Signed Distance Fields for Polygon Soup Meshes

Hongyi Xu* Jernej Barbič†

University of Southern California

ABSTRACT

Many meshes in computer animation practice are meant to approximate solid objects, but the provided triangular geometry is often unoriented, non-manifold or contains self-intersections, causing inside/outside of objects to be mathematically ill-defined. We describe a robust and efficient automatic approach to define and compute a signed distance field for arbitrary triangular geometry. Starting with arbitrary (non-manifold) triangular geometry, we first define and extract an offset manifold surface using an unsigned distance field. We then automatically remove any interior surface components. Finally, we exploit the manifoldness of the offset surface to quickly detect interior distance field grid points. We prove that exterior grid points can reuse a shifted original unsigned distance field, whereas for interior cells, we compute the signed field from the offset surface geometry. We demonstrate improved performance both using exact distance fields computed using an octree, and approximate distance fields computed using fast marching. We analyze the time and memory costs for complex meshes that include self-intersections and non-manifold geometry. We demonstrate the effectiveness of our algorithm by using the signed distance field for collision detection and generation of tetrahedral meshes for physically based simulation.

Index Terms: Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1 INTRODUCTION

Given a collection of 3D triangles and a query 3D location x, there exists some triangle (and a feature on this triangle) closest to x. *Distance field* is a scalar function that gives the minimum distances for points x from some region of space, such as a bounding box enclosing the triangular geometry. Distance fields sampled on regular 3D grids are a popular datastructure in computer graphics [21], and have been used in many applications, such as collision detection and morphing. Distance fields can be signed or unsigned. Signed distance fields store the sign specifying whether the query point is inside/outside of the object. Sign is only meaningful, however, if the input mesh is a watertight mesh with manifold geometry. If the input is a general "triangle soup", the sign is not well-defined and in principle only an unsigned distance field can be computed.

We present an approach to both *define* and *compute* a signed distance field for any input triangular geometry, including geometry containing self-collisions, gaps, holes or inconsistently oriented, disconnected, non-closed, noisy or duplicated geometry (see Figure 2). Such geometry is very common in computer graphics practice, for example, with 3D characters, mechanical components, and surgery simulation. Unlike most signed distance field computation methods that assume a well-defined watertight manifold input mesh and then optimize the distance field computation, we address the problem of how to first define the sign for any triangular geometry,

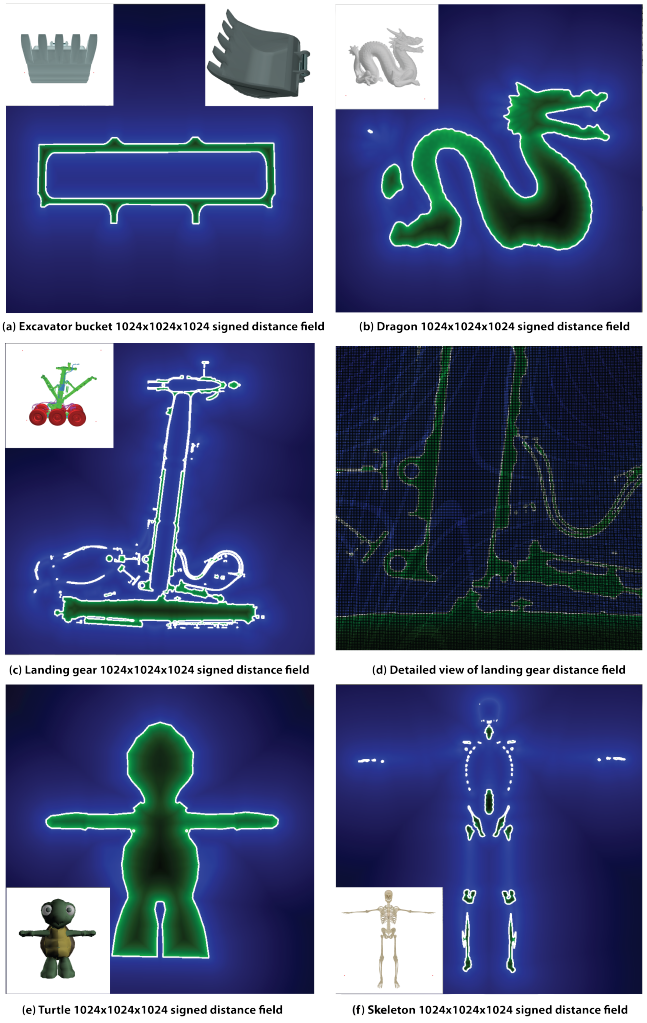


Figure 1: Signed distance fields for non-manifold geometry.

and then compute the signed distance field efficiently. Our approach is not specific to any distance field computation method. We define and rapidly compute exact signed distance fields using an octree, and approximate distance fields using fast marching [39]. Exact signed distance fields are useful, for example, in collision detection, where inexact signed distance fields will lead to missed collisions if the distance field is queried against a bounding volume hierarchy. Fast marching only gives approximate distance fields, and is advantageous for speed, especially with geometrically complex meshes.

