A Comparison of Sampling Strategies for Computing General Sweeps☆

Hüseyin Erdim[a], Horea T. Ilieş[*,a]

[a]*Computational Design Laboratory, Department of Mechanical Engineering, University of Connecticut, Storrs, CT 06269-3139*

Abstract

Sweeping moving objects has become one of the basic geometric operations used in engineering design, analysis and physical simulation. Despite their relevance, computing the boundary of the set swept by a non-polyhedral moving object is largely an open problem due to well known theoretical and computational difficulties of the envelopes.

We have recently introduced a generic point membership classification (PMC) test for general solid sweeping. Importantly, this PMC test provides complete geometric information about the set swept by the moving object, including the ability to compute the self-intersections of the sweep itself. In this paper, we compare two recursive strategies for sampling points of the space in which the object moves, and show that the sampling based on a fast marching cubes algorithm possesses the best combination of features in terms of performance and accuracy for the boundary evaluation of general sweeps. Furthermore, we show that the PMC test can be used as the foundation of a generic sweep boundary evaluator in conjunction with efficient space sampling strategies for solids of arbitrary complexity undergoing affine motions.

Key words: Solid sweeping, boundary evaluation, space sampling, marching cubes, octrees.

1. Introduction

Solid sweeping is an essential concept in many applications involving moving shapes, including motion planning, collision detection, ergonomics, robot workspace analysis and NC machining, and it is considered to be one of the fundamental solid representation schemes in solid modeling [1]. The swept volume is defined as the subset of space occupied by a moving object (i.e., the generator) as it moves in that space. It is well known that the swept volume is bounded by a subset of the envelopes to the family of shapes formed by the generator during the motion, and that points of these envelopes are solutions to specific differential equations [2]. The mathematical foundations of envelopes clearly imply that computing the sweep of a moving set of points[1] in space is, in its most general form, a hard problem. Despite the fact that these foundations seem to be well understood, the validity and computational properties of the envelopes (and, as a result, those of sweeps) are not. Consequently, since these envelopes can not be generated in closed form for any case of reasonable complexity, a variety of algorithms focus on restricted classes of shapes and motions for which envelope surfaces (or certain approximations) can be computed, as discussed in section 1.1.

The typical approaches to computing the boundary of the set swept by a moving solid shape follow a procedure that is similar to the usual "generate and test" boundary evaluation [3]. The first step focuses on generating the candidate surfaces bounding the swept set. This is a difficult undertaking for arbitrary shapes and motions since the envelopes are generated by the so-called "grazing points", which are the boundary points of the generator that are tangent to the envelopes at *a given* configuration during the motion. The difficulties come from the fact that the geometry and topology of the set of grazing points change during the motion as discussed below. Nevertheless, a successful generation of candidate surfaces must be followed by a trimming step in which the candidate surfaces are intersected to produce a set of candidate faces. These faces are then evaluated against the boundary of the sweep, and those candidate faces that do not belong to the boundary are eliminated. Note that such an evaluation requires a point membership classification (PMC) test[2], but a generic PMC test for solid sweeps did not exist until recently.

In [4] we introduced the first generic PMC test for sweeping solids of arbitrary complexity moving according

☆To appear in Computer Aided Design, accepted June 8, 2009.

*Corresponding author.

Email addresses: herdim@engr.uconn.edu (Hüseyin Erdim), ilies@engr.uconn.edu (Horea T. Ilieş)

[1]Throughout this paper, we refer to *point sets* simply as *sets* as long as this simplification in terminology does not cause ambiguity.

[2]For a given geometric representation, the PMC test provides the ability to classify a point against a point set X as "in", "on" or "out" of X. Such a PMC test provides complete geometric information about set X in the sense that any geometric property of X can, in principle, be computed [1].

to a one-parameter affine motion. This test, whose main features are summarized in section 2, provides a mechanism to classify *any* point of the space against the set swept by a moving *solid* generator, including those points that are "on" the boundary of the sweep, or points of the internal envelopes that are not part of the sweep boundary. Importantly, this PMC test relies on curve-solid intersections against the stationary generator, can be implemented in *any* geometric representation that supports curve-solid (or curve-surface) intersections, and is well-suited to hardware accelerated computations as well as parallelization (see [5] for example). Moreover, our test defines a decomposition of space that is finer than the typical ternary decomposition of the usual PMC test for r-sets [6], which can provide solutions to a host of other applications where sweeps play a critical role, such as contact analysis [7].

This paper is based on the premise that the PMC test described in [4] can be used in conjunction with efficient space sampling algorithms to perform boundary evaluation of sets swept by arbitrarily complex moving solid generators. Importantly, such an approach relies on: (1) efficient sampling algorithms, and (2) on robust curve-solid or curve-surface intersection algorithms extensively studied in the literature [8], and available in all geometric modeling kernels. Therefore, the class of problems that can be addressed by the proposed sampling-based sweep boundary evaluation is significantly larger than the current state of the art discussed below, and includes both rigid and possibly deformable complex solid geometry.

1.1. Prior Work

Sweep Boundary Evaluation

Many approaches to compute the boundary of swept volumes have been published in the literature, and a good review appears in [9]. Most of these approaches rely on simplifying assumptions on the geometry and motion of the generator in order to manage the difficulties mentioned above, which, in turn, limits the class of sweeps that they can handle. For example, most algorithms either restrict the type of the moving object to be polyhedral [10]; use simple translations, rotations, or screw motions and do not allow motions that produce self-intersections in the boundary of the sweep [11]; approximate the set swept by the moving object by a discrete union of instances of the object sampled along the motion (see [12] for a discussion); or use rendering methods to compute the image of the sweep without computing the complete representation of the sweep [13]. While some of these methods are relatively fast, they can handle only a subclass of problems of practical interest.

The SDE (Sweep Differential Equation) approach initially proposed in [14] and subsequently extended in [15], identifies the sweep with a first-order (linear) ordinary differential equation. The advantage of the SDE approach is that one can compute the grazing points only once at the initial position of the object, and the evolution of the grazing points is dictated by the accompanying differential equation, which must be solved numerically. One important limitation of this approach comes from the fact that it requires the moving solid to be represented implicitly, which means that other solid representations must first be implicitized. This, however, is a non-trivial step [16].

Discrete grazing points are computed at discrete (sampled) time intervals in [17] following a generate and test procedure that results in a set of points that are on the boundary of the sweep. This point cloud is input into a standard surface reconstruction procedure to construct the boundary surfaces of the sweep. In other words, the approach discussed in [17] discretizes the sweep both in time and space to obtain an approximation of the set swept by a moving object. Note that this approach also requires an implicit representation of the moving object.

It is worth mentioning that the last two methods described above address the most general class of problems among all published approaches to sweep boundary evaluation, but they do not provide details about the associated computational cost. One expects that the efficiency of boundary evaluation methods would rapidly decrease with the increase in the complexity of the class of problems (shapes and motions) being addressed. This is so because the algorithms that are designed for restricted classes of problems exploit as many of the properties of the class as possible to improve computational efficiency.

Space sampling approaches

Among the many sampling methods, brute force (grid-based) or Monte Carlo based approaches are trivial to implement, but inefficient as *sole* sampling strategies. There are many hierarchical data structures that can be used, in principle, to sample points more efficiently, including Binary Space Partitions (BSPs) [18], Bounding Volume Hierarchies (BVHs) and other space decompositions such as octrees [19], and marching cubes [20, 21, 22] (see also section 3).

Binary space partitions recursively divide an n dimensional space into (preferably convex) subsets by using hyperplanes. The data is stored in a BSP tree, which is essentially a binary tree for which each node has a *front* and *back* leaf. BSP trees are precomputed, can be traversed in linear time, and are used in many applications such as rendering, collision detection, and convex decompositions. BSPs are efficient in obtaining decompositions of *known* sets, but we do not know the set swept by the generator a priori.

A bounding volume hierarchy (BVH) is a tree of bounding volumes[3] such that the bounding volumes subdivide the subset of the space of interest. Due to their efficiency, BVHs have been adopted in many interactive simulations, including collision detection algorithms and deformable body simulations. The so called sphere trees detailed in

[3] A bounding volume of a set X is any set containing X.

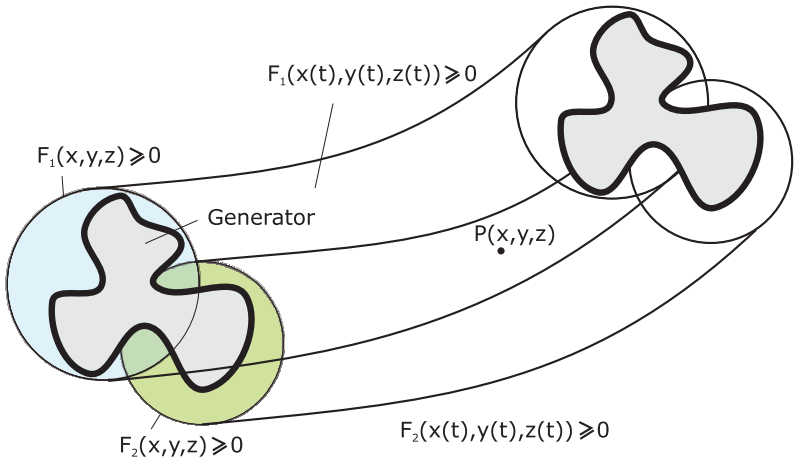


Figure 1: Spherical approximations can be used to reduce the number of points for which the inverted trajectories are computed and intersected with the generator.

