Computer-Aided Design 44 (2012) 522–536

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

High accuracy NC milling simulation using composite adaptively sampled distance fields

Alan Sullivan *, Huseyin Erdim, Ronald N. Perry, Sarah F. Frisken [1]

Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 5 August 2011
Accepted 11 February 2012

Keywords:
Distance fields
NC milling simulation
ADF
Swept volumes

ABSTRACT

We describe a new approach to shape representation called a *composite adaptively sampled distance field (composite ADF)* and describe its application to NC milling simulation. In a *composite ADF* each shape is represented by an analytic or procedural signed Euclidean distance field and the milled workpiece is given as the Boolean difference between distance fields representing the original workpiece volume and distance fields representing the volumes of the milling tool swept along the prescribed milling path. The computation of distance field of the swept volume of a milling tool is handled by an inverted trajectory approach where the problem is solved in tool coordinate frame instead of a world coordinate frame. An octree bounding volume hierarchy is used to sample the distance functions and provides spatial localization of geometric operations thereby dramatically increasing the speed of the system. The new method enables very fast simulation, especially of free-form surfaces, with accuracy better than 1 micron, and low memory requirements. We describe an implementation of 3 and 5-axis milling simulation.

1. Introduction

Machining is defined as the process of removing material from a workpiece in the form of chips. Machining processes, such as milling, turning and drilling, are very important manufacturing operations with a wide range of applications including prototyping, low volume manufacturing, fabrication of molds and dies, and the finishing of parts fabricated by casting or stamping.

During NC milling a computer controlled rotary cutting tool follows a prescribed path to cut a workpiece. The essential goal of NC milling simulation and verification is mathematically removing the swept volume generated by each cutter movement along the NC trajectory thus obtaining a model of the in-process and final machined surface.

Simulation of NC milling can be used for optimization of the cutting conditions, tool path planning, error analysis, collision avoidance and modeling of cutting forces. NC milling simulation seeks to avoid the cost and delays associated with fabrication of defective parts. Fabrication mistakes prior to actual milling process can be corrected by quickly and accurately simulating the process. However, to be useful, the simulation has to be much faster than the real fabrication time and has to provide sufficient accuracy so that small defects can be detected. Overall, NC simulation has become an important step in CAM (Computer Aided Manufacturing) systems.

2. Related work

Numerically controlled (NC) milling simulation systems have made great progress after several years of development. The basic idea is to mathematically remove the volume swept by cutter movements along the NC trajectory from the model of the raw stock, and thus obtaining a model of the in-process or final machined workpiece.

In NC machining, a swept volume is represented by a set of points on the moving cutter as it moves over the machined surface. 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 geometric and solid modeling. The mathematical formulation of the swept volume computation has been investigated using singularity theory [1], envelope theory [2–4], Jacobian rank deficiency method [5,6], sweep differential equation (SDE) [7,8], Minkowski sums [9], implicit modeling [10,11], error bounded dual-contouring like approach [12] and kinematics [13]. Many approaches to computing the boundary of swept volumes have been published in the literature and an extensive review appears in [14].

NC simulation methods can be categorized into three major approaches: solid modeling, spatial partitioning and discrete vectors.

* Corresponding author. Tel.: +1 617 621 7596; fax: +1 617 621 7550.
E-mail addresses: sullivan@merl.com (A. Sullivan), erdim@merl.com (H. Erdim), perry@merl.com (R.N. Perry), sfrisken@gmail.com (S.F. Frisken).
[1] 61 Solutions Inc, USA.

0010-4485/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2012.02.002

Solid modeling-based simulators use a boundary representation (*B*-rep) to represent the milled workpiece and explicitly perform Boolean subtraction operations between a solid model of the workpiece and the volume swept by a cutter as it moves between two adjacent tool positions. Although solid modeling can provide accurate verification and error assessment, the computation cost is known to increase rapidly as the number of cutter movements increases. Solid modeling methods have been used for simple tool paths in 2.5D end-milling [15], for free-form surfaces in 3-axis ball-end milling [16–18], and as well as for 5-axis flank milling with tapered ball-end mills [19].

B-rep based milling simulators are theoretically capable of providing a highly accurate simulation of machining, but suffer from high computational cost in terms of time, data storage, and complexity. A boundary representation consists of interconnected lists of vertices, edges, and faces. Therefore, all computations on boundary representations involve traversing these lists (also known as *boundary traversal*). The amount of memory required to represent new vertices, edges, faces and all the connectivity information between them grows with the number of tool motions. The essential requirement is that a *B*-rep must specify unambiguously the boundary of a bounded regular solid in $\mathbb{E}^3$ [20]. Section 7 will give a detailed comparison of our approach with a solid modeler based simulation.

Another approach to NC milling simulation, cell decomposition, uses spatial partitioning to represent the workpiece and cutter. In this approach, the workpiece, tool and swept volume are decomposed into simple geometric elements, or cells, using spatial partitioning approaches such as ray casting [2,21], *Z*-buffer [22], *G*-buffer [23], dexel [24–26], Graf-tree [27], voxel and octree, etc. These approaches approximate the swept volume generated by each tool motion to a user defined accuracy that depends on the size and shape of the simple geometric element. Although these methods are computationally efficient, the results are view dependent, and changes in viewing direction require the simulation to be run again. Furthermore, in order to increase the accuracy of cell decomposition in the model, the sizes of the cells have to be reduced. As a result, these decompositions consume a considerable amount of memory and time.

The third approach, called the point vector method, approximates the machined surface by a discrete set of points and the vectors originating from these points. The cutting is simulated by calculating the intersection of these vectors with the cutter swept volumes [28–30]. The vectors are clipped if the tool passes at that location. The point vector method is one of the most efficient methods in NC simulation. It has the advantage of being able to detect regions where undercutting or overcutting occurred. However this method is not suitable when the surface normal vectors change abruptly which limits the applicability of the method to general cutting programs. In addition to these methods, Graphical processing units (GPUs) have advanced greatly over the last few years. They gain orders of magnitude of speed over existing solutions due to their processing power such as dexel representation for 3-axis milling [31] and triple dexel representation for 5-axis milling [32].

2.1. Goals and outline

Free-form surfaces are extensively used in the die-mold, aerospace and automotive industries. These surfaces are often used as functional parts and require highly accurate surface finishing. In die-mold applications, a tool path can contain on the order of 10^6 tool motions making the computational cost for characterizing the swept volumes of all tool movements very expensive. The existing simulation methods are slow and require too much memory for practical use. In this paper, we propose a new approach to NC milling simulation that can rapidly generate a highly accurate representation of the milled workpiece. This new method consists of three elements that help to overcome the shortcomings of the previous methods:

1. Each surface in the simulation is implicitly represented as the zero level iso-surface of an analytically or procedurally defined signed Euclidean distance field.
2. The machined surface of the workpiece is implicitly represented as the Boolean difference between distances fields representing the original workpiece and the distance fields of volumes swept by the tool.
3. An octree bounding volume hierarchy is used to sample the distance field functions to obtain spatial localization of geometric operations.

We show that the result is a high speed milling simulation system with sub-micron accuracy and a small memory footprint.

Our method is based on a new approach to computing the distance field of a swept tool, and its application to 3 and 5-axis NC milling simulation. We formulate our approach in Sections 3–5 outline the procedure of the proposed method. Section 6 gives details about the rendering approach. We illustrate different examples for different tools in Section 7. Finally, Section 8 summarizes the contributions of this paper and explores the extension of our approach to distance field rendering and generation using GPU.

3. Distance fields

A distance field is a scalar field that specifies the minimum distance to an object from a point in space. A distance field is an effective representation of shape, and is a special case of shape modeling with implicit functions [33]. Unlike the more common boundary representation (*B*-rep), the distance field of an object is defined everywhere in space. Given a signed distance field, several properties can be derived including the boundary of an object. The zero value iso-surface can be implicitly defined as the set of points where the distance field is zero. Different forms of distance functions have various uses in Computer Aided Design and Manufacturing (CAD/CAM), computer graphics, and in other applications, such as collision detection, surface offsetting, path planning, rendering and shape morphing.

Let S be a closed 3-dimensional C^0 manifold embedded in $\mathbb{E}^3$ and denote the boundary of S as ∂S. We define the signed Euclidean distance function of S, $d_S(\mathbf{p})$, as the function that yields the Euclidean distance from a point $\mathbf{p}$ to the closest point in the boundary of the set ∂S,

$$d_S(\mathbf{p}) = \begin{cases} \inf\limits_{\forall \mathbf{q} \in \partial S} \|\mathbf{p} - \mathbf{q}\|_2 & \mathbf{p} \in S \\ -\inf\limits_{\forall \mathbf{q} \in \partial S} \|\mathbf{p} - \mathbf{q}\|_2 & \mathbf{p} \notin S \end{cases} \tag{1}$$

where $\|\cdots\|_2$ is the 2-norm or Euclidean norm. Alternatively we can define S from its distance function

$$S_d = \{\mathbf{p} \in \mathbb{E}^3 : d(\mathbf{p}) \geq 0\}. \tag{2}$$

Note that we define positive distances as inside the object boundary. Since S_d includes its boundary, the complement of S_d, S_d^c, is an open set which is not a permissible solid.

