Eurographics Symposium on Rendering (2005)
Kavita Bala, Philip Dutré (Editors)

Stippling and Silhouettes Rendering in Geometry-Image Space

Xiaoru Yuan Minh X. Nguyen Nan Zhang Baoquan Chen

University of Minnesota at Twin Cities, MN, USA [†]

Abstract

We present a novel non-photorealistic rendering method that performs all operations in a geometry-image domain. We first apply global conformal parameterization to the input geometry model and generate corresponding geometry images. Strokes and silhouettes are then computed in the geometry-image domain. The geometry-image space provides combined benefits of the existing image space and object space approaches. It allows us to take advantage of the regularity of 2D images and yet still have full access to the object geometry information. A wide range of image processing tools can be leveraged to assist various operations involved in achieving non-photorealistic rendering with coherence.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

Keywords: non-photorealistic rendering, geometry image, stippling, silhouette, sampling

1. Introduction

Non-photorealistic rendering (NPR) has become an alternative to traditional photorealistic rendering in computer graphics due to its effectiveness in conveying geometric information and directing attention. Existing NPR operations are mostly performed in two domains: image space and object space. Image space methods are normally fast due to the image regularity. However, since the strokes are placed in image space, undesirable effects (e.g. "shower door" effect [Mei96]) may appear. For object space methods, geometry features are detected and depicted in 3D, hence they are coherent during animation. These operations require traversing the entire input geometry model, thus achieving interactive rendering can be challenging.

In this paper, we present a novel non-photorealistic rendering method that operates in geometry-image space. A geometry image is obtained by first parameterizing the geometry model and then resampling it based on the regular parameterization grids [GGH02, GY03, JWYG04, GWC*04]. Hence, a geometry image resembles a conventional image in terms of its storage regularity. In contrast to most algorithms, which work in either object space or image space, or a hybrid of both, our algorithm performs most computations in the *geometry-image space*. Here we focus on one type of rendering in which shading tones and surface textures are depicted using a significant numbers of primitives, mainly stipples.

Our method is effective in achieving various NPR effects including stippling and stroke hatching and silhouette rendering. Compared with the image-space and object-space methods, our 2D geometry-image domain based NPR rendering method has the following advantages:

1. The computations of stroke primitive distribution and silhouette detection are both computed in the same 2D geometry-image domain. Part of their computation can be shared.
2. Fast hierarchical sampling in the 2D geometry-image domain can be applied to achieve rendering coherence with high performance.
3. Efficiency can be achieved by taking advantage of the 2D

[†] Email:{xyuan, mnguyen, nanzhang, baoquan}@cs.umn.edu

geometry-image regularity to leverage the programmability of modern commodity graphics hardware.

4. 2D image processing techniques such as edge detection can be applied to assist various operations for non-photorealistic rendering.

The remainder of this paper is organized as follows. In Section 2, we briefly review related work. After giving an overview of our method in Section 3, we describe major operations involved: preprocessing (Section 4), surface tone depiction (Section 5), silhouette rendering (Section 6), and visibility computation (Section 7). Results are discussed in Section 8, and finally the paper is concluded in Section 9.

2. Related Work

To render a surface with stippling or hatching effects, which are depicted by a significant number of primitives, there are mainly two approaches. In the first approach, stroke textures, which are collections of strokes arranged in different patterns, are mapped to the surface of the input geometry. Prioritized stroke textures [WS94, SABS94], art maps [KLK*00], and Tonal Art Maps (TAMs) [PHWF01] have been developed to render stroke textures on geometry surfaces to obtain hatching effects. In the second approach, locations of individual strokes are computed. Meier [Mei96] associated painterly strokes with actual locations on the model to achieve frame-coherence. Cornish et al. [CRL01] developed a view-dependent particle system, where a hierarchical clustering algorithm is used to regulate the density and placement of the strokes. Similarly, Pastor and Strothotte [MS02, MFS03] considered every vertex of an input model as the location of a potential stipple and dynamically performed mesh simplification and subdivision according to the shading changes. Secord et al. [SHS02] used a probability density function (PDF) derived from the input image to select a subset of pre-computed uniformly distributed random points. Voronoi diagrams with relaxation have also been applied to generate evenly distributed stippling image [DHvOS00, Sec02].

To render silhouettes, a school of methods have been developed. Existing silhouette generation algorithms for polygon models can be divided into three categories based on where the algorithm detects and draws the silhouettes: object space, image space, and hybrid algorithms. In object space methods, the silhouette of a free-form object is typically defined as the set of points on the object's surface where the surface normal is perpendicular to the viewing ray [HZ00]. For triangle based surfaces, silhouettes can be detected in object space by finding the edges in the mesh sharing both front and back facing polygons. Such brute force approaches require traversing every edge of the whole geometry model. Some acceleration methods have been developed [MKT*97, GSG*99]. For example, Buchanan and Sousa introduced the data structure of *the edge buffers* [BS00] that assists in finding all silhouettes by searching only faces of which the number is less than edges. On the other hand, image space methods avoid explicitly searching the 3D silhouette edges, hence they are usually efficient. Discontinuities in the rendered image buffer(s) are exploited by applying image processing methods such as edge detection [ST90]. Northrup and Markosian [NM00] applied a hybrid image/object space method to apply stylization to strokes that follow the visual silhouettes of the input object. However, these methods depict only portions of silhouettes that contribute to the final rendered image. The non-visible silhouettes cannot be detected by image space methods. Readers can refer to an in-depth review of silhouette detection algorithms by Isenberg et al. [IFH*03] for more information.

