BSP Shapes

Carsten Stoll Hans-Peter Seidel
Max Planck Institut für Informatik
cstoll,hpseidel@mpi-sb.mpg.de

Marc Alexa
Faculty of EE&CS, TU Berlin
marc@cs.tu-berlin.de

Abstract

We discuss a shape representation based on a set of disconnected (planar) polygons. The polygons are computed by creating a BSP that contains approximately linear surface patches in each cell. This is achieved by employing two heuristics for finding appropriate split planes in each cell. Leaf nodes in the BSP tree represent either polygonal surface approximations or empty (clip) cells rather than split planes. We show that the resulting set of disconnected primitives typically leads to a better two-sided Hausdorff error for a given number of primitives than meshes. The BSP cells can be coded with few bits and, consequently, the tree is a compact shape representation. The special properties of BSPs are very useful in applications that need to perform spatial queries on the primitives, such as for occlusion and view frustum culling, and proximity or collision tests.

1 Introduction

Shapes are represented in computer graphics mostly by collections of primitives. Typically, primitives are connected, yielding at least a C^0 approximation of the surface. Lately, surface representations based on disconnected primitives are becoming more and more popular.

On one side of the spectrum of potential primitives, points are used because they naturally result from most acquisition systems (see, e.g., [13]) . By connecting a surface to the set of points [12, 1, 5] points can also be used for modeling shapes [21].

On the other side of the spectrum, polynomial patches are used as individual surface approximations [2], which could be later blended to form a continuous surface [16, 17]. Surfaces have also been described by mixtures of these ideas, e.g. associating higher order information to point primitives [18, 6].

Our main goal here is to provide an accurate surface approximation consisting of individual surface patches. As was noted by Wu & Kobbelt [20], this type of representation could be viewed as C^{-1} continuous across patch boundaries and, as usual, smaller degree of continuity corresponds to more degrees of freedom and, consequently, more accurate approximations for a given primitive count. In particular, we show that a set of linear surface patches (i.e. planar polygons) can approximate a given surface with smaller *symmetric Hausdorff error* than connected linear pieces. At the same time, we generate and represent these patches using binary space partitions. The linear patches are space partitions as well. This approach combines an accurate surface approximation with a spatial data structure resulting in an interesting new shape representation.

Most shape approximation approaches only optimize one sided approximation errors, i.e. the distance of each point on the surface being approximated to the closest point on the approximation. A cardinal example is the reconstruction of a surface from points, where mostly the distance of the input points to the reconstructed surface is optimized; yet, parts of the reconstructed surface might be arbitrarily far away from the point set.

More precisely, given a surface $\mathcal{S}$ to be approximated by a set of patches $\mathcal{P}$. Typically, the approximation error is computed as the one-sided distance from points on the surface to points on the patches

$$d(\mathcal{S}, \mathcal{P}) = \sup_{\mathbf{s} \in \mathcal{S}} \inf_{\mathbf{p} \in \mathcal{P}} \|\mathbf{s} - \mathbf{p}\| \tag{1}$$

as each patch is computed to minimize the distance of the surface to the patch. However, the Hausdorff error takes into account also any point on one of the patches in $\mathcal{P}$ and their distance to the surface $\mathcal{S}$:

$$h(\mathcal{S}, \mathcal{P}) = \max\left(d(\mathcal{S}, \mathcal{P}),\ d(\mathcal{P}, \mathcal{S})\right) \tag{2}$$

In practice, the error is computed by sampling both sets, computing the closest point in the other set for each sample, and taking the maximum distance over all samples (for meshes see [7]). While sampling introduces an additional error, this error is bounded by the sampling resolution, i.e. a given error can be easily bounded by asking that the sampling resolution plus measured distances satisfy the bound.

As we will show, minimizing only one-sided distances could lead to bad approximation results. In this context,

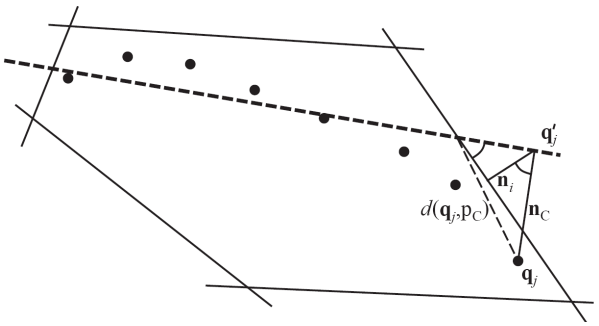

Figure 1. Calculating the error for sample points whose projection lies outside of the cell to conservativeley bound the distance.

it is interesting to mention a recent and very successful method for computing optimized piecewise linear approximations using variational techniques [9]. While this approach achieves very good quality approximations minimizing a one-sided error, according to the authors [3], also bounding the error in the other direction is an important yet open problem. Note that this is also the reason why it is hard to compare these different results: because it is unclear how to generate shape approximations with bounded symmetric Hausdorff error using variational techniques.

