Cone Carving for Surface Reconstruction

Shy Shalom
Bar-Ilan University

Ariel Shamir
The Interdisciplinary Center, Herzliya

Hao Zhang
Simon Fraser University

Daniel Cohen-Or
Tel-Aviv University

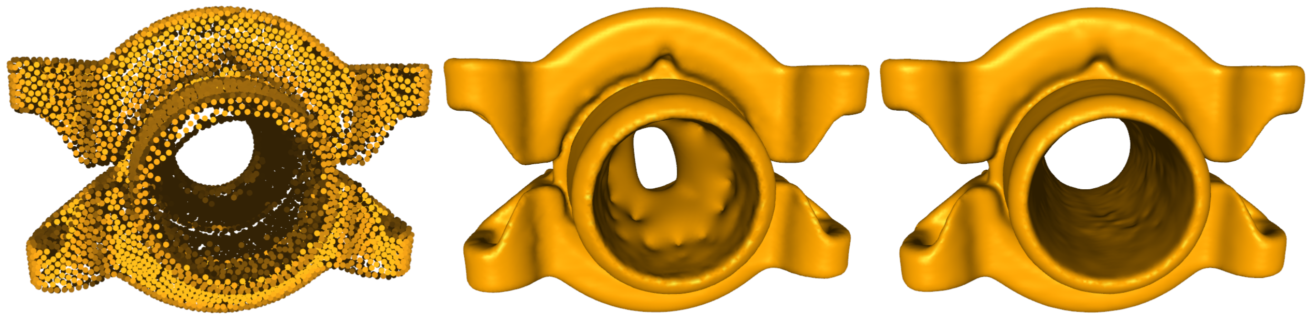

Figure 1: *With a significant amount of missing data in a scanned point cloud (left), conventional surface reconstruction schemes such as RBF are error-prone (middle). Cone carving, a space carving technique, utilizes global visibility information to more accurately infer the captured shape. It creates additional off-surface points specifically in areas of sparse input to produce a more faithful reconstruction (right).*

Abstract

We present cone carving, a novel space carving technique supporting topologically correct surface reconstruction from an incomplete scanned point cloud. The technique utilizes the point samples not only for local surface position estimation but also to obtain global visibility information under the assumption that each acquired point is visible from a point lying outside the shape. This enables associating each point with a generalized cone, called the *visibility cone*, that carves a portion of the outside ambient space of the shape from the inside out. These cones collectively provide a means to better approximate the signed distances to the shape specifically near regions containing large holes in the scan, allowing one to infer the correct surface topology. Combining the new distance measure with conventional RBF, we define an implicit function whose zero level set defines the surface of the shape. We demonstrate the utility of cone carving in coping with significant missing data and raw scans from a commercial 3D scanner as well as synthetic input.

1 Introduction

Surface reconstruction from a scanned point cloud is a central problem in computer graphics that has inspired constant research over the past years. Given a set of 3D points acquired by a scanning device, the goal is to construct a 2D watertight surface embedded in 3D which approximates the scanned shape as closely as possible, from its global topology down to the finer geometric details. Apart from noise, the most common difficulty with scanned point clouds is the existence of holes in the data: areas of the shape that were not properly acquired. In particular, large holes often occur in deep, hard to reach cavities of the object, or places where object parts are too close to each other. These impose serious adverse effects on the quality of the reconstruction, particularly in terms of its topology.

One of the more successful techniques to recover a watertight 2D surface from point clouds has been to define an implicit function over the ambient space of the underlying shape and take the zero level-set of the function. Examples of this technique include Radial Basis functions [Carr et al. 2001], Partition of unity [Ohtake et al. 2003], and Poisson reconstruction [Kazhdan et al. 2006]. In many cases, the implicit function is defined using some *blending kernel* of local neighborhoods defined over a point set. To better estimate this function, the set of sample points, which are considered *on-surface points*, is augmented by a set of *off-surface* points that carry a signed distances to the perceived surface. For instance, in [Carr et al. 2001] *dipole* off-surface points are created by offsetting the input points along their estimated normal directions, on either side of the perceived surface. However, a major problem with using dipole points for surface reconstruction occurs over areas with sparse or missing data. Without sufficient point sampling, one cannot define the dipole points reliably. For similar reasons, previous implicit surface reconstruction methods can only fill holes smoothly. More often than not, this leads to topological or geometric reconstruction errors, e.g., see Figure 1 (middle).

While previous algorithms have effectively used blending of the *positional* information of points, in this paper we utilize blending of *global visibility* information as well. This information can effectively define a signed distance function to the surface even in regions where sample data is missing. We assume that the scan of the surface is performed in such a way that each acquired sample is visible from a point outside the shape. In other words, from any point sample, there exists a ray emanating from the point that lies completely outside the shape. We call the direction of this ray the *outward* direction. This is a rather weak and realistic assumption since for most optical scanners and under typical acquisition settings, an outward direction is already known for each sample point — it is the direction from the point to the scanner head position at the moment of the capture.

Using the outward direction we convert each sample point to a *visibility cone* apexed at the point (see Figure 2). These cones are assumed to be exterior to the underlying shape. The key idea is that

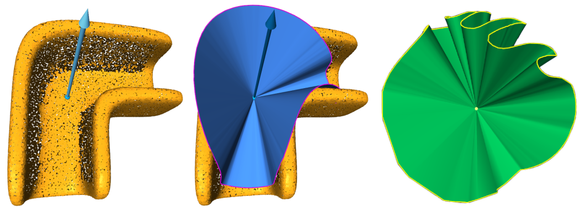

Figure 2: *Visibility cones. (a) A 3D point cloud with a point sample and an outside direction at the sample. (b) The corresponding ideal (maximal) visibility cone. (c) In general such 3D visibility cones can have rather complex boundaries.*

such cones extend beyond the direct outward direction, carve out the outside space and therefore, carry global visibility information that can be used to define a more reliable distance measure to the shape near regions with sparse input data.

Let p be a point in the input point cloud P which samples the scanned shape S. We define the visibility cone of p as the largest generalized cone apexed at p which contains an outward direction at p, with no other point from P in its interior. The ideal visibility cone at p would be one that is extruded from p along the *silhouette* of S as viewed from p, as shown in Figure 2. The union of all the visibility cones from P provides a virtual *carving* of the ambient 3D space around the underlying shape S. The set of visibility cones collectively carry global information about the shape and offers a robust indicator of *outside* relations with respect to S, specifically near regions of sparse data and cavities.

The interface between the inside and outside space defined by the visibility cones is given by the boundary of the volume formed by the union of all cones. We call this boundary surface E the *lower envelope* of the set of visibility cones as it is viewed from the inside-out perspective. In the ideal continuous setting when $P = S$ and under the visibility assumption, the lower envelope E is simply S. In our discrete settings, where $E \neq S$, we still utilize the distance to E as an approximation to the distance to S. We define a new set of off-surface points and use E to approximate the signed distances from these points to the surface. Instead of blending the cones to construct the lower envelope E, we use statistical *blending* of individual distances from the new off-surface points to the cones to arrive at a reliable and robust distance measure.

The practical challenges of creating the cones in the discrete setting also include the construction of approximate 3D visibility cones based on the point cloud P. Towards this end, we adopt an image-based approach, exploiting the rendering speed and capabilities of the GPU. Using a point-rendering process with adaptive splat sizes, we project the point cloud onto a small cube around each $p \in P$ and trace the appropriate silhouette contour on the cube for extruding a visibility cone from p. In essence, this process can be seen as a view dependent *blending* of the input point cloud defining the local visibility silhouette at each point sample p.