Our method first computes an unsigned distance field to the input geometry. It then extracts a *manifold* offset isosurface, using the marching cubes algorithm with topological guarantees [26]. The offset surface distance σ is the only parameter that the user needs to adjust. This parameter has an intuitive meaning: geometry imperfections smaller than σ are considered "noise" and are automati-

*e-mail: hongyixu@usc.edu

†e-mail:jnb@usc.edu

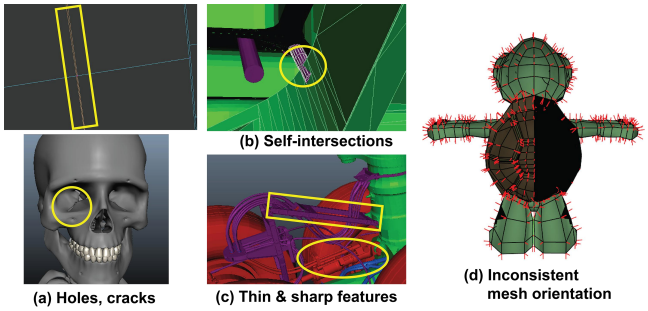


Figure 2: Non-manifold, self-intersecting and inconsistently oriented input meshes.

cally fused together, whereas gaps larger than σ are above the "engineering tolerance" and contribute to the signed distance field. The resulting isosurface in general consists of several disjoint (potentially nested) components, and we automatically remove components completely contained in other components (interior components), using a pseudonormal test [5]. Because the remaining components form non-nested manifold watertight surfaces that enclose well-defined solids, one could compute the signed distance field directly from their geometry [6]. However, at high resolutions, offset surfaces usually contain many more triangles than the input mesh, which results in signed computations much slower than unsigned field computation. We accelerate the signed computation by two orders of magnitude by proving that the exterior signed distance field is simply a shifted original unsigned distance field, which can therefore be re-used in the exterior region. We exploit the manifoldness of the offset surface to quickly identify the interior grid points, and then compute the signed distance field only for the interior grid points. The final distance field is accurate with respect to the isosurface geometry. Our method is very efficient for meshes where the exterior space is much larger than the interior space, as is the case with many mechanical components, characters and medical meshes. We use our signed distance fields to perform collision detection between objects with non-manifold triangular geometry. Our signed distance fields can also serve as input to isosurface meshers that require well-defined inside/outside of objects, such as those available in the CGAL library [1]. This makes it possible to create quality anisotropic tetrahedral meshes for non-manifold geometry (Figure 9), suitable, say, for physically based simulation. The required number of parameters for such meshing is minimal in our method: we need a single offset surface parameter σ, and standard mesh quality (minimum angle, radius, etc.) parameters of the 3D tet mesher library. Our contributions include:

- an efficient approach to *both* define and compute a signed distance field for arbitrary input geometry,
- algorithm for removing nested interior isosurfaces,
- mathematical proof that the distance field of an offset surface equals to a shifted distance field of the original (non-manifold) surface,
- rapid global sign determination via S-shaped traversal.

2 Related work

Many efficient algorithms are available to compute unsigned and signed distance fields for *manifold watertight* meshes; see, for example, the survey [21]. To accelerate distance computation, hierarchical data structures can cull branches that cannot contain the shortest distance [17, 45]. Fuhrmann [16] computed distance fields by analyzing prisms emanating from each triangle along the normal direction. Similarly, the characteristic/scan conversion (CSC) method [28] computes the exact distance field up to some maximum value by scan converting distance fields of individual vertices, edges and faces. The [15, 43] accelerate the CSC method with an efficient GPU implementation. To speed up the hardware-assisted distance field computation, [47, 48] explores a Voronoi-based culling and clamping algorithm and [46] presents an interactive algorithm for surface distance maps with the affine transformation. Based on GPU, distance fields can also be computed on adaptive grids [8, 34], or on local narrow bands for complex geometry [12]. Fast marching method, a propagation method that numerically solves the Eikonal equation, updates the distance voxel-by-voxel with increasing distance [39, 40]. Linear computational complexity for the fast marching method has been demonstrated by [13, 49]. Similarly, jump flooding algorithms can compute approximate distances in constant time for an input set of seeds [36]. Several robust algorithms have been proposed to determine the sign for triangular *manifold* surfaces [2, 5, 43]. All of these algorithms, however, require the input mesh to be manifold and watertight for the sign to be defined. In our work, we address the problem of how to *define* the sign for non-manifold or non-closed geometry, and then rapidly compute the resulting signed distance field.

For "polygon soup" geometry, [32] determines the sign at each grid point with ray stabbing and voting. However, ray stabbing is a global operation and could assign incorrect signs in regions where the surface has high variation or self-overlaps. This problem can be partially overcome by observing that the grid points far from the surface exhibit less sign variation [44]. Similarly, [30] determines the sign by counting the intersections of voxel edges with the geometry, to successfully resolve self-intersections of closed manifold meshes. Robustness can also be improved using ray voting against ε-bands of the unsigned distance field [29]. Ray voting methods in general, however, have difficulties handling structured outliers in the geometry. Space can also be partitioned into polyhedral regions and then the interior solid region is determined based on region adjacency [31]. However, unlike our work, this method does not support open objects (shells) or intersecting solid objects.

Another approach is to create manifold surface with surface repair methods which try to resolve artifacts of the input surface meshes. The mesh-based methods repair the input surface by removing gaps [11, 35] and filling holes [24, 25, 41]. In practice, however, the input geometry may consist of several (non-manifold) surfaces that intersect or are in close proximity, as is the case with mechanical components (landing gear), or 3D characters (turtle). In such cases, it is difficult to apply the above geometric mesh-repair mechanisms to extract a single well-defined manifold mesh, requiring user interaction to resolve ambiguities. Volume-based methods usually can resolve more general complex configurations, but they usually also introduce distortion in trouble-free parts [10, 22]. The input mesh can be modified only locally within the neighborhood of undesired configuration [4]. CAD models can be repaired while avoiding the global sampling problems of the volume base method [9]. This method, however, requires the input geometry to consist of triangular manifold meshes; we make no such assumption. Similarly, the method of [42] requires the input triangle orientation to be consistent; otherwise, suboptimal results are produced (see Section 3.3 in [42]).

