

# Illustrative Rendering of Segmented Anatomical Data

Zein Salah\*, Dirk Bartz†, and Wolfgang Straßer‡  
WSI/GRIS - VCM, Universität Tübingen  
Sand 14, 72076 Tübingen, Germany

## Abstract

Medical illustrations use simple drawing styles to demonstrate shapes and features of organs and organ systems, while omitting irrelevant details. In this contribution, we present a non-photorealistic rendering algorithm for illustrating anatomical data. Unlike other existing methods that rely on available triangulations of the organs to be rendered, our approach relies only on a subset of surface points, together with their normals. The advantage is that surface points and normals can be quite easily extracted from medical data, without an intermediate triangulation.

## 1 Introduction

Anatomical illustrations serve an important purpose for education in medical professions. They accurately depict the characteristics of organs and organ systems within the human body, while at the same time remove details that are not necessary for the understanding of their shapes and functions. Anatomical illustrations go back to early anatomic studies (eg. by Leonardo da Vinci) and are part of every anatomical text book [GC95]. Unfortunately, they heavily depend on the artistic skills of the medical illustrator.

With the introduction of photography and modern scanning methods, new techniques became available for the representation of anatomy [HPP<sup>+</sup>00]. However, these techniques frequently provide too many details and too complex lighting operations that can block the view of the essentials of the organ systems.

Non-photorealistic rendering (NPR) in contrast provides means for enriching the repertoire of computer graphics techniques with expressive methods. In particular, NPR is a useful alternative to conventional volume or surface rendering for medical illustrations, since it highlights relevant information better than photorealistic-rendered images by focussing on the shape and features of the targeted object, thus mimicking traditional illustration techniques.

In this paper, we introduce an algorithm for anatomical data illustration. Our approach belongs to the class of silhouette algorithms, and is based on surface points, extracted from medical volume datasets, where the data is arranged on a cartesian grid. Only a subset of

---

\*salah@gris.uni-tuebingen.de

†bartz@gris.uni-tuebingen.de

‡strasser@gris.uni-tuebingen.de

generated surface points and their normals are needed for silhouette estimation and rendering. Halftoning is adopted for producing illumination and shading effects. Although we only focus on anatomical illustrations, the presented ideas and techniques are equally well applicable to other objects, if a sufficiently covering surface point set representation is available.

The remainder of this paper discusses these issues in details. Section 2 provides an overview of related work and Section 3 describes the various steps of the rendering algorithm. Experimental results are demonstrated in Section 4. Finally, Section 5 summarizes our approach and describes areas of future work.

## 2 Related Work

One goal of scientific visualization is to provide visual representations that allow to efficiently grasp the shape and features of objects (eg. an organ). NPR has proved successful applicability in this domain, where the general idea is to exploit artistic techniques in generating approximate illustrations based on 2D images as well as volume datasets. Rheingans and Ebert [RE01] introduce *volume illustration*, where the familiarity of a physics-based volume rendering model is combined with the ability to enhance important features using NPR techniques. Kindlmann et al. [KWTM03] utilize curvature-based transfer functions to enhance the expressiveness in non-photorealistic volume rendering. Volume rendering with silhouette enhancement of targeted objects was also presented by Csébfalvi et al. [CMH<sup>+</sup>01]. Hadwiger et al. [HBH03] used a similar method for volume rendering of segmented datasets. Lu et al. [LME02] present a framework for an interactive direct volume illustration system that simulates traditional stipple drawing. Volumetric hatching, introduced by Dong et al. [DCLK03], extracts silhouette points from volumetric data, projects them on the final image, and generate 2D lines connecting these projections.

Some methods that use hardware-accelerated rendering have also been introduced. In their rendering algorithm, Raskar and Cohen [RC99] use traditional z-buffering along with back-and front face culling for automatically determining polygons that define the silhouette.

Using point-based representations as rendering primitives has been introduced by Levoy and Whitted [LW85] and has gained an increasing interest. Zwicker et al. [ZPvBG01] extended this concept for high quality image reconstruction from point clouds. In [CLL<sup>+</sup>88], points with attributes are used to render medical data. The points are essentially generated on the voxel grid, similar to the triangle vertices generated by the Marching Cubes algorithm [LC87].

