Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, S. Rusinkiewicz, (Editors)

Interactive Silhouette Rendering for Point-Based Models

Hui Xu Minh X. Nguyen Xiaoru Yuan Baoquan Chen

University of Minnesota at Twin Cities, MN, USA †

Abstract

We present a new method for rendering silhouettes of point-based models. Due to the lack of connectivity information, most existing polygon-based silhouette generation algorithms cannot be applied to point-based models. Our method not only bypasses this connectivity requirement, but also accommodates point-based models with sparse non-uniform sampling and inaccurate/no normal information. Like conventional point-based rendering, we render a model in two passes. The points are rendered as enlarged *opaque disks in the first pass to obtain a visibility mask, while being rendered as regular size splats/disks in the second pass. In this way, edges are automatically depicted at depth discontinuities, usually at the silhouette boundaries. The silhouette color is the disk color used in the first pass rendering. The silhouette thickness can be controlled by changing the disk size difference between two passes.*

We demonstrate our method on different types of point-based models from various sources. The simplicity of our method allows it to be easily integrated with other rendering techniques to cater to many applications. Our method is capable of rendering large scenes of millions of points at interactive rates using modern graphics hardware.

Categories and Subject Descriptors (according to ACM CCS): I.3.0 [Computer Graphics]: General

1. Introduction

Depicting view-dependent geometric features such as silhouettes represents the simplest form of line art. It is also expressive and effective in conveying the essence of complex geometry. Since silhouettes are view-dependent, it is nontrivial to determine and render them at interactive rates.

Many techniques have been explored to generate silhouettes for polygon models and most of them take advantage of the models' connectivity information. However, few have been developed to render silhouettes for point-based models due to the lack of such information.

In this paper, we propose a new method to generate silhouettes for point-based models. Our method minimally modifies the conventional point-based rendering pipeline and does not require normal information. Therefore, not only can it be efficiently implemented, but also can be integrated with other existing point-based rendering techniques for achieving various additional effects.

The rest of the paper is organized as follows. We first review prior work in this area (Section 2). We then discuss our rendering pipeline in detail (Section 3). We present our results on different types of point-based models (Section 4), and conclude this work (Section 5).

2. Prior Work

Point-based modeling and rendering have gained significant momentum in recent years. While most of the research in this area has been focusing on improving rendering efficiency and/or visual quality [PZvBG00, RL00, RPZ02, BK03, DVS03], there have been efforts to extract and depict features for point-based models, parallel to the non-photorealistic rendering of polygon models. In fact, point models obtained through laser scanning may naturally be better rendered with certain abstraction due to the inherent complexity, incompleteness, and uncertainty embedded in such data. Pauly et al [PKG03] and Xu and Chen [XC04] have lately developed methods for extracting and depicting geometric features from point models. Both methods recognize and address the inherent ambiguity of the data.

Most existing silhouette generation algorithms are based

† Email:{hxu, mnguyen, xyuan, baoquan}@cs.umn.edu

on polygon models. They can be mainly divided into three categories based on where the algorithm detects and draws the silhouettes: only object-space, only image-space, and hybrid. For a detailed overview of each category, readers can refer to [IFH*03, Her99].

Most object-space algorithms rely on the connectivity information of a polygon model, hence can hardly be applied to point-based models. However, since the silhouette detection operates on intermediate image buffers (e.g., z-buffer, normal buffer) in image-space algorithms, they can usually be extended to point-based models. Hybrid algorithms operate in object space and render the silhouettes in image space with the assistance of the z-buffer. These methods rely less on the connectivity information. For example, silhouettes and ridges can be automatically rendered by view-dependently modifying polygonal objects [RC99], or by procedurally introducing extra polygons along each edge during the primitive shader stage [Ras01].

For point-models, Alexa et al [ABCO*01] use environment/normal mapping to generate silhouettes. This requires that points have accurate normals. Other methods have been proposed to render silhouettes for trees and smoke [DS00, SMC04]. Some existing polygon based silhouette generation methods can also be extended to point-based models. Nevertheless, our goal here is to develop a method that can be fully integrated with the existing point-based rendering pipeline without significant extra processing.