Several methods have been proposed to reconstruct surfaces from implicit functions [19, 42, 50]. Robustness to noisy data [23] and unoriented data [3] has been improved. Implicit functions are commonly used to track fluid surfaces [7, 30]. The goal of these methods is accurate reconstruction of surfaces, and therefore these methods generally employ implicit functions in narrow bands around the surface. In our work, we compute global signed distance fields, both inside and outside the object, sampled on regular

3D grids. Global signed distance fields can be computed directly from isosurfaces of unsigned distance fields [6]. We greatly accelerate such global signed distance field computation by proving that the distance field can be re-used in the exterior region. We also give an efficient algorithm to remove interior nested isosurface components (Section 3.1), as well as an S-shaped traversal algorithm (Section 3.3) to rapidly determine the sign globally, with a minimal number of pseudonormal tests.

Given a distance field, the marching cubes algorithm [26, 27], or *dual* methods [38] can robustly extract a closed, manifold and intersection-free triangulated surface. Unless the distance field is signed, this surface will contain interior components, which may not be desirable. The resulting mesh is also typically high resolution and follows a regular pattern. If such a surface is used to create a 3D tetrahedral mesh directly (e.g., using TetGen [18]), one typically obtains a highly detailed tet mesh whose resolution cannot be easily adjusted. Our signed distance fields can serve as input to anisotropic tet meshers that mesh the volume enclosed by an isosurface, such as those implemented in the CGAL [1] library. This makes it possible to create "cage" tet meshes enclosing input *non-manifold* geometry, useful, e.g., for level-of-detail simulation, or the multigrid method [33].

Recently, Jacobson et. al [20] presented a robust approach to compute tetrahedral meshes for "polygon soup" input geometry, using generalized winding numbers and constrained Delaunay triangulation. Although the method is robust to many polygon soup geometries and can preserve the input mesh, it requires consistent orientation of the input triangles whereas our method does not. Duplicated geometry will result in incorrect winding numbers and may cause ambiguities. The most important difference, however, is that the winding number field is *not* a signed distance function to any particular geometry. Multiplying an unsigned distance field with the winding number (or its sign) does not give a signed distance field. The resulting function is not even continuous: it has, for example, a discontinuity when leaving the open container in Figure 7 (b). As such, this method does not address our problem (signed distance field computation). Our implicit functions are *signed distance fields* with respect to some meaningful geometry (the isosurface), with all the resulting benefits: continuity, unit gradient, and the field absolute values are exact distances to some meaningful geometry (the isosurface). We compare our method to [20] in Section 4.

3 Computing the Signed Distance Field

Given the input "triangle soup" geometry Ω, a box where the distance field is to be computed, and grid resolutions in x, y, z, we first compute the unsigned distance field. Our implementation is accelerated using an octree and multiple cores for exact computation, and uses the fast marching method for approximate distance field computation. In exact computation, we compute exact distance to the nearest triangle. The octree in the exact method only serves to accelerate the distance queries. The octree and fast marching are orthogonal to our method; we apply them equally to unsigned and signed distance field computation.

3.1 Offset surface and removal of interior components

We proceed by defining an offset surface of the unsigned distance field

$$S_\sigma = \left\{ X \in \mathbb{R}^3 \,|\, d_U(X,\Omega) = \sigma \right\}, \tag{1}$$

where the function $d_U(X,\Omega)$ returns the unsigned distance from point X to geometry Ω, and $\sigma \geq 0$ is an offset. Typically, we set $\sigma = 3h$, where h is the distance field grid spacing, but other values can be chosen to preserve or remove local detail. We then extract a triangular mesh of S_σ, by applying the marching cube algorithm [26] to the unsigned distance field. This algorithm is guaranteed to generate a manifold offset surface for any input. We observe that the offset surface in general consists of several *disjoint* connected components, which may be nested:

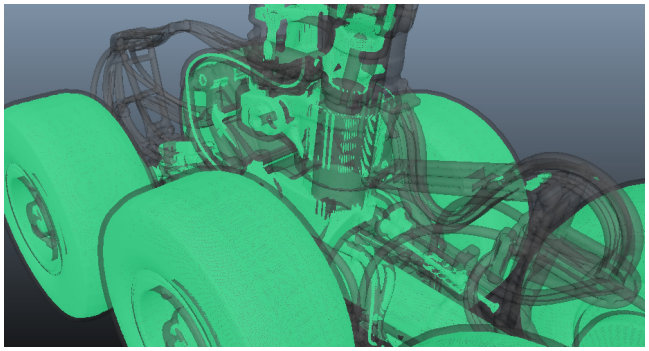


Figure 3: Offset surface S_σ for the Boeing 777 landing gear. Interior components (848 total) are shown green.

$$S_\sigma = \left(\amalg_i E^i_\sigma\right) \amalg \left(\amalg_i I^i_\sigma\right), \tag{2}$$

where E^i_σ are connected components that are not enclosed by any bigger components (*exterior* components), and I^i_σ are all the other components (*interior* components). Note that $\amalg$ denotes the "disjoint union", i.e., set union with the additional understanding that the operands are disjoint. The marching cube algorithm will create both the E^i_σ and I^i_σ surfaces (Figure 3), and we need to remove the latter. We first detect all the connected components of S_σ using a union-find datastructure. For each component, we then efficiently determine if it is of type E or I, as follows. First, compute a tight bounding box for each component, and sort them according to their volumes. Starting from the component with the smallest bounding box volume, we check it against all the other components with a larger bounding box volume (starting from the largest one). If the smaller bounding box is not totally enclosed by the bigger box, there exists a point on the smaller component that is outside the bigger component, and the two components are therefore not nested. Otherwise, we pick a random vertex on the smaller component, and test if it is inside the bigger component. We do so by finding the nearest site on the larger mesh and then perform the pseudonormal test [5]. Testing a single vertex is sufficient because the components are disjoint; they are either nested or enclose disjoint volumes.