To summarize, our algorithm relies on blending at three levels. First, the local position of sample points are blended similar to previous algorithms. Second, view-dependent positional information is blended to define visibility cones for all sample points. Third, distances to the cones from new off-surface points, which we term *cone-based off-surface points*, are blended to arrive at a robust distance-to-surface measure. These blendings combine local positional and global visibility information for more effective surface reconstruction as shown in Figures 1 and 11.

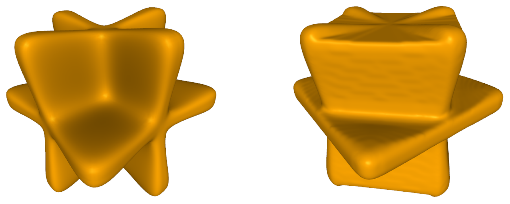

Figure 3: *For objects containing cavities (left), space carving from the outside (e.g. silhouette carving) can fail to reconstruct the correct shape (right). In contrast, cone carving can exploit the full potential of visibility information (see Figure 11)*

2 Related works

[Curless and Levoy 1996] were the first to combine positional and visibility information for surface reconstruction using space carving in VRIP. However, they use the conservative line-of-sight from the sampled points to the scanner to clear out “empty” voxels. We carve larger parts of the outside space using continuous cones based on adaptive sampling of the distance between the sample points. Our technique does not depend on the voxel resolution and does not need explicit knowledge of the scanner head’s location. Moreover, in cases similar to Figure 1, the VRIP algorithm would need to place a “backdrop” surface while scanning to recover the inner part of the cylinder. In contrast, our cones carry global visibility information that successfully handles such cases.

Space carving is also used for reconstructing a 3D shape using images taken from multiple views [Kutulakos and Seitz 2000; Montenegro et al. 2004; Nitschke et al. 2007]. These methods create a rough volumetric estimation of the shape and do not aim at surface reconstruction from point clouds (see Figure 3). Furthermore, the outside space is defined by carving out space using rays emanating from the camera position towards the object and through its silhouette (Figure 3), while we use the inverse direction and carve out cones originating at each sample of a point cloud. One exception to this is the work by Laurentini [1994], where the underlying concept of cone carving, has been explored from a theoretical stand point. Bounds on the types of surfaces that may be reconstructed with this technique are given with an $O(n^6)$ theoretical algorithm. However, an implementation of this algorithm in a real world setting using point clouds was not explored.

In general, reconstructing a surface from a scanned, unstructured point cloud is widely regarded as a difficult and ill-posed problem. The pioneering work of Hoppe et al. [1992] computes a signed distance field to the underlying surface using local tangent plane construction, and extracts an isosurface from the distance field using the marching cubes algorithm. Over the years, other implicit formulations have been proposed, including RBF [Carr et al. 2001], Partition of unity [Ohtake et al. 2003], and Poisson reconstruction [Kazhdan et al. 2006]. These methods all rely on acquired [Nehab et al. 2005] or estimated [Alexa et al. 2003; Hoppe et al. 1992] point normals. However, with the presence of a variety of data artifacts due to measurement noise, scan mis-registration, or self-occlusions, obtaining reliable normals everywhere is not an easy task and is a subject of intensive research in its own right [Huang et al. 2009].

Voronoi-based approaches typically do not require normals. They attempt to reconstruct the surface using Voronoi diagrams or Delaunay triangulations and provide proper sampling conditions under which the reconstructed surface has provable theoretical guarantees [Amenta and Bern 1998; Amenta et al. 2001; Dey and Goswami 2003; Cazals and Giesen 2006]. These methods, however, are limited when faced with severe under-sampling or non-uniform sampling. A particular Delaunay-based technique, which uses the concept of α-shapes [Edelsbrunner and Mücke 1994], is worth mentioning, as the closely related concept of α-hulls can be presented using a "carving" analogy. Using empty balls of radius α to carve the ambient space around a mesh, the α-hull technique can be used for topological simplification [El-Sana and Varshney 1998]. Surface reconstruction using α-shapes extracts a surface from the Delaunay tetrahedralization of the input point cloud using empty α-balls as a means of filtering. Giesen et al. [2006] use α-balls with adaptive radii to deal with non-uniform sampling. However, this class of methods still only return *sub-complexes of the Delaunay tetrahedralization*. With significant missing data, the Delaunay tetrahedralization may not contain any sub-complex that is a proper surface reconstruction; see Figure 4(b).

When data is missing in the input scans, some existing surface reconstruction techniques, e.g., Voronoi-based Power Crust [Amenta et al. 2001], tight Cocone [Dey and Goswami 2003], or implicit schemes using radial basis functions (RBF) [Carr et al. 2001], can still guarantee an output that is a water-tight 2-manifold. However, in the presence of large holes or close-by surface sheets, these approaches typically produce topological or geometric errors. These are complicated scenarios where purely local considerations, such as point normals or Voronoi constructions, are insufficient to infer the shape correctly and additional information or assumptions are needed. One can try and identify these as surface boundaries [Dey et al. 2009], or utilize hole-filling. Several existing algorithms for hole filling rely on additional data or assumptions, such as multiple data scans [Turk and Levoy 1994], templates [Pauly et al. 2005] or specialized shape prior [Gal et al. 2007; Schnabel et al. 2009; Tagliasacchi et al. 2009]. Visibility information can also be utilized by volumetric diffusion [Davis et al. 2002]. When the holes are easy to detect or are specified as part of the input, several approaches can be applied to achieve high-quality surface completion; these include the use of smooth interpolants [Sorkine and Cohen-Or 2004] or self similarities in shapes [Sharf et al. 2004].

Our cone carving technique deals with significant missing data in surface reconstruction while relying on a weaker and more practical assumption: the existence of an outward direction at each point sample. Typically, the outward directions can be returned by the scanner head positions, which are available from some brands of laser scanners. Some commercial products, such as the software bundled with the Polhemus FastSCAN scanner [Aranz 2009], are known to use this information to some extent – but usually only for coarse tasks such as determining the correct sign of normal vectors. We are not aware of previous works that utilize such information for surface reconstruction.

3 Visibility cone carving

To illustrate the idea behind cone carving, we first describe a simplified version of the algorithm in 2D. Our input is a 2D point set with missing data such as the one shown in Figure 4(a), which is similar to a typical output of a range scan for a "π" shape. Boundary lines that are closely parallel to the scanning direction are likely to suffer from sparse sampling and result in large holes since the grazing angle in which the scanner beam hits the shape boundary is too shallow and does not reflect light properly.

Dipole distance and distance to lower envelope With an input point set given in Figure 4(a), standard reconstruction algorithms such as RBF [Carr et al. 2001] would produce undesirable results. Regular RBF uses dipole off-surface points, which are created at a constant distance $+c$ and $-c$ from an input point p along the direction of the estimated normal; see 4(c). As a convention, the signed distance takes on positive values when inside the shape and negative values when outside. Based on the dipole points, RBF interpolation implies a signed distance field, which we refer to as the *dipole distances* or DP, over the ambient space, as shown in Figure 4(d). We can observe that in areas where there are large gaps between samples, no off-surface points are created. Consequently, the interpolation produces smooth curves as part of the zero level-set of the DP distance field, which may not give a correct representation of the underlying shape. One the other hand, the cone carving algorithm uses the additional global visibility information, resulting in a better implicit distance function to infer the shape boundary.