3. Silhouette Rendering Pipeline

Our method renders silhouettes through the conventional two-pass rendering pipeline, which is widely employed for generating high-quality images of point-based models. In general, points are rendered as opaque disks to obtain a correct *visibility mask* in the first pass. The points that pass the visibility test are then rendered using splatting techniques in the second pass [PZvBG00, RPZ02].

The basic idea of our silhouette rendering method is to slightly *enlarge* each opaque disk in the first pass. The frame buffer rendered in the first pass is *not* cleared. Splats in the second pass are rendered as regular sizes on top of the frame buffer. When the visibility mask is properly used, the extra edges of the enlarged opaque disks from the first pass will *not* be overdrawn by splats from the second pass when they are on silhouettes. Therefore, the silhouettes are automatically conveyed. The silhouette color is the disk color used in the first pass rendering.

It is worth noting that blending artifacts near contours during conventional point-based rendering have been recognized by researchers in this field [BK03]. Here we turn such artifacts into graceful silhouette lines whose thickness and color can be controlled. In our point-based model, only basic information such as coordinates and size (radius) is assumed for each point. We now summarize the two-pass rendering pipeline for completeness while emphasizing the changes we make in it.

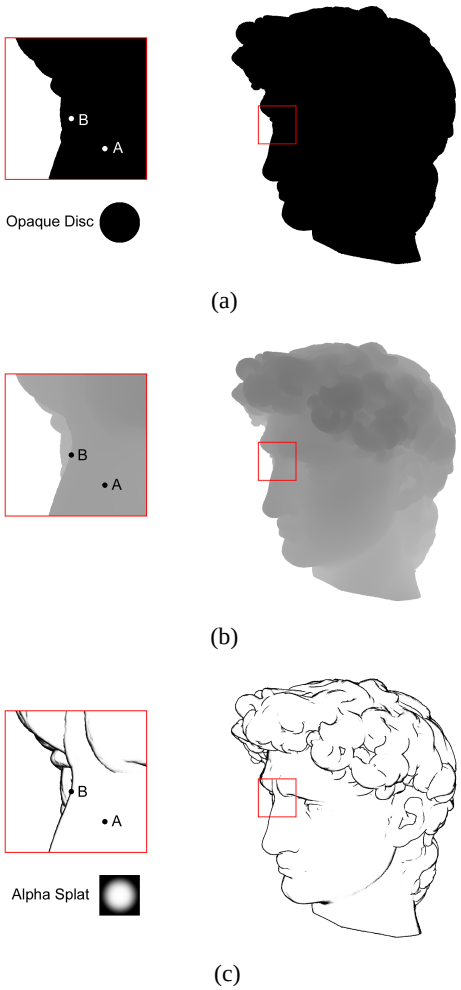


Figure 1: *Demonstration of our silhouette rendering method. In the first pass, the frame buffer (a) and the depth buffer (b) are generated by rendering the points as enlarged black opaque disks. In the second pass, the frame buffer (c) is overlayed by the points rendered as white alpha splats.*

3.1. First Pass

In the first pass, a depth buffer (visibility mask) is generated by projecting all the points and rendering them as opaque disks on screen. The projection size of each point is first calculated to ensure hole-free surfaces, and then *enlarged* for generating silhouettes. Each point's screen size is computed based on its size, its distance from the viewpoint, the field of view, and the screen resolution. Figures 1(a) and 1(b) show an example of the frame buffer and the corresponding visibility mask after the first pass rendering.