For construction of the linear pieces we follow along the lines of Ohtake et al. [16] and derive surface patches by subdividing space. However, rather than using a regular spatial subdivision, we divide cells by split planes, which are chosen to reduce either the distance of the shape to the patches or vice versa. The resulting binary space partition (BSP) consist of cells that contain approximately planar pieces of the surface, or no surface at all. An additional split plane in non-empty cells is used to represent the planar patch and the boundary of this patch results from clipping the plane against the cell. Empty cells are created to clip away empty space and, in that way, minimize the patch size to reduce the distance from patches to the surface. The construction of the BSP is described in detail in Section 2.

It is not new to use BSP trees to describe the faces of a surface. This concept has been used to facilitate set operations [14] on polyhedra or, more generally, do solid geometry on polyhedra [15]. BSP trees have also been used to approximate a distance field to the surface [19]. We discuss the similarities with these approaches in Section 2 as well.

Apparently, the shape approximation consists entirely of a BSP tree, which is why we call this representation *BSP shapes*. The only additional information needed is to distinguish between planar patches and clip planes among the leaf nodes in this tree. This means the shape can be stored by a tree traversal; for each plane we only need to store a normal and an offset. For moderate quantization of this information we achieve good bit rates for error-bounded shape representation. In an application, the tree can be queried while being traversed, e.g., for sending only the set of patches to the graphics card that are contained in the view frustum. Coding and traversals of the tree, and resulting storage sizes and polygon counts are discussed in Section 3.

2 Construction of the BSP shape

The main strategy for generating the BSP is to minimize both one-sided distances by choosing appropriate split planes. Let $\mathcal{S}$ be the shape to be approximated by a set of linear patches $\mathcal{P}$. Linear patches are defined as the intersection of a plane and a BSP cell. During the first phase of the process, mostly $d(\mathcal{S},\mathcal{P})$ is minimized by splitting cells so that a linear patch approximates the surface points in that cell well. In a second phase, the distance $d(\mathcal{P},\mathcal{S})$ of patches to surface points is minimized, by clipping away parts of the polygonal patches that are far away from any surface point.

Both splitting phases are controlled by bounds $\epsilon_{\mathcal{S}\to\mathcal{P}}$ and $\epsilon_{\mathcal{P}\to\mathcal{S}}$ on the maximum distances over all cells. In the following we first explain the basic computations per cell, then how to split cells in the first phase based on two tests and two heuristics, and finally how to clip cells.

2.1 Per cell computations

Let a cell C be defined as the intersection of half spaces $\mathbf{n}_i\mathbf{x} - d_i > 0$. The cell contains a part of the surface $s_C \subset \mathcal{S}$. This part is approximated by a planar patch $p_C \in \mathcal{P}$. The planar patch is defined as the intersection of a plane $\mathbf{n}_C\mathbf{x} - d_C = 0, \|n_C\| = 1$ with the cell.

We approximate the distance $d(s_C, p_C)$ by computing distances of samples $\{\mathbf{q}_j\} \in s_C$ to the polygon. Each sample is projected to the plane yielding $\mathbf{q}'_j = \mathbf{q}_j - \mathbf{n}_C(\mathbf{q}_i\mathbf{n}_C - d_C)$. This projection is checked against the half spaces, i.e. the sign of $\mathbf{n}_i\mathbf{q}'_j - d_i$ is evaluated. If any of the signs are negative, a maximum distance between the projection and the interior of the polygon is approximated as (see also Figure 1)

$$d(q'_j, p_C) = \max_i \frac{\max(0, -\mathbf{q}'_j\mathbf{n}_i + d_i)}{\sqrt{1 - (\mathbf{n}_C\mathbf{n}_i)^2}}, \qquad (3)$$

which evaluates to zero if the projection $\mathbf{q}'_j$ is inside the polygon. The distance of $\mathbf{q}_i$ to p_C is then conservatively approximated as

$$d(\mathbf{q}_j, p_C) = |\mathbf{n}_C\mathbf{q}_i - d_C| + d(q'_i, p_C) \qquad (4)$$

and eventually the surface to patch distance is set to

$$d(s_C, p_C) = \max_j d(\mathbf{q}_j, p_C), \quad \{\mathbf{q}_j\} \in s_C. \qquad (5)$$

