Eurographics Symposium on Point-Based Graphics (2005)
M. Pauly, M. Zwicker, (Editors)

Voronoi Rasterization of Sparse Point Sets

Jerry O. Talton Nathan A. Carr John C. Hart

University of Illinois at Urbana-Champaign

Abstract

Point-based representations are becoming increasingly common in computer graphics, especially for visualizing data sets where the number of points is large relative to the number of pixels involved in their display. When dealing with sparse point sets, however, many traditional rendering algorithms for point data perform poorly, either by generating blurry or non-occluding surface representations or by requiring extensive pre-processing to yield good results.

In this paper we present a novel method for point-based surface visualization that we call Voronoi rasterization. Voronoi rasterization uses modern programmable graphics hardware to generate occluding surface representations from sparse, oriented point sets without preprocessing. In particular, Voronoi rasterization clips away overlapping flaps between neighboring splats and generates an approximation of the Voronoi diagram of the points under the surface's geodesic distance. To approximate smooth shading and texturing on top of this clipped surface, our method uses existing techniques to construct a smoothly blended screen-space attribute field that implicitly accounts for neighborhood relations between points.

1. Introduction

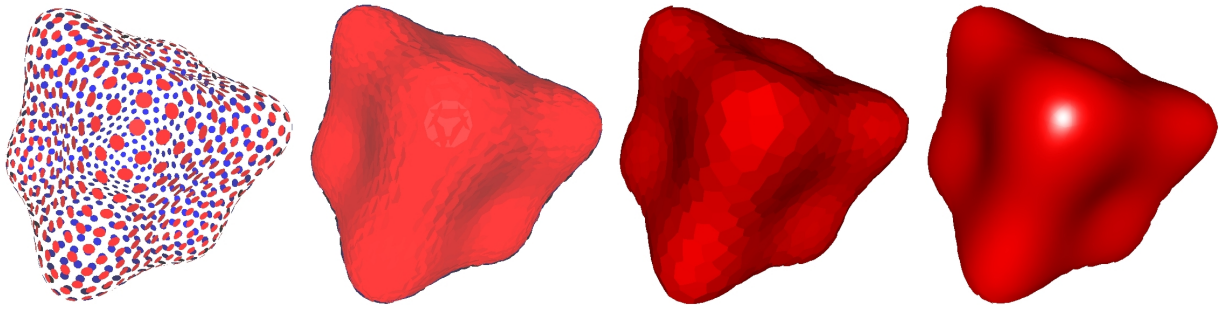
Since the point first drew consideration as a rendering primitive several decades ago [LW85], it has proven to be a useful and practical tool in a wide variety of visualization problems. Point-based techniques have been successfully applied in volume rendering [ZPvBG01a], ray tracing [ABCO*03], and dense mesh visualization [RL01]. Point-based schemes are especially suited for visualization problems where the number of rendering primitives is much greater than the number of pixels involved in their final display.

In contrast, sparse collections of points, where the desired number of activated pixels is large relative to the number of elements in the point set, challenge these traditional display methods. Techniques have been developed to interpolate a sparse collection of points into a smooth surface [Lev03, MYC*01], but methods for accurately rendering such surfaces typically require the re-introduction of dense point-sets and are far from real-time. Splatting approaches suitable for sparse sample display exist, but the resulting image can appear blurry or contain other visual artifacts.

Witkin and Heckbert [WH94] revolutionized the interactive display of implicit surfaces by constraining a sparse collection of mutually repelling particles to the surface. These particles may then be easily rendered as oriented disks, yielding a polka dot pattern that adds a texture perspective cue to the perception of the implicit surface. The main drawback of this approach is that it sacrifices occlusion, though back-facing particles can be culled. Occlusion can be restored by increasing the disk radii until neighboring disks overlap, but this leads to unsightly and distracting flaps with traditional rendering schemes.

Inspired by this approach, and given the programmability of the modern graphics processor, we have developed a multi-pass algorithm that removes such flaps from these overlapping oriented disks. What remains is an approximation of the Voronoi diagram of the points under the surface's geodesic distance. We call this method *Voronoi rasterization*, since we are replacing the rasterization of the (ideally Delaunay) triangular faces connecting surface samples with an approximated Voronoi face surrounding each surface sample. Furthermore, absolutely no inter-sample topology is needed: neighbor-relations between points are detected implicitly in image space by the existence of mutually overlapping flaps.

