\mathbf{n}$ is too small. With increasing θ_c, suggestive contours are only visible the closer they are to actual contours.

If two or more edges of a triangle meet the conditions, suggestive contour patches are generated similarly to the method in Section 3.2. The suggestive contour patches are oriented orthogonally to their corresponding triangle surface. This has the effect that suggestive contour lines close to actual contours appear thicker, since they are oriented (almost) perpendicular to the view ray. Recall that actual contours appear where the surface normal and the view ray are orthogonal. In contrast, suggestive contours further away from actual contours will be drawn in a less thick style, because their orientation tilts away from the camera.

3.4. Line Search based Hatching

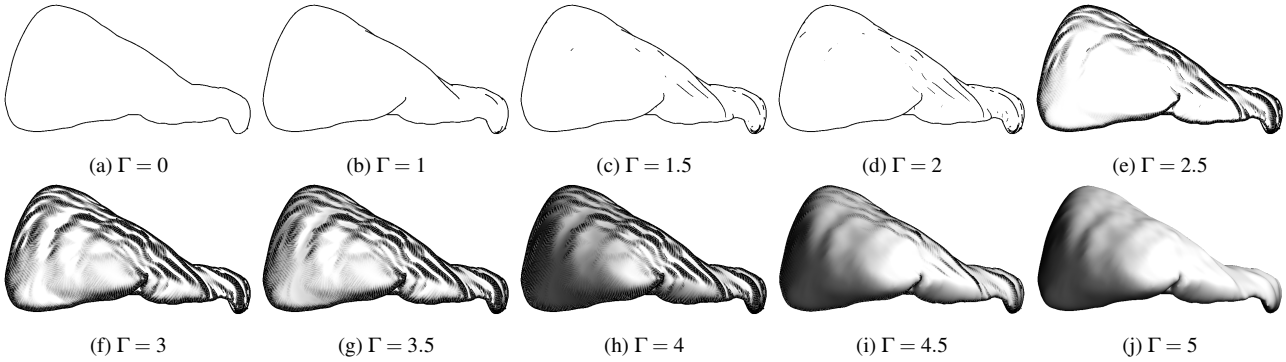
In NPR, hatching is a technique to cover a surface with a large number of line strokes that support the spatial impression of the surface. Line integral convolution (LIC) has been used to achieve hatching-like results [LKEP14], based on a LIC algorithm by Huang et al. [HPW*12]. The method by Huang et al. proposes to integrate the streamlines over a depth dependent noise texture, resulting in a very dense set of thin lines with a low black-white contrast. In our work, we are interested in more distinctive strokes with more contrast. A noise texture is not suitable for our needs, because a coarse noise texture is either blocky or blurred. Furthermore, we want to disregard texture coordinates with respect to medical data sets, because surface data generated from medical volume data does not usually feature texture coordinates. To address the goal of distinctive strokes, we suggest to map a coarse distribution of seed regions - called *smart seeds* - onto $\mathcal{M}$, from which the strokes can pick up their color. Since we are not integrating over a noise texture, we simply perform a line search, which we describe in Section 3.4.3. The following tasks arise from our requirements:

1. Map seeds onto $\mathcal{M}$ without texture coordinates (Section 3.4.1).
2. Define a function to generate seed regions from the seed locations (Section 3.4.2).
3. Compute a line search such that single strokes do not merge (Section 3.4.3).

Merging strokes occur when two strokes touch and result in a single, broadened stroke, which is undesirable. One stroke ending in the start of another stroke is still allowed, because that results in an extended stroke of equal width.

3.4.1. Seed Mapping

A solution for task 1 can be found in the polycube maps approach by Tarini et al. [THCM04]. They subdivide the space around a



(a) $\Gamma = 0$ (b) $\Gamma = 1$ (c) $\Gamma = 1.5$ (d) $\Gamma = 2$ (e) $\Gamma = 2.5$

(f) $\Gamma = 3$ (g) $\Gamma = 3.5$ (h) $\Gamma = 4$ (i) $\Gamma = 4.5$ (j) $\Gamma = 5$

Figure 2: The Sline method applied to a surface model of a clinical CT scan of the liver. Sline provides a seamless transition between various levels of abstraction, from silhouettes to contours, to suggestive contours (top row), to hatching, and finally to Phong Shading (bottom row) with intermediate states. The user parameter Γ (see Section 3.5) is given for each state.

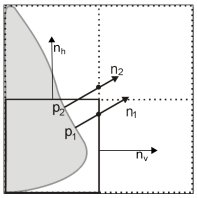


Figure 3: Projection example: points are projected along their normals.