The other direction is also computed based on distances to a sample set $\{\mathbf{q}_j\}$. This sample set is projected on the

patch leading to a set $\{q'_j\}$. Strictly speaking, one would have to compute the convex hull of this set and find the maximum distance between the line segments of this convex hull and the patch boundary. We approximate this distance by sampling a set of directions; this has the additional advantage that the directions can be later used to define clip planes. Let directions $\mathbf{n}_k$ be uniformly sampled from a circle perpendicular to $\mathbf{n}_C$ (i.e. all normals are parallel to p_C) and let $\{\mathbf{p}_i\}$ be the vertices of the polygon $\mathbf{p}_C$. Then the separation between the points $\{\mathbf{q}'_j\}$ and the vertices $\{\mathbf{p}_i\}$ in direction $\mathbf{n}_k$ is

$$d_{\mathbf{n}_k}(p_C, s_C) = \min_i \mathbf{n}_k \mathbf{p}_i - \min_j \mathbf{n}_k \mathbf{q}'_j \tag{6}$$

and the largest such directional distance is

$$d(p_C, s_C) = \max_k d_{\mathbf{n}_k}(p_C, s_C) \tag{7}$$

Ideally, a plane (n_C, d_C) should be computed to minimize the above mentioned distances. However, we find that this is too expensive and not really necessary in practice. Based on the sampled points, the normals n_C is usually computed as the direction of smallest co-variance of $\{q_j\}$, and d_C chosen that the plane contains the centroid of the samples.

2.2 Splitting to bound $d(\mathcal{S}, \mathcal{P})$

The basic idea is to minimize the co-variance of surface patches in the cell. The co-variance is connected to curvatures and also intuitively, flat patches result from splitting along areas of high curvature. We compute a suitable plane for splitting the surface in a cell into two surfaces with small total curvatures using the following

Primary Heuristic: The direction of largest co-variance of surface normals defines the normal of the split plane.

This choice is intuitive and computationally feasible: along the direction of largest normal curvature the co-variation of normal vectors is expected to be large. Splitting orthogonal to this direction minimizes the variation in normals and, thus, in total curvature. Computation only involves solving an eigen-problem of the 3×3 co-variance matrix of a set of sampled normals.

We have found this heuristic to perform better in many cases than those proposed in [19]. However, it fails in two cases: First, (approximately) cylindrical surfaces are always cut along one direction, leading to cells with very high aspect ratios and the known associated numerical problems. Second, cells covering unconnected parts of the surface need special treatment, which cannot be concluded from the Gauss image.

We have designed two tests to check if cells are suitable for the primary heuristic.

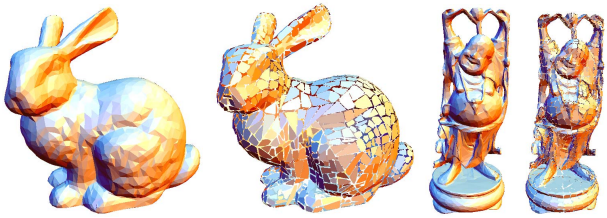

Figure 2. Comparing accurate piecewise linear approximation of surfaces. The Hausdorff errors to the original surfaces are small in all cases (ca. 0.002).

1. To approximate the aspect ratio of a possible planar approximation to the surface we compute the ratio of the second largest and largest eigenvalue of the surface inside the cell. This involves taking samples of the surface, computing the eingenvalues $\lambda_1, \lambda_2, \lambda_3$ of their co-variance matrix, and then checking if λ_3/λ_2 is too large. In practice, if the ratio exceeds 8 we conclude the cell is too elongated for proceeding with the primary heuristic.

2. To check if the cell contains more than one connected component, we compare the area of s_C with the area of a potential planar patch p_C. The area $A(s_C)$ is approximated based on a sampling: Let the average distance among neighboring samples be ρ (known from the sampling procedure or approximated from nearest neighbors) then

$$A(s_C) = \sum_j \pi \left(\frac{\rho}{\sqrt{2}}\right)^2 \tag{8}$$

   If $A(s_C) < \frac{1}{2} A(p_C)$ we conclude that the cell contains more than one connected component and should not be split based on the primary heuristic

In both cases we resort to a more conservative

Secondary Heuristic: The direction of largest co-variance of the surface defines the normal of the split plane.