One advantage of meshed rasterization is the barycentric interpolation of vertex attributes (e.g. color data, normal vectors, texture coordinates) across faces. Point-based methods lack explicit topological information and thus pose



(a) (b) (c) (d)

Figure 1: *An implicit surface displayed with floater particles does not occlude (a). Increasing the floater radii creates an occluding surface with unsightly flaps (b). Voronoi rasterization trims these flaps yielding a faceted, occluding surface (c). Attributes such as the surface normal can be smoothly blended between point samples (d).*

a challenging problem in attribute interpolation. Our solution, similar to others' [BSK04], is inspired by blending via the sum of basis functions. First, we scale the size of each disk so they not only meet their neighbors but extend to cover the centers of neighboring disks. We then center a quasi-Gaussian kernel over each Voronoi cell, and the sums of these kernels, computed in an intermediate pass before the disks are clipped to each other, blend the attributes stored at each surface sample point. Unlike in meshes, this blending is not a linear interpolation, but it suffices for the visual display of the Gouraud and Phong smooth shading methods, as well as for texturing.

2. Related Work

The idea of splatting point samples by displaying overlapping oriented ellipses was driven by the generation of object representations from laser scanners, which typically produce a dense collection of surface samples [RL01, ZPvBG01b]. Subsequent work focused heavily on the development of robust tools for the editing [ZPKG02] and modeling [PKKG03] of such representations.

Similar to many other point splatting hardware accelerations [RPZ02, GP03, BK03, ZRB*04], our three-pass algorithm uses an initial visibility pass to determine occlusion, a second to interpolate attributes, and a third to perform final display. The difference with our approach is the result, which mutually clips the ellipses to each other to yield surface Voronoi cells with smoothly blended attributes and a near-continuous depth field. We display our point samples using quadrilateral polygons [RPZ02] which are trimmed to pixel-accurate ellipses using a fragment program [GP03, BK03, ZRB*04].

The idea of clipping sparse splats to reproduce sharp features arose amidst a derivation of perspective correct splatting [ZRB*04]. This work displayed truncated disks by first extracting sharp feature curves from the point data, and then performing clipping in the fragment program that converted the OpenGL point primitive into an ellipse, presumably by simply evaluating the clipping plane equation on the fragment location. In contrast, our Voronoi rasterization method clips projected disks to each other. An explicit feature detection step is not necessary as discontinuities in the orientation of the disks create feature curves as a result of this mutual clipping.

Botsch *et al.* proposed an efficient technique that makes use of programmable graphics hardware to blend normals across neighboring splats by associating a linearly varying normal field with each ellipse [BSK04]. Their technique, however, requires an $O(n^2)$ preprocessing algorithm to generate these fields, and would not be suitable for highly dynamic point sets. In addition, overlapping flaps between neighboring disks are alpha blended and not culled, making flat shaded representations impossible.

3. Algorithm

Voronoi rasterization is fundamentally a two pass algorithm that removes overlapping flaps from neighboring disks. Smooth attribute interpolation and silhouette clipping may each be approximated in an additional pass. A visual overview of the algorithm can be found in Fig. 2.

3.1. Assumptions.

We assume a distribution of points sampling an embedded 2-manifold, where each sample consists of a position, a surface normal, and a radius. We center a disk of the sample's radius at each sample position oriented perpendicular to the sample's normal. The radii are collectively large enough such that a line segment extending from a disk's center to any point on its boundary intersects a neighboring disk. We call such a distribution of disks an *occluding disk sampling* and the rendering of such a sampling appears in Fig. 1(b).

While such a disk sampling provides the desirable perceptual cue of occlusion, it also yields the distraction of extrane-

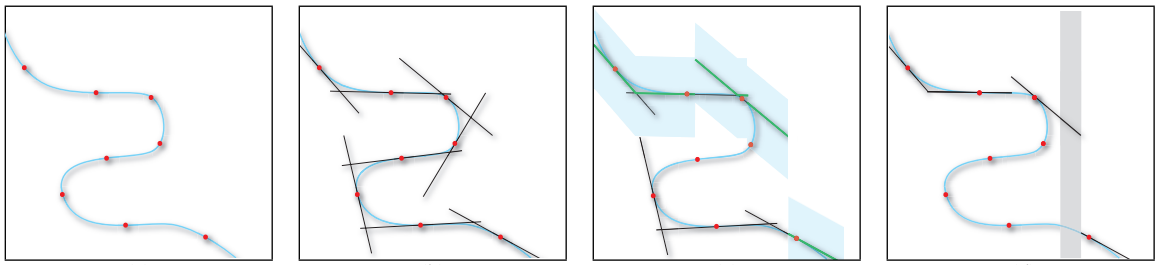


Figure 2: *A sparsely sampled surface (a). An occluding disk sampling splatted onto the surface (b). The depth buffer values generated by this splatting in green, surrounded by a blue region in which fragments are considered in our algorithm (c). The result of the algorithm, with regions where the algorithm makes an "incorrect" assignment on a silhouette shaded in grey (d).*

