# Complete Distance Field Representation

Jian Huang[+], Yan Li[*], Roger Crawfis[+] and Shao-Chiung Lu[$]
[+]Computer and Information Science, The Ohio State University, Columbus, OH
[*]Electrical Engineering, The Ohio State University, Columbus, OH
[$]Visteon Inc., Dearborn, MI
{huangj,crawfis}@cis.ohio-state.edu, liy@ee.eng.ohio-state.edu, slu3@visteon.com

## Abstract

Distance field is an important volume representation. A high quality distance field facilitates accurate surface characterization and gradient estimation. However, due to Nyquist's Law, no existing methods based on the linear sampling theory can fully capture the sharp 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 anymore. Rather we construct a volume, in which each voxel has a complete description of all portions of surface that affect the local distance field. For any desired distance contour, we are able to extract a point-based contour in true Euclidean distance, at any error tolerance, from the CDFR. Leveraging the advantages offered by CDFR, we can efficiently visualize any distance contour at real-time frame rates using the point based model. We also demonstrate applications of CDFR to CAD applications involving high-complexity parts at un-precedented high accuracy using very limited computational resources.

**CR Categories:** I.3.6 [Computer graphics]: Methodology and techniques - Graphics data structures; I.3.5 Computational Geometry and Object Modeling - Object modeling

**Keywords:** distance fields, volume modeling, polygonal surfaces, point-based models, graphics.

## 1. Introduction

Volume, as a 3D raster, holds discrete sample points representing a certain multi-dimensional entity. In an alias-free discretization, only frequency components below half the Nyquist sampling rate can be present in the volume data set. As a solid description of physical entities, volume has found applications in a wide range of fields, such as medicine, mechanical engineering, scientific computing and simulations, etc.

In order to utilize volume technologies, it has been common to convert surface models, such as a polygonal mesh exported in STL format 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 [1][5][7][8]. Second, a distance transform is computed to construct a solid volume that encompasses a thickness field computed based on distances to the surface shapes. 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, and chaffer distance.

Voxelization techniques that convert surface shapes into binary volumes, with 1's representing occupancy and 0's representing empty space, have been developed in [5][7][8]. Those methods are practical and commonly used nowadays. Surface shapes are infinitely thin in space. To sample that thin shape, one needs infinitely high sampling rates. In essence, Kaufman's algorithms [7][8] increase the thickness of a surface shape, and therefore, to some extent, band limit the frequency spectrum, before the sampling, or scan-conversion process, take place. Huang et. al [5] discovered and proved the sufficient and necessary amount of increase in thickness of the surface shape, which is also dependent on the resolution of the volume, that guarantees correctness in discrete topology in the resulting volume representation. Unfortunately, all these methods that are based on binary volume representation, which is very susceptible to aliasing artifacts. To address this issue, Sramek and Kaufman initiated to represent data sets in non-binary formats [10]. In their paper, they show one has to use higher order smoothing functions to pre-filter and band limit the spectrum of the volume. They also provide a measure to determine the amount of preservable details in shape after applying common smoothing filters.

Over the years, in addition to the search for optimal voxelization, the community has also been exploring to represent surface shapes with distance fields. Distance fields have typically been regarded as 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 outside or inside of the shape. Compared to the surface shapes that correspond to impulses in 3D space, distance fields are much smoother. On one hand, for a shapes free from sharp corners and edges, both the surface position and gradient can be reconstructed relatively accurately with a distance field [1][4]. On the other hand, as soon as corners and sharp edges, features that carve out details on the shape, are introduced, high frequency components are also brought into the spectrum. Exceptionally high volume resolution may be necessary to achieve an alias-free representation. Adaptively sampled distance fields (ADF) [3] can help in reducing volume resolution when not as much details are locally present. But, their methods still have to rely on limiting the bandwidth of spectrum in local neighborhoods of corners and edges, and hence discard all details beyond the cut-off bandwidth supported by the leaf level in their tree structure.