This heuristic simply minimizes the co-variance of the surface patches. It is known to fail in many cases (see also [19]), however, these cases are usually covered by the primary heuristic. In the two cases we make use of it, it generates splits exactly where they are reasonable: Either along the longest axis, or separating disconnected components.

In our implementation, we always start with computing both eigensystems for each cell. This yields all the information necessary for defining the planar patch and applying the heuristics. Cells are split as long as $d(s_C, p_C) < \epsilon_{\mathcal{S}\to\mathcal{P}}$ and if they appear to contain more than one component (i.e. if they fail the second test as described above). While we cannot prove convergence for a continuous surface, the procedure necessarily converges for any fixed non-degenerate

sampling. Once converged, the overall one-sided error $d(\mathcal{S}, \mathcal{P})$ is bounded by $\epsilon_{\mathcal{S}\rightarrow\mathcal{P}}$.

2.3 Clipping to bound $d(\mathcal{P}, \mathcal{S})$

Once the distance from the surface to the patches is bounded, we further split cells to minimize the distance of patches to the surface. In our experience, in contrast to [19], this is necessary for many cells if a symmetric error is required. Our approach is related to the one described in [19]: remember that the distance $d(p_C, s_C)$ is computed based on a discrete set of normals $\mathbf{n}_k$ as the maximum of directional distances $d_{\mathbf{n}_k}(p_C, s_C)$ (see Section 2.1). We simply use n_k defining $d(p_C, s_C)$, i.e. the direction that leads to largest separation of one the vertices and the projected points on the polygon. The offset for this plane is so chosen that one of the two cells contains all points and the other one is empty; this effectively clips the polygon.

This procedure effectively decreases $d(p_C, s_C)$; the process terminates if $d(p_C, s_C) < \epsilon_{\mathcal{P}\rightarrow\mathcal{S}}$ for all non-empty cells.

2.4 Resulting shapes

We have applied the procedure described above to generate surface approximations for various models and error bounds. In our experiments we started from existing samplings (i.e. scanned data), rather than sampling a continuous surface, which would have been possible as well. Figures 2 and 3 show several examples. In particular we have compared meshes simplified based on quadric error metrics [11]: Meshes have been simplified to a given number of faces. The Hausdorff distance of the simplified meshes to the original vertex set has been computed using Metro [7]. Then we have generated BSP shapes with similar Hausdorff distance. The results are displayed in Figures 2 and 3, where meshes and BSP shapes in a column have similar Hausdorff distances (but different number of primitives).

Note that we have also checked the results of Simplification Envelopes [8], as this method explicitly bounds the Hausdorff error. Yet, the number of faces this approach generates is generally much larger than QSlim, which is why we have chosen the comparison described above.

Figure 3 compares the number of planar patches to the number of faces in the mesh relative to the Hausdorff distance. It also shows the number of triangles resulting from triangulating the faces. For coarse approximations (Figure 3, BSP shapes have significantly less faces (and also less triangles). This advantage is decreasing as the error bounds get tighter and the number of faces increase (Figure 2. Note that in applications where a BSP over the polyhedral shape is needed the number of faces indeed describes the complexity of the data structure and, therefore, is a practical predictor for the efficiency of spatial queries. For pure rendering purposes, of course, the number triangles is important.

This comparison is based on floating precision numbers and the images in Figures 2 and 3 show the faces flat shaded. In the following, we discuss how to store the BSP shape more efficiently using quantization and how to render the BSP shape in more visually pleasing manner.

3 Coding the BSP shape

The main observation for efficiently coding the BSP tree is that the precise orientation of planes for most cells is not that crucial. Consequently, normals and offsets can be quantized without too much effect on the resulting representation. Furthermore, quantization can be incorporated into the construction so that the resulting shape still satisfies the same error bounds, only with slightly more cells.

3.1 Coding cells

Normals are represented by a quantization scheme similar to [10]. The unit sphere is subdivided into 8 quadrants, which are again subdivided into 6 triangular sextants. Each of the sextants is now quantized by a regular grid in angle space. While this encoding is not perfect (ambigous codes exist for a few normals) it still delivers good results.

It might seem advantageous to code normals relative to the normals of prior split planes. However, we have found that normals in the BSP tree are almost randomly distributed so that there is no benefit in differential coding.

However, inner nodes can be coded using less bits, because their influence on the approximation is small. An additional bit can be saved because the orientation of split planes is irrelevant. For the leaf nodes, planes representing patches should be coded rather accurately, while clip planes can be coded based on a regular subdivision of a circle.

