Volume 0 (1981), Number 0 pp. 1–26

A Survey of Surface-Based Illustrative Rendering for Visualization

Kai Lawonn,[1] Ivan Viola,[2] Bernhard Preim,[3] Tobias Isenberg [4]

[1]University of Koblenz-Landau, Germany; [2]TU Wien, Austria; [3]University of Magdeburg, Germany; [4]Inria, France

Abstract

In this article we survey illustrative rendering techniques for 3D surface models. We first discuss the field of illustrative visualization in general and provide a new definition for this sub-area of visualization. For the remainder of the survey we then focus on surface-based models. We start by briefly summarizing the differential geometry fundamental to many approaches and discuss additional general requirements for the underlying models and the methods' implementations. We then provide an overview of low-level illustrative rendering techniques including sparse lines, stippling and hatching, and illustrative shading, connecting each of them to practical examples of visualization applications. We also mention evaluation approaches and list various application fields, before we close with a discussion of the state of the art and future work.

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

1. Introduction

Modern rendering and visualization techniques offer us a multitude of possibilities to convert 3D datasets into visual representations. The subgroup of illustrative visualization techniques, however, focuses specifically on learning from and/or being inspired by centuries of experience of scientific illustration [Hod03]. Such techniques aim to provide meaningful, expressive, and sometimes simplified representations of a problem, a scene, or a situation. For example, illustrative techniques can introduce deliberate abstraction [VI18] to reduce visual disorder, allowing viewers to focus their attention on one or more regions of interest in the visualization (e. g., Figure 1).

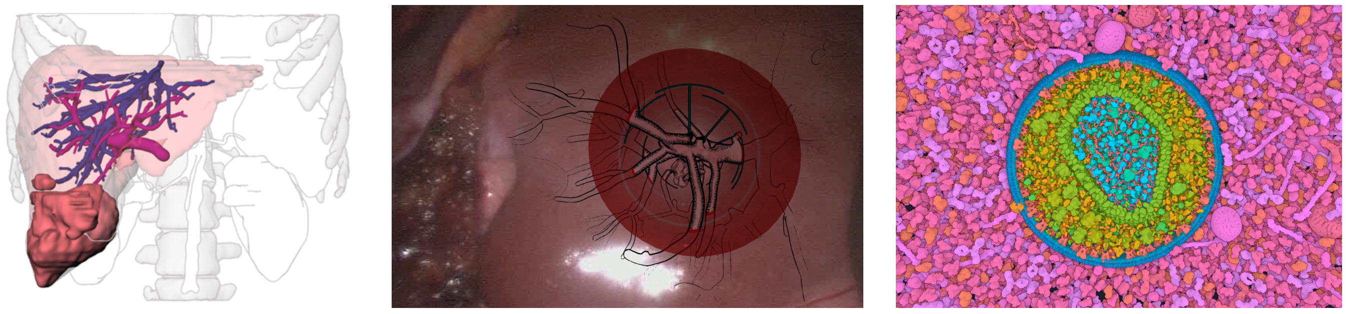
Illustrative visualization techniques are often applied to medical applications and the generation of textbook material, but applications in the natural sciences and engineering have also been discussed. However, while the previous definition of illustrative visualization [RBGV08] related it to traditional fine arts, it lacked a clear distinction from the general field of visualization. We thus begin our discussion by proposing a definition, based on past discussions of its goals in the literature, that reinforces the unique characteristics of the discussed sub field.

We then provide a structured overview of this subfield of visualization by surveying the approaches that are considered to generate illustrative visualizations. Our goal is to provide researchers and practioners with a guide to select appropriate techniques for their problem at hand. Yet, the field is still rather large and encompasses many visualization methods. In this survey we thus focus on illustrative rendering techniques for 3D surface shapes. Specifically, we contribute:

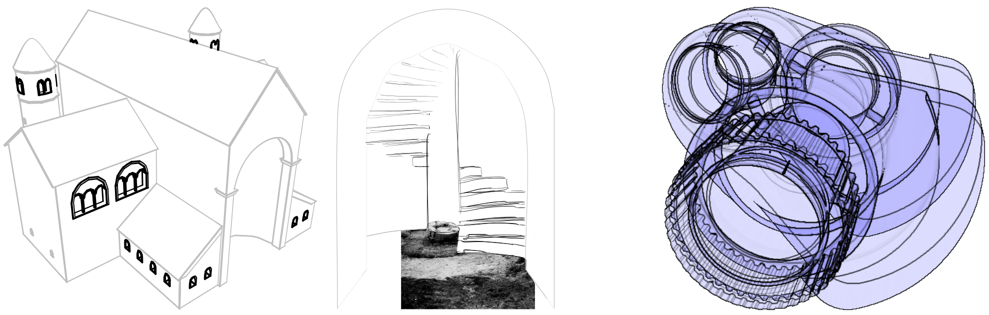
- a definition of illustrative visualization based on the literature,
- an overview of illustrative visualization concepts, and
- a discussion of open problems in illustrative visualization and a perspective on future research directions.

Paper Selection. We searched for relevant papers to include in this survey on the EG Digital Library, IEEE Xplore, and the ACM digital library. Furthermore, we used Google Scholar to identify additional research, in particular older papers. We looked for the following keywords and combinations thereof: *illustrative*, *NPR*, *non-photorealistic rendering*, *silhouettes*, *contours*, *feature lines*, *hatching*, *stippling*, *line drawings*, *focus and context*, *simplification*, *abstraction*, and *low-level visualization*.

Organization. In SECTION 2 we explain the concept of illustrative visualization techniques and attempt a definition. In SECTION 3 we then provide the background that is necessary to implement the surveyed illustrative visualization techniques. This section covers the basics of differential geometry, its application to surface meshes, and we discuss requirements that need to be fulfilled for most of the rendering methods. In the following SECTION 4 we present an overview of low-level illustrative visualization techniques. We introduce the most commonly used rendering styles, ordered according to their level of abstraction. Afterwards, in SECTION 5 we describe evaluation techniques that exemplify how these illustration techniques can be assessed and list typical application areas in SECTION 6. Based on previous discussions, we then analyze the state of



(a) Focus+context illustration [TIP05]. (b) Projection in the OR: liver with vessel [LLH17]. (c) Illustration in structural biology [LMAPV15].



(d) Illustration in archeology, including hybrid depiction [SMI99]. (e) Technical illustration [ND04].

Figure 1: *Examples of the use of illustrative visualization techniques in various application contexts.*

the art in SECTION 7 and outline unsolved problems and challenges for future research. Finally, we conclude with a brief summary in SECTION 8. This survey is an extension of parts of the PhD thesis by Lawonn [Law14].

2. Concept of Illustrative Visualization and Survey Scope

Visualization is a method to "convey salient information about [...] data" by means of visual encoding [HJ05], ultimately to "amplify cognition" [CMS99]. Over the past decades, visualization researchers have been successful in finding numerous ways to effectively represent data aspects visually in a wide variety of application domains. The field of visualization shares, however, the goal of the communication of information and insights about complex phenomena with the field of traditional illustration. Consequently, as the necessary computational and representational tools were developed within computer graphics,[†] visualization researchers took inspiration from traditional illustration to be able to generate *illustrative visualizations* [BCP*12, Ise15, Ise16, LM02, RBGV08, VHE10].

In addition to this inspiration from the traditional craft, illustrative visualization is characterized by abstraction and simplification [BHI*05, Ise16, LMP13, LP16, RBGV08, VHE10, VI18], by emphasizing the important/relevant and suppression of the context [BHI*05, LMP13], thus leading to clearer [BCP*12], more expressive [BCP*12, LP16, LM02, VHE10] depiction, and consequently more effective visualization [RBGV08, VHE10]. Thus, it allows visualization creators to use more visual variables [Ise15]and several layers of information [Ise15, Ise16], reduces clutter and occlusion [BCP*12], and improves the perception of shape and depth [BCP*12, Ise15, PBC*16]. It thus also relies on insights from perception and cognition [BHI*05, VHE10], ultimately to show the relevant information [BHI*05], to improve the way people understand what they see [BCP*12, HWR*10, LP16], explore data [BCP*12, Ise16], gain knowledge from the visualization [BCP*12, Ise16, VHE10], and communicate insights from data exploration [Ise16, VHE10]. Based on these notions, we can thus provide the following definition:

Definition 1 An illustrative visualization is a visualization technique for measured, simulated, and modeled data that—inspired by traditional illustration—uses established illustration principles such as abstraction and emphasis, layered organization, dedicated use of multiple visual variables, or the support of perception of depth and shape—with the goal to create more effective, more understandable, and/or more communicative data representations than is possible with other visualization techniques.

Nonetheless, the question remains whether illustrative visualization is a "new technology," or simply a "tautology" [RBGV08] since it is simply a characteristic of 'good visualization'—after all, its goals and approaches are arguably shared with many if not most visualization techniques. Rautek et al. [RBGV08] thus answer that it is the latter, arguing that illustrative visualization will ultimately be an aspect of virtually all visualization work. While we generally agree with this notion, we still point out that seeing illustrative visu-

[†] Specifically as part of the computer graphics sub-field of non-photorealistic rendering [GG01, SS02, KCWI13, RC13].

alization as only a “useless tautology” [RBGV08] is selling it short. The focus on a dedicated reflection of principles that have long been established in traditional visualization and the exploration of how they are best applied to visualizations of “real” data embedded in a problem-solving situation allows illustrative visualization to ultimately advance the field of visualization as a whole.

It is thus essential to understand the state of the art of the field. Several surveys have already summarized the research for subdomains of visualization including flow visualization [BCP*12] and the visualization of brain connectivity [Ise15], provided a general tutorial [VGH*05], as well as recently reflected in general on abstraction [VI18]. In this state-of-the-art report we instead focus specifically on the rendering part for visualization of surface-based data as it results, for example, from iso-suface extraction (e. g., in physical simulations), from segmentation (e. g., in medical datasets), or from dedicated surface models (e. g., in the visualization technical objects and processes). We thus describe, in particular, the extraction and application of sparse line illustrations, surface-filling marks, and illustrative surface shading.

3. Background

This section provides a brief discussion of discrete differential geometry and other prerequisites. This background is necessary for many of the illustrative rendering techniques of surface geometries and allows the interested reader to look up the terms and principles that are essential for successful implementations of the presented methods. We introduce basic terms and explain specific mathematical concepts such as curvature and directional derivatives.

3.1. Discrete Differential Geometry

In this section, we provide an introduction on how the operators and measures from continuous differential geometry can be adapted to polygonal meshes. We use the following notation in the remainder of this paper. Let $M \subset \mathbb{R}^3$ be a triangulated surface mesh with vertices $i \in V$ and its associated positions $\mathbf{p}_i \in \mathbb{R}^3$, edges $E = \{(i,j) \,|\, i,j \in V\}$, and triangles $T = \{(i,j,k) \,|\, (i,j),(j,k),(k,i) \in E\}$. We write $\mathbf{n}_i$ as the normalized normal vector at vertex i. If nothing else is mentioned, we refer to normal vectors at vertices. Furthermore, $\mathscr{N}(i)$ denotes the neighbors of i such that $(i,j) \in E$ holds for every $j \in \mathscr{N}(i)$. Furthermore, if we use a triangle for calculation, we always use this notation: given a triangle $t = (i,j,k)$ with vertices $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k$, its edges are defined as $\mathbf{e}_{ij} = \mathbf{p}_i - \mathbf{p}_j$.

3.1.1. Voronoi Area

Generally, the important discrete differential geometry measures can be determined for every triangle. Afterwards, the question arises of how to compute the measures for every vertex. Intuitively, one idea is to use measures of the incident triangles and weight them based on the underlying triangle. One way to weight them is by using the Voronoi area. Given is a triangle $t = (i,j,k)$, which is divided in three regions $\mathscr{R}_i, \mathscr{R}_j, \mathscr{R}_k$ with $\mathbf{p}_i \in \mathscr{R}_i, \mathbf{p}_j \in \mathscr{R}_j, \mathbf{p}_k \in \mathscr{R}_k$. The regions are called *Voronoi regions*, if for every point $\mathbf{p}$ inside the triangle and $\mathbf{p} \in \mathscr{R}_m$ with $m \in \{i,j,k\}$ follows that $\|\mathbf{p} - \mathbf{p}_m\| \leq \|\mathbf{p} - \mathbf{p}_{\{i,j,k\}\backslash m}\|$. The area of the Voronoi region can be computed according to Meyer et al. [MDSB02] as:

$$\mathscr{A}_t(i) = \begin{cases} \frac{\|\mathbf{e}_{ji} \times \mathbf{e}_{ki}\|}{4} & \text{if triangle } t \text{ is obtuse at } \mathbf{p}_i \\ \frac{\|\mathbf{e}_{ji} \times \mathbf{e}_{ki}\|}{8} & \text{if triangle } t \text{ is obtuse} \\ \frac{1}{8}\left(\|\mathbf{e}_{ij}\|^2 \cdot \cot(\mathbf{e}_{jk}, \mathbf{e}_{ki}) + \|\mathbf{e}_{ki}\|^2 \cdot \cot(\mathbf{e}_{ij}, \mathbf{e}_{jk})\right) & \text{otherwise.} \end{cases} \quad (1)$$

3.1.2. Normals

The normals can be calculated for every triangle as well as for every vertex. For the triangle $t = (i,j,k)$, the normal is the cross product of the edges: $\mathbf{n}_t = \frac{\mathbf{e}_{ji} \times \mathbf{e}_{ki}}{\|\mathbf{e}_{ji} \times \mathbf{e}_{ki}\|}$. Note that it must be assured that the orientation is consistent in the surface mesh. The normal for every vertex can be obtained by a weighted accumulation on the incident triangle normals:

$$\mathbf{n}'_i = \sum_{t \in T, i \in t} \omega_t \mathbf{n}_t.$$

The natural choice to use $\omega_t = 1$ or $\omega_t = area(t)$ leads to unexpected behaviour in the shading. Alternatives are to use the Voronoi area $\omega_t = \mathscr{A}_t(i)$ or the angle $\omega_t = \angle(\mathbf{e}_{ji}, \mathbf{e}_{ki})$ which both work well. The normal at i still needs to be normalized as follows: $\mathbf{n}_i = \frac{\mathbf{n}'_i}{\|\mathbf{n}'_i\|}$.

3.1.3. Discrete Curvature

The calculation of the principal curvature directions and their curvatures can be carried out by fitting higher-order polynomials to the mesh [CP05, GI04] or by calculating the normal curvatures along the edges and then estimating the shape operator [CS92, HS03, MDSB02, PKS*01, Tau95a]. We focus on another category of methods that estimate the shape operator directly [ACSD*03, CSM03, Rus04, HP11], see also Váša et al. [VVP*16] for an analysis of curvature estimations. In this section, we provide the curvature estimation according to Rusinkiewicz [Rus04]. First, we determine the shape operator for every triangle and then for every vertex. We thus first need to create an orthonormal basis for each triangle. Given the triangle $t = \{i,j,k\}$, the basis $(\mathbf{x}_t, \mathbf{y}_t)$ is constructed by:

$$\mathbf{x}_t := \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|}, \quad \mathbf{y}_t := \frac{\mathbf{x}_t \times (\mathbf{e}_{kj} \times \mathbf{x}_t)}{\|\mathbf{x}_t \times (\mathbf{e}_{kj} \times \mathbf{x}_t)\|}. \quad (2)$$

The property that the shape operator yields the change of the normal in a certain direction leads to the following equation system:

$$S\begin{pmatrix} \langle \mathbf{x}_t, \mathbf{e}_m \rangle \\ \langle \mathbf{y}_t, \mathbf{e}_m \rangle \end{pmatrix} = \begin{pmatrix} \langle \mathbf{x}_t, \mathbf{n}_m \rangle \\ \langle \mathbf{y}_t, \mathbf{n}_m \rangle \end{pmatrix}, \quad (3)$$

with $m \in \{(i,j),(j,k),(k,i)\}$ and $\mathbf{n}_{ij} = \mathbf{n}_i - \mathbf{n}_j$. This approach results in an overdetermined system, which can be approximated by the least square method. Next, we need to calculate S for each vertex of the mesh. Given an orthonormal basis $(\mathbf{x}_i, \mathbf{y}_i)$ for the vertex i, we first rotate the basis $(\mathbf{x}_t, \mathbf{y}_t)$ of the incident triangle such that the basis is coplanar with vertex' basis. Given the shape operator in the triangle basis $S_t = \begin{pmatrix} e_t & f_t \\ f_t & g_t \end{pmatrix}$, the matrix elements can be obtained by multiplying a combination of the 2D basis vectors $(1,0),(0,1)$ from left and right, e.g., $f_t = (1\ 0)S\binom{0}{1}$. The matrix entries in the new basis $(\mathbf{x}_i, \mathbf{y}_i)$ are given by

$$e_i = \begin{pmatrix} \langle \mathbf{x}_t, \mathbf{x}_i \rangle \\ \langle \mathbf{y}_t, \mathbf{x}_i \rangle \end{pmatrix}^T S \begin{pmatrix} \langle \mathbf{x}_t, \mathbf{x}_i \rangle \\ \langle \mathbf{y}_t, \mathbf{x}_i \rangle \end{pmatrix}.$$

The other entries are determined by analogous calculations. For every incident triangle of a vertex, we thus re-express the shape operator with the vertex' basis. We then weight the shape operator by the Voronoi area of its corresponding triangle and sum up the shape operators. Finally, we divide the resulting shape operator by the sum of the weights, yielding a shape operator for every vertex. The eigenvalues and eigenvectors of the shape operator are the curvatures and principal curvature directions, respectively.

3.1.4. Discrete Gradient

In this section, we adapt the concept of the gradient to calculate it on surface meshes. First, we start to determine the gradient for every triangle and then we compute it for every vertex. First, we consider a scalar field φ on the surface that gives a value for every vertex i: $\varphi(i)$. For simplification, we write $\varphi_i := \varphi(i)$. The gradient of the triangle $t = \{i, j, k\}$ can now be determined by either constructing a basis for t and then building a linearized 2D scalarfield that coincides with $\varphi_i, \varphi_j, \varphi_k$ at the position of the vertices according to the basis. The scalar field's gradient can then be calculated in a straightforward way to yield a 3D vector in $\mathbb{R}^3$ on the triangle:

$$\nabla\varphi_t = (\varphi_j - \varphi_i)\frac{(\mathbf{p}_j - \mathbf{p}_i)^{\perp}}{2A_{\triangle}} + (\varphi_k - \varphi_i)\frac{(\mathbf{p}_k - \mathbf{p}_i)^{\perp}}{2A_{\triangle}}, \qquad (4)$$

where $A_{\triangle}$ denotes the area of the triangle and $\perp$ stands for a counter-clockwise rotation by 90° in the triangle plane, see Botsch et al. [BKP*10]. The gradient per vertex is determined by transforming the basis of incident triangles to the basis of the vertex tangent space. Then, the gradients are weighted according to the Voronoi area of the associated vertices. Finally, the accumulated vector is divided by the sum of weights, recall Section 3.1.3.

3.1.5. Discrete Laplace-Beltrami Operator

The Laplace-Beltrami operator is required for a particular feature line method, i. e., Laplacian Lines, and we thus briefly introduce it next. For a more comprehensive overview of discrete Laplace-Beltrami operators which are commonly used in graphics, however, we refer to Sorkine's [Sor05] and Patané's [Pat16] state of the art reports. In general, the Laplace-Beltrami operator of a scalar field φ on surface meshes can be written by:

$$\Delta\varphi_i = \sum_j w_{ij}\left(\varphi_j - \varphi_i\right). \qquad (5)$$

Different weights lead to different Laplace operators [WMKG07]. For the most part, the weights only operate on the neighbors. In the following, we introduce the commonly used weights.

Combinatorial: The combinatorial Laplace-Beltrami operator is determined by:

$$w_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Uniform: Taubin [Tau95b] proposed the uniform Laplace-Beltrami operator whose weights are determined by $\mathbf{p}_i$'s neighbor count:

$$w_{ij} = \begin{cases} \frac{1}{\mathcal{N}(i)}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Floater's mean value: Floater [Flo03] suggested to use the mean value as a weight:

$$w_{ij} = \begin{cases} \frac{\tan(\delta_{ij}/2)+\tan(\gamma_{ij}/2)}{\|\mathbf{p}_i - \mathbf{p}_j\|}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Cotangent weights: MacNeal [Mac49] proposed to use an average of cotangents as a weight:

$$w_{ij} = \begin{cases} \frac{\cot(\alpha_{ij})+\cot(\beta_{ji})}{2}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise.} \end{cases}$$

Belkin weights: Belkin [BSW08] determined the weights over the whole surface:

$$\Delta\varphi_i = \frac{1}{4\pi h^2(\mathbf{p}_i)} \sum_{\triangle_k} \frac{A(\triangle_k)}{3} \sum_{j \in \triangle_k} e^{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{4h(\mathbf{p}_i)}} \left(\varphi_j - \varphi_i\right),$$

where $A(\triangle_k)$ denotes the area of the triangle $\triangle_k$ and h corresponds intuitively to the size of the neighborhood.