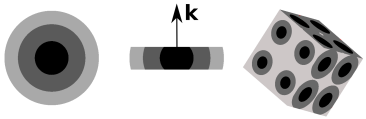


Figure 4: Core region (black), halo region (dark gray), contact region (light gray) (left), gradient of the seed center ($\mathbf{k}$) (center), and larger radii on faces turning away from the camera (right).

mesh into an isotropic grid and practically inflate the mesh until it fills a specific set of grid cells. Figure 3 depicts how the mapping of surface points to the grid faces (or edges in 2D) is done. Each point is projected along its surface normal $\mathbf{n}$. For $\mathbf{p}_1$ we can see that $\langle \mathbf{n}_v, \mathbf{n}_1 \rangle > \langle \mathbf{n}_h, \mathbf{n}_1 \rangle$, so that $\mathbf{p}_1$ is projected onto the vertical edge. In the case of $\mathbf{p}_2$ we see that $\langle \mathbf{n}_v, \mathbf{n}_2 \rangle > \langle \mathbf{n}_h, \mathbf{n}_2 \rangle$ as well, so that p_2 is projected onto the vertical edge, even though the projection crosses the horizontal edge. This concept is used analogously for the faces of a 3D grid cell. In general, a point with surface normal $\mathbf{p}_n$ is projected on face f with normal f_n, if $\langle \mathbf{p}_n, f_n \rangle$ is the maximum of the dot products of all face normals and $\mathbf{p}_n$. The fact that surface points can be projected outside their original grid cell, results in less distortion of the mapped texture. We define the grid in local coordinates of $\mathcal{M}$ and use the cell faces to generate smart seeds. Therefore, we do not need any further processing for the mapping. For each cell face, we generate a texture as shown in the left of Figure 4.

3.4.2. Smart Seeds

Our goal is to generate thick lines that do not merge with other lines. To achieve this, our seeds are made up from three different regions, namely a *core region*, a *halo region* and a *contact region* (see Figure 4).

The core region is used to produce the actual stroke and is always black, while the halo region can be adjusted in brightness to control the contrast among different strokes. The contact region will be used to prevent multiple strokes from merging. Strokes that run through the contact region of other seeds will be drawn in a lighter way, with the effect that different core strokes cannot appear in immediate contact. Due to our implementation of the line search (see Section 3.4.3), we trim the seed in the direction of the local light gradient $\mathbf{k}$ to make all regions accessible from outside (see Figure 4, center). Additionally, the radius of a seed is increased for faces that turn away from the view, to compensate for perspective distortion (see Figure 4, right and Eq. 3). We do this because small, perspectively clinched seeds contain less fragments and are thus more likely to be discarded by the rasterizer.

The computation of the smart seeds proceeds dependent on their local $\mathbf{k}$. A screen space representation of Mesh $\mathcal{M}$ is rendered into a preparation texture TEX_p. In the fragment shader, each fragment's 3D location $\mathbf{p}$ is projected based on their normal as described in Section 3.4.1 and we get the projected point $\mathbf{p}'$. $\mathbf{k}(\mathbf{p})$ is projected onto the corresponding grid cell face and yields the normalized vector $\mathbf{k}'(\mathbf{p})$. Let the center of the grid cell face be $\mathbf{c}$, the size of the cell be c_s and the seed radius be $s_r = \frac{1}{4} c_s$. c_s should be chosen dependent on $\mathcal{F}$ and the distance of $\mathcal{M}$ to the camera.

Dependent on its location, $\mathbf{p}'$ is classified as a seed region $C \in \{\texttt{Core}, \texttt{Halo}, \texttt{Contact}\}$. The boundaries of the regions are given by the distance $\|\mathbf{p}' - \mathbf{c}\|$ at $\frac{1}{2} s_r$, s_r and βs_r, where $\beta \geq 1$ is a factor to determine the contact region size, which is set to $\beta = 1.8$ in this work. The trimming of the seed is done using:

$$\begin{aligned} \mathbf{q}_1 &= \mathbf{c} + \frac{s_r \cdot \mathbf{k}'(\mathbf{p})}{4} \\ \mathbf{q}_2 &= \mathbf{c} - \frac{s_r \cdot \mathbf{k}'(\mathbf{p})}{4} \\ d_1 &= \langle \mathbf{k}'(\mathbf{p}), (\mathbf{q}_1 - \mathbf{p}') \rangle \\ d_2 &= \langle \mathbf{k}'(\mathbf{p}), (\mathbf{q}_2 - \mathbf{p}') \rangle \end{aligned} \tag{2}$$

If d_1 and d_2 have the same sign, the fragment at $\mathbf{p}$ is invalid. Using $\mathbf{k}'(\mathbf{p})$ instead of $\mathbf{k}'(\mathbf{c})$ allows the trimmed seed to adapt to the surrounding vector field $\mathbf{k}$ (see Figure 5, right). Seeds on faces that turn away from the viewer have their radius increased by

$$s_{r_m} = s_r(2 - |\langle \mathbf{v}, \mathbf{n} \rangle|), \tag{3}$$

where s_{r_m} is the modified seed radius. Figure 5 (left) shows a sec-

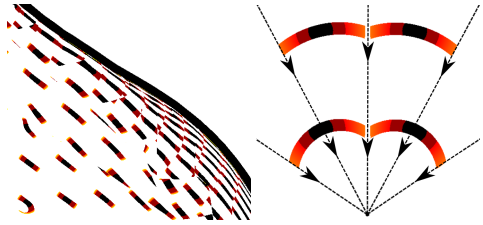


Figure 5: The seed trimming is calculated per fragment, based on the fragment's projected light gradient $\mathbf{k}'$. This aligns the valid seed region with the vector field (right). Valid regions are colored based on their properties (black to red).