Note that all cells in the BSP are closed, because we intersect the BSP with the bounding box of the object, which is explicitly stored. Further, each splitting plane passes through the cell it divides. For these reasons, we encode the offset relative to the centroid of vertices of the cell and the magnitude relative to the size of the cell.

For the magnitude we compute the distances of cell vertices to the centroid along normal direction of the plane and use the largest distance to define 2^k regular intervals. Storing a quantized distance value then requires $k + 1$ bits.

The tree can be coded by traversing it in any order and storing the normals and offsets. In addition, leaf nodes have to be flagged (because the tree is not balanced) and these leaf nodes have to be distinguished into empty (resulting from clipping) and non-empty nodes. This requires a bit to distinguish inner nodes from leaves, and then an additional

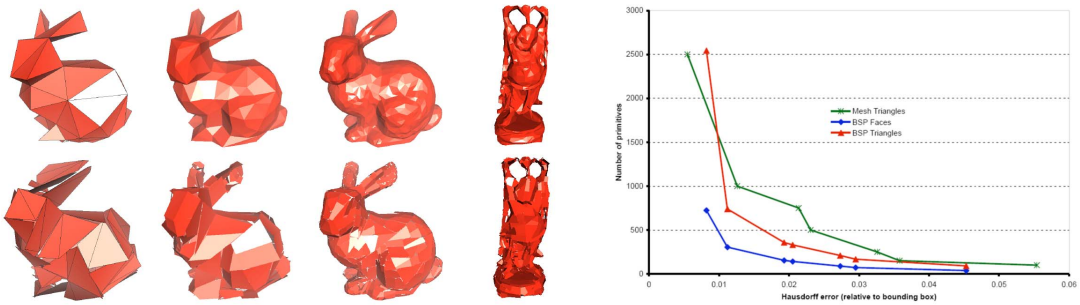

Figure 3. Left: Comparison of coarse piecewise linear approximation of surfaces. The Hausdorff errors to the original surface are similar for each column (ca. 0.05, 0.02, 0.005, 0.01), but the number of faces and triangles for disconnected pieces are smaller than for meshes. Right: A graph showing that the number of primitives for the same Hausdorff error is generally smaller for BSP shape than for meshes while the number of triangles is near equal.

bit for the leaf nodes to distinguish among surface patches and clip planes.

3.2 Resulting BSP shape codes

Aggressive quantization leads to small codes for each cell, but larger deviations of the planes from their optimal orientation and position, thus, larger errors and, eventually, more cells. More accurate representation requires more bits per cell but leads to overall fewer cell. So, in general, an error bound can be satisfied with a wide range of quantization settings, just leading to different BSP shapes (with possibly different code sizes). There is, of course, a limit to the quantization relative to the desired accuracy, i.e. a small error cannot be achieved with only 8 directions for the planar patches.

It has turned out to be difficult to deduce a general rule for choosing the quantization levels of the different types of cells. We distinguish the number of bits for interior nodes, planar patches and clip planes. For each of the three types the number of bits for the direction of the normal and the offset can be chosen.

Based on experiments we generated a sequence of quantizations with bit ratios roughly 3:4:2 for interior nodes, planar patches, and clip planes, and fixed error of 1% of the bounding box diagonal. As expected, neither too few nor too many bits per cell lead to overall good code sizes, however, few bits lead to already quite good accuracy, few cells, and good code sizes. Figure 4 shows the resulting code sizes relative to the bits per cell.

When comparing the resulting code sizes to compressed meshes with similar Hausdorff error, BSP shapes usually require slightly more data: In our example of 1% Hausdorff error, the corresponding mesh can be compressed to 1.5 KByte using state of the art mesh codecs (e.g. [4]), while the minimum tree size of BSP shapes among the ones we generated is 2.3 KByte. This code size is achieved without entropy coding, though we have found that standard compression tools have little effect on the code size. Only for very coarse approximations, we find bit rates that are similar to those of meshes. The spatial relationship among neighboring faces available in a mesh can be used for predictive coding of the geometry – an information that is missing in BSP shapes.

4 Conclusions

BSP shapes are a surface approximation combining a useful spatial data structure with accurate Hausdorff error bounded approximation. The construction of the representation is fast and depends mostly on the required precision. The resulting number of patches necessary for the representation is smaller than the number of faces in mesh representation with similar Hausdorff error in our experiments. The BSP tree can be stored efficiently using only quantization without any additional coding techniques. This means, the small code can be directly used for further processing without the need to perform any decompression steps. Code sizes are slightly larger than compressed meshes, but smaller than a mesh plus BSP constructed from the mesh faces.