Xu et al. [XNYC04] have recently presented an approach for interactive rendering of silhouette for point-based models. An adaptation of this approach is used for abstraction and rendering of 3D scanned outdoor environments [XC04]. In [PKG03], line-type features are extracted from point-sampled surfaces using principal component analysis and used to create line drawings. Deussen and Strothotte [DS00] introduced an algorithm for generating pen-and-ink illustration of trees. In their method, the foliage of a tree is represented as a set of particles, which are rendered by randomly oriented disks. Alexa et al. [ABCO<sup>+</sup>01] presented an approach for rendering point set surfaces at different resolution using repre-

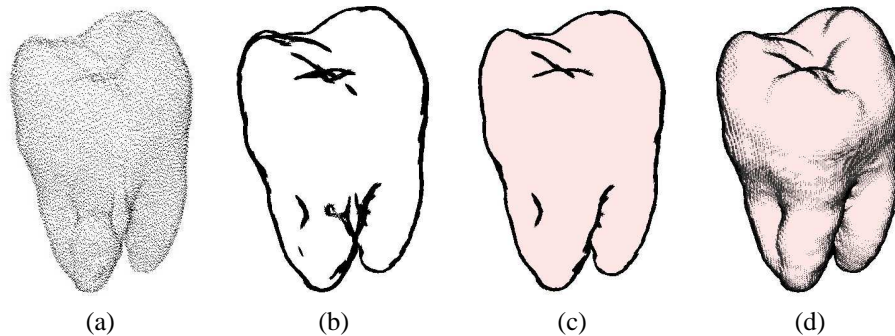


Figure 1: NPR-Method: (a) Surface points, (b) silhouette detection and rendering, (c) hidden silhouette parts are removed, (d) final shaded rendering.

sending polynomials.

Several other artistic techniques have been utilized to create non-photorealistic renderings. Stroke-based Rendering (SBR) is perhaps among the most common techniques [Her98, KGC00]. Other proposed algorithms and styles include painting [Her98], pen-and-ink drawing, tile mosaics [Hau01], and stippling [MPFS03].

In rendering 3D objects, hatching is frequently used to add more impression for NP-rendered surfaces or volumes, where different alternatives are applied for determining hatching directions. Some methods rely on material properties in the original volume datasets, for example the direction of muscle fibers [DCLK03]. Other surface-NPR methods estimate the hatching direction by computing the surface principal curvatures [HZ00]. Unlike most other methods, our approach does not rely on existing polygonal meshes (triangulations) of the organ to be illustrated. The input of the algorithm is a subset of surface points together with the corresponding normals, which we extract directly from segmented voxel data of the targeted organ, without generating a surface representation. Since vertex/normal data is inherently available in triangulations, our algorithm can be applied equally well - with minor adaptation - to triangulated surfaces.

### 3 Rendering Algorithm

Our approach belongs to the class of silhouette algorithms. The silhouette of an object is typically defined as the set of points on the object's surface where the surface normals are perpendicular to the view vector. Silhouette-based algorithms principally perform two steps to illustrate an object: first, the silhouette of the object is detected and represented using one of the several available methods. Second, visibility culling approaches are introduced afterwards to hide invisible parts of the silhouette. Isenberg et al. [IFH<sup>+</sup>03] provide a developer's guideline for silhouette algorithms that apply to polygonal models.

Our algorithm renders an object relying on its surface points. In other words, the input to

the algorithm is a set of surface points along with their corresponding normals. The output is an ink- or coal-like illustration of the object. In short, the whole pipeline performs the following steps: A set of surface points is first extracted, object silhouette is estimated and rendered, hidden parts of the silhouette are removed, and lighting effects are finally added.

### 3.1 Surface Points Extraction

The way in which surface points are extracted depends on the form in which the targeted object is available. In the case of anatomical data, it is usually provided by a binary segmentation of an organ. These segmentations are based on scanned volume datasets (CT and MRI scans). Surface points are then defined by uniformly selecting a subset of contour points (in 3D context). If  $S$  denotes the segmented voxel data, we apply volumetric binary morphological erosion on  $S$  with radius = 1, i.e., a  $3 \times 3 \times 3$  structuring element, producing  $S_E$  (eroded version of  $S$ ). With  $C = S - S_E$  (difference image), contour points are all non-zero points in  $C$ . This is equivalent to extracting a one-voxel thin shell of surface voxels (the outer skin of the object). Normals are acquired by computing the gradient of the binary image  $S$  at the surface points. Figure 1a shows a set of extracted points for a tooth. These extracted surface points and computed normals are fed to the software tool as an *initial* point set. In order to reduce the points and render load, the initial set can be uniformly sub-sampled.