tion of the resulting texture TEX_p, which includes the contour of the mesh and will be used as input for our line search computation. The last step is to encode seed region properties in the color channel of each fragments output. The red (R) color channel stores the stroke color, which is set to 0 for the core region and some value $\in [0,1[$ for the halo region (we use 0.6 for our figures). For the contact region we set R to 1, since 1 is used as the background color, e.g., no stroke should be drawn from here. To prevent different strokes from merging we assign a permeability factor ρ to each fragment and store it in the green (G) color channel. Core and halo regions are set to zero permeability, to stop any merging strokes. For the contact region we set

$$\rho = \frac{\|\mathbf{p} - \mathbf{c}'\| - s_r}{s_r \cdot \beta - s_r} \tag{4}$$

to produce a smooth gradient from outside the contact region to the edge of the halo region. The next section will describe the application of TEX_p.

3.4.3. Line Search

For any starting point on $\mathcal{M}$ our line search algorithm follows the underlying vector field $\mathbf{k}$ in positive and negative direction, until it hits a smart seed. The color for $\mathcal{M}$ at that starting point is determined by the distance traveled from the starting point to the smart seed and by the seed region that has been hit. The computation is done in the fragment shader. Each fragment has access to a texture TEX_v, containing the view projected vector field $\mathbf{k}$ and TEX_p. In contrast to common LIC algorithms we follow the vector field in both, its forward and backward direction. Another difference is that we do not integrate over the color along a line. Instead, we search for the first occurrence of a seed region to stop the iteration. Thus we use a line search method, instead of LIC. Let $\mathbf{x}$ be the position of a fragment in screen space, then $TEX_p(\mathbf{x})_{\{\mathrm{R},\mathrm{G}\}}$ gives access to the texture's color channels and $TEX_v(\mathbf{x})$ gives access to the vector field at position $\mathbf{x}$. Let $d \in [-1,1]$ define the positive or negative direction of the iteration, i be the number of iterations, τ be a dampening factor that accumulates the loss of permeability when iterating through a seed's contact region with τ_w controlling the impact of τ and λ be the step size of the iteration. $\mathcal{S}$ is the aggregate of all smart seed Core and Halo pixel locations except for pixels of the smart seed covering $\mathbf{x}$, when $\mathbf{x}$ is used as the starting position of our line search. Our algorithm to find a fragment's color in one direction of the vector field is then given by Algorithm 1.

Algorithm 1 runs for $d = -1$ and $d = 1$ and we get the result

$i_d = 0$;
$\mathbf{x}' = \mathbf{x}$;
$\tau = 1$;
while $\mathbf{x}' \notin \mathcal{S} \wedge \mathbf{x}' \neq$ *contour* **do**
  i_d++;
  $\mathbf{x}' = \mathbf{x}' + \mathbf{d}\ \lambda\ TEX_v(\mathbf{x}')$;
  $\tau = \tau\ (1 - \tau_w\ (1 - TEX_p(\mathbf{x}')_{\mathrm{G}})$;
end
$\mathbf{R_d} = \tau\ TEX_p(\mathbf{x}')_{\mathrm{R}} + (1 - \tau)$

Algorithm 1: Line search algorithm

R_d with the number of required iterations i_d. The final color for the fragment is computed as an interpolation of both results:

$$R = \frac{R_{-1}\ i_1 + R_1\ i_{-1}}{i_1 + i_{-1}} \tag{5}$$

The interpolation yields a smooth transition between core and halo regions of extendingextending strokes. Figure 6 (right) shows the result of this process. It can be observed that our method is capable of generating thick strokes with good contrast.

We do not want to draw hatching lines over the full mesh, but rather only in regions where they support the (suggestive) contour lines. A solution for this problem is presented in the next section.

3.4.4. Blending

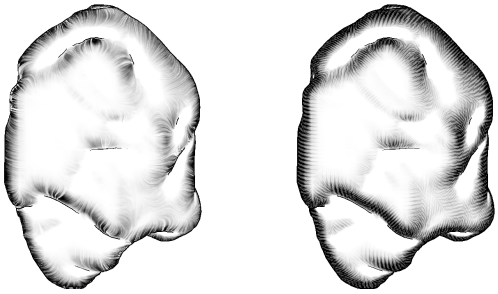
Our intention is to show the hatching lines only in regions where they support the contour and suggestive contour lines of the model. The contours basically rely on the light gradient direction $\mathbf{k}$ and on $\langle \mathbf{n}, \mathbf{v} \rangle$, while actual contours are restricted to both vectors being orthogonal and the suggestive contours allow the threshold $\theta_\mathbf{c}$. Thus, we want to find a function that takes $\mathbf{n}$, $\mathbf{v}$, $\mathbf{k}$ and θ_c and outputs a value $\mathcal{O}_h \in [0,1]$ that would be used as opacity for the hatching strokes. We define $\mathcal{O}_h$ as follows:

$$\begin{aligned} w_\mathbf{k} &= \langle \mathbf{k}, \mathbf{v} \rangle \\ w_a &= clamp_{[0,1]}\left(\frac{acos(-\langle \mathbf{n}, \mathbf{v} \rangle) - \theta_c}{\frac{1}{2}\pi - \theta_c}\right) \\ \mathcal{O}_h &= \max(w_a \cdot w_\mathbf{k} \cdot \mathcal{F}, 0) \end{aligned} \tag{6}$$