Temporal coherence is important for smooth animation. Since the density of strokes reflects the surface tone, it changes when view or illumination condition change. To guarantee temporal coherence, strokes must be attached to fixed locations on the object and incrementally added and removed. Existing methods address this issue by first building a mesh hierarchy and then dynamically selecting nodes of different levels during animation [CRL01, MS02]. For silhouette rendering, most methods first parameterize silhouettes and then texture map them to obtain stylization. To guarantee temporal coherence, parameterization between frames must be consistent. This is achieved recently by Kalnins et al. [KDMF03] by propagating silhouette parameterization from one image frame to another. We adapt a similar approach in our method but perform the propagation in the geometry-image domain.

3. Overview

We illustrate our NPR rendering method in Figure 1. In the preprocessing stage, we first apply global conformal parameterization to the input geometry model and generate a 2D geometry image. Then, we compute auxiliary images derived from the initial geometry image. These auxiliary images describe properties such as parameterization distortion, principle direction, etc. The principle directions are used to guide stroke orientations. All these images are both view and illumination independent.

For each frame, we compute the illumination image in geometry-image space. We perform hierarchical importance sampling [ODJ04] based on the illumination intensity which has been modulated by the distortion image. The output sampling points in 2D geometry-image space are used for placing stipples or strokes in 3D.

To render the silhouettes, we first compute the dot product of normal (N) and view vector (V) for each pixel in the geometry image. The silhouette pixels are then detected, linked and parameterized. 2D arc-length parameterization [Bou98] is propagated from the previous frame to achieve frame coherence. The final rendering combines the stroke and silhouette rendering.

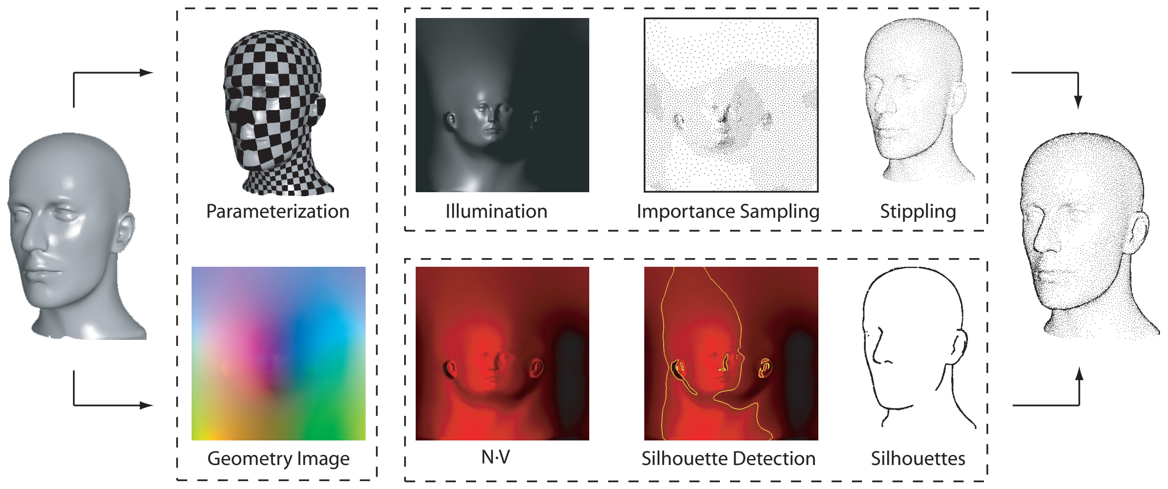

Figure 1: *The work flow of our non-photorealistic rendering method. The original input geometry model is first parameterized using global conformal parameterization and converted to geometry image. For rendering stipples, the illumination of the geometry image is computed. From that a set of samples are calculated and used to place stipples to reflect the tone of the surface. For silhouette rendering, an $N \cdot V$ image is computed, from which silhouettes are detected, linked and parameterized. The final rendering combines the stipples and silhouettes.*

4. Preprocessing

In this stage we parameterize the input geometry. Properties that are view and illumination independent are computed in geometry-image space.