ous flaps in the outer regions of each disk, beyond the overlap. Classical splatting algorithms typically deal with these regions by performing a Gaussian weighted alpha blending, but this technique is wholly unsuited for flat-shaded representations and can result in blurry images on sparse point distributions. Our goal is to remove these flaps from the rendered geometry entirely, rather than simply blending their color values. We accomplish this via a two-pass algorithm that combines elements of surface line drawing with the idea of storing distance in the z-buffer [WND97, HCK*99].

3.2. Occluding Disk Samplings

Given a point set, the problem of picking disk radii appropriately so as to generate an occluding sampling is nontrivial. Our method, however, does not require this sampling to be tight: a liberal estimate for r produces error only along the silhouette boundaries of the rendered surface, although there is some slight performance degradation from overdraw. As a result, rather than resorting to expensive and complex spatial computations, we allow the user to adjust the disk radius interactively. When displaying Witkin-Heckbert floater particles, we find that setting $r = 1.3\sigma$ is typically sufficient, where σ is the global repulsion radius of the particle system [WH94].

3.3. Disk Geometry.

We model each disk geometrically as a square with sides of length twice the disk radius. A single texture coordinate represents a distance value, which is set to zero at the disk's center and $\sqrt{2}\times$radius at each vertex on the square's boundary. Perspective-correct interpolation thus sets this texture coordinate to the distance to the disk's center in each of the disk's fragments, and we discard fragments for which this coordinate is greater than the disk's radius to achieve per-pixel accuracy when rendering.

3.4. Voronoi Rasterization.

The first pass of our algorithm renders all front-facing disks into a z-buffer in the usual fashion and records the z-buffer in a texture. The second pass runs the rasterized fragments of each front-facing disk through a shader that:

(A) discards the fragment if its depth is farther than $\varepsilon = r/2$ away from the stored z-buffer depth value, and
(B) replaces the depth of the fragment with the texture coordinate corresponding to its distance from the disk center.

Fragments competing for a single pixel use the z-buffer (B) to ensure that only the fragment with smallest "depth" (distance to disk center) is plotted. The initial z-test (A) limits the competition for a pixel to the neighborhood of visible overlaps, so that fragments near the center of a front facing disk far behind the visible surface are not considered for rendering.

Thus, the first pass of Voronoi rasterization uses the z-buffer as a depth buffer, whereas the second pass uses it as a distance buffer similarly to previous Voronoi accelerations that leverage z-buffering hardware [WND97, HCK*99].

3.5. Attribute Blending

Although we can create flat-shaded occluding representations in two passes, the resulting images are not particularly visually pleasing. To allow for smooth shading and texturing, we approximate per-point vector attribute interpolation by inserting an additional rendering pass between passes one and two of the previous algorithm.

First, for this intermediate pass, we extend the radius of each disk to not only overlap the boundary of its neighboring disks, but to now overlap the center of each of its neighboring disks. We also enable blending and accumulate into a 16-bit floating point texture. We render only the front-facing disks and use the same epsilon depth test as before.

The intermediate pass utilizes a fragment shader that constructs a quasi-Gaussian kernel over each disk i of radius r_i