As $clamp_{[l,u]}(...)$, we define a mapping to $[l,u]$, where values $< l$ or $> u$ are set to l or u respectively. $w_\mathbf{k}$ weighs the light gradient in dependence of the view. w_a is a mapping of valid angles $\in [\theta_c, \frac{1}{2}\pi]$ to $[0,1]$.

We multiply all of these factors. Thus, either the viewing angle or the projected $\mathbf{k}$ can be responsible for eliminating hatching lines. As we disregard negative results, only regions where $\mathbf{k}$ points away from the viewer can have hatching lines, e.g., where curvature increases with distance to the camera. Note that $\mathbf{k}$ is not normalized to take the curvature magnitude into account. See Figures 2f and 6 (right) for the final result of our hatching method.

We can interpret the blending as a hint to where suggestive contours might appear in nearby camera positions and that it relates to suggestive contours in the same way as suggestive contours relate to actual contours. If we rotate the camera such that a suggestive



(a) LIC (b) Line Search

Figure 6: Comparison of the noise texture based LIC (a) and our method (b) applied to a tumor model. While the LIC method produces blurry lines, our method yields characteristic strokes.

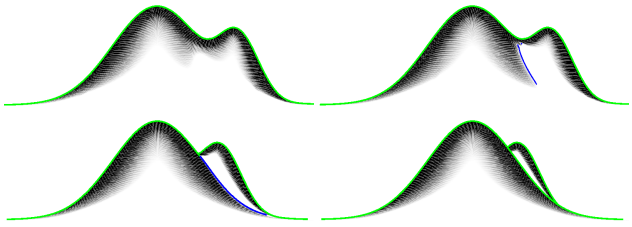


Figure 7: From top left to bottom right, the camera is rotated to the left. The suggestive contour (blue) appears nearby a hatched area and then moves along the hatched area and becomes an actual contour (green).

contour line appears where hatching was visible in the previous camera position, the line becomes more distinct and will finally transition into an actual contour. Rotating into the opposite direction leads the suggestive contour line to fade away (see Figure 7).

3.4.5. Robustness

The visual output of the presented hatching method mainly depends on the grid cell size (or seed radius) and the brightness of the halo region. The grid is defined in model space, so if we zoom out the seeds shrink in screen space. This will ultimately result in heavy flickering when moving the camera, as soon as the seeds are smaller than a pixel. In that case, seeds may or may not appear in TEX_p and so will the hatching lines that depend on them. A simple approach to reduce this kind of artifact is to increase the grid cell size with increasing distance of the camera to the object. Consequently, the hatching strokes of distant objects are bigger and fewer, in relation to the object's size. As we zoom in, the hatching becomes more detailed. Another factor that can reduce flickering is the color of the halo region. Hatching strokes of the lighter halo region color are less salient and if these pop in and out, the flickering is less severe. The curvature vector field is view dependent, because we compute the light gradient with the light source positioned in the camera center. Thus, it changes smoothly as the camera moves. This is true, except for regions where the surface normal is (almost) perpendicular to the view vector. Here, the direction of $\mathbf{k}$ can change rapidly. This also affects the hatching lines and causes visual artifacts. Using Eq. 6, these regions are discarded, since $\mathcal{O}_h = 0$ at these locations if θ_c is large enough.

3.5. Transition Parameterization

The rendering techniques described in Section 3 depend on one or more parameters. For simplicity and intuitive interaction, without requiring user knowledge about the underlying parameters, we map a single parameter to all feasible input of one shading technique. Feasibility of a parameter to be mapped is given, if that parameter can support the seamless transition among our shader stages. We will now define the mapping of our transition parameters for the different stages defined at the beginning of this section.
The user sets a parameter $\Gamma \in [0,5]$ to define the transition level. The parameters $\mathcal{T}_{\{S,C,SC,H,P\}} \in [0,1]$ describe the transition from one rendering style to the next and depend on Γ (recall Figure 2).

$$\begin{aligned} \text{Silhouettes: } \mathcal{T}_S &= 1 \\ \text{Contours: } \mathcal{T}_C &= clamp_{[0,1]}(\Gamma) \\ \text{Suggestive contours: } \mathcal{T}_{SC} &= clamp_{[0,1]}(\Gamma - 1) \\ \text{Hatching: } \mathcal{T}_H &= clamp_{[0,1]}(\Gamma - 2) \\ \text{Phong shading: } \mathcal{T}_P &= clamp_{[0,1]}(\Gamma - 3) \end{aligned} \tag{7}$$

That way, the stage in transition is modified by Γ, while all previous stages are in a fixed, visible state.