For an input point p, the visibility cone at p in 2D is the largest sector that can be stretched from an outward direction without containing any other input points, as shown in 4(e). The assumption at the base of cone carving is that the area inside the visibility cone is completely outside the shape. To take advantage of the global visibility information we construct a visibility cone for each point of the input set. The union of all the cones serves as an approximation of the exterior ambient space around the shape, as shown in 4(f).

In 2D, the visibility cone sector is delimited by two rays emanating from p. Each of these rays must pass through some other point of the input point set. If the position of the scanning head is known for point p, an outward direction is known. Hence, we calculate a ray from p to the scanner head and sweep it radially as much as possible until hitting other cloud points. When this direction is unknown, we assume that the cone sector with the *largest opening angle* is the one which contains the outward direction. Hence, we can search for the largest empty opening between two points as viewed from p to construct the visibility cone for p. Note that this assumption may not always hold and erroneous cones may be created. However, in practice these situations are rare and do not effect the algorithm in general. This is because, as will be seen later in the 3D case, the individual visibility cones are essentially blended statistically to achieve robustness against the potential errors.

Recall that the boundary of the union of the visibility cones has been defined as the *lower envelope*. In the 2D case, each visibility cone assumes a particularly simple structure and therefore we can explicitly construct the lower envelope E. Next, we define the DTE or *Distance-to-Envelope* as the shortest distance to E. Figure 4(g) shows, for the 2D example, a plot of the DTE distance field. As a convention, the signed distances take on positive values when outside the union of the visibility cones, and negative values when inside. Note that this choice of signs is consistent with the choice of distance signs with respect to the underlying shape.

Choosing between distances Using a quadtree subdivision of the ambient space (octree in 3D), we sample the DTE field to create cone-based off-surface points. The tree is initialized with a uniform grid of points and subsequently subdivided into cells containing sign transitions in the DTE measure. This creates an adaptive sampling of points that concentrate near the surface, but does not rely on the distribution of the input points. This allows one to sample the distance field in areas of sparse or missing data. On the other hand, areas with sufficiently dense input data are well covered by the dipole points and the implied DP distance field. The decision on which distances, DP or DTE, to choose in each region dictates how the off-surface points are generated for the final RBF process.

Let us recall that the collection of visibility cones serve to carve out the *exterior* or *outside* space with respect to the underlying shape. As such, these cones are only reliable as *outside* indicators. Indeed, while any point inside a visibility cone should be assumed to be outside the underlying shape, a point outside all visibility cones should not be assumed to be inside. In our discrete setting, gaps

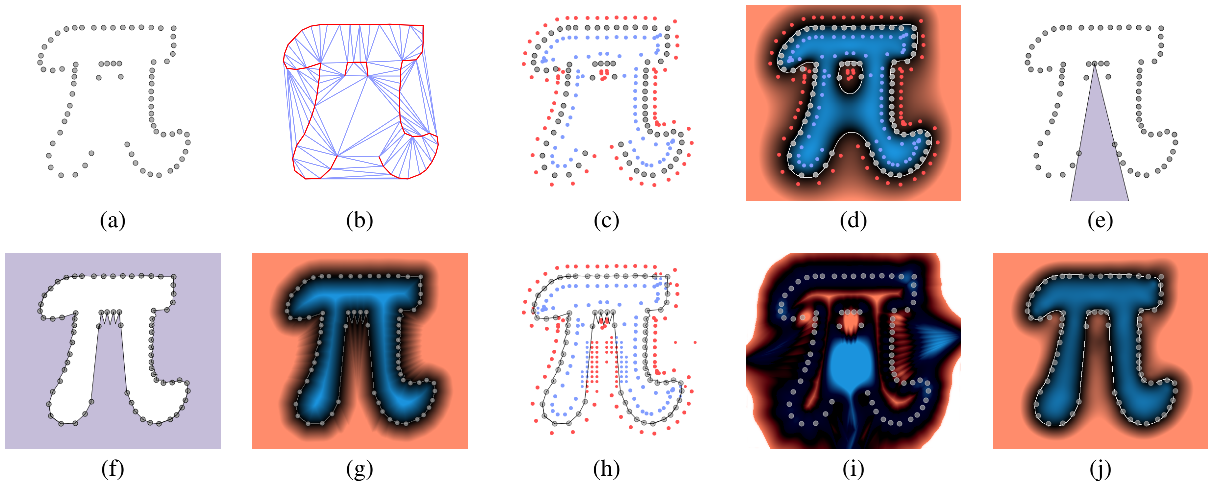

Figure 4: *Cone carving illustrated in 2D. (a) Input point set. (b) The Delaunay triangulation - with significant missing data, no sub-complex of the triangulation (e.g. using α-balls) is a proper surface reconstruction. (c) Dipole off-surface points for RBF reconstruction. (d)* DP*: implicit distance field created by RBF (blue: positive and inside the shape; orange: negative and outside), also does not provide proper reconstruction due to missing pieces. (e) A single visibility cone. (f) Outside carved space using all cones. (g)* DTE*: implicit distance field derived from the lower envelope of the visibility cones (blue: positive and outside the union of cones; orange: negative and inside). (h) Combined set of off-surface points, dipole points and cone-based points generated from an octree using* DTE*. (i) Difference between the two implicit distance functions $\Delta =$ DP $-$ DTE. (j) Final RBF reconstruction from (h) using (i).*

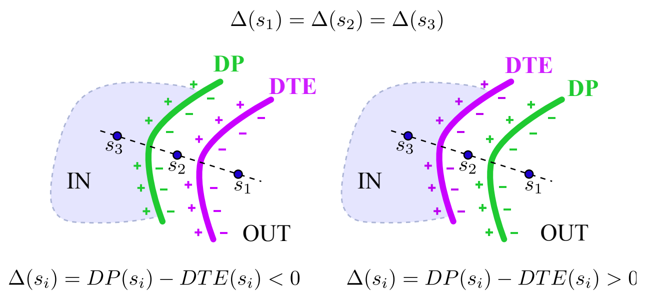
$\Delta(s_1) = \Delta(s_2) = \Delta(s_3)$
DP
DTE
IN
OUT
s_3
s_2
s_1
$\Delta(s_i) = DP(s_i) - DTE(s_i) < 0$
$\Delta(s_i) = DP(s_i) - DTE(s_i) > 0$

Figure 5: *The difference Δ between the* DP *and* DTE *distances dictates the conjectured position of the boundary of the underlying shape, either purple or green. $\Delta(s_i)$ is the same in both situations, $i = 1, 2, 3$. When $\Delta(s_i) > \lambda > 0$ (right), where λ is a threshold, the* DTE *measure carves into the interior (positive sign) space indicated by the* DP *measure. As* DTE *is a reliable indicator of outside (negative sign) relation to the shape, the purple boundary is assumed and the sign of s_2 is inverted. In contrast, when $\Delta(s_i) < 0$ (left),* DP *takes precedence over* DTE.

between the lower envelop and the actual boundary of the shape may appear; see top of the cavity in the "π" in Figure 4(f). Thus the key situation where the DTE distance should be chosen over the DP distance is at a region which the DP distance incorrectly indicates to be inside the shape – this is illustrated at the right of Figure 5.