The signed distance function returns the distance to the boundary ∂S, and, as indicated in Eq. (1), the sign of the distance field distinguishes whether the point is inside or outside of ∂S. The signed Euclidean distance field has the property that the gradient of the distance field $|\nabla d_S(\mathbf{p})| = 1$ everywhere for objects with smooth boundary except on the medial axis where it is undefined, and for $\mathbf{p} \in \partial S$ the gradient is the surface normal vector, while elsewhere it points in the direction of the surface. We can define

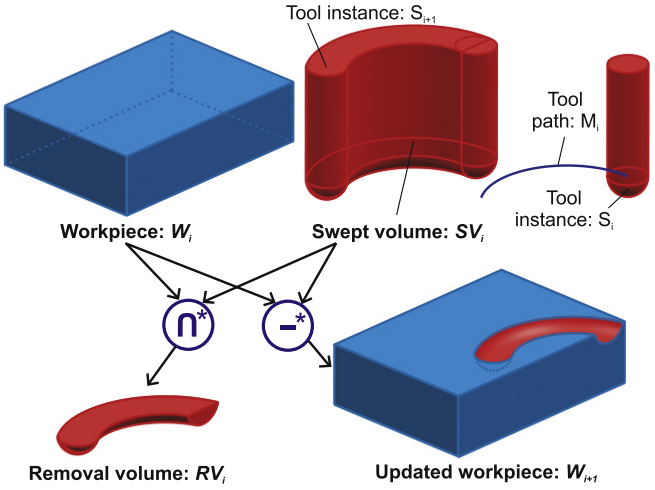

Fig. 1. Calculation of the removed volume and milled workpiece from a single tool motion in NC milling simulation using regularized Boolean operations.

the foot point of the point $\mathbf{p}$ as the orthogonal projection of the point onto the surface,

$$\mathbf{p}_{\text{foot}} = \mathbf{p} - d_S(\mathbf{p})\nabla d_S(\mathbf{p}). \tag{3}$$

The ability to directly find a point on the surface is a very useful property as shall be shown.

3.1. Distance fields of swept tools

In this section, we develop a mathematical formulation for computing the swept volumes of general surfaces of revolution in $\mathbb{E}^3$. In this paper, our focus is on NC machining and thus surfaces of revolution. In conventional NC machine tools, the path of a milling tool is controlled by *G*-code instructions which represent a set of tool motions for cutting the workpiece. In our method, the in-process and final workpiece is obtained by performing a Boolean difference operation to remove the volume swept by the cutter following the tool-path. Fig. 1 illustrates the process. A cylindrically symmetric milling tool at initial position S_i moves to final position S_{i+1} along tool path M_i and removes any part of the initial workpiece W_i within the swept volume SV_i. The Boolean intersection of the swept volume of the tool with the workpiece is the removed volume RV_i associated with this tool sweep, and the Boolean difference is the updated workpiece W_{i+1}. The final finished workpiece is shown by W.

Sweeping an arbitrary set of points S along a motion M in a d-dimensional Euclidean space $\mathbb{E}^d$ is usually formulated as an infinite union operation expressed formally as

$$\text{sweep}(S, M) = \bigcup_{q \in M} S^q, \tag{4}$$

where S^q denotes the set S positioned according to a configuration q of motion $M(t)$, and $t \in [0, 1]$ is the time-like parameter of the motion within a normalized interval. In this paper we focus on the case where set S is a 3-dimensional cylindrically symmetric tool moving in a 3-dimensional Euclidean space $\mathbb{E}^3$. We assume that motion M is a one parameter family of rigid body transformations $M(t)$, with $t \in [0, 1]$. Rigid body transformations in $\mathbb{E}^3$ can be represented using homogeneous coordinates by (4×4) matrices, where the transformations will be defined in terms of time dependent 3-dimensional translation vector, $T(t)$ and a time dependent orthogonal 3-dimensional rotation matrix, $R(t)$ as shown.

$$M(t) = \begin{bmatrix} R(t) & T(t) \\ 0 \cdots 0 & 1 \end{bmatrix}. \tag{5}$$

For any closed cylindrically symmetric set S moving along a rigid body motion $M(t)$ as seen in Fig. 2(a), the distance from a point $\mathbf{P}$ in the space to the boundary of swept volume $\text{sweep}(S, M(t))$ is defined as

$$\text{dist}(\mathbf{P}, \text{sweep}(S, M(t))) = \inf_{\mathbf{q} \in \partial \text{sweep}(S, M(t))} \|\mathbf{q} - \mathbf{P}\|. \tag{6}$$

From Eq. (6) we see that finding the distance field of a swept volume requires computing the envelopes of the swept volume. This procedure is easy for simple tools such as ball-end mill tool moving along linear paths, however it is much more complicated for general tools and motions.

The first reason for the difficulty is the changing geometry of grazing points. The grazing points are the subset of the boundary points of the swept object that are tangent to the motion at any time during the motion. For general motions, the grazing points are not fixed on the moving object, but change with time. [34,35] presented closed-form solutions of the swept profile of the most commonly used cutters for the simulation of 5-axis machining. Most of the methods in the literature sample the sweeping trajectory and compute the grazing points for the intermediate configurations by using numerical or analytical methods. However, this sampling approach is problematic; the general boundary of a swept volume can generate self-intersections and other high frequency features, and the sampling may miss small topological features leading to surface discontinuity in the constructed swept volume. An alternative way of finding the grazing points is to use screw motions [36,37], where the grazing points are identical for all instances. The instantaneous screw axis, which is widely used in spatial kinematic analysis, can be used to solve the grazing points of a swept volume.

The second reason for the difficulty is in the approximation of the boundary of a swept volume from grazing points. Although the grazing points can be found analytically for limited motions and tools, the corresponding grazing points on adjacent tool positions are interpolated by a piecewise linear or higher order surfaces such as NURBS. The envelope surface cannot be represented by simple surfaces, however an approximation algorithm is applied to generate an envelope surface such as skinned surface from a set of profiles. This process is not reliable, because the grazing points for the consecutive tool instances may be drastically different, and the matching between them could not be found easily.

All points of S will move through space along a trajectory T determined by transformation $M(t)$ in an absolute coordinate system. The trajectory of a point Q of S can be written as

$$T_Q = \{Q^m \mid Q \in S,\ m \in M\}. \tag{7}$$

Clearly, curve T_Q contains all points of the space that will be occupied by point Q at *some* time during the motion. However, when observed from a moving coordinate system rigidly attached to the moving set S, a point P appears to describe a different trajectory denoted by $\hat{T}_P$. This *inverted trajectory* can be written as

$$\hat{T}_P = \{P^b \mid b \in \hat{M}\} \tag{8}$$

where $\hat{M}$ is the *inverted motion*, i.e., $\hat{M}(t)$ is the inverse of $M(t)$ for every value of t [38].

In order to overcome the difficulty of computing grazing points, we instead determine the distance field of a swept volume by looking at the system from a moving coordinate system rigidly attached to the moving set S.

For the same problem described by Eq. (6), consider an inverted trajectory curve $\hat{T}_P$ of test point P and a moving set S in three dimensional space. We define the minimum distance function from a point $\mathbf{y}$ on the boundary of S to the point $\mathbf{z}$ on inverted trajectory $\hat{T}_P$.

$$\text{dist}(S, \hat{T}_P) = \min_{\mathbf{y} \in \partial S, \mathbf{z} \in \hat{T}_P} \|\mathbf{y} - \mathbf{z}\|. \tag{9}$$

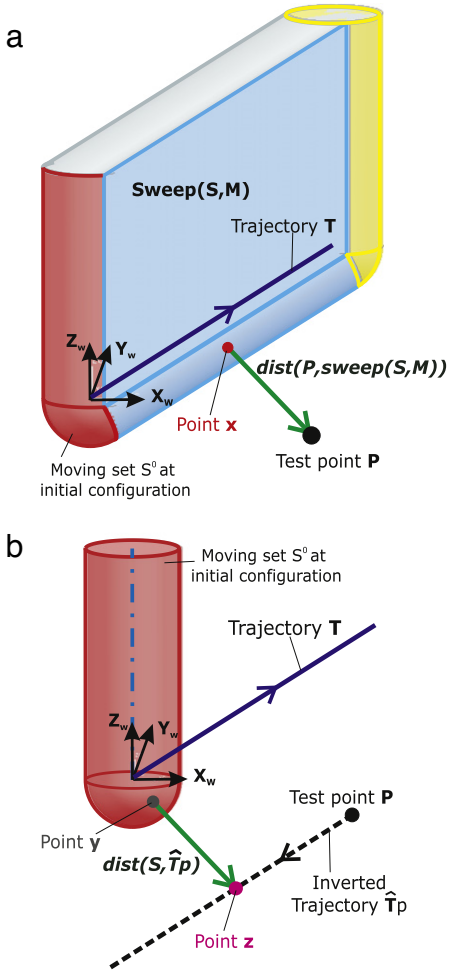

Fig. 2. The sweep of a cylindrical symmetric set S according to a linear motion; (a) The distance between the point and swept set, (b) The inverted trajectory for point P and the distance with the initial set.

