Adaptively Represented Complete Distance Fields

Jian Huang[1], Roger Crawfis[2],
[1]Computer Science, The University of Tennessee, Knoxville, TN
[2]Computer and Information Science, The Ohio State University, Columbus, OH

Abstract

Distance fields are an important volume representation. A high quality distance field facilitates accurate surface characterization and gradient estimation. However, due to Nyquist's Law, no existing volumetric methods based on the linear sampling theory can fully capture surface details, such as corners and edges, in 3D space. We propose a novel complete distance field representation (CDFR) that does not rely on Nyquist's sampling theory. To accomplish this, we construct a volume where each voxel has a complete description of all portions of surface that affect the local distance field. We also show here that CDFR can be adaptively represented, without comprising accuracy. The adaptively represented complete distance field is shorted for ARCDF. For any desired distance, we can extract a surface contour in true Euclidean distance, at any level of accuracy, from the same CDFR representation. Such point-based iso-distance contours have faithful per-point gradients and can be interactively visualized using splatting, providing per-point shaded image quality. We also demonstrate applying CDFR to a cutting edge design for manufacturing application involving high-complexity parts at un-precedented accuracy using only commonly available computational resources.

1. Introduction

A voxel-based volume, as a 3D raster, holds discrete sample points representing a certain multi-dimensional entity. In an alias-free volume discretization, only frequency components below half the Nyquist sampling rate would be stored. As a natural description of solid physical entities, volume representations have found applications in a variety of areas, including medicine, mechanical engineering, scientific computing and simulations. In order to utilize volume technologies, it has been common to convert surface models, such as a polygonal mesh exported by a CAD package, to a volume representation. In this process, first, one needs to voxelize the surface model into a hollow volume representing the surface shape [11][14][15]. Second, a distance transform is computed to construct a solid volume that encompasses a distance or thickness field recording distances to the surface. Euclidean distance has not been commonly used due to both efficiency concerns and the fact that accuracy is already compromised in the binary surface volume model. Instead, most applications use less accurate distance heuristics such as Manhattan or chessboard distance, or a Chamfer distance [3].

Voxelization techniques that convert surface shapes into binary volumes, with 1's representing occupancy and 0's representing empty space, have been developed in [11][14][15]. These methods are practical and commonly used nowadays. Surface shapes, however, are infinitely thin in space. To sample this thin shape, one needs infinitely high sampling rates. To avoid such a difficulty, Kaufman's algorithm [14][15] increases the voxel thickness of a surface, and therefore, to some extent, band limits the frequency spectrum before the sampling, or scan-conversion, process. Huang et. al [11] proved the sufficient and necessary thickness of the surface shape that guarantees a correct discrete topology in the resulting volume representation. Unfortunately, all these methods are based on binary volume representations, which are highly susceptible to aliasing artifacts. To address this issue, Sramek and Kaufman initiated data representations in non-binary formats [22]. In their paper, they show one has to use higher order smoothing functions to pre-filter and band limit the spectrum of the volume. Later, an incremental voxelization method for non-binary volume is reported in [5].

Over the years, in addition to the search for optimal voxelization, the community has also been exploring other representations of surfaces, such as distance fields. Distance fields are scalar fields, with each element in the 3D volume representing the minimal distance to a certain shape. It is common practice to use signed values to distinguish between interior and exterior of the shape. Compared to the surface shapes that correspond to impulses in 3D space, distance fields are much smoother. For shapes without sharp corners and edges, both the surface position and gradient can be reconstructed relatively accurately using a distance field [1][9]. When corners and sharp edges are introduced, high frequency components are also brought into the spectrum. To preserve such details, super-sampling with exceptionally high volume resolution, as well as low-pass filtering, is necessary to achieve an alias-free representation.

All traditional distance field algorithms are based on the Nyquist sampling theory. In other words, distance fields can only be sampled and then stored into discrete volumes. Unfortunately, for CAD applications, without a practically usable definition of accuracy loss after band-limiting filtering, all previous discrete distance field algorithms could in no way be reliably used. The alias-free definition to the graphics community does not address the accuracy issues for CAD applications. This limitation in error-tolerance, as well as the overwhelming costs of high-resolution volume applications, has limited applications of volume techniques in CAD/CAM.

In [13], a novel method to accurately represent distance fields in a volumetric form was proposed. The new method of volumetric representation of distance fields is based on a complete distance definition, which is disparate from the theory of linear sampling. The new distance field representation was named a "complete distance field representation (CDFR)", because once the distance volume is constructed, one can extract any distance contour to any error-tolerance directly from the distance field. As a comparison, conventional approaches based on a single valued distance field can only achieve higher accuracy by re-building the whole distance volume at an increased resolution. However, in most cases building a high resolution distance volume is non-trivial both in computational time and storage space. Another recent work [16] stores the estimated local edge positions in *x*, *y* and *z* directions with each voxel and can extract triangle meshes from volume data at a much improved quality compared to conventional methods [2][10]. However, their distance field representation is not complete and the surface extraction is still based on estimation. The method of CDFR only relies on exact computations, and, different from the approach in [2][10][16], uses a point-based approach to represent the extracted contour surfaces.

Hierarchical data structures representing distance fields efficiently have been reported in [7], where adaptively sampled distance fields (ADF) were introduced. ADFs help in reducing volume storage size when fewer details are locally present. The specific ADF implementation described in [7] relies on a discretely sampled distance representation, therefore that implementation still depends on a band-limited spectrum that discards all details beyond the cut-off bandwidth supported by the leaf level in the tree structure. Although CDFR does not abide by the sampling theory, the adaptive structures of ADF can be utilized by CDFR as well. As shown here, however, the Adaptively Represented Complete Distance Fields (ARCDF) only helps to speed up the iso-contour extraction phase, but not more efficient in storage.

2. Introduction to Distance Fields

Traditionally, distance fields are defined as spatial fields of scalar distances to a surface geometry or shape. Each element in a distance field specifies its minimum distance to the shape. As long as the shape is represented by an oriented manifold, positive and negative distances can be used to distinguish outside and inside of the shape, for instance, using negative values on the outside and positive on the inside. Distance fields have a number of applications in constructive solid geometry [1][7], surface reconstruction and normal estimation [9] and morphing [1][3]. Distance fields are also applied to concurrent engineering [17] where simulations and analysis involving the interior of geometries, such as die-casting simulation or thickness analysis of parts [23], are routine.

For an alias-free sampling of a signal, Nyquist's Law dictates that the sampling rate must be at least two times the highest frequency component in the signal. In spatial domain, geometry is infinitestismally thin, and has an infinitely wide frequency spectrum. The sharp details on the surface, such as corners and edges, also reside on the high ends in the spectrum. Even with an overwhelmingly large volume resolution, one still needs extensive low-pass filtering to limit the bandwidth of the geometric shape. These low-pass filtering operations, with either simple box filters [11][14][15] or specifically designed higher order filters [22], inevitably cause a loss of the exact surface details. Converting the surface shape to a distance field, which is smoother, provides a way to exactly locate the surface [9] during reconstruction. But the underlying assumption of having a completely smooth surface that is free from sharp corners and edges is unrealistic for most scenarios.