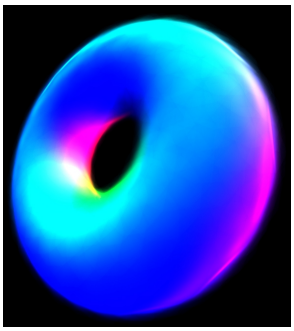


Figure 3: *The image-space interpolated normal attribute field for a torus.*

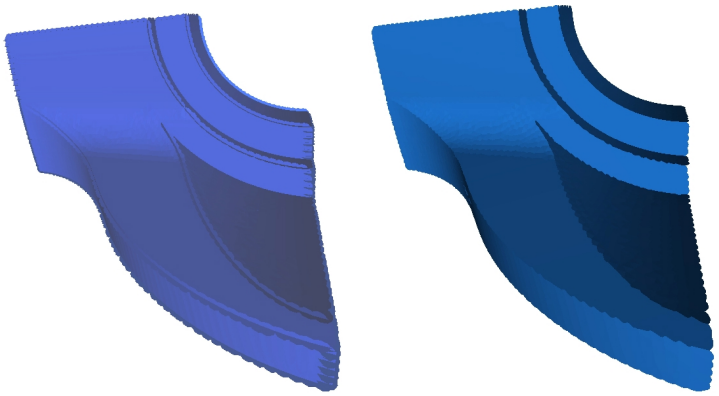


Figure 4: *The fandisk surface rendered with flat splatting (left) and Voronoi rasterization without an attribute field (right). Note the visually distracting overlap present in the left image, and how disks are automatically clipped against each other in our method.*

as:

$$w_i(d) = 3\frac{d^2}{r_i^2} - 2\frac{d^3}{r_i^3} + 1 \qquad (1)$$

where d is the value of the texture coordinate representing the distance to the center of disk i. The fragment shader then scales the disk's attribute value by the fragment's weighting function $w_i(d)$ and adds the result to the output texture's corresponding pixel.

This results in a smoothly blended floating-point screen-space attribute field over the projection of the surface shown in Fig. 3. Because these blending functions are not partitions of unity, this field is not strictly interpolating. It nevertheless suffices to produce a smooth Phong shaded appearance over sparse point sets.

3.6. Silhouette Clipping

In two dimensions, Voronoi cells extend until they meet a neighboring Voronoi cell except for the cells of points on the convex hull, which extend infinitely outward. The surface Voronoi cells of our front-facing point samples near the silhouette likewise would extend outward infinitely. Our algorithm stunts the growth of Voronoi cells by rasterizing disks whose radius is larger than half the diameter of the largest Voronoi cell, which results in the scalloped appearance of arc-shaped flaps extending around the silhouette (Fig. 4).

In some instances, we can clip these scallops with the silhouette of the shape we are displaying. When displaying implicit surfaces with Witkin-Heckbert floater particles, we can utilize a second particle system to adhere to the silhouettes whose job it is to clip against the floater particles [SH05]. For more general point data, we can employ another rendering pass to clip some silhouette particles. In this new pass, we render all back-facing disks into the z-buffer and store this buffer in a second texture. In subsequent passes fragments are tested against this texture, and any fragment with a z-value greater than the stored z-buffer value is discarded. This process, however, is not highly reliable, and can introduce rendering artifacts that are more visually distracting than the silhouette artifacts it seeks to prevent.

4. Results and Discussion

We implemented Voronoi rasterization in Wickbert, an open-source shape modeling library based on particles on surfaces [SH05]. As with other multi-pass methods, the performance of our algorithm is heavily dependent on the screen-space resolution of the rendered image. At 1024×768 on an Athlon 64 3500+ with a GeForce 6 series card, we can render on the order of 50,000 clipped and smoothly shaded disks at interactive rates (10 FPS). For most sparse visualizations, the number of points is much smaller and the algorithm runs in real time (30+ FPS). These results are not particularly competitive with existing mutli-pass point-based rendering algorithms [SPL04], but this is largely due to the developmental nature of the Wickbert library and to our desire to emphasize the simplicity of our implementation. It is worth observing that there is nothing inherent in the algorithm that would prevent it from being aggressively optimized in the manner of [BSK04]: a more conservative implementation would likely yield an order of magnitude performance improvement.

Voronoi rasterization presents two attractive features over existing techniques for sparse point-based visualizations. Firstly, it requires no preprocessing whatsoever. Point sets in which the number, location, or orientation of the points are changing rapidly present no special challenge to our method since we avoid any nearest-neighbor or topological queries. That the computational complexity of the algorithm is linear in the number of points is an especially attractive feature for implicit surface visualization, since the floater particles

used to display the surface in the Witkin-Heckbert system are highly dynamic.

Secondly, Voronoi rasterization is extremely easy to implement. Little complicated mathematics is needed (the bulk of which occurs in the epsilon test due to the non-linear nature of the z-buffer) and the fragment programs at the heart of the algorithm require only around twenty lines of CG code. This simplicity ensures that the technique can be integrated into existing point-based visualization systems with little time and effort.