If the 3D polygonal model of the object (a triangulation) is already available, the extraction of surface points is straightforward. The vertices of the mesh or the center points of triangles can be chosen. In both cases, the normals can be computed easily.

### 3.2 Silhouette Estimation and Rendering

Silhouettes are used to depict object outlines. The subset of surface points that defines the silhouette ( $P_{Sil}$ ) of the object is determined by examining the normals of all surface points.  $P_{Sil}$  is defined as the set of all points  $p_i$  satisfying:

$$|n_i \cdot (x_i - VP)| \leq T_{Sil} \quad (1)$$

where  $n_i$  is the normal at the point  $i$ ,  $x_i$  is its position,  $VP$  is the view point, and  $T_{Sil}$  is a threshold value controlling the tolerance of choosing silhouette points. For an absolute silhouette,  $T_{Sil}$  must be 0.

The rendering element in our algorithms is a disk, i.e., a filled circle. Each silhouette point in  $P_{Sil}$  is rendered with a disk of radius  $r_{disk}$  with a chosen silhouette color, centered at the point position, and oriented such that the point normal is perpendicular to its surface, resulting in an ellipsoid on the screen (see Figure 2a).

Obviously, points that lie *exactly* at the silhouette (the dot product in Equation 1 is exactly zero) will be rendered with disks that are parallel to the viewing direction, i.e., the ellipsoids will be infinitely thin. Therefore, the tolerance  $T_{Sil}$  is chosen within a small range around zero so that more disks are rendered to produce the required effect (for most of the models we rendered, a tolerance  $T_{Sil} = 0.2$  generated good silhouettes). In other words, the tolerance  $T_{Sil}$  controls the thickness of the rendered silhouette. Rendering all silhouette

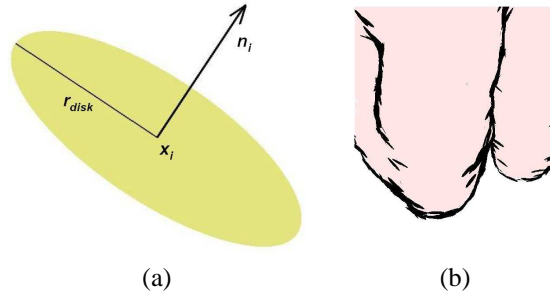


Figure 2: Silhouette Rendering: (a) a disk as a drawing element, (b) The effect of rendering silhouette points using properly oriented disks

disks will produce the overall effect of pen-stroke drawing (see Figure 1b). A zoomed view of the tooth roots is shown in Figure 2b.

For producing convincing renderings, the input points should be dense enough and uniformly covers the surface of the rendered object. If input points are not sufficiently distributed, in particular in silhouette regions, a noncontinuous silhouette is rendered. This problem mainly arises when the input point set is extracted from a triangulation of the object, where the surface is usually triangulated less finely in regions of low curvature. However, in our application, we extract points directly from a segmentation of the object (we actually extract surface voxels), which provides a quite dense point set that uniformly covers the object surface.

### 3.3 Hidden Silhouette Removal

We actually render disks at each non-silhouette surface point with a chosen color for the object and shift silhouette disks slightly in their normal directions. As shown in Figure 1c, non-silhouette disks, rendered with a proper scaling factor  $S_{all}$ , will occlude hidden silhouettes.  $S_{all}$  is adjusted so that the occluding disks sufficiently overlap and no holes appear (even when the user zooms in). However, the overall rendering of occluding disks may still permit very few (still very small) pieces of the hidden silhouette to appear. Since the goal is a non-photorealistic illustration, this possible effect is negligible and the resulting rendering is quite acceptable. However, this effect can be controlled by increasing the scaling factor  $S_{all}$ . Since this increase may cause some of the visible lines to disappear, silhouette disks are tuned with an additional scaling factor  $S_{sil}$  which allows for further controlling the size of silhouette disks to compensate for the disappearing parts.

Such artifacts in hidden silhouette removal arise usually in regions where the surface points are too sparse. In this context, please note that the availability of a dense and uniformly distributed input point set is important also for this step to succeeds. As stated in the previous section, in our application, the input points resemble extracted surface voxels of the object, which ensures, to a great extent, that the mentioned artifacts very rarely arise.