Silhouettes and Contours: According to Eq. 1, we set the width a:

$$a = width \cdot \mathcal{T}_{\{S,C\}}. \tag{8}$$

This will fade in contours from 0 to *width*, where *width* is a predefined parameter controlling the maximal width of the contour and is set to 0.006 for our examples.
Note that a is scaled by $\mathcal{F}$ in Eq. 1, so that *width* defines the contour width as a fraction of the bounding sphere of $\mathcal{M}$.

Suggestive Contours: According to Condition 2 and 3 in Section 3.3, we set t_d and θ_c:

$$\begin{aligned} t_d &= (1 - \mathcal{T}_{SC}) t_{d\,min} + t_{d\,min} \\ \theta_c &= \frac{\pi}{2} - (\frac{\pi}{2} - \theta_{c\,min}) \mathcal{T}_{SC} \end{aligned} \tag{9}$$

where $t_{d\,min}$ and $\theta_{c\,min}$ are predefined minimal thresholds. This will cause suggestive contours to grow from actual contours, by decreasing θ_c and t_d from high values to their respective minimal values.

Hatching: According to Eq. 6, we set θ_c:

$$\theta_c = \frac{1}{2}\pi(1 - \mathcal{T}_H) \tag{10}$$

This will expand the regions around contours and suggestive contours where hatching will be enabled.

Phong Shading: We want to fade in Phong shading from locations where hatching is not visible and define the Phong opacity $\mathcal{O}_p$ as:

$$\begin{aligned} \theta_c(\mathcal{T}_P) &= \frac{1}{2}\pi(1 - \mathcal{T}_P) \\ w_a(\mathcal{T}_P) &= clamp_{[0,1]}\left(\frac{acos(-\langle \mathbf{n}, \mathbf{v} \rangle) - \theta_c(\mathcal{T}_P)}{\frac{1}{2}\pi - \theta_c(\mathcal{T}_P)}\right) \\ w_{\mathbf{k}}(\mathcal{T}_P) &= -\langle \mathbf{k}, \mathbf{v} \rangle (1 - \mathcal{T}_P) - \mathcal{T}_P \\ \mathcal{O}_p &= clamp_{[0,1]}(w_a(\mathcal{T}_P)\, w_{\mathbf{k}}(\mathcal{T}_P)) \end{aligned} \tag{11}$$

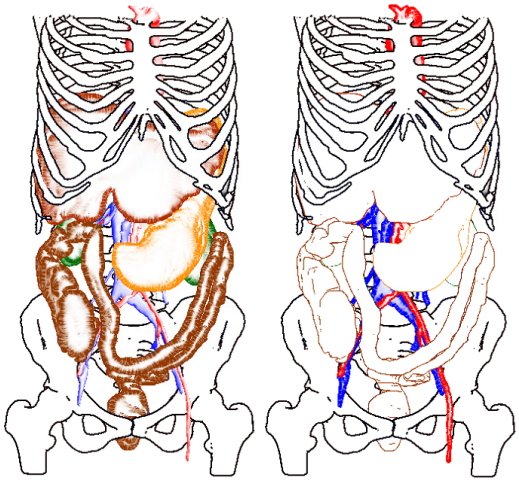


Figure 8: Abdomen dataset: by setting the rendering styles of the structures, users can shift the focus from the internal organs (left) to the vascular structures (right).

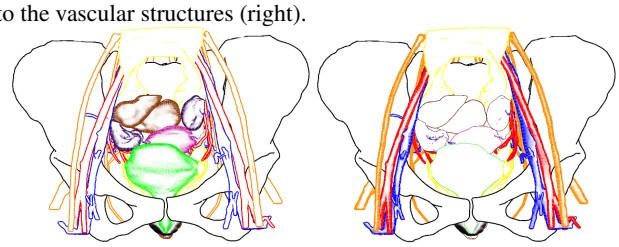


Figure 9: Pelvis dataset: hatching the organs as focus with other anatomy as context (left), and branching structures as focus (right).

$[\theta_c(\mathcal{T}_P), \frac{\pi}{2}]$ defines the range of valid angles between $\mathbf{v}$ and $\mathbf{n}$. $w_a(\mathcal{T}_P)$ will always be positive for valid angles and zero for invalid angles. $w_{\mathbf{k}}(\mathcal{T}_P)$ is only positive if $\mathbf{k}$ points towards the camera, thus $\mathcal{O}_p$ can only be positive in that case. $\theta_c(\mathcal{T}_P)$ and $w_{\mathbf{k}}(\mathcal{T}_P)$ practically result in the opposite of what we defined for valid hatching regions (recall Eq. 6) and Phong shading will first be visible in regions without hatching. With $\Gamma = 4$ we get a combined result of all shading techniques (see Figure 2h). A final transition (with $\Gamma > 4$) applies $\mathcal{T}_{\{S,C,SC,H,P\}} = 1 - clamp_{[0,1]}(\Gamma - 4)$ equally to all stages and we modify $\mathcal{O}_p$ for the Phong shading as

$$\mathcal{O}_{p_m} = \mathcal{T}_P \, \mathcal{O}_p + (1 - \mathcal{T}_P), \tag{12}$$

where $\mathcal{O}_{p_m}$ is the modified Phong opacity. At the highest parameter setting, all illustrative shading will fade away an leave simple Phong shading as the final stage.