Once a surface is known to be of type I, we can remove it from all future pairs. At the end of this process, all surfaces of type E have been identified, and we can define

$$E_\sigma = \amalg_i E^i_\sigma. \tag{3}$$

The time to compute E_σ was small compared to the signed distance field computation: under 2 minutes in our most complex example (landing gear), and much shorter in the other examples.

We note that the exterior component E_σ could be found using a voxel flood-fill from the bounding box boundary. However, such an approach slows down with increasing resolution. Also, due to geometric detail close to the "surface", multiple components of S_σ may intersect the same surface voxel (landing gear), so decomposition into connected components and determination which are interior is still needed. Our proposed method is very fast: for each component, only a single vertex must be tested (single pseudonormal test).

3.2 Unsigned distance field re-use

Our ultimate goal is to compute the signed distance field for the input mesh Ω. Currently, we have the manifold exterior offset surface

	#tri	resolution	unsigned field	memory	#isosurface tri	#interior components	naive signed field	signed field
excavator bucket	12,825	1024^3	12 min	26.4 GB	4,826,772	4	576 min	9 min
dragon	871,414	1024^3	103 min	27.0 GB	2,532,564	27	354 min	7 min
skeleton	379,184	1024^3	86 min	14.1 GB	812,572	119	116 min	1 min
landing gear	1,847,976	1024^3	53 min	29.0 GB	4,167,414	848	290 min	11 min
turtle	3,654	1024^3	6 min	24.9 GB	3,145,624	268	393 min	11 min

Table 1: **Distance field computation performance.** "Naive signed field" refers to computing the signed field directly from E_σ [Barbič and James 2008], whereas "signed field" refers to our method, which can be seen to be significantly faster. Time to construct E_σ is included. All computations use eight cores.

	#tri	resolution	unsigned field	memory	#isosurface tri	#interior components	naive signed field	signed field
excavator bucket	12,825	1024^3	68.5 min	29.1 GB	4,857,930	73	94.5 min	8.1 min
dragon	871,414	1024^3	85.1 min	30.0 GB	2,597,974	37	89 min	4.6 min
skeleton	379,184	1024^3	81.5 min	29.1 GB	844,394	138	81.9 min	1.7 min
landing gear	1,847,976	1024^3	73 min	30.4 GB	4,199,760	746	90.1 min	8 min
turtle	3,654	1024^3	83 min	29.1 GB	2,930,126	272	96 min	4.9 min

Table 2: **Distance field computation performance with the fast marching method.** The time to initialize marching via exact distance field computation in a narrow band around E_σ is included. All computations use a single core.

E_σ, and the unsigned distance field for Ω. One could now compute a signed distance field for E_σ, using an octree and the pseudonormal test [5], or using fast marching [39]. However, we observed that such an approach is computationally slow in practice, significantly slower than the unsigned computation (see Tables 1 and 2, "naive signed field"). Instead, we prove the following lemma.

Lemma 3.1 *If X is a point in the* exterior *region of E_σ, then the signed distance d_S equals*

$$d_S(X,E_\sigma) = d_U(X,\Omega) - \sigma \geq 0. \tag{4}$$

Let Z and P' be the closest sites to X on surfaces E_σ and Ω, respectively. Let P be the closest site to Z on Ω. Because E_σ is manifold, XP' intersects E_σ; let Z' be any intersection point.

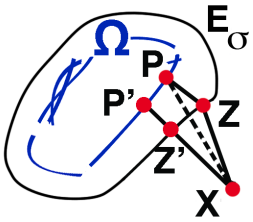


Figure 4: Proof illustration

Proof: Triangle inequality gives

$$\begin{aligned} d_S(X,E_\sigma) &\geq |XP| - |ZP| \\ &\geq d_U(X,\Omega) - \sigma. \end{aligned} \tag{5}$$

Suppose $d_S(X,E_\sigma) > d_U(X,\Omega) - \sigma$. Then we have

$$|XZ| > |XP'| - |ZP| = |XZ'| + (|Z'P'| - |ZP|). \tag{6}$$

Because $Z' \in E_\sigma$, we have $|Z'P'| \geq \sigma = |ZP|$. Therefore, $|XZ| > |XZ'|$, contradicting that Z is closest to X on E_σ. ■

Therefore, in the exterior region of E_σ, we can simply re-use our previously computed unsigned distance field $d_U(X,\Omega)$, leading to large computational savings (two orders of magnitude, see Tables 1 and 2, "signed field").

We also tried extending Equation 4 into the interior of E_σ (or at least into a narrow band on the interior side of E_σ), but discovered a counter-example. Figure 5 gives a counter-example where $\sigma > 0$ and $0 < a < 2\sigma$ are arbitrary, and $\varepsilon = \sqrt{\sigma^2 - a^2/4}$. The point X can be made arbitrarily close to E_σ ($\varepsilon > 0$ can be made arbitrarily small), yet $d_S(X,E_\sigma) \neq d_U(X,\Omega) - \sigma$. This example also rules out another seemingly "intuitive" equality, as we have $|d_S(X,E_\sigma)| \neq d_U(X,\Omega) + \sigma$.