Let DP(s) and DTE(s) denote the scalar values at a point s in the ambient space as given by the DP and DTE distance fields, respectively. We denote the difference between DP(s) and DTE(s) as $\Delta(s)$ = DP(s) − DTE(s) and examine the scalar field $\Delta(s)$, as shown in Figure 4(i). We differentiate between three possible cases of $\Delta(s)$ with regards to a threshold value $\lambda > 0$:

- $|\Delta(s)| < \lambda$: When the two distances are similar, we choose the DP distance since it tends to be more accurate. Indeed, the DP distances are derived from dipole points, which in turn are obtained from local on-surface point samples. On the other hand, the DTE distances use global visibility information which can be less accurate.
- $\Delta(s) > \lambda$: This occurs when the DTE distance "places" point s significantly (more than a threshold) towards the outside of the shape than the DP distance. We choose DTE since it is a reliable outside indicator. In this case, the visibility cones carve out a region which the DP distance indicates to be inside the shape, as shown in the right of Figure 5. This is precisely the situation where the global visibility information serves to correct potential reconstruction errors.
- $\Delta(s) < -\lambda$: This occurs when the DTE distance "places" the point s significantly towards the inside of the shape; see left of Figure 5. We choose DP over DTE since the latter is not a reliable indicator of the inside relation.

To summarize, we use the DTE distance only where the difference Δ is positive, indicated by color blue in Figure 4(i), and above the threshold λ. Over these areas, the DTE distance is most likely to provide a better estimation for the distance from the underlying shape. In practice, such scenario typically occurs over regions with sparse or missing input data and cavities. We have used the value $\lambda = d_{\text{avg}}$, where d_{avg} is the average for all points in the data-set of the distance to their nearest neighbor (see Section 6.2). Combining the two distances we obtain a hybrid set of off-surface point values, as shown in Figure 4(h). These points together steer the final RBF reconstruction, combining the merits of traditional local position blending and global visibility information as shown in Figure 4(j).

4 Visibility cones in 3D

The 3D visibility cone of a point p is an empty generalized cone apexed at p. It is a cone-like volume of rays, originating from p and extending towards the outside of the shape. The generalized cone is empty in the sense that it contains no other input points. If the *exact continuous* boundary surface of the shape is known, the 3D visibility cone of a point p can be defined by projecting the silhouette of the shape from p's viewpoint towards the outside direction. In practice, the construction of visibility cones from a point cloud in 3D is an ill-posed problem since there is no trivial way to delimit a generalized 3D sector as we did in 2D using two rays. It seems that to construct the visibility cone one must first reconstruct the smooth surface. Instead, we employ a view-dependent blending of the point cloud using splatting [Westover 1990; Pfister et al. 2000] from the perspective of each point p.

View dependent blending We extend the definition of a 3D cone to exclude not just points from the data-set, but neighborhoods of size σ around each point. The value of σ is key to producing useful visibility cones that conform accurately to the shape of the point set. A small σ may produce cones that protrude considerably into the shape, leading to "cracks" between adjacent points, while a large σ can produce cones that are too far removed from the point set, losing vital information regarding the original scanned shape. We use an adaptive neighborhood size σ as the splat size of each rendered point, depending on its distance to the center of projection p.

To construct the approximation of an ideal visibility cone of a point p, we perform two passes of rendering with a similar setup. Such a two pass approach is often used in kernel density estimations as well. Both rendering passes create a projection of the point set onto an axis-aligned six sided cube, centered around p, which is regarded as the camera position. In the first rendering pass, we estimate the adaptive neighborhood size σ for each rendered point. In the second rendering pass, we use this size as the splat size for each point and trace the cone silhouette on the view-dependent point-splat image of the point set. A single rendering pass results in six square images, each created by a perspective projection with a 90° field of view, pointed in the directions of a major axis and centered at p. This rendering setup is similar to the radiosity hemi-cube method [Cohen and Greenberg 1985] commonly used for calculating form-factors, with the difference that we render a full 6-sided cube instead of a 5-sided Hemi-cube; see Figure 6.

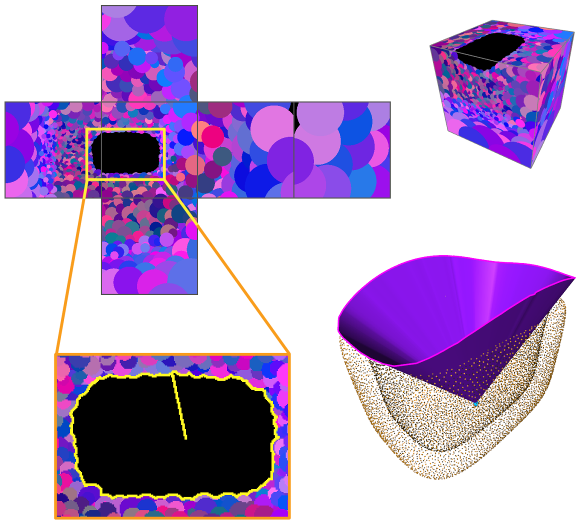

Figure 6: *Rendering the point set as circular splats on a cube around a center point reveals the visibility direction. The silhouette of the outside region is traced to define the visibility cone.*

Splat size The purpose of the first rendering pass is to determine the adaptive splat size for every rendered point. This can be seen as reconstructing a complete surface in image space using a k-nearest neighbor kernel function around each point.

Our camera is positioned at point p and all other points in the point set are rendered on the cube as a *single* pixel with a *unique* color and hence can be uniquely identified. The algorithm then sweeps over the six resulting hemi-cube images, calculating for each visible point q the splat radius r_p^q that produces a sufficient cover of the area around q when viewed from p. The immediate vicinity of the pixel representing q is scanned in growing concentric squares in search for other near-by non-background pixels. This scan is terminated either when one pixel is detected on $N_{\text{near}} = 3$ different sides of the growing squares, or N_{far} pixels are observed in total (see Figure 7). We use $N_{\text{far}} = 3 \times N_{\text{near}}$. The splat size is then set to the distance to the last pixel but not more than half the hemi-cube size. These values work well for all data-sets we have experimented with (see Section 6.2 for discussion). Note that some points may not receive any splat radii since they were occluded by other pixels in the rendered images. These points are excluded from the second rendering step since they do not contribute to the desired shape silhouette.

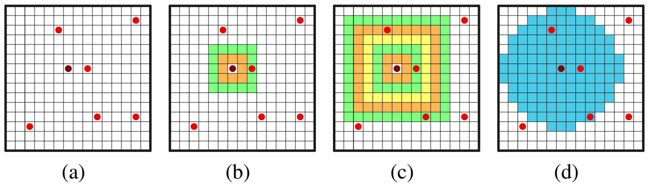

Figure 7: *Determining the splat size. (a) Every point is rendered as a single pixel. (b),(c) Scanning in concentric squares until 3 widely spread pixels are seen. (d) The splat is sized according to the distance to the third neighbor.*

Visible silhouette tracing In the second pass of rendering, the point set is again rendered on a cube around p, where each rendered point q is a circular splat with radius r_p^q and with a unique color. Next, we trace the silhouette edge between background and non-background pixels on the rendered image. The tracing begins at a point that is known to be on such an edge, and proceeds in a clockwise path along the silhouette. The trace is guaranteed to eventually reach the starting point since the edge between background and non background pixels is closed and continuous. The complete path will define the silhouette of the visibility cone from p.