4.1. Parameterization

Parameterization of a connected 3D mesh is a one-to-one mapping of the vertices of the input mesh onto a 2D space called parameterization domain. In our implementation, we choose a global conformal parameterization method [GY03, JWYG04] that can handle arbitrary geometry topologies, with or without boundaries. This parameterization guarantees visual smoothness of iso-parametric lines and minimizes the distortion. One issue to consider in this process is the resolution of the geometry image which resamples the original mesh. If the image resolution is not high enough, highly curved shapes in the geometry may be missed. For this purpose, with an assumption that the vertices in a geometry mesh are distributed uniformly, we usually set the geometry image dimension to $m \times m$ where $m = \lceil c\sqrt{n} \rceil$ with n being the number of vertices and c being a constant that the user can specify. In practice, we set c to $2-5$, which is usually sufficient for both quality and speed. For geometries with highly non-uniform scaling of parameterization, the factor c may be larger.

After the mesh parameterization is completed, we generate the corresponding geometry image [GGH02] of the input mesh. The image is created by drawing all triangles of the input mesh into a 2D domain, where the coordinate of each vertex is its 2D parameterization value, and the color is its 3D coordinate.

4.2. Computing Auxiliary Images

We pre-compute mainly two auxiliary images with the same resolution as the geometry image. Although distortion is minimized in the global conformal parameterization method, it still exists. We compute the distortion factor to compensate for this effect for the later computation and rendering stages. Since the global conformal parameterization merely affects scaling, only isotropic distortion may be introduced. The distortion factor is approximated as the ratio of the local surface area in the 3D space to the parameterized 2D area in the geometry image. The distortion image is used to modulate the illumination image to obtain a correct sampling density for surface tone representation (Section 5).

We generate another auxiliary image denoting the vector field for orienting strokes. Principle directions have been suggested as an effective means of defining the stroke orientation following the underlying surface shape [Int97, HZ00]. We compute the principle directions on the original mesh using the theory of normal cycles [CSM03] and then map them to geometry-image space. However, the computation of the principle directions is prone to error and not stable for discrete triangle meshes. Computed principle direction information can be unreliable, especially for isotropic regions (e.g. flat or spherical surfaces). We smooth the direction field by only keeping regions with highly reliable principle direction information and interpolating unreliable regions using the 2D thin plate spline method [Mei79]. The thin plate spline interpolation method is a two-dimensional analogy of the cubic spline in one dimension. It finds a "minimally bent" smooth surface that passes through all given points. Hertzmann and Zorin [HZ00] used an optimization procedure based on minimizing an energy function in 3D for such a smoothing process. As our smoothing algorithm is conducted entirely in 2D, it can be easily and efficiently implemented. Figures 2(a) and 2(c) illustrate the direction field on a geometry model before and after smoothing. Figures 2(b) and 2(d) are their corresponding color encodings in 2D geometry-image space, respectively.

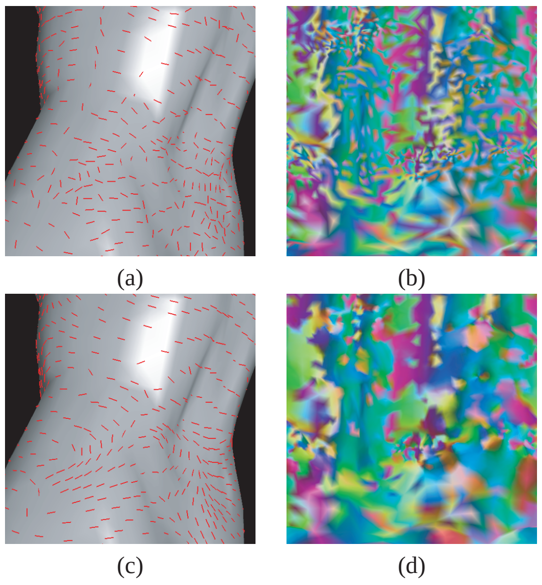

Figure 2: *(a) A direction field on a Venus model defined as major axis or principle direction. (c) Direction field after applying thin plate spline smoothing. (b) and (d), Corresponding color encoded direction fields of (a) and (c) in 2D geometry-image space, respectively.*

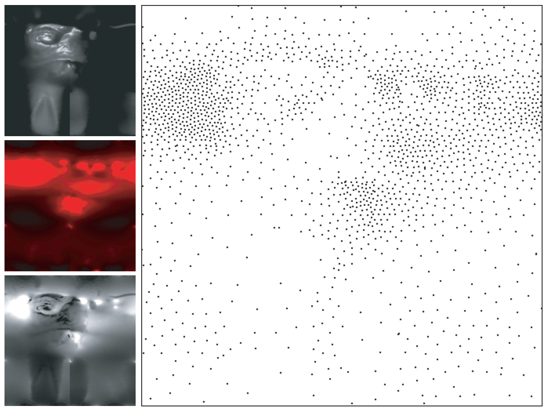

Figure 3: *Importance sampling for stroke distribution. Left column from top to bottom: illumination image, distortion image and importance image. The large image on the right shows sample locations.*