In this paper, we present a novel scheme of representing distance field. Although the general steps of voxelization and distance transform are still followed, we build our distance representation based on a complete distance definition (CDD). We would rather not use the simplistic scalar distance field representation, because the band-limited spectrum required by the Nyquist's Law is opposite to the need of most graphics applications favoring elaborate details of miniature sizes. Since representations of distance fields in CDD, that is, complete distance field representation (CDFR), can not be efficiently rendered with current rendering algorithms, we also discuss the method to reconstruct a point-based contour of any distance value at any error tolerance from CDFR. Such sparse point-based models that we extract from CDFR can be efficiently rendered with splatting [6].

The paper is organized in the following way. We give a brief introduction to distance fields and the limitations of such representations in Section 2. In Section 3, we present complete distance definition (CDD) and complete distance field representation (CDFR) based on CDD. We provide a proof of correctness for our

scheme of constructing a CDFR and extracting the final point-based contour from CDFR in Section 4. Results on simple test data sets and real world geometric designs are shown in Section 5 and Section 6. Finally, we conclude and discuss future work in Section 7.

## 2. Distance Fields

Traditionally, distance field is defined as, a spatial field of distance values to a surface geometry or shape. Each element in a distance field specifies its minimum distance to the surface shape. Positive and negative signs are used to distinguish outside and inside of the geometry, for instance, using negative values on the outside, and positive on the inside. Distance fields have a number of applications in distance mapping [1], constructive solid geometry operations [3], surface reconstruction and normal estimation[4], morphing [1][2], and etc. Distance fields are also widely applied in simulations and analysis involving interior of geometries, such as die-casting simulation in manufacturing, thickness analysis of mechanical parts [11].

For a non-aliased discrete 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 spectrum. The sharp details on the surface, such as corners and edges, also reside on the high ends in the frequency 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 [5][7][8] or specifically designed higher order filters [10], all inevitably cause a loss of the exact surface location and also surface details. Converting the surface shape to a distance field, which is smoother, provides a way to exactly locate the [4] during reconstruction. But the underlying assumption of having a smooth surface that is free from sharp corners and edges is too strong to hold for practical scenarios. Geometric models, such as CAD parts, render themselves almost pointless without the high frequency details.

Frisken et. al [3] developed a nice 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 stride over different levels of detail during rendering as needed. They also vary sampling rates according to the amount of details that are available locally. Because the authors limited their applications to rendering, they went by using tri-linear interpolation to reconstruct distances, and were able to demonstrate a suite of applications with impressive visual quality using relatively small memory storage.

However, ADF [3] does not fundamentally solve the problem of losing surface details in discrete representations. When the primary goal of an application shifts from rendering to analysis based on the distance values, ADFs with trilinear interpolation do not satisfy our accuracy needs with a guarantee anymore, simply because distance fields are not linear as soon as corners or edges are introduced into the models. Furthermore, what the hierarchies provide is an ability to save computational and storage resources when less details are encountered. After the leaf level of ADF is constructed, the loss in surface details is already final and irreversible. For the applications where accuracy in analysis is sought after more highly than pleasing visual experiences, current ADFs based on scalar distances are not satisfactory, because they can not provide a high level of accuracy in a finite amount of storage space. Hence, extensive low pass filtering is often used. But most practical geometrical models are rich in details at a wide range of scales and in lack of universal smoothness. Due to the fact that a reliable and trusted metric to characterize geometric details still remains an open question in the field, one has to always use the highest possible resolution he can afford, but still, has no guarantee that he has captured all the details to the accuracy he needs unless the size of each voxel is below half of the error tolerance. Furthermore, as soon as the user decides to increase his accuracy needs, the process to construct an ADF tree in a much larger scale is un-avoidable. Applications requiring accuracy are thus hindered from using volume technology on practical computing platforms.