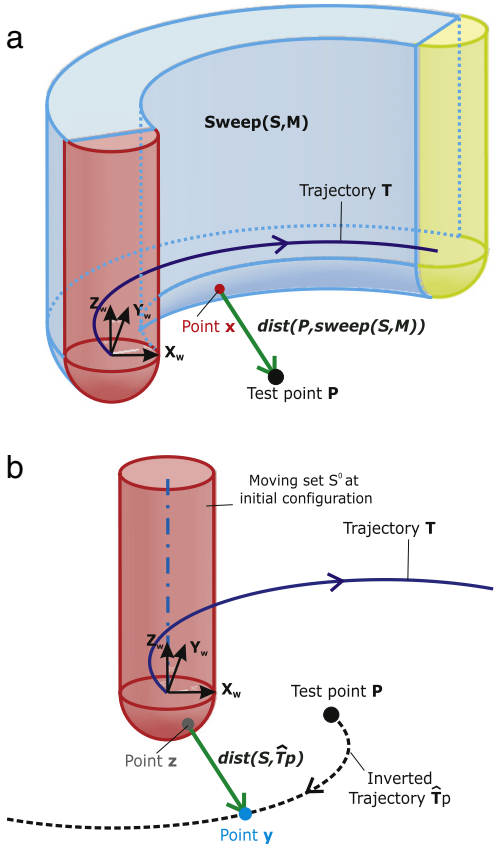

Fig. 3. The sweep of a cylindrical symmetric set S according to an arc motion; (a) The distance between the point and swept set, (b) The inverted trajectory for point P and the distance with the initial set.

If the maximum distance is reached at two points **y** and **z**, then the line spanned by **y** and **z** is normal to both sets. By using this inverted trajectory approach, the same distance can be computed without considering the envelopes of the swept set as shown

$$\text{dist}(\mathbf{P}, \text{sweep}(S, M(t))) = \text{dist}(S, \hat{T}_P). \tag{10}$$

The minimum distance can be computed by using an analytical or numerical search methods. The proposed method can be generalized to APT (automatically programmed tools) cutters with several different distance formulas, such as tapered ball-end, fillet-end and tapered fillet-end mills. According to the automatically programmed tools definition, the geometry of mill cutters can be described by certain major parameters.

In NC milling, there are generally two types of motions; 3-axis and 5-axis motions. In 3-axis milling case, the tool axis is always constant in one direction, translates in space and only rotates around its own axis, however, in 5-axis milling case, addition of two rotational axes allow to machine variety of different workpieces and motions. 5-axis capability enables the cutting tool to reach workpiece from many sides and many angles. For 5-axis NC machine tools, besides the three translational movements, the tool spindle can also be rotated along the two of three axes. Complicated tool motions are achieved by simultaneously performing these two kinds of motions, comprising three translations and two rotation motions around the two of them. For the most common tool shapes (ball-end, flat-end, taper-end mill, taper ball-end mill, conical-end mill etc.) and motion types (linear and circular arc), the distance field calculation can be performed analytically. However, for more complex tools such as 5-axis motions, a numerical approach is used where the minimum distance along the inverted trajectory is determined by conventional optimization algorithms such as Newton's method. Direct analytical solution of the minimum distance function is rather difficult; hence it is solved by employing an iterative numerical method. In the case of linear tool motion the inverted trajectory is a line segment as seen in Fig. 2, for the case of circular arc path, the inverted trajectory is an arc as seen in Fig. 3, and for the case of 5-axis tool path, the inverted trajectory could be represented by a third order spline curve as seen in Fig. 4.

3.2. Modeling of machined workpiece

Expressing the distance field of a complex object, such as a machined workpiece, in an analytic form is usually impossible. Therefore, a discrete representation is often used wherein the distance field is sampled at a set of discrete locations [39]. Regularly sampled distance fields have drawbacks because of the need to trade off size and resolution. In order to overcome these limitations, ADFs (adaptively sampled distance fields) have been proposed [40] to adaptively sample the distance field at the vertices of an octree bounding volume hierarchy. In this approach octree cells are sub-divided when the reconstructed surface error exceeds a predefined tolerance. ADFs are a practical representation of solids that provide high quality surfaces, efficient processing, and a reasonable memory footprint in order to minimize number of distance evaluations and reduce storage requirements.

In standard Constructive Solid Geometry (CSG), basic Boolean operations such as union, intersection and difference (akin to

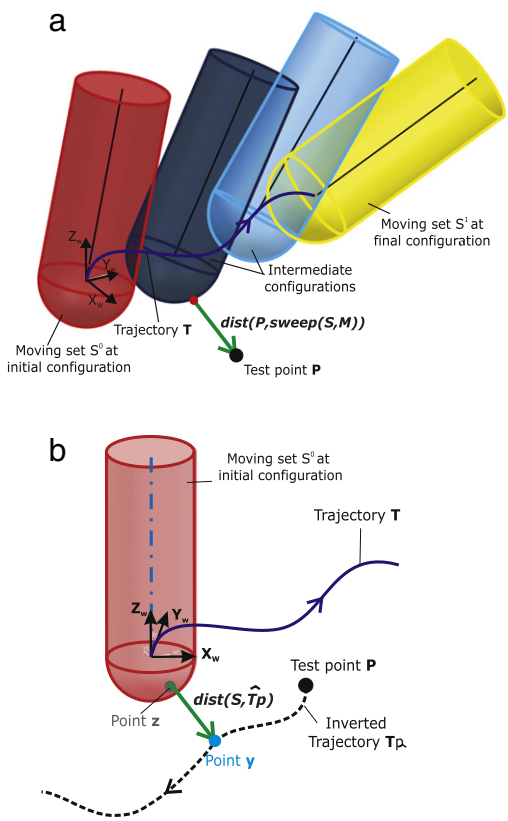

Fig. 4. The sweep of a cylindrical symmetric set S according to a 5-axis motion; (a) The distance between the point and swept set, (b) The inverted trajectory for point P and the distance with the initial set.

Table 1
CSG Operations on distance fields.

Operation	Symbolic representation	Combined distance
Intersection	dist($A \cap B$)	min(dist(A), dist(B))
Union	dist($A \cup B$)	max(dist(A), dist(B))
Difference	dist($A - B$)	min(dist(A), −dist(B))

set-theoretic operations) on solid objects can be used to define complex shapes and features, as well as to model and simulate manufacturing processes. This concept can be extended to solids represented by scalar distance fields [41]. In order to construct a Boolean expression for distance fields which is strictly Euclidean, additional halfspaces need to be introduced at the edges and vertices where two sets meet. However, current algorithms are unable to robustly evaluate the intersections or handle degenerate cases for general models [42]. As a result, our goal is to develop good approximation method. Instead of imposing strictly Euclidean distance values outside or inside the solid, we perform Boolean combinations simply by using min() and max() operators as shown in Table 1. Although Boolean combinations of Euclidean distance fields using these operators is no longer strictly Euclidean, the combined distances are Euclidean everywhere on the boundary of surface which is fine to represent the surface accurately.

The octree-based ADF representation used in [43] is well suited for modeling a shape edited using CSG. The CSG operations in Table 1 can be used to modify the sampled values in an ADF depending on the distance field of a milling tool. As the simulated tool moves along the milling path, the tool's distance field is sampled at the octree cell vertices and combined with the existing samples using the min() operator. The distance field is then reconstructed from the modified samples at non-vertex positions using trilinear reconstruction and compared to the exact

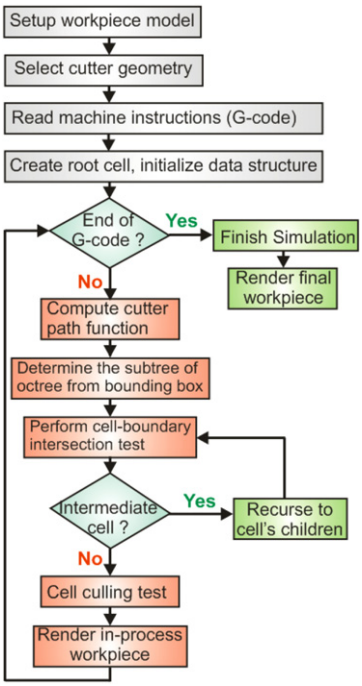

Fig. 5. The flow chart of NC milling simulation system.

values obtained by evaluating the tool distance function. If the reconstruction error exceeds the predefined threshold, and the cell level is less than a predefined maximum, the cell is subdivided (error-based subdivision).