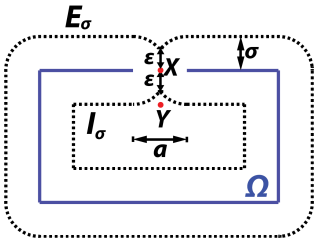
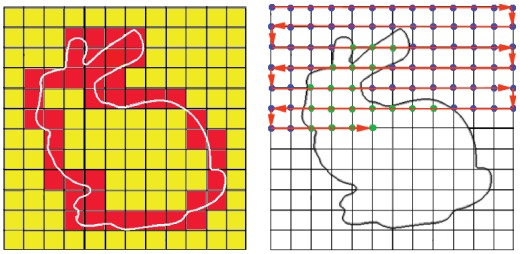


Figure 5: Counter-example



(a) Boundary voxels (b) S-shaped traversal

Figure 6: **Fast inside/outside determination using boundary voxels.** Left: voxelized mesh and the boundary voxels (red). Right: "snake"-like traversal.

Therefore, we cannot re-use the unsigned field in the interior of E_σ, and must recompute it. In determining the sign, we can, however, use the following lemma to accelerate the computation.

Lemma 3.2 *If a point X satisfies $d_U(X,\Omega) < \sigma$, then X must be in the interior of E_σ.*

Proof: Suppose X is exterior to E_σ. Then by Lemma 3.1, we have

$$d_S(X,E_\sigma) = d_U(X,\Omega) - \sigma < 0, \tag{7}$$

which contradicts the assumption that X is exterior to E_σ. ■

Lemma 3.2 makes it possible to avoid a pseudonormal test for any grid point with $d_U(X,\Omega) < \sigma$, which applies to approximately half of the boundary voxel grid points during the S-shaped traversal (Section 3.3). Note that the pseudonormal test can be replaced with any other test to determine the sign, e.g. using winding number [20]; this is orthogonal to our method. In any case, Lemma 3.2 makes it possible to decrease the number of sign computations. We also note that the converse of Lemma 3.2 is not true. The counter-example is shown in Figure 5, where $d_U(Y,\Omega) \geq \sigma$, but Y is in the interior of E_σ.

3.3 Sign determination and interior distance field

Our remaining task is to compute the signed distance field in the interior region of E_σ. We first efficiently compute the sign for all the grid points. To do so, we first voxelize E_σ, and tag all the distance field voxels which intersect E_σ as *boundary* (Figure 6, left). We then compute unsigned distances and closest features (vertex, edge, or face) on E_σ for all the vertices of the boundary voxels. For exact distance field computation, we do so by computing an octree to E_σ, and then use it to compute distances and closest features. For

fast marching, we compute the exact distance not just for boundary voxel vertices, but also for vertices of neighboring voxels (call this set of vertices $\mathcal{I}$), so that marching (below) can be properly initialized. We do so by traversing all the triangles of E_σ. To ensure correct 3D instances everywhere on $\mathcal{I}$, we "rasterize" [28] the distance function of each triangle to the vertices of intersecting voxels, their neighbors and neighbor's neighbors, storing the minimum distance.

Next, we perform a S-shaped traversal (Figure 6, right) over all the grid points to determine the sign. For multicore computation, we divide the grid into distinct (and equal in size, modulo #cores) slices, based on the z-coordinate, and process each slice on an individual core. During the traversal, we maintain a boolean flag that corresponds to whether the currently visited grid point is inside or outside of E_σ. The traversal starts at the distance field box corner, and the flag is initialized to *outside*. When a grid point is visited, we need to update the flag. However, if the grid point does not belong to a boundary voxel, the line segment joining the previous and current grid point cannot intersect E_σ, otherwise, the current grid point would belong to a boundary voxel. Therefore, in this case, we can keep the old flag value (whether *outside* or *inside*), without explicitly performing an inside-outside test. If the grid point belongs to a boundary voxel, we first check the condition $d_U(X,\Omega) < \sigma$ of Lemma 3.2. If satisfied, the grid point is in the interior of E_σ. Otherwise, we perform the inside/outside test by using the previously computed nearest site on E_σ and the pseudonormal test [5]. We update the boolean inside/outside flag accordingly.

At the end of the S-shaped traversal, the sign is known for all the grid points. Finally, we compute the distances for interior points. For fast marching, we do so by initializing the marching using the distances and sign computed in the narrow band in the first step above, and then march into the interior. For exact distance fields, we perform another S-shaped traversal, skipping the exterior points and computing the distances for the interior points using the octree. S-shaped traversal order is beneficial so that we can use the triangle inequality to provide a good initial upper bound on the distance for the octree traversal, $|d_S(X+H,E_\sigma) - d_S(X,E_\sigma)| \leq |H|$, where $H \in \mathbb{R}^3$ is an arbitrary vector. In our S-shaped traversal, X and $X+H$ are adjacent grid points.

After we compute the signed distance field for E_σ, we can offset it by $-\sigma$, producing the final signed distance field. We note that this last offsetting step is optional (Figure 10), and works best when the input non-manifold surface is intended to approximate a closed mesh. If the input surface is a non-manifold shell, E_σ enlarges it into a volume with a manifold boundary; this is useful, e.g., to define a collision volume with well-defined repulsive normals.

4 Results

Our experiments were performed on an Intel Xeon 2.9 GHz CPU (2x8 cores) machine with 32GB RAM, and an GeForce GTX 680 graphics card with 2GB RAM. Table 1 gives the performance of our signed distance field computation on five non-manifold meshes. All exact signed distance field examples use eight cores for both unsigned and signed distance fields, whereas fast marching uses a single core. Figure 1 shows the distance field results for the five models. The Boeing 777 landing gear, the turtle and the skeleton have self-intersections in the input mesh. The triangles of the turtle and skeleton were not consistently oriented in these meshes which we downloaded from the Internet, which is not a problem for our method as consistent input mesh orientation is not required. Our distance field plausibly resolves non-manifold geometric detail and thin features. Table 3 analyzes scalability under increasing resolutions. In Figure 11, we give an example illustrating a plausible signed distance field computed for non-manifold input geometry.