Frisken et. al [7] developed a well analyzed framework for adaptively sampled distance fields (ADF), by which one can build hierarchies of distance fields at different levels of detail and be able to cross over different levels of detail as needed. They also vary sampling rates according to the amount of details that are available locally. They used tri-linear interpolation to reconstruct distances, and were able to demonstrate a suite of applications with impressive rendering quality. However, ADF [7] does not fundamentally solve the problem of losing surface details in discrete representations. After the leaf level of ADF is constructed, the loss in surface details is final and irreversible. When the primary goal of an application shifts from visual quality to accuracy, ADFs with trilinear interpolation may not satisfy the accuracy needs with a guarantee, simply because the true distance fields are not linear where corners or edges are present. What the hierarchies provide is an ability to save computational and storage resources when less details are encountered. For models with fine details everywhere, the ADF eventually resorts to an extremely large voxelization. For the applications where accuracy is highly sought after, current ADFs based on single valued distances incur overwhelming costs, because most practical geometrical models are rich in details at a wide range of scale.

A high quality distance field should be accurate, efficiently stored and can be efficiently processed. There are two fundamental issues involved in building a high quality distance field. First, we need an accurate way to represent the distance from an arbitrary point in 3D space to an arbitrary shape. Second, how should we optimally organize the distance representations in space? The first question can be answered by CDFR as a fundamental fix that preserves all geometric details in the true distance field. The answer to the second question lies in applying the hierarchical concept presented in ADF[7] to efficiently organize the spatial data structure of CDFR. Depending on the amount of surface details available, one needs an adaptive and smooth transition between resolution levels for efficient querying.

3. Complete Distance Definition (CDD) & Complete Distance Field Representation (CDFR)

We now introduce a complete distance definition (CDD). Corresponding to different surface representations, such as parametric surfaces, implicit surfaces or subdivision/polygonal mesh surfaces, there could be different instantiations of CDDs. Here, we focus on a simple case where the surface is represented by polygonal meshes. When accuracy is paramount, Euclidean distances are preferred over other distance metrics, such as Chamfer distance or Manhattan distance. The CDD distances are true Euclidean distances. Before discussing CDD, we will discuss a few observations that motivated the CDFR research.

3.1 Some Observations

Distance fields are very smooth in some simple scenarios. For instance, suppose in a 1-dimensional space, we have an impulse. It's frequency components extend to infinity. There is no way to use a finite sampling frequency to sample the impulse without aliasing. But on the other hand, as illustrated in Fig. 1, the signed distance field of the impulse is a linear function which extends from negative infinity to positive infinity. Sampling this linear function can be accurate with a relatively low sampling rate.

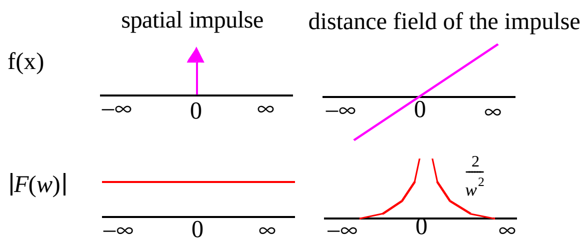

Figure 1: Without low-pass filtering, it's impossible to sample the impulse (left), but we can sample its distance field (right).

Unfortunately, this feature does not hold in higher dimensions where corners are present. As presented in [11], when extended into 2D or 3D, the discrepancies and discontinuity on corners makes the distance field non-smooth. For instance, in the triangle in Fig. 2, we have a rather faithful sampling in the light grey grids on the edges, because the geometry is locally linear*. But the sampling is not sufficient in the dark grey grids that have corners. The non-linear distance fields within the dark grey grids make it impossible to accurately recover the correct distance distribution from the grid samples.

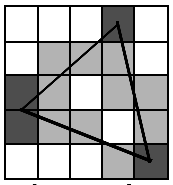

Figure 2: Corners in the triangle cause complexity in the distance field, resulting in an aliased spectrum after sampled.

According to Nyquist's Law, to sample such complicated distance fields, one must low-pass filter the corners and smooth out the sharpness. Using ADF [7], for the grids on the corners, a higher resolution would be used, whereas in the light grey grids, a much lower resolution might suffice.

To capture the exact location of the impulse in Fig. 1, we do not have to use sampling. Alternatively, all one needs is to place an anchor point at some location, and record the signed distance from the anchor to the impulse. In this method, preserving the exact position of the impulse is made straight-forward. This observation motivated our research towards a new distance representation for distance fields.

3.2 Complete Distance Definition (CDD)

CDD is a set of parameters describing both the distance from a 3D point to a surface geometry primitive and the geometry primitive itself. Specifically, when the shape is represented as a mesh of triangles, CDD reduces to a tuple that consists of a scalar canonical distance value, and a description of the triangle with a vertex list and an edge list:

$$\langle \text{distance}, \langle v_1, v_2, v_3 \rangle, \langle e_1, e_2, e_3 \rangle \rangle \tag{1}$$

The value *distance* is the true Euclidean distance from the voxel center to a finite triangle. This distance is defined in the pseudo code in Fig. 3.

While the return value is the CDD distance, the input parameters include a triangle, *tri*, and a 3D point, *pnt*. If *pnt* orthogonally projects into *tri* (case C_1), the return value is the orthogonal distance from *pnt* to the plane where *tri* lies. Otherwise, we check whether *pnt* orthogonally projects onto any of the three edges. If yes (case C_2), then the returned distance value is the shortest distance from *pnt* to an edge that *pnt* projects orthogonally onto. In case neither C_1 or C_2 applies (case C_3), the distance is the minimal distance from *pnt* to the three vertices. This definition of distance to a finite triangle is further illustrated in Fig. 4.

```
float CDD (triangle tri, vec3 pnt)
{
    float mindist = MAXIMUM;
    if (pnt projects orthogonally into tri's interior)          // C_1
          mindist = distance from pnt to tri's plane;
    else
    {
        for each edge of tri, e_i,
            if (pnt projects orthogonally onto e_i)             // C_2
               mindist = min(mindist, distance from pnt to e_i);
        for each vertex of tri, v_i,                            // C3
```

Figure 3: Definition of distance from a point to a finite triangle

*Locally linear: reconstruction is accurate with linear interpolations.

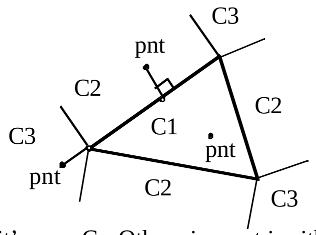

Figure 4: If *pnt* projects into *tri*, it's case C_1. Otherwise, *pnt* is either C_2 or C_3, depending on whether it's closer to an edge or a vertex. This diagram is drawn in 2D for ease of illustration.

We can still use positive and negative distance to distinguish inside and outside. We term the triangle that is the closest to *pnt* as the *base triangle* of *pnt*. If *pnt* is closest to a triangle and the distance is of case C_1, then this triangle is *pnt*'s base triangle. If *pnt*'s distance is not case C_1, rather, it's case C_2 or C_3, looking for *pnt*'s *base triangle* is more complicated. For C_2 cases, let's label the projection point of *pnt* on that corresponding edge as, *p'*, and we record the vector pointing from *p'* to *pnt* as *V*. Between the two triangles sharing that edge, the triangle with a normal direction closer to *V*'s direction, i.e. larger absolute dot product value, $|V \bullet Normal|$, is *pnt*'s *base triangle*. Very similarly, in C_3 cases, among the triangles sharing that closest vertex, we can easily find out the *base triangle* of *pnt* by comparing dot product values. We are interested in finding out *pnt*'s *base triangle*, because by using the outward normal direction of the *base triangle* and the relative position of *pnt*, we can determine the sign of the distance at *pnt* without ambiguity.