3.2. Second Pass

In the second pass, the depth buffer is initialized using the visibility mask and its updating is disabled. Each point is rendered as an alpha splat with the regularly calculated projection size (hence smaller than its corresponding opaque disk size). The size difference between the opaque disk rendered in the first pass and its corresponding splat here is used to control the thickness of the silhouettes. Only pixels passing the depth test are blended with the frame buffer. It has to be ensured that points on interior visible surfaces are not discarded by visibility testing. Similar to conventional approaches, we achieve this by adding a small offset ε to every depth value in the depth buffer. This offset can be applied by moving each point along its viewing direction for a certain distance during the first pass [RPZ02]. This operation can be efficiently implemented in vertex shaders.

In our method, silhouettes are generated at places where the depth difference between two adjacent points is greater than the offset ε. Let us look at two different points *A* and *B* in Figure 1. As shown in Figure 1(b), point *A* is at a place with continuous depth values, while point *B* is at a place with discontinuous depth values. During the second pass, the extra edge of a opaque disk at *A* is overdrawn by splats. However, the visibility test prevents point *B* from being overdrawn by nearby splats, hence forming a silhouette point. Figure 1(c) shows the final image rendered after the second pass. Note that the silhouettes are automatically conveyed.

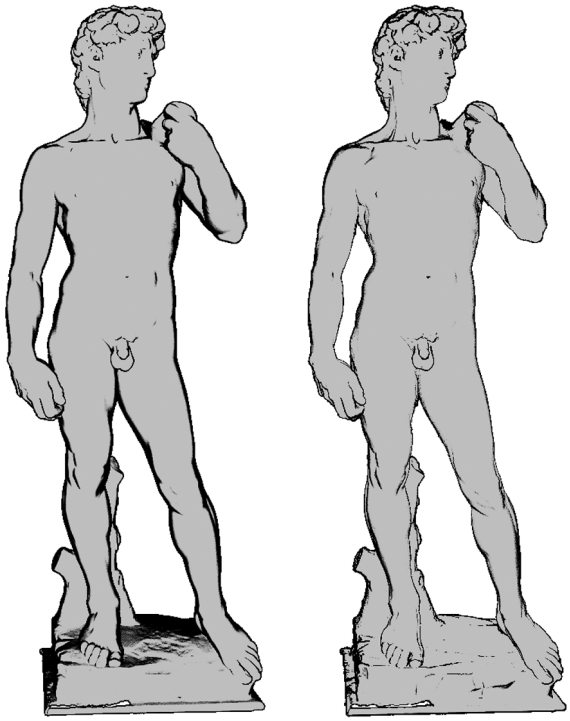
3.3. Additional Notes

It should be noted that it is not necessary to render points using splats in the second pass. In situations where objects have uniformly colored interior surfaces, points can also be rendered as opaque disks with a selected color. The rendering efficiency can then be improved since no alpha blending is needed. However, rendering points as splats in the second pass ensures soft silhouette lines because of the alpha blending.

Although we use a single color (white) for all the splats in Figure 1(c), different colors may be assigned to different splats to illustrate certain color features. Figure 3(b) shows an example of using color splats to simulate a cartoon style.

The simplicity of our method allows it to not only be efficiently implemented but also easily integrated with other existing point-based rendering techniques to achieve various effects.

It is also noteworthy that we require only the coordinate and size information for each point. Therefore, our method can be applied to most point-based models, including point clouds obtained through scanning where additional geometric information such as normals is unavailable. Nevertheless, additional geometric information can be utilized to obtain more effects if they do exist. For example, shading can be applied on such models as shown in Figure 4.



(a) (b)

Figure 2: *The Michelangelo's David model rendered using thick silhouettes (a) without and (b) with considering the disk orientation.*

When a point is rendered as a point sprite, all pixels within its coverage have a constant depth value. For surfaces that are oblique to the viewing direction, interior neighboring points may have a depth difference greater than the offset ε resulting in artificial silhouette regions. A solution alleviating this artifact is to render points using oriented disks if point normal is available. Individual pixel's depth within each disk is computed separately to ensure a smooth depth transition. Methods similar to [BK03] can be employed.