4. Results

To evaluate the utility of our application, we had a semi-structured interview with five domain experts, based on three case studies shown in demo videos. We chose to use videos to reach as many domain experts as possible, including ones that are not in the same physical location as us, and to avoid having these experts needing to install our software. In this way, all users have the same experience and impression of the presented technique. The five experts consisted of two medical doctors, two pharmacists, and one biologist. For the case studies, we used anatomical data from the abdomen and pelvis (see Figure 8 and 9), as well as a protein dataset (see Figure 1 (left)). The abdomen dataset features surface reconstructions based on segmentations from the Visible Human cryosection and CT scan. The pelvis dataset is the Virtual Surgical Pelvis atlas [KSJ*14], which is based on cryosectional data from the Visible Korean Female and insights gained from histology studies. The protein dataset features PDB-ID 5H9N from the protein data bank (PDB), and contains the crystal structure of LTBP1 Y114A mutant in complex with leukotriene C4.

Medical experts: After seeing the pelvis case study, the first expert envisions using Sline for treatment planning of oncologic surgical procedures, e.g., cervix or prostate carcinoma, since these procedures are usually accompanied by interdisciplinary communication of different domain experts. The same applies to patient-doctor communication in such cases, as well as for teaching purposes to educate students in the spatial relations of the anatomy. According to her, patient-doctor communication is a very important field and especially operations are very difficult to explain, because there are no information sheets. She stated that patients do appreciate when physicians explain complicated operations to them, and Sline can definitely help there. A function to export user defined 3D scenes and annotations, with certain structures highlighted, would improve the usability of Sline for communication with students.

The second expert said that the tool looks very intuitive, quick and easy to use. She estimates she could set up a visualization for the whole Virtual Surgical Pelvis dataset, featuring 28 anatomical structures, within ten minutes. She appreciates the schematic and simplified representations that can be made using Sline, and sees potential for use in treatment planning, education and doctor-patient communication. Furthermore, she proposes to use it in interdisciplinary communication, in presentations to peers for instance. She states that the combination of rendering styles and colors can really help focus attention of the viewer, but care must be taken to pick the right colors for the right structures.

Pharmacy/biology experts: Experts from the pharmaceutical domain see a potential to quickly highlight different protein and ligand configurations for comparison. This requires the overlay of multiple 3D objects, to make structural differences visible. With our technique, different structures can be quickly set to focus or context, e.g., to flexibly support a presentation in interdisciplinary communication or for teaching purposes. The size and shape of docking cavities in the protein, where ligands can bind to, are also important information and would preferably be highlighted separately, which is not possible at this time, since we set Γ per complete mesh.

The biology expert stated that Sline helped him to remedy visual overload by reducing the amount of detail for uninteresting parts of the model, while still keeping enough details to see the shape of the context. By using a medium level of abstraction in combination with color, interesting objects can be shown in a clear way. The color also provides a highlight, which makes it easier to focus on certain objects. He states the main advantage of Sline is the simplicity and ease of use, and having just one parameter to obtain the desired level of abstraction. He commented that the hatching-like effect seemed to be a bit too strong, since it creates darker images than the local lighting at the end. An additional, hierarchical list of groups of available structures contained in the scene would help to

select objects of interest even quicker. Besides the current application domain, he believes Sline could also be useful for preparation of illustrative renderings for black-and-white print.

Case study conclusion: All experts agreed that Sline can help focus the viewer's attention on important structures, and that the ordering of decreasing abstraction levels is intuitive. They also affirm that creating a meaningful visualization, based on the available options, is intuitive, easy and fast, which is credited to the single parameter concept.

5. Discussion

Our new hatching algorithm is inspired by LIC, but is capable of producing more characteristic strokes, as illustrated in Figure 6. To be consistent with the suggestive contours, it is based on the light gradient $\mathbf{k}$ and the radial curvature $\kappa_{\mathbf{w}}$. Since the light gradient changes if the camera position changes, the hatching technique has to be computed in real time. We achieved interactive frame rates for scenes of up to $800k$ faces on an NVIDIA GTX 580 graphics card using smart seeds and a simple line search. To address clinical data, the hatching does not require texture coordinates. Another positive property of our hatching is, that it relates to suggestive contours such as suggestive contours relate to actual contours (see Section 3.3). E.g. the hatching represents locations of suggestive contours in nearby camera locations. As Figure 10 shows, we can easily reduce the hatching to a stippling effect.