To better illustrate the process in determining the distance sign, in Fig. 5a, we show several 3D examples, shown in 2D. p_1 through p_6 are 2D points. t_1 through t_6 are triangles that form the surface mesh. $p_{2,\ 3,\ 4\ and\ 5}$ are all case C_1. From the normal direction of t_2, we can tell p_2 is outside, p_3 is inside. Similarly, using the normal of t_3 and t_5, one can tell that p_4 is inside, and p_5 is outside, respectively. p_1 and p_6 are both C_2. We show an enlarged view of these two cases in Fig. 5b. By comparing the dot products, we can tell p_1's sign is determined by t_2, and for p_6, it is decided by t_1.

Finally, to save space, we store the description of all triangles in a separate array and only keep a triangle index in a CDD tuple.

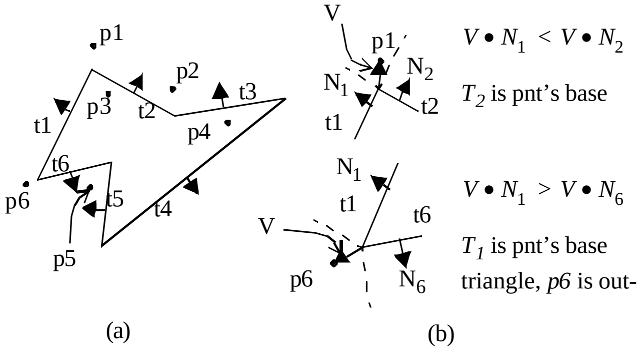

Figure 5: (a) 2D illustrations of the process to determine the sign of the distance of a point. The solid black arrows depict the outward normal direction of each triangle. Points p_2 through p_5 project into the triangles, i.e. case C_1. The signs of the distances of p_2 through p_5 are determined by evaluating the normal direction of each point's base triangle. p_1 and p_6 are examples of C_2 cases. (b) Enlarged view of p_1 and p_6. Both p_1 and p_6 are outside.

3.3 A Complete Distance Field Representation (CDFR)

In this section, we show the process that uses CDD to build a complete distance field representation (CDFR), allowing exact capture of all geometric details, e.g. sharp corners and edges, to any level of accuracy.

Given a surface mesh, in the voxelization step, we store CDD tuples with each surface voxel, rather than single valued distances.

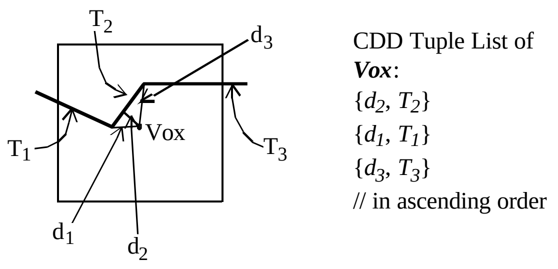

Figure 6: A 2D illustration of building a CDD tuple list for a surface voxel, *Vox*. There are 3 triangles intersecting *Vox*. The CDD tuple list is organized in ascending distance order, with the minimal distance of *Vox* being d_2.

For each triangle touching a surface voxel, a CDD tuple is stored with that voxel. The end result of the voxelization step leaves all surface voxels with a list of CDD tuples, sorted in ascending order by distance values. Fig. 6 provides an example of voxelizing a single surface voxel, *Vox*. There are three triangles touching *Vox*. T_2 is case C_1, with T_1 and T_3 being case C_2 or C_3. The minimal distance of *Vox*, measured from the center of *Vox* is d_2. As a result, *Vox* has a sorted list of 3 CDD tuples.

At the end of voxelization, we have a volume where each voxel which the surface intersects contains a list of polygons cutting through it.

3.4 Distance Transform

For a distance transform, initially, we use an outside flooding algorithm to eliminate all outside voxels from our computation. For the remaining voxels, a contour by contour CDD propagation is performed from the surface voxels to the interior. During this process, a voxel looks for CDD tuples that have been newly propagated to anyone of its 26-neighbors [11]. It inherits all new CDD tuples from its neighbors, and for each triangle, it computes the true Euclidean distance from its own position. An updated list of CDD tuples are then sorted into ascending order and the first CDD tuple in the list contains the current distance, *cur_distance*, of this voxel. All the CDD tuples that contain a distance value within the range:

$$[cur\_distance, cur\_distance + \sqrt{3} * voxel\ size]$$

are stored with that voxel. This is a sufficient range to guarantee correctness in the distance transform, as we will prove in Section4. The CDD tuples out of this range are discarded. This process of distance transform iterates until no voxels find new CDD tuples from its 26-neighbors affecting its current CDD tuples list.

3.5 Extracting A Distance Contour

The most frequent way in which a distance field is used is by reconstructing or extracting an iso-distance contour. For instance, a user asks the following request, “show me the zero distance contour with an error tolerance of 0.5mm”. The conventional way of reconstructing sub-voxel distance is to trilinearly interpolate in-between voxels [7]. Often times this reconstruction step is embedded in ray-casting procedures at rendering time. While this works for some applications, there is no guarantee on the level of accuracy. From CDFR, we extract a distance contour with a fulfillment of an arbitrarily high accuracy requirement. We store the extracted distance contours as point-based models [6][21], so that we can render the contours at high interactive rates with splatting [4][12][19].

The extracting procedure works as following. Given a requested interior thickness, t $(t>0)$, we traverse these voxels with a distance value in the range:

$$\left(t - \frac{\sqrt{3}}{2} voxelsize, t + \frac{\sqrt{3}}{2} voxelsize\right) \tag{2}$$

The requested iso-contour will pass through the span of these voxels. Unlike marching cubes [10], We do not use conditions like $\text{minimal thickness} \le t$ and $\text{maximal thickness} \ge t$, because the underlying assumption of having a linear function is not true in our case. There could be cases where the 8 corner voxels are just surrounding the maximal thickness point in the model, and none of the 8 voxels exactly captures that maximum.

After identifying the relevant voxels, we then subdivide the voxels into sub-voxels, or points [2]. We only extract the sub-voxels that are close to the desired surface into a point-based iso-distance contour represented the surface. In order to support the error tolerance, *E*, picked by users, the size of each sub-voxel must be:

$$\frac{\sqrt{3}}{2} subvoxelsize < E \tag{3}$$

For each sub-voxel, or point, we compute the signed true distance for all the CDD tuples resident on each of the 8 cornering voxels. The points that have the minimal positive distance value within the range [*t* - *E/2*, *t* + *E/2*] are extracted into the point-based iso-distance contour.

3.6 High Quality Gradients

Besides using the distance contour for analysis, visualizations of the distance contours are also highly desired in applications. For point-based models, having high quality normal information on each point is essential for high image quality.