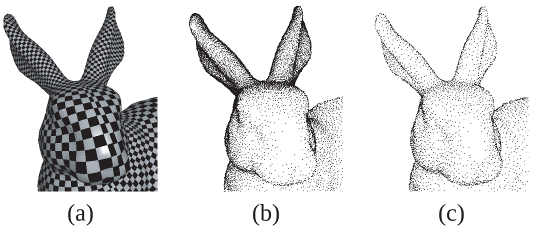

Figure 4: *(a) Bunny model with parameterization texture. Stippling without (b) and with (c) distortion compensation. The change in the ear region is apparent.*

5. Surface Tone Depiction

The density of strokes is changed to depict the surface tone. To compute the stroke distribution, we first compute the illumination of each pixel in the geometry-image domain using OpenGL diffuse shading. Because higher stroke density reflects darker surface tone, we invert the illumination image. To further discount the parameterizations distortion, the inverted illumination image is multiplied with the precomputed distortion image to obtain an *importance image*. Note that when using strokes instead of points for stippling, a constant factor is used to globally modulate the importance image, hence tuning the stippling density to better match the original shading tone. The constant is proportional to the area of the stroke used.

To generate strokes with appropriate distribution, Salisbury et al. [SWHS97] applied an updating procedure to match the rendered output image with the importance image by incrementally adding more strokes. This process is very expensive to perform. Because our algorithm operates in the geometry-image domain, we apply a fast hierarchical importance sampling method developed by Ostromoukhov et al. [ODJ04]. This method is based on Penrose tiling and the Fibonacci number system to distribute samples according to the local importance density. In this method, sample points generated for brighter levels are always a subset of darker levels, hence, coherence is guaranteed. The generated points are floating points with 2D space coordinates. Their corresponding 3D locations can be computed by interpolating geometry image. In the original work of Ostromoukhov et al., each generated point has a changing offset (although small) to Penrose tiling grids in order to optimize the local distribution with the blue noise property. This causes the stipples to slightly move during the animation. We choose to *disable* the offsetting during animation for guaranteed coherence, but *enable* it to achieve better random distribution for static rendering. It is worth pointing out that using geometric properties instead of illumination for the importance sampling can lead to other applications such as remeshing of the geometry; a relevant work is done by Alliez et al. [AMD02].

Figure 3 illustrates importance sampling using the fast hierarchical method. Figure 4 demonstrates the necessity of distortion compensation. In Figure 4(a), the ear region of the bunny has been expanded in the geometry-image space. The tone of the ears is too dark when point sampling is only based on the shading illumination (Figure 4(b)). By applying the distortion compensation, sampling density reflects correct tone (Figure 4(c)).

The sampling density is also affected by the projected screen size of the geometry model. To compensate for this effect, the computation of the sampling density takes into consideration the distance of the model from the viewer. The distance and its effect on the importance image are computed on every pixel in the geometry image during rendering. As shown in Figure 5, the sampling densities are consistent with the screen projection area change.

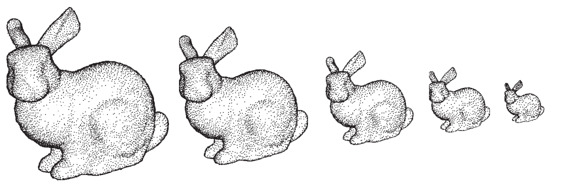

Figure 5: *The density of stipples is adjusted to achieve consistent tone when object size on the screen changes.*

6. Silhouette Rendering

In addition to stippling and hatching, line drawings effectively convey shapes using minimal strokes. We enhance our hatching/stippling drawings by rendering silhouettes. Our algorithm extracts silhouettes in the 2D geometry-image domain. We first compute the $N \cdot V$ image, then use the Marching Squares algorithm to extract the silhouette segments. A contour tracking algorithm is then applied to connect the silhouettes into polylines.

As illustrated in Figure 6, we detect pixels with zero-crossing in the rendered $N \cdot V$ image. These pixels are on silhouettes and are then linked and parameterized into an analytical form to facilitate stylization. Figure 6 shows the silhouette lines detected from the $N \cdot V$ image (red channel) in the geometry-image domain. To void negative values, $N \cdot V$ values are shifted to $(N \cdot V + 1)/2$. In the right column of Figure 6, the top image is the rendering of the detected silhouettes. The bottom image shows the model viewed from a different angle and mapped with the left image.