[23] have been used to construct spherical approximations of shapes and have the advantage of using a simple, rotationally invariant shape as the primitive. This is appealing for our problem because spheres have simple implicit representations and the set swept by a sphere can, in principle, be represented implicitly as well. Thus, the PMC against the set swept by one of the spheres reduces to a simple function evaluation. If a point is in at least one of these sets swept by the spheres, then the point is in the union of these sets, and therefore the point is in the approximation of the swept set (as illustrated in Figure 1). Consequently, such a point is a candidate for the PMC outlined in section 2. Those points that do not belong to any of the swept sets can be discarded without further testing. The quality of the approximation increases with the number of spheres, but so does the number of required implicit function evaluations. Moreover, the computational cost of sampling points in the union of the sphere-swept volumes (or most other BVHs) is unknown. BVH algorithms tend to work well as long as the object being approximated is known.

On the other hand, octree/quadtree decompositions, which can be considered to be specialized variants of bounding volume hierarchies, only require the ability to perform a PMC test against the set that they approximate. The same is true for the marching cubes algorithm [20] and its variants discussed in more detail in section 3.2. However, the sampling schemes based on octrees and marching cubes possess some features that make them appealing as sampling approaches for performing sweep boundary evaluation. Specifically: (1) they are widely used, well understood, and easy to implement recursive procedures; (2) they only require an initialization cell (bounding volume for octrees and initial marching cell for marching cubes) and a PMC test during recursion; (3) each scheme adapts differently to the geometry of the set being approximated so their computational costs are expected to be different for the same accuracy; and (4) they both have efficient parallel versions of the corresponding decomposition algorithms.

Surface Reconstruction

The task of converting a given point cloud into surfaces is known as the surface reconstruction problem. Sampling the space followed by cell classifications (based on the PMC tests) as proposed in this paper outputs a cloud of points that are classified as "on" with respect to the boundary of the sweep or the fold regions generated by the moving object (see section 2). Consequently, performing a boundary evaluation of the point cloud requires capable surface reconstruction algorithms, which is an intense area of research, but is outside the scope of this paper.

There are many proposed techniques to reconstruct surfaces from point clouds, and a good review of the existing approaches can be found in [24]. Some methods produce a piecewise linear representation of the surface [20, 25, 26, 27, 28, 29], while others go one step further and construct piecewise polynomial surfaces [30, 31]. Subdivision surfaces could be constructed from point clouds [32, 33, 34], and there are new proposals for surface representations based on point-sets (so called "surfels") [35, 36, 37].

1.2. Scope and Outline

We explore the computational properties of two space decomposition algorithms, namely octree decomposition and marching cubes, that we employ as sampling strategies for performing the boundary evaluation of general solid sweeps. The performance and accuracy of these algorithms are benchmarked against a brute force sampling of known accuracy that visits all the cells of a prescribed grid. In this paper "accuracy" is used as a measure of *closeness* of

the approximation in terms of the number of partial cells in the decomposition[4] – see also section 3. We show that the sampling based on a fast marching cubes algorithm has substantially better performance and accuracy characteristics compared to those of the octree-based sampling for a similar quality of the approximation (i.e., similar number of partial cells in the decomposition).

To perform the boundary reconstruction, the sampled points are classified according to our PMC test summarized in section 2, and segmented into two point clouds: one for the sweep boundary points, and the other one for the fold boundary points (or the envelope points that are interior to the swept set). These clouds are then input into a standard surface reconstruction algorithm to extract the envelope surfaces that are either on the boundary of the sweep or interior to the set swept by the moving set. Note that the accuracy of the envelope points output by our approach is controlled by the density of the sampling (can be arbitrarily small up to machine precision) as well as by the accuracy of the curve-surface intersection algorithm being used. In this paper we use a commercial geometric kernel with known precision and proven robustness of geometric computations to perform the curve-solid intersections.

The paper is organized as follows. Section 2 summarizes the Point Membership Classification test that is used by the sampling strategies described in section 3 to identify envelope points of the sweep. The comparison of four sampling strategies, namely the brute force, octree decomposition and marching cubes with and without coherence, is discussed in section 4 based on several practical examples in both 2D and 3D. We conclude in section 5 by summarizing the main findings and the importance of this work.

2. PMC for Sweeping Solids

The approach that we take in this paper is to generate points in the same space as the moving object, classify these points according to the Point Membership Classification described below, which results in several different point clouds as explained below. As a proof of concept, we apply a standard surface reconstruction procedure to the corresponding point clouds.

Consider a 3D solid, i.e., the generator, moving according to a one-parameter affine motion M. A change in coordinate systems allows us to compute the *inverted motion* $\hat{M}$ that will describe how points of the space move *relative* to the generator. The inverted motion $\hat{M}(t)$ is the inverse of $M(t)$ for every value of $t = a$. In other words, each instantaneous transformation $M(a)$ has a unique inverse $\hat{M}(a)$ such that $x = \hat{M}(a)[M(a)x]$ for any point x of the space where motion takes place[5]. Thus, the trajectory $\hat{T}_x$ of an arbitrary point x of the space is the curve described by x according to the inverted motion $\hat{M}$.

It is not difficult to show that the inverted trajectory $\hat{T}_x$ contains **all** points y of the space that will pass through the given point x when y is moved according to motion M [39]. It follows that if the inverted trajectory of a point x does not intersect the generator in its original (static) configuration, then there are no points of the generator that will pass through point x of the space, and therefore point x *must* be outside of the set swept by the generator. On the other hand, if the inverted trajectory $\hat{T}_x$ intersects or is tangent to the generator, then *all* points of intersection and tangency will pass through x during motion M. The PMC test introduced in [4] is based on a careful examination of how this inverted trajectory curve intersects the generator to decide whether point x is "in" or "on" the set swept by the generator. Moreover, our test classifies the "in" points into

- *regular* interior points;
- *fold boundary* points that are on the internal envelopes or part of ∂S^0 or ∂S^1; and
- *fold* points, which are the interior points bounded by the fold boundary points.

Figure 2 illustrates the different types of points in our classification. Importantly, our decomposition of space into "in," "on," and "out" points described above is not only disjoint, but also complete (the union of these sets must be the universal set $\mathcal{W}$):

$$PMC(P, sweep(S, M)) = \begin{cases} in, & \text{if } P \text{ is a } \textit{regular}, \textit{fold}, \\ & \text{or } \textit{fold boundary} \text{ point}; \\ on, & \text{if } P \text{ is a } \textit{sweep} \\ & \textit{boundary} \text{ point}; \\ out, & \text{if } \hat{T}_P \cap S^0 = \emptyset. \end{cases} \quad (1)$$

This PMC test provides the ability to classify *every* point of the space relative to the set swept by the (solid) generator as it moves according to a prescribed motion M. Moreover, it can be implemented in any geometric representation that supports curve-solid intersection, including all commercial solid modeling kernels. If the solid is given as a boundary representation, the curve-solid intersection can be reduced to a curve-surface intersection.

3. Space Sampling and Surface Reconstruction

It is not difficult to see that one can use this PMC in conjunction with appropriate sampling algorithms to

[4]The other meaning of "accuracy" sometimes used in geometric modeling as a synonym of *correctness* of geometric computations as well as some of the accompanying computational problems are explored elsewhere [38] in a more general context.

[5]If $M(t)$ is given as a 4×4 matrix $A(t)$ in homogenous coordinates, the inverted motion $\hat{M}(t)$ is given by the inverse of this matrix, i.e., by $A^{-1}(t)$, for every value of t.

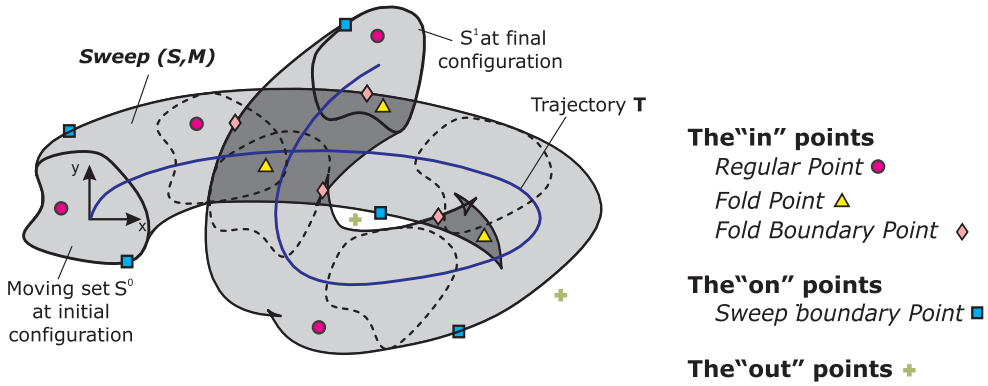


Figure 2: The sweep of a set S according to a rigid body motion consisting of a translation along trajectory T and rotation by $\pi/2$.

generate points of the space that fall into each category defined by the PMC test with any desired level of accuracy up to machine precision. Consequently, this PMC test can be used not only to perform sweep boundary evaluation, but also compute the fold and fold-boundary points that are critical in many applications such as contact analysis [7]. It is important to note that the PMC test outlined in equation (1) provides the ability to not only identify the envelope points, but also the boundary points of the moving solid that generate each individual fold region [40].

The computational cost of the PMC-based sweep boundary evaluation depends strongly on the cost of (1) sampling a sufficiently dense set of points in the space; and (2) intersecting the corresponding inverted trajectory curves with the solid generator in its original configuration. A number of available techniques can be used to sample points of the space to any desired accuracy up to machine precision, such as a variety of cellular decompositions, as well as brute force and Monte Carlo methods, or combinations of these as discussed below.