Although conventional ADFs provide an efficient and reasonably accurate representation of a machined part, they are inadequate for high accuracy NC milling where high simulation speeds and extremely high accuracy are required. To reconstruct a 4 mm diameter spherical surface from distance field samples using trilinear interpolation with an error of 1 μm requires a cell size less than 91 μm. However, to reconstruct edge details such as the cusp between adjacent sweeps of a 4 mm diameter ball end mill can require cells smaller than 24 μm. For any reasonably large workpiece such small cells result in very deep octrees with correspondingly large memory and processing requirements.

The flow diagram of creating the machine workpiece is given in Fig. 5. This diagram basically illustrates the sequence of operations to be performed to get the solution of a machined surface representation and cutter locations.

3.3. Composite ADFs

In this work we introduce a new ADF representation which we call a *composite ADF* and describe its application to NC milling simulation. Unlike conventional ADFs, a composite ADF samples the distance field functions within octree cells to determine the minimal subset of functions whose boundaries, i.e., $d = 0$ isosurfaces, contributes to the boundary of the workpiece, the composite boundary, within a cell. Additionally, we form an implicit, rather than explicit, combination of the individual distance fields. We determine the value of the composite distance field at a point by computing the value at the point for all of the distance fields within a cell and then combine the values using the CSG relationships of their distance fields as given in Table 1. As mentioned above, the resulting composite distance field value is only exact at the composite boundary, elsewhere it is approximate. This is acceptable since the goal of this work is a highly accurate representation of the composite boundary of the workpiece.

Stated simply, the composite ADF representation consists of an octree hierarchy of cells that are either interior to, exterior to or contain the composite boundary. Each boundary leaf cell stores an array of references to distance fields of the swept volumes of the milling tools that contribute to the composite boundary within the cell. Since a given distance field may contribute to the composite boundary within many cells we keep the parameters needed to compute the distance fields in a common table and boundary cells contain only references to entries in that table. We employ locational codes [44] to efficiently perform a number of important functions of octrees, such as point location (identifying the cell that contains a point), or neighbor finding.

$$\begin{aligned} &d_i(\mathbf{p}) = 0 \qquad d_j(\mathbf{p}) > 0 \\ &\{i \mid i \in I \text{ and } I \subseteq [1, \ldots, N]\} \\ &\{j \mid j \in [1, \ldots, N] \text{ and } j \notin I\} \end{aligned} \tag{11}$$

where N is the number of distance fields. In other words a point $\mathbf{p}$ is on the boundary of the composite distance field iff it is on the boundary of one or more distance fields and inside the rest of the distance fields.

4. Detail directed adaptive subdivision

Milling programs for free-form surfaces typically consist of a very large number (up to millions) of milling instructions that are each associated with a distinct distance field. In order to vastly improve geometric operations such as editing the workpiece and rendering, we use an octree bounding volume hierarchy to adaptively spatially localize the distance fields that form the boundary of the workpiece surface. By using an octree to spatially localize the distance fields forming the composite surface geometric operations can be performed within octree boundary cells that contain only a small fraction of the total number of distance fields in the surface representation. This provides substantial performance gains resulting in a practical and very accurate simulation system.

The degree of octree subdivision is determined by the local complexity of the composite boundary as determined by the local density of distance field boundary regions. Octree cell subdivision in *composite ADFs* is driven by a count based subdivision rule: a cell is subdivided if it contains more than a specified maximum number of distance field references and its level is less than the maximum octree depth. To avoid the loss of information we allow the smallest octree cells to contain references to an arbitrary number of distance fields. As will be explained in greater detail in Section 7, the performance of the system has only a weak dependency on the maximum number of distance fields per cell. We have found that using a maximum number of 4 distance fields per cell yields good results.

Fig. 6 shows an simple example where a rectangular solid is formed by the intersection of 6 planar distance fields whose boundaries are shown in light blue. Each cell is limited in this example to containing maximum 2 distance fields which causes root cell to be subdivided into 8 children. The top surface has been milled by 3 horizontal sweeps of a ball end mill shown in red, dark blue and green driving further subdivision of some boundary leaf cells.

5. Generation and update of CADFs

To perform milling simulation using the *composite ADF* representation we first construct a composite ADF of the initial workpiece, usually from simple primitives such as planes or cylinders. Then for each tool motion, we determine from its bounding box a sub-tree of the octree whose cells are potentially altered by the tool. For each cell in the sub-tree we first perform cell/boundary intersection testing, where it is determined whether the cell contains a portion of the boundary of the swept tool's distance function, or is entirely outside the boundary. Any cell that is entirely outside of the boundary of the shape instance's distance function is also outside of the composite surface and so the cell's type is changed to exterior leaf cell and any data or children it contains are freed. If the cell is not entirely outside of the boundary of the swept tool and the cell is an intermediate cell, then cell/boundary intersection testing recurses to the cell's children.

For each interior or boundary leaf cell that contains a portion of the boundary of the swept tool, a cell culling operation is performed. During cell culling the set of distance fields that have a portion of their boundaries within the cell is refined to the minimal set of distance fields that have a portion of their boundary that is part of the composite boundary. References to any distance field that does not contribute to the composite boundary within the cell are removed from the cell's data structure. This may include the new distance field as well as any or all of the previous distance fields. A cell whose minimal set is the $\emptyset$ set is an exterior cell, and so its type is changed and any data it contains are freed.

In the third step, once the minimal set of distance fields is found a cell will be subdivided if its level is greater than the maximum depth of the octree and the number of distance field references in its data structure are greater than a maximum number, e.g. 4 surfaces. Each new child cell is edited by the distance fields of its parent cell using the same process as just described. We now describe both of these operations in greater detail.

5.1. Cell/boundary intersection testing

In order for a distance field to contribute to the composite boundary of the workpiece within a cell C it is a necessary, but not sufficient, condition that some portion of its boundary enters the cell. Therefore a cell/boundary intersection test is performed in three phases: a coarse test, a vertex test and then a face/edge test. Formally we seek to determine

$$D_c \subseteq D \quad \text{s.t.} \forall d \in D_c : C \cap \partial S_d \neq \emptyset \tag{12}$$

where D is the set of all distance fields.

The initial coarse intersection test is performed by computing the value of the distance field at the cell center $d(\mathbf{p}_c)$ and comparing it to the cell size, c_s. If $d(\mathbf{p}_c) < -c_s$, the cell is entirely outside of the swept volume and unaffected by it. If $d(\mathbf{p}_c) > c_s$, the cell is entirely within the swept volume of the tool, and hence the cell is completely outside to the workpiece surface. Therefore, the cell's type is changed to exterior and references to any distance fields within it are deleted. If the cell's type is intermediate, then its children are recursively deleted before the cell is changed to exterior.

If $-c_s \leq d(\mathbf{p}_c) \leq c_s$, cell/boundary intersection testing proceeds to the vertex test where the distance field is computed at each cell vertex and its sign is determined. If the distance field at two vertices has opposite signs, the boundary of the swept volume intersects the cell.

The absence of a distance field sign change between cell vertices does not preclude the possibility of the distance field boundary intersecting the cell; it is still possible that the boundary intersects a face or edge of the cell. Therefore, cell intersection testing precedes the final iterative phase where we search along the direction gradient for a change in sign. Starting from the distance field magnitude and gradient at the cell vertices obtained in the preceding phase, the distance field is iteratively computed using the update rule

$$p \leftarrow p - (d_S(\mathbf{p}) + \epsilon)\nabla d_S(\mathbf{p}) \quad \text{s.t.} \, p \in C \tag{13}$$

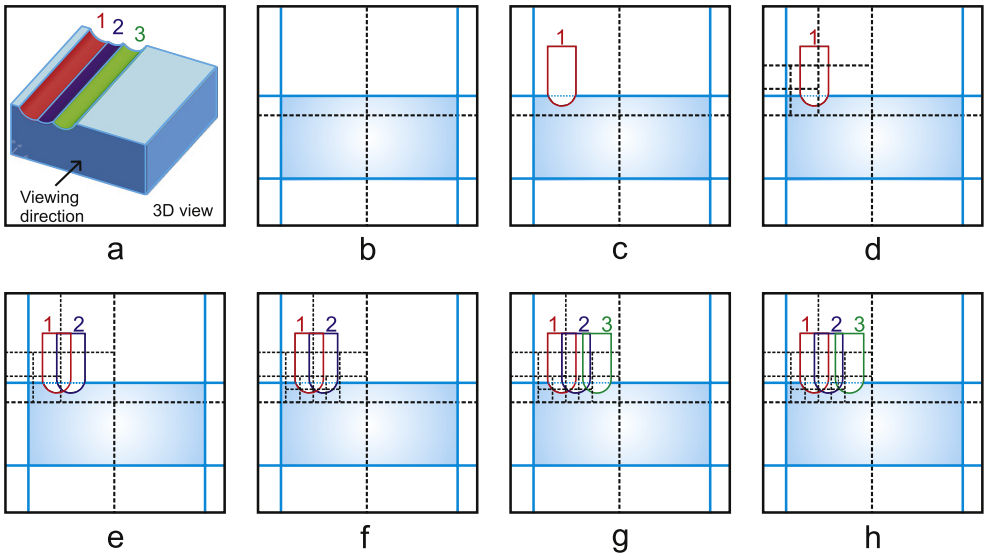

Fig. 6. An octree bounding volume hierarchy is formed using count-based subdivision. The cells is subdivided when the number of distance fields within it is greater than two: (a) shows the swept volumes in 3D, (b)–(h) shows a sequence of milling operation that result in the addition of distance fields and subdivision of cell in consecutive order.