We have applied our distance field computation algorithm to collision detection and tetrahedral mesh computation. Figure 8 illustrates collision detection performed by testing one object's

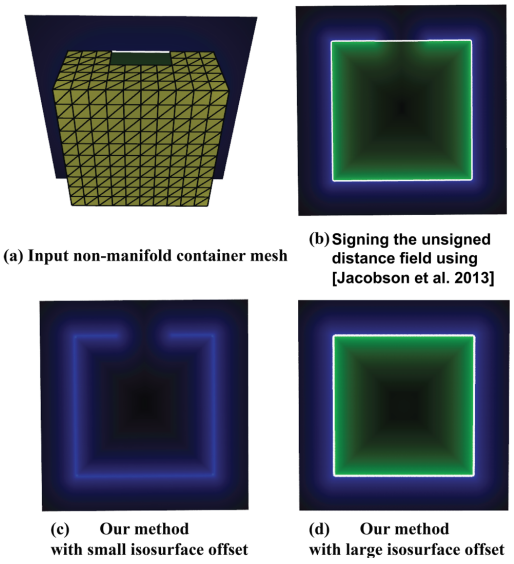
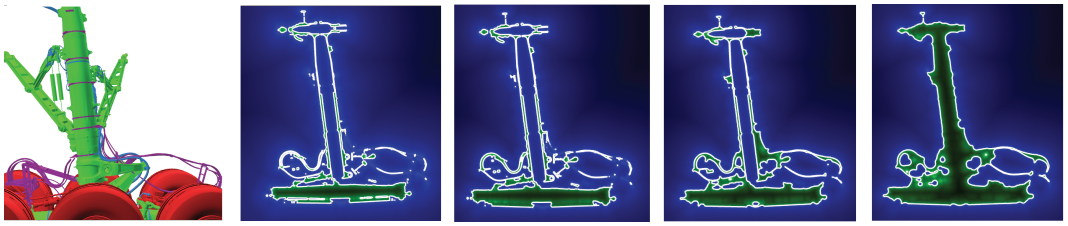


Figure 7: **Comparison to generalized winding numbers.** Input mesh (a) is a box with a square hole cut from the top face. The field obtained by multiplying the unsigned distance field by the sign of the thresholded generalized winding number [20] is not continuous across the opening of the box (b). We can treat the box as open space by setting the offset surface value smaller than half of the width of the hole (c), or closed space otherwise (d).

pointshell against the signed distance field of the other object [6]. Figure 9 demonstrates the tetrahedral mesh computed for the turtle model, by using our signed distance field as the input to the anisotropic mesher in CGAL [1]. As commonly done in computer graphics, the original non-manifold triangle mesh geometry can then be animated by performing a FEM deformable object simulation on the tetrahedral mesh.

We compare our signed function to a modified algorithm of [20] in Figure 7. The input mesh is a 3D box with a square hole cut from its top face. The size of the hole a is adjustable. Jacobson et. al [20] computes the inside-outside segmentation based on a graph-cut over the generalized winding number, which here is greater than 0.5 everywhere inside the box, regardless of a. Therefore, the entire box is assigned the interior sign. Our method, however, treats the gap a as either a genuine gap or an artifact, depending on the value of the isosurface input parameter σ relative to a. If σ is small, our method will treat the gap as genuine and the box as open; if σ is large, the box will be closed. Our method therefore provides control (via parameter σ) over what features are considered too small and can be neglected, versus features that are above the "engineering tolerance" σ. It is not easily possible to convert the winding number field into a signed distance field. Because the sign of the thresholded generalized winding number is not continuous, the implicit function obtained by signing the unsigned distance field with the sign of the thresholded generalized winding number is not continuous (Figure 7, b). Similarly, multiplying the winding number field with the unsigned distance field, or using the winding field directly, gives an implicit function that does not have unit gradient and whose values are not distances to some geometry; it is therefore not a signed distance function.

Performance for exact signed distance fields computed using the octree is provided in Table 1. Our method is not limited to octree-based distance field accelerations, but can be used with any uniform-grid distance field computation method. Regardless of what algorithm is used to compute the unsigned distance field, we can always use Equation 4 to avoid recomputing the distance field outside of the manifold offset surface. This speedup also applies



(a) Input landing gear mesh (b) Distance field with σ=2h (c) Distance field with σ=3h (d) Distance field with σ=7h (e) Distance field with σ=10h

Figure 10: Signed distance fields obtained under different offsets σ, followed by a shift-back by $-\sigma$. It can be seen that the method progressively treats larger voids as solid as σ is increased ("engineering tolerance"). In (b) and (c), observe that the walls of the main vertical landing gear support structure (green in (a)) remain equally thin. In (e), σ is so large that the entire interior of the hollow support structure is filled.

resolution	time (unsigned)	time (signed)
128x128x128	2.25 min	0.05 min
256x256x256	5.42 min	0.15 min
512x512x512	22 min	1.5 min
1024x1024x1024	105 min	9 min

Table 3: Computation times (dragon) vs distance field resolution.

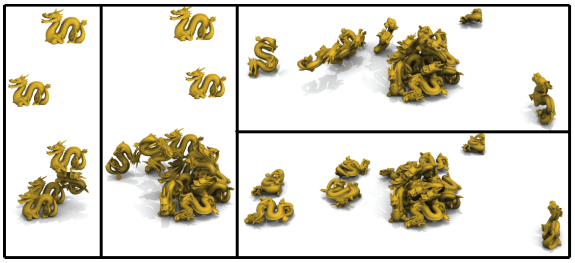


Figure 8: **Collision detection using the computed signed distance fields.** Sixteen non-manifold dragons falling onto the ground. Distance field resolution is 1024x1024x1024.