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

We believe there is in need of a paradigm shift to overcome such overwhelming costs for achieving high accuracy guarantees that can be modified by the user efficiently as needed. In this paper we propose to use complete distance definition (CDD) instead of the traditional single value distance. Not surprisingly, corresponding to different ways of representing surfaces, such as parametric surfaces, implicit surfaces or subdivision/polygonal mesh surfaces, there could be different instantiations of CDDs. In order for a clear elucidation of CDDs, among the several very similar CDD instantiations, we opt to focus on a simple case where the surface is represented by polygonal meshes. When accuracy is asked for, Euclidean distances are often preferred over other distance metrics, such as chaffer distance or Manhattan distance. The CDD fields that we discuss in this paper are all true Euclidean distances. Before discussing CDD, we would like to show off a few simple observations that motivated our work based on CDDs.

### 3.1 Some Observations

#### 3.1.1 Observation 1

Distance field has a very interesting feature in some very simple scenarios. For instance, suppose in a 1-dimensional space, there is an impulse. It's frequency components extend to infinity. There is no way that one can use a finite sampling frequency to sample that impulse with no aliasing. But on the other hand, as illustrated in Fig. 1, the distance field of that impulse is a linear function that extends from negative infinity to positive infinity. Sampling this linear function is very easy with a low sampling frequency.

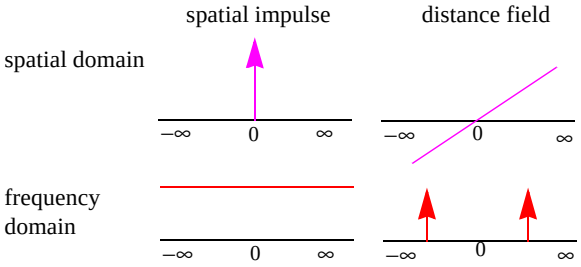


Figure 1: It's impossible to sample the impulses (left), but we can sample the distance field (right).

However, this feature does not hold in higher dimensions where corners are present. Just the same as the complexity encountered in [5], as we extend into 2D or 3D, the discrepancies and discontinuity on corners make it difficult to preserve a faithful distance field during sampling.

For instance, in Fig. 2, we have a rather faithful sampling in the blue grids, because the geometry is locally linear*, but not in the red grids where there are corners. The non-linear distance fields

within the red grids, makes it impossible to accurately recover the right distance distribution from the samples in the grid.

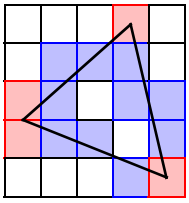

Figure 2: Corners in this 2D case cause significant complexities in the distance field, which corresponds to high frequency components that will result in aliasing in spectrum after sampled.

According to Nyquist's Law, to sample such complicated distance fields, one must first band-limit the fields by performing a low-pass filtering, which essentially smooths out the sharp corners. With the ADF instantiations using Octree based regular grid representing distance field [3], for corners (the red grids), a higher resolution would be used, whereas in blue grids, a much lower resolution might suffice.

### 3.1.2 Observation 2

To capture the exact location of the impulse in Fig. 1, we do not have to use sampling. Essentially, one only needs to have one sample at any location, and note which side the impulse is on as well as the distance from that sample to the impulse, unlike sampling theory that requires an infinite sequence of samples above the Nyquist's rate, and use a sinc function to convolve and reconstruct for the original continuous space function when needed.

## 3.2 Complete Distance Definition (CDD)

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

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

The value *distance* is the true distance of a 3D point to a finite triangle. It is defined in the following function:

```
float distance (triangle tri, vec3 pnt)
{
  if (in(tri, projToPlane_tri(pnt)) // pnt projects to within tri,     C1
      return (distance from pnt to the plane which tri is on);
  else if (on(any edge of tri, e, projToLine_e(pnt))                  // C2
      return ⟨mindist(e, pnt)|∀e,tri⟩
  else                                                                // C3
      return ⟨mindist(v, pnt)|∀v,tri⟩ ;
}
```

Figure 3: Pseudo code that defines the distance of a 3D point, *pnt*, to a finite triangle, *tri*, positioned in a 3D space.