The primary failing of our method occurs at surface silhouettes. Although clipping can be performed in certain cases, silhouette artifacts persist for the majority of surfaces. This problem, however, is not unique to our algorithm and becomes less perceptible as the size and density of the point set increases. Silhouette artifacts simply tend to be more visible in Voronoi rasterization than in traditional splatting methods because our technique is specifically intended for sparse sets of points.

5. Conclusion

In this paper we proposed the use of a two-pass GPU algorithm to generate occluding representations from sparse point sets. In particular, our algorithm leverages the z-buffer to generate splatted surfaces with no overlap between neighboring disks. The flat shaded representations generated by our method are superior to those produced by existing techniques when the sampling is very sparse or is heterogeneous with sparse regions and real-time display is desired. Our algorithm is also quite simple and easy to implement.

To create smooth visualizations, we add an intermediate pass in which a screen-space attribute field is composited over the projection of the point set surface. While the images produced by this field are of slightly lower quality than those generated by more complex techniques, our method requires no explicit nearest-neighbor computations, making it ideal for point sets with varying attributes or cardinality.

6. Acknowledgments

This research was sponsored in part by the NSF under the ITR SCI-0113968 and by NVIDIA.

References

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., T. SILVA C.: Computing and rendering point set surfaces. *IEEE TVCG 9*, 1 (2003), 3–15.

[BK03] BOTSCH M., KOBBELT L.: High-quality point-based rendering on modern GPUs. In *Proc. Pacific Graphics* (2003), pp. 335–442.

[BSK04] BOTSCH M., SPERNAT M., KOBBELT L.: Phong splatting. In *Proc. Point Based Graphics* (2004).

[GP03] GUENNEBAUD G., PAULIN M.: Efficient screen space appoach for hardware accelerated surfel rendering. In *Proc. Vision, Modeling and Visualization* (2003), pp. 1–10.

[HCK*99] HOFF K., CULVER T., KEYSER J., LIN M., MANOCHA D.: Fast computation of generalized voronoi diagrams using graphics hardware. In *Proc. SIGGRAPH* (1999).

[Lev03] LEVIN D.: Mesh-independent surface interpolation. In *Geometric Modelling for Scientific Visualization* (2003), Brunnett G., Hamann B., Muellller K.,, Linsen L., (Eds.), Springer-Verlag.

[LW85] LEVOY M., WHITTED T.: *The Use of Points as a Display Primitive*. Tech. Rep. 85-022, UNC, Chapel Hill, 1985.

[MYC*01] MORSE B. S., YOO T. S., CHEN D. T., RHEINGANS P., SUBRAMANIAN K. R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Proc. Shape Modeling Intl.* (2001), pp. 89–98.

[PKKG03] PAULY M., KEISER R., KOBBELT L., GROSS M.: Shape modeling with point-sampled geometry. *ACM TOG 22*, 3 (2003), 641–650.

[RL01] RUSINKIEWICZ S., LEVOY M.: Streaming qsplat: a viewer for networked visualization of large, dense models. In *Proc. I3D* (2001), pp. 63–68.

[RPZ02] REN L., PFISTER H., ZWICKER M.: Object space EWA surface splatting: a hardware accelerated approach to high quality point rendering. In *Proc. Eurographics* (2002), pp. 461–470.

[SH05] SU W., HART J.: A programmable particle system framework for shape modeling. In *Proc. Shape Modeling Intl.* (2005).

[SPL04] SAINZ M., PAJAROLA R., LARIO R.: Points reloaded: Point-based rendering revisited. In *Proc. Point Based Graphics* (2004), pp. 121–128.

[WH94] WITKIN A., HECKBERT P.: Using particles to sample and control implicit surfaces. In *Proc. SIGGRAPH* (1994), pp. 269–277.

[WND97] WOO M., NEIDER J., DAVIS T.: *OpenGL Programming Guide*. Addison Wesley, 1997.

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3d: an interactive system for point-based surface editing. In *Proc. SIGGRAPH* (2002), pp. 322–329.

[ZPvBG01a] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: EWA volume splatting. In *Proc. Visualization* (2001), pp. 12–17.

[ZPvBG01b] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *Proc. SIGGRAPH* (2001), pp. 371–378.

[ZRB*04] ZWICKER M., RASANEN J., BOTSCH M., DACHSBACHER C., PAULY M.: Perspective accurate splatting. In *Proc. Graphics Interface* (2004).