The first step of most sampling algorithms is to compute a tight yet geometrically simple bounding volume for the generator. The smaller this sampling subset is, the fewer intersection tests will be required for the same sampling resolution. Note that the choice of a bounding volume must satisfy two possibly competing constraints: (1) the volume must fit around the object as tightly as possible, and (2) the computational cost of sampling for each sampled point to be as low as possible.

There are several types of volumes bounding a given set of points, which include bounding spheres, axis aligned bounding boxes (AABBs), oriented bounding boxes (OBBs), discretely oriented polytopes (k-DOPs), and convex hulls. The bounding spheres are simple, efficient, and rotationally invariant; AABBs are simple to compute, but the OBBs tend to provide a tighter fit around the generator, particularly when the generator is "thin" (e.g., has one dimension much smaller than the other two). A k-DOP is a convex polytope containing the object, and is constructed by taking a number of planes at infinity having predefined normals and translating them until they come in contact with the object. In general the convex hulls provide a tighter fit than the k-DOPs because they are not restricted by a preset number of planes and directions. Recent developments in fast computations of bounding volumes can be found in [41, 42, 43, 44], but note that we require a bounding volume of a set (sweep) that is not known *a priori*. In this paper we compute a tight bounding box of the sweep by: (1) computing the forward trajectory of the centroid of the generator; (2) computing the smallest bounding box of this forward trajectory, and (3) increasing the size of this bounding box by the radius of the smallest sphere enclosing the generator and centered at the centroid of the generator. This process is illustrated in Figure 3.

The envelopes generated by the moving generator may bound "thin" non-necessarily connected subsets as illustrated in section 4. One strategy to improve the sampling density near the envelopes is to exploit the information obtained from the inverted trajectory intersections to detect proximity of the envelope set relative to a point x that is being tested, as illustrated in Figure 4. Such a test can use the fact that, if a point is on a boundary or internal envelope, the inverted trajectory of that point will necessarily be tangent to the boundary of the generator, which follows from the definition of the envelopes [4]. For example, in the case of the 2D elliptical solid shown in Figure 4, point x is near the boundary envelope and therefore the intersection points are "close" to each other as measured along the inverted trajectory curve. Note that such intersection points that are close to each other is a necessary, but not sufficient condition to imply proximity of the sampled point to the envelope. However, developing such a metric is outside the scope of this paper.

3.1. Octree Decomposition

The first sampling strategy implemented in this work uses octrees, which are well-known hierarchical data structures [45, 46] that can be used to approximate a 3D set

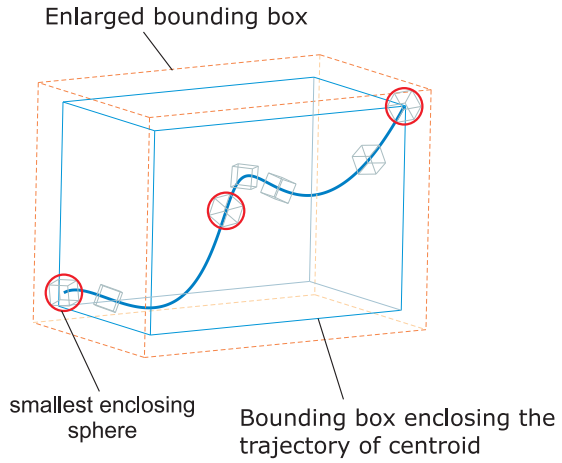


Figure 3: The tight bounding box is obtained by enlarging the smallest enclosing box of the trajectory of the centroid of the moving object (here a cube) by the radius of the smallest sphere enclosing the moving object.

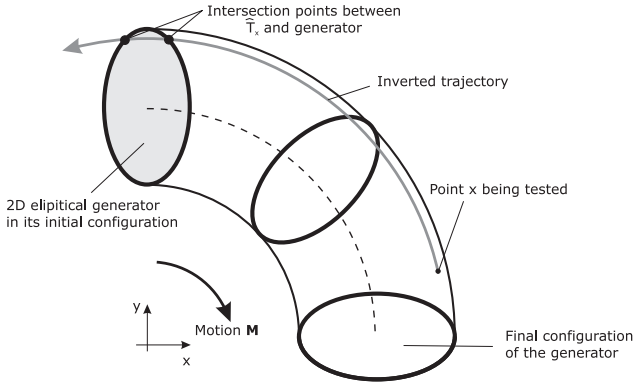


Figure 4: A 2D elliptical generator in planar motion. If point x is "near" an envelope, then the intersection points between $\hat{T}_x$ and generator are "close" to each other as measured along the inverted trajectory curve. Observe that the inverse is not always true.

of points by parallelepiped (often cubes) obtained via recursive subdivision. Each parent cell is decomposed into eight children, called octants, according to some prescribed subdivision criteria applied to points of the cells until an end condition is reached. The subdivision criteria usually consist of membership tests against a set of points, but can contain other information as well, such as differential properties of the shape being approximated. In our case, we follow the standard classification of each octant into "full", "empty" or "partially full" cells relative to the sweep (or, alternatively, the fold regions) based on the PMC test described above. The partially full octants are subdivided until the minimum prescribed resolution has been reached. Once all branches of the octree have reached the end condition, the list of partially full octants will be used to estimate a point of each octant that is considered to be a boundary point for the swept set or fold regions.

In our implementation, we test different points of each cell at various levels of the hierarchy in order to control the tradeoff between accuracy and computational cost. For example, the user can choose to test multiple points in each face or edge of the octree cell to improve the likelihood of detecting "thin" regions (see section 4 for details). Since faces, edges and vertices are shared by multiple cells, we perform the intersection tests for a given vertex only once. This essentially eliminates redundant tests, and significantly improves the overall computational cost. Each cell is recursively subdivided as long as the tested points of the cell are not uniformly classified and the cell has not reached the prescribed minimum cell size. In our implementation, we test the eight vertices of each cell that reaches the minimum size, and classify the cell as full, empty or partially full. Once all the octree branches have reached the end condition, we report the geometric center of each partially full octant as a point of the sweep boundary or a point of the fold boundaries (e.g., internal envelopes). Note that one can report other points of

the cell as an estimate for the envelope point, for example as a weighted average of the cell corners whose weights can depend, for example, on an estimated distance from each corner to the sweep boundary as suggested above. Although reporting different points of the partial octants will influence the smoothness of the resulting boundary, it will not change the number of partial octants that we use to compare the accuracy of the two sampling methods.

3.2. Marching along the Envelopes

The marching cubes algorithm was first introduced in [20] to extract triangulated surface information from 3D field data, and uses a divide and conquer approach to estimate the boundary of the field data in a cube that marches along the boundary. Then, based on this estimation, the algorithm recursively visits a subset of the neighboring cubes. There are several enhancements to the basic algorithm both in the original work as well as in subsequent papers [22] that primarily address the triangulation part of the algorithm. Thus, the marching cubes algorithm could be used to construct a triangulated surface approximating the envelopes. However, in order to be able to compare its performance and accuracy against those of other sampling methods, we only implemented here the marching step.

Marching along the boundary is straightforward once the initial cell is identified. For generic field data, identifying an initial cell that is partially inside the set being evaluated may be problematic, and may require user intervention. However, for our case this can be done automatically since we know *a priori* that a subset of the boundary of the generator will be part of the boundary of the set swept by the generator. Therefore, once the size of the marching cells is prescribed, one can initialize the algorithm by determining a partial cell that contains such a subset. A simple way to locate such a cell is to compute the inverted trajectory of an interior point of the generator, intersect this curve with the boundary of the generator, and center the initial cell at one of the intersection points. However, since the purpose of this paper is to compare different sampling strategies, we chose an initial marching cell that aligns with a cell of the brute force decomposition in order to maintain consistency with the other sampling approaches and therefore minimize the extraneous sources of variation.

We choose a marching cell whose size equals the minimum cell size in the octree decomposition. The eight vertices of every marching cell are classified with the PMC test, and the results are then used to decide which neighboring cells will be visited, as illustrated in Figure 5. The simplest implementation results by recursively visiting all neighbors of the marching cell that have not been already visited – i.e., 3 in 2D and 5 in 3D, but this also results in the least efficient implementation. In most cases the overall number of traversed cells can be reduced substantially by exploiting spatial and temporal coherence information. Note that in our implementation, we only visit one neighbor of the marching cell for all 2D planar cases[6], that is, one third (33.(3)%) of all neighbors that have not been already visited as illustrated in Figure 5(a). For 3D motions, the number of visited neighbors of the marching cell is 3, 4, or 5 (Figure 5(b)), that is, over 60% of all neighbors, which is required because we only test eight vertices of the marching cell. In turn, this implies that our specific marching cubes implementation is more efficient in 2D than in 3D. One can further improve the efficiency of the 3D marching cube traversal by testing more points of each marching cell, which would allow one to visit fewer neighbors than currently possible. However, this will also require additional curve-surface intersections between inverted trajectories, and the boundary of the generator for each visited cell.

3.3. Surface Reconstruction

Once the points on the boundary of sweep are computed, the last step is to reconstruct the surfaces of the envelope set. As already mentioned, developing new surface reconstruction algorithms is outside the scope of this paper. In this work we used 3DReshaper [47] to construct triangulated surfaces from the point clouds output by our sampling approaches, but there are many other potential avenues for surface reconstruction resulting in either tesselated or piecewise polynomial surfaces as discussed in section 1.1.