3.2. Isolines on Surface Meshes

For *feature lines*, an important category of illustrative visualization techniques (see Section 4.2), it is usually necessary to extract lines from an underlying scalar field on the surface mesh. The location of the lines are given by the zero-crossing of the scalar field φ, i.e., the loci of points $\mathbf{p}$ with $\varphi(\mathbf{p}) = 0$. To obtain a more tessellation-independent result, it is essential to not restrict the lines to the mesh's edges. Instead the lines should be drawn such that they can also intersect a triangle. This can be achieved by checking, for every triangle, the sign of the scalar field with respect to the corresponding vertices [HZ00]. If only one sign is positive (and the rest is negative) or only one sign is negative (and the rest is positive) then a zero-crossing inside the triangle occurs. First, we determine the position of the zero-crossing on the two edges and then we connect them with a line. Let φ_i be the value at vertex i, which is the only one that is negative (or positive). The zero-crossing on the edge $\mathbf{e}_{ij}$ is determined by first determining α for which $\varphi_i + \alpha(\varphi_j - \varphi_i) = 0$ is fulfilled: $\alpha = \frac{\varphi_i}{\varphi_i - \varphi_j}$ and then plugging this into $\mathbf{p}_i + \alpha(\mathbf{p}_j - \mathbf{p}_i)$, which yields the position on the edge:

$$\mathbf{p}' = \mathbf{p}_i + \frac{\varphi_i}{\varphi_i - \varphi_j}(\mathbf{p}_j - \mathbf{p}_i) \qquad (6)$$

where the zero-crossing occurs. An analogous calculation can be carried out on the other edge, and then both points are connected.

3.3. General Requirements for Illustrative Visualization

The illustrative visualization of surface meshes leads to several crucial requirements that need to be met to achieve the high-quality results typically expected from illustration-like visuals. In particular, the following requirements need to be fulfilled: smoothing meshes, robustness, filtering the result, and frame-coherent behavior.

Smoothing. Most illustrative visualization techniques use higher-order derivatives, such as gradient calculation or curvature estimation, thus most techniques require sufficiently smooth surface meshes. This property cannot be expected if the used data were

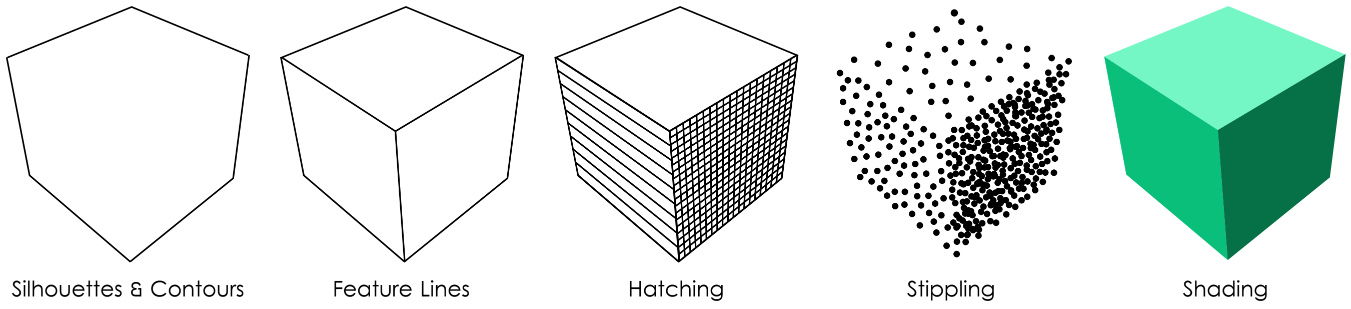


Figure 2: *The different illustrative rendering styles that are presented in this paper. From left to right: silhouettes and contours (Section 4.1), feature lines (Section 4.2), hatching (Section 4.3), stippling (Section 4.4), and shading (Section 4.5).*

acquired by a laser scanner or tomographic devices, such as industrial or medical CT. A smoothing algorithm to address these issues should thus keep prominent surface features, but suppress insignificant details [SCBW14, WYP*15, ZDZ*15].

Robustness. The result of the visualization technique should not significantly depend on the triangulation or tessellation quality of the surface mesh. The result should also not deviate significantly if the number of triangles is strongly reduced, as it is often necessary to achieve faster computation times.

Refinement. Even smoothing the surface may lead to annoying or distracting illustrative visualization results, due to small local irregularities such as noise. One possibility to restrict unwanted results is to give the user the possibility to change the result manually. User-defined thresholds may be a way to control the outcome, if the illustrative visualization technique comes with a quantitative measure that influences the result. Another possibility may be to let the user directly change the result by drawing on the surface or to change a parameter, which influences the result locally. In essence, it is crucial to give the user opportunities to directly or indirectly affect the visualization result.

Frame Coherence. Illustrative visualization techniques need to be frame-coherent because the resulting visualizations are typically explored in interactive contexts. This means that user interactions with the surface mesh (e. g., rotations, zooming, etc.) must not lead to sudden changes of the visual representation because these can be annoying and can disrupt the analysis. The appearance should be constant or changes should be introduced in a smooth manner, both during an interaction or in an animation.

4. Low-level Illustrative Visualization Techniques

Many of the fundamental principles discussed in Section 2 that are essential for illustrative visualization can be realized by applying one or more from a number of low-level techniques. These low-level illustrative visualization techniques include the creation of sparse line drawings based on *silhouettes and contours* (Section 4.1) and *feature lines* (Section 4.2), the use of marks on the surface including *hatching* (Section 4.3) and *stippling* (Section 4.4), and the application of *illustrative shading methods* (Section 4.5); Figure 2 shows a schematic illustration of these different class of approaches. They are based on different degrees of abstraction and we discuss them in this section, beginning with those that introduce the largest amount of abstraction and ending with the ones with less abstraction. At the end of the discussion of all technical approaches we provide a classification of the mentioned approaches in Figure 10 to which we also refer throughout this section.

4.1. Silhouettes and Contours

Silhouettes, (occluding) contours, and feature lines (for the latter see Section 4.2) are part of the group of sparse line drawings.[‡] They restrict the depiction to only a few lines that can potentially be stylized and have been used for centuries in illustrations.

4.1.1. Silhouette and Contour Detection

Among the lines that are used for sparse line drawings, the silhouette is defined as the illustration of an object's outline, i. e., the border of the object to the background. This definition can be traced back to Étienne de Silhouette, the finance minister of the French king Louis XV, who is often connected to the paper-cut shadow profile portraits which were popular at his time. The contour (of a completely smooth object), on the other hand, is defined as the loci of all the points for which the normal vector $\mathbf{n}$ and the view vector $\mathbf{v}$ are mutually perpendicular:

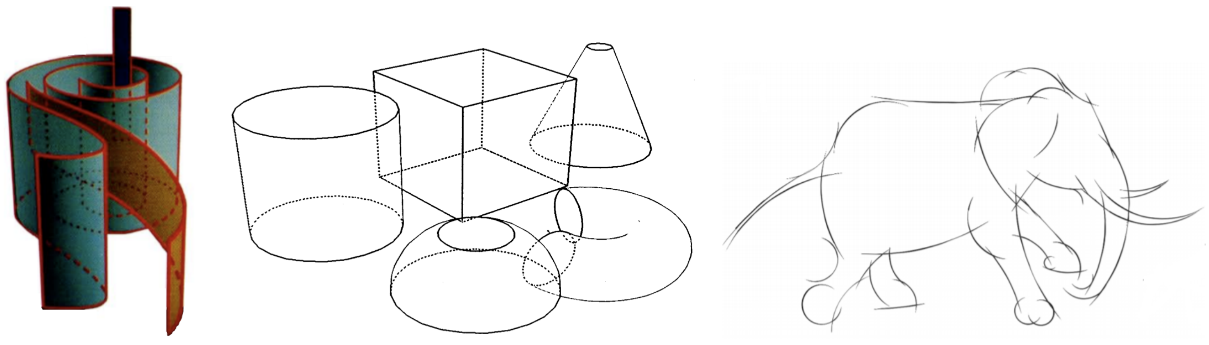
$$\langle \mathbf{n}, \mathbf{v} \rangle = 0. \tag{7}$$

This contour is a very important cue for the understanding of a shape's surface. For example, perceptual studies [Mar76] confirmed that the first stage of visual perception includes the detection of the contours. Because the contour, at the same time, does not provide information about the shape of the surface itself, it provides a strong abstraction of a shape—yet one that can typically still be recognized.

Techniques for the detection and illustration of contours can roughly be divided into three categories [Her99, IFH*03]:

- *image-based techniques*,

[‡] A general overview of sparse line drawings is provided by Rusinkiewicz et al. [RCDF08], the state of the art reports by Hertzmann [Her99], Isenberg et al. [IFH*03], Li et al. [LZL*12], and DeCarlo [DeC12], and a discussion of the advantages of stylization by Al-Rousan et al. [ARSK15].



(a) [DC90] (b) [RvE92] (c) [BLC*12]

Figure 3: *Contour examples with varying styles.*

- *object-based techniques*, and
- *hybrid techniques*;

see the examples in Figure 3 and a summary of existing techniques in Figure 10 which also includes approaches not mentioned in the text. The first among them, *image-based algorithms*, detect discontinuities of the pixel values, i. e., they operate on the rendered image in the view plane. Some techniques also make use of the z-buffer to find relevant discontinuities [ST90, MBC02]. *Object-based methods*, on the other hand, employ the geometry with their primitives, e. g., triangles and vertices, and use the 3D coordinates as well as additional information such as the normals to detect contours. It is important to note that these object-based approaches, unlike the other two groups, also need a dedicated processing of the visibility of the resulting contours [IFH*03]. By using the surface mesh, the contours can be generated as a geometrical object. The generated object can then be further processed for additional stylization or information [HSC12, LLPH15]. *Hybrid methods* combine aspects of both approaches, e. g., by first performing operations on the object space and then using image-space operations [Rus89, RvE92], see Figure 3(b), or plain rendering [GSG*99].

For surface-based illustrative visualization, the state of the art is the extraction of a sequence of edges that represent the contours to facilitate further stylization. Initially, this sequence was extracted as those edges from the surface mesh for which the sign of the dot product between the view vector and the normals of the incident triangle normals changes [Her99], which unfortunately leads to artifacts [IHS02]. Hertzmann and Zorin [HZ00] thus interpreted the surface mesh as the approximation of a smooth surface and extracted the *sub-polygon* approximation of the smooth surface's contour by detecting zero-crossings of the dot product between the interpolated vertex normals and the view vector, and then connecting these zero-crossings across the triangles of the mesh. This process is just as fast as the mesh-based edges, but leads to strokes with much higher quality. A process for the correct computation of the contour based on these sub-polygon contours exists as well [BHK14], but it currently does not facilitate computation at interactive frame rates.

4.1.2. Stylizing Silhouettes and Contours

Pioneering work for the use of stylization was done by Appel [App67] as well as Dooley and Cohen [DC90]. In particular, they derived the visibility of lines and then applied various line styles to encode spatial relations. Dooley and Cohen, for instance, illustrated hidden lines as dashed lines, as shown in Figure 3(a). This general concept was later replicated and realized numerous times; e. g., by Saito and Takahashi [ST90] (for image-based line extraction), Strothotte et al. [SPR*94], Markosian et al. [MKG*97], and Gooch et al. [GSG*99]—all focusing on slightly different aspects and different line styles.

Generally, the stylization (for both artistic and illustrative purposes) requires the existence of object-based line strokes for which the visibility and other properties have been established. To simplify the stylization process, Grabli et al. [GTDS04, GTDS10] and Isenberg and Brennecke [IB06] conceived approaches to systematically capture, process, and finally render the styles based on different stylization schemes, scene properties, and the application domain. Inspired by hardware-assisted rendering, Grabli et al. [GTDS04, GTDS10] used programmable style sheets, and their FreeStyle system (`http://freestyle.sourceforge.net/`) is now available as part of the Blender rendering suite. Isenberg and Brennecke [IB06], on the other hand, generalized the concept of G-buffers [ST90] to be applied as G-strokes to line-based rendering.

Beyond the technical realization of stroke stylization, authors have also addressed stylistic problems. In particular, it is essential for illustrators to be able to apply stylization locally, without being restricted to an existing object segmentation and/or hierarchy. For this purpose, Neumann et al. [NIC07] as well as Cardona and Saito [CS15] combine object-space and image-space methods to allow illustrators to interactively apply local stroke stylization, for example, for technical and medical illustrative visualization (e. g., [NIC07]). A similar local stroke stylization can also be automatically guided by illumination or artistic constraints to achieve a better illustration style. For example, Isenberg et al. [IMS00] use illumination and semantics to change the style locally. Goodwin et al. [GVH07] and Chen et al. [CZLX15] derive stylization rules by being inspired by artworks and illustrations. Finally, for interactive visualizations it is essential that the stylization remains frame-coherent. For this purpose, Kalnins et al. [KDMF03], Bénard et al. [BCGF10, BLC*12], and Lou et al. [LWM15] describe ways to ensure that no sudden changes are introduced by the stroke parameterization or the style, see Figure 3(c).

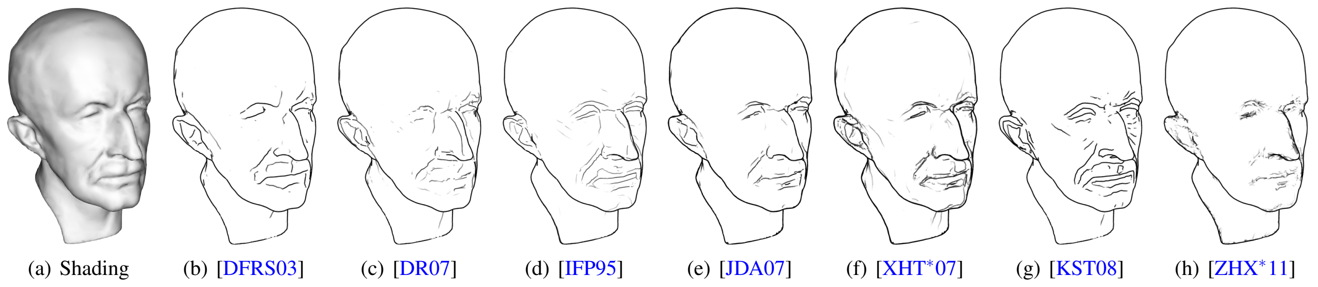


Figure 4: *This overview shows the different feature lines techniques applied to the Max Planck model. From left to right: shading, suggestive contours, suggestive highlights, ridges and valley, apparent ridges, photic extremum lines, demarcating curves, and Laplacian lines.*

4.2. Feature Lines

In addition to silhouettes and contours, sparse line illustrations also comprise feature lines—lines that characterize particular features on the surface of the object that are not necessarily characterized by (potential) changes in visibility [LP16]. These feature lines are similarly important to convey the shape of the depicted objects and are typically placed in regions where discontinuities occur. For example, strong changes of curvature or a strong increase of the illumination values may warrant the use of feature lines. In general, feature lines can be divided in two main classes:

- view-independent feature lines
  - illumination-independent
  - illumination-dependent
- view-dependent feature lines.

See Figure 4 for an overview of the most commonly used feature line techniques.

View-independent feature lines are determined based on features derived from surface information (independent from the orientation and location of the surface) or from light sources. Thus, this category can be further divided into illumination-independent and illumination-dependent techniques. The advantage of the illumination-independent lines over the illumination-dependent or the view-dependent lines is that they can typically be computed in a pre-processing stage and only their visibility has to be determined at run-time, which makes the overall computation faster. View-dependent and illumination-dependent feature lines, in contrast, have the advantage that they can be more stably computed and are derived based on the features that are perceived by a viewer from a certain vantage point, thus making them generally the accepted choice for sparse line drawings [DFRS03, JDA07, CGL*08, LBSP14, BLSP15, LP16]. Only for objects with mathematically ideal features (e. g., illustrative visualizations of technical models) do the two classes produce equivalent output [PBC*16], which makes the use of view-independent lines a better choice in this case.

4.2.1. View- and Illumination-Independent Feature Lines

Crease Lines. To detect such feature lines one can use a definition inspired by that for contours: instead of looking at the set of points where $\langle \mathbf{n}, \mathbf{v} \rangle = 0$ holds, the algorithm for *crease lines* looks for edges where the dihedral angle, i. e., the angle between two incident triangles, exceeds a user-defined threshold:

$$\langle \mathbf{n}_t, \mathbf{n}_{t'} \rangle \geq \tau, \tag{8}$$

where $\mathbf{n}_t, \mathbf{n}_{t'}$ are the normals of neighbored triangles and τ is the user-defined threshold. The value $\tau = \cos\alpha$ denotes the cosine of the dihedral angle which should be exceeded to display the edge. This method, however, can only detect features in the surface mesh if they are characterized by large angles between adjacent triangles—features on smooth surfaces that are represented by highly-tessellated meshes cannot be found. Moreover, it is difficult or often impossible to find a well-suited angle α that leads to an adequate set of feature lines, even for a single mesh. For example, for meshes with strong as well as small features this method may not be able to depict both features, especially if the mesh suffers from noise.

Ridges and Valleys. A more stable approach is thus to treat meshes as approximations of smooth surfaces and then to derive features based on properties of these smooth surfaces, in particular based on the local curvature. Interrante et al. [IFP95] described such *ridges and valleys* (RV) for volume data, an approach that was later adapted to surface meshes by Ohtake et al. [OBS04]. The definition of *ridges and valleys* is based on local curvature values with $|\kappa_1| \geq |\kappa_2|$ and the principal curvature direction $\mathbf{k}_1$ which corresponds to κ_1. The *ridges and valleys* are then defined as the loci of points where the directional derivative of the curvature κ_1 in direction of the principal curvature direction $\mathbf{k}_1$ reaches an extremum:

$$D_{\mathbf{k}_1}\kappa_1 = 0. \tag{9}$$

Depending on the extremum and the sign of the curvature, ridges and a valleys are distinguished: according to two constraints, the sets of points are called

$$D_{\mathbf{k}_1} D_{\mathbf{k}_1} \kappa_1 \begin{cases} < 0, & \text{and } \kappa_1 > 0\text{: ridges} \\ > 0, & \text{and } \kappa_1 < 0\text{: valleys.} \end{cases} \tag{10}$$

Intuitively, one can think of a plane that cuts the surface along the principal curvature direction and looks for an extremum with respect to the curvature. The point with the local highest curvature value is then defined as a ridge or valley point. To filter the lines, which may occur at noisy regions, a user-defined threshold is employed. For every connected series of lines, an integral can be used to measure the curvature along the line. The line is drawn if the magnitude of the integral surpasses the user-defined threshold, otherwise it is discarded. Because the ridge and valley lines are 3rd order derivatives, however, this method is susceptible to noise and requires a smooth mesh. Small discontinuities may lead to erroneous results of the derivatives, thus resulting in visually unpleasant results. The ridges and valleys are, nevertheless, of good quality if the mesh has strong features and is guaranteed to be smooth.

Several approaches to extract salient features similar to ridge and valley lines exist. Watanabe and Belyaev [WB01], for example, use an approximation of the mean curvature, a non-linear averaging of the curvature maps, a histogram-based curvature extrema filtering, and a skeletonization procedure. They used a simplification to determine salient curvature extrema triangles. This approach was later improved to a fully automatic approach by Belyaev and Anoshkina [BA05]. Their method consists of two steps: first, a smoothing of normals with a non-linear diffusion filter and second, an application of a Canny-like non-maximum suppression using a hysteresis threshold operation. Finally, Yoshizawa et al. [YBS05,YBYS07,YBYS08] presented various techniques to detect robust ridge and valley lines on surface meshes based on a better estimation of the shape operator, yielding more reliable curvature measures.

Demarcating Curves. Kolomenkin et al. [KST08] defined *demarcating curves* (DEM) as points of maximum curvature derivative:

$$\langle \mathbf{w}, S\mathbf{w}\rangle = 0 \ \text{ with } \ \mathbf{w} = \arg\max_{\|\mathbf{v}\|=1} D_{\mathbf{v}}\kappa. \tag{11}$$

S is the shape operator and κ the curvature along the direction $\mathbf{v}$. The direction $\mathbf{w}$ that satisfies this property can analytically be determined as the root of a 3rd order polynomial, but is consequently susceptible to noise. This noise, however, can be reduced by discarding line parts where the curvature derivative in the gradient direction is lower than a user-defined threshold.