4. Results

We have implemented our method on the nVidia GeForce FX 5800 Ultra graphics card with 128MB video memory. Our PC has a 2.4GHz Xeon processor, 1GB of main memory and runs Windows XP.

We present our results for three different types of point data sets: conventional point-based models, outdoor scanned data, and point-based isosurfaces. We use the term conventional point-based model in this paper to differentiate the point-based model obtained from outdoor scanning and the point-based isosurface data of volumes. The accompanying video shows these examples in motion.

4.1. Conventional Point-Based Models

The model used in our paper is the Michelangelo's David model. The original polygon model was obtained from Stanford's Digital Michelangelo Project website. To demonstrate our method, we construct a point-based model by using only the vertices and their normal vectors. The connectivity information in the original model is just used for estimating the size of the point, and is discarded afterwards.

Figure 2(a) shows an image of this model rendered by our method. There are 4,129,614 points in this model. The average rendering speed is about 6-8 frames per second. In Figure 2(a), we use opaque disks four pixels larger than their calculated projection size in the first pass. As expected, the silhouettes rendered in Figure 2(a) appear thicker than those in Figure 1 where the opaque disks are only two pixels larger.

As discussed earlier, using point sprites may generate unwanted silhouette pixels at places where surface depth change is rapid. As shown in Figure 2(a), this happens on the David's pedestal, which is colored mostly black. With the availability of normal for each point, this artifact is lessened by rendering points as oriented disks with more accurate depth approximation [BK03]. The improvement can be clearly seen in Figure 2(b).

4.2. Outdoor Scanned Data Sets

Recently, researchers have shown increasing interest in capturing and processing real-world 3D objects and scenes. As laser scanning technology improves, outdoor scanning will emerge as an efficient way to acquire real world environments.

We apply our silhouette rendering technique on outdoor scanned point data. The scanned data is stored in the form of a 2D range image. Each pixel represents a 3D point of a certain size, which can be estimated based on the range value and the pixel azimuth/altitude angle. Since coordinates and size are the only critical geometry information required in our method, the scanned data can be easily processed by our system.

Figure 3(a) shows an image generated from an outdoor scan. The points are rendered as unorganized point cloud where only the coordinates and the size of a point are used. This style is achieved by rendering the points as white splats in the second pass. This image demonstrates the effectiveness of our method, in that although points are rendered individually the combined results convey long continuous silhouettes naturally.

Artifacts caused by the reason discussed in Section 3.3 can be noticed on the ground. However, these artifacts can not be effectively eliminated by rendering points using oriented disks with estimated normals as suggested in Section 4.1. The reason is that the point set scanned from the outdoor environment is noisy in those areas, therefore the estimated normals are not reliable.

Next, we show the ability of our method to generate different rendering styles as discussed in Section 3.3. Figure 3(b) shows the same outdoor environment data in cartoon style achieved by rendering both the silhouettes and large regions of constant color. This style shares some features with the one described in [DS02]. We utilize the fact that the scanned points are stored in the form of a 2D image. Therefore, we are able to partition the points into contiguous regions with similar color or intensity by an image segmentation process. In our system, we apply the image segmentation via the region growing method using the Mumford-Shah energy function [MS85]. The segmentation is done in a preprocessing stage. At run time, we use the average color in a segmented region to render all the points in that region during the second pass. In this way, large regions of constant color are formed in the final image.

4.3. Point-Based Isosurfaces of Volumes

Finally, we demonstrate the application of our silhouette rendering method for visualizing isosurfaces. Visualizing an isosurface that is defined by a particular value (a.k.a. isovalue) in a volume is one of the fundamental techniques for effectively analyzing and comprehending scalar volumetric data.

Instead of resorting to conventional polygon mesh based surface extraction methods [LC87], we extract a point-based surface model [YC04]. In this context, the point-based representation can be more efficiently constructed and rendered than polygon meshes [CHJ03]. We extract the points directly from the volume. An isopoint and its attributes (i.e., normal) are linearly interpolated from two adjacent voxels. An oriented ellipse (splat) is then defined at the position of that point. Splats lie on the isosurfaces and overlap with each other forming hole-free surfaces. After the point isosurface model of the volume is constructed, our silhouette rendering method can then be easily integrated with a typical isosurface rendering pipeline without interfering with other existing operations like shading.