Our CDFR offers an additional advantage in this perspective. When extracting the distance contour from the base triangle of each sub-voxel, the normal of this point is computed. If this point is of case C_1 to its base triangle, then the normal of the base triangle is this point's true gradient. If the point is one of the cases C_2 or C_3, the gradient is the vector *V* in Fig. 5. For instance, in a C_2 case, the 3D point, P, first gets projected onto the closest edge. The gradient is the vector connecting P and its projection. In C_3 cases, the gradient direction is obtained by connecting P and the closest vertex. Therefore, the normal vectors computed for the whole point-based model is continuous and accurate. High quality per point shading is thus supported.

4. Proof of Sufficiency

To prove the correctness of CDFR, we need a proof of sufficiency. That is, when we need to reconstruct the local distance field in the span of any voxel, all the surface primitives affecting this local area are present on that voxel.

A surface primitive, such as a triangle, affects a local field in 3D space by being the closest surface triangle to at least one position in this local area. Based upon this observation, we devise our proof of sufficiency with a proof by contradiction:

Suppose in the CDFR, *R*, there exists a local voxel, *V*, in whose span there exists at least one point, *P(x,y,z)*, whose base triangle, *T*, is not resident on the voxel, *V*.

Without loss of generality, we write the distance from *P* to *T* as *D*. All distance fields are continuous functions, although they may not have continuous derivatives. For a point, *P'(x+dx,y+dy,z+dz)*, that is closely neighboring *P*, the minimal distance from *P'* to *T* is bounded by:

$$[D - \sqrt{dx^2 + dy^2 + dz^2}, D + \sqrt{dx^2 + dy^2 + dz^2}] \tag{4}$$

Due to deduction, when *P'* incrementally moves from *P* towards *V'*s center point, it logically follows that the distance from *V*'s center point to *T* is bounded by:

$$[D - \int_p^v ds, D + \int_p^v ds] \tag{5}$$

Equation (5) can be rewritten as:

$$[D - dist(V, P), D + dist(V, P)] \tag{6}$$

However, the minimum distance to *P*, which is *D*, must also be smaller than $minD + dist(V, P)$, with *minD* denoting the minimum distance of the surface to *V*. Therefore, the distance of *T* to *V*, must be within the following range:

$$[minD, minD + 2dist(V, P)] \tag{7}$$

Since P is in the span of V, the maximum possible distance is $(\sqrt{3})/2$ *voxel size*, the range in (7) is actually a subset of:

$$[minD, minD + \sqrt{3} voxelsize] \tag{8}$$

Contradiction. Since during our distance propagation process, Equation (8) is exactly the range that we maintain on each voxel. Hence, triangle T must be resident on voxel V. The assumed case can not exist. Proof completed.

We do not claim our storage is minimal. We might have kept more CDD tuples on each voxel than necessary. However, enforcing that minimality would introduce more complexity. As long as we use a triangle index in CDD tuples instead of complete description of each triangle, the extra storage cost that we spend is low. We have traded for simplicity in implementation.

5. Adaptively Represented Complete Distance Fields

CDFR stores on each voxel the indices of all surface triangle that affect the distance field in the local neighborhood around that voxel. We term these triangles as resident triangles. The accuracy of CDFR is guaranteed no matter what the volume resolution is. However, the resolution of the volumetric grid does affect the efficiency of computation when a distance iso-contour needs to be extracted. In areas of high complexity, it helps to use a higher resolution for efficiency. While in computational geometry and CAD fields, exactly defining what geometric complexity is a thorny issue; for the purpose of adaptively representing CDFR, complexity can be straightforwardly defined as the number of triangles resident on each voxel. We should also note here that CDFR is defined on a uniform grid only. Therefore, an octree type of subdivision is inevitable. We construct the octree in a top-to-bottom fashion for all voxels that need to be subdivided.

The average number of triangles on each voxel across the whole volume is quite low, usually in the range of 2 to 7 or 8 triangles per voxel, for models with a modest complexity. However, in most data sets, about 1 to 3% of the non-empty voxels have as many as 15 triangles or more. For some parts with more than 30 thousand triangles, the maximal number of triangles resident on a voxel can be as many as 100, at a 128 CDFR resolution. If these voxels with a large number of voxels are incurred when an iso-contour is extracted, a large amount of computation is inevitable. We subdivide these voxels. The process is quite straightforward. For each voxel that needs to be subdivided, we partition it into 8 octree subdivided children. All triangles that affect the local neighborhood are guaranteed to be resident on the parent voxel. We just compute the distance from each sub-voxel center to all the triangles, respectively. For each sub-voxel, we then sort the triangles in ascending distance values, and discard all triangles with distances larger than:

$$minimal\_distance + \sqrt{3} * sub\text{-}voxel\ size \tag{9}$$

This subdivision process can be recursively iterated until a certain stopping criteria is met. Our current criteria for subdivision is met when a voxel has more than k times the average number of resident triangles across the volume. The value k can be any integer above two, if one doesn't want to subdivide the whole volume. In that case, it's easier to use a higher initial resolution.

One interesting result we obtained actually shows that subdivision does not reduce number of resident triangles very significantly on all subdivided voxels. In other words, the number of resident triangles does not scale down. In areas where a large number of triangles affect the local distance field, no matter how small that region is, the number of resident triangles does not reduce in a large amount as a result of subdivision, although such reduction is always distinguishable between neighboring levels in an octree. The lower limit seems to be around 2 to 5 resident triangles per voxel, depending on the complexity of each model. Our experimental results confirmed this hypothesis repeatedly and consistently (Section 6.4). Also as expected, adaptively using a smaller voxel size in area of high complexity does help to accelerate the time to extract final distance contour.

6. Results and Analysis

The resolution of CDFR volumes do not affect the accuracy of the distance field. Also, the CDFR construction step is independent from the step that reconstructs iso-distance contours. Before we analyze the performance of our approach, we show images of distance contours on a few sample parts to demonstrate the accurate Euclidean distance fields obtained. All point-based models are rendered with splatting [12]. All results are collected on a SGI Octane with a 300MHz processor and 512MB memory. Table 1 provides a full description of the models used for test and analysis of our algorithm. We have also applied our algorithm to a very complicated part, Engine Cylinder Head, for heavy section detection. The details of the Engine Cylinder Head model is described in Section6.6.

Model	No. Triangles	Bounding Box Size (x,y,z) (inch)	Maximal Thick-ness(inch)
Cube	12	(5, 5, 5)	2.5
Tetrahedron	4	(1, 1, 1)	0.2
1-Tooth	16	(1, 2, 2)	0.48
6-Star	48	(1, 3, 3.46)	0.49
Connector	242	(6.9, 2.0, 2.9)	0.50
Brevi	1812	(38.1, 34.9, 96.0)	13.00

TABLE 1. Physical Information of Test Models.

6.1 Proof-of-Concept Experiments

As a proof of concept, we first examine some simple cases. Cubes and tetrahedral cells are the simplest. They are convex and symmetric. With a true Euclidean distance field, the thickness contours of different values are in the exact shape of the outer surfaces, including the sharp edges and corners, as shown in Fig. 7(a)(b). Concavities cause additional complexity in distance fields. Two concave examples, a one-ended tooth and a six-pointed star are shown in Fig. 7(c)(d). For the one-ended tooth, we choose a small thickness value to extract a contour close to the surface, while for the six-pointed star, a larger thickness is chosen. The evolving effects in Euclidean distance fields are interesting, with corners being smoothed out on the interior distance contours in both Fig. 7a and Fig. 7b. Small thickness contours closer to the surfaces retain more detail of the surface shape. The shape of the deeper contours manifest more global features of the shape (Fig. 7c,d).