4. Implementation and Results

We implemented the generic sweep PMC test in Visual Studio and Parasolid geometric kernel. All curve-surface intersections have been performed within Parasolid. The boundary reconstruction employed 3DReshaper [47] simply as a means to visualize the computed point clouds.

4.1. Comparison of Performance and Accuracy of Octrees and Marching Cubes Space Sampling

As mentioned above, closed form solutions for sweep boundary evaluation exist today only for trivial objects and motions. Therefore, we believe that the most meaningful comparison between the two sampling methods discussed in section 3 is to perform the comparison relative to a regular grid traversed by a brute force algorithm, which is the best approximation for a prescribed minimum cell size (i.e., resolution). Therefore, we consider the number of partial cells obtained by a grid decomposition as a (reasonable) upper limit for the accuracy of both octree and marching cubes decompositions. Specifically, we compare the octree and marching cubes decompositions against the brute force (grid based) sampling such that the minimum size of the octree cell equals the size of the cube that "marches" along the envelopes, which, in turn,

[6]These are the cases in which the 2D solid generator moves according to a planar motion.

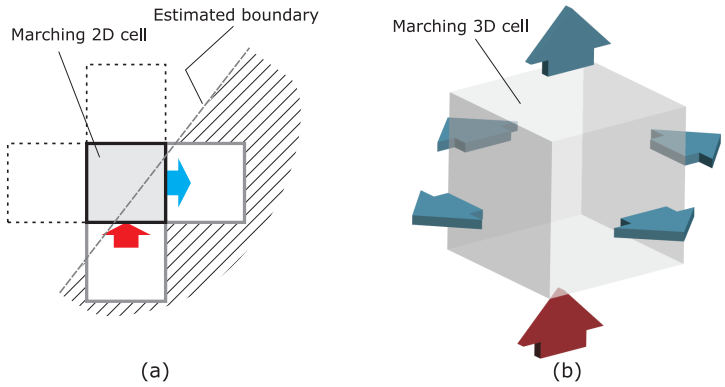


Figure 5: The results of the PMC test for a partial cell are used to march towards the cell's neighbors. By exploiting coherence information one can reduce the number of neighbors being visited at each step.

equals the size of the grid cell used in the brute force sampling. The only difference between the two versions of the marching cubes algorithm that we implemented is that one exploits coherence as explained in section 3.2, while the other visits all neighbors. The complete data set for all the examples discussed in this section is given in appendix A.

As already mentioned, we can choose the number of points that can be tested for each octree cell for all the intermediate levels of the octrees. Once an octree cell reaches the minimum prescribed size, we only test its corners according to our PMC. Our marching cubes implementation tests only the corners of the marching cell as well in order to maintain consistency with the octree implementation. We performed multiple tests in 2D and 3D to evaluate the CPU time of four sampling approaches: reference brute force (grid based) sampling (**BF**); octree decomposition (**O**); marching cubes that visits all neighbors (**MC**), and marching cubes that exploits coherence (**MCwc**). The collected data is given in appendix A.

The main observation is that the performance (in terms of CPU time) of the marching cubes algorithm that exploits coherence was consistently better than that of all the other implemented algorithms, while the accuracy (number of partial cells detected) was essentially the same as that of the brute force sampling. Both versions of the marching cubes algorithms detected the same number of partial cells as the brute force decomposition. The octree sampling approached this number as the number of points/cell increased (as shown in appendix B).

Another important observation is that testing only the corners of the cells in an octree decomposition leads almost always to a number of partial cells (and therefore accuracy) significantly below that obtained by the brute force or marching sampling techniques. The difference is clearly seen in Figure 13 shown in appendix B. One can improve the level of accuracy of the octree-based sampling by testing additional points of the octree cells, but this increases the computational cost exponentially with the depth of the octree[7], and this cost is already larger than that of the marching cubes algorithm.

In the first example in Figure 6(a), a planar non-convex shape is moving according to a planar rigid body motion that includes translation along a prescribed planar trajectory coupled with a uniform rotation around the centroid of the shape by $\pi/2$. Figure 6(b) shows the corresponding CPU time for the four sampling approaches implemented. The CPU times corresponding to the brute force sampling for the last two resolution levels are significantly higher than all the other CPU times (namely 81.7 sec. for resolution $r4$ and 324.6 sec. for $r5$ – see appendix A) and are not shown in this figure. For the highest resolution examined here ($r5$ in Figure 6(b) corresponding to over 410k cells in the bounding volume), the sampling based on the marching cubes with coherence is as accurate as the brute force sampling (that is, same number of partial cells), and more than twice as fast as the octree based sampling.

The second example shown in 6(c) illustrates a planar non-convex shape moving according to a planar affine motion that contains a uniform deformation. The data collected in this example has the same trend as the data from the first example. Specifically, the number of partial cells detected by the sampling based on the marching cubes with coherence is the same as that of the brute force sampling, while the CPU time is less than half of that of the octree based sampling for the highest resolution that was evaluated.

We have also collected data for three 3D examples that are shown in Figure 7. Again, the performance of the marching cubes algorithm is consistently better than that of the octree-based sampling, while the accuracy is as good as that of the brute force sampling. Note that for all the 3D examples the marching cubes sampling has the same accuracy as that of the brute force, while the octrees have a

[7]Each additional point that we test in a given octree cell implies 8 more points tested for the children of the cell, which increases the complexity by $O(8^d)$, where d is the depth of the octree.

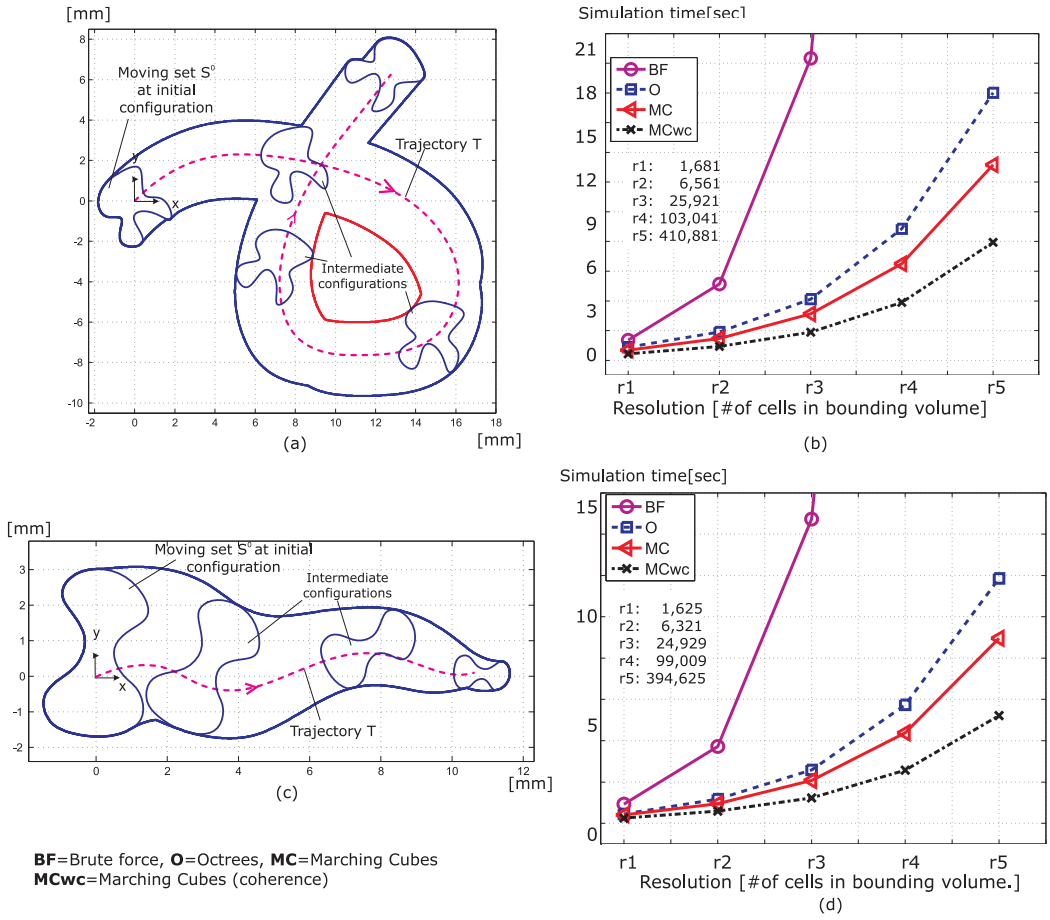


Figure 6: CPU time for a rigid (a) or deformable (c) planar non-convex shape, bounded by piecewise rational curves moving according to planar affine motions. The CPU time was recorded in (b) and (d) for the four implemented sampling approaches based on brute force (**BF**), octrees (**O**), marching cubes (**MC**) and marching cubes with coherence (**MCwc**).

slightly lower accuracy (of about 1% – see appendix A.2). However, the difference between the corresponding CPU times has decreased compared to the 2D case. This decrease is primarily due to a larger percentage of neighbors visited in 3D than in 2D (see also ection 3.2).

4.2. Examples with Reconstructed Boundary

The swept sets of several 3D shapes undergoing general affine motions with and without deformations have been computed based on our sampling-based sweep boundary evaluation approach. Note that these examples are beyond the capabilities of all current commercial geometric modeling systems. While we use these examples to illustrate some of the features and limitations of the octree and marching cubes sampling investigated in this paper, our approach can be applied to the class of problems of arbitrarily complex 3D solids undergoing affine motion. This is so because the only assumption that we made in our PMC test was that the initial and final configuration of the moving solid generator should not overlap (see [4] for details).