Finding a starting edge point to initiate the tracing can be achieved in several ways. If the scanner position is known for point p, it is mapped to a pixel on the rendered cube. From this pixel, the silhouette edge can be reached by shooting a ray in any direction. If the scanner position is unknown, we randomly sample pixels on the cube and search for a silhouette edge in random directions. If there is more than one visible opening to the outside from p, more than one silhouette can be found. We select the silhouette with the longest traced path as the visibility silhouette. Four to five such random points are usually sufficient for finding the major silhouettes which define the maximal visibility cones.

During its pass on the cube, the tracing process stores the sequence of points it touches on the silhouette according to their splat colors. This sequence creates a closed 3D polyline. We create the boundary surface of the generalized cone C as a closed triangle fan $C = \{f_1, \ldots, f_k\}$, with p as the apex and triangles f_i extending between every two adjacent points of this polyline and p. The definition of a generalized cone has infinite boundary rays, hence, ideally, the triangle fans should be extended to infinity. In practice, we use finite triangle fans to represent the cones since that is where the cones touch the surface and beyond it they are irrelevant for the purpose of calculating the DTE distances.

The polyline produced by silhouette tracing can often be jagged, containing short sequences of repeating edges or even local self intersections. These aberrations can be attributed to the discrete nature of the point set and its rendering and may hinder the following steps. Therefore, after the silhouette polyline is created, we use a simple smoothing step to ensure that the cone conforms better to the shape of the point set. We discard repeated points as well as high curvature points and repair self-intersections by reordering points. Furthermore, due to discretization, the obtained visibility cone may deviate from the ideal one which would have been produced with a continuous surface. These inaccuracies can cause a wrong indicator both inside and outside the shape: small penetration of cones into the shape, and outside regions which remain outside the cones. Still, the robust blending of the DTE distances alleviates such inaccuracies, as we will explain in the next section.

5 3D reconstruction by cone carving

The surface constructed by cone carving is the result of an RBF fitting which requires as an input a set of off-surface points, each coupled with a signed distance value. The distance value is positive for points inside the shape and negative for points outside, and its magnitude can be perceived as the shortest distance from the off-surface point to the surface to be reconstructed. Using these off-surface points, we compute a set of radial basis functions whose combination produces an implicit scalar function in 3D. The zero iso-surface of this implicit function interpolates the shape. The final surface mesh is produced by a triangulation of the iso-surface using a variant of the marching tetrahedra algorithm [Carr et al. 2001].

Similar to the 2D case, the distance field to be sampled combines the DP and DTE distances. The DTE distances are created by using an *octree* to sample cone-based off-surface points as described in Section 3. However, in contrast to the 2D case, the collection of 3D visibility cones do not provided a guaranteed outside indicator. In other words, a point q inside the union of the visibility cones is not guaranteed to be outside the underlying shape. This is due to the approximative nature of the 3D visibility cones. Hence the computation of the DTE distance needs to be more elaborate.

Recall that DTE is intended to represent the distance from a point in the ambient space to the surface of the underlying shape. Without knowing the actual shape, we approximate its surface by the lower envelop E, which is the boundary of the union of the 3D visibility cones. Due to the imprecisions of the 3D visibility cones, the approximation quality of the lower envelope is degraded. One solution would be to blend the cones to represent a more robust lower envelope E. However, the geometric complexity of the lower envelope in 3D prohibits constructing it explicitly. Moreover, we are mainly interested in computing the DTE distances and not E itself. Hence instead of blending the visibility cones to create a surface, we resort to robustly blending the distances to the cones to define DTE.

DTC: Distance-to-Cone A 3D visibility cone C divides the ambient space into two regions: the region inside or on C and the region outside C. Let s be a point in the ambient space, we define the scalar function $\text{DTC}(s, C)$, where DTC stands for *Distance-To-Cone*, to be the signed distance from s to the cone boundary:

$$\text{DTC}(s, C) = \begin{cases} -\min_i d(s, f_i), & \text{if } s \text{ is inside or on } C, \\ \min_i d(s, f_i), & \text{if } s \text{ is outside } C, \end{cases}$$

where $d(s, f_i)$ is the Euclidean distance from s to a boundary (triangle) face f_i of the visibility cone. The DTC can be obtained by iteratively searching over all the triangle faces of the cone. Whether s is inside or outside the cone is determined by observing the orientation of the closest face and by using an appropriate normal computed at the apex of the cone [Baerentzen and Aanaes 2005].

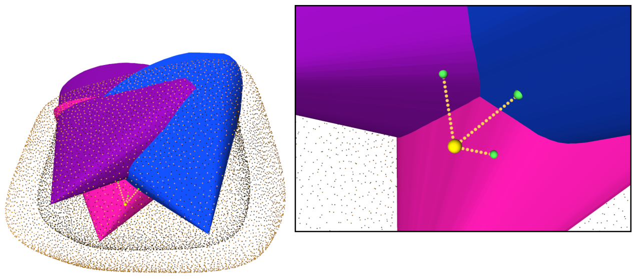

Figure 8: *Examples of three visibility cones of a points set (left). The distance $\text{DTC}(s, C_i)$ from a point s (yellow) to the cone C_i is the distance between s and its projection on the cone (right, green points). Note that although the distance values may be similar, they can be along totally different directions from s.*

Representative cone We are interested in the DTE distance $\text{DTE}(s)$ yet we do not explicitly construct the lower envelope; what we have been able to compute are the DTC distances to all the visibility cones. To make a connection between the two distances, we note that there is a simple relationship between them:

$$\text{DTE}(s) \leq \min_j \text{DTC}(s, C_j), \tag{1}$$

where the C_j's are all the visibility cones. To prove this result, we note that $\text{DTE}(s)$ is the shortest distance from s to some point v on the lower envelope. By definition, v must lie on or outside each visibility cone. Let C_j be *any* visibility cone and we consider two possible cases:

1. s lies inside or on C_j: Then $\text{DTC}(s, C_j) \leq 0$ and $\text{DTE}(s) \leq 0$. Since v is on or outside C_j, we must have

$$|\text{DTC}(s, C_j)| \leq ||s - v|| = |\text{DTE}(s)|.$$

Hence, $\text{DTE}(s) \leq \text{DTC}(s, C_j)$.

2. s lies outside C_j: Then $\text{DTC}(s, C_j) > 0$. If s lies on the lower envelope or inside the union of all visibility cones, then $\text{DTE}(s) \leq 0$. It follows that $\text{DTE}(s) \leq \text{DTC}(s, C_j)$. In the final scenario, we have s lying outside the union of visibility cones. Let u be on the boundary of C_j such that $||s - u|| = \text{DTC}(s, C_j)$. Then clearly u is either on the lower envelope or it lies inside the union of cones. Therefore, we must have $\text{DTE}(s) \leq ||s - u|| = \text{DTC}(s, C_j)$.

Since $\text{DTE}(s) \leq \text{DTC}(s, C_j)$ for all C_j, we get equation (1).

Hence, we see that the best approximation of $\text{DTE}(s)$ using the set of all DTC distances is given by the DTC distance from point s to the *representative cone* C_j^*, and we define:

$$\text{DTE}(s) = \text{DTC}(s, C_{j^*}), \quad j^* = \arg\min_j \text{DTC}(s, C_j)$$

To compute distance to the representative cone C_j^*, we can find the projection $Proj(s, C_j^*)$ of s onto the surface of C_j^* (see Figure 8). Note that such a projection can reside on the interior of a cone face, in which case it would be a perpendicular projection onto that face. In other cases, the projection may be along an edge of the cone or may be the apex of the cone.