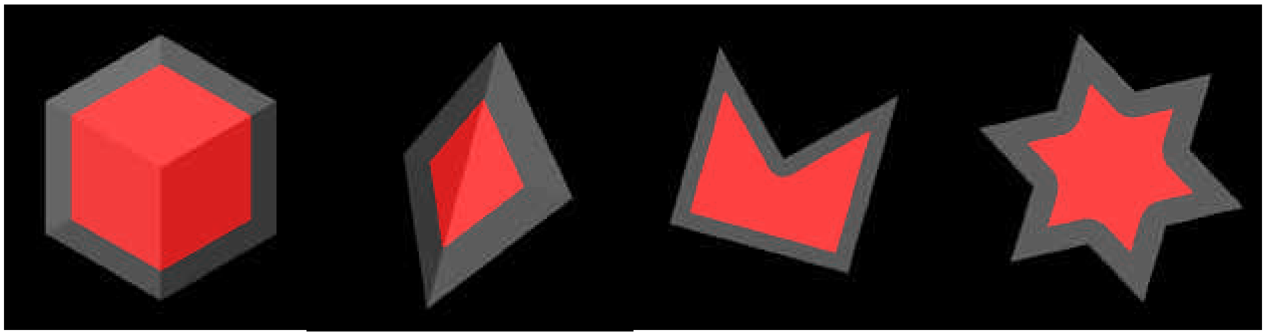

Figure 7: A cube and tetrahedron, with the surface mesh shown in semi-transparency. The distance contours (shown in red, per-point shaded) are of thickness (a) 0.6 inch and (b) 0.1 inch. Two concave examples, a 6-pointed star and a one-ended tooth. (c) For small thickness values (0.2 inch), the distance field retains most corners and edges, with little smoothing. (d) As the thickness increases (0.35 inch), the distance field evolves into the model, showing more smoothing.

All four models have a CDFR at a low resolution of $32 \times 32 \times 32$. To reconstruct the thickness contours in Fig. 7 and 7, we set the accuracy to 1/500 of the longest dimension of the model. For all 4 models, there are less than 400K points in the point-based contour. We obtained 3 frames/second rendering rates of the accurate distance contours with per point shading.

6.2 Real-World Models

Surface graphics based CAD/CAM systems often spend hours to perform jobs involving the interior of real world designs. Volume techniques have been considered. But due the overwhelming costs incurred in high resolution volumes, the affordable accuracy is very limited with current computing systems when applied to design for manufacturing purposes [17]. The application of volume graphics to CAD is limited. We tested our approach on two industry production models, ‘connector’ and ‘brevi’, for which both accuracy and interactive frame rates are highly sought after.

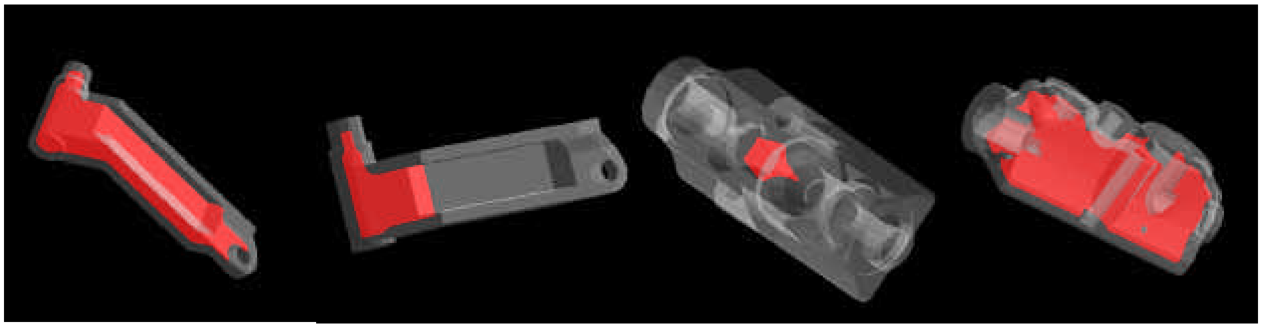

Figure 8: (1)Results of ‘connector’. The surface mesh is shown in semi-transparency, and the per-point shaded distance contours are at thicknesses (a) 0.2 inch and (b) 0.35 inch. (2) Sample images of ‘brevi’. With the per-point shaded contours at thicknesses (a) 10 inches and (b) 4 inches.

All contours in Fig. 8 are extracted to an accuracy of 1/1024 of the length of each part. In Fig. 8(a,b), the thickness contours within the ‘connector’ part show crisp edges and corners, while at the same time, retaining topological features in the surface geometry at different levels of scale. In Fig. 8(c,d), we present the thickness contours within the ‘brevi’ part at thicknesses of 10 and 4 inches. Fig. 8a demonstrates that even at the core of a part, sharp corners in the distance fields still exist at a variety of scales. The hole on the lower right corner of Fig. 8d is a discontinuous point in the distance field, and is one of the causes of failures during previous manufacturing processes. (See color plate for greater details). The contours in Fig. 8(b,c) are small, with 400K and 120K points, respectively. Around 2 to 5 frames/second have been recorded. However, Fig. 8(a,d) both have relatively large contours, with 1.5 to 2 million points, and only about 0.5 frames/second rates are obtained.

6.3 CDFR Size and Construction Time

The detailed storage structure on each voxel is shown in Fig. 9. Each voxel contains a 1-byte flag, *vCnt*. Empty voxels, i.e. outside voxels, have *vCnt* set to zero. Surface voxels have a *vCnt* value in the range between 1 and 127, denoting the number of triangles present on this voxel. Voxels entirely in the interior are distinguished by having a *vCnt* larger than 127. The value (vCnt -127) is the count of triangles present. For very complicated models at very low CDFR resolution, *vCnt* may overflow. In that case, a larger number is needed for *vCnt*, or an adaptive subdivision scheme is required. By subdividing each voxel into an even number of sub-voxels, we also break up areas having a central curvature point, such as a sphere. In total, $(4 \cdot n + 5)$ bytes are needed per non-empty voxel, with *n* being the number of triangles present on that voxel.

```
CDD_voxel
{
   unsigned char vCnt; // in/suf/out, and counter of triangles
   float cur_distance;   // current minimal distance on this voxel
   int triangles[triangle count]; // dynamic array of triangle indice
}
```
Figure 9: The storage on each voxel in a constructed CDFR.

To analyze storage costs of a CDFR, we constructed CDFR volumes for each model at different resolutions. Cube, Tetrahedron, 1-Tooth (one-ended tooth) and 6-Star (six-pointed star) are simple models, for which we built

CDFR volumes at 32 and 64 resolutions. At 32 initial resolution, the average number of triangles resident on each voxel ranges between 2 to 4. When constructed at 64 resolution, the average number of triangles per voxel drop to 1.5 to 3. For such simple models, there is no need to adaptively store CDFR. For the two industry parts, 'connector' and 'brevi', we use higher resolutions, since there are more surface details. We chose 128 and 256 resolutions. A 128 resolution seems high enough to limit the average number of triangles on each voxel for 'connector'. However, for 'brevi', a resolution of 256 is necessary to cut down the (number of triangles)/voxel. In Table 2, the 'Resolution' column shows the actual dimension of the CDFR volume. The construction time and final sizes of the CDFRs are shown under 'Timing' and 'Size'. 'In/Sur/Out' indicates the distribution of interior, surface and outside voxels in the CDFR. Finally, the average number of triangles on surface and interior voxels are presented in the last two columns.