Silhouettes are closed curves in the geometry-image space for closed geometry objects (silhouettes may intersect each other in cusps regions) and evolve when the viewing parameters change. We utilize the method proposed by Bourdev et al. [Bou98] to achieve coherent silhouette parameterization. We conduct and propagate 2D arc-length parameterization in geometry-image space instead of image space. When the first frame is created, the silhouette pixels in geometry image are connected and parameterized. Each silhouette line is then segmented by placing sample points. Starting from the second frame, silhouette lines are parameterized using frame

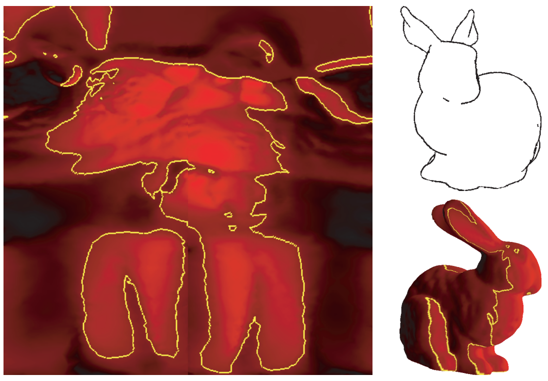

Figure 6: *Left: silhouettes detected in the geometry-image space; top-right: rendering of silhouettes; bottom-right: model viewed from a different direction; the model is mapped with the left image.*

coherence. We assume that the view changes are moderate and the new silhouette lines are usually not far from the silhouette lines of the previous frame. For each sample point on the current silhouette line, its parameterization value is derived by finding its nearest sample on the silhouette of the previous frame. Usually we use a tolerance of 1 pixel in searching the sampling neighborhood but this tolerance value can be varied. When a proper reference sample cannot be found for a sample point, a tag is recorded on this point. Its parameterization value is inferred from its neighboring sample points. Often times, the parameter values are not monotonically increasing or decreasing, such as the cases when the topology of the silhouette lines changes. In such cases, we split the silhouette lines in the questioned areas.

7. Visibility Computation

We compute the visibility mask of the model, including self occlusion and window clipping. With the visibility mask, the system is able to distinguish the visible and occluded portions of the silhouettes and render them into different styles. It also helps improving the efficiency of the importance sampling process by avoiding processing the occluded and clipped surface regions.

The visibility mask is computed in the programmable graphics hardware. First, the model is rendered and its depth image is stored into a texture. Then, the model is rendered in geometry-image space. For each pixel in the geometry-image, its depth value is computed and compared with the corresponding depth value in the texture obtained from the first pass rendering. Visibility caused by depth occlusion is thus determined. We also compute the visibility based on the view frustum culling.

Figure 7 shows the color coded visibility information. The green channel is used to encode visibility caused by depth occlusion and the blue channel is used to encode visibility

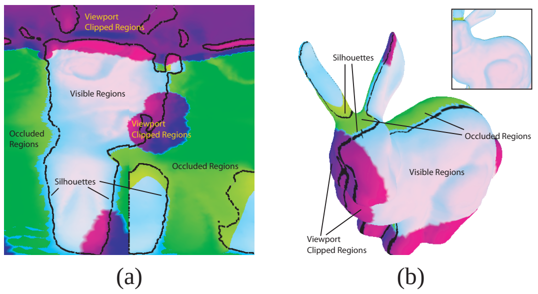

Figure 7: *(a) Visibility encoded in 2D geometry-image space (red: $N \cdot V$; green: visibility by depth occlusion; blue: visibility by view frustum culling). (b) Model mapped with (a) viewed from a different angle. The image in the square is the model rendered in 3D.*

by view frustum culling (1 indicates visible; 0 indicates invisible). The previously computed $N \cdot V$ values are stored in the red channel. The silhouettes coincide with the boundaries of visible and invisible regions. Note that in our implementation, we are able to render the $N \cdot V$ and shading images with visibility mask in one single pass. Utilizing the modern commodity programmable graphics hardware, the fragment shader can render the $N \cdot V$ and shading intensity into two color channel simultaneously.

The image in 7(a) is mapped to the model and viewed in a different angle in Figure 7(b). The image in the square is the model rendered in its original 3D viewing direction. With the visibility mask available, we disable the z-buffer and directly draw visible and invisible silhouettes with desired styles.

8. Results and Discussion

Figures 8-10 demonstrate some NPR effects generated by our method. In Figure 8, we demonstrate rendering a torus model in various non-photorealistic styles. Three different silhouette stroke textures are applied to parameterized silhouettes in Figures 8(a) through 8(c). In Figure 8(d), we also render the invisible silhouettes determined by the visibility mask described in Section 7 as dotted lines. In Figure 8(e), silhouettes are rendered together with surface stippling. Stipple points are directly rendered as OpenGL points at the locations generated by the importance sampling. By replacing stipples with directional strokes and applying a texture on each stroke, we produce various hatching effects, shown in Figures 8(f)-8(h). The orientation of the strokes follows the calculated direction field. Figure 9 depicts four models illustrated in stippling and hatching styles. In Figure 10, a Venus model is rendered using stippling and hatching of different stroke textures.