### 3.4 Shading and Illumination

We implement lighting effects by assuming a light source with a diffuse component and adopting Phong illumination model. For simplicity, we assumed an attenuation factor of 1 and ignored specular reflection. If  $LP$  is the light position, for each non-silhouette surface point, we define the light intensity as:

$$L_i = n_i \cdot (x_i - LP) \quad (2)$$

The implemented shading is also using our rendering elements, the solid disks. Gradual lighting is implemented by adapting halftoning techniques [HH94, SS02]. Usually, modern graphical output devices implement halftoning using rectangular pixel regions, called *half-tone patterns*. Different tones are produced by setting different number of pixels in the pattern. In other words, the more set pixels, the darker the pattern is. Alternatively, different intensity values can be obtained by drawing a filled circle of different radii. We adopted this alternative for producing gradual shading intensities. We render disks whose radii are inversely proportional to the illumination intensity of the corresponding area. More precisely, we scale a disk with a scaling factor  $S_{shd}$  defined as:

$$S_{shd} = I_{shd} \frac{A_{shd} - L_i}{S_{all}} \quad (3)$$

where the parameter  $I_{shd}$  controls the overall intensity (darkness) of shading,  $A_{shd} \in [0, 2]$  controls the area of shaded regions, i.e., the stretching of the shading. A value of  $A_{shd} = 2$  will stretch the shading to covers the whole model.  $L_i$  is the illumination intensity defined in Equation 2, and  $S_{all}$  is the global scaling factor used for all rendering and culling disks (Sections 3.2 and 3.3).

## 4 Implementation and Experimental Results

We implemented and tested our algorithm using C++ and OpenGL. We also developed a Qt-based user interface to test the effect of different values of parameters controlling the algorithm behavior. Volume processing operations, discussed in Section 3.1 were implemented using ITK filters [ISNC03]. Mainly, the erosion was performed with the *itkBinaryErodeImageFilter*. Computation of the normals is principally based on the computation of the gradient of the segmented image. This was performed by combining the results of applying the *itkDerivativeImageFilter* in each of the x, y, and z-directions. Rendering time of the algorithms is linearly proportional to the model complexity. On a Pentium 4 PC (2.6 GHz, ATI Radeon 9600 graphic card), the algorithms runs in interactive time. For most models, we achieved more than 5 fps.

In Figure 1, we used an example of rendering a tooth to demonstrated how the algorithm works. The tooth model points were extracted from a segmented CT scan [NLM]. In Figure 3, renderings of different organs are depicted, where the surface points and normals of the brain (left) and the spine (bottom) were extracted from available triangulations, based on MRI and CT scans. Surface points of the cerebral ventricular system (right) were extracted directly from a segmented dataset measured by an MRI scanner.

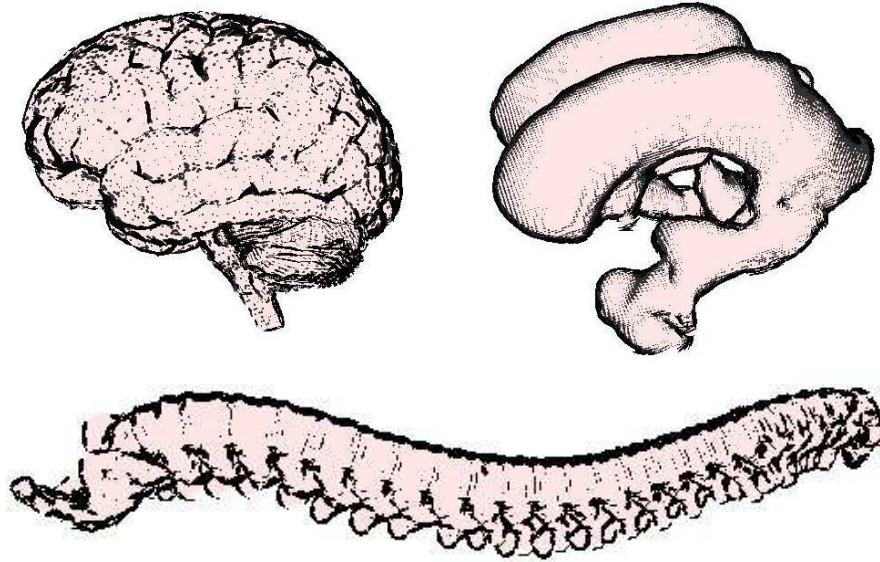


Figure 3: Examples of anatomical data rendered with our algorithms.

In Figure 4, two front view renderings of the colon are demonstrated. The rendering on the right shows that if sufficient surface points are available, our algorithm can depict object details, even if shading is disabled. Important is only the availability of a point set that uniformly covers the object surface.