For both simple models and industrial parts, storage size, as well as construction time, of CDFR increase by a factor ranging from 8 to 10 times, as the volume resolution is doubled each time.

Model	Resolution	Timing (sec)	Size (KB)	In/Sur/Out (K voxels)	Avg Tri/ Sur	Avg Tri/ In
Connector	128,43,58	8.05	970	26.4/20.8/272.0	1.95	2.91
Connector	256,81,112	82.72	7,548	30.7/91.5/1,924.0	1.43	2.53
Brevi	56,52,128	51.53	3,459	100.4/50.5/221.7	3.01	4.66
Brevi	106,98,256	448.2	25,260	1,075.4/205.2/1,379	1.96	3.69

TABLE 2. CDFR facts for 'connector' and 'brevi'.

6.4 ARCDF Subdivision Results

For complicated industrial parts, such as connector and brevi, we select the threshold to be 3 times overall average number of triangles (column 3, Table 3) across the volume. Those voxels make up a very low percentage in the total volume (column 4, Table 3). Before subdivision, those voxels having more resident triangles above the threshold, of course, have a relatively high number of average resident triangles (column 5, Table 3). We subdivide those voxels above the threshold to two different levels, 2 by 2 by 2 subdivision (column 6, Table 3) and 4 by 4 by 4 subdivision (column 7, Table 3). As we have discussed in Section 5, the average number of resident triangles does not decrease significantly. Until we use a high subdivision factor of 4 by 4 by 4, the average triangle count only reduces to about half of the number before subdivision.

Part Name	Initial Res	Overall Avg Tri Cnt	Percentage over threshold	Avg Tri Cnt above threshold	Avg Tri Cnt above threshold (2-sub)	Avg Tri Cnt above threshold (4-sub)
Connec-tor	128	2.55	1.28%	9.57	7.12	4.17
	256	2.34	0.30%	9.60	7.01	4.73
Brevi	128	4.28	1.48%	15.21	11.93	4.92
	256	3.40	0.58%	10.20	10.23	4.52

TABLE 3. CDFR Triangle Count after Subdivision

6.5 Point-based Iso-Distance Contour Extraction Time

Using a higher resolution CDFR has no effect on the accuracy in the final representation. However, it causes an dramatic cubic increase in storage size and construction time. The main motivation in using higher CDFR resolutions is to have more efficient distance contour extraction, due to less CDD tuples in each voxel and more accurate localization of voxel spans that may contain the requested iso-contour. We tested all of the models on the time to extract iso-distance point-based contours from different CDFR resolutions. In Table 4, we show these timings, in seconds, to extract a contour from both 32-res and 64-res CDFRs of the four simpler models. The 'Thickness' column shows the iso-distance value chosen. For each CDFR resolution, we collect timings for 3 levels of accuracy and organize the results in regard to which conventional volume resolution the extracted contours would correspond to in accuracy.

We list the three corresponding conventional volume resolutions, 128, 192, 384, under both ‘32 Res CDFR’ and ‘64 Res CDFR’.

Model	Thickness (inch)	Timing (32-Res CDFR) (sec)			Timing (64-Res CDFR) (sec)		
		128 res	192 res	384 res	128 res	192 res	384 res
Cube	0.6	1.13	3.96	29.66	0.84	1.95	14.48
Tetra	0.1	0.22	0.55	4.02	0.15	0.29	1.61
1-Tooth	0.2	0.73	2.15	14.79	0.43	1.61	7.75
6-Star	0.35	0.54	1.53	12.23	0.22	0.61	4.30

TABLE 4. Time to extract iso-distance contour from simple models. Two CDFR resolutions have been tested with 3 levels of accuracy.

It is obvious, that finer accuracy results in longer extraction time. Using a higher resolution CDFR, this extraction time significantly drops, as shown by the 32-res versus 64-res CDFRs in Table 4. In Table 5, we show the speed-ups by using higher resolution CDFR’s. For any CDFR resolution, the number of points extracted for a certain distance contour stays roughly the same. Using a higher resolution CDFR, the average amount of time spent to extract a point on the contour is much less.

Model	Thickness (inch)	time/point (μs /point) 32-Res CDFR			time/point (μs /point) 64-Res CDFR		
		128 res	192 res	384 res	128 res	192 res	384 res
Cube	0.6	11.74	18.49	34.36	6.42	6.61	12.37
Tetra	0.1	11.98	12.91	19.63	4.72	4.99	6.92
1-Tooth	0.2	15.17	19.80	27.10	5.09	10.93	13.56
6-Star	0.35	34.03	43.83	78.36	10.37	12.80	20.41

TABLE 5. Comparing per-point extraction time (μs /voxel) at different CDFR resolutions.

On the two industrial parts, we chose a thickness of 0.3 inches for ‘connector’, and 4 inches for ‘brevi’. We used 128-res and 256 res CDFRs and the 3 levels of accuracy correspond to conventional volume resolutions of 512, 768 and 1024. The results in Table 6 confirm our findings from Table 4 and Table 5. Using higher resolution CDFR effectively cuts down iso-contour extraction time.

Model	128-Res CDFR			256-Res CDFR		
	512 res	768 res	1024 res	512 res	768 res	1024 res
Connector Timing (sec)	2.43	7.12	16.92	1.03	2.69	5.67
Connector: time/point (μs /point)	24.08	30.92	41.45	9.88	11.00	13.38
Brevi Timing (sec)	23.66	74.65	174.79	9.25	28.29	64.46
Brevi: time/point (μs /point)	32.51	45.27	59.72	12.23	16.63	21.36

TABLE 6. Connector’ and ‘brevi’ iso-contour extraction timing (sec) and per-point extraction time (μs /point).

We also tried to collect timing that extracts point-based model from ARCDF. However, we find the benefit of ARCDF difficult to gauge. If we subdivide a small portion of the volume using a high threshold, such as three times the average resident triangles (Table 3), since the subdivided voxels only comprise around 1% of all voxels, the acceleration is not very observable. If the threshold is decreased, the acceleration starts to become distinguishable, but effectively, the resulting timing is roughly on-par with using a higher initial resolution. For example, in the case described by Table 6, using a low-threshold, the contour extraction speed of 128-res CDFR with a 2 by 2 by 2 ARCDF come very close to the timing of using 256-res CDFR. In a way, this should be the expected results, due to the fact that ARCDF is a trade-off between low-resolution CDFR and high-resolution CDFR.