Finally, our method also promises rendering efficiency. The sizes of geometry images and rendering images are 512×512. All the results shown in the paper and the companion video are generated interactively on a PC with one Xeon 3.2GHz CPU and Geforce 6800 Ultra GPU. Most operations are performed on GPU except the silhouette extraction and the hierarchical sampling. The preprocessing of our method involves harmonic parameterization of the geometry models. Depending on model size, this step takes between 1 to 10 minutes. The runtime performance of our method is shown in Table 1. We achieve frame rate of around 10 fps for various models. T_1, time of rendering illuminance and $N \cdot V$, is proportional to the mesh size. T_2, silhouette detection and parameterizing, is independent of mesh size. The bottleneck of our method is performing the importance sampling on CPU (T_3). The frame rate is roughly inversely proportional to the number of generated stipples. The size of the original model has very little impact on the over all performance.

9. Concluding Remarks

In this paper, we have presented a non-photorealistic rendering method in geometry-image space. By transforming the 3D geometry into 2D parameterization domain, we gain both efficiency and flexibility on the computation of stroke primitive distribution and silhouettes. We ease the computation without resorting to complex 3D operations. Like other methods (e.g. [AMD02]) that have effectively taken advantage of geometry image, our work provides another evidence of the benefits of shifting 3D operations from conventional geometry to geometry image. This method opens the door to a rich set of well established 2D image processing methods, which can be utilized to achieve various non-photorealistic rendering effects. We plan to pursue further investigation towards this direction.

Acknowledgments: Support for this work includes a University of Minnesota Computer Science Department Start-up funds, NSF ACI-0238486 (CAREER), and NSF EIA-0324864 (ITR). We especially thank Xianfeng (David) Gu and Miao Jin for their help on mesh parameterization and valuable discussions. We also thank the anonymous reviewers for helpful suggestions; Nathan Gossett and Amit Shesh for proofreading and audio production of the video; and Hui Xu and Lijun Qu for discussions and suggestions.

References

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. In *SIGGRAPH* (2002), pp. 347–354.

[Bou98] BOURDEV L.: Rendering nonphotorealistic strokes with temporal and arc-length coherence. *Master's thesis, Brown University.* (1998).

[BS00] BUCHANAN J. W., SOUSA M. C.: The edge buffer: a data structure for easy silhouette rendering. In *NPAR '00* (2000), pp. 39–42.

[CRL01] CORNISH D., ROWAN A., LUEBKE D.: View-dependent particles for interactive non-photorealistic rendering. In *Proceeding of Graphics Interface* (2001), pp. 151–158.

Model	$Faces(K)$	$N_{stipple}$	T_1(sec)	T_2(sec)	T_3(sec)	T_4(sec)	T_5(sec)	$Frames/sec$
Torus	9600	2974	0.0087	0.010	0.0834	0.00065	0.0021	9.5
Face	13606	5383	0.0107	0.011	0.142	0.0017	0.0027	5.9
Bunny	26057	2442	0.0204	0.014	0.073	0.0034	0.0014	8.9
Fandisk	13250	1914	0.0100	0.011	0.061	0.0021	0.0012	11.7
Venus	5780	2441	0.0077	0.0098	0.0700	0.0020	0.0014	11.0

Table 1: *Performance of our rendering method for various models. $N_{stipple}$ is the number of stipples rendered. T_1 is the time for rendering the illuminance and computing the $N \cdot V$ in the geometry image space. T_2 is the time for detecting and parameterizing the silhouette. T_3 is the time for sampling point on the importance map. T_4 and T_5 are the time for rendering silhouettes and points, respectively.*

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *SCG* (2003), pp. 312–321.

[DHvOS00] DEUSSEN O., HILLER S., VAN OVERVELD C., STROTHOTTE T.: Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3 (2000).

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *SIGGRAPH* (2002), pp. 355–361.

[GSG*99] GOOCH B., SLOAN P.-P., GOOCH A., SHIRLEY P., RIESENFELD R.: Interactive technical illustration. pp. 31–38.

[GWC*04] GU X., WANG Y., CHAN T., THOMPSON P., YAU S.-T.: Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transaction on Medical Imaging 23*, 7 (2004), 949–958.

[GY03] GU X., YAU S.-T.: Global conformal surface parameterization. In *SGP* (2003), pp. 127–137.

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

[IFH*03] ISENBERG T., FREUDENBERG B., HALPER N., SCHLECHTWEG S., STROTHOTTE T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Comput. Graph. Appl. 23*, 4 (2003), 28–37.

[Int97] INTERRANTE V.: Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *SIGGRAPH* (1997), pp. 109–116.

[JWYG04] JIN M., WANG Y., YAU S.-T., GU X.: Optimal global conformal surface parameterization. In *IEEE Visualization* (2004), pp. 267–274.

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. *ACM Trans. Graph. 22*, 3 (2003), 856–861.

[KLK*00] KLEIN A. W., LI W., KAZHDAN M. M., CORREA W. T., FINKELSTEIN A., FUNKHOUSER T. A.: Non-photorealistic virtual environments. In *SIGGRAPH* (2000), pp. 527–534.

[Mei79] MEINGUET J.: Multivariate interpolation at arbitrary points made simple. *J. Appl. Math. Phys. 30* (1979), 292–304.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *SIGGRAPH* (1996), pp. 477–484.