We find that BSP shapes open the door to an interesting new area of surface representation techniques. Many things have to be further explored, though. Most obvious is the lack of considering accuracy in the normals, which adversely affects the visual appearance when using shading models. This effect is amplified when normal directions of patches are quantized aggresively . Controlling not only the Hausdorff error but also the error in normal approximation might remedy the problem.

We have already experimented with higher order approximations in the leaf nodes. However, when only the Hausdorff error is considered they yield similar results: fewer leaf cells are needed but they require more coefficients.

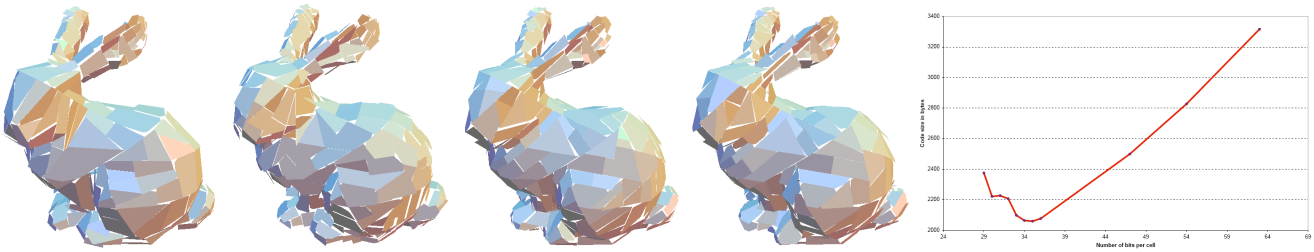

Figure 4. Shape approximations with an equal Hausdorff error of 1% pf the bounding box diagonal and succesively smaller overall code sizes. From left to right : No quantization, 17kByte. 15:20:13 bits, 3kByte. 14:18:12 bits, 2.5kBye. 13:16:10 bits, 2.35kBye. Right : Plot of the error compared to the average bits per cell. Using too few bits per cell requires extra cells to compensate for the quantization artifacts and leads to larger code sizes.

They could solve the appearance problem, but they are also more costly in rendering.

More fundamentally, while disconnected representations have the promise of delivering more accurate representations for the same number of primitives, it is not yet clear how they should be generated and exploited.

References

[1] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 230–239. Eurographics Association, 2003.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proceedings of the conference on Visualization '01*, 2001.

[3] P. Alliez. Personal communication about variational shape approximation, 2005.

[4] P. Alliez and M. Desbrun. Valence-driven connectivity encoding for 3d meshes. *Computer Graphics Forum*, 20(3):480–489, 2001.

[5] N. Amenta and Y. J. Kil. Defining point set surfaces. *ACM Transactions on Graphics (SIGGRAPH 2004 issue)*, 23(3), 2004. to appear.

[6] M. Botsch, M. Spernat, and L. Kobbelt. Phong splatting. In *Symposium on Point-based Graphics*, pages 25–32, June 2004.

[7] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measruring error on simplifiifed surfaces. In *Computer Graphics Forum*, pages 167–174, 1998.

[8] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, J. Frederick P. Brooks, and W. Wright. Simplification envelopes. *Proceedings of SIGGRAPH 96*, pages 119–128, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[9] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, Aug. 2004.

[10] M. Deering. Geometry compression. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 13–20, New York, NY, USA, 1995. ACM Press.

[11] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[12] D. Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*, 2003.

[13] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144. ACM Press/Addison-Wesley Publishing Co., 2000.

[14] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yields polyhedral set operations. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, volume 24, pages 115–124, Aug. 1990.

[15] B. F. Naylor. Interactive solid geometry via partitioning trees. In *Graphics Interface '92*, pages 11–18, May 1992.

[16] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, July 2003.

[17] Y. Ohtake, A. G. Belyaev, and H.-P. Seidel. 3D scattered data approximation with adaptive compactly supported radial basis functions. In *Shape Modeling International 2004*, pages 31–39, Genova, Italy, June 2004.

[18] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 185–194. ACM Press, 1992.

[19] J. Wu and L. Kobbelt. Piecewise linear approximation of signed distance fields. In *Vision, Modeling and Visualization 2003 Proceedings*, pages 513–520, 2003.

[20] J. Wu and L. Kobbelt. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum*, 23(3), August 2004.

[21] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 322–329. ACM Press, 2002.