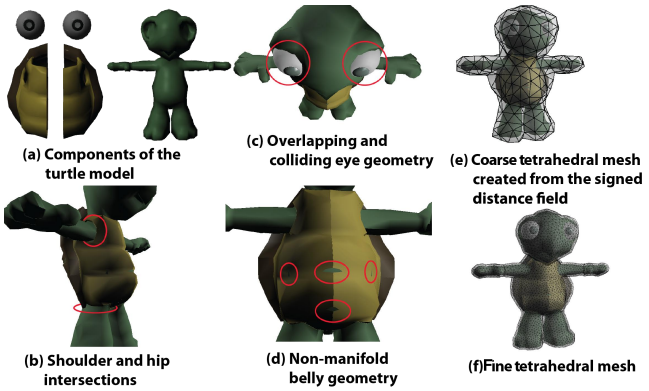


Figure 9: Tetrahedral meshes created from a signed distance field computed from non-manifold self-intersecting geometry.

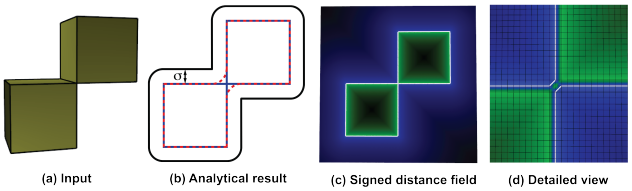


Figure 11: **Signed distance field for non-manifold input.** (a) Two boxes share an edge. (b) Blue: input geometry. Black: analytical offset surface at offset σ. Dashed red: manifold surface obtained by shifting the black offset surface by $-\sigma$ (our result). It can be seen that this surface matches input geometry, but regularizes it in the vicinity of the non-manifold "pinch" geometry. (c) and (d): Computed signed distance field (resolution is $256 \times 256 \times 256$).

to approximate distance field computation methods such as vector distance transform [37] and fast marching transform [40]. Table 2 provides computation times for fast marching with second-order finite difference approximations to the partial derivatives, computed using the publicly available implementation of [14]. The marching times are dominated by maintaining a priority queue of grid points, largely depend on grid resolution, and are less dependent on input geometry complexity than the octree exact implementation (see "unsigned field" column in Tables 1 and 2). Our method yields a substantial speedup in the computation of the signed field, both for exact computation and fast marching, because we can avoid traversing the exterior space of E_σ. Parameter σ can be used to provide a cut-off for the geometric size that is deemed significant (Figure 10).

5 Conclusion

We presented a simple and robust approach to define and compute a signed distance field for non-manifold input geometry. Our approach is compatible with any uniform-grid distance field computation method. Compatibility with adaptive-grid distance field method needs future investigation. Our distance field is accurate with respect to the offset isosurface. The interior signed distance field is computed from the polygonal offset surface computed using marching cubes, which is an approximation to the analytical offset surface, and may introduce a small amount of discretization error in the distance field. Our method has a single parameter: the isosurface σ value. Small and large values of σ will result in loss of geometric detail. If an automated choice of σ is desired, $\sigma = 3h$ produced good results in practice. A single global parameter σ may fuse geometrically close parts, despite them being semantically distant. We resolve the problem with careful σ tuning (see legs of turtle in Figure 1(e)), which fortunately is very fast in practice since the isosurface can be recomputed and visualized rapidly before computing the signed distance field. In our work, we remove the interior components I_i, but if the intent is to model hollow objects, interior components can be kept. Only a trivial change is required: perform pseudonormal tests at voxels intersecting interior geometry. Adaptive σ based on the local features of the geometry could be useful future work. We applied our method to polygonal input; but our method could also be applied to more general input, such as point clouds, polygonal lines or parametric curves.

Acknowledgments: This research was sponsored in part by the National Science Foundation (CAREER-53-4509-6600), USC Annenberg Graduate Fellowship to Hongyi Xu, and a donation of two workstations by the Intel Corporation. We thank the Boeing Company for the landing gear model.

REFERENCES

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] H. Aanæs and J. A. Bærentzen. Pseudo-normals for signed distance computation. In *Proc. of Vision, Modeling and Visualization*, 2003.

[3] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proc. of Eurographics Symp. on Geometry Processing*, pages 39–48, 2007.

[4] M. Attene. A lightweight approach to repairing digitized polygon meshes. *Vis. Comput.*, 26(11):1393–1406, Nov. 2010.

[5] J. Bærentzen and H. Aanæs. Signed distance computation using the angle weighted pseudo-normal. *IEEE Trans. on Visualization and Computer Graphics*, 11(3):243–253, 2005.

[6] J. Barbič and D. L. James. Six-dof haptic rendering of contact between geometrically complex reduced deformable models. *IEEE Transactions on Haptics*, 1(1):39–52, 2008.

[7] A. W. Bargteil, T. G. Goktekin, J. F. O'Brien, and J. A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25(1), 2006.

[8] T. Bastos and W. Celes. Gpu-accelerated adaptively sampled distance fields. In *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on*, pages 171–178, 2008.

[9] S. Bischoff and L. Kobbelt. Structure preserving cad model repair. In *Computer Graphics Forum*, volume 24, pages 527–536. Wiley Online Library, 2005.

[10] S. Bischoff, D. Pavic, and L. Kobbelt. Automatic restoration of polygon models. *ACM Trans. Graph.*, 24(4):1332–1352, Oct. 2005.

[11] P. Borodin, M. Novotni, and R. Klein. Progressive gap closing for mesh repairing. In *Advances in Modelling, Animation and Rendering*, pages 201–213, 2002.