The number of points used for rendering the different models as well as the rendering times are summarised in Table 1. The right-most column shows whether the input points were extracted from a segmented volume (S) or a triangulation (T).

Table 1: Number of points, rendering time, and source of the different models (S: from a segmented volume, T: from a triangulation).

Model	Number of points	Rendering Time (fps)	Source
Cerebral Ventricular System	43816	4.97	S
Tooth	41364	5.48	S
Spine	11706	16.94	T
Brain	36758	5.90	T
Colon	55095	3.88	S

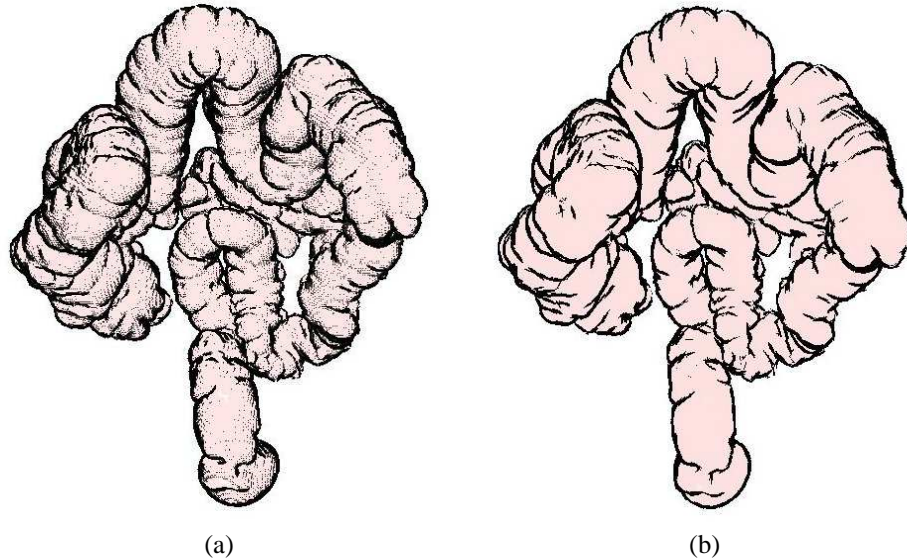


Figure 4: Rendering of the colon: with shading enabled (left), with no shading(right).

## 5 Conclusion

We have presented an algorithm for non-photorealistic rendering of segmented anatomical data. Our approach uses only a subset of surface points together with their normals to produce silhouette-based renderings. Silhouette detection is based on point normals investigation. We used properly oriented, filled disks as rendering elements. Variations of disk renderings were used also for hidden silhouette removal as well as for gradual-tone shading. Halftoning was adopted for producing illumination and shading effects by controlling the radii of shading disks. The system runs with interactive performance for models of moderate sizes.

In addition to testing alternative shading methods, our future work will focus on improving the applicability of the method for polygonal representations that do not provide sufficiently distributed surface point set. In this context, adaptive point sampling methods are going to be investigated. The algorithm will also be extended to hybrid representations, composed of points sets from segmented volume data and polygonal data.

## Acknowledgment

This work has been supported by the German Academic Exchange Service (DAAD) and the DFG focus program 1124 on "Medical Navigation and Robotics".