Figure 4(a) shows the point based isosurface of a C_{60} molecular model rendered by our method. Figure 4(b) and 4(c) show the same CT scanned engine data with two different isovalues. Shading is enabled in rendering all the images. From these examples, the shapes of the isosurface data are shown to be effectively conveyed by our method.

5. Conclusion

We have presented an automatic method for rendering silhouettes of general point-based models. Our method is easy to implement, and can be fully integrated with many existing rendering techniques to cater to various applications. We

have applied our method on different types of point-based models including conventional point-based models, outdoor scanned data, and point-based isosurfaces of volumes. Our method is capable of rendering large scenes of millions of points at interactive rates by leveraging modern graphics hardware.

6. Acknowledgements

We thank Amit Shesh, Nathan Gossett and the anonymous reviewers for their constructive comments. We acknowledge Stanford Computer Graphics Laboratory for the David model (the Michelangelo project from Stanford University [LPC*00]). The Bucky ball (C_{60})data set was created by Dr. Oliver Kreylos at the University of California, Davis. The CT scanned Engine data is from General Electric.

This work was supported by NSF CAREER ACI-0238486, NSF EIA-0324864 (ITR), and a University of Minnesota Digital Technology Center seed grant.

References

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *Proceedings of IEEE Visualization* (2001), pp. 21–28.

[BK03] BOTSCH M., KOBBELT L.: High-quality point-based rendering on modern gpus. In *Proceedings of the Conference on Computer Graphics and Applications (PacificGraphics'03)* (2003).

[CHJ03] CO C. S., HAMANN B., JOY K. I.: Iso-splatting: A point-based alternative to isosurface visualization. *Proc. of Pacific Graphics 2003* (2003), 325–334.

[DS00] DEUSSEN O., STROTHOTTE T.: Computer-generated pen-and-ink illustration of trees. *Proc. of SIGGRAPH* (2000), 13–18.

[DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. In *ACM SIGGRAPH* (2002), pp. 769–776.

[DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential point trees. *ACM Trans. Graph. 22*, 3 (2003), 657–662.

[Her99] HERTZMANN A.: Introduction to 3d non-photorealistic rendering: Silhouettes and outlines. *Non-Photorealistic Rendering (SIGGRAPH 99 Course Notes)* (1999).

[IFH*03] ISENBERG T., FREUDENBERG B., HALPER N., SCHLECHTWEG S., STROTHOTTE T., VON GUERICKE O.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphcis & Application* (July/August 2003), 28–37.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3-d surface construction algorithm. *Computer Graphics (SIGGRAPH '87) 21* (1987), 163–169.

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital michelangelo project: 3D scanning of large statues. In *Siggraph 2000, Computer Graphics Proceedings* (2000), pp. 131–144.

[MS85] MUMFORD D., SHAH J.: Boundary detection by minimizing functionals, I. In *Proc. IEEE Conf. on CVPR* (1985), pp. 22–26.

[PKG03] PAULY M., KEISER R., GROSS M.: Multi-scale feature extraction on point-sampled surfaces. *Eurographics 22*, 3 (2003).

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface elements as rendering primitives. In *SIGGRAPH '00 Proc.* (2000), pp. 335–342.

[Ras01] RASKAR R.: Hardware support for non-photorealistic rendering. *2001 SIGGRAPH / Eurographics Workshop on Graphics Hardware* (August 2001), 41–46.

[RC99] RASKAR R., COHEN M.: Image precision silhouette edges. *Proc. ACM Sym. on I3D* (1999), 135–140.