[12] B. Chang, D. Cha, and I. Ihm. Computing local signed distance fields for large polygonal models. *Computer Graphics Forum*, 27(3):799–806, 2008.

[13] F. Chen and Y. Zhao. Distance field transform with an adaptive iteration method. In *IEEE Intl. Conf. on Shape Modeling and Applications*, pages 111–118, 2009.

[14] K. T. Chu and M. Prodanovic. Lsmlib. http://ktchu.serendipityresearch.org/software/lsmlib/.

[15] K. Erleben and H. Dohlmann. Signed distance fields using single-pass gpu scan conversion of tetrahedra. In *GPU Gems 3*, pages 741–763. Addison-Wesley, 2008.

[16] A. Fuhrmann, G. Sobotka, and C. Groß. Distance fields for rapid collision detection in physically based modeling. In *Proc. of GraphiCon 2003*, pages 58–65, 2003.

[17] A. Guezlec. Meshsweeper: dynamic point-to-polygonal mesh distance and applications. *IEEE Trans. on Visualization and Computer Graphics*, 7(1):47–61, 2001.

[18] Hang Si. TetGen: A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator, 2011.

[19] H. Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, Department of Comp. Science and Engineering, University of Washington, 1994.

[20] A. Jacobson, L. Kavan, , and O. Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics*, 32(4):33:1–33:12, 2013.

[21] M. Jones, J. Bærentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *IEEE Trans. on Visualization and Computer Graphics*, 12(4):581–599, 2006.

[22] T. Ju. Robust repair of polygonal models. *ACM Trans. Graph.*, 23(3):888–895, 2004.

[23] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proc. of Eurographics Symp. on Geometry processing*, 2006.

[24] A. Kumar, A. Shih, Y. Ito, D. Ross, and B. Soni. A hole-filling algorithm using non-uniform rational b-splines. In *Proc. of the 16th Intl. Meshing Roundtable*, pages 169–182, 2008.

[25] B. Lévy. Dual domain extrapolation. In *Proc. of ACM SIGGRAPH 2004*, pages 364–369, 2003.

[26] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of Marching Cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[27] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (Proc. of ACM SIGGRAPH 87)*, 21(4):163–169, 1987.

[28] S. Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Inst. of Technology, 2003.

[29] P. Mullen, F. De Goes, M. Desbrun, D. Cohen-Steiner, and P. Alliez. Signing the unsigned: Robust surface reconstruction from raw pointsets. In *Computer Graphics Forum*, volume 29, pages 1733–1741, 2010.

[30] M. Müller. Fast and robust tracking of fluid surfaces. In *Symp. on Computer Animation (SCA)*, pages 237–245, New York, NY, USA, 2009. ACM.

[31] T. M. Murali and T. A. Funkhouser. Consistent solid and boundary representations from arbitrary polygonal data. In *Proc. of Symposium on Interactive 3D Graphics*, pages 155–ff., 1997.

[32] F. S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *Visualization and Computer Graphics, IEEE Transactions on*, 9(2):191–205, 2003.

[33] M. A. Otaduy, D. Germann, S. Redon, and M. Gross. Adaptive Deformations with Fast Tight Bounds. In *Symp. on Computer Animation (SCA)*, pages 181–190, Aug. 2007.

[34] T. Park, S.-H. Lee, J.-H. Kim, and C.-H. Kim. Cuda-based signed distance field calculation for adaptive grids. In *IEEE Intl. Conf. on Computer and Information Technology (CIT)*, pages 1202–1206, 2010.

[35] P. Patel, D. Marcum, and M. Remotigue. Stitching and filling: Creating conformal faceted geometry. In *Proc. of the 14th Intl. Meshing Roundtable*, pages 239–256, 2005.

[36] G. Rong and T.-S. Tan. Variants of jump flooding algorithm for computing discrete voronoi diagrams. In *Intl. Symp. on Voronoi Diagrams in Science and Engineering*, pages 176–181, 2007.

[37] R. Satherley and M. W. Jones. Vector-city vector distance transform. *Computer Vision and Image Understanding*, 82(3):238 – 254, 2001.

[38] S. Schaefer, T. Ju, and J. Warren. Manifold dual contouring. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):610–619, 2007.

[39] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[40] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, volume 3. Cambridge university press, 1999.

[41] A. Sharf, M. Alexa, and D. Cohen-Or. Context-based surface completion. In *Proc. of ACM SIGGRAPH 2004*, pages 878–887, 2004.

[42] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *Proc. of ACM SIGGRAPH 2004*, pages 896–904, 2004.

[43] C. Sigg, R. Peikert, and M. Gross. Signed Distance Transform using Graphics Hardware. In *Proc. of IEEE Visualization Conference*, pages 83–90, 2003.

[44] J. Spillmann, M. Wagner, and M. Teschner. Robust tetrahedral meshing of triangle soups. In *Symp. on Computer Animation (SCA), Posters and Demos*, pages 1–2, 2006.

[45] J. Strain. Fast tree-based redistancing for level set computations. *J. Comput. Phys.*, 152(2):664–686, 1999.

[46] A. Sud, N. Govindaraju, R. Gayle, E. Andersen, and D. Manocha. Surface distance maps. In *Proc. of Graphics Interface*, pages 35–42. ACM, 2007.

[47] A. Sud, N. Govindaraju, R. Gayle, and D. Manocha. Interactive 3D Distance Field Computation using Linear Factorization. In *Proc. ACM Symp. on Interactive 3D Graphics and Games (I3D)*, 2006.

[48] A. Sud, M. Otaduy, and D. Manocha. DiFi: Fast 3D Distance Field Computation Using Graphics Hardware. *Comp. Graphics Forum*, 23(3):557–556, 2004.

[49] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005.

[50] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proc. of IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 194–201, 2001.