Our hatching concept has several limitations. In Section 3.4.5, we mentioned the flickering that we were not able to fully suppress. Also, Figure 5 reveals, that the simple seed mapping approach is not optimal. It can be observed, that some seeds are not drawn completely and are cut off at triangle edges. We assume that this is due to the ratio between grid cell size and the seed radius, which is not optimal. Another visual artifact appears when zooming in and out as we adjust the grid cell size. Doing so leads the seeds to move over the object, because they are bound to their grid cell. This effect could be eliminated, if we would allow multiple seeds per grid cell face. From a far distance, only a few large seeds should be visible. As the camera moves closer, the large seeds would shrink and new smaller seeds would fade in. We expect this to result in a seamless transition from a coarse to fine hatching.

One more drawback is illustrated in Figure 11. The hatching strokes do neither merge, nor do they extend each other. The result is a region of many short strokes that give the impression of a stippling technique. This happens due to the uniform grid and the radius that we use for the mapping of our seeds. In the presented case, a stroke can not penetrate the contact region of neighboring strokes and is prevented from merging with strokes further away. Contrarily, the reduction of the impact of the contact region might allow other strokes to merge. Imagine the seeds in Figure 11 were larger, then the strokes would probably merge in vertical direction and produce a thick, vertical hatching line. This would give a false impression of the underlying vector field. The issue might be addressed by a more sophisticated line search algorithm, which does not only consider single pixels on a line, but also pixel neighborhoods. We hope that such an approach will also reduce the flickering effect while rotating an object. Our feature size parameter $\mathcal{F}$ is based on the approximation of $\mathcal{M}$ as a sphere, which is feasible for compact structures. More filigree structures, such as vessels, would profit of a more precise approach, that incorporates the average surface curvature of $\mathcal{M}$, for example.

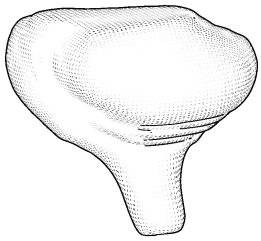


Figure 10: Stippling applied to a clinical MRI-based bladder model, using only one iteration of Algorithm 1.

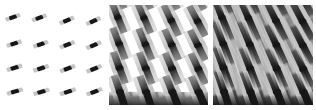


Figure 11: From left to right: Result of increasing number of iterations in Algorithm 1. Single strokes are prevented from merging with others, but also fail to extend other strokes.

6. Conclusion and Future Work

We have presented Sline, a technique for seamless interactive transitioning along various levels of abstraction, from silhouettes to Phong shading, for biomedical surface data. The transition among the involved rendering states is reduced to a single parameter, to allow for simple interaction without any additional training for the user. For this, we defined a set of parameters that can be mapped to that single input parameter and thus modify the individual rendering techniques. By adjusting this single parameter, users can quickly and intuitively generate focus-and-context visualizations for biomedical surface data.

As future work, we would like to explore smart visibility techniques to deal with occlusion, while still providing shape cues. In its current state, our framework only supports simple blending to suggest transparency of multiple objects. A proper use of order independent transparency, like the approximating algorithm by Vasilakis [VF14], could open up more areas of application and flexibility for our framework to deal with occlusion issues. For instance, the biology experts suggested to highlight cavities, which is done by a transparency mapping in the work by Borland [Bor11]. It could be of advantage to apply this transparency mapping, or any other mapping of features, on our levels of abstraction, such that regions of interest are highlighted based on their functionality.
From the usability point of view, the simplicity of parameterization presented here, can be utilized to conduct a large scale evaluation on which rendering techniques users prefer to create their own focus and context scenes. Results from such an evaluation might be used to automate the creation of focus-and-context scenes. We plan to release Sline as an open source tool, so that it becomes freely available for people to use in their own biomedical applications.

Acknowledgements This project was partly funded by the DFG: LA 3855/1-1 and HA 7819/1-1. We would like to thank the domain experts D. Komm, M. Krone, D. Imhof, P. Heimer and A. Kraima for their support.

References

[BMGS13] BORN S., MARKL M., GUTBERLET M., SCHEUERMANN G.: Illustrative Visualization of Cardiac and Aortic Blood Flow from 4D MRI Data. In *IEEE Pacific Visualization* (2013), pp. 129 –136. 3

[Bor11] BORLAND D.: Ambient occlusion opacity mapping for visualization of internal molecular structure. *Journal of WSCG 19*, 1 (2011), 17–24. 9

[BSW08] BELKIN M., SUN J., WANG Y.: Discrete laplace operator on meshed surfaces. In *Proceedings of Symposium on Computational Geometry* (2008), ACM, pp. 278–287. 2

[CCG*08] CHU A., CHAN W.-Y., GUO J., PANG W.-M., HENG P.-A.: Perception-aware depth cueing for illustrative vascular visualization. *BioMedical Engineering and Informatics, International Conference on 1* (2008), 341–346. 3

[DCLK03] DONG F., CLAPWORTHY G. J., LIN H., KROKOS M. A.: Nonphotorealistic rendering of medical volume data. *IEEE Computer Graphics and Applications 23*, 4 (2003), 44–52. 3

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *Proceedings of SIGGRAPH* (2003), 848–855. 2, 4

[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. 3

[Her99] HERTZMANN A.: Introduction to 3D Non-Photorealistic Rendering : Silhouettes and Outlines. *Non-Photorealistic Rendering. SIGGRAPH 99* (1999), 1–14. 4