Relief Edges. Kolomenkin et al. [KST09] presented another approach to detect characteristic lines. They assume that the surface mesh consists of a (smooth) base mesh with a height field. The *relief edges* are then defined as the edge of the height field. The height field of a surface S can locally be expressed as

$$S(\mathbf{p}) = \frac{1}{2}\langle \mathbf{p}, S\mathbf{p}\rangle + \frac{1}{2}C(\mathbf{p},\mathbf{p},\mathbf{p}), \tag{12}$$

where C denotes a $2\times2\times2$ rank-3 tensor defined as $C := (D_{\mathbf{u}}S \ \ D_{\mathbf{v}}S)$. A smooth step edge function E:

$$E(\theta,\alpha,u,v) = \frac{1}{6}\alpha(u\cos(\theta)+v\sin(\theta))^3, \tag{13}$$

is fitted to the height field S with the local coordinates u,v and the edge intensity α. *Relief edges* are then defined as

$$\langle \mathbf{w}, S\mathbf{w}\rangle = 0 \ \text{ with } \ (\theta,\alpha) = \arg\min \int (E-S)^2 \rho \,\mathrm{d}\rho \,\mathrm{d}\omega, \tag{14}$$

where $\mathbf{w} = (\cos(\hat{\theta}), \sin(\hat{\theta}))$. Note that we write θ as the minimization argument, but used $\hat{\theta}$ as the solution. We simplified the condition for a relief edge; Kolomenkin et al. [KST09] describe a detailed derivation of the relationship between θ and $\hat{\theta}$. Similar to the demarcating curves, this approach uses 3rd order derivatives.

4.2.2. View-Independent, Illumination-Dependent Features

Photic Extremum Lines. While all previously described feature lines are based on discontinuities of the surface mesh, photic extremum lines (PELs) [XHT*07] analyze discontinuities in the illumination. These feature lines are motivated by the importance of illumination as a shape cue or visibility and are thus based on the shading of the surface. PELs lines are defined as those locations where the variations of the shading reaches a maximum based on Lambertian reflectance $f := \langle \mathbf{n}, \mathbf{l}\rangle$, with $\mathbf{n}$ being the normal and $\mathbf{l}$ being the light vector (equivalent to the view vector $\mathbf{v}$ for headlight setups). Using the light gradient $\mathbf{w} = \frac{\nabla f}{\|\nabla f\|}$ PELs are then defined as:

$$D_{\mathbf{w}}\|\nabla f\| = 0 \ \text{ and } \ D_{\mathbf{w}}D_{\mathbf{w}}\|\nabla f\| < 0. \tag{15}$$

In this case, the PELs only depend on a single light source and the result can be improved by adding additional light sources. For example, additional local light sources can reduce erroneous lines in noisy regions. Similar to ridges and valleys, the filtering is done by measuring the strength of the integral along a line with respect to the magnitude of the light gradient. A line is drawn if the integral exceeds the user-defined threshold, otherwise it is discarded. Zhang et al. [ZHS10] later improved the computation of PELs to achieve real-time rendering. While the computation of the PELs also relies on 3rd order derivatives, it is only necessary to smooth the normals to obtain reasonable results on noisy surfaces. Later, the PELs were extended such that it can be applied to volume rendering [RMC11].

Lines via Abstracted Shading. The line generator (LAS) proposed by Lee et al. [LMLH07] determines view-dependent regions around potential ridge and valley lines and is based on two passes. The first pass computes the shading, for example based on Lambertian reflectance $f := \langle \mathbf{n}, \mathbf{l}\rangle$. The second pass then identifies image pixels that form ridges and valleys of the image intensities due to the shading. The authors fit a second-degree polynomial $f(x,y) = a_0x^2 + 2a_1xy + a_2y^2 + a_3x + a_4y + a_5$ to the shading values at each fragment. In practice, they use nine (image-independent) sample points (x_i, y_i), arrange them in a 3×3 grid, and construct the matrix X which is made up of the rows $(x_i^2 \ \ 2x_iy_i \ \ y_i^2 \ \ x_i \ \ y_i \ \ 1)$. The matrix $H = (X^TX)^{-1}X^T$ can then be calculated as a pre-processing step. At run-time, the authors determine the matrix $A = HT$ where T is a matrix that consists of the shading values of the sample points. The matrix A yields the coefficients of the fitted second-degree polynomial. Then, the function f can be written as a quadratic form:

$$Q(\mathbf{x}) = (\mathbf{x}-\mathbf{c})^T \underbrace{\begin{pmatrix} a_0 & a_1 \\ a_1 & a_2 \end{pmatrix}}_{M} (\mathbf{x}-\mathbf{c}) \text{ , with} \tag{16}$$

$$\mathbf{c} = -\frac{1}{2}M^{-1}(a_3 \ \ a_4)^T. \tag{17}$$

The eigenvalues and eigenvectors of the matrix M serve as principal curvatures and principal curvature directions on the image. *Abstracted shading lines* are then defined as the curve through $\mathbf{c}$ in direction of the lower curvature value. As such, with a headlight lighting setup, the line generator extracts pixels that contribute to contours and will, in part, reproduce the suggestive contours generator. Through the use of different lighting, however, the visualization designer has the freedom to define where the lines will depict more or less detail. As an image-based technique, the second pass automatically handles the depicted level of detail. Interestingly, in contrast to other feature line approaches the lines are not identified as zero-crossings. Therefore, this method uses first order derivatives, but if the principles curvatures and the directions were determined by the Hessian matrix on the shaded image and the ridge and valley line were determined as isolines, the technique would be of third order.

Laplacian Lines. The *Laplacian lines* (LL) were introduced by Zhang et al. [ZHX*11]. They determine the Laplacian of the Lambertian reflectance $f := \langle \mathbf{n}, \ll \rangle$ and then find the zero-crossings:

$$\Delta f = 0 \text{ and } \|\nabla f\| \geq \tau, \qquad (18)$$

Noisy lines can be filtered with a user-defined threshold τ which measures the magnitude of the light gradient. The *Laplacian lines* are of third order, but similar to the PELs one can restrict the smoothing to the normals to get a better result for noisy surfaces. An advantage of this method is that, instead of determining the Laplacian of the actual shading, it can be calculated on the normals: $\Delta f = \langle \Delta \mathbf{n}, \mathbf{v} \rangle$. This yields a simplified pre-processing step and increases the frame rates during the interaction.

Difference of Gaussians. Inspired by their use in image processing [MH80], Zhang et al. [ZXY*12] adapted the Difference of Gaussians (DoG) concept to surface meshes for the depiction of characteristic lines. Their main idea is to apply two different Gaussian kernels $G_{\sigma_e}, G_{\sigma_r}$ to the illumination $f := \langle \mathbf{n}, \mathbf{l} \rangle$ of the surface. They then obtain the final image by subtracting the smoothed illumination results from each other, formally:

$$H(f, \sigma_e, \sigma_r, \tau) = G_{\sigma_e}(f) - \tau \cdot G_{\sigma_r}(f). \qquad (19)$$

Zhang et al. define the Gaussian of the illumination as:

$$G_\sigma(f)\big|_{\mathbf{x}} = \frac{1}{2\pi\sigma} \int f(\mathbf{y}) \exp\left(- \frac{\|\mathbf{x} - \mathbf{y}\|}{2\sigma^2} \right) \mathrm{d}\mathbf{y}. \qquad (20)$$

The strength of a feature line at a point is then defined as:

$$D = \begin{cases} 1 & \text{if } H > 0 \\ 1 + \tanh(\phi H) & \text{otherwise,} \end{cases} \qquad (21)$$

where ϕ determines the sharpness of the rendered result and D determines the fragment's color. Zhang et al. simplified the Gaussian with $G_\sigma(f)\big|_{\mathbf{x}} \approx \mathbf{v}(\mathbf{x}) \cdot G_\sigma(\mathbf{n})\big|_{\mathbf{x}}$ and used σ_e as the average edge length and $\sigma_r = \sqrt{1.6}\sigma_e$. The computation of the Gaussians is, therefore, just a pre-processing step. Zhang et al. also extended the difference of Gaussians to an anisotropic version. Here, they determine a local parametrization $h(\mathbf{u}, \mathbf{v})$ such that $\mathbf{u}$ points in direction of the principal curvature direction $\mathbf{k}_2$ with $|\kappa_1| \geq |\kappa_2|$ being the minimal absolute value and $\mathbf{u}$ being orthogonal to $\mathbf{v}$. Then, they smooth the normals along $\mathbf{v}$, yielding a visually more pleasing result. In contrast to other feature line approaches, the DoG method detects regions rather than lines defined by zero-crossings. Due to the use of preprocessing and the simple subtraction step, however, is this method faster than the other approaches.

4.2.3. View-Dependent Feature Lines

In contrast to the view-independent lines, other feature lines are computed based on the given viewing (and sometimes illumination) conditions. Below we review the set of suggested line concepts.

Suggestive Contours. DeCarlo et al. [DFRS03] introduced *suggestive contours* (SC) as the first view-dependent feature line method. They describe features that are occluding contours in nearby viewpoints, and that naturally extend occluding contours in the 2D view plane. Two equivalent definitions exist: the first is based on the surface normal $\mathbf{n}$, the view vector $\mathbf{v}$ which points towards the camera, and the projected view vector on the tangent space $\mathbf{w} = (\mathbf{Id} - \mathbf{nn}^T)\mathbf{v}$. Suggestive contours are then defined as those points where $\langle \mathbf{n}, \mathbf{v} \rangle$ reaches a minimum in direction of $\mathbf{w}$:

$$D_{\mathbf{w}} \langle \mathbf{n}, \mathbf{v} \rangle = 0 \text{ and } D_{\mathbf{w}} D_{\mathbf{w}} \langle \mathbf{n}, \mathbf{v} \rangle > 0. \qquad (22)$$

Another definition is based on the radial curvature κ_r. Given the principal curvature directions $\mathbf{k}_1, \mathbf{k}_2$ with curvatures κ_1, κ_2, the projected view vector $\mathbf{w}$ can be written as a linear combination of the principal curvature directions $\mathbf{w} = \alpha \mathbf{k}_1 + \beta \mathbf{k}2$. These coefficients then yield the radial curvature $\kappa_r = \alpha \kappa_1 + \beta \kappa_2$ and the *suggestive contours* can be defined as the loci of points where

$$\kappa_r = 0 \text{ and } D_{\mathbf{w}} \kappa_r > 0. \qquad (23)$$

is fulfilled. Filtering can be applied based on a user-defined threshold which tests if the radial curvature exceeds the given threshold. *Suggestive contours* thus use 2$^{\text{nd}}$ order derivatives, making them less susceptible to noise than lines based on 3$^{\text{rd}}$ order derivatives.

Several authors have adjusted and extended the concept to date. For example, DeCarlo et al. [DFR04] themselves explored the real-time computation of the lines, while McGuire and Hughes [MH04] discussed a hardware-based implementation that also facilitates line stylization. Jeong et al. [JNLM05] and Ni et al. [NJLM06] have presented an approach controlling the depicted amount of line detail by adjusting the underlying mesh in a progressive fashion based on the current view. Goodwin et al. [GVH07] addressed line stylization issues as mentioned above. Other authors have explored the application of suggestive contours specifically to illustrative visualization applications. For example, Burns et al. [BKR*05] demonstrated the computation for volumetric models as used in medical visualization, while Lawonn et al. [LGP14] employed them for vessel visualization.

Highlight Lines. DeCarlo and Rusinkiewicz [DR07] extended the concept of suggestive contours and added two new classes of feature lines, *suggestive highlights* and *principal highlights* (HL), that provide shape cues analogous to shading. The *suggestive highlights* are defined as the loci of points where $\langle \mathbf{n}, \mathbf{v} \rangle$ reaches a positive maxima in the direction of $\mathbf{w}$:

$$\kappa_r = 0 \text{ and } D_{\mathbf{w}} \kappa_r < 0. \qquad (24)$$

Note that, in contrast to Eq. 23, this definition evaluates where the second condition is negative. The *principal highlights*, on the other hand, are defined as strong positive maxima of $\langle \mathbf{n}, \mathbf{v} \rangle$ in the direction of $\mathbf{w}_\perp := \mathbf{n} \times \mathbf{w} / \|\mathbf{n} \times \mathbf{w}\|$. For this purpose the authors define the radial torsion τ_r as $\langle S(\mathbf{w}_\perp), \mathbf{w} \rangle = \tau_r \|\mathbf{w}\|$ and use the

Table 1: *Different feature line methods with their derivative order and if the method is view-dependent, illumination-dependent, and if the method can be applied to animated surfaces in real-time.*

Name	Order	V.-dep.	I.-dep.	Anim.
Occluding Contours	1	yes	no	yes
Crease Lines	1	no	no	yes
Ridges & Valleys	3	no	no	no
Demarcating Curves	3	no	no	no
Relief Edges	3	no	no	no
Suggestive Contours	2	yes	no	yes
Highlight Lines	2	yes	no	yes
Apparent Ridges	3	yes	no	no
Photic Extremum Lines	3	no	yes	yes
L. Abstracted Shading	1 (3)	no	yes	yes
Laplacian Lines	3	no	yes	no
Difference of Gaussian	3	no	yes	yes

principal curvature direction $\mathbf{k}_1$ with $|\kappa_1| \geq |\kappa_2|$ as follows:

$$\langle \mathbf{k}_1, \mathbf{w} \rangle = 0 \text{ and } D_{\mathbf{w}_\perp} \tau_r < 0. \tag{25}$$

A user-defined threshold is then used to discard lines whose derivative is lower than this value. Like suggestive contours, *suggestive* and *principal highlights* use 2nd order derivatives and are thus less susceptible to noise than lines based on 3rd order processing.

Apparent Ridges. Judd et al. [JDA07] presented *apparent ridges* (AR) as an extension to the ridges and valley concept that uses a view-dependent curvature term. Formally, apparent ridges are the loci of points at which the view-dependent curvature assumes an extremum in the view direction:

$$D_{\mathbf{t}'} \kappa' = 0 \text{ and } D_{\mathbf{t}'} D_{\mathbf{t}'} \kappa' < 0. \tag{26}$$

In this definition, the sign of κ' is always positive, and the authors use the sign of curvature (defined on the mesh) to distinguish between ridges and valleys. A ridge occurs whenever Eq. 26 is fulfilled and the curvature on the mesh is negative, while a valley is detected if the curvature is positive. The view-dependent curvature is defined by a projection operator P that maps points on the surface mesh to the view plane. Then, κ' is defined as

$$\kappa' = \max_{\|P(\mathbf{x})\|=1} \|S(\mathbf{x})\| \tag{27}$$

and $\mathbf{t}'$ is the corresponding vector. Like in the approaches discussed before, the authors filter out undesired lines with a user-defined threshold for the view-dependent curvature. *Apparent ridges* combine the advantages of static ridge and valley lines with the benefits of view-dependent features, depicting strong and thus salient changes in the surface mesh as observed by the viewer/camera. Their disadvantage is the use of 3rd order derivatives, which can lead to cluttered/noisy results for insufficiently smooth meshes.

4.2.4. Sparse Lines Summary

The line concepts discussed in Section 4.1 and Section 4.2 differ in their specific characteristics, Table 1 provides an overview. Occluding contours are essential for illustrative sparse line visualizations, while a good choice of feature lines is recommended—in typical cases view-dependent ones. For most of these concepts, the respective authors have demonstrated their suitability for illustrative visualization, for example suggestive contours for terrain models [NJLM06], medical objects [JNLM05, NJLM06, LGP14], biological illustrations [GVH07], and even medical volume scans [BKR*05], highlight lines for mathematical shapes [DR07], apparent ridges for mechanical shapes and medical objects [JDA07], photic extremum lines for mathematical shapes and volumetric medical data [XHT*07], abstracted shading for terrain visualizations [LMLH07], and Laplacian lines for medical and biological surface models [ZHX*11]. Other line concepts exist as well but, such as Sousa and Prusinkiewicz's [CSP03] approach, are often geared more toward an artistic representations of shapes.

The specific choice of lines depends on the application and the line properties. For example, some line concepts have noise issues with non-smooth surfaces, which makes them less suited for surfaces that are directly derived from segmentations or 3D scans. Other feature line methods cannot easily be applied to animated and deforming objects because surface curvature and the curvature derivatives cannot easily be computed in real-time for deforming objects. To address this last issue, Kalogerakis et al. [KNS*09] presented an approach that learns a mapping from a set of animation parameters to surface curvatures for a deforming surface mesh. With this model they are able to predict the changing curvatures as well as their derivatives, based on which they derived a fast algorithm to extract feature lines from deforming objects at runtime.

4.3. Hatching

Hatching is another category of line drawing visualization techniques that is inspired by traditional illustration styles. In contrast to sparse lines that are placed at prominent features, hatching consists of a set of compound lines which cover larger parts of the surface to convey a spatial impression on the surface. The hatching approaches can generally be divided into three categories (see examples for these classes in Figure 5: *image-space*, *texture-space*, and *object-space* methods. These three classes have in common that, for most of them, the hatching strokes are placed along the principal curvature directions (PCDs). Early work has provided several arguments for this choice [IFP96, GIHL00, SW04], in particular in the case of visualization. They differ, however, in the specific domain/space in which the hatching strokes are placed, with implications for their use in illustrative visualization. In addition to using hatching on surface meshes, it can also be applied to volume data; e. g., see the work by van Pelt et al. [VPVVDW08].

4.3.1. Image-Space Hatching

As the name suggests, image-space hatching approaches operate on the projection of the surface mesh. This is a natural approach to hatching as the traditional technique was created by manually drawing lines on a 2D image (e. g., see Dürer's artwork). Image-space approaches can again be roughly categorized into three classes based on how they compute the hatching strokes: *texture projection*, *streamline calculation*, and *line integral convolution*.

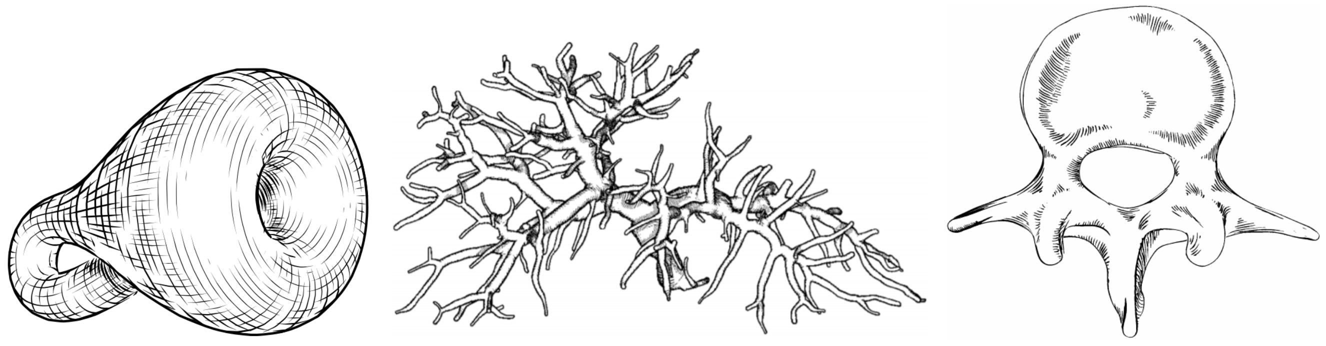

(a) [HZ00] (b) [RHD*06] (c) [GI13]

Figure 5: *Three hatching examples, (a) as an image-based hatching, (b) as a texture-based hatching, and (c) as an object-based hatching.*

Texture Projection. This first class uses dedicated hatching textures for rendering the shapes. These textures are defined in the 2D image plane and are either aligned with a fixed direction depending on the view [LMHB00] or are aligned with the principle curvature directions [LKL06, KYYL08]. This class of hatching methods thus relies on an image-plane mask that ensures that the area of the surface mesh is filled with the hatching texture. While this general approach facilitates an easy implementation and is fast to compute (even for animated shapes in a frame-coherent manner [KYYL08]), it can lead to the shower door effect. This effect results in strokes being perceived as if they were connected to the image plane and not to the object itself, which leads to distraction which is not suitable for most interactive visualization applications.

Streamline Calculation. The approaches in this class generate hatching strokes by tracing streamlines along the PCD (or other directional properties such as gradients of properties such as illumination or depth [LUS13, LUS14]) in the image plane. Typically, these fields are first computed in object space and then projected into image space [HZ00, RK00]. In principle, the method starts by defining seed positions and then applying the streamline calculation using a Euler scheme along the PCD to derive the hatching strokes. To deal with the noise in the PCDs, this approach can use a segmentation of the PCD buffer into regions with homogeneous principal direction fields [RK00] or by smoothing the direction field before the integration of strokes [HZ00]. The resulting strokes can then be stylized, for instance based on illumination, while cross-hatching can be used for particularly dark regions (e. g., [HZ00]). Essential for this class of methods is the seed point placement [LUS13] as each stroke is derived independent from the ones already placed. The seeding is important, in particular, for animations to avoid the shower door effect by ensuring that the seed points stay locked with respect to their initial positions on the 3D surface [LUS14]. Because each stroke is independent, they can still come close or even overlap, even for a good set of seed points. This issue can, however, be addressed by ensuring that strokes only have limited influence, for example using Lichtenberg et al.'s [LSHL16] stroke regions that are maintained for each seed. Alternatively, the seeds are placed iteratively such that a new seed is placed at a given distance from a previous hatching stroke and the stroke is stopped when it comes too close to an already existing line [HZ00]. The result of this approach is typically visually pleasing (e. g., Figure 5(a)), partially because it resembles the manual hatching process.