While the return value is the computed distance, the input parameters include a triangle, *tri*, and a 3D point, *pnt*. If *pnt* projects to a point within the triangle, *tri*, the final distance is the orthogonal distance from *pnt* to the plane in which *tri* lies. If *pnt* projects to a point out side of *tri*, then, we check whether *pnt* projects to a point onto any one of the three edges. If yes, then the returned distance value is shortest distance from *pnt* to an edge that *pnt* projects orthogonal onto. In case none of the above two criteria applies, we define the distance as the minimal distance from *pnt* to each of the three vertices. This definition of distance to a finite triangle can be further illustrated in Fig. 4.

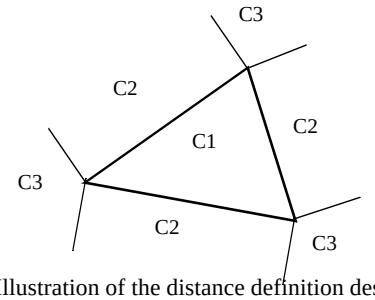


Figure 4: Illustration of the distance definition described in Fig. 3. In the diagram, we set of viewing angle in the direction of the normal(*tri*). If pnt projects into the tri, then it's case C1. Otherwise, if pnt orthogonally projects onto an edge, then we have one of the three C2 case. Finally, if pnt falls into neither C1 or C2, it has to be one of the three C3 cases, where it is closest to a vertex.

## 3.3 CD Field Representation (CDFR)

In a conventional distance field representation, on each grid point, only one scalar value, the minimal distance to shape, is stored. This assumes the overall distance function to be band-limited. Based upon such an assumption, when a reconstruction of the distance field is needed, an interpolation or a filtering operation, such as trilinear interpolation [3], is thus sufficient. In this paper, to capture all high frequency details, we would like to construct a representation that do not make such strong assumptions as requiring band-limited distance fields. In most practical cases, it is the geometric details that are most interesting to users.

We use CDD to build a complete distance field representation (CDFR) that allows us to exactly trace all geometric details, such as sharp corners and edges, to any level of accuracy. We prove that our CDFR is sufficient, and flexible in trading storage space for speed, and vice versa.

Firstly, given a geometric representation of shape, such as a triangle mesh exported from CAD software, a generally accepted approach would be to voxelize the surface mesh into a surface volume, in which voxels on the surface are set while all the other voxels remain blank. For all triangles touching a surface voxel, a CDD tuple is stored with that voxels. The end result of the modified voxelization approach would leave all surface voxels with a list of CDD tuples, sorted in ascending order in distance values.

*Locally linear: perfect reconstruction is possible with linear interpolation.

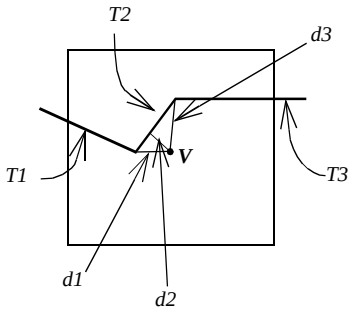


Figure 5: An illustration of building a CDD tuple list for a surface voxel, *V*. The CDD tuple list is organizes in ascending distance order, with the minimal distance of *V* being *d2*, in the above case.

The second step is distance transformation. Initially, we use an outside flooding to eliminate all outside voxels from our computational pipeline. For voxels that still remains, a contour-by-contour CDD propagation is performed from outside to inside. During this process, a voxel keeps looking for CDD tuples that have been newly propagated to anyone of its 26-neighbors [5]. It inherits all new CDD tuples from its neighbors, and for each triangle, it computes the true Euclidean distance from its own position. The updated list of CDD tuples are then sorted into ascending order again, so that the first CDD tuple in list contains the current distance, *cur_distance*, from this voxel to the closest surface triangle. All the updated CDD tuples that correspond to a distance value that is within the range:

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

are stored with that voxel. At the mean time, those CDD tuples that fall out of this range are discarded. This whole process of distance transformation iterates until no voxels can find new CDD tuples from its 26-neighbors that affects its own minimal distance to the surface geometry.