## References

- [ABCO<sup>+</sup>01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleichman, D. Levin and C. Silva. *Point Set Surfaces*. In: Proc. of IEEE Visualization, pp. 21–28, 2001.
- [CLL<sup>+</sup>88] H. Cline, W. Lorensen, S. Ludke, C. Crawford and B. Teeter. *Two Algorithms for the Three-Dimensional Construction of Tomograms*. Medical Physics, 15(3)(3):320–327, 1988.
- [CMH<sup>+</sup>01] B. Csébfalvi, L. Mroz, H. Hauser, A. König and E. Gröller. *Fast Visualization of Object Contours by Non-Photorealistic Volume Rendering*. Computer Graphics Forum, 20(3):452–460, 2001.
- [DCLK03] F. Dong, G. Clapworthy, H. Lin and M. Krokos. *Nonphotorealistic Rendering of Medical Volume Data*. IEEE Computer Graphics and Applications, 23(4):44–52, 2003.
- [DS00] O. Deussen and T. Strothotte. *Computer-Generated Pen-and-Ink Illustration of Trees*. In: Proc. of ACM SIGGRAPH, pp. 91–100, 2000.
- [GC95] H. Gray and H. Carter. *Gray's Anatomy*. Barnes & Noble Books, New York, 15th. edition, 1995.
- [Hau01] A. Hausner. *Simulating Decorative Mosaics*. In: Proc. of ACM SIGGRAPH, pp. 573–578, 2001.
- [HBH03] M. Hadwiger, C. Berger and H. Hauser. *High-Quality Two-Level Volume Rendering of Segmented Data Sets on Consumer Graphics Hardware*. In: Proc. of IEEE Visualization, pp. 301–308, 2003.
- [Her98] A. Herzmann. *Painterly Rendering with Curved Brush Strokes of Multiple Sizes*. In: Proc. of ACM SIGGRAPH, pp. 453–460, 1998.
- [HH94] D. Hearn and M. Hearn. *Computer Graphics*, Chapter Illumination Models and Surface Rendering Methods, pp. 516–522. Prentice Hall, 1994.
- [HPP<sup>+</sup>00] K. Höhne, B. Pflessner, A. Pommert, K. Priesmeyer, M. Riemer, T. Schiemann, R. Schubert, U. Tiede, H. Frederking, S. Gehrman, S. Noster and U. Schumacher. *VOXEL-MAN 3D Navigator: Inner Organs. Regional, Systemic and Radiological Anatomy*. Springer-Verlag Electronic Media, Heidelberg, Germany, CD-ROM. edition, 2000.
- [HZ00] A. Herzmann and D. Zorin. *Illustrating Smooth Surfaces*. In: Proc. of ACM SIGGRAPH, pp. 433–438, 2000.
- [IFH<sup>+</sup>03] T. Isenberg, B. Freundenberg, N. Halper, T. Schlechtweg and T. Strothotte. *A Developer's Guide to Silhouette Algorithms for Polygonal Models*. IEEE Computer Graphics and Applications, 23(4):28–37, 2003.

- [ISNC03] L. Ibáñez, W. Schroeder, L. Ng and J. Cates. *The ITK Software Guide*. Kitware Inc., 2003.
- [KGC00] M. Kaplan, B. Gooch and E. Cohen. *Interactive Artistic Rendering*. Proc. of Symposium on Non-Photorealistic Animation and Rendering, pp. 67–74, 2000.
- [KWTM03] G. Kindlmann, R. Whitaker, T. Tasdizen and T. Möller. *Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications*. In: Proc. of IEEE Visualization, pp. 513–520, 2003.
- [LC87] W. Lorensen and H. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*. In: Proc. of ACM SIGGRAPH, pp. 163–169, 1987.
- [LME02] A. Lu, C. Morris and D. Ebert. *Non-Photorealistic Volume Rendering Using Stippling Techniques*. In: Proc. of IEEE Visualization, pp. 211–218, 2002.
- [LW85] M. Levoy and T. Whitted. *The Use of Points as a Display Primitive*. TR 85-022, University of North Carolina at Chapel Hill, 1985.
- [MPFS03] O. Meruvia-Pastor, B. Freudenberg and T. Strothotte. *Real-Time Animated Stippling*. IEEE Computer Graphics and Applications, 23(4):62–68, 2003.
- [NLM] *National Library of Medicine of the National Institutes of Health*. <http://erie.nlm.nih.gov/evc/data/>.
- [PKG03] M. Pauly, R. Keiser and M. Gross. *Multi-scale Feature Extraction on Point-Sampled Surfaces*. In: Proc. of Eurographics, pp. 281–289, 2003.
- [RC99] R. Raskar and M. Cohen. *Image Precision Silhouette Edges*. In: Proc. of ACM Symposium on Interactive 3D Graphics, pp. 135–140, 1999.
- [RE01] P. Rheingans and D. Ebert. *Volume Illustration: Nonphotorealistic Rendering of Volume Models*. IEEE Transactions on Visualization and Computer Graphics, 7(3):253–264, 2001.
- [SS02] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann, 2002.
- [XC04] H. Xu and B. Chen. *Stylized Rendering of 3D Scanned real world Environment*. In: Proc. of Symposium on Non-Photorealistic Animation and Rendering (NPAR), pp. 25–34, 2004.
- [XNYC04] H. Xu, M. X. Nguyen, X. Yuan and B. Chen. *Illustrative Silhouette Rendering for Point-Based Models*. In: Proc. of EG Symposium on Point-Based Graphics, pp. 13–18, 2004.
- [ZPvBG01] M. Zwicker, H. Pfister, J. van Baar and M. Gross. *Surface Splatting*. In: Proc. of ACM SIGGRAPH, pp. 371–378, 2001.