[HPW*12] HUANG J., PEI W., WEN C., CHEN G., CHEN W., BAO H.: Output-coherent image-space lic for surface flow visualization. In *Visualization Symposium (PacificVis), IEEE Pacific* (2012), pp. 137–144. 4

[HWR*10] HANSEN C., WIEFERICH J., RITTER F., RIEDER C., PEITGEN H.-O.: Illustrative visualization of 3d planning models for augmented reality in liver surgery. *Int J Comput Assist Radiol Surg 5*, 2 (2010), 133–141. 3

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of SIGGRAPH* (2000), pp. 517–526. 3

[IFP95] INTERRANTE V., FUCHS H., PIZER S.: Enhancing transparent skin surfaces with ridge and valley lines. *IEEE Transactions on Visualization and Computer Graphics* (1995), 52–59. 2, 3

[JBB*08] JAINEK W. M., BORN S., BARTZ D., STRASSER W., FISCHER J.: Illustrative hybrid visualization and exploration of anatomical and functional brain data. *Computer Graphics Forum 27*, 3 (2008), 855–862. 3

[JDA07] JUDD T., DURAND F., ADELSON E.: Apparent ridges for line drawing. In *Proceedings of SIGGRAPH* (2007), p. 19. 2, 4

[KSJ*14] KRAIMA A., SMIT N., JANSMA D., WEST N., QUIRKE P., RUTTEN H., VILANOVA A., VAN DE VELDE C., DERUITER M.: 62. the virtual surgical pelvis: A highly-detailed 3D pelvic model for anatomical education and surgical simulation. *European Journal of Surgical Oncology 40*, 11 (2014), S32. 8

[KYYL08] KIM Y., YU J., YU X., LEE S.: Line-art illustration of dynamic and specular surfaces. vol. 27, pp. 156:1–156:10. 3

[LKEP14] LAWONN K., KRONE M., ERTL T., PREIM B.: Line integral convolution for real-time illustration of molecular surface shape and salient regions. *Computer Graphics Forum 33(3)* (2014), 181–190. 3, 4

[LLPH15] LAWONN K., LUZ M., PREIM B., HANSEN C.: Illustrative Visualization of Vascular Models for Static 2D Representations. *Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI) 9350*, Pt 2 (2015), 399–406. 3

[LMP13] LAWONN K., MÖNCH T., PREIM B.: Streamlines for illustrative real-time rendering. *Computer Graphics Forum 32(3)* (2013), 321–330. 3

[LP15] LAWONN K., PREIM B.: Feature lines for illustrating medical surface models: Mathematical background and survey. *Visualization in Medicine and Life Sciences III* (2015), to appear. 3

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *Proceedings of SIGGRAPH 23* (2004), 609–612. 2

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-time hatching. In *Proceedings of SIGGRAPH* (2001), pp. 579–584. 3

[PRV13] PARULEK J., ROPINSKI T., VIOLA I.: Seamless Visual Abstraction of Molecular Surfaces. In *Spring Conference* (2013), ACM, pp. 107–114. 3

[RHD*06] RITTER F., HANSEN C., DICKEN V., KONRAD O., PREIM B., PEITGEN H.-O.: Real-time illustration of vascular structures. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 877–884. 3

[SEA09] SVAKHINE N. A., EBERT D. S., ANDREWS W. M.: Illustration-Inspired Depth Enhanced Volumetric Medical Visualization. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (2009), 77–86. 3

[SEI10] SVETACHOV P., EVERTS M. H., ISENBERG T.: DTI in context: Illustrating brain fiber tracts in situ. *Computer Graphics Forum 29*, 3 (2010), 1023–1032. 3

[TCM06] TARINI M., CIGNONI P., MONTANI C.: Ambient occlusion and edge cueing for enhancing real time molecular visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1237–1244. 3

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *ACM transactions on graphics (TOG) 23*, 3 (2004), 853–860. 4

[TIP05] TIETJEN C., ISENBERG T., PREIM B.: Combining silhouettes, surface, and volume rendering for surgery education and planning. In *Computer Graphics Forum* (2005), pp. 303–310. 3

[vdZLBI11] VAN DER ZWAN M., LUEKS W., BEKKER H., ISENBERG T.: Illustrative molecular visualization with continuous abstraction. *Computer Graphics Forum 30*, 3 (2011), 683–690. 3

[VF14] VASILAKIS A. A., FUDOS I.: k+-buffer: Fragment synchronized k-buffer. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2014), pp. 143–150. 9

[Web09] WEBER J. R.: Proteinshader: illustrative rendering of macromolecules. *BMC Structural Biology 9:19* (5 2009). 3

[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

[ZHS10] ZHANG L., HE Y., SEAH H. S.: Real-time computation of photic extremum lines (PELs). *The Visual Computer 26*, 6-8 (2010), 399–407. 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

[ZISS04] ZANDER J., ISENBERG T., SCHLECHTWEG S., STROTHOTTE T.: High quality hatching. *Computer Graphics Forum 23*, 3 (2004), 421–430. 3