[MFS03] MERUVIA PASTOR O., FREUDENBERG B., STROTHOTTE T.: Real-time animated stippling. *IEEE Comput. Graph. Appl. 23*, 4 (2003), 62–68.

[MKT*97] MARKOSIAN L., KOWALSKI M. A., TRYCHIN S. J., BOURDEV L. D., GOLDSTEIN D., HUGHES J. F.: Real-time nonphotorealistic rendering. In *SIGGRAPH* (1997), pp. 415–420.

[MS02] MERUVIA-PASTOR O., STROTHOTTE T.: Frame-coherent stippling. In *Eurographics, Short Presentations* (2002), pp. 145–152.

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic silhouettes: a hybrid approach. In *NPAR* (2000), pp. 31–37.

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph. 23*, 3 (2004), 488–495.

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

[SABS94] SALISBURY M. P., ANDERSON S. E., BARZEL R., SALESIN D. H.: Interactive pen-and-ink illustration. In *SIGGRAPH* (1994), pp. 101–108.

[Sec02] SECORD A.: Weighted voronoi stippling. In *NPAR* (2002), pp. 37–43.

[SHS02] SECORD A., HEIDRICH W., STREIT L.: Fast primitive distribution for illustration. In *EGRW* (2002), pp. 215–226.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-D shapes. pp. 197–206.

[SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH* (1997), pp. 401–406.

[WS94] WINKENBACH G., SALESIN D. H.: Computer-generated pen-an-ink illustration. pp. 91–100.

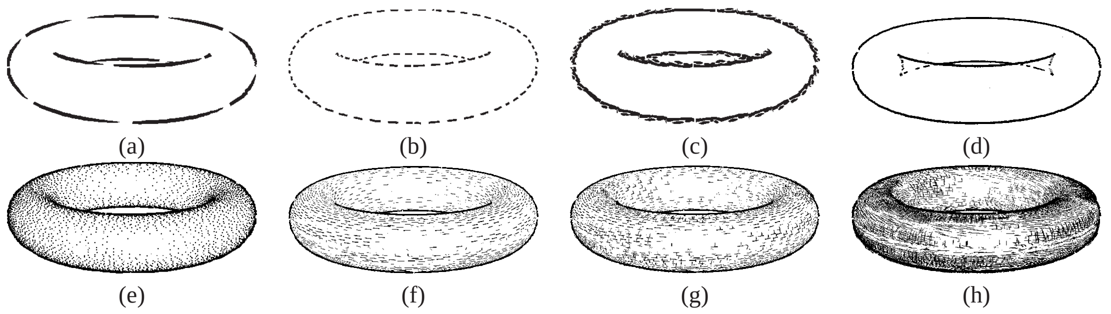

Figure 8: *A torus model rendered in various styles. (a) - (c) There different silhouette styles using different textures. (d) Invisible silhouettes are drawn in dotted lines. (e) Stippling; (f) - (h) Three hatching styles using different hatching strokes. Silhouettes are also rendered in (e) - (h).*

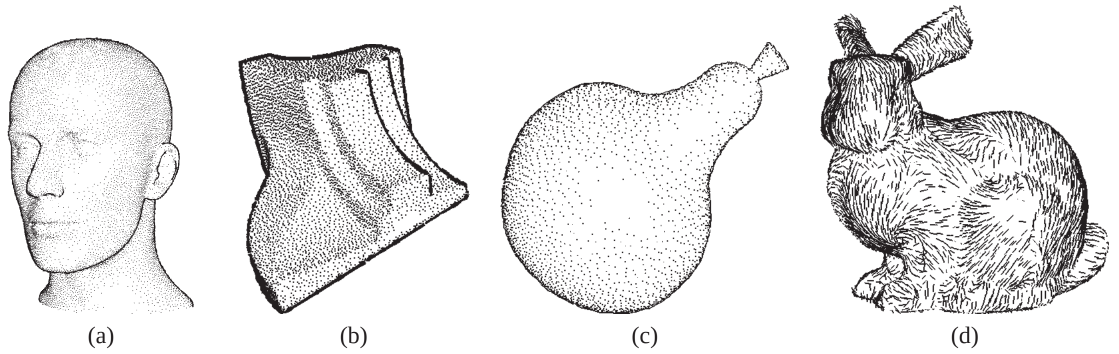

Figure 9: *Various rendering effects on different models: (a) male face; (b) fandisk; (c) pear; (d) bunny.*