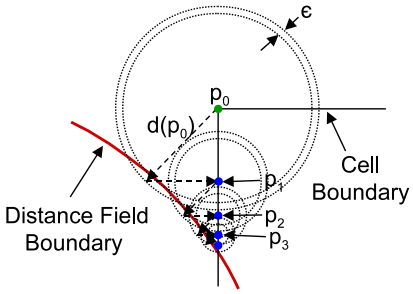

Fig. 7. Cell intersection test. The value and gradient of the distance field are computed at a test point initially located at the cell vertices and then used to move the test point so that it cross the cell boundary subject to the constraint that the test point remain in the cell.

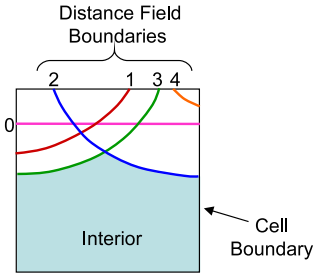

Fig. 8. Cell culling eliminates distance field boundaries that do not contribute to the composite boundary.

where ϵ is a small positive number. Eq. (13) is very similar to Eq. (3) for the foot point of a distance field except that each step is slightly greater than the value of the distance field. This is because we do not wish to find a point on the surface, but instead to have a sign change that indicates that the boundary enters the cell. Additionally, this keeps the limited numerical precision of floating point numbers from leading to extra operations.

Enforcing the constraint that the test point is within the cell usually leads to the use of an iterative procedure. It is possible that the updated point location might cross the boundary in a single step. However, it is more likely that the constraint will prevent crossing the boundary in one step and multiple steps will be required. Therefore, we iteratively compute an updated test point from the previous point, apply the cell constraint, compute the distance field at the constrained point, and then test for either a distance field sign change or for stalling of the point update, i.e. $\|\mathbf{p}' - \mathbf{p}\| < \delta$, where δ is a small cutoff value. Fig. 7 gives an illustration of the algorithm where repeatedly applying the algorithm results in a sequence of test points p_1 to p_3 before p_4 crosses the distance field boundary.

5.2. Cell culling

Once it has been determined that the boundary of the distance field of the swept tool enters the cell, it is then useful to determine whether (a) the boundary contributes to the composite boundary within the cell, and if so (b) are there any distance fields that are currently referenced within the cell that now no longer contribute to the composite boundary. Unless the minimum cell size is very small there is the strong likelihood that many distance field boundaries will pass through the cell without contributing to the composite boundary. For example, consider the case illustrated in Fig. 8 where we see a cell that contains the boundaries of 5 distance fields labeled 0 through 4. Label 0 corresponds to the original flat planar surface of the workpiece. It is clear that although all of these distance field boundaries enter the cell, only two of them (2 and 3) actually contribute to the composite boundary. The others may have contributed at an earlier point in the simulation, but are now not only unnecessary but also degrade system performance by needing to be processed during geometric operations.

Therefore, the next step in editing the workpiece representation involves determining whether a new distance field should be added to the cell's edit data, and also whether any existing distance fields should be removed, a process called cell culling. More formally given a set D_c of distance fields whose boundaries enter the cell we wish to find a minimal subset of distance fields, $D_{c,\min}$ such that

$$D_{c,\min} \subseteq D_c \quad \text{and} \quad \bigcap_{m \in D_c} (S_d^c)^m = \bigcap_{n \in D_{c,\min}} (S_d^c)^n \tag{14}$$

where ${S_d}^c$ is the complement of the distance field S_d. S_d^{cm} and S_d^{cn} denote the set ${S_d}^c$ at elements m and n of sets D_c and $D_{c,\min}$ respectively. Basically, the intersection of the complement of the distance fields within the cell should be the same after the culling

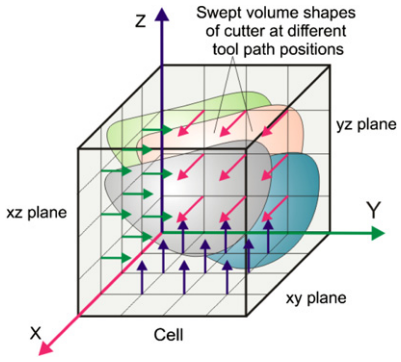

Fig. 9. Rays are propagated from the x, y and z faces of the cell for culling operation.

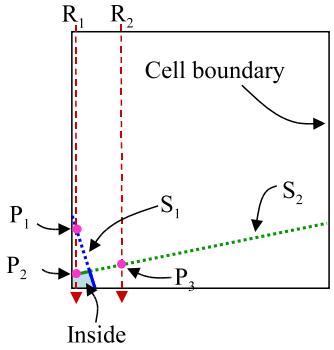

Fig. 10. The portion of a distance field boundary within a cell can be arbitrarily small. If culling fails to retain surface S_1 then the surface reconstructed with the cell will only consist of S_2 which is incorrect.

operation. The topology of the cell after the culling should not be altered.

For a simulation system where the tool shapes and motions are severely restricted such that all swept surfaces are low-order (e.g. ball end mill with linear motion) an analytic approach to cell culling can be used. However, in the general case that includes the wide variety of tools and 5-axis motions commonly found in NC milling a purely analytic approach is not possible. Therefore, to find the minimal set D_c, min we use a ray-sampling approach; we sample ∂W_{i+1} using rays that originate at the cell face and propagate perpendicular to it until they intersect either ∂W_{i+1} or the opposite cell face. Since ∂W_{i+1} is defined implicitly as the intersection of the sets bounded by the individual distance fields within the cell, sampling ∂W_{i+1} with a ray consists of computing the intersection point of the ray with each distance field in D_c and for each intersection point computing the value of the other distance fields. A ray intersection point is on ∂W_{i+1} if it passes the surface test given by Eq. (11). For each sampling ray that results in one or more points on ∂W_{i+1} a distance field in the set D_c is recognized to contribute to ∂W_{i+1}. After all ray sampling is completed, any distance field that is not found to contribute to ∂W_{i+1} is removed from D_c.

To avoid an orientation bias rays are propagated from the x, y and z faces of the cell subject to an orientation test as follows as seen in Fig. 9. During cell intersection testing a rough determination of the dominant direction of the boundary of the shape instance is found by finding the maximum values g_i of the absolute values of the components of the distance field gradient at the cell vertices

$$g_i = \max(g_i, |\nabla_i d(\mathbf{p}_j)|) \quad i \in [x, y, z]\, j \in [1, 8] \tag{15}$$

where ∇_i is the i-th component of the gradient and $\mathbf{p}_j$ are the vertex positions. If the maximum value in a particular direction is less than an empirically determined predefined tolerance rays are not propagated from the corresponding face. We have found that a tolerance of 0.3 accelerates culling significantly without a loss of accuracy.

Our current approach to ray sampling is very simple: rays regularly sample ∂W across the area of the cell from 3 directions in a uniform 5×5 square grid with a spacing of $c_s/4$ where c_s is the cell size. We find that a common minimum spacing of milling passes during surface finishing is roughly 100 μm allowing minimum cell size of 300 μm to hold on the order of 9 distance fields, a number that is reasonably well sampled by 25 rays per direction.

Culling is only applied to minimum size cells; a reference to any distance functions of swept tool that passes the cell intersection test is simply added to larger cells. This is because culling is an expensive test whose purpose is to limit the maximum number of distance field references in a cell to only those that contribute to the composite surface. Due to the use of count-base subdivision, cells larger than the minimum size are guaranteed to contain no more than the maximum count and therefore do not benefit significantly from culling. Not culling larger cells does have the affect of driving more cells to the minimum size, but we have found this to not result in a loss of performance.

An obvious concern in any sampling method is to ensure that the sampling rate is high enough to accurately reconstruct the signal. However, as demonstrated in Fig. 10, it is possible for the portion of the composite surface contributed a distance field with the cell to be arbitrarily small while still having a significantly effect on the composite surface. Furthermore the failure to maintain references to this distance field within the cell can result is a defect in the reconstruction of the surface that is as large as the cell. Clearly there is no sampling rate that is sufficient to guarantee accurate sampling.