## 3.4 Reconstructing A Distance Contour

The most frequent way in which a distance field is used is by reconstructing an iso-distance contour. For instance a user may specify, "Show me the zero distance contour with at an error tolerance of 0.5mm". Sometimes, the corresponding gradients are also needed on that iso-distance contour. The conventional way of reconstructing the distance field in-between voxels, i.e. at sub-voxel resolution, is to use an interpolation scheme [3], such as trilinear interpolation. While this works well for some data sets, there is no guarantee on the ascertained level of accuracy. We opt to use a different method, which provides much more leverage in accuracy based on CDFR.

Given a requested thickness, *t*, we traverse the whole volume, and go through the inter-voxel regions, marked by 8 corner voxels, one by one. We look for those inter-voxel regions whose 8 corner voxels have a minimal thickness that is less than or equivalent to *t*, and maximal thickness above or equivalent to *t*. The requested iso-contour goes through these inter-voxel regions that are identified as such. We then subdivide the inter-voxel region into sub-voxels. In order to support the error tolerance picked by the user, we need to make sure:

$$\frac{\sqrt{3}}{2} subvoxelsize < E \qquad (2)$$

For each sub-voxel, we compute the true distance value according to all the CDD tuples resident on anyone of the 8 cornering voxels. The sub-voxels that have a minimal true distance value within the range $[t - E/2,\ t + E/2]$ are deemed to be on the contour. The normal of each sub-voxel is the same as the normal of its closest triangle. High quality per sub-voxel shading is thus provided.

# 4. Proof of Sufficiency

To prove the correctness of CDFR, we first need a proof of sufficiency, that is, when we need to reconstruct the local distance field in any inter-voxel region, all the surface primitives affecting this local area are present on at least one of the 8 cornering voxels.

A surface primitive, such as 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 exist a local inter-voxel region, *L*, in which exists at least one point, *P(x,y,z)*, to which the closest surface triangle, *T*, is not resident on any of the 8 corners of *L*. We label the 8 corners as, $C_i$, $i = 1, 2 \ldots 8$.

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* in 3D, the distance from *P'* to *T* is bounded by:

$$[D - \sqrt{dx^2 + dy^2 + dz^2}, D + \sqrt{dx^2 + dy^2 + dz^2}] \qquad (3)$$

Due to deduction, when *P'* incrementally moves towards any one of the 8 corners of *L*, say, $C_i$, it logically follows that the distance from $C_i$ to *T* is bounded by:

$$[D - \int_{p'}^{C_i} ds, D + \int_{p'}^{C_i} ds] \qquad (4)$$

Equation (4) can be rewritten as:

$$[D - dist(C_i, P'), D + dist(C_i, P')] \qquad (5)$$

In a very similar way in which Equation (3), (4) and (5) are set up, we can also find out that the minimum distance on *P*, which is *D*, must be smaller than $minD_i + dist(C_i, P)$, with $minD_i$ denoting the minimum distance on $C_i$. Therefore, the distance of *T* to $C_i$, must be within the following range:

$$[minD_i, minD_i + 2dist(C_i, P')] \qquad (6)$$

Since for a corner voxel that is the closest to *P*, the maximum distance is $(\sqrt{3})/2\ \ voxel\ size$, Equation (6) is actually a subset of:

$$[minD_i, minD_i + \sqrt{3} voxelsize] \qquad (7)$$

Contradiction. Since during our distance propagation process, Equation (7) is exactly the range that we maintain on each voxel. Proof completed.

We do not claim our scheme is minimal, in that, we might have kept more surface primitives on each voxel than necessary. However, it is obvious that minimality criteria 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 in storing extra CDD tuples is low, with which we have traded for simplicity in implementation and speed in execution.