In all examples shown in this section we used both the octree and the marching cubes decompositions to generate the point clouds corresponding to the sweep and fold boundary points. These point clouds are then used to reconstruct a piecewise linear approximation of the corresponding boundary surfaces. As mentioned above, the boundary reconstruction was performed with a commercial code [47], but reconstructing piecewise polynomial surfaces is also possible [17, 31].

The first example illustrated in Figure 8 shows a sphere moving according to an affine motion. The center of the sphere moves along a prescribed trajectory, while undergoing a uniform scaling transformation. The two point clouds shown in Figure 8(a) contain all points that have been classified either as a sweep boundary point or as a fold boundary point. Figure 8(b) shows the result of our sweep boundary evaluation based on a marching cubes sampling, while the boundary of the fold regions A and B have been computed with an octree (Figures 8(c) and (e)) or a marching cubes sampling (Figures 8(d) and (f)). It can be observed that the marching cubes detect signif-

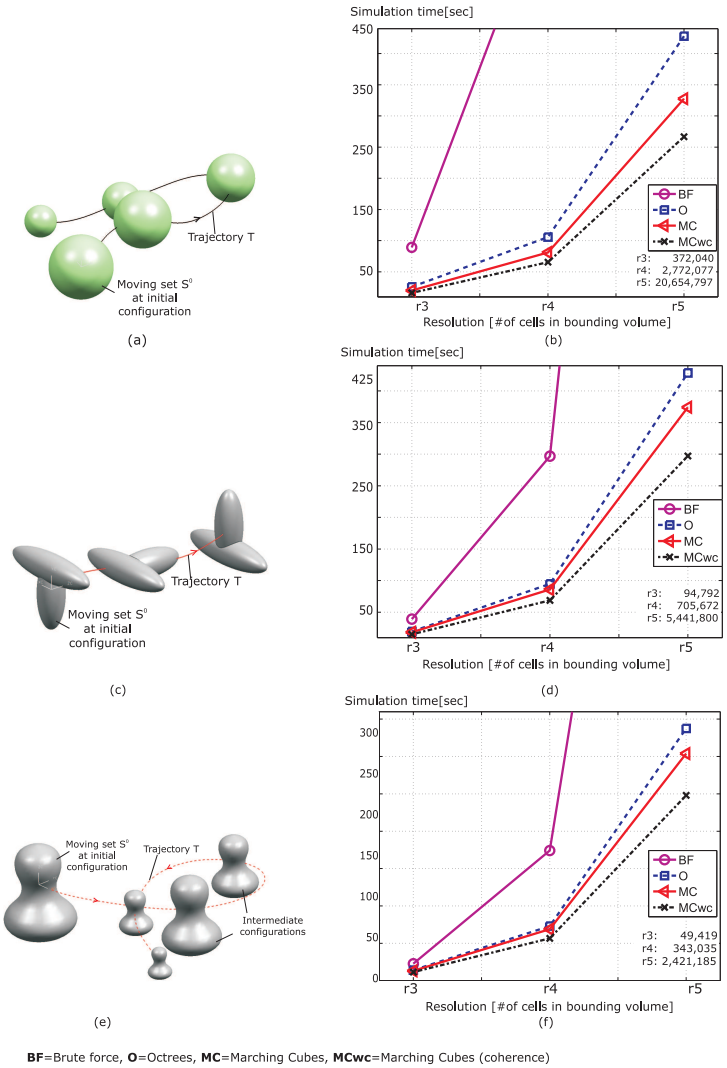


Figure 7: Performance metrics for 3-dimensional solid shapes.

icantly better the sharp/thin edges of the boundaries of these fold regions and the corresponding approximations of the fold regions are therefore more accurate than the results obtained from an octree based sampling. This behavior of the octree decomposition near the sharp features is due to (1) the hierarchical nature of the octrees, which suggests that octree cells at the coarser level could contain sharp features of the boundary yet result in identical classifications of the tested points; and (2) our subdivision criterion that considers only the inverted trajectory of a *given* point.

In the second example shown in Figure 9, we have an x-shaped solid object moving in 3D space: translating along a prescribed trajectory, and rotating by $5\pi/6$ around y axis (not shown). Figure 9(a) illustrates this motion by displaying five configurations of the object along the motion. The two point clouds containing the sweep boundary points and the fold boundary points are displayed in Figure 9(b), and the computed sweep surface based on a marching cubes sampling is shown in Figure 9(c). Once again, the marching cubes sampling (with or without coherence) generated exactly the same number of partial cells as the brute force sampling.

The third example shows a tubular non-convex shape homeomorphic to a torus (Figure 10), which translates in the x direction along a linear trajectory, while rotating around the x axis by $\pi/2$ and undergoing a uniform scaling. Several intermediate configurations of the object are shown in Figure 10(a), while the computed sweep boundary based on a marching cubes sampling is displayed in Figure 10(b).

A stepped cutter is shown in Figure 11 as it moves during a 5-axis NC machining simulation. The cutting tool is modeled as a piecewise polynomial surface of revolution because the cutter rotates around its axis much faster than it travels along the 5-axis motion. Note that this is a standard assumption in any NC machining simulation, but it

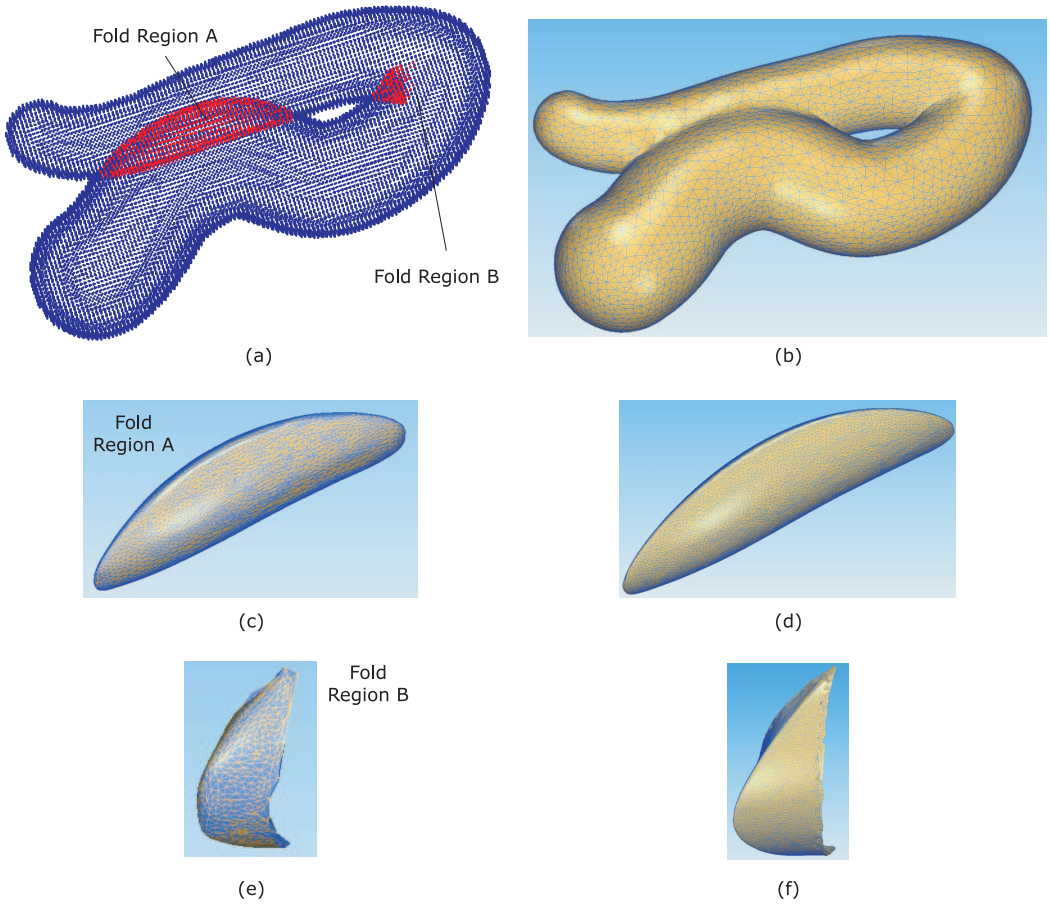


Figure 8: Sweep and fold boundary computation for a deformable sphere. During the prescribed motion, the sphere generates both local and global self intersections that form two connected subsets of fold points. Figure (b) shows the reconstructed sweep boundary from the point cloud illustrated in (a) generated by a marching cube sampling. Figures (c) and (e) show the reconstructed boundary of the two connected components of the fold region with the octree sampling, while Figures (d) and (f) show the boundary evaluation of these fold regions based on the marching cubes with coherence.

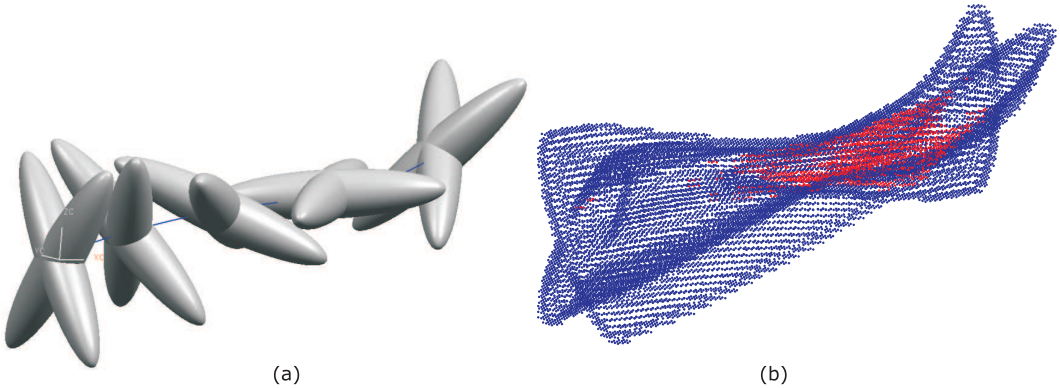
is *not* required by our approach. The tip of the cutter moves along a prescribed cutter location (CL) trajectory seen in Figure 11(b), while rotating around the y and x axes by $\pi/3$ and $5\pi/12$ respectively. Figures 11(a) and (b) illustrate this motion by displaying several configurations of the cutter along the motion. The computed triangular approximation of the boundary of the set swept by the cutter based on an octree space sampling is shown in Figure 11(c).