6.6 A Cutting Edge Application

We also applied our algorithm to a challenging design part, an engine cylinder head with 135,429 surface triangles. It is modified from a real design model for our research purposes. When built, it weighs 40Kg’s, and measures $266 \times 480 \times 157$ cm in size. Typically, the maximal thickness of engine cylinders is only about 9 to 10 mm. In engines blocks, heavy sections are an important source of physical failure. It is highly desired to be able to detect heavy sections at an accuracy higher than 0.15mm. Unfortunately, there has been no reliable and affordable way to perform such detection in the early design stages. Extensive resources have to be spent in the dreadfully long and cyclic design, prototyping and verification process. For conventional volume techniques to handle this task, one needs to build a volume having at least $1774 \times 3200 \times 1046$ voxels in floating-point numbers, amounting to 24GB. Even so, there is still no guarantee of an accurate Euclidean distance field, due to the binary surface volumes in voxelization, inaccurate distance map and linear interpolations used in reconstruction. Furthermore, constructing or rendering of data sets at such a size is overwhelmingly difficult.

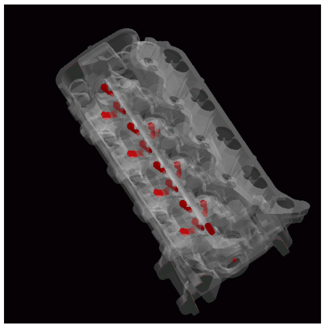
Figure 10: The thickness contour of a cylinder model, at 8.5mm thickness and 0.137mm error tolerance.

With the CDFR representation, at an CDFR resolution of $142 \times 250 \times 87$, we are able to construct a CDFR of size 37MB in 30 minutes. From this CDFR, we extract the distance contour as a point-based representation with 0.137 mm accuracy, corresponding to a $1988 \times 3500 \times 1218$ conventional volume resolution. For a thickness contour of 8.5mm, the extraction stage takes about 632 seconds, and the resulting point-based model has 450K points and is rendered interactively at 2 frames/sec, with the surface triangles being sorted and rendered semi-transparently at the same time (see Fig. 10). This whole process is done on our 512MB memory, 300MHz processor SGI Octane.

This result is strong, in that it brings an unprecedented accuracy to commonly available computing platforms. It is a tool that provides a guarantee of correctness as well as an interactive capability for visual investigation of highly complex models. This specific model is courtesy of Ford Motor Company.

7. Discussions and Future Work

Distance fields have traditionally be treated as another application of the Nyquist sampling theory. Using volume graphics techniques, a lot of new capabilities have been made available to CAD engineers, medical practitioners, etc. However, in some cases, such as, heavy section detection, thin section detection, tooling feasibility evaluation and die-castability evaluation, when accuracy is at a high priority, convention discrete distance field methods can not provide satisfactory results. The main reason was due to our lack of understanding of losses in geometric details after low-pass filtering in frequency domain.

CDFR has been proposed as an accurate description of distance fields resulting from a surface shape. CDFR volumes are not band-limited and exactly capture surface details in the 3D volume. Specifically, the advantages offered by CDFR on triangle surface meshes have been demonstrated. High accuracy and visual quality is achievable with a point-based iso-distance contour extraction. In addition to providing a proof of correctness and visualization results on real data sets, performance and storage issues have been discussed. With CDFR, the initial resolution of the volume does not affect the accuracy in subsequent iso-surface extraction. The initial resolution, however, provides a trade off between storage and speed. CDFR can be rendered directly with ray-casting using an adapted method of ray-object intersection check. However, we propose in this paper to use point-based approaches to rendering distance contours from CDFR with genuine per-point shading.

Similar to ADF methods, where conventional discrete distance fields were adaptively sampled, CDFR can also be organized hierarchically. In this paper, a way to create an adaptively represented complete distance field (ARCDF) is discussed. It is shown that the number of triangles resident on each voxel can be dynamically reduced by subdivision. However, such reduction in number of resident triangles is not very scalable as the number of subdivisions increase. The reason is obviously that in most distance fields that are non-linear, more than one surface triangle affects any local neighborhood in 3D space. In such cases, the number of resident triangles on each voxels reflects such property accordingly.

As a future extension, hierarchically organized CDFR provides a general framework to compute the Euclidean distance from a 3D point to polygonal mesh. While the applications discussed in this paper strictly deal with the interior of a shape, this framework can also be applied to the exterior of complex models and scenes. All distance values are computed exactly at a low constant time. No complicated approximation schemes [8] based on progressive meshes etc. is necessary. Applications that may benefit from this framework include path planning with a guaranteed accuracy in highly complicated scenes, such as assembly of airplanes, large scale machinery. General graphics applications such as hypertexture renderings can also make use of this novel distance field representation.

8. Acknowledgment

This project was funded by Ford Motor Company, Visteon, Inc., and NSF CAREER Award #9876022. A previous version of this paper was published in the Proceedings of IEEE Visualization Conference'2001 [13].

Reference

[1] D. Breen, S. Mauch and R. Whitaker, "3D scan conversion of CSG models into distance volumes", Proc. 1998 IEEE Symposium on Volume Visualization, pp. 7-14, 1998.

[2] H. Cline, W. Lorensen, S. Ludke, C. Crawford, B. Teeter, "Two algorithms for the three-dimensional reconstruction of tomograms", Medical Physics, 15(3), May/June, 1988, pp. 320-327.

[3] D. Cohen-Or, D. Levin, and A. Solomovici, "Three-dimensional distance field metamorphosis", ACM Transactions on Graphics, Vol. 17, No. 2, pp. 116-141, April 1998.

[4] Crawfis, R., Max, N., Texture Splats for 3D Scalar and Vector Field Visualization, IEEE Visualization'93 Proceedings, October, 1993, 261-267.

[5] F. Dachille, A. Kaufman, "Incremental triangle voxelization", Proc. of Graphics Interface 2000, pp. 205-212, May 2000.

[6] H. Pfister, J. Barr, M. Zwicker, M. Gross, "Surfel: surface elements as rendering primitives", Proc. of Siggraph 2000, New Orleans, 2000.

[7] S. Frisken, R. Perry, A. Rockwood, T. Jones, Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics, Proc. of SIGGRAPH'2000, New Orleans, LA, July, 2000.

[8] A. Gueziec, "Meshsweeper: dynamic point-to-polygonal mesh distance and applications", IEEE Transactions on visualization and computer graphics, vol. 7, no. 1, pp. 47 - 61, Jan - Mar, 2001.

[9] S. Gibson, Using Distance Maps for smooth surface representation in sampled volumes, Proc. 1998 IEEE Volume Visualization Symposium, pp. 23-30, 1998.

[10] W. Lorensen, H. Cline, Marching Cubes: a high resolution 3D surface construction algorithm, Computer Graphics (SIGGRAPH 87 Proceedings), 1987, 163-169.

[11] J. Huang, R. Yagel, V. Fillipov, Y. Kurzion, An Accurate Method to Voxelize Polygonal Meshes, Proc. of IEEE/ACM Symposium on Volume Visualization, October, 1998, Chapel Hill, NC.

[12] J. Huang, K. Mueller, N. Shareef, R. Crawfis, "FastSplats: optimized splatting on rectilinear grids", Proc. of IEEE Conference on Visualization, October, 2000, Salt Lake City, Utah

[13] J. Huang, Y. Li, R. Crawfis, S. Lu, S. Liou, "A Complete Distance Field Representation", Proc. of IEEE Conference on Visualization, October, 2001, San Diego, CA.

[14] A. Kaufman, "An algorithm for 3D scan-conversion of polygons", Proc. of Eurographics'87, pp. 197-208, North Holland, August, 1987.