# 5. Experimental Results

With CDFR, the initial resolution of the volume does not dictate the accuracy of the distance field any more. The conventional single closely coupled pipeline, involving voxelization, distance transform and graphics operations, is now broken into a two independent steps, constructing a CDFR and reconstruction of a certain distance or a set of distances contours as a point-based model at a given level of accuracy.

We first present some simple cases to justify the correctness of our algorithm with experimental results, in addition to our theoretical proof in Section 4.

Cube and tetrahedral cells are the simplest possible examples. They are convex and can be symmetric. With true Euclidean distance field, the thickness contours of different values are still in the shape of the outer surfaces throughout the whole range of thickness, Fig. 6.

Figure 6: A sample model cube and a sample model of a tetrahedral cell, with the surface mesh shown in semi-transparency, and the inner shell at a distance value that is half the size of the cube, shown in red.

Concaveness causes much complexity in distance fields. To demonstrate our methods, we use two sample models in Fig. 7, a two-ended tooth and a 6-star. For the two-ended tooth, we chose a very small thickness value and extracts a shell close to the surface mesh, while for the 6-star, a large thickness value, corresponding to a distance deep into the core of the model, is chosen. In doing so, we would like to demonstrate the interesting evolving effects that can only be shown with a true Euclidean distance field representation.

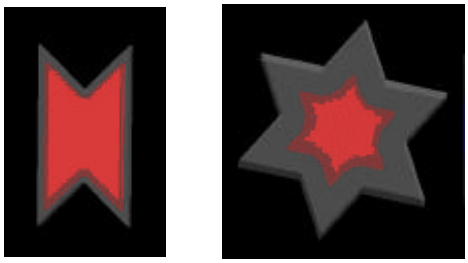

Figure 7: Two more complicated examples, a 6-star and a two-ended tooth. As we can see, when the thickness value picked is fairly small (close to the surface, the case for the two-ended tooth) the distance field remains very close to the shape of the surface, retaining most corners and sharp edges, with some smoothing. When the thickness value increases and the distance field evolves into core of the model, the true Euclidean distance field shows more smoothing effects, losing most surface details and but still obviously showing the global feature of the shape.

In the distance field, small thickness contours are closer to the surfaces, and retain much detail of the surface shape or topology. The larger the thickness gets, the more effects of smoothing are introduced. The shape of the thickness contours start to manifest more global features of the shape than small-scale details.

All four data models are voxelized at a resolution of $32 \times 32 \times 32$. In reconstructing the thickness contour that we show in Fig. 6 and 7, we set the accuracy to one 1000th of the size of each of the geometric model. Due to the space saving advantage of our method, we only reconstruct, store and render the subvoxels exactly on the desired thickness contour, within our acceptable error tolerance. In this case, for all 4 models, there are only several thousand subvoxels, or points, to render. We achieved 30 frames/second rendering of our high accuracy point-based representation of the distance contour, which would throttle most rendering engines if traditional simple scalar distance field is instead used. At a volume resolution of 1000, it is overwhelmingly difficult to stream the huge volumetric distance field otherwise needed by traditional approaches into the system and render at real time, even with ADF hierarchies.

## 6. Real-world Application Results

Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) has become an important field. CAD techniques play indispensable roles in shortening time to market, early detection of design artifacts, and so on. But, traditional surface graphics based CAD/CAM systems lack the ability to efficiently carry out projects involving the interior of part designs. Volume techniques have been considered. But due to the limitation in achievable accuracy with current computing resources, the application of volume graphics has been limited to where only a trend of change is of interest, and accuracy is not too much of a concern. While we have demonstrated both the correctness and efficiency that we can achieve with CDFR on some simple test data sets, we also tested on large real-world models, for which both accuracy and high frame rates are highly sought after.

We tested our algorithm on several real application data sets, Connector, Brevi, and Engine, each of a different amount of complexity. We present both the timing of construction and size of CDFR storage, with regard to different initial resolution. We also present the timing in reconstructing the final distance shell for usage, at a number of final thickness values, accuracy level, and of course, at different initial CFDR resolution.