Line Integral Convolution. Another approach to address the issue of close lines is to use line integral convolution (LIC) which was initially conceived for 2D vector field visualization [CL93, SH95]. In contrast to streamline computation, the LIC method uses an underlying noise field to generate lines in form of shaded pixels. Different noise fields can be used, for example based on shading, color, or features [KYM12]. This method is also not reduced to using the PCDs as the primary directions for the strokes but can also use other directions such as tangents of isocurves of view-dependent features, tangents of isophote curves [Min13, Min15], or light gradients [LKEP14]. While the LIC approach produces generally hatching-like images that can convey shape, the resulting images only exist in form of pixels and are not as similar to the traditional illustration style as the results from the streamline-based approach.

4.3.2. Texture-space Hatching

Texture-space hatching approaches rely on special textures which encode the hatching style. These textures are attached to the 3D surface mesh of the visualized object and then rendered using the traditional graphics pipeline. Brightness variations and the use of single and cross hatching is possible. This approach has the advantage that no shower door effect is created and it results in the impression of an illuminated surface that is illustrated by hatching. Two different techniques exist to parametrize the texture mapping, either classified as *local* or as *global*, as we explain next.

Local Parametrization. A local parametrization maps a region on the surface to a 2D coordinate system, and vice versa. The local parametrization may include a region consisting of several triangles, where the number of neighbored triangles depends on the variation of the curvature [PHWF01] or it may just include a fixed number of neighbors [SBB13]. Praun et al. [PHWF01] presented an interactive hatching approach in which they interactively control the viewing and lighting conditions in a frame-coherent manner. In a pre-processing step, they create hatching textures by starting with a stroke and create several instances of it. The creation of these *Tonal*

Art Maps comprises textures that vary in both tone and resolution. They impose a nesting property, i. e., hatching textures in one image appear in all the finer-resolution images and all the darker-tone images of the same shape. At run-time, a subset of the textures is selected. These textures are then blended together and finally applied to the mesh surface. The projection is based on the work by Praun et al. [PFH00] which cuts patches from an example image. Then, each patch is repeatedly pasted in random places onto the surface until it is completely covered. A direction field is needed to cover the patches onto the surface mesh because the hatching strokes have a certain direction. This direction field consists of the principal curvature direction with the maximal absolute curvature value. Webb et al. [WPFH02] extended Praun et al.'s [PHWF01] work and introduced *Volume Tonal Art Maps* to enhance the control of tone and the use of colors. Instead of using a set of textures to control the brightness, they used a volume texture that allows a finer transition in case of a change of the illumination. Another extension of Praun et al.'s [PHWF01] work was introduced by Gasteiger et al. [GTBP08] who used a hatching method for anatomical meshes which were derived from clinical volume datasets. Their main contribution is to add the model-based preferential directions of the underlying anatomical structures. A real-time hatching on large scenes was proposed by Suarez et al. [SBB13]. In their approach the stroke direction depends on the curvature direction as well as on the light direction. The triangle adjacency is employed to guarantee the coherence of strokes on the surface mesh. Later, they extended their approach [SBB17] and presented a hatching scheme that can be applied on surface meshes with an associated texture. Different hatching patterns are generated based on the surface mesh's texture, which also vary in their tone. The resulting textures are stored in a multi-resolution tonal art map. In addition, different shading types are possible including regular shadows, soft/cast shadows, and self-shadowing. Moreover, Suarez et al.'s algorithm works on static as well as on animated surface meshes.

Global Parametrization. A global parametrization assumes a bijective map from every vertex on the surface mesh to a 2D coordinate system. The calculation of a global parametrization, however, is a challenging task as a *good* parametrization should preserve angles and areas. Such a parametrization does not exist in general, thus several methods exist that find an approximate solution [FH05]. For example, Ritter et al. [RHD*06] presented a hatching technique for vascular surfaces, see Figure 5(b). They first acquire the surface meshes including a segmentation of the CT/MR volume data with the underlying skeleton structures. Based on the skeleton and the distances from the centerline to the surface, they then create texture coordinates. These texture coordinates are then used for the hatching by, in a fragment shader, identifying the coordinate for which the fragment needs to be black. Their hatching scheme was used to illustrate the distance of the camera to the surface, the distance of a lesion to the surface, and to encode distance-encoded shadows. In contrast, Szécsi et al. [SSK16] provided a hybrid hatching solution by introducing *Tonal Art Maps with Image Space Strokes (TAMISS)*. First, they assign every stroke of a tonal art map a unique ID. Then, they fit a curve on each fragment that shares the same ID. Finally, these curves are extruded to image-space stylized strokes.

4.3.3. Object-space Hatching

Object-space hatching relies on explicit line primitives on the 3D surface. The lines are seeded on the surface mesh and are then traced along a direction field, typically the PCDs. Because this processing is independent of the later projection into image space (as the final visualization is generated), it generally cannot guarantee a minimum distance between two strokes and have to use dedicated treatment to address cases where lines end up too close to each other. The advantage of object-space hatching is, however, that it inherently supports the interactive exploration of the depicted objects. These approaches thus avoid a re-computation of the hatching lines at run-time and do not exhibit any visual animation artifacts that arise in image-space approaches from the 2D character of lines.

Elber [Elb99] first presented an object-based approach for freeform surfaces. He uniformly distributed points on the surface, integrated lines along inherent direction fields such as the parametric directions or isophote directions, and parameterized the lines according to illumination and surface properties. Elber and Cohen [EC06] later also explored PCDs and the parametrization according to a contour-based notion of visual importance. This general approach has also been applied to surface meshes [RKS00]. All these techniques, however, lead to rather random stroke distances due to the more or less random placement of seed points. To address this issue, Deussen et al. [DHR*99] and Medeiros et al. [MSVF09] explored a hatching approach that is based on computing intersections of the 3D geometry with evenly placed plane objects. The planes are kept more or less parallel to each other, oriented according to some skeleton or spine. The resulting illustrations resemble some hand-drawn techniques well, but work best only for more or less cylinder-like structures. To also be able to deal with general shapes, authors thus turned to careful seeding and line integration, combined with the use of direction fields on the surface. Singh et al. [SS10], for example, used illumination-based directions, orienting their hatching lines similar to view-dependent feature lines such as suggestive contours and apparent ridges. Zander et al. [ZISS04] instead used direction fields based on the PCDs and described an elaborate line shading technique that can be controlled and changed at run-time. Moreover, they addressed the issue of object-space hatching lines coming too close to each other at regions perpendicular to the view direction by locally changing the line stylization to become less visually prominent. Lawonn et al. [LMP13] then described a GPU-based approach for the line integration on the 3D surface to make it possible to use the techniques in interactive scenarios and for complex meshes. All these techniques, however, rely on purely mathematical concepts to guide the line placement and parametrization. Gerl and Isenberg [GI13] thus use an example-based approach (Figure 5(c)) that first learns line placement and parametrization from a hand-made example illustration and associated model, to later be able to automatically or interactively create new illustrations for other shapes, but using the initially captured hatching style.

Various application scenarios for such object-based hatching have been discussed. The authors of the cited papers have applied their techniques, for example, to the illustrative visualization of medical objects [DHR*99, EC06, GI13, LMP13, MSVF09, SS10], of mathematical shapes [Elb99, SS10, ZISS04], and of botanical models [ZISS04]. Generally, the line integration techniques are useful

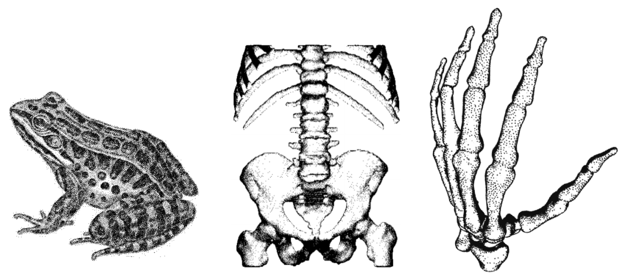

(a) [Sec02] (b) [BTBP07] (c) [MPS04]

Figure 6: *Three stippling examples: (a) image-based hatching, (b) texture-based hatching, and (c) object-based stippling.*

for shapes with well-defined direction fields (e. g., mathematically defined objects), while the surface intersection techniques are more robust to noise and can also be useful for 'less clean' objects such as those resulting from medical scans [SEI10].

4.4. Stippling

Similar to computer-based hatching, computer-based stippling is inspired by a traditional illustration style and uses primitives distributed over the surface of the depicted objects to illustrate its shape and properties. In contrast to hatching, however, stippling does not use directional primitives and restricts itself to dots. The placement of the dots and their size and potential overlapping is thus used to indicate, for example, form, illumination, and materials. Several computer-based stippling approaches exist [MARI17, DI13, KCWI13]. We roughly categorize them again into *image-space*, *texture-space*, and *object-space* methods, see Figure 6 for examples and Figure 10 for an overview.

4.4.1. Image-space Stippling

Generally it is possible to use virtually any traditional image-space stippling technique for the illustrative visualization of surface meshes as well—simply by rendering the surface into an image and then applying the stippling (for an overview see the existing surveys [DI13, MARI17]). Often, dot distributions are computed based on relaxation (e. g., [DHvOS00, Sec02, BSD09], see Figure 6(a))); or using dedicated distribution functions (e. g., [KCODL06, VBTS07]), example-based dot distributions (e. g., [KMI*09]), or scale-dependent schemes (e. g., [MALI11]). These approaches are well suited for producing illustrative visualizations; see, e. g., examples for medical illustration [KMI*09, INC*06, SHS02], biological objects [DHvOS00, HHD03, KMI*09], technical models [DHvOS00], and archeological artifacts/sites [INC*06, MALI11].

4.4.2. Texture-space Stippling

Similar to the equivalent hatching approach, texture-space stippling relies on dedicated stippling textures that are projected onto the surface mesh. Baer et al. [BTBP07], for instance, used the basic idea of tonal art maps [PHWF01] and created stipple illustrations for medical applications. To minimize distortions of the circular points, they employed a polycube representation (e. g., Figure 6(b)). Krüger and Westermann [KW07] instead use 3D volumetric noise textures to assign a noise value to each surface fragment, which is then compared to the fragment's illumination to either render it as a stipple dot or not. This approach works for both surface and volume models, and the authors demonstrated its use in medical visualization.

4.4.3. Object-space Stippling

Finally, object-space stippling places the stipple dots directly onto the surface of the object to be illustrated. Like in object-space hatching, this process has the major advantage that animation or interaction artifacts such as the shower door effect are avoided. Moreover, this process facilitates the use of animated models, such as animations in biological education (e. g., [MPS04], see Figure 6(c)). For example, Meruvia Pastor et al. [MPS02, MPFS03] assign a dot to each vertex of the mesh, while Costa Sousa et al. [CSFWS03] assign marks to each mesh edge. Meruvia Pastor et al. then randomly perturb the dot positions to avoid artifacts from regular meshes and, to adjust the density of the dots to the illumination, use mesh subdivision and progressive meshes. Costa Sousa et al., in contrast, rely on dense meshes and adjust each mark according to the illumination and other properties. In an alternative technique, Meruvia Pastor and Strothotte [MPS04] use point hierarchies to adjust the stipple density. Similarly, Lu et al. [LTH*02, LMT*03] place several dots per polygon and only render some based on the illumination and size on the screen, the latter to take scale issue into account. In a fundamentally different approach, Yuan et al. [YNZC05] use a conformal parametrization to obtain the stippling results, effectively operating in the geometry-image domain. This approach combines the benefits from image-space approaches (e. g., use of edge detection) with benefits from the purely 3D techniques (e. g., frame-coherence in animation). Purely object-space techniques, on the other hand, have been demonstrated to also be extendable to volumetric models [LMT*03]. Examples for the use of object-space stippling for visualization purposes include the application to medical models [LTH*02, LMT*03, MPFS03, MPS04, CSFWS03], technical models [LMT*03], terrain models [CSFWS03], and archeological artifacts [MPS04, CSFWS03].

Remark on Stippling. In contrast to curvature-based hatching, where studies indicate a clear advantage for depth perception (see Section 5), no rigorous study shows an advantage of stippling over conventional shading. It thus remains an open question whether stippling is merely an aesthetically pleasing approach.

4.5. Illustrative Shading

In contrast to the former techniques, where the model was illustrated with primitives such as lines or points, this section reviews various techniques that illustrate the model based on lighting conditions. The term *illustrative shading* refers to those techniques that perform shading that is neither a physically-based simulation of light propagation nor does it attempt to mimick such a simulation process. The central property of illustrative shading is that the shading emphasizes specific structural aspects and communicates them more effectively than a result of a 'photorealistic' shading.

There is, however, not a clear separation between line-drawing

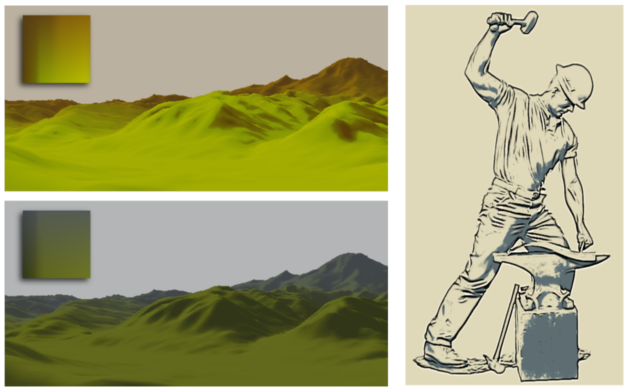

(a) [BTM06] (b) [LMLH07]

Figure 7: *In (a) the results of the X-Toon Shader by Barla et al. [BTM06] is shown, where the 2D look up table is presented as an inlet. In (b) the line drawing by abstracted shading combined with toon shading is depicted by Lee et al. [LMLH07] is illustrated.*

techniques and illustrative shading techniques, as some line drawing generators, in fact, use the shading in the line generation algorithm. The work of Lee et al. [LMLH07] discussed in Section 4.2 presents one example centered around the idea that line drawing can be considered as an abstraction of shading. The example of their work in Figure 7(b) shows lines combined with a toon shading, which we discuss below. The PEL-generator by Xie et al. [XHT*07] produces slightly different results, but is based on the same claim that illumination should be an important factor in the line drawing design. While these two techniques exemplify that shading can serve as a basis for a line generator, the relation between shading and line generation can be also the other way around: shading can be expressed by means of lines [DR07] as it is the case in the highlight lines described in Section 4.2.3. The entire category of hatching techniques uses lines to convey shading. So, the first category of shading techniques applies to both lines as well as illustrative shading.

Recently, Liu et al. [LML16] extended line rendering by introducing the notion of a *global tone*, an illustration method based on the visibility of a region or a point on the surface that is related to ambient occlusion. Their mild toning adds additional shape cues to the line drawing and the entire illustration looks closer to a hand-crafted depiction where charcoal is used as the artistic medium.

Toon Shading. The most classical approach to illustrative shading is inspired by the cartoon industry. There, because of lack of colors, one can quantize shading into few bands of colors or tones only, instead of a continuous threshold. This approach creates a distinct, appealing effect. Such shading style is known as *toon* or *cel* shading. There are many variants of toon shading and it is hard to date its first computerized use, yet an early version was described by Decaudin [Dec96]. The most straightforward implementation of toon shading first defines a desired amount of luminance bands. Then, the shaded intensity coming from a Lambertian shading is quantized to a central intensity of a corresponding band. Toon shading can be used in illustrative rendering for encoding quantity intervals, or it can be used for shading the context decreased detail such that the focus area is differentiated by a richer shading scheme.

Barla et al. [BTM06] extended the idea of toon shading and added eye-space depth as an additional parameter in their *X-Toon Shader* technique. The dot product between the surface normal and the light direction serves as an argument together with the depth value into a two-dimensional lookup table with pre-computed shading values—basically a distinct rendering style that creates an appealing haze effect for distant objects (see examples in Figure 7(a)). This principle has been extended by Hao et al. [HCZW10] who used a 2D texture to assign a specific color to the fragment of a surface. The x-component is used as the radial curvature, similar to the curvature in suggestive contours. The y-component of the texture encodes the surface shading, i. e., the positively clamped dot product of the surface normal with a light vector.

The idea of capturing the shading behavior in a texture has further been extended in Vergne et al.'s *Apparent Relief Descriptor* [VBGS08]. This approach essentially uses a texture that defines how the surface with particular curvature properties will be shaded, a concept similar to *Curvature-based Transfer Functions* for volume rendering [HKG00]. The shape descriptor is characterized by the principal curvatures. A lookup texture is used to encode the shading (or coloring) based on the combination of these two curvature parameters. This way, ridge and valley areas can be made visually distinct, or any other features that have distinct curvature properties.

Cool-to-Warm Shading. Another well-known example of illustrative shading is Gooch et al.'s [GGSC98] technical illustration rendering. Their key idea comes from observation of fine arts, where painters rarely use black paint for shaded objects or objects in shadow. Instead they use a more *cool* color (typically darker blue) and encode the light propagation even in areas in the shadow. In the proposed shading scheme, therefore, not only does the luminance of an object change at a particular location on its surface, but also the color simultaneously changes in hue. This technique is known as *Cool-to-Warm* or *Gooch* shading. Highlighted areas become more yellowish, while strongly shaded areas become more blue. An example result of this technique is depicted in Figure 8.

Shading by Transfer. The X-Toon Shader [BTM06] mentioned above uses a pre-computed texture for fetching the values during the shading process, based on light vector, normal vector, and the eye-space distance as input parameters. The idea of pre-computing the shading values and fetching them during the image synthesis has been used already for many years. The famous example on how to transfer hand-drawn shading using the *pre-computed* approach is known as the *Lit-Sphere* concept [SMGG01]. The artist can provide various visual styles by drawing a 3D sphere. This sphere is then used as a lookup texture using the eye-space normal as input. On the particular location in the target geometry, the shading value is determined by computing the eye-space normal vector at this location that is input into the lit-sphere lookup texture. The point on the sphere which has the same normal vector will be taken as an output of the lit-sphere lookup and will be used for shading the target geometry.

Recently, the lit-sphere concept has been extended by a *sphere on the table* in the StyLit approach [FJL*16]. This way, not only

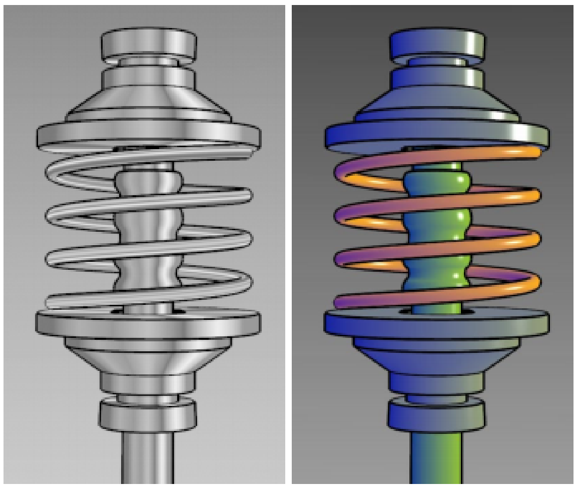


Figure 8: *One of the first illustrative shading techniques by Gooch et al. [GGSC98]. Strong edges are illustrated in black, highlights in white, and the surface shading is based on the luminance.*

can the local shading be captured, but also the global-illumination effects such as shadows and color bleeding. Previous techniques for style transfer such as *Image Analogies* [HJO*01] or *Neural Stylization* [GEB16, SID17] use image statistics or convolutional neural networks for transferring the style. These turn out not to lead to satisfactory results for a given purpose of the illumination transfer. However, it is possible to decompose the light into *light-path expressions* [Hec90], namely into the direct diffuse, direct specular, indirect light, and first and second bounce components. By computing image statistics for each light path expression independently, we can thus achieve a notably better result. This result can be improved further by using a texture synthesis that controls the level of usage of particular image patches to avoid image patch overuse.

Prior to the above work, Tietjen et al. [TPB*08] proposed the *shading maps* technique where each map contains a particular light type or geometric property encoding. These maps are then combined with various weights in the compositing stage to obtain a final image. First, the surface mesh is rendered in different ways, e. g., with shading of different light positions, the depth, contours, feature lines, curvature, etc. The results are stored in the framebuffer. The final image is generated as a weighted sum of all previously calculated results. Therefore, the final image slightly attenuates features that would not be shown with standard shading.

Dynamically Coherent Stroke Shading. The examples of stylized shading discussed above work, in principle, for one image only. In a naïve animation sequence, each image is thus created independently which naturally leads to temporal visual artifacts. This problem was addressed by Breslav et al. [BSM*07] who used a coherent approach to fix 2D texture patterns on the surface mesh. As a first step, they distributed 3D sample points over the surface whose 2D positions on the view plane is tracked at run-time. The apparent motion is then determined by comparing the positions on the the previous with those on the current frame. A resulting transformation is applied to the stylized pattern such that the visualization becomes frame-coherent during interactive exploration.

Scaled Shading. A prominent category of illustrative shading consists of techniques that put a special emphasis on strongly conveying surface details by modifying the standard shading equations. Rusinkiewicz et al. [RBD06], with their *Exaggerated Shading* technique, enhanced details on the surface mesh, motivated by the rules for cartography heightmap design. The standard diffuse Lambertian shading is therefore modified to support levels of exaggeration of slope changes based on the parameter a:

$$\frac{1}{2} + \frac{1}{2} \operatorname*{clamp}_{[-1,1]} a\langle \mathbf{n}, \mathbf{l} \rangle \tag{28}$$

This exaggeration further utilizes a spectral approach, where the surface is decomposed into several bands of surface details. These levels can then be exaggerated individually and the final appearance is composited as a weighted sum of all surface-detail frequencies. To achieve various detail levels, the surface normals are smoothed in multiple iterations, each iteration representing one scale.