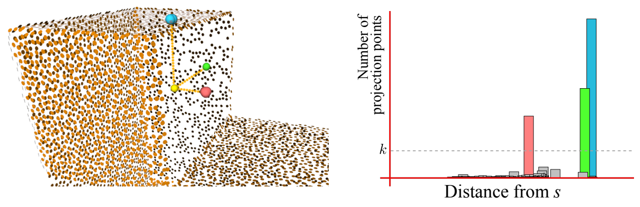

Figure 9: *Example of distance blending: the yellow point has several significant clusters of distances to various areas (see histogram on the right).* DTE(s) *is chosen as the median of the minimal significant cluster (red). The cluster is found using a sliding kernel window on the histogram starting from the lowest value. Note that this selected cluster actually lies on a missing face (left) exemplifying the strength of cone carving to fill in missing pieces based on visibility information.*

In practice, we can not rely on any single representative cone since such a cone may be erroneous due to the discrete way the 3D visibility cones were constructed. Instead, we blend the distances to several cones robustly to provide a better estimator for the ideal DTE distance (free from erroneous visibility cones), which in turn approximates well the distance to the underlying shape.

Cone distances blending There are two main reasons to combine several values of cone distances: first, to remove outliers, and second, to create a larger support for a specific distance measure. A naïve approach to blend cone distances would be to create the histogram of distances from a point s to all cones, use a blending kernel function, and take the minimal significant value according to equation (1). A significant value would be a value whose support (i.e. frequency) is larger than a threshold. Nevertheless, the same distance value from s may originate from cones in a totally different regions of the shape (see Figure 8), only one of which gives the correct minimum. Therefore, we employ a two-stage approach by using the histogram to search for the minimum, while blending 3D positions instead of distances using clustering (see Figure 9).

Starting from the minimal DTC value d_0 at s, we use a sliding window of size $2r$. For each value d, we examine all the cone projections v from s whose distance to point s falls inside the bin $[d, d+2r]$. Denote the set of cones corresponding to these projected points by $V_d(s)$. Since these points can come from various regions in 3D space, we cluster them using a simple hierarchical scheme [Everitt et al. 2001] in 3D space with a threshold of r. We use the cones of the first cluster found that contains more than k points to define DTE(s). Denote these cones by $L(s) \subset V_d(s)$, we take the *median* of the distances from s to all cones in $L(s)$ as DTE(s): DTE$(s) = \text{median}\{\text{DTC}(s, C) | C \in L(s)\}$. To adequately filter out outliers but still blend several cones we used the values $k = 5, r = 0.8 \cdot d_{\text{avg}}$ for all examples in this paper, where d_{avg} is the average for all points in the data-set of the distance to their nearest neighbor.

6 Results

We have tested our algorithm on several challenging examples in terms of missing parts and self occlusions, where the tested data were either scanned, taken from the AIM@SHAPE shape repository or artificial. The point clouds used typically contain 5,000-14,000 points. The algorithm is implemented in C++ and the GPU part using GLSL and CUDA. We used Intel Core2 Quad T9550 with an Nvidia GeForce 275 GTX video card. Table 1 shows the timing results for the models shown in the paper.

Model	# Points	Cones time	DTE time	Total time
Open L shape	5,124	1:52 m	0:31 m	2:30 m
Inukshuk	11,808	5:10 m	1:11 m	6:35 m
12-point star	11,710	5:02 m	0:30 m	5:57 m
Mannequin	12,220	4:55 m	1:50 m	7:04 m
Hand	13,162	5:19 m	1:23 m	6:58 m
elephant	14,295	6:30 m	2:05 m	9:05 m

Table 1: *Running times for the different results models in minutes. The total time includes all stages of the algorithm, including the final points merge and surfacing.*

In Figures 11 we show several examples of our results. We first show the point set using splats to give a sense of the overall shape and data. Note that these objects present a challenge to previous reconstruction algorithms, since they contain parts that are inaccessible from the outside for sampling. We compare our results to three other algorithms: Radial Basis functions [Carr et al. 2001], Partition of unity [Ohtake et al. 2003], and Poisson reconstruction [Kazhdan et al. 2006]. All these methods encounter problems with misinterpreting the outside as inside and thus connecting close parts to form wrong topology. Using our global visibility prior these ambiguities are solved, and a correct topological interpretation is given to the shape.

6.1 Noise Handling

Due to the blending characteristics in all stages of cone carving, the assumptions at the base of using visibility cones remain robust to a certain degree even when noise is present. However, unlike Poisson reconstruction, cone carving is not designed to smooth or filter out noise. In a sense, the effort to filter out noise is orthogonal to the problem of dealing with missing data, and any point-based noise filtering algorithm could be used prior to reconstruction using cone carving. Figure 10 presents some reconstruction results under noise and compares the behavior of cone carving based RBF with that of Poisson reconstruction. Note that although the output surface from the former is indeed noisy, the topology of the object is still correct (in contrast to Poisson) and remains correct even under more severe level of noise.

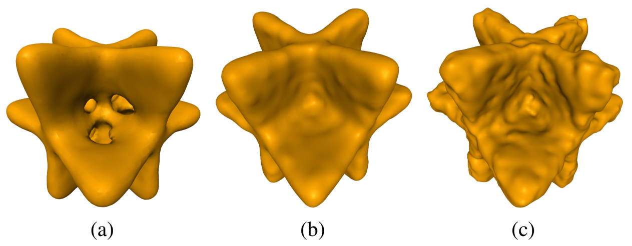

Figure 10: *(a) Poisson reconstruction of a point cloud with 10% random noise on 10% of the points is smooth but contains topological errors. (b) Cone carving for the same point cloud. (c) Result of cone carving on a point cloud with 10% random noise on all points still maintains the correct topology.*

6.2 Limitations

The main disadvantage of cone-carving is that the current naive implementation is quadratic, and therefore slow. To find the cone boundary for each point sample, splatting of all other points in the dataset is used. This means it is impractical for the larger data sets