Finally, a suspension arm (Figure 12(a)) bounded by NURBS surfaces undergoes a spatial rigid body motion. The marching cubes sampling described in section 3.2 outputs the point cloud shown in Figure 12(b), which contains the center of the cells that were classified as partially full. The triangulated boundary shown in Figure 12(c) represents the result of the sweep boundary evaluation. Many of the sharp edges that can be seen in the point cloud are not reflected in the reconstructed surface due to the specific triangulation algorithm used. This example clearly illustrates the need for surface reconstruction algorithms that can detect sharp features in the point cloud such as those presented in [48, 49].

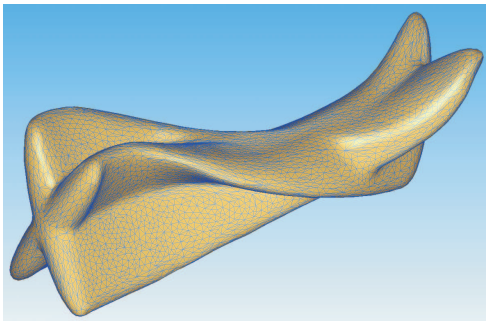
5. Conclusions

The set swept by a moving object in space has found many applications not only as an effective design tool for creating highly complex three-dimensional shapes, but also in computer aided manufacturing, as well as in robotics and computer graphics. The high level of complexity of performing boundary evaluation for three dimensional sweeping is well-documented. The typical two-step approach to perform such a boundary evaluation generates candidate faces that can potentially belong to the sweep boundary followed by an elimination of those faces that are not actually part of the boundary. The latter step has traditionally relied on heuristics for eliminating the candidate faces that are not part of the sweep boundary.

In this paper we are using the first known Point Membership Classification test for general solid sweeping that

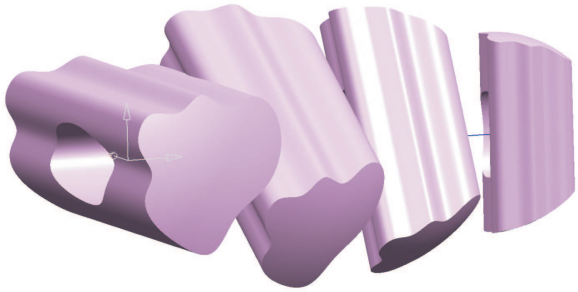


(a) (b)

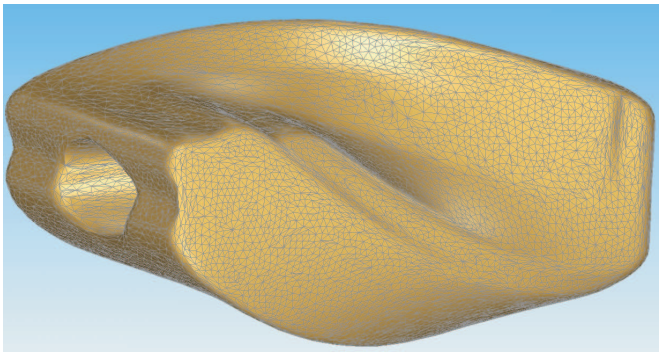


(c)

Figure 9: Sweep boundary evaluation for an x-shaped non-convex free-form object with a marching cubes sampling.



(a)



(b)

Figure 10: Sweep boundary evaluation for a non-convex deformable free-form object using a marching cubes space sampling.

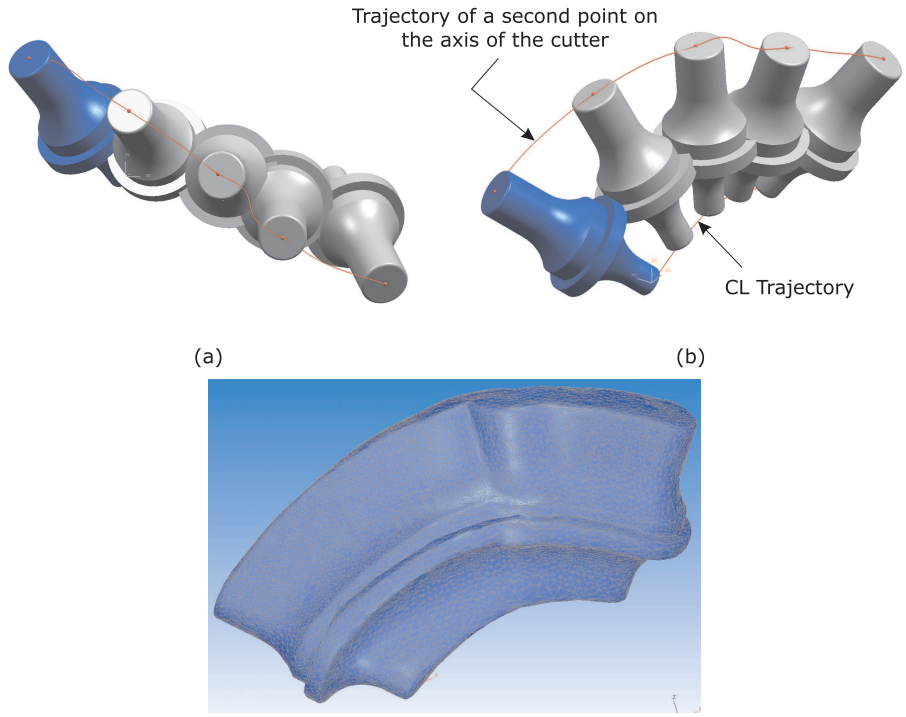


Figure 11: A milling cutter moving through space. The sweep boundary evaluation used an octree decomposition to sample points.

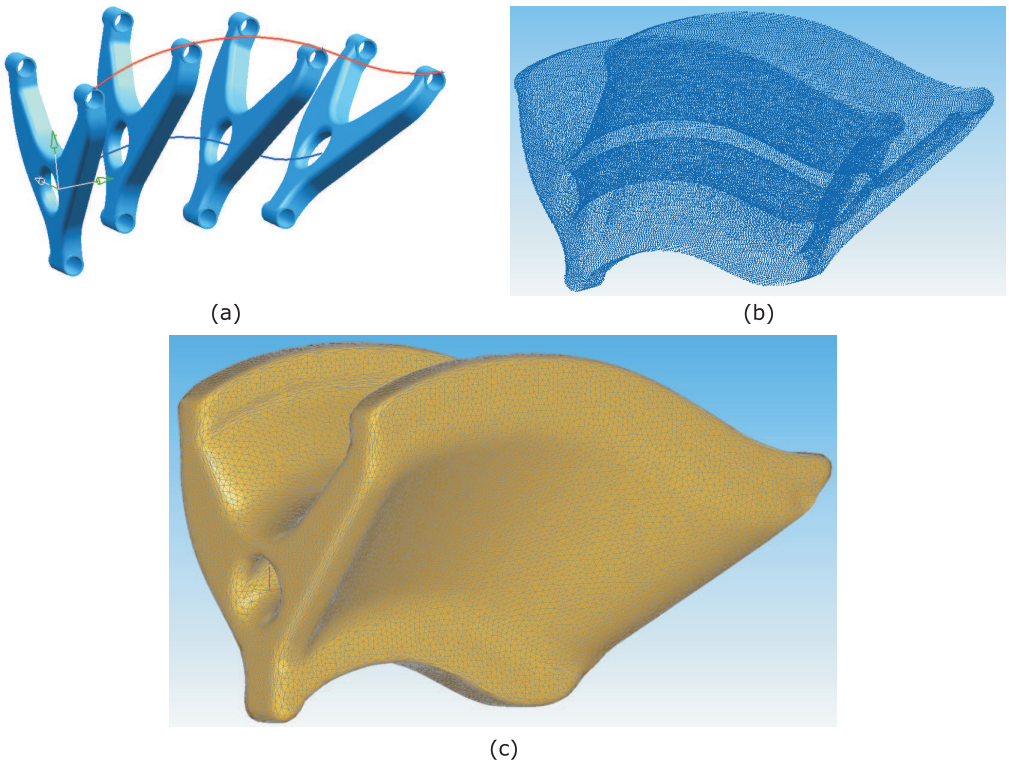


Figure 12: A suspension arm [50], whose boundary representation contains 90 parametric faces, sweeps a subset of the space.

we introduced in [4] to compute points on the boundary of the set swept by a solid object moving according to an affine motion. Our PMC test provides the ability to compute not only points that are on the boundary of the sweep, but also envelope points that are in the interior of the set swept by the moving object. Our formulation relies on two assumptions: the initial and final configuration of the moving solid generator should not overlap – as discussed in [4], and the curve-surface intersection function is robust and available.