A work that is visually similar to the exaggerated shading is based on light warping driven by view-dependent curvature [VPB*09]. The reason for this technique to effectively convey shape comes from visual perception research that describes the way how curvature depends on the compression of reflected light patterns on the surface. Interesting surface details are characterized by non-zero curvature values. The technique essentially exaggerates the reflection vector based on view-dependent curvature so that it enriches the light pattern that is compressed onto the interesting surface area.

While these above techniques emphasize surface features, a negative side-effect is that the modification of the shading changes also the perception of the material. A glossy material might easily change to a brushed appearance, for example. Vergne et al. [VPB*11] have thus presented a technique that both enhances the surface depiction and preserves the material appearance called *Radiance Scaling*. Their key idea is to modify reflected light intensities based on the view-dependent surface curvature as well as on the material characteristics. In the case of Phong's illumination model, each light component, i. e., ambient, diffuse, and specular light, has a separate scaling function. An example of a surface landscape that has been emphasized by radiance scaling is shown in Figure 9(a).

Light Collages. Several of the techniques discussed above decompose particular aspects of illumination or geometry into a number of aspects that can be controlled individually. In the *Radiance Scaling* or the *StyLit* techniques, it was the different light expressions; in the case of *Exaggerated Shading* it was the surface detail frequency. The technique called *Light Collages* [LHV06] also decomposes the geometry of the surface into several patches that are separated by high-curvature borders. For each patch, an optimal light source is calculated, so that the diffuse and the specular lights convey the surface details to the viewer. Then, the light corresponding to a particular patch illuminates only that patch. The illumination originating from different light sources is blended at patch borders to create a continuous illumination effect that is locally optimal but globally inconsistent.

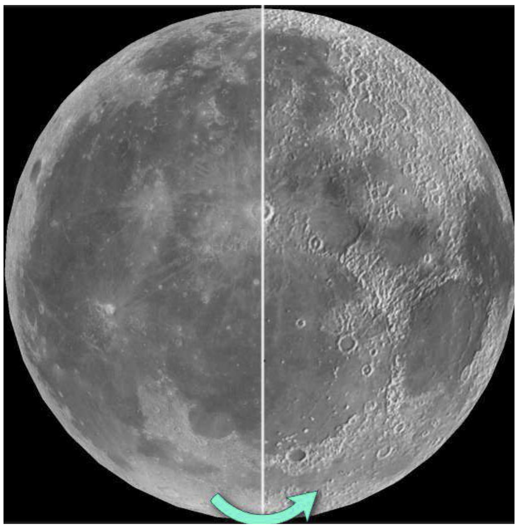


Figure 9: *The method by Vergne et al. [VPB*11] shows how radiance scaling can significantly improve the perception of surface features, without affecting the perception of material.*

5. Evaluation

Evaluation for this specific sub field of visualization are at the intersection between evaluation for visualization in general [IIC*13] and evaluation for non-photorealistic/expressive graphics [Ise13]. Below we give specific examples and discuss how they shed light on the generation and use of illustrative visualizations.

Since illustrative visualization both derives its inspiration from traditional illustration and uses actual data to create *case-specific* (i. e., not generalized) *visual representations*, evaluation techniques can both be used to assess the quality of the specific representation and to understand the underlying *general illustrative principles*. Moreover, illustrative visualizations are often either evaluated *qualitatively* to assess people's preferences and the suitability of a given visualization for a specific task, or *quantitatively* to assess depth and shape perception. Perceptual studies (e. g., see [PBC*16]), in particular, are often used to understand perception aspects.

As an example for an evaluation of a specific technique, Tietjen et al. [TIP05] applied different rendering styles to medical CT datasets. Based on an existing segmentation, their technique illustrated the anatomy in a combination of contours and shading. 33 participants (8 of them surgeons) filled out a questionnaire to assess the suitability of the generated illustrative visualizations. For each set of questions, two visualization results of the same anatomy were shown, each using a different combination of visualization techniques. The questionnaire then asked about a preferred visualization and posed further questions about the images. The analysis of the questionnaires showed that, in the used example of thorax and liver visualization, the use of contours are appropriate for surgical planning. Additional information such as colored contours and transparent surfaces, however, were rated as useful. Based on their observations, the authors express their belief that such hybrid visualizations are useful in general for medical applications.

Later, Ritter et al. [RHD*06] evaluated their illustrative visualizations of vascular structures. Their real-time technique relied on perceptual principles and in their evaluation they specifically concentrated on the used techniques of stroke hatching, distance-encoded surfaces, and shadows. They conducted a web-based perceptual study with a large pool of participants (160 people), with almost a quarter being medical professionals. Because their techniques were designed to improve depth perception, they based their evaluation on precise depth judgments. Their results demonstrated that the explicit encoding of distance by illustrative means leads to a more accurate perception of relative distance to the viewers compared to traditional shading. Moreover, the shape of the vessels was equally well communicated by illustrative means compared to traditional shading. Finally, their illustrative distance-encoded shadows were also effective—they led to more accurate depth judgments compared to visualizations without such shadows.

In a subsequent evaluation the same group [HWR*10], conducted a lab study that also relied on precise judgments of the perceived distance of parts of the visualization. This evaluation looked at three application scenarios, designed for liver surgery. They also added the illustrative visualization of resection surfaces compared to the original technique. The authors also captured the participants' reasoning by applying a think-aloud protocol. Therefore, the authors were able to not only confirm the effectiveness of the illustrative techniques (i. e., illustrative visualization being faster than traditional techniques) but also captured valuable qualitative feedback (e. g., suggestions for improved parameterization and for a better encoding of specific data configurations).

Hummel et al. [HGH*10] described an informal and brief evaluation of illustrative visualization in a different application domain: the illustration of integral surfaces. They collected feedback from domain experts and collaborators on the effectiveness of the different illustrative techniques they used. In line with the results reported by Tietjen et al. [TIP05], they reported that the use of silhouettes/contours improved the visualizations and that purely abstract (lines only) illustrations were not considered to be the best choice. Moreover, respondents confirmed the effectiveness of Hummel et al.'s adaptive transparency technique and preferred techniques that also made use of high-quality illumination.

In a final example by Lawonn et al. [LLPH15], the authors again evaluated illustrative visualizations of vascular structures. They first conducted a qualitative evaluation with 15 expert participants, a third of whom were surgeons. Discussion with the experts led to a slight change of the visualization technique. Lawonn et al. then conducted a subsequent quantitative evaluation in form of a web-based questionnaire with 50 participants, 16% of whom were physicians. The results showed that their technique improved depth perception in comparison with other well-known methods.

While these evaluation results of visualization examples can likely be generalized beyond their very specific application scenarios, there are also more fundamental questions that relate to the use of illustrative techniques for visualization. The utility of certain illustration principles is one aspect that can be established in general such that they can be applied to illustrative visualizations of 3D shapes more broadly. Also, due to the fundamental motivation of the field based on the tradition of hand-made illustrations it is important

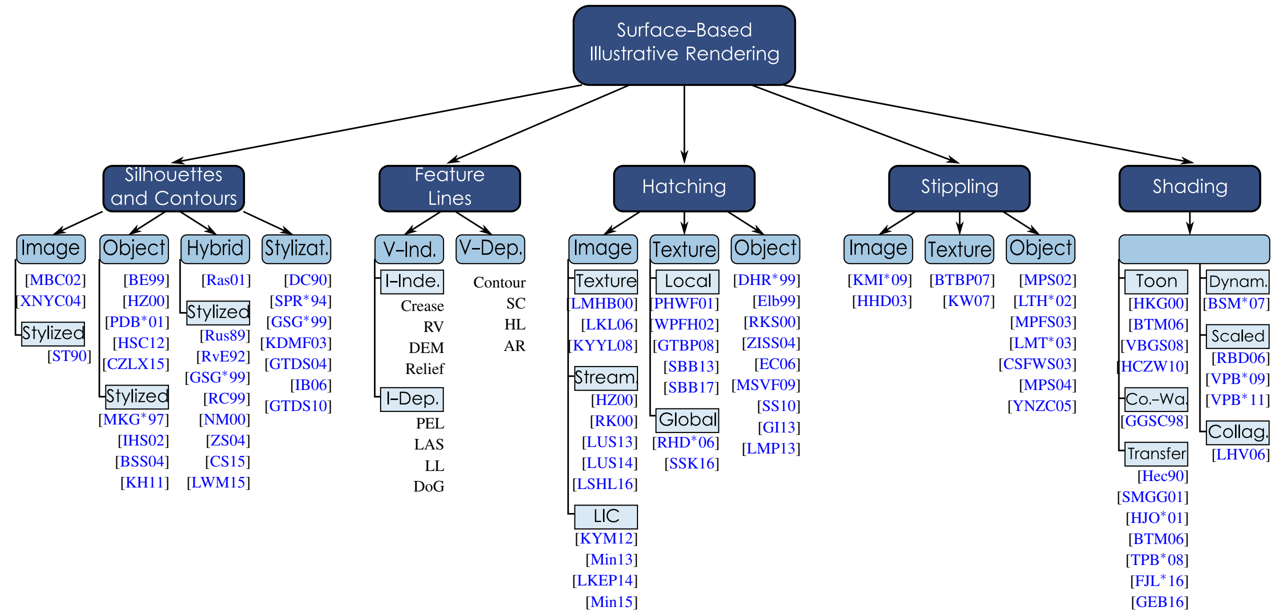


Figure 10: *An overview of the discussed illustrative visualization techniques. Every field is divided in different categories, which reflects the subsections of the corresponding section. Note that the lists of specific papers for each approach only serve as examples.*

to question the relationship of computer-supported techniques to hand-made examples.

In particular to explore this second point, Isenberg et al. [INC*06] conducted an observational study for non-photorealistic rendering (NPR) techniques compared with traditional hand drawings—using specifically illustrations and illustrative renderings of 3D shapes. They used three surface models (a human torso, a plant part, and a scan of an archeological artifact) and, with five NPR techniques (primarily hatching and stippling, combined with contours and feature lines), created illustrative visualizations. In addition, they also asked five professional illustrators to produce illustrations of the same models. The resulting 30 images were then printed and used in a pile-sorting study. The study participants were asked to sort the illustrations into groups according to their own preferences, with virtually no constraints, and then to discuss these groups. After this discussion, the experimenters conducted a semi-structured interview, also asking specific questions. The results of the study included a better understanding of the differences between computer-generated and hand-made illustrations, including insights on what makes an illustration look hand-made and recommendations on what to improve for illustrative rendering. In particular, the study results demonstrated that, for the specific illustrative techniques used in the study, participants were generally able to tell apart computer-generated from hand-drawn illustrations. Nonetheless, participants appreciated both: the computer-generated ones for their precision and detail, and the hand-drawn ones for their "character."

To better understand the fundamental differences between hand-drawn and computer-generated line drawings, Cole et al. [CGL*08] compared algorithmically generated lines (silhouettes/contours and view-dependent feature lines) with hand-dawings of the same shape. These drawings were produced by 20 artists for 12 different shapes using a clever study design that asked the artists to first freely draw and then to copy the drawn lines onto a low-contrast image to make them comparable with each other. The comparison of the scanned hand drawings and the computer-generated illustrations showed that most algorithmic lines indeed match with hand-drawn ones, for both object-space and image-space line extraction. There are, however, also some hand-drawn lines that cannot be explained by the analysis of local properties—artists sometimes choose to place lines in regions of weak ridges or valleys, sometimes depending on the semantics of the subject matter.

A follow-up study by Cole et al. [CSD*09] then raised the question of how well line drawings can depict the underlying shape. They thus again used twelve 3D models and created visualizations using shading and different line-based techniques. They asked 560 participants (on Mechanical Turk) to orient randomly placed surface gauges such that they marked the local surface orientation, resulting in a total of 275,000 gauge placements. The analysis of this data showed that, for half of the examined 3D models, the line drawings could depict shape nearly as well as a shaded image. For other shapes, the participants had more trouble interpreting the surface shape solely based on the used sparse line renderings. In all but one case, however, the best computer-generated drawing led to a slightly lower error than the artist's drawing, but the specific algorithm depended on the depicted 3D shape.

A comparative evaluation to assess feature lines was conducted by Lawonn et al. [LBSP14]. They asked 149 participants to order six computer-generated line illustrations generated using ridge and valley lines, suggestive contours, apparent ridges, PELs, demarcating curves, and Laplacian lines. The participants were asked to rank based on realistic assessment, aesthetic depiction, and general preference. Each of the six line illustrations used a different type of feature line, and the study was carried out for two different 3D shapes. Based on the analysis of these rankings, the authors concluded that suggestive contours, apparent ridges, and Laplacian lines were the preferred techniques. Overall, the authors recommend the use of suggestive contours because they do not use third-order derivatives. A later analysis by Baer et al. [BLSP15] re-tested these results and confirmed their statistical significance.

6. Applications

As already pointed out in the discussion of the individual approaches, illustrative visualization techniques can be applied to numerous domains. Below we name a selection of them and briefly discuss some examples of how illustrative techniques have been used.

Archaeology. In the past, archeology has relied on traditional illustrations, both for the documentation of excavations and artifacts as well as for the illustration of reconstructions. For many of these tasks it is possible to also use illustrative techniques, based on 3D models or 3D scans. For example, virtually reconstructed buildings [MS98, SMI99] (Figure 1(d)) were visualized with sparse lines, using line stylization to encode semantics of the excavation or reconstruction. In illustrative renderings of scanned ancient objects [KST08, LLZ11, LTPH17] the visualization techniques can enhance, for example, the shape of the depicted objects. Illustrative visualization becomes especially relevant for archeology and for other fields when concrete information is combined with information that is uncertain or even fully speculative. In excavations one can find only a few pieces and speculate what the entire object looked like. In such scenarios, illustrative rendering techniques based on line or point elements may communicate the uncertain part effectively, in contrast to the certain information which could be depicted with a detailed or even photorealistic rendering technique.

Molecular sciences. In the context of molecular sciences, illustrative visualization techniques can be applied to molecules to explain the continuous character of the different abstraction schemes [vdZLBI11], make additional visual variables available to visualize surface information [CG07, CPJG09, LKEP14], or illustrate complex temporal aspects and reactions [BPJG12, MPSV14, MWPV15]. The abstraction introduced by many illustrative techniques can not only make it possible to portray complex assemblies like the contents of a cell [KAK*18, LMAPV15] (Figure 1(c)) but also facilitate the rendering of such scenes at interactive rates.

Medicine. The applications of illustrative visualization in medicine are manifold since medical illustrations have traditionally played an important role in the field. For instance, vascular structures were displayed with distance encoding to a tumor or to the user [RHD*06, LLPH15], illustrative augmented reality has been used for liver surgery [HWR*10, LLH17] (Figure 1(b)), and virtual endoscopy has been augmented with illustrative methods [LGP13]. Illustrative visualization has also been used for medical training [CSESS05]. Even fluoroscopic images with extracted 3D surface were combined with contours, stippling, or hatching illustration [RSB*13]. Finally, some approaches have combined various illustrative techniques [TIP05, LSHL16] (Figure 1(a)).

Geosciences. For geovisualization, the use of illustrative rendering may communicate complex and spatial information [DÖ7]. For 3D city scenes, for example, visualization techniques can convey level-of-abstraction transitions during the interaction [STKD12]. Furthermore, illustrative visualization can emphasize the salience of user-specified regions of interest to depict the whole scene with a focus on such areas [PCG*11].

Flow visualization. Another important application area is flow visualization (for a more comprehensive discussion of illustrative flow visualization techniques refer to Brambilla et al.'s [BCP*12] survey). Major trends are flow pattern enhancement [BMGS13], the support of depth perception, e. g., by means of halos [EBRI09], and the illustration of blood flow [BPMS12, vPOBB*10]. Further possibilities to visualize flow are surfaces, e. g.streaming surface, path surface, and streak surface [BWF*10, HGH*10, CFM*13, SJEG05] which use illustrative style elements such as contour lines, silhouettes/occluding contours for iso-surfaces, lines for cuts, streamlines, halftoning, and local transparency. Also additional elements such as arrows [LGV*16] or line styles with specific patterns [FG98, LS07, EBRI11, EBRI15] can be added to illustrate flow direction. Other approaches cluster the flow information and use illustrative visualization techniques [CYY*11, OJCJP16].

Physics. Illustrative techniques have been applied in relativity and astrophysics [WBE*05, WBE*06]. The main goal in this context was to convey various aspects of the theories of special and general relativity and of related fields of cosmology and astrophysics.

Miscellaneous. Illustrative visualization approaches are applied in many more fields such as biology [DHvOS00], technical illustrations [HBP*07, ND05], and mathematics [SS10]. Interaction frameworks exist [KMM*02, GI13] to support both the automatic generation of illustrative visualizations and their manual adjustment. Another interesting application is the retrieval of a 3D model from a simple line drawing. If the user draws a model based on an abstract depiction, a matching 3D surface can be obtained [ERB*12]. Finally, illustrative visualization techniques can also be used to produce physical visualization objects such as 3D laser-engraved glass [HCW10].

7. Discussion & Future Work

As our survey has shown, illustrative visualization has a lot of potential for visualization in general. As we discussed in Section 2 and indicated throughout our discussion, it allows visualization designers to convey salient information, encode (additional) information, improve perception, and guide attention through abstraction and emphasis. We largely focused on surface-based models, but many techniques can be adapted to other types of data. Moreover, we focused on low-level techniques because they are more universal, while higher-level techniques tend to be more domain-specific (even though some more general approaches exist [RBGV08]). Yet, even for the discussed low-level techniques there are a number of constraints and limitations which we discuss next.

The field of contour detection has been extensively addressed in the literature as shown in Section 4.1. In fact, today it is easy and effective with modern GPU power. It is still challenging, however, to stylize the resulting strokes consistently and continuously using a parametrized curve. If a contour is closed with length L, for example, it would be desirable to have the position on the curve by a value in $[0, L]$. This setup would allow us to compute view-aligned quads along the contour with a parametrization such that the quad could be stylized with arbitrary patterns. The challenge, however, still is to change the parametrization consistently during the interaction, without sudden or large changes to maintain frame coherence. First approaches tackle this problem by searching nearby contour triangles to assign the new value after rotation [LWM15].

As we discussed in Section 4.2, a menagerie of feature lines exists, each with their different advantages and challenges. For most of them the computation is based on local criteria, so the exploration of more global approaches would be interesting. For example, we see potential to include information theory approaches to detect features in the surface mesh. It may be possible to analyze lines with respect to the information they would add if they were drawn. Large lines that are distant from each other may add more information to the mesh than lines that are close to each other. Another interesting idea is to analyze smoothing approaches that remove noise without removing essential features from the surface [JDD03, KCL09, HS13, YWQ*14, WYP*15]. These approaches could be analyzed to generate new feature line techniques or to apply these approaches using existing feature line extraction techniques, without having to assuming a perfectly smooth surface.

Some feature line techniques use ambient lighting to determine the lines. Here, it might be interesting to analyze different lighting methods before the feature line technique is applied. Especially exaggerated methods [RBD06] may be a promising candidate.

The evaluation of novel feature line techniques should be an interesting point for future work. Cole et al.'s [CGL*08] study compared hand-drawn images with feature line techniques. We believe that an artist would draw the features of an object differently if the object were well-known. For instance, if the artist were to draw a cow model, he or she would likely use fewer lines because the shape of a cow can be recognized from a few contours and features. In contrast, we conjecture that the same artist would draw more lines when confronted with an unknown object [SNEA*16]. To better study such questions, eye trackers could be used in observational studies to analye how the eyes scan an illustrative visualization.

The actual use of illustrative visualization in interactive systems, e. g., for education, engineering, therapy planning, etc. deserves more attention. In this same context, more work is needed to study the illustration of animated surfaces as many existing approaches for line extraction cannot easily deal with them at the interactive frame rates needed for practical applications.

Hatching methods have also been explored in detail. A big challenge is applying hatching methods to animated surfaces, in particular when trying to ensure minimal distance between the hatching strokes. Lichtenberg et al. [LSHL16] employed an image-based approach that used LIC with seed points determined with a *contact region* instead of the normal noise texture. This method ensures a small distance between neighboring hatching strokes, but a minimal user-defined distance cannot be achieved. Another idea to apply hatching strokes on a surface mesh would be to determine a global parametrization, cf. Section 4.3.3. On the 2D map of the surface, the method by Jobard and Lefer [JL97] could be applied to ensure a minimal distance. A reasonable parametrization might be the *least squares conformal map* by Lévy et al. [LPRM02]. This parametrization, however, is generally not rigid, i. e., not distance-preserving; so one should take care that the distances are encoded in the 2D map.