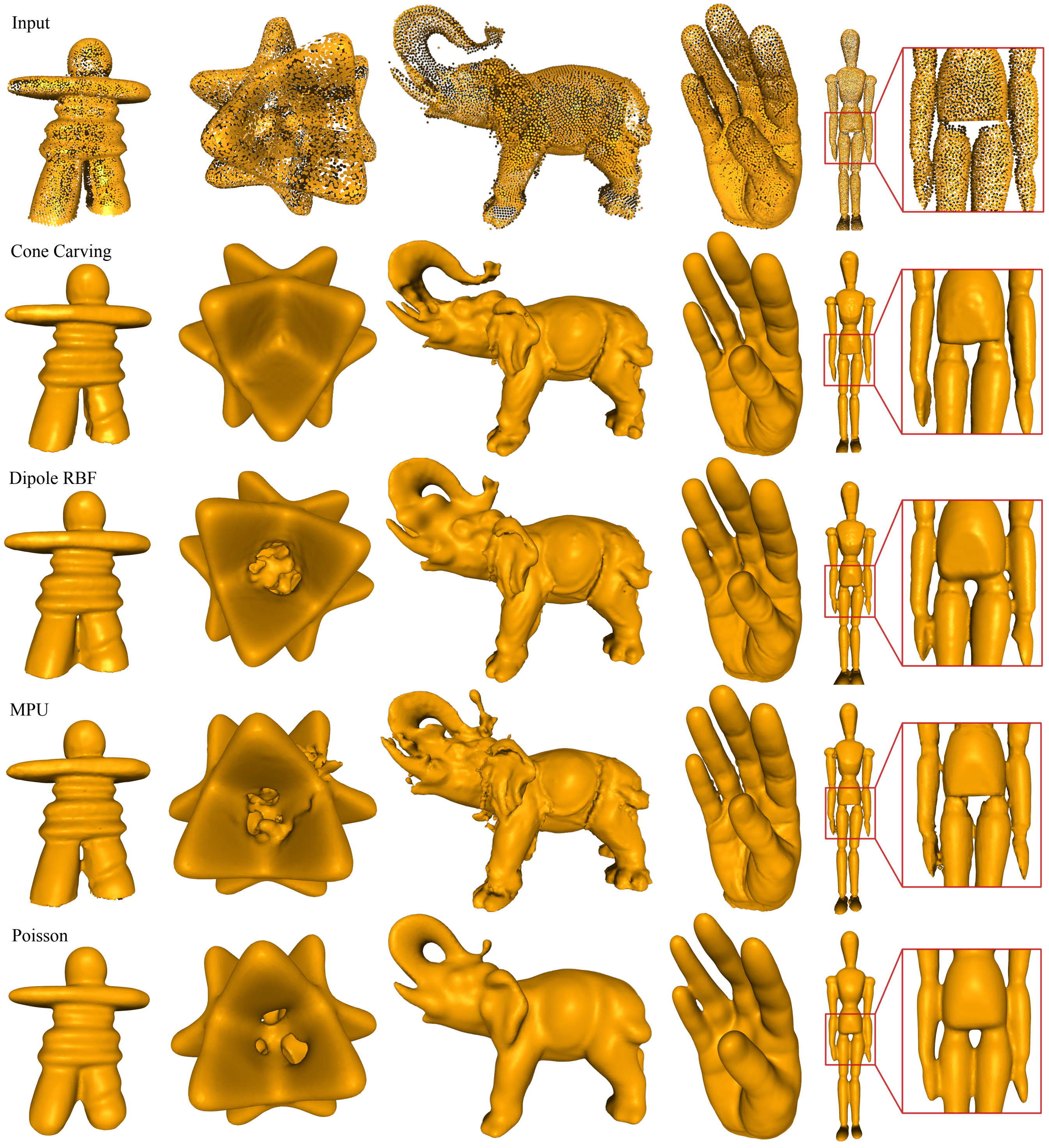

Figure 11: *Reconstruction results. The output of the cone carving algorithm is compared to reconstruction results of the dipole RBF reconstruction, Multi-level Partition of Unity, and Poisson reconstruction. Cone carving interprets the topology and approximates the geometry of the shape better than existing techniques.*

currently being produced. Advanced spatial search structures as well as random sampling on larger datasets can alleviate this problem. However, converting it to linear time (e.g. using fixed size neighborhoods) will still be difficult as the main point of cone carving is utilizing additional *global* information: the visibility of the scanned data. In general, quality is more important than speed in surface reconstruction, as it is part of a long process of data collection, which is rarely interactive.

The success of cone carving also relies on some tuning of parameters. There are three main stages that were defined by parameters: the cone creation stage, the cone distance blending stage and choosing between DTE and DP. In the cone creation stage we determine the splat size of each point based on N_{near}. This number defines the k-nearest neighbor search for pixels (Section 4). Figure 12 shows an example of the cube rendering for one point for varying values of N_{near}. It is clear that a small value for N_{near} creates holes in the surface, but a large value is time consuming and redundant creating large overlaps between splats. In practice, we found $N_{\text{near}} = 3$ strikes the best balance, correctly interpreting the shape for all points in all data-sets.

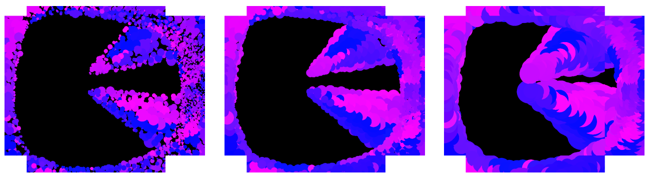

Figure 12: *Effects of varying the parameter N_{near} on point splatting: too small a value (1) creates holes, while too large a value (6) results in redundant overlapping splats. We use $N_{near} = 3$*

At the cone distance blending stage (Section 5), two parameters govern the blending: r, the blending kernel neighborhood size, and k the threshold defining the support for a significant cluster size. These can be seen as defining the size and cardinality of the cluster that defines DTE. Figure 13 illustrates the effect of changing these values on the reconstruction. For choosing between the DTE and DP distance measures we use a parameter λ. Figure 14 shows how changing λ can effect the reconstruction. In summary, there is a small set of parameters control cone carving results. However, we have shown that in all but one example in this paper (for the Mannequin figure we used $k = 2$ as it is more sparsely sampled) the same set of parameter values was used. These are governed by one characteristic value of the point set, namely, what we defined as d_{avg}.

7 Conclusion and future work

We have presented cone carving, enhancing surface reconstruction techniques from scanned point clouds by combining local positional and global visibility information. Cone carving utilizes the visibility of the existing points to infer new points that complete the missing parts of the captured model. Unlike previous attempts to exploit visibility to assist surface reconstruction, here the visibility is taken from an inside-out perspective, creating visibility cones from points on the surface outward. These cones are effective exactly where they are needed the most: away from existing points, where data is missing. We have demonstrated the effectiveness of cone carving in several reconstructions of both synthetic input and raw scans from a commercial 3D scanner.

With speed as the major limitation, we plan to investigate a few options to accelerate cone carving:

- Creating generalized cones only for a subset of points, but utilizing the entire point-set for tracing their silhouettes.
- Creating cones and off-surface points only near missing parts rather than across the whole shape and focus only on the parts where they make a positive difference.
- Using a pre-computed spatial data structures to accelerate various geometric computations such as the computation of DTC.

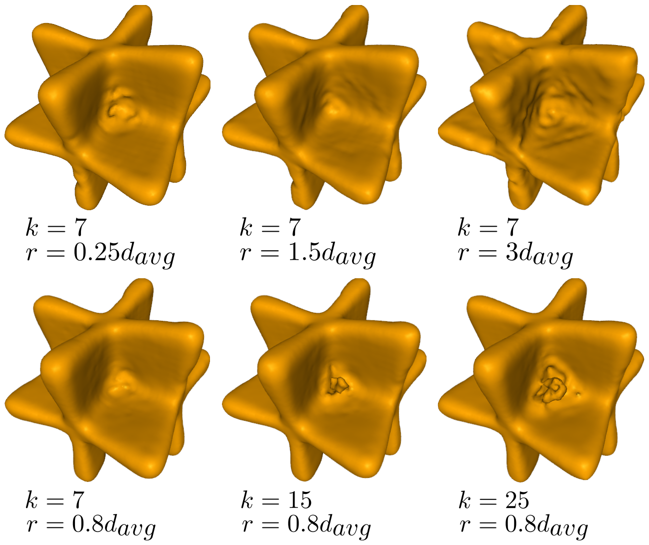
$k = 7$ $r = 0.25d_{avg}$
$k = 7$ $r = 1.5d_{avg}$
$k = 7$ $r = 3d_{avg}$
$k = 7$ $r = 0.8d_{avg}$
$k = 15$ $r = 0.8d_{avg}$
$k = 25$ $r = 0.8d_{avg}$

Figure 13: *The size and cardinality of the cluster defining* DTE *have opposite effects. Too large r can insert outliers to the cluster while too small r does not blend enough information and may determine a wrong distance. Large k can reject the correct distance cluster with too few counts, while small k will use outlier cones clusters. We have used $k = 5, r = 0.8 \cdot d_{avg}$ in all but one of the examples in the paper.*

- Fully exploiting the highly parallel nature of the algorithm using a GPU implementation.