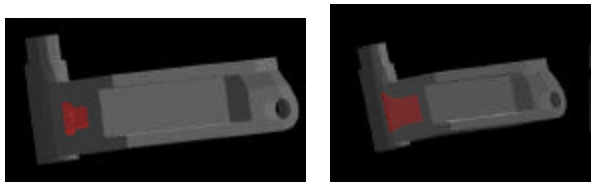

Figure 8: Results of a part design, Connector, with 327 triangles. The surface mesh is shown in semi-transparency, and the inner distance shell picked are at thickness 0.6 and 0.5, both at an accuracy of 1000th the size of the part's length. Picking a different initial resolution of CDFR, does not affect the final results, but results in different timings.

In Fig. 8, we show results of the Connector part. It is a relatively simple part, with only 327 triangles on its surface. Choosing a different initial CDFR resolution results in different timings in both constructing CDFR and extracting a distance shell after that.

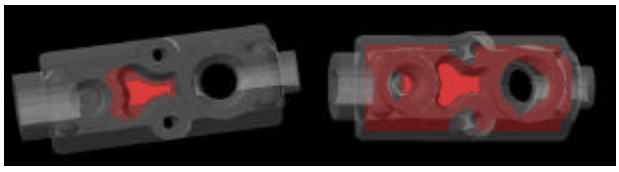

Figure 9: Brevi. With the final results being picked at different thickness.

We also tried our algorithm out on a very large design part, Engine, with more than 130K triangles on its surface. When built, it should be of 40KGs in weight, and $60 \times 30 \times 25$ cm in size. A very interesting feature of this part is that the maximal thickness in this part is only 11mm. To heavy sections in the part, for instance, thicker than 10mm, at the designer's required accuracy, e.g. 0.1mm, one would need to build a floating-point volume at least $6000 \times 3000 \times 2500$ in size using the traditional approach. With CDFR representation, we use an initial CDFR resolution of 300, and extract the distance shell at 10mm thickness, at 0.1 mm accuracy, in 30 seconds, and render the results, Fig. 10, at 30 frames/sec.

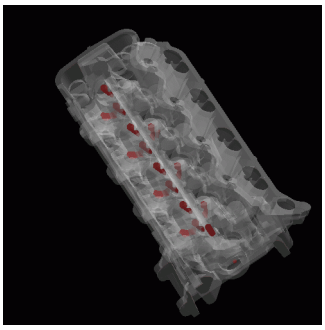

Figure 10: The thickness shell of the Engine, at 10mm thickness and 0.1mm error tolerance. This image can be rendered at 30frames/sec.

## 7. Discussions and Future Work

In conclusion, in this paper we propose to use CDFR to efficiently and accurately describe distance fields resulting from a surface shape. Unlike the linear sampling theory based on Nyquist's law, our volume representation of distance field does not have to abide by the requirements of limiting bandwidth. We can exactly capture details that have infinite bandwidth in the 3D space. Specifically, we demonstrate the advantages offered by CDFR on surface meshes represented with triangles. In addition to providing a proof of correctness, timing and storage issues have been presented and discussed as well. In general, we believe CDFR is an accurate way to represent a distance field.

For the first time, the initial resolution of the volume representation does not affect the accuracy that is achievable later on any more. Rather, the initial resolution provides a measure of trading storage for speed or vice versa. One good extension that we have not implemented is to use hierarchical data structures to organize CDFR. Such a scheme could expedite the reconstruction process without causing a significant increase in storage needs. It would be straightforward to use number of triangles resident on a voxel as a measure of determining whether further subdivision should be carried out. Combined with hierarchical frameworks, such as ADF [3], CDFR would become a practical representation for use in both the general graphics domains and highly applied fields, such as CAD/CAM.

## 8. Acknowledgment

This project has been funded by Visteon, Inc., under a one-year grant to Department of Computer and Information Science, the Ohio State University, Columbus, OH.

### 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] 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.

[3] 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.

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

[5] 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.

[6] 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

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

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

[9] 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.

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

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