Stippling methods were also extensively investigated. An interesting extension would be to encode information on the radius of the stippling points, e. g., the distance to a region of interest—this idea was sketched in the work by Ritter et al. [RHD*06]. This general approach can also be extended to the drawing of glyphs on the stippling circles to encode information.

The field of illustrative shading is quite large and the presented methods did not follow a specific common goal. It is thus difficult to judge what is missing in this field and we can only add some ideas of what could be of interest or be considered for further improvement. An interesting illustrative approach by Tietjen et al. [TPB*08] (Section 4.5) combined various shading maps—shading by transfer. These shading maps could be arbitrarily extended by other maps. It might be interesting, for instance, to combine these techniques with *exaggerated shading* [RBD06]. The combination could not only be determined by the user but also automatically. Objects that might be of interest should be illustrated with maps that enforce an emphasis on structures, while surrounded context objects could be illustrated with low-information maps. This idea could also be applied to single objects with an underlying scalar field of interest, e. g., a distance map. In the case of a close distance, the shading map should then be chosen differently from region with a high/distant value.

Lit spheres [SMGG01, FJL*16] may be extended to take the volume into consideration. Instead of mapping the normals to the sphere and applying the color scheme to the model, one could add information inside the sphere such as a volumetric sphere that contains information. Then, the shading could be defined that first assigns the normals to the sphere and then uses a second scalar field that yields the radius. Overall, this approach would provide a volumetric lookup texture that encodes two types of information.

8. Concluding Remarks

Illustrative visualization techniques show potential to convey information and to abstract surface objects. They are suited for simplifying structures, but they may also support depth and shape perception. Various studies confirmed the potential use of illustrative visualization methods in comparison with standard visualization techniques, e. g., Phong shading. In different applications, illustrative visualization methods are employed to gain insight or to transfer knowledge. The manifold application areas confirm the potential of illustrative visualization techniques. However, they are still rarely used in commercially available software. Thus, more exchange between academic research and developers is desirable.

Contours and feature lines give a first impression on the surface mesh and may be a good alternative to other visualization methods. They can be applied if the perception of the spatiality is not paramount. Feature line techniques also show potential to depict the shape, but it strongly depends on the underlying surface mesh. For an improved shape perception hatching strokes, stippling points, or illustrative shading methods should be employed. Thus, the order in which we presented the different techniques in this survey may reflect the way humans perceive shape; but the more certain a shape is, the more information is needed to give that impression. Here, we regard hatching and stippling as similar. The question of which visualization technique should be applied to a scene with various objects depends on underlying problem. The more information, e. g., color, primitives are used to illustrate the objects the more attention it will receive.

Most of the presented methods were used as an alternative depiction for surface meshes. Alternatively, these methods can be extended to not only improve the spatial or depth perception, but also to encode additional information by the style of the used primitives. Overall, we hope that this survey inspires future development and sparks ideas for additional applications.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Transactions on Graphics 22*, 3 (July 2003), 485–493. doi: 10.1145/882262.882296 3

[App67] APPEL A.: The notion of quantitative invisibility and the machine rendering of solids. In *Proc. ACM National Conference* (1967), ACM, New York, pp. 387–393. doi: 10.1145/800196.806007 6

[ARSK15] AL-ROUSAN R., SUNAR M. S., KOLIVAND H.: Stylized line drawings for shape depiction. In *Proc. ICIDM* (2015), IEEE Computer Society, Los Alamitos, pp. 28:1–28:5. doi: 10.1109/IDM.2015.7516335 5

[BA05] BELYAEV A., ANOSHKINA E.: Detection of surface creases in range data. In *Proc. Mathematics of Surfaces XI*. Springer, Berlin/Heidelberg, 2005, pp. 50–61. doi: 10.1007/11537908_4 8

[BCGF10] BÉNARD P., COLE F., GOLOVINSKIY A., FINKELSTEIN A.: Self-similar texture for coherent line stylization. In *Proc. NPAR* (2010), ACM, New York, pp. 91–97. doi: 10.1145/1809939.1809950 6

[BCP*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative flow visualization: State of the art, trends and challenges. In *Eurographics State of the Art Reports* (2012), Eurographics Association, Goslar, Germany, pp. 75–94. doi: 10.2312/conf/EG2012/stars/075-094 2, 3, 18

[BE99] BENICHOU F., ELBER G.: Output sensitive extraction of silhouettes from polygonal geometry. In *Proc. Computer Graphics and Applications* (1999), IEEE Computer Society, Los Alamitos, pp. 60–69. doi: 10.1109/PCCGA.1999.803349 16

[BHI*05] BARTZ D., HAGEN H., INTERRANTE V., MA K.-L., PREIM B.: Illustrative rendering techniques for visualization – Future of visualization or just another technique? In *Proc. Visualization*, no. 4 in Panels. IEEE Computer Society, Los Alamitos, 2005, pp. 715–718. doi: 10.1109/VISUAL.2005.1532863 2

[BHK14] BÉNARD P., HERTZMANN A., KASS M.: Computing smooth surface contours with accurate topology. *ACM Transactions on Graphics 33*, 2 (Apr. 2014), 19:1–19:21. doi: 10.1145/2558307 6

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., UNO LEVY B.: *Polygon Mesh Processing*. AK Peters, 2010. doi: 10.1201/b10688 4

[BKR*05] BURNS M., KLAWE J., RUSINKIEWICZ S., FINKELSTEIN A., DECARLO D.: Line drawings from volume data. *ACM Transactions on Graphics 24*, 3 (July 2005), 512–518. doi: 10.1145/1073204.1073222 9, 10

[BLC*12] BÉNARD P., LU J., COLE F., FINKELSTEIN A., THOLLOT J.: Active strokes: Coherent line stylization for animated 3D models. In *Proc. NPAR* (2012), Eurographics Association, Goslar, Germany, pp. 37–46. doi: 10.2312/PE/NPAR/NPAR12/037-046 6

[BLSP15] BAER A., LAWONN K., SAALFELD P., PREIM B.: Statistical analysis of a qualitative evaluation on feature lines. In *Proc. Bildverarbeitung für die Medizin* (2015), Springer, Berlin/Heidelberg, pp. 71–76. doi: 10.1007/978-3-662-46224-9_14 7, 17

[BMGS13] BORN S., MARKL M., GUTBERLET M., SCHEUERMANN G.: Illustrative visualization of cardiac and aortic blood flow from 4D MRI data. In *Proc. PacificVis* (2013), IEEE Computer Sociey, Los Alamitos, pp. 129–136. doi: 10.1109/PacificVis.2013.6596137 18

[BPJG12] BRYDEN A., PHILLIPS JR. G., GLEICHER M.: Automated illustration of molecular flexibility. *IEEE Transactions on Visualization and Computer Graphics 18*, 1 (Jan. 2012), 132–145. doi: 10.1109/TVCG.2010.250 18

[BPMS12] BORN S., PFEIFLE M., MARKL M., SCHEUERMANN G.: Visual 4D MRI blood flow analysis with line predicates. In *Proc. PacificVis* (2012), IEEE Computer Society, Los Alamitos, pp. 105–112. doi: 10.1109/PacificVis.2012.6183580 18

[BSD09] BALZER M., SCHLÖMER T., DEUSSEN O.: Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Transactions on Graphics 28*, 3 (Aug. 2009), 86:1–86:8. doi: 10.1145/1531326.1531392 13

[BSM*07] BRESLAV S., SZERSZEN K., MARKOSIAN L., BARLA P., THOLLOT J.: Dynamic 2D patterns for shading 3D scenes. *ACM Transactions on Graphics 26*, 3 (July 2007), 20:1–20:6. doi: 10.1145/1275808.1276402 15, 16

[BSS04] BROSZ J., SAMAVATI F., SOUSA M. C.: Silhouette rendering based on stability measurement. In *Proc. SCCG* (2004), ACM, New York, pp. 157–167. doi: 10.1145/1037210.1037235 16

[BSW08] BELKIN M., SUN J., WANG Y.: Discrete Laplace operator on meshed surfaces. In *Proc. SCG* (2008), ACM, New York, pp. 278–287. doi: 10.1145/1377676.1377725 4

[BTBP07] BAER A., TIETJEN C., BADE R., PREIM B.: Hardware-accelerated stippling of surfaces derived from medical volume data. In *Proc. EuroVis* (2007), Eurographics Association, Goslar, Germany, pp. 235–242. doi: 10.2312/VisSym/EuroVis07/235-242 13, 16

[BTM06] BARLA P., THOLLOT J., MARKOSIAN L.: X-toon: An extended toon shader. In *Proc. NPAR* (2006), ACM, New York, pp. 127–132. doi: 10.1145/1124728.1124749 13, 14, 16

[BWF*10] BORN S., WIEBEL A., FRIEDRICH J., SCHEUERMANN G., BARTZ D.: Illustrative stream surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov./Dec. 2010), 1329–1338. doi: 10.1109/TVCG.2010.166 18

[CFM*13] CARNECKY R., FUCHS R., MEHL S., JANG Y., PEIKERT R.: Smart transparency for illustrative visualization of complex flow surfaces. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (May 2013), 838–851. doi: 10.1109/TVCG.2012.159 18

[CG07] CIPRIANO G., GLEICHER M.: Molecular surface abstraction. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov./Dec. 2007), 1608–1615. doi: 10.1109/TVCG.2007.70578 18

[CGL*08] COLE F., GOLOVINSKIY A., LIMPAECHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 88:1–88:11. doi: 10.1145/1360612.1360687 7, 17, 19

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proc. SIGGRAPH* (1993), ACM, New York, pp. 263–270. doi: 10.1145/166117.166151 11

[CMS99] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B.: *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Francisco, 1999. 2

[CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design 22*, 2 (Feb. 2005), 121–146. doi: 10.1016/j.cagd.2004.09.004 3

[CPJG09] CIPRIANO G., PHILLIPS JR. G. N., GLEICHER M.: Multi-scale surface descriptors. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov./Dec. 2009), 1201–1208. doi: 10.1109/TVCG.2009.168 18

[CS92] CHEN X., SCHMITT F.: Intrinsic surface properties from surface triangulation. In *Proc. ECCV* (1992), Springer, Berlin/Heidelberg, pp. 739–743. doi: 10.1007/3-540-55426-2_83 3

[CS15] CARDONA L., SAITO S.: Hybrid-space localized stylization method for view-dependent lines extracted from 3D models. In *Proc. NPAR* (2015), Eurographics Association, Goslar, Germany, pp. 79–89. doi: 10.2312/exp.20151181 6, 16

[CSD*09] COLE F., SANIK K., DECARLO D., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S., SINGH M.: How well do line drawings depict shape? *ACM Transactions on Graphics 28*, 3 (July 2009), 28:1–28:9. doi: 10.1145/1531326.1531334 17

[CSESS05] COSTA SOUSA M., EBERT D. S., STREDNEY D., SVAKHINE N. A.: Illustrative visualization for medical training. In *Proc. CAe* (2005), Eurographics Association, Goslar, Germany, pp. 201–208. doi: 10.2312/COMPAESTH/COMPAESTH05/201-208 18

[CSFWS03] COSTA SOUSA M., FOSTER K., WYVILL B., SAMAVATI F.: Precise ink drawing of 3D models. *Computer Graphics Forum 22*, 3 (Sept. 2003), 369–379. doi: 10.1111/1467-8659.00684 13, 16

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted Delaunay triangulations and normal cycle. In *Proc. SCG* (2003), ACM, New York, pp. 312–321. doi: 10.1145/777792.777839 3

[CSP03] COSTA SOUSA M., PRUSINKIEWICZ P.: A few good lines: Suggestive drawing of 3D models. *Computer Graphics Forum 22*, 3 (Sept. 2003), 381–390. doi: 10.1111/1467-8659.00685 10

[CYY*11] CHEN C.-K., YAN S., YU H., MAX N., MA K.-L.: An illustrative visualization framework for 3D vector fields. *Computer Graphics Forum 30*, 7 (Sept. 2011), 1941–1951. doi: 10.1111/j.1467-8659.2011.02064.x 18

[CZLX15] CHEN D., ZHANG Y., LIU H., XU P.: Real-time artistic silhouettes rendering for 3D models. In *Proc. Computational Intelligence and Design* (2015), vol. 1, IEEE Computer Society, Los Alamitos, pp. 494–498. doi: 10.1109/ISCID.2015.201 6, 16

[DÖ07] DÖLLNER J.: Non-photorealistic 3D geovisualization. In *Multimedia Cartography*, Cartwright W., Peterson M., Gartner G., (Eds.). Springer, Berlin/Heidelberg, 2007, pp. 229–240. doi: 10.1007/978-3-540-36651-5_16 18

[DC90] DOOLEY D., COHEN M. F.: Automatic illustration of 3D geometric models: Lines. *ACM SIGGRAPH Computer Graphics 24*, 2 (Mar. 1990), 77–82. doi: 10.1145/91394.91422 6, 16

[Dec96] DECAUDIN P.: *Cartoon-Looking Rendering of 3D-Scenes*. Tech. Rep. 2919, INRIA Rocquencourt, France, June 1996. 14

[DeC12] DECARLO D.: Depicting 3D shape using lines. In *Proceedings of Human Vision and Electronic Imaging XVII* (2012), vol. 8291 of *Proceedings of SPIE*, SPIE, Bellingham, Washington, pp. 829116:1–829116:16. doi: 10.1117/12.916463 5

[DFR04] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S.: Interactive rendering of suggestive contours with temporal coherence. In *Proc. NPAR* (2004), ACM, New York, pp. 15–145. doi: 10.1145/987657.987661 9

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Transactions on Graphics 22*, 3 (July 2003), 848–855. doi: 10.1145/882262.882354 7, 9

[DHR*99] DEUSSEN O., HAMEL J., RAAB A., SCHLECHTWEG S., STROTHOTTE T.: An illustration technique using intersections and skeletons. In *Proc. Graphics Interface* (1999), Morgan Kaufmann, San Francisco, pp. 175–182. doi: 10.20380/GI1999.23 12, 16

[DHvOS00] DEUSSEN O., HILLER S., VAN OVERVELD C., STROTHOTTE T.: Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3 (Sept. 2000), 41–50. doi: 10.1111/1467-8659.00396 13, 18

[DI13] DEUSSEN O., ISENBERG T.: Halftoning and stippling. In *Image and Video-Based Artistic Stylisation*, Rosin P., Collomosse J., (Eds.). Springer, London/Heidelberg, 2013, ch. 3, pp. 45–61. doi: 10.1007/978-1-4471-4519-6_3 12, 13

[DR07] DECARLO D., RUSINKIEWICZ S.: Highlight lines for conveying shape. In *Proc. NPAR* (2007), ACM, New York, pp. 63–70. doi: 10.1145/1274871.1274881 7, 9, 10, 14

[EBRI09] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Depth-dependent halos: Illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov. 2009), 1299–1306. doi: 10.1109/TVCG.2009.138 18

[EBRI11] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Illustrative line styles for flow visualization. In *Proc. PG* (2011), Eurographics Association, Goslar, Germany, pp. 105–110. doi: 10.2312/PE/PG/PG2011short/105-110 18

[EBRI15] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: *Interactive Illustrative Line Styles and Line Style Transfer Functions for Flow Visualization*. arXiv.org report 1503.05787, Mar. 2015. 18

[EC06] ELBER G., COHEN E.: Probabilistic silhouette based importance toward line-art non-photorealistic rendering. *The Visual Computer 22*, 9–11 (Sept. 2006), 793–804. doi: 10.1007/s00371-006-0065-8 12, 16

[Elb99] ELBER G.: Interactive line art rendering of freeform surfaces. *Computer Graphics Forum 18*, 3 (Sept. 1999), 1–12. doi: 10.1111/1467-8659.00322 12, 16

[ERB*12] EITZ M., RICHTER R., BOUBEKEUR T., HILDEBRAND K., ALEXA M.: Sketch-based shape retrieval. *ACM Transactions on Graphics 31*, 4 (July 2012), 31:1–31:10. doi: 10.1145/2185520.2185527 18

[FG98] FUHRMANN A., GRÖLLER E.: Real-time techniques for 3D flow visualization. In *Proc. Visualization* (1998), IEEE Computer Society, Los Alamitos, pp. 305–312. doi: 10.1109/VISUAL.1998.745317 18

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: A tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Dodgson N. A., Floater M. S., Sabin M. A., (Eds.). Springer, Berlin/Heidelberg, 2005, pp. 157–186. doi: 10.1007/3-540-26808-1_9 12

[FJL*16] FIŠER J., JAMRIŠKA O., LUKÁČ M., SHECHTMAN E., ASENTE P., LU J., SÝKORA D.: Stylit: Illumination-guided example-based stylization of 3D renderings. *ACM Transactions on Graphics 35*, 4 (2016), 92:1–92:11. doi: 10.1145/2897824.2925948 14, 16, 19

[Flo03] FLOATER M. S.: Mean value coordinates. *Computer Aided Geometric Design 20*, 1 (Mar. 2003), 19–27. doi: 10.1016/S0167-8396(03)00002-5 4

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proc. CVPR* (2016), IEEE Computer Society, Los Alamitos, pp. 2414–2423. doi: 10.1109/CVPR.2016.265 15, 16

[GG01] GOOCH B., GOOCH A. A.: *Non-Photorealistic Rendering*. A K Peters, Ltd., Natick, 2001. 2

[GGSC98] GOOCH A., GOOCH B., SHIRLEY P., COHEN E.: A non-photorealistic lighting model for automatic technical illustration. In *Proc. SIGGRAPH* (1998), ACM, New York, pp. 447–452. doi: 10.1145/280814.280950 14, 16

[GI04] GOLDFEATHER J., INTERRANTE V.: A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics 23*, 1 (Jan. 2004), 45–63. doi: 10.1145/966131.966134 3

[GI13] GERL M., ISENBERG T.: Interactive example-based hatching. *Computers & Graphics 37*, 1–2 (2013), 65–80. doi: 10.1016/j.cag.2012.11.003 11, 12, 16, 18

[GIHL00] GIRSHICK A., INTERRANTE V., HAKER S., LEMOINE T.: Line direction matters: An argument for the use of principal directions in 3D line drawings. In *Proc. NPAR* (2000), ACM, New York, pp. 43–52. doi: 10.1145/340916.340922 10

[GSG*99] GOOCH B., SLOAN P.-P. J., GOOCH A., SHIRLEY P., RIESENFELD R.: Interactive technical illustration. In *Proc. I3D* (1999), ACM, New York, pp. 31–38. doi: 10.1145/300523.300526 6, 16

[GTBP08] GASTEIGER R., TIETJEN C., BAER A., PREIM B.: Curvature- and model-based surface hatching of anatomical structures derived from clinical volume datasets. In *Proc. SmartGraphics* (2008), Springer, Berlin/Heidelberg, pp. 255–262. doi: 10.1007/978-3-540-85412-8_25 12, 16

[GTDS04] GRABLI S., TURQUIN E., DURAND F., SILLION F. X.: Programmable style for NPR line drawing. In *Proc. EGSR* (2004), Eurographics Association, Goslar, Germany, pp. 33–44. doi: 10.2312/EGWR/EGSR04/033-044 6, 16

[GTDS10] GRABLI S., TURQUIN E., DURAND F., SILLION F. X.: Programmable rendering of line drawing from 3D scenes. *ACM Transactions on Graphics 29*, 2 (Mar. 2010), 18:1–18:20. doi: 10.1145/1731047.1731056 6, 16

[GVH07] GOODWIN T., VOLLICK I., HERTZMANN A.: Isophote distance: A shading approach to artistic stroke thickness. In *Proc. NPAR* (2007), ACM, New York, pp. 53–62. doi: 10.1145/1274871.1274880 6, 9, 10

[HBP*07] HUANG J., BUE B., PATTATH A., EBERT D. S., THOMAS K. M.: Interactive illustrative rendering on mobile devices. *IEEE Computer Graphics and Applications 27*, 3 (May/June 2007), 48–56. doi: 10.1109/MCG.2007.63 18

[HCW10] HUANG C.-H., CHANG K.-C., WEN C.: Non-iterative stippling of greyscale threedimensional polygon meshed models. *IET Computer Vision 4*, 2 (June 2010), 138–148. doi: 10.1049/iet-cvi.2008.0051 18

[HCZW10] HAO W., CHE W., ZHANG X., WANG Y.: 3D model feature line stylization using mesh sharpening. In *Proc. Virtual-Reality Continuum and Its Applications in Industry* (2010), ACM, New York, pp. 249–256. doi: 10.1145/1900179.1900233 14, 16

[Hec90] HECKBERT P. S.: Adaptive radiosity textures for bidirectional ray tracing. *ACM Transactions on Graphics 24*, 4 (Aug. 1990), 145–154. doi: 10.1145/97880.97895 15, 16

[Her99] HERTZMANN A.: Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. In *SIGGRAPH Course Notes*. ACM, New York, 1999. 5, 6

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: IRIS: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov./Dec. 2010), 1319–1328. doi: 10.1109/TVCG.2010.173 17, 18

[HHD03] HILLER S., HELLWIG H., DEUSSEN O.: Beyond stippling – Methods for distributing objects on the plane. *Computer Graphics Forum 22*, 3 (Sept. 2003), 515–522. doi: 10.1111/1467-8659.00699 13, 16