Visibility cones may also prove to be useful for problems other than surface reconstruction. For instance, the shape and distribution of the cones can assist in line and point feature detection. Another interesting research direction would be to study the potential of the visibility cones to assist in planning the positioning of the scanning device in order to improve its coverage.

References

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Trans. Vis. & Comp. Graphics 9*, 1, 3–15.

AMENTA, N., AND BERN, M. W. 1998. Surface reconstruction by Voronoi filtering. In *Proc. of Symp. on Comp. Geom.*, 39–48.

AMENTA, N., CHOI, S., AND KOLLURI, R. 2001. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications 19*, 2-3, 127–153.

ARANZ. 2009. *FastSCAN, Cobra and Scorpion, Handheld Laser Scanner User Manual.* Aranz Scanning Ltd, http://www.fastscan3d.com/.

BAERENTZEN, J. A., AND AANAES, H. 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics 11*, 3, 243–253.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. of ACM SIGGRAPH*, 67–76.

CAZALS, F., AND GIESEN, J. 2006. Delaunay triangulation based surface reconstruction. In *Effective Computational Geometry for*

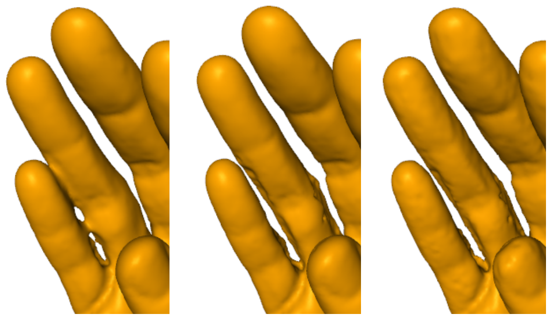

Figure 14: *Varying values of λ, for choosing between* DTE *and* DP *distances. When $\lambda = 0$,* DTE *is used whenever it is larger than* DP. *This creates correct topology but can result in some noise at well sampled parts, since in those areas and close to the surface* DP *distance is more reliable. When λ is large ($2 \cdot d_{avg}$), the* DTE *distance is not utilized, and wrong topology based on* DP *distances returns. The value used throughout the paper is $\lambda = d_{avg}$.*

Curves and Surfaces, J.-D. Boissonnat and M. Teillaud, Eds. Springer, 231–276.

COHEN, M. F., AND GREENBERG, D. P. 1985. The hemi-cube: a radiosity solution for complex environments. In *Proc. of ACM SIGGRAPH*, 31–40.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *Proc. of ACM SIGGRAPH*, 303–312.

DAVIS, J., MARSCHNER, S. R., GARR, M., , AND LEVOY, M. 2002. Filling holes in complex surfaces using volumetric diffusion. In *Proceedings of the first International Symposium on 3D Data Processing Visualization and Transmission*, 354–369.

DEY, T., AND GOSWAMI, S. 2003. Tight cocone: A water tight surface reconstructor. In *Proc. of ACM Sympos. on Solid Modeling & Appl.*, 127–134.

DEY, T. K., LI, K., RAMOS, E. A., AND WENGER, R. 2009. Isotopic reconstruction of surfaces with boundaries. *Computer Graphics Forum, special issue SGP '09: Proceedings of the Symposium on Geometry Processing 28*, 5, 1371–1382.

EDELSBRUNNER, H., AND MÜCKE, E. P. 1994. Three-dimensional alpha shapes. *ACM Trans. on Graphics 13*, 1, 43–72.

EL-SANA, J., AND VARSHNEY, A. 1998. Topology simplification for polygonal virtual environments. *IEEE Trans. Vis. & Comp. Graphics 4*, 2, 133–144.

EVERITT, B. S., LANDAU, S., AND LEESE, M. 2001. *Cluster Analysis*. Oxford University Press.

GAL, R., SHAMIR, A., HASSNER, T., PAULY, M., AND COHEN-OR, D. 2007. Surface reconstruction using local shape priors. In *Proc. Eurographics Symp. on Geometry Processing*, 253–262.

GIESEN, J., CAZALS, F., PAULY, M., AND ZOMORODIAN, A. 2006. The conformal alpha shape filtration. *The Visual Computer 22*, 8, 531–540.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *Proc. of ACM SIGGRAPH*, 71–78.

HUANG, H., LI, D., ZHANG, H., ASCHER, U., AND COHEN-OR, D. 2009. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 28*, 5, Article 176.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. In *Proc. Eurographics Symp. on Geometry Processing*, 61–70.

KUTULAKOS, K. N., AND SEITZ, S. M. 2000. A theory of shape by space carving. *Proc. Int. J. Comp. Vis. 38*, 3, 199–218. Marr Prize Special Issue.

LAURENTINI, A. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pat. Ana. & Mach. Int. 16*, 2, 150–162.

MONTENEGRO, A. A., CARVALHO, P. C. P., GATTASS, M., AND VELHO, L. C. P. R. 2004. Adaptive space carving. In *Proc. of 3D Data Processing, Visualization, and Transmission*, 199–206.

NEHAB, D., RUSINKIEWICZ, S., DAVIS, J., AND RAMAMOORTHI, R. 2005. Efficiently combining positions and normals for precise 3D geometry. *ACM Trans. on Graphics 24*, 3, 536–543.

NITSCHKE, C., NAKAZAWA, A., AND TAKEMURA, H. 2007. Real-time space carving using graphics hardware. *IEICE Trans. Inf. Syst. E90-D*, 8, 1175–1184.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Trans. on Graphics 22*, 3, 463–470.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., AND GUIBAS, L. J. 2005. Example-based 3D scan completion. In *Proc. Eurographics Symp. on Geometry Processing*, 23–32.

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. Surfels: surface elements as rendering primitives. In *Proc. of ACM SIGGRAPH*, 335–342.

SCHNABEL, R., DEGENER, P., AND KLEIN, R. 2009. Completion and reconstruction with primitive shapes. *Computer Graphics Forum (Proc. of Eurographics) 28*, 2, 503–512.

SHARF, A., ALEXA, M., AND COHEN-OR, D. 2004. Context-based surface completion. *ACM Trans. on Graphics 23*, 3, 878–887.

SORKINE, O., AND COHEN-OR, D. 2004. Least-squares meshes. In *Proc. IEEE Conf. on Shape Modeling and Applications*, 191–199.

TAGLIASACCHI, A., ZHANG, H., AND COHEN-OR, D. 2009. Curve skeleton extraction from incomplete point cloud. *ACM Trans. on Graphics 28*, 3, Article 71, 9 pages.

TURK, G., AND LEVOY, M. 1994. Zippered polygon meshes from range images. In *Proc. of ACM SIGGRAPH*, 311–318.

WESTOVER, L. 1990. Footprint evaluation for volume rendering. In *Proc. of ACM SIGGRAPH*, 367–376.