Our results show that a space sampling based on a fast marching cubes algorithm has substantially better performance characteristics compared to those of the octree-based sampling for the same accuracy of the sweep boundary approximation. Specifically, the CPU time for marching cubes with coherence is, on average, 47.8% in 2D and 73% in 3D of the CPU time of the octree-based sampling (based on the data shown in appendix A). Moreover, the octree decomposition with uniform subdivision performs poorly in detecting thin features of the sweep or fold boundary, as can be see in Figure 8(e). Note that the fold regions generated by the local or global self-intersections of the envelopes are usually smaller, and have more sharp/thin features than the sweep itself. One possible approach to overcome this limitation is to use adaptive subdivision criteria for the octrees, but this will require even more points to be classified against the sweep. On the other hand, the marching cubes algorithm that exploits coherence exhibits the same accuracy as the brute force (grid based) sampling in terms of the number of partial cells detected, but at a fraction of the computational cost of the octree-based sampling.

The computational efficiency of our marching cubes implementation could be further increased in several different ways. First, a better understanding of how the number of points that need to be tested for each cell influence the number of neighbors that have to be traversed would lead to a more efficient exploitation of the coherence information, and hence to a more efficient algorithm. Furthermore, the inverted trajectories could, in principle, be constructed directly from the motion representation, which would eliminate the need to interpolate the points generated along each inverted trajectory. However, the most significant improvement in the computational efficiency would probably come from an optimized curve-surface intersection algorithm that could take advantage of a variety of techniques for speeding up the computations, including some of the fast hierarchical bounding volumes discussed above.

Importantly, a sampling-based boundary evaluation for general solid sweeps is not only generic, i.e., is applicable to arbitrarily complex geometries and affine motions, but can generate an approximation of the envelope surfaces to any desired level of detail up to machine precision. Moreover, the proposed approach can be implemented in any geometric representation that supports curve-solid or curve-surface intersections, and is suitable to hardware accelerated computations as well as to parallelization. Our approach remains valid for arbitrarily complex moving solid objects and complex motions as long as the intersection between the initial and final configurations of the solid generator remains empty.

Acknowledgments

This work was supported in part by the National Science Foundation grants CAREER award CMMI-0644769, and CMMI-0555937. All 3D examples were created by using Parasolid, courtesy of Siemens-PLM. The authors would like to thank the anonymous reviewers for their constructive comments.

Appendix A:

All CPU times have been recorded on a PC Workstation with a 2.66GHz QuadCore processor with 8GB of RAM.

A.1. Collected 2D test data

2D Example - Figure 6 (a)	Resolution (%)	r1	r2	r3	r4	r5
Brute force	# of tested points	1681	6561	25921	103041	410881
	# of partial cells	216	436	872	1744	3492
	time[sec]	1.352	5.134	20.328	81.664	324.598
Octrees	# of tested points	976	2032	4141	8476	16997
	# of partial cells	216	436	872	1740	3449
	time[sec]	0.882	1.913	4.114	8.856	18.009
Marching Cubes(MC)	# of tested points	737	1487	2985	5978	11973
	# of partial cells	216	436	872	1744	3492
	time[sec]	0.693	1.475	3.126	6.5	13.172
MC with coherence	# of tested points	430	870	1743	3485	6983
	# of partial cells	216	436	872	1744	3492
	time[sec]	0.441	0.937	1.899	3.903	7.942

2D Example - Figure 6 (c)	Resolution (%)	r1	r2	r3	r4	r5
Brute force	# of tested points	1625	6321	24929	99009	394625
	# of partial cells	156	312	624	1256	2506
	time[sec]	0.933	3.724	14.729	58.769	236.813
Octrees	# of tested points	660	1437	2955	6092	12239
	# of partial cells	156	312	624	1250	2483
	time[sec]	0.453	1.174	2.582	5.743	11.852
Marching Cubes(MC)	# of tested points	545	1100	2202	4420	8828
	# of partial cells	156	312	624	1256	2506
	time[sec]	0.403	0.95	2.081	4.381	8.956
MC with coherence	# of tested points	310	624	1248	2512	5012
	# of partial cells	156	312	624	1256	2506
	time[sec]	0.259	0.598	1.235	2.572	5.222

A.2. Collected 3D test data

3D Example - Figure 7 (a)	Resolution (%)	r1	r2	r3	r4	r5
Brute force	# of tested points	8400	52059	372040	2772077	20654797
	# of partial cells	1432	5844	23584	94530	378806
	time[sec]	1.956	12.362	89.349	665.708	4960
Octrees	# of tested points	5022	20716	84808	343319	1379831
	# of partial cells	1416	5808	23490	94053	377356
	time[sec]	1.452	6.145	25.865	105.569	427.854
Marching Cubes(MC)	# of tested points	4520	18539	75122	301693	1209686
	# of partial cells	1432	5844	23584	94530	378806
	time[sec]	1.143	4.831	20.161	80.974	327.599
MC with coherence	# of tested points	3591	14798	59935	240514	963793
	# of partial cells	1432	5844	23584	94530	378806
	time[sec]	0.937	3.939	16.379	65.413	266.742

3D Example - Figure 7 (c)	Resolution (%)	r1	r2	r3	r4	r5
Brute force	# of tested points	2200	13608	94792	705672	5441800
	# of partial cells	352	1533	6297	25275	101400
	time[sec]	0.765	5.181	39.198	296.79	2332.49
Octrees	# of tested points	1698	5221	21539	88479	363512
	# of partial cells	329	1514	6229	24951	100387
	time[sec]	0.69	3.462	19.547	94.815	428.459
Marching Cubes(MC)	# of tested points	1081	4314	19302	78599	315926
	# of partial cells	352	1533	6297	25275	101400
	time[sec]	0.548	3.133	18.3	85.961	374.281
MC with coherence	# of tested points	713	3362	14831	60599	244102
	# of partial cells	352	1533	6297	25275	101400
	time[sec]	0.443	2.782	15.157	68.633	297.096

3D Example - Figure 7 (e)	Resolution (%)	r1	r2	r3	r4	r5
Brute force	# of tested points	1575	8019	49419	343035	2421185
	# of partial cells	174	792	3324	13497	54182
	time[sec]	0.534	3.237	22.776	174.17	1298.7
Octrees	# of tested points	788	2679	13598	48953	189869
	# of partial cells	160	765	3276	13279	53114
	time[sec]	0.384	2.145	14.072	73.058	337.483
Marching Cubes(MC)	# of tested points	636	2344	10103	42570	171988
	# of partial cells	174	792	3324	13497	54182
	time[sec]	0.334	1.98	13.03	68.902	303.987
MC with coherence	# of tested points	396	1790	8089	33839	136755
	# of partial cells	174	792	3324	13497	54182
	time[sec]	0.277	1.839	11.478	56.499	247.93

Appendix B:

The accuracy of the octree-based sampling as a function of the number of tested points in each octree cell.

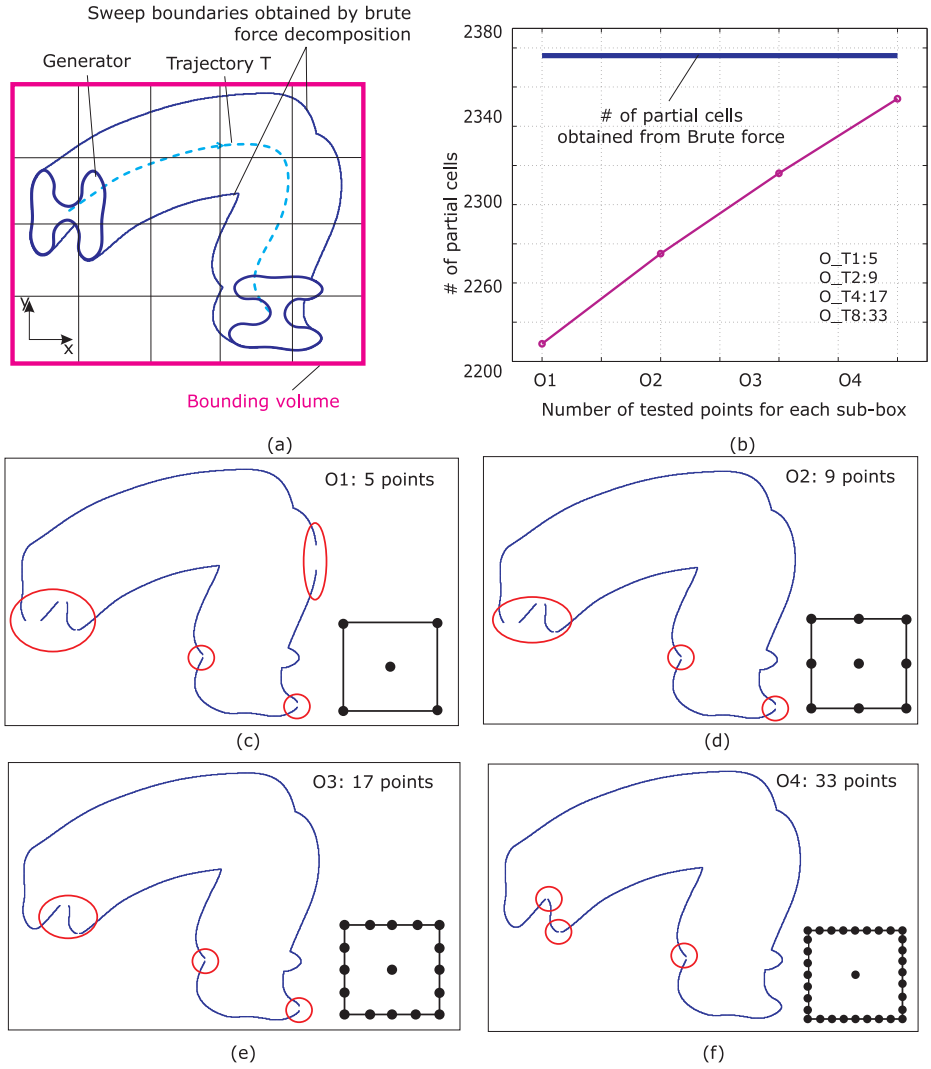


Figure 13: Octree/quadtree performance as a function of the number of tested points in each cell. Figure (b) shows how the number of tested points per cell influences the accuracy for the shape and motion illustrated in Figure (a). Figures (c-f) show the corresponding effect on the boundary points.