[HJ05] HANSEN C. D., JOHNSON C. R. (Eds.): *The Visualization Handbook*. Elsevier, Amsterdam, 2005. doi: 10.1016/B978-012387582-2/50001-0 2

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *Proc. SIGGRAPH* (2001), ACM, New York, pp. 327–340. doi: 10.1145/383259.383295 15, 16

[HKG00] HLADŮVKA J., KÖNIG A., GRÖLLER E.: Curvature-based transfer functions for direct volume rendering. In *Proc. Spring Conference on Computer Graphics and its Applications* (2000), pp. 58–65. 14, 16

[Hod03] HODGES E. R. S.: *The Guild Handbook of Scientific Illustration*, 2nd ed. John Wiley & Sons, Hoboken, NJ, 2003. 1

[HP11] HILDEBRANDT K., POLTHIER K.: Generalized shape operators on polyhedral surfaces. *Computer Aided Geometric Design 28*, 5 (June 2011), 321–343. doi: 10.1016/j.cagd.2011.05.001 3

[HS03] HAMEIRI E., SHIMSHONI I.: Estimating the principal curvatures and the Darboux frame from real 3-D range data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B 33*, 4 (Aug. 2003), 626–637. doi: 10.1109/TSMCB.2003.814304 3

[HS13] HE L., SCHAEFER S.: Mesh denoising via L_0 minimization. *ACM Transactions on Graphics 32*, 4 (July 2013), 64:1–64:8. doi: 10.1145/2461912.2461965 18

[HSC12] HAJAGOS B., SZÉCSI L., CSÉBFALVI B.: Fast silhouette and crease edge synthesis with geometry shaders. In *Proc. SCCG* (2012), ACM, New York, pp. 71–76. doi: 10.1145/2448531.2448540 6, 16

[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. *International Journal of Computer Assisted Radiology and Surgery 5*, 2 (Mar. 2010), 133–141. doi: 10.1007/s11548-009-0365-3 2, 16, 18

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proc. SIGGRAPH* (2000), ACM, New York, pp. 517–526. doi: 10.1145/344779.345074 4, 6, 11, 16

[IB06] ISENBERG T., BRENNECKE A.: G-strokes: A concept for simplifying line stylization. *Computers & Graphics 30*, 5 (Oct. 2006), 754–766. doi: 10.1016/j.cag.2006.07.006 6, 16

[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. doi: 10.1109/MCG.2003.1210862 5, 6

[IFP95] INTERRANTE V., FUCHS H., PIZER S.: Enhancing transparent skin surfaces with ridge and valley lines. In *Proc. Visualization* (1995), IEEE Computer Society, Los Alamitos, pp. 52–59. doi: 10.1109/VISUAL.1995.480795 7

[IFP96] INTERRANTE V., FUCHS H., PIZER S.: Illustrating transparent surfaces with curvature-directed strokes. In *Proc. Visualization* (1996), IEEE Computer Society, Los Alamitos, pp. 211–218,487. doi: 10.1109/VISUAL.1996.568110 10

[IHS02] ISENBERG T., HALPER N., STROTHOTTE T.: Stylizing silhouettes at interactive rates: From silhouette edges to silhouette strokes. *Computer Graphics Forum 21*, 3 (Sept. 2002), 249–258. doi: 10.1111/1467-8659.00584 6, 16

[IIC*13] ISENBERG T., ISENBERG P., CHEN J., SEDLMAIR M., MÖLLER T.: A systematic review on the practice of evaluating visualization. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2818–2827. doi: 10.1109/TVCG.2013.126 16

[IMS00] ISENBERG T., MASUCH M., STROTHOTTE T.: 3D ilustrative effects for animating line drawings. In *Proc. IV* (2000), IEEE Computer Society, Los Alamitos, pp. 413–418. doi: 10.1109/IV.2000.859790 6

[INC*06] ISENBERG T., NEUMANN P., CARPENDALE S., SOUSA M. C., JORGE J. A.: Non-photorealistic rendering in context: An observational study. In *Proc. NPAR* (2006), ACM, New York, pp. 115–126. doi: 10.1145/1124728.1124747 13, 17

[Ise13] ISENBERG T.: Evaluating and validating non-photorealistic and illustrative rendering. In *Image and Video-Based Artistic Stylisation*, Rosin P., Collomosse J., (Eds.), vol. 42 of *Computational Imaging and Vision*. Springer, London/Heidelberg, 2013, pp. 311–331. doi: 10.1007/978-1-4471-4519-6_15 16

[Ise15] ISENBERG T.: A survey of illustrative visualization techniques for diffusion-weighted MRI tractography. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*, Hotz I., Schulz T., (Eds.). Springer, Berlin/Heidelberg, 2015, ch. 12, pp. 235–256. doi: 10.1007/978-3-319-15090-1_12 2, 3

[Ise16] ISENBERG T.: Interactive NPAR: What type of tools should we create? In *Proc. NPAR* (2016), The Eurographics Association, Goslar, Germany, pp. 89–96. doi: 10.2312/exp.20161067 2

[JDA07] JUDD T., DURAND F., ADELSON E.: Apparent ridges for line drawing. *ACM Transactions on Graphics 26*, 3 (July 2007), 19:1–19:8. doi: 10.1145/1276377.1276401 7, 9, 10

[JDD03] JONES T. R., DURAND F., DESBRUN M.: Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics 22*, 3 (July 2003), 943–949. doi: 10.1145/1201775.882367 18

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. In *Proc. EG Workshop on Visualization in Scientific Computing* (1997), Springer, Vienna, pp. 43–55. doi: 10.1007/978-3-7091-6876-9_5 19

[JNLM05] JEONG K., NI A., LEE S., MARKOSIAN L.: Detail control in line drawings of 3D meshes. *The Visual Computer 21*, 8 (Sept. 2005), 698–706. doi: 10.1007/s00371-005-0323-1 9, 10

[KAK*18] KLEIN T., AUTIN L., KOZLÍKOVÁ B., GOODSELL D. S., OLSON A., GRÖLLER M. E., VIOLA I.: Instant construction and visualization of crowded biological environments. *IEEE Transactions on Visualization and Computer Graphics 24* (2018). To appear. doi: 10.1109/TVCG.2017.2744258 18

[KCL09] KIM H. S., CHOI H. K., LEE K. H.: Feature detection of triangular meshes based on tensor voting theory. *Computer-Aided Design 41*, 1 (Jan. 2009), 47–58. doi: 10.1016/j.cad.2008.12.003 18

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics 25*, 3 (July 2006), 509–518. doi: 10.1145/1141911.1141916 13

[KCWI13] KYPRIANIDIS J. E., COLLOMOSSE J., WANG T., ISENBERG T.: State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (May 2013), 866–885. doi: 10.1109/TVCG.2012.160 2, 12

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. *ACM Transactions on Graphics 22*, 3 (July 2003), 856–861. doi: 10.1145/882262.882355 6, 16

[KH11] KARSCH K., HART J. C.: Snaxels on a plane. In *Proc. NPAR* (2011), ACM, New York, pp. 35–42. doi: 10.1145/2024676.2024683 16

[KMI*09] KIM S., MACIEJEWSKI R., ISENBERG T., ANDREWS W. M., CHEN W., SOUSA M. C., EBERT D. S.: Stippling by example. In *Proc. NPAR* (2009), ACM, New York, pp. 41–50. doi: 10.1145/1572614.1572622 13, 16

[KMM*02] KALNINS R. D., MARKOSIAN L., MEIER B. J., KOWALSKI M. A., LEE J. C., DAVIDSON P. L., WEBB M., HUGHES J. F., FINKELSTEIN A.: WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics 21*, 3 (July 2002), 755–762. doi: 10.1145/566570.566648 18

[KNS*09] KALOGERAKIS E., NOWROUZEZAHRAI D., SIMARI P., MCCRAE J., HERTZMANN A., SINGH K.: Data-driven curvature for real-time line drawing of dynamic scenes. *ACM Transactions on Graphics 28*, 1 (Feb. 2009), 11:1–11:13. doi: 10.1145/1477926.1477937 10

[KST08] KOLOMENKIN M., SHIMSHONI I., TAL A.: Demarcating curves for shape illustration. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 157:1–157:9. doi: 10.1145/1409060.1409110 7, 8, 18

[KST09] KOLOMENKIN M., SHIMSHONI I., TAL A.: On edge detection on surfaces. In *Proc. CVPR* (2009), IEEE Computer Society, Los Alamitos, pp. 2767–2774. doi: 10.1109/CVPR.2009.5206517 8

[KW07] KRÜGER J., WESTERMANN R.: Efficient stipple rendering. In *Proc. IADIS Computer Graphics and Visualization* (2007), ADIS, Lisbon, pp. 19–26. 13, 16

[KYM12] KWON Y., YANG H., MIN K.: Pencil rendering on 3D meshes using convolution. *Computers & Graphics 36*, 8 (Dec. 2012), 930–944. doi: 10.1016/j.cag.2012.08.002 11, 16

[KYYL08] KIM Y., YU J., YU X., LEE S.: Line-art illustration of dynamic and specular surfaces. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 156:1–156:10. doi: 10.1145/1409060.1409109 10, 16

[Law14] LAWONN K.: *Illustrative Visualization of Medical Data Sets*. PhD thesis, University of Magdeburg, Germany, 2014. 2

[LBSP14] LAWONN K., BAER A., SAALFELD P., PREIM B.: Comparative evaluation of feature line techniques for shape depiction. In *Proc. Vision, Modeling and Visualization* (2014), Eurographics Association, Goslar, Germany, pp. 31–38. doi: 10.2312/vmv.20141273 7, 17

[LGP13] LAWONN K., GASTEIGER R., PREIM B.: Qualitative evaluation of feature lines on anatomical surfaces. In *Proc. Bildverarbeitung für die Medizin* (2013), Springer, Berlin/Heidelberg, pp. 187–192. doi: 10.1007/978-3-642-36480-8_34 18

[LGP14] LAWONN K., GASTEIGER R., PREIM B.: Adaptive surface visualization of vessels with animated blood flow. *Computer Graphics Forum 33(8)* (Dec. 2014), 16–27. doi: 10.1111/cgf.12355 9, 10

[LGV*16] LAWONN K., GLASSER S., VILANOVA A., PREIM B., ISENBERG T.: Occlusion-free blood flow animation with wall thickness visualization. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan. 2016), 728–737. doi: 10.1109/TVCG.2015.2467961 18

[LHV06] LEE C. H., HAO X., VARSHNEY A.: Geometry-dependent lighting. *IEEE Transactions on Visualization and Computer Graphics 12*, 2 (Mar./Apr. 2006), 197–207. doi: 10.1109/TVCG.2006.30 15, 16

[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 (June 2014), 181–190. doi: 10.1111/cgf.12374 11, 16, 18

[LKL06] LEE H., KWON S., LEE S.: Real-time pencil rendering. In *Proc. NPAR* (2006), ACM, New York, pp. 37–45. doi: 10.1145/1124728.1124735 10, 16

[LLH17] LAWONN K., LUZ M., HANSEN C.: Improving spatial perception of vascular models using supporting anchors and illustrative visualization. *Computers & Graphics 63* (Apr. 2017), 37–49. doi: 10.1016/j.cag.2017.02.002 2, 18

[LLPH15] LAWONN K., LUZ M., PREIM B., HANSEN C.: Illustrative visualization of vascular models for static 2D representations. In *Proc. MICCAI* (2015), Springer International, Cham, Switzerland, pp. 399–406. doi: 10.1007/978-3-319-24571-3_48 6, 17, 18

[LLZ11] LUO T., LI R., ZHA H.: 3D line drawing for archaeological illustration. *International Journal of Computer Vision 94*, 1 (Aug. 2011), 23–35. doi: 10.1007/s11263-010-0394-y 18

[LM02] LUM E. B., MA K.-L.: Interactivity is the key to expressive visualization. *ACM SIGGRAPH Computer Graphics 36*, 3 (Aug. 2002), 5–9. doi: 10.1145/570332.570337 2

[LMAPV15] LE MUZIC M., AUTIN L., PARULEK J., VIOLA I.: cellVIEW: A tool for illustrative and multi-scale rendering of large biomolecular datasets. In *Proc. VCBM* (2015), Eurographics Association, Goslar, Germany, pp. 61–70. doi: 10.2312/vcbm.20151209 2, 18

[LMHB00] LAKE A., MARSHALL C., HARRIS M., BLACKSTEIN M.: Stylized rendering techniques for scalable real-time 3D animation. In *Proc. NPAR* (2000), ACM, New York, pp. 13–20. doi: 10.1145/340916.340918 10, 16

[LML16] LIU X., MA L., LIU Y.: Global tone: Using tone to draw in pen-and-ink illustration. *Multimedia Tools and Applications 76*, 10 (May 2016), 12853–12869. doi: 10.1007/s11042-016-3649-y 14

[LMLH07] LEE Y., MARKOSIAN L., LEE S., HUGHES J. F.: Line drawings via abstracted shading. *ACM Transactions on Graphics 26*, 3 (July 2007), 18:1–18:5. doi: 10.1145/1276377.1276400 8, 10, 13, 14

[LMP13] LAWONN K., MÖNCH T., PREIM B.: Streamlines for illustrative real-time rendering. *Computer Graphics Forum 32*, 3 (June 2013), 321–330. doi: 10.1111/cgf.12119 2, 12, 16

[LMT*03] LU A., MORRIS C. J., TAYLOR J., EBERT D. S., HANSEN C., RHEINGANS P., HARTNER M.: Illustrative interactive stipple rendering. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (April 2003), 127–138. doi: 10.1109/TVCG.2003.1196001 13, 16

[LP16] LAWONN K., PREIM B.: Feature lines for illustrating medical surface models: Mathematical background and survey. In *Visualization in Medicine in Life Sciences III*, Linsen L., Hamann B., Hege H.-C., (Eds.). Springer, Berlin/Heidelberg, 2016, pp. 93–132. doi: 10.1007/978-3-319-24523-2_5 2, 6, 7

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (July 2002), 362–371. doi: 10.1145/566654.566590 19

[LS07] LI L., SHEN H.-W.: Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (May 2007), 630–640. doi: 10.1109/TVCG.2007.1009 18

[LSHL16] LICHTENBERG N., SMIT N., HANSEN C., LAWONN K.: Sline: Seamless line illustration for interactive biomedical visualization. In *Proc. Visual Computing for Biology and Medicine* (2016), Eurographics Association, Goslar, Germany, pp. 133–142. doi: 10.2312/vcbm.20161281 11, 16, 18, 19

[LTH*02] LU A., TAYLOR J., HARTNER M., EBERT D. S., HANSEN C. D.: Hardware accelerated interactive stipple drawing of polygonal objects. In *Proc. VMV* (2002), Aka GmbH, pp. 61–68. 13, 16

[LTPH17] LAWONN K., TROSTMANN E., PREIM B., HILDEBRANDT K.: Visualization and extraction of carvings for heritage conservation. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan. 2017), 801–810. doi: 10.1109/TVCG.2016.2598603 18

[LUS13] LENGYEL Z., UMENHOFFER T., SZÉCSI L.: Screen space features for real-time hatching synthesis. In *Képfeldolgozók és Alakfelismerők Társaságának* (2013), pp. 82–94. 11, 16

[LUS14] LENGYEL Z., UMENHOFFER T., SZÉCSI L.: Realtime, coherent screen space hatching. In *Proc. Hungarian Computer Graphics and Geometry Conference* (2014), pp. 131–137. 11, 16

[LWM15] LOU L., WANG L., MENG X.: Stylized strokes for coherent line drawings. *Computational Visual Media 1*, 1 (Mar. 2015), 79–89. doi: 10.1007/s41095-015-0009-1 6, 16, 18

[LZL*12] LI L., ZHOU Y., LIU C., XU Y., FU J.: State-of-the-art line drawing techniques. In *Proc. CICA* (2012), Springer, Berlin/Heidelberg, pp. 1249–1257. doi: 10.1007/978-94-007-1839-5_135 5

[Mac49] MACNEAL R.: *The Solution of Partial Differential Equations by Means of Electrical Networks*. PhD thesis, California Institute of Technology, 1949. 4

[MALI11] MARTÍN D., ARROYO G., LUZÓN M. V., ISENBERG T.: Scale-dependent and example-based stippling. *Computers & Graphics 35*, 1 (Feb. 2011), 160–174. doi: 10.1016/j.cag.2010.11.006 13

[Mar76] MARR D.: Early processing of visual information. *Philosophical Transactions of the Royal Society of London B: Biological Sciences 275*, 942 (1976), 483–519. doi: 10.1098/rstb.1976.0090 5

[MARI17] MARTÍN D., ARROYO G., RODRÍGUEZ A., ISENBERG T.: A survey of digital stippling. *Computers & Graphics 67* (Oct. 2017), 24–44. doi: 10.1016/j.cag.2017.05.001 12, 13

[MBC02] MITCHELL J. L., BRENNAN C., CARD D.: Real-time image-space outlining for non-photorealistic rendering. In *Proc. SIGGRAPH* (2002), ACM, New York, pp. 239–239. doi: 10.1145/1242073.1242252 6, 16

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Proc. VisMath* (2002), Springer, Berlin/Heidelberg, pp. 35–57. doi: 10.1007/978-3-662-05105-4_2 3

[MH80] MARR D., HILDRETH E.: Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences 207*, 1167 (1980), 187–217. doi: 10.1098/rspb.1980.0020 9

[MH04] MCGUIRE M., HUGHES J. F.: Hardware-determined feature edges. In *Proc. NPAR* (2004), ACM, New York, pp. 35–47. doi: 10.1145/987657.987663 9

[Min13] MIN K.: Drawing features of 3D meshes in pencil-drawing style. *Software Engineering and its Applications 7*, 6 (Nov 2013), 359–366. doi: 10.14257/ijseia.2013.7.6.30 11, 16

[Min15] MIN K.: Feature-guided convolution for salient rendering of 3D meshes. *Engineering Systems Modelling and Simulation 7*, 1 (2015), 1–5. doi: 10.1504/IJESMS.2015.066122 11, 16

[MKG*97] MARKOSIAN L., KOWALSKI M. A., GOLDSTEIN D., TRYCHIN S. J., HUGHES J. F., BOURDEV L. D.: Real-time nonphotorealistic rendering. In *Proc. SIGGRAPH* (1997), ACM, New York, pp. 415–420. doi: 10.1145/258734.258894 6, 16

[MPFS03] MERUVIA PASTOR O., FREUDENBERG B., STROTHOTTE T.: Real-time animated stippling. *IEEE Computer Graphics and Applications 23*, 4 (July 2003), 62–68. doi: 10.1109/MCG.2003.1210866 13, 16

[MPS02] MERUVIA PASTOR O. E., STROTHOTTE T.: Frame-coherent stippling. In *Eurographics Short Presentations* (2002), Eurographics Association, Goslar, Germany, pp. 145–152. doi: 10.2312/egs.20021002 13, 16

[MPS04] MERUVIA PASTOR O., STROTHOTTE T.: Graph-based point relaxation for 3D stippling. In *Proc. Mexican International Conference in Computer Science* (2004), IEEE Computer Society, Los Alamitos, pp. 141–150. doi: 10.1109/ENC.2004.1342599 13, 16

[MPSV14] MUZIC M. L., PARULEK J., STAVRUM A.-K., VIOLA I.: Illustrative visualization of molecular reactions using omniscient intelligence and passive agents. *Computer Graphics Forum 33*, 3 (June 2014), 141–150. doi: 10.1111/cgf.12370 18

[MS98] MASUCH M., STROTHOTTE T.: Visualising ancient architecture using animated line drawings. In *Proc. Information Visualization* (1998), IEEE Computer Society, Los Alamitos, pp. 261–266. doi: 10.1109/IV.1998.694230 17

[MSVF09] MEDEIROS J., SOUSA M., VELHO L., FREITAS C. M. D. S.: Perspective contouring in illustrative visualization. In *Proc. Computer Graphics and Image Processing* (2009), IEEE Computer Society, Los Alamitos, pp. 48–55. doi: 10.1109/SIBGRAPI.2009.49 12, 16

[MWPV15] MUZIC M. L., WALDNER M., PARULEK J., VIOLA I.: Illustrative timelapse: A technique for illustrative visualization of particle simulations on the mesoscale level. In *Proc. PacificVis* (2015), IEEE Computer Society, Los Alamitos, pp. 247–254. doi: 10.1109/PACIFICVIS.2015.7156384 18

[ND04] NIENHAUS M., DÖLLNER J.: Blueprints – Illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering. In *Proc. Graphics Interface* (2004), CHCCS, Waterloo, ON, Canada, pp. 49–56. doi: 10.20380/GI2004.07 2

[ND05] NIENHAUS M., DÖLLNER J.: Blueprint rendering and sketchy drawings. In *GPU Gems 2*, Pharr M., (Ed.). Addison-Wesley, 2005, ch. 15, pp. 235–252. 18