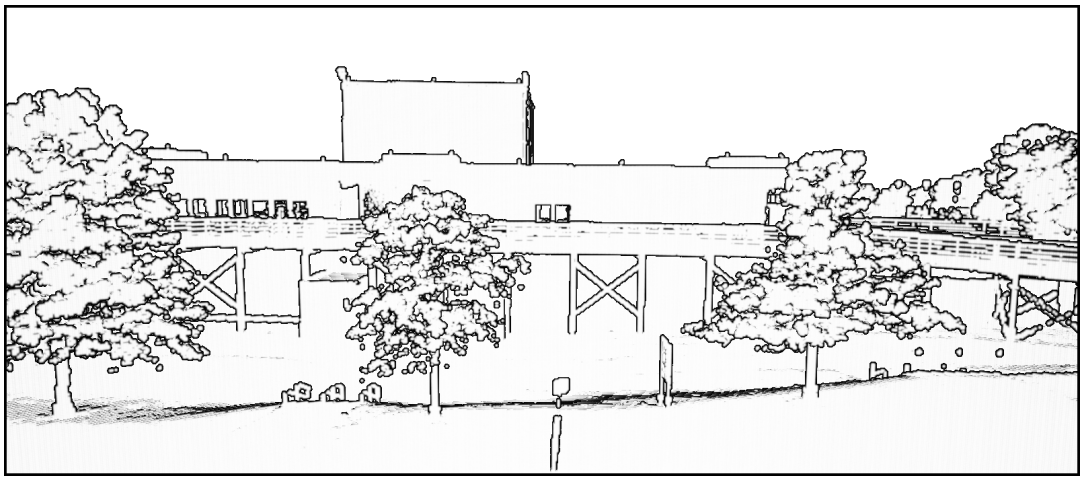
[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH '00 Proc.* (2000), pp. 343–352.

[RPZ02] REN L., PFISTER H., ZWICKER M.: Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Proc. of Eurographics* (2002).

[SMC04] SELLE A., MOHR A., CHENNEY S.: Cartoon rendering of smoke animations. In *Proc. of the 3rd Annual Sym. on Non-Photorealistic Animation and Rendering* (2004). to appear.

[XC04] XU H., CHEN B.: Stylized rendering of 3D scanned real world environments. In *Proceedings of the 3rd Annual Symposium on Non-Photorealistic Animation and Rendering* (2004). to appear.

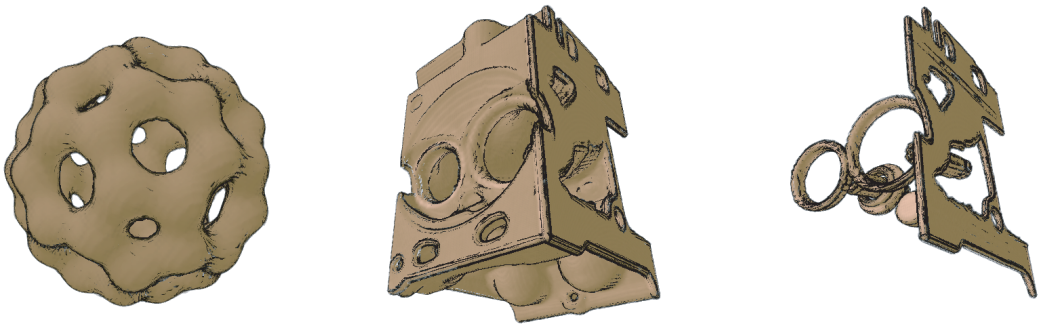
[YC04] YUAN X., CHEN B.: Illustrating surfaces in volume. *Proc. of Joint Eurographics - IEEE TCVG Symposium on Visualization* (2004).



(a)

(b)

Figure 3: *A scanned outdoor environment rendered by our method in (a) an outline style and (b) a cartoon style.*



(a) (b) (c)

Figure 4: *Demonstration of point-based isosurfaces rendered by our method with shading enabled. (a) The isosurface of a C_{60} molecular model (64^3); (b) The isosurface of a CT scaned engine; (c) The same data in (b) but with a different isosurface value.*