References

[1] A. Requicha, Representations for rigid solids: Theory, methods and systems, Computing Surveys 12 (4) (1980) 437–463.

[2] J. W. Bruce, P. J. Giblin, Curves and Singularities, Cambridge University Press, 1992.

[3] A. A. G. Requicha, H. B. Voelcker, Boolean operations in solid modeling: Boundary evaluation and merging algorithms, Proceedings of the IEEE 73 (1) (1985) 30–44.

[4] H. Erdim, H. T. Ilieş, Classifying points for sweeping solids, Computer-Aided Design 40 (9) (2008) 987–998.

[5] A. Krishnamurthy, R. Khardekar, S. McMains, K. Haller, G. Elber, Performing efficient NURBS modeling operations on the GPU, in: SPM '08: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, ACM, New York, NY, USA, 2008, pp. 257–268.

[6] R. B. Tilove, Set membership classification: A unified approach to geometric intersection problems, IEEE Transactions on Computer C-29 (10) (1980) 874–883.

[7] H. Erdim, H. T. Ilieş, Contact analysis between a moving solid and the boundary of its swept volume, in: Proceedings of ASME 2008 IDETC & CIE, Design Automation Conference, New York, NY, USA, 2008.

[8] N. M. Patrikalakis, T. Maekawa, Intersection problems, in: M.-S. K. G. Farin, J. Hoschek (Ed.), Handbook of Computer Aided Geometric Design, Elsevier Science Publishers, 2001, pp. 623–650.

[9] K. Abdel-Malek, J. Yang, D. Blackmore, K. Joy, Swept volumes: Foundations, perspectives, and applications, International Journal of Shape Modeling 12 (1) (2006) 87–127.

[10] Y. J. Kim, G. Varadhan, M. C. Lin, D. Manocha, Fast swept volume approximation of complex polyhedral models, in: SM '03: Proceedings of the 8^{th} ACM Symposium on Solid Modeling and Applications, ACM, New York, NY, USA, 2003, pp. 11–22.

[11] J. Rossignac, J. J. Kim, S. C. Song, K. C. Suh, C. B. Joung, Boundary of the volume swept by a free-form solid in screw motion, Computer-Aided Design 39 (9) (2007) 745–755.

[12] E. E. Hartquist, J. Menon, K. Suresh, H. B. Voelcker, J. Zagajac, A computing strategy for applications involving offsets, sweeps, and Minkowski operations, Computer-Aided Design 31 (3) (1999) 175–183.

[13] J. Ahn, S. J. Hong, Approximating 3D general sweep boundary using depth-buffer, in: Computational Science and Its Applications ICCSA, 2003, pp. 508–517.

[14] D. Blackmore, M. Leu, A differential equation approach to swept volumes, Proceedings of the 2nd International Conference on Computer Integrated Manufacturing (1990) 143–149.

[15] D. Blackmore, M. Leu, L. Wang, The sweep-envelope differential equation algorithm and its application to NC machining verification, Computer Aided Design 29 (1997) 629–637.

[16] T. W. Sederberg, F. Chen, Implicitization using moving curves and surfaces, Computer Graphics 29 (Annual Conference Series) (1995) 301–308.

[17] M. Peternell, H. Pottmann, T. Steiner, H. Zhao, Swept volumes, Computer-Aided Design Applications 2 (2005) 599–608.

[18] M. S. Paterson, F. F. Yao, Binary partitions with applications to hidden surface removal and solid modelling, in: SCG '89: Proceedings of the fifth annual symposium on Computational geometry, ACM, New York, NY, USA, 1989, pp. 23–32.

[19] H. Samet, Spatital data structures, in: SIGGRAPH '07: ACM SIGGRAPH 2007 courses, ACM, New York, NY, USA, 2007, p. 1.

[20] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, SIGGRAPH Comput. Graph. 21 (4) (1987) 163–169.

[21] G. M. Nielson, Dual marching cubes, in: VIS '04: Proceedings of the conference on Visualization '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 489–496.

[22] R. Shu, C. Zhou, M. S. Kankanhalli, Adaptive marching cubes, The Visual Computer 11 (4) (1995) 202–217.

[23] G. Bradshaw, C. O'Sullivan, Sphere-tree construction using dynamic medial axis approximation, in: SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ACM Press, New York, NY, USA, 2002, pp. 33–40.

[24] F. Remondino, From point cloud to surface: the modeling and visualization problem, in: Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV-5/W10, 2003.

[25] H. Edelsbrunner, E. Mücke, Three-dimensional alpha shapes, ACM Transactions of Graphics 13 (1) (1994) 43–72.

[26] T. K. Dey, J. Giesen, J. Hudson, Delaunay based shape reconstruction from large data, in: PVG '01: Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics, IEEE Press, Piscataway, NJ, USA, 2001, pp. 19–27.

[27] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of Hermite data, in: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 2002, pp. 339–346.

[28] L. P. Kobbelt, M. Botsch, U. Schwanecke, H. Seidel, Feature sensitive surface extraction from volume data, in: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 2001, pp. 57–66.

[29] G. Varadhan, S. Krishnan, Y. J. Kim, D. Manocha, Feature-sensitive subdivision and isosurface reconstruction, in: VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03), IEEE Computer Society, Washington, DC, USA, 2003, p. 14.

[30] H. Pottmann, S. Leopoldseder, A concept for parametric surface fitting which avoids the parametrization problem, Comput. Aided Geom. Des. 20 (6) (2003) 343–362.

[31] Geomagic Inc., Geomagic Studio, http://www.geomagic.com/.

[32] K.-S. Cheng, W. Wang, H. Qin, K.-Y. Wong, H. Yang, Y. Liu, Design and analysis of optimization methods for subdivision surface fitting, IEEE Transactions on Visualization and Computer Graphics 13 (5) (2007) 878–890.

[33] W.-K. Jeong, K. Kähler, J. Haber, H.-P. Seidel, Automatic Generation of Subdivision Surface Head Models from Point Cloud Data, in: M. McCool, W. Stürzlinger (Eds.), Graphics Interface 2002, Canadian Human-Computer Communications Society, A K Peters, Calgary, Canada, 2002, pp. 181–188.

[34] W.-K. Jeong, C.-H. Kim, Direct reconstruction of a displaced subdivision surface from unorganized points, Graph. Models 64 (2) (2002) 78–93.

[35] A. Adamson, M. Alexa, Approximating and intersecting surfaces from points, in: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 230–239.

[36] N. Amenta, Y. J. Kil, Defining point-set surfaces, in: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, ACM Press, New York, NY, USA, 2004, pp. 264–270.

[37] M. Pauly, R. Keiser, L. P. Kobbelt, M. Gross, Shape modeling with point-sampled geometry, in: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, ACM Press, New York, NY, USA, 2003, pp. 641–650.

[38] C. M. Hoffmann, The problems of accuracy and robustness in geometric computation, Computer 22 (3) (1989) 31–40.

[39] H. T. Ilieş, Continuous collision and interference detection for 3D geometric models, ASME Transactions, Journal of Computing and Information Science in Engineering 9 (2) (2009) 021007–1 – 021007–7.

[40] H. Erdim, H. T. Ilieş, Detecting and quantifying envelope singularities in the plane, Computer Aided Design 39 (10) (2007) 829–840.

[41] S. Cameron, Approximation hierarchies and s-bounds, in: SMA '91: Proceedings of the first ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications, ACM, New York, NY, USA, 1991, pp. 129–137.

[42] D. L. James, D. K. Pai, Bd-tree: output-sensitive collision detection for reduced deformable models, in: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, ACM, New York, NY, USA,

2004, pp. 393–398.
[43] T. Klug, M. Alexa, Bounding volumes for linearly interpolated shapes, in: CGI '04: Proceedings of the Computer Graphics International, IEEE Computer Society, Washington, DC, USA, 2004, pp. 134–139.
[44] D. Steinemann, M. A. Otaduy, M. Gross, Tight and efficient surface bounds in meshless animation, Computers & Graphics 32 (2) (2008) 235–245.
[45] D. Meagher, Geometric modeling using octree encoding, Computer Graphics and Image Processing 19 (2) (1982) 129–147.
[46] H. Samet, The quadtree and related hierarchical data structures, ACM Comput. Surv. 16 (2) (1984) 187–260.
[47] Technodigit, 3DReshaper, http://www.3dreshaper.com/.
[48] G. Stylianou, G. Farin, Crest lines for surface segmentation and flattening, IEEE Transactions on Visualization and Computer Graphics 10 (5) (2004) 536–544.
[49] Y. Ohtake, A. Belyaev, H.-P. Seidel, An integrating approach to meshing scattered point data, in: SPM '05: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling, ACM, New York, NY, USA, 2005, pp. 61–69.
[50] SolidWorks, 3D ContentCentral, http://www.3dcontentcentral.com/.