[NIC07] NEUMANN P., ISENBERG T., CARPENDALE S.: NPR lenses: Interactive tools for non-photorealistic line drawings. In *Proc. Smart Graphics* (2007), vol. 4569 of *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, pp. 10–22. doi: 10.1007/978-3-540-73214-3_2 6

[NJLM06] NI A., JEONG K., LEE S., MARKOSIAN L.: Multi-scale line drawings from 3D meshes. In *Proc. Interactive 3D Graphics and Games* (2006), ACM, New York, pp. 133–137. doi: 10.1145/1111411.1111435 9, 10

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic silhouettes: A hybrid approach. In *Proc. NPAR* (2000), ACM, New York, pp. 31–37. doi: 10.1145/340916.340920 16

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 609–612. doi: 10.1145/1015706.1015768 7

[OJCJP16] OELTZE-JAFRA S., CEBRAL J. R., JANIGA G., PREIM B.: Cluster analysis of vortical flow in simulations of cerebral aneurysm hemodynamics. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan. 2016), 757–766. doi: 10.1109/TVCG.2015.2467203 18

[Pat16] PATANÉ G.: Laplacian spectral kernels and distances for geometry processing and shape analysis. *Computer Graphics Forum 35*, 2 (May 2016), 599–624. doi: 10.1111/cgf.12866 4

[PBC*16] PREIM B., BAER A., CUNNINGHAM D., ISENBERG T., ROPINSKI T.: A survey of perceptually motivated 3D visualization of medical image data. *Computer Graphics Forum 22*, 5 (June 2016), 501–525. doi: 10.1111/cgf.12927 2, 7, 16

[PCG*11] PAN B., CHEN X., GUO X., CHEN W., PENG Q.: Interactive expressive illustration of 3D city scenes. In *Proc. Computer-Aided Design and Computer Graphics* (2011), IEEE Computer Society, Los Alamitos, pp. 406–410. doi: 10.1109/CAD/Graphics.2011.61 18

[PDB*01] Pop M., Duncan C., Barequet G., Goodrich M., Huang W., Kumar S.: Efficient perspective-accurate silhouette computation and applications. In *Proc. Computational Geometry* (2001), ACM, New York, pp. 60–68. doi: 10.1145/378583.378618 16

[PFH00] Praun E., Finkelstein A., Hoppe H.: Lapped textures. In *Proc. SIGGRAPH* (2000), ACM, New York, pp. 465–470. doi: 10.1145/344779.344987 11

[PHWF01] Praun E., Hoppe H., Webb M., Finkelstein A.: Real-time hatching. In *Proc. SIGGRAPH* (2001), ACM, New York, pp. 581–586. doi: 10.1145/383259.383328 11, 12, 13, 16

[PKS*01] Page D. L., Koschan A., Sun Y., Paik J., Abidi M. A.: Robust crease detection and curvature estimation of piecewise smooth surfaces from triangle mesh approximations using normal voting. In *Proc. CVPR* (2001), IEEE Computer Society, Los Alamitos, pp. 162–167. doi: 10.1109/CVPR.2001.990471 3

[Ras01] Raskar R.: Hardware support for non-photorealistic rendering. In *Proc. Workshop on Graphics Hardware* (2001), ACM, New York, pp. 41–47. doi: 10.1145/383507.383525 16

[RBD06] Rusinkiewicz S., Burns M., DeCarlo D.: Exaggerated shading for depicting shape and detail. *ACM Transactions on Graphics 25*, 3 (July 2006), 1199–1205. doi: 10.1145/1141911.1142015 15, 16, 19

[RBGV08] Rautek P., Bruckner S., Gröller E., Viola I.: Illustrative visualization: New technology or useless tautology? *ACM SIGGRAPH Computer Graphics 42*, 3 (Aug. 2008), 4:1–4:8. doi: 10.1145/1408626.1408633 1, 2, 18

[RC99] Raskar R., Cohen M.: Image precision silhouette edges. In *Proc. I3D* (1999), ACM, New York, pp. 135–140. doi: 10.1145/300523.300539 16

[RC13] Rosin P., Collomosse J. (Eds.): *Image and Video based Artistic Stylisation*, vol. 42 of *Computational Imaging and Vision*. Springer, London/Heidelberg, 2013. doi: 10.1007/978-1-4471-4519-6 2

[RCDF08] Rusinkiewicz S., Cole F., DeCarlo D., Finkelstein A.: Line drawings from 3D models. In *ACM SIGGRAPH Classes* (2008), ACM, New York, pp. 39:1–39:356. doi: 10.1145/1401132.1401188 5

[RHD*06] Ritter F., Hansen C., Dicken V., Konrad-Verse O., Preim B., Peitgen H.-O.: Real-time illustration of vascular structures. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct. 2006), 877–884. doi: 10.1109/TVCG.2006.172 11, 12, 16, 18, 19

[RK00] Rössl C., Kobbelt L.: Line-art rendering of 3D-models. In *Proc. Pacific Graphics* (2000), IEEE Computer Society, Los Alamitos, pp. 87–96. doi: 10.1109/PCCGA.2000.883890 11, 16

[RKS00] Rössl C., Kobbelt L., Seidel H.-P.: Line art rendering of triangulated surfaces using discrete lines of curvature. In *Proc. WSCG* (2000), University of West Bohemia, Plzen, Czech Republic, pp. 168–175. doi: 11025/15452 12, 16

[RMC11] Rocha A., Miranda F. M., Celes W.: Illustrative volume visualization for unstructured meshes based on photic extremum lines. In *Proc. SIBGRAPI* (2011), IEEE Computer Society, Los Alamitos, pp. 101–108. doi: 10.1109/SIBGRAPI.2011.20 8

[RSB*13] Raab J., Schäfer H., Brost A., Stamminger M., Pfister M.: Non-photorealistic rendering for minimally invasive procedures. In *Proc. Medical Imaging* (2013), vol. 8671 of *SPIE Proceedings Series*, SPIE/IS&T, Bellingham, Washington. doi: 10.1117/12.2001385 18

[Rus89] Rustagi P.: Silhouette line display from shaded models. *IRIS Universe 1989*, 9 (Fall 1989), 42–44. 6, 16

[Rus04] Rusinkiewicz S.: Estimating curvatures and their derivatives on triangle meshes. In *Proc. 3DPVT* (Sept. 2004), IEEE Computer Society, Los Alamitos, pp. 486–493. doi: 10.1109/TDPVT.2004.1335277 3

[RvE92] Rossignac J. R., van Emmerik M.: Hidden contours on a frame-buffer. In *Proc. EGGH* (1992), Eurographics Association, Goslar, Germany, pp. 188–203. doi: 10.2312/EGGH/EGGH92/188-203 6, 16

[SBB13] Suarez J., Belhadj F., Boyer V.: GPU real time hatching. *Journal of WSCG 21*, 2 (June 2013), 97–105. doi: 11025/6972 11, 12, 16

[SBB17] Suarez J., Belhadj F., Boyer V.: Real-time 3D rendering with hatching. *The Visual Computer 33*, 10 (Oct. 2017), 1319–1334. doi: 10.1007/s00371-016-1222-3 12, 16

[SCBW14] Solomon J., Crane K., Butscher A., Wojtan C.: *A General Framework for Bilateral and Mean Shift Filtering*. arXiv.org report 1405.4734, Apr. 2014. 4

[Sec02] Secord A.: Weighted Voronoi stippling. In *Proc. NPAR* (2002), ACM, New York, pp. 37–43. doi: 10.1145/508530.508537 13

[SEI10] Svetachov P., Everts M. H., Isenberg T.: DTI in context: Illustrating brain fiber tracts in situ. *Computer Graphics Forum 29*, 3 (June 2010), 1024–1032. doi: 10.1111/j.1467-8659.2009.01692.x 12

[SH95] Stalling D., Hege H.-C.: Fast and resolution independent line integral convolution. In *Proc. SIGGRAPH* (1995), ACM, New York, pp. 249–256. doi: 10.1145/218380.218448 11

[SHS02] Secord A., Heidrich W., Streit L.: Fast primitive distribution for illustration. In *Proc. EGWR* (2002), Eurographics Association, Goslar, Germany, pp. 215–226. doi: 10.2312/EGWR/EGWR02/215-226 13

[SID17] Semmo A., Isenberg T., Döllner J.: Neural style transfer: A paradigm shift for image-based artistic rendering? In *Proc. NPAR* (2017), ACM, New York, pp. 5:1–5:13. doi: 10.1145/3092919.3092920 15

[SJEG05] Svakhine N. A., Jang Y., Ebert D. S., Gaither K.: Illustration and photography inspired visualization of flows and volumes. In *Proc. Vis* (2005), IEEE Computer Society, Los Alamitos, pp. 687–694. doi: 10.1109/VIS.2005.53 18

[SMGG01] Sloan P.-P., Martin W., Gooch A., Gooch B.: The lit sphere: A model for capturing NPR shading from art. In *Proc. Graphics Interface* (2001), CIPS, Toronto, pp. 143–150. doi: 10.20380/GI2001.17 14, 16, 19

[SMI99] Strothotte T., Masuch M., Isenberg T.: Visualizing knowledge about virtual reconstructions of ancient architecture. In *Proc. Computer Graphics International* (1999), IEEE Computer Society, Los Alamitos, pp. 36–43. doi: 10.1109/CGI.1999.777901 2, 17

[SNEA*16] Schulz C., Nocaj A., El-Assady M., Frey S., Hlawatsch M., Hund M., Karch G. K., Netzel R., Schätzle C., Butt M., Keim D. A., Ertl T., Brandes U., Weiskopf D.: Generative data models for validation and evaluation of visualization techniques. In *Proc. BELIV* (2016), ACM, New York, pp. 112–124. doi: 10.1145/2993901.2993907 19

[Sor05] Sorkine O.: Laplacian mesh processing. In *Eurographics – State of the Art Reports* (2005), Eurographics Association, Goslar, Germany, pp. 53–70. doi: 10.2312/egst.20051044 4

[SPR*94] Strothotte T., Preim B., Raab A., Schumann J., Forsey D. R.: How to render frames and influence people. *Computer Graphics Forum 13*, 3 (Aug. 1994), 455–466. doi: 10.1111/1467-8659.1330455 6, 16

[SS02] Strothotte T., Schlechtweg S.: *Non-Photorealistic Computer Graphics. Modeling, Animation, and Rendering*. Morgan Kaufmann Publishers, San Francisco, 2002. doi: 10.1016/B978-1-55860-787-3.50019-0 2

[SS10] Singh M., Schaefer S.: Suggestive hatching. In *Proc. CAe* (2010), Eurographics Association, Goslar, Germany, pp. 25–32. doi: 10.2312/COMPAESTH/COMPAESTH10/025-032 12, 16, 18

[SSK16] Szécsi L., Szirányi M., Kacsó A.: Tonal art maps with image space strokes. In *Eurographics Posters* (2016), The Eurographics Association. doi: 10.2312/egp.20161046 12, 16

[ST90] Saito T., Takahashi T.: Comprehensible rendering of 3-D shapes. *ACM SIGGRAPH Computer Graphics 24*, 4 (Sept. 1990), 197–206. doi: 10.1145/97880.97901 6, 16

[STKD12] Semmo A., Trapp M., Kyprianidis J. E., Döllner J.: Interactive visualization of generalized virtual 3D city models using level-of-abstraction transitions. *Computer Graphics Forum 31*, 3 (June 2012), 885–894. doi: 10.1111/j.1467-8659.2012.03081.x 18

[SW04] Sweet G., Ware C.: View direction, surface orientation and texture orientation for perception of surface shape. In *Proc. Graphics*

Interface (2004), CHCCS, Waterloo, ON, Canada, pp. 97–106. doi: 10.20380/GI2004.13 10

[Tau95a] TAUBIN G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. ICCV* (1995), IEEE Computer Society, Los Alamitos, pp. 902–907. doi: 10.1109/ICCV.1995.466840 3

[Tau95b] TAUBIN G.: A signal processing approach to fair surface design. In *Proc. SIGGRAPH* (1995), ACM, New York, pp. 351–358. doi: 10.1145/218380.218473 4

[TIP05] TIETJEN C., ISENBERG T., PREIM B.: Combining silhouettes, surface, and volume rendering for surgery education and planning. In *Proc. EuroVis* (2005), Eurographics Association, Goslar, Germany, pp. 303–310. doi: 10.2312/VisSym/EuroVis05/303-310 2, 16, 17, 18

[TPB*08] TIETJEN C., PFISTERER R., BAER A., GASTEIGER R., PREIM B.: Hardware-accelerated illustrative medical surface visualization with extended shading maps. In *Proc. Smart Graphics* (2008), Springer, Berlin/Heidelberg, pp. 166–177. doi: 10.1007/978-3-540-85412-8_15 15, 16, 19

[VBGS08] VERGNE R., BARLA P., GRANIER X., SCHLICK C.: Apparent relief: A shape descriptor for stylized shading. In *Proc. NPAR* (2008), ACM, New York, pp. 23–29. doi: 10.1145/1377980.1377987 14, 16

[VBTS07] VANDERHAEGHE D., BARLA P., THOLLOT J., SILLION F. X.: Dynamic point distribution for stroke-based rendering. In *Rendering Techniques* (2007), Eurographics Association, Goslar, Germany, pp. 139–146. doi: 10.2312/EGWR/EGSR07/139-146 13

[vdZLBI11] VAN DER ZWAN M., LUEKS W., BEKKER H., ISENBERG T.: Illustrative molecular visualization with continuous abstraction. *Computer Graphics Forum 30*, 3 (June 2011), 683–690. doi: 10.1111/j.1467-8659.2011.01917.x 18

[VGH*05] VIOLA I., GRÖLLER M. E., HADWIGER M., BÜHLER K., PREIM B., SOUSA M. C., EBERT D., STREDNEY D.: Illustrative visualization. In *IEEE Visualization Tutorials*. IEEE Computer Society, Los Alamitos, 2005, ch. 4. 3

[VHE10] VIOLA I., HAUSER H., EBERT D.: Editorial note for special section on illustrative visualization. *Computers & Graphics 34*, 4 (Aug. 2010), 335–336. doi: 10.1016/j.cag.2010.05.011 2

[VI18] VIOLA I., ISENBERG T.: Pondering the concept of abstraction in (illustrative) visualization. *IEEE Transactions on Visualization and Computer Graphics 24* (2018). To appear. doi: 10.1109/TVCG.2017.2747545 1, 2, 3

[VPB*09] VERGNE R., PACANOWSKI R., BARLA P., GRANIER X., SCHLICK C.: Light warping for enhanced surface depiction. *ACM Transactions on Graphics 28*, 3 (Aug. 2009), 25:1–25:8. doi: 10.1145/1531326.1531331 15, 16

[VPB*11] VERGNE R., PACANOWSKI R., BARLA P., GRANIER X., SHLICK C.: Improving shape depiction under arbitrary rendering. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (Aug. 2011), 1071–1081. doi: 10.1109/TVCG.2010.252 15, 16

[vPOBB*10] VAN PELT R., OLIVÁN BESCOS J., BREEUWER M., CLOUGH R. E., GRÖLLER M. E., TER HAAR ROMENIJ B., VILANOVA A.: Exploration of 4D MRI blood flow using stylistic visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov. 2010), 1339–1347. doi: 10.1109/TVCG.2010.153 18

[VPVVDW08] VAN PELT R. F. P., VILANOVA A., VAN DE WETERING H. M. M.: GPU-based particle systems for illustrative volume rendering. In *Proc. Eurographics/IEEE VGTC Point-Based Graphics* (2008), Eurographics Association, Goslar, Germany, pp. 89–96. doi: 10.2312/VG/VG-PBG08/089-096 10

[VVP*16] VÁŠA L., VANĚČEK P., PRANTL M., SKORKOVSKÁ V., MARTÍNEK P., KOLINGEROVÁ I.: Mesh statistics for robust curvature estimation. *Computer Graphics Forum 35*, 3 (Aug. 2016), 271–280. doi: 10.1111/cgf.12982 3

[WB01] WATANABE K., BELYAEV A. G.: Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum 20*, 3 (Sept. 2001), 385–392. doi: 10.1111/1467-8659.00531 8

[WBE*05] WEISKOPF D., BORCHERS M., ERTL T., FALK M., FECHTIG O., FRANK R., GRAVE F., KING A., KRAUS U., MÜLLER T., NOLLERT H. P., MENDEZ I. R., RUDER H., SCHAFHITZEL T., SCHÄR S., ZAHN C., ZATLOUKAL M.: Visualization in the Einstein year 2005: A case study on explanatory and illustrative visualization of relativity and astrophysics. In *Proc. Visualization* (2005), IEEE Computer Society, Los Alamitos, pp. 583–590. doi: 10.1109/VISUAL.2005.1532845 18

[WBE*06] WEISKOPF D., BORCHERS M., ERTL T., FALK M., FECHTIG O., FRANK R., GRAVE F., KING A., KRAUS U., MÜLLER T., NOLLERT H.-P., RICA MENDEZ I., RUDER H., SCHAFHITZEL T., SCHÄR S., ZAHN C., ZATLOUKAL M.: Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (July 2006), 522–534. doi: 10.1109/TVCG.2006.69 18

[WMKG07] WARDETZKY M., MATHUR S., KAELBERER F., GRINSPUN E.: Discrete Laplace operators: No free lunch. In *Geometry Processing* (2007), pp. 33–37. doi: 10.2312/SGP/SGP07/033-037 4

[WPFH02] WEBB M., PRAUN E., FINKELSTEIN A., HOPPE H.: Fine tone control in hardware hatching. In *Proc. NPAR* (2002), ACM, New York, pp. 53–59. doi: 10.1145/508530.508540 11, 16

[WYP*15] WEI M., YU J., PANG W. M., WANG J., QIN J., LIU L., HENG P. A.: Bi-normal filtering for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics 21*, 1 (Jan 2015), 43–55. doi: 10.1109/TVCG.2014.2326872 4, 18

[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* (Nov./Dec. 2007), 1328–1335. doi: 10.1109/TVCG.2007.70538 7, 8, 10, 14

[XNYC04] XU H., NGUYEN M. X., YUAN X., CHEN B.: Interactive silhouette rendering for point-based models. In *Proc. Point-Based Graphics* (2004), Eurographics Association, Goslar, Germany, pp. 13–18. doi: 10.2312/SPBG/SPBG04/013-018 16

[YBS05] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: Fast and robust detection of crest lines on meshes. In *Proc. Solid and Physical Modeling* (2005), ACM, New York, pp. 227–232. doi: 10.1145/1060244.1060270 8

[YBYS07] YOSHIZAWA S., BELYAEV A. G., YOKOTA H., SEIDEL H.-P.: Fast and faithful geometric algorithm for detecting crest lines on meshes. In *Proc. Pacific Graphics* (2007), IEEE Computer Society, Los Alamitos, pp. 231–237. doi: 10.1109/PG.2007.24 8

[YBYS08] YOSHIZAWA S., BELYAEV A., YOKOTA H., SEIDEL H.-P.: Fast, robust, and faithful methods for detecting crest lines on meshes. *Computer Aided Geometric Design 25*, 8 (Nov. 2008), 545–560. doi: 10.1016/j.cagd.2008.06.008 8

[YNZC05] YUAN X., NGUYEN M. X., ZHANG N., CHEN B.: Stippling and silhouettes rendering in geometry-image space. In *Rendering Techniques* (2005), Eurographics Association, Goslar, Germany, pp. 193–200. doi: 10.2312/EGWR/EGSR05/193-200 13, 16

[YWQ*14] YU J., WEI M., QIN J., WU J., HENG P.-A.: Feature-preserving mesh denoising via normal guided quadric error metrics. *Optics and Lasers in Engineering 62* (Nov. 2014), 57–68. doi: 10.1016/j.optlaseng.2014.05.002 18

[ZDZ*15] ZHANG W., DENG B., ZHANG J., BOUAZIZ S., LIU L.: Guided mesh normal filtering. *Computer Graphics Forum 34*, 7 (Oct. 2015), 23–34. doi: 10.1111/cgf.12742 4

[ZHS10] ZHANG L., HE Y., SEAH H. S.: Real-time computation of photic extremum lines (PELs). *The Visual Computer 26*, 6-8 (June 2010), 399–407. doi: 10.1007/s00371-010-0454-x 8

[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*, 7 (July 2011), 993–1006. doi: 10.1109/TVCG.2010.118 7, 9, 10

[ZISS04] ZANDER J., ISENBERG T., SCHLECHTWEG S., STROTHOTTE T.: High quality hatching. *Computer Graphics Forum 23*, 3 (Sept. 2004), 421–430. doi: 10.1111/j.1467-8659.2004.00773.x 12, 16

[ZS04] ZAKARIA N., SEIDEL H.-P.: Interactive stylized silhouette for point-sampled geometry. In *Proc. GRAPHITE* (2004), ACM, New York, pp. 242–249. doi: 10.1145/988834.988876 16

[ZXY*12] ZHANG L., XIA J., YING X., HE Y., MUELLER-WITTIG W., SEAH H.-S.: Efficient and robust 3D line drawings using difference-of-Gaussian. *Graph. Models 74*, 4 (July 2012), 87–98. doi: 10.1016/j.gmod.2012.03.006 9