To improve the quality of the culling algorithm we add a constraint derived from the fact that the distance field value at any point within the cell must not be altered by the culling process. That is, the minimum distance to the surface from any point cannot change due to culling. To enforce this constraint we compute the values of distance field at the cell vertices for each distance field whose boundary intersects the cell and combine them use the Boolean operators to determine the composite values for the cell. After the culling algorithm is used to remove distance field references from the cell, the Boolean combination of the remaining distance fields is computed. For each vertex where there is a discrepancy one of the distance fields that had been culled must be added back to correct the discrepancy. This simple technique essentially eliminates the sort of major topological errors illustrated in Fig. 10.

To accelerate culling we make use of a memory cache. Each cache entry stores, for all culling rays, the spatial position of a ray's intersection with the surface as well as the index of the swept tool on whose boundary the intersection is located. This can be represented by a hash table and we use a prime number of cache entries so that a simple Modulo hash function is sufficient. Using a modest cache of 8191 entries (11.7 MB for 75 culling rays) we see a 96.1% hit rate and an increase in culling performance by approximately a factor of 6.

Typically culling is applied in a continuous fashion; the culling algorithm is executed on minimum level cells every time they are edited. This enables optimized rendering of the *composite ADFs*while the milling simulation runs, a benefit for real-time monitoring of the milling process. However, it is often the case that the operator does not want to watch every step of the simulated milling and instead wants to see the final result as quickly as possible. In this case we can obtain a significant increase in speed by recognizing that there can often be many boundary leaf cells that exist at intermediate times in the simulation, but later become exterior as the milling proceeds deeper into the workpiece (temporary cells); any time spent culling these cells

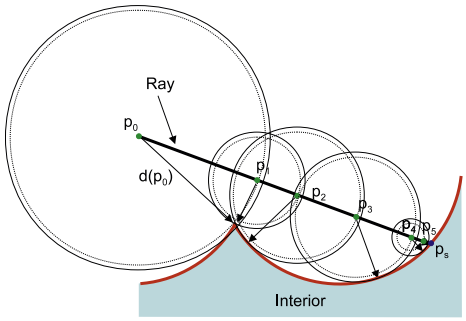

Fig. 11. Ray casting of a distance field is handled by sphere casting where steps are taken along the ray equal to the value of the distance field at each point.

will be wasted. Therefore, a fast-to-final mode is provided where the milling program is simply run backwards and culling of cells is deferred until after editing is completed. Fast-to-final editing provides very high simulation rates for rough cutting operation. For example, the Dome workpiece shown in Fig. 18 takes 103 min to actually cut at a feed rate of 2000 mm/min whereas the simulated milling in fast-to-final mode can be completed in 6.5 s, 952× faster. Fast-to-final editing provides much less benefit for finish milling where there are few temporary cells.

6. Rendering

6.1. Rendering of distance fields

The rendering of distance fields has been handled by many methods, including both indirect and direct methods. In indirect rendering an explicit secondary representation is first determined from the distance field boundary and then the secondary representation is rendered. Indirect rendering methods include point rendering [43] where the distance field boundary is sampled with points and the points are then rendered, as well as conversion to polygonal representations by marching cubes [45] or surface nets [46]. Direct methods include ray casting, where a ray is numerically propagated from its origin until the boundary of the distance field is reached. This is essentially a root finding problem in which we seek the value of the time-like coordinate t of a ray $\mathbf{r}(t) = \mathbf{r}_0 + \mathbf{r}_d t$, where $\mathbf{r}_0$ and $\mathbf{r}_d$ are the ray origin and direction respectively, such that $d(\mathbf{r}(t)) = 0$.

Common root finding methods, such as Newton's method or Regula-Falsi [47] can be used, but in most cases the distance field can have local minima before the root so that convergence to global minima can be difficult to guarantee. A more robust approach that is well suited for ray casting distance fields is sphere tracing [48] which uses the value of the distance field to move along the ray in steps small enough to be guaranteed not miss the surface. A ray is assumed to intersect the surface when the distance to the surface becomes smaller than a predefined tolerance $d(\mathbf{p}) < \epsilon$. Convergence rates can be very good except when the ray passes close to the surface as shown in Fig. 11.

6.2. Rendering of composite ADFs

To render composite distance fields our system employs either image-order ray casting [41] or a variant of hybrid ray casting [49,50]. In hybrid ray casting each octree boundary leaf cell is rendered into a frame buffer independently and their results are combined together using a Z buffer. As shown in Fig. 12, during an initial object-order phase the vertices of a cell are projected into screen space using the standard modelview, projection and viewport matrices. A rectangular screen region that tightly bounds the projected vertices is determined and clipped against the

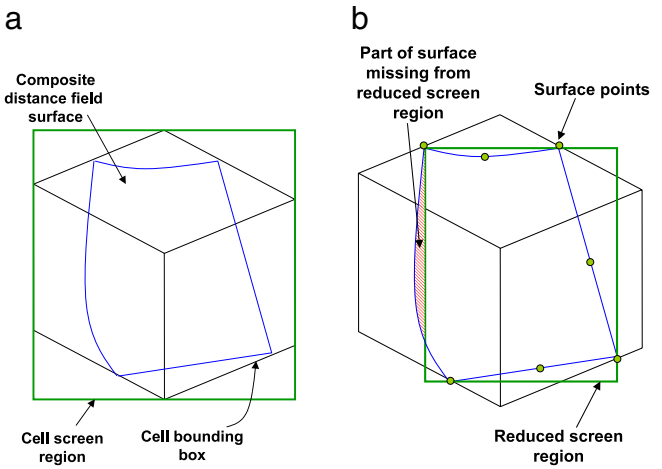

Fig. 12. The screen region for rendering the *composite ADF* within a cell; (a) The rectangular cell screen region, (b) A reduced screen region.

viewport. If the cell is an intermediate cell and its clipped screen region is not NULL, the algorithm recurses to the cell's children.

Boundary leaf cells are then rendered using image-order ray casting by propagating rays from the pixels within the rectangular screen region. The intersection of each ray with the set of distance fields within the cell is computed and the intersection point with the minimum ray coordinate that satisfies the surface test in Eq. (11) is the surface point whose color and brightness are computed using a lighting model such as Phong. Additionally the depth of each ray intersection is used to correctly combine contribution from overlapping cells using a z-buffer.

We improve on the conventional hybrid ray casting algorithm by recognizing that it is common for the screen region enclosing the projections of the cell vertices to be much larger than the projection of the composite ADF boundary within the cell as illustrated Fig. 12(a). Therefore, we use an algorithm very similar to cell/boundary intersection testing to iteratively move test points from the cell vertices onto the boundary of the composite ADF. The tolerance for considering a point to be on the composite surface is deliberately relaxed for this process as a balance between the need for increased number of iterations that results from a tight tolerance, and an excessively large screen region for resulting from loose tolerance. Once the test points have approximately reached the composite surface, their positions are projected to the screen and the reduced rectangular screen region bounding these points, as shown in Fig. 12(b) is used during the succeeding image-order rendering phase.

Using a reduced screen region dramatically improves rendering time, but at the risk of missing a part of the projected boundary that falls outside of the smaller region as indicated in Fig. 12(b). Uncorrected this will result in image artifacts such as holes. To avoid image artifacts we require that there be no ray/surface intersection along the boundary of the screen region. If there is, the screen region is grown by one row or column in the corresponding direction. This region growing process continues until there are no intersections along the region boundary.

When rendering during editing a further performance gain is obtained by only re-rendering those cells that have been edited. In most cases this provides a significant reduction in rendering time.

Both hybrid and image-order ray casting rendering methods are used since both have various strengths and weaknesses. These will be discussed in Section 7.

7. Results

To demonstrate the capabilities of our approach to 3-axis milling simulation we have simulated the fabrication of a

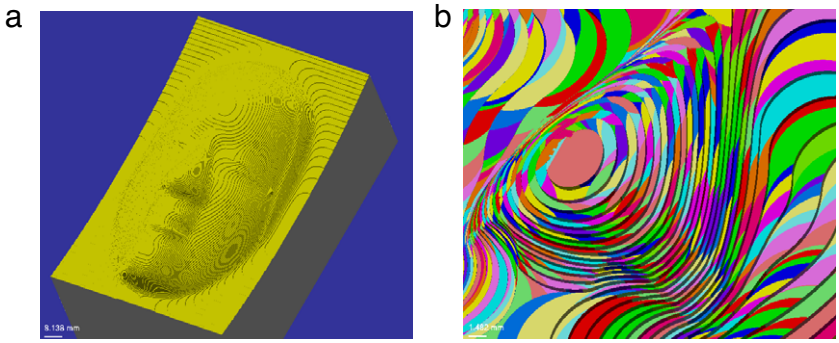

Fig. 13. Image of a simulation of the rough cutting of a Japanese Noh mask. (b) Close-up false color image of the nose where each color corresponds to the patch of the surface contributed by a different distance field.