[15] A. Kaufman, "An algorithm for 3D scan-conversion of parametric curves, surfaces, and volumes", Proc. of SIGGRAPH'87, pp. 171-179, July, 1987.

[16] L. Kobbelt, M. Botsch, U. Schwanecke, H. Seidel, "Feature sensitive surface extraction from volume data", Proc. of SIGGRAPH'2001, July, 2001.

[17] S. Lu, A. Rebello, R. Miller, G. Kinzel, R. Yagel, "A simple visualization tool to support concurrent engineering design", Journal of Computer-Aided Design, Vol. 29, No. 10, pp. 727-735.

[18] Meissner, M., Huang, J., Bartz, D., Mueller, K., Crawfis, R., A Practical Evaluation of Popular Volume Rendering Algorithms, Proc. of Symposium of Volume Graphics 2000, Salt Late City, Utah.

[19] Muller, K., Shareef, N., Huang, J., Crawfis, R., High-Quality Splatting on Rectilinear Grids with Efficient Culling of Occluded Voxels, IEEE Transaction on Visualization and Computer Graphics, Vol. 5, No. 2, pp 116-135, 1999.

[20] S. Parker, M. Parker, Y. Livnat, P. Sloan, C. Hansen, and P. Shirley, "Interactive ray tracing for volume

visualization" IEEE Transactions On Visualization and Computer Graphics, Vol. 5 (3), pp. 238-250, 1999.

[21] S. Rusinkiewicz, M., Levoy, "QSplat: a multi-resolution point rendering system for large meshes", Proc. of Siggraph 2000, New Orleans, 2000.

[22] M. Sramek, A. Kaufman, "Alias-free voxelization of geometric objects", IEEE Transactions On Visualization and Computer Graphics, Vol. 5, (3), pp. 250-267, 1999.

[23] R. Yagel, S. Lu, A. Rubello, R. Miller, "Volume-based reasoning and visualization of dicastability" In Proc. IEEE Visualization '95, pp. 359-362, 1995.

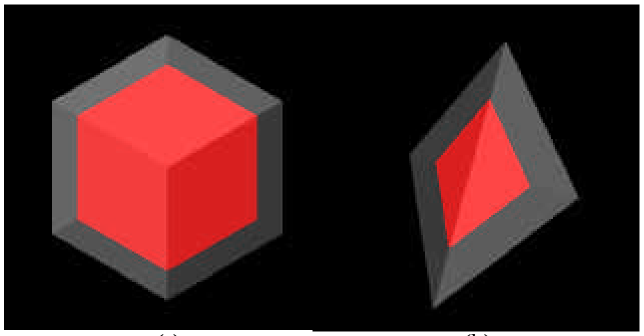

Figure 7. A cube and tetrahedron, with the surface mesh shown in semi-transparency. The distance contours (shown in red, per-point shaded) are extracted with an error tolerance of 1/500 of the longest dimension of the models. The thickness in (a) is 0.6 inch

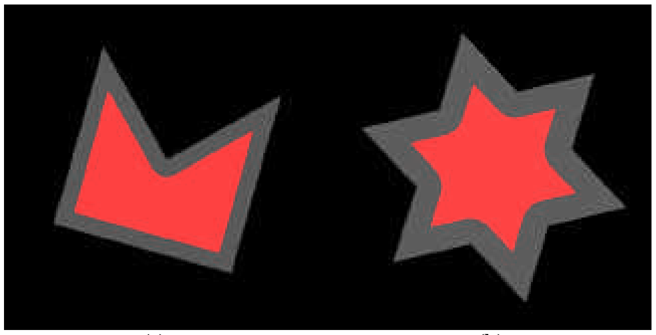

Figure 8. Two concave examples, a 6-pointed star and a one-end tooth. (a) At small thickness (0.2 inch), the distance field retains most corners and edges, with little smoothing. (b) As thickness increases (0.35 inch), the distance field evolves into the core of the

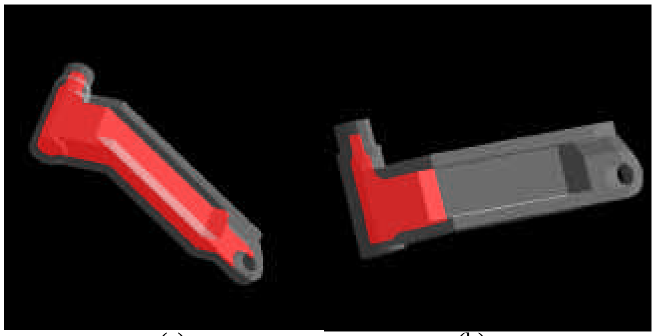

Figure 9. Results of ‘connector’. The surface mesh is shown in semi-transparency, and the per-point shaded distance contours are at thicknesses (a) 0.2 inch and (b) 0.35 inch.

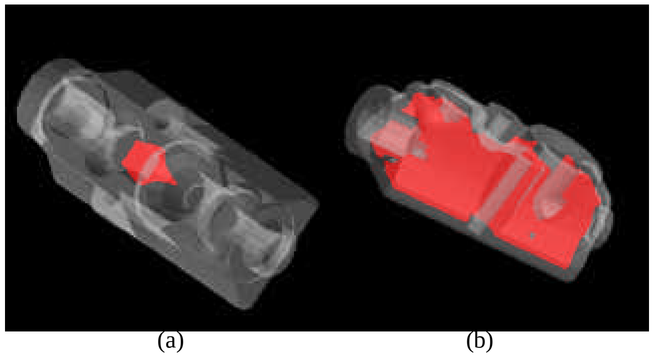

Figure 10. The sample images of ‘brevi’. With the per-point shaded contours at thickness (a) 10 inches and (b) 4 inches.

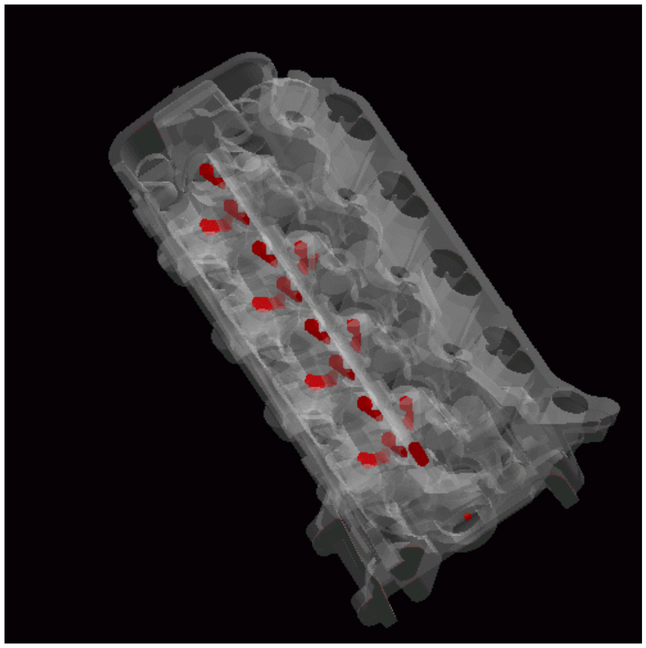

Figure 12. The thickness contour of a modified engine cylinder head model, at 8.5mm thickness and 0.137mm error tolerance.