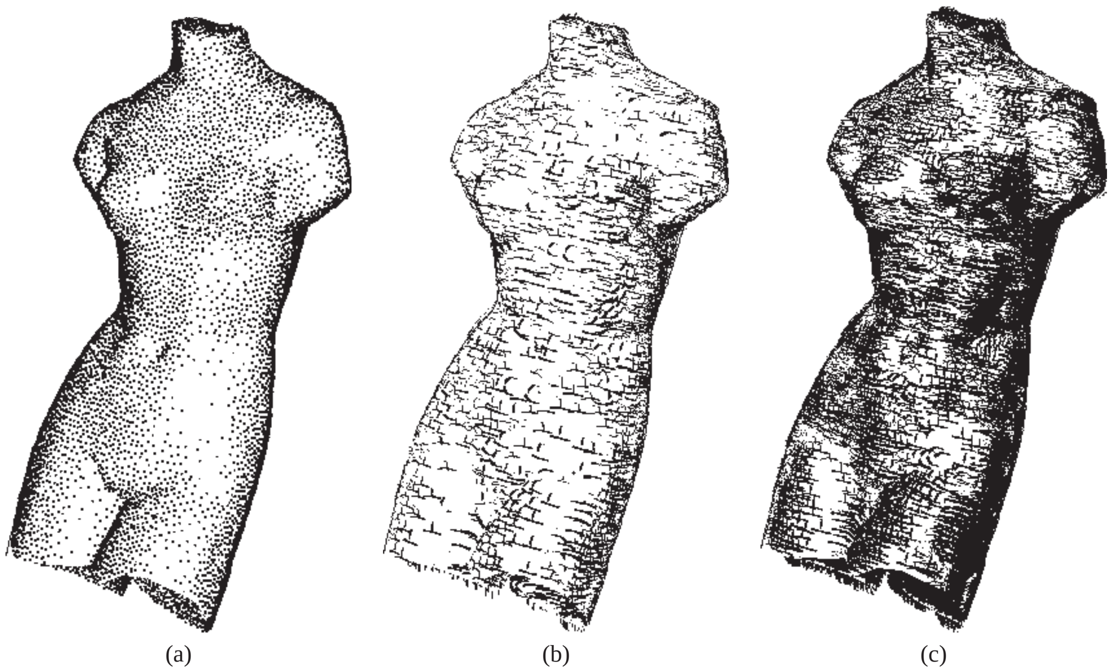

Figure 10: *A Venus model rendered with different styles.*

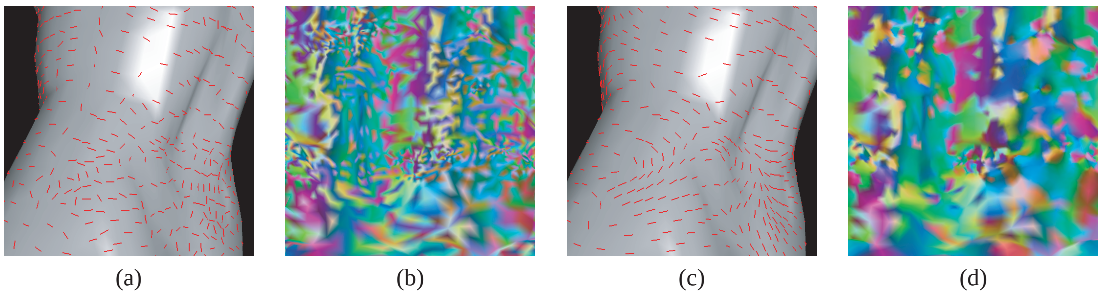

Figure 2: *(a) A direction field on a Venus model defined as major axis or principle direction. (c) Direction field after applying thin plate spline smoothing. (b) and (d), Corresponding color encoded direction fields of (a) and (c) in 2D geometry-image space, respectively.*

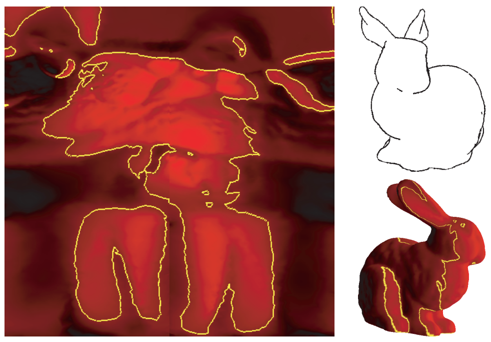

Figure 6: *Left: silhouettes detected in the geometry-image space; top-right: rendering of silhouettes; bottom-right: model viewed from a different direction; the model is mapped with the left image.*

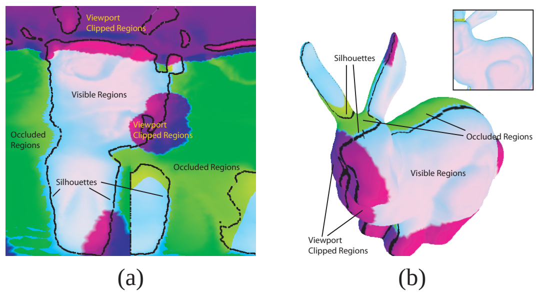

Figure 7: *(a) Visibility encoded in 2D geometry-image space (red: $N \cdot V$; green: visibility by depth occlusion; blue: visibility by view frustum culling). (b) Model mapped with (a) viewed from a different angle. The image in the square is the model rendered in 3D.*