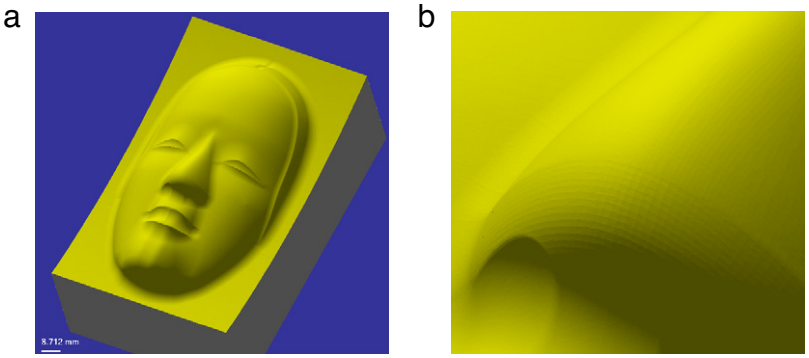

Fig. 14. (a) Image of the whole finished Japanese Noh mask simulation and (b) a close-up simulation image of the nose of the simulated workpiece.

traditional Japanese mask called a Noh mask whose milling requires more than 700,000 cutter locations. The results are shown in Table 2. The initial workpiece is a rectangular solid that is 90 mm × 145 mm × 85 mm. The simulation was performed using a single core of a 3 GHz Intel Core 2 Quad with 4 GB of DRAM. The maximum depth of the octree is 9, corresponding to a 333 μm minimum cell size, and the maximum number of distance fields per cell is 4, except for the smallest cells as noted previously. Images were rendered by software at an 800 × 600 pixel resolution with the rendered surface filling the window. Simulated milling times for the 3 milling programs are, 560×, 9.5× and 21×, respectively, faster than actual cutting times for the given cutting conditions, and the memory requirements are very modest.

The result of the rough milling operation is shown in Fig. 13(a). Fig. 13(b) shows a false color close-up of the nose of the mask. In this image each patch of the surface that corresponds to a distinct distance field boundary of the swept tool is rendered in a different color (due to the use of a color table with 32 entries some distance field boundaries have the same color). The ability to maintain an exact correspondence of each part of the milled surface to a particular milling instruction (cutter locations from NC program file) is invaluable for correcting defects in the milling program. In contrast to standard ADFs, the very sharp edges of each cut reconstructed by composite ADFs should be noted. Fig. 14(a) shows a image of the whole finished mask, and (b) is a close-up of the nose.

Fig. 15(a) and (c) show an image of the simulated Noh mask and photograph of the real machined Noh mask, respectively. Fig. 15(b) and (d) show close-up images of the simulated and real Noh mask, respectively, within the black box indicated in Fig. 15(a). The simulated shape of the milled surface agrees extremely well with the actual shape and replicates fine details down to a scale of approximately 50 μm. Below this limit the dynamics of the machining process such as tool deflection due to cutting forces, tool chatter, tool runout, thermal effects, machine dynamics, etc., become significant. Since the current generation of this simulator only performs a geometric simulation of NC milling, the resultant simulated workpiece does not include the contributions of the dynamics. Additionally, we treat the tool as cylindrically symmetric when in fact it is not. The existence of discrete cutter flutes leads to a pattern of cycloidal cutter marks that can been seen in the close-up photograph but are not included in the simulation. However, despite these shortcomings, which are also not handled by any known simulator, the very high quality of the simulation is readily apparent from the images.

Table 2
Noh mask simulation results.

Operation	Roughing	Finish 1	Finish 2
# Cutter locations	44,742	64,541	524,721
Length of path (m)	236	36	359
End mill type	Flat	Ball	Ball
Tool dia. (mm)	10	6	6
Feed rate (mm/min)	2000	1000	1000
Cutting time (min)	118	36	359
Sim. time (min)	0.21	3.8	16.9
Sim. memory (MB)	11.1	15.8	47.3
Render time (ms)	635	856	1200
# Total cells	319,889	528,393	550,809
# Boundary cells	141,732	212,346	210,703

In order to demonstrate the capabilities of 5-axis milling simulation, we have simulated the fabrication of impeller which is one of the most complex 5-axis machining task. The initial workpiece is a cylindrical solid and all the milling stages require around 650,000 milling instructions. The same simulation environment and conditions are applied as in the case of above 3-axis milling examples, and it takes 12 min to complete all the stages of simulation by ball-end mill at different diameters. The results are shown in Fig. 16 together with the rough and finish cutting. Finishing of the impeller blades requires high accuracy

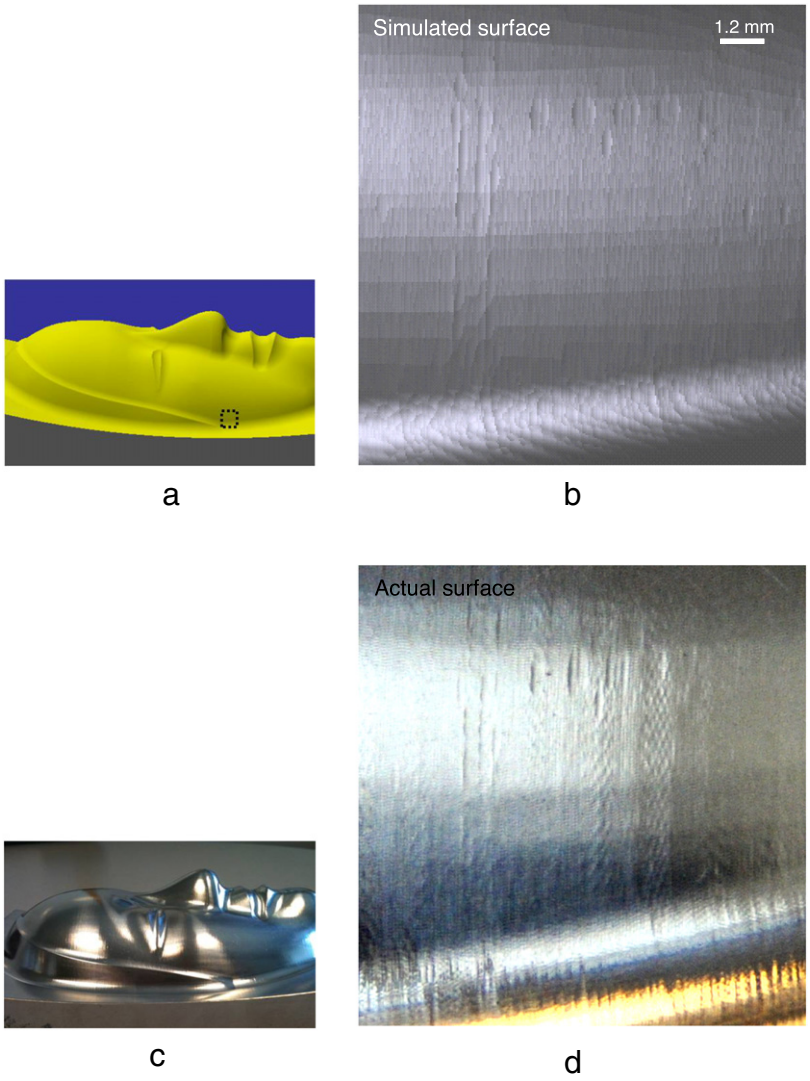

Fig. 15. (a) Image of the finished simulation of mask part, (b) Close up image of the simulated part within black box indicated in (a), (c) Photograph of the actual finished mask and (d) Close up photograph of the actual part within black box indicated in (a).

and smooth axis movements to avoid collision and to achieve good surface quality. Fig. 16(d) shows a false color close-up of the impeller blade surface. In this image each patch of the surface corresponds a distinct distance field boundary of swept volume of 5-axis tool motions.

As a test of the accuracy of this approach to milling simulation we simulated the milling of a cubic solid by a 4 mm diameter ball end mill. The milling program consisted of a set of parallel linear sweeps of the tool with a separation of 100 μm. Each sweep results in a cylindrical cut with a 2 mm radius, and the height of the cusps between each sweep should be 0.625 μm. Fig. 17 shows a plot of a row the z buffer of a close-up image (solid blue line) as well as a plot of the height of a 4 mm diameter circle (dashed red line) for comparison. The shape of the milled surface agrees extremely well with the expected shape and the cusp height is correct to within approximately 4 nm which is the limit of floating point precision in this implementation.

It is important to point out that accuracy of the simulated surface and maximum workpiece size are not related. The high accuracy of the system results from the use of analytic or procedural distance fields whose zero level isosurface can be found with an accuracy limited only by the precision of floating point numbers. The maximum workpiece size that can be simulated is limited only by the memory of the computer system on which the simulation runs. Since new 64-bit computers can use more than 100 GB of memory, this limitation is not significant.

To evaluate the quality of the culling algorithm the simulation was performed with the number of culling rays set to 25, 225, 361, and 625 rays per direction. A close-up image of the two areas of the Noh mask in the resulting representation was rendered and the z buffers were compared to determine differences in height. We find that there are no differences in height between the case of 361 rays and 625 rays indicating that the minimal set has been obtained. When we compare the default case of 25 rays to the case of 625 rays we find that there is a distribution of height differences whose mean is 47 nm and standard deviation is 162 nm. The maximum height error is 2 μm. As expected, we found that the culling time increases linearly with the number of rays.

Rendering of composite ADFs by both hybrid ray casting (HRC) and image-order ray casting (IRC) is included in the system since both offer advantages under certain conditions. Fig. 19 shows the rendering times versus pixel count for two composite ADFs of different complexities, the Noh mask with 550,000 cells and the simpler "Dome" workpiece (Fig. 18) with 77,000 cells, using either HRC and IRC. The pixel count was varied by changing the image scale such that the image size varied from invisibly small to overfilling the full (800×600 pixel) viewport. We find that IRC and HRC each possess rendering domains in which one outperforms

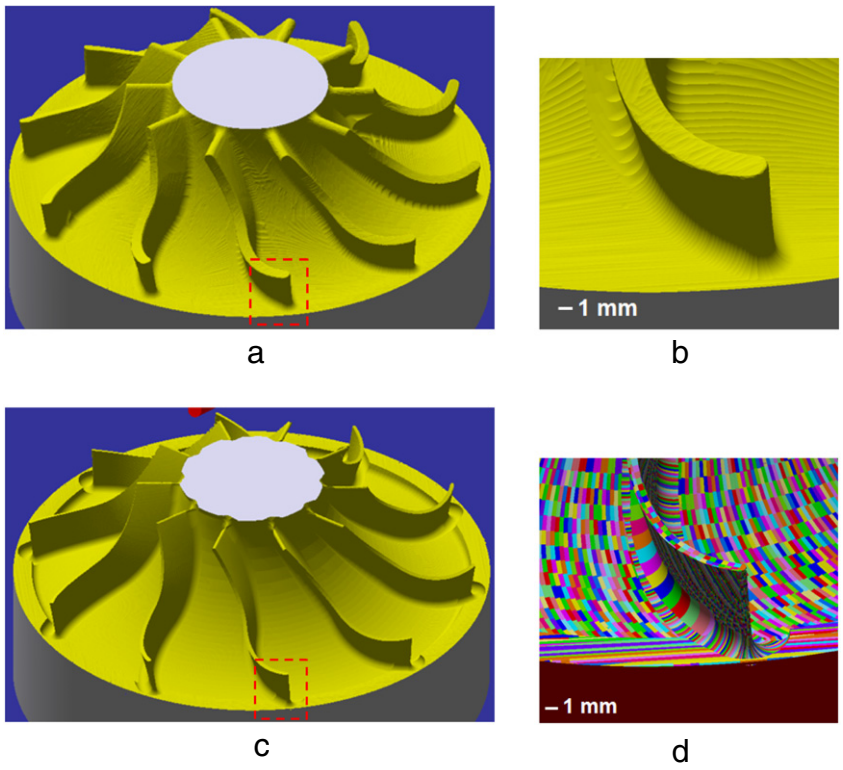

Fig. 16. (a) and (c) Image of the simulation of rough and finish cutting of impeller part respectively, (b) Close up simulation image of the rough cutting within the box indicated in (a), (d) Close up false color image of the finish cutting within the box indicated in (c).

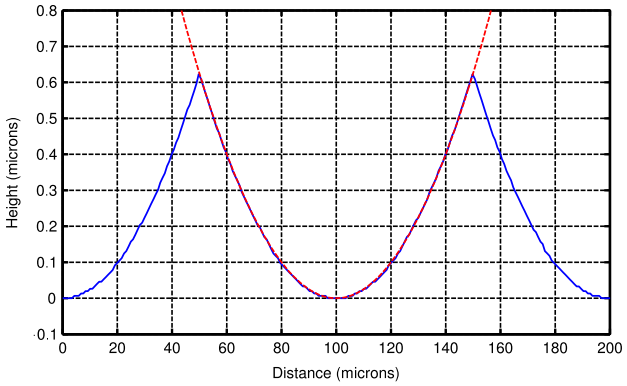

Fig. 17. Height profile of a workpiece (solid blue line) and tool (dashed red line) are shown for comparison. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

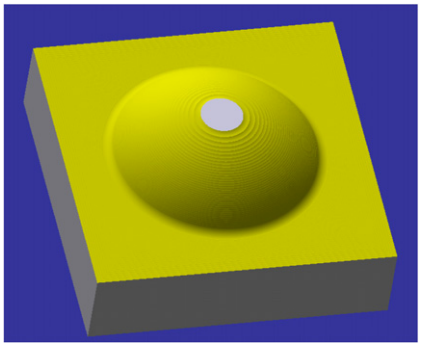

Fig. 18. The Dome workpiece has 77,000 boundary leaf cells and demonstrates the relative performance of image-order ray casting and hybrid ray casting.

the other. IRC has lower per-pixel processing costs and higher per-cell processing costs. Therefore, it provides better rendering and simulator performance. However, HRC exhibits much better CPU cache behavior despite the fact that significantly more total pixel operations (e.g. cell ray casting, Z testing, etc.) are performed. So HRC is better suited to full screen rendering that is associated with interactive rendering of the completed simulation, whereas IRC is better suited to localized rendering of the localized region during the simulation.

A current shortcoming of this approach to milling simulation is the long rendering times. The long rendering times are caused by the need to compute the ray intersection with each distance field boundary with a cell and then compute the value of all distance fields in a cell for each intersection point, an $O(N^2)$ operation where N is the number of distance fields in a cell. While increasing the maximum depth of the octree reduces the number of distance fields in the cell, it does so at a slower rate than it increases the total number of cells. Strategies for improving the rendering time are being explored. Attempts to exploit spatial coherence have not been found to be beneficial at low image scales because adjacent rays do not usually intersect the same distance field boundary.

As mentioned earlier, the performance of the system has only a weak dependence on the maximum number of distance fields permitted within a cell (maximum edit parameter). We have determined empirically that system performance is optimized by setting the maximum edit parameter to a value of 4. Too low a value results in increased memory consumption and editing times, while too high a value results in slow rendering times. This is illustrated by Fig. 20 which shows the relative memory requirements, editing time and rendering time as a function of maximum edits. A value of 4 is a reasonable compromise to maximize overall system performance.

7.1. Comparison with other approaches

In this section, we compare the performance of our system with other systems which use spatial partitioning and solid modeling

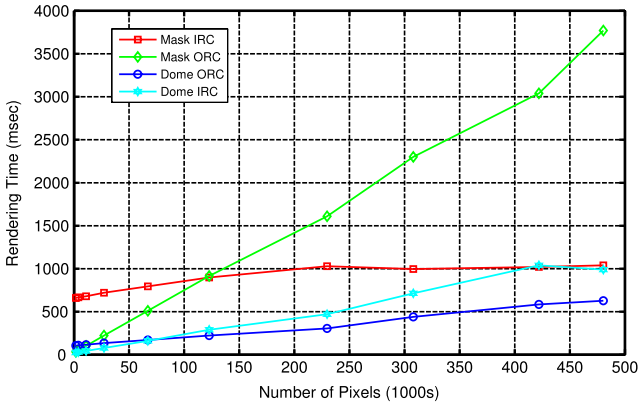

Fig. 19. Rendering time vs. number of rendered pixels for hybrid ray casting(HRC) and image-order ray casting(IRC) of Box+Sphere and Mask examples.

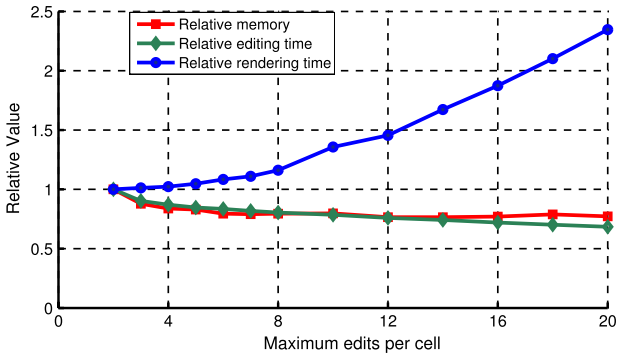

Fig. 20. Relative memory consumption, editing time and rendering time vs. the count-based subdivision maximum edit parameter.

Table 3
Comparison of our system with a voxel based system.

Part	Noh mask	Golf club
Representation	Composite ADF	Octree voxel
Dimensions (mm)	$90 \times 145 \times 85$	$161 \times 131 \times 53$
# cutter locations	634,004	251,844
Path length (m)	631	394.5
End mill type	Flat ($r = 5$ mm) + Ball ($r = 3$ mm)	Ball ($r = 3$ mm)
Tolerance (mm)	0.001	0.01
Min. cell size (mm)	0.4894	0.3333
Sim. time (min)	41.82	55.27
Sim. memory (MB)	63	700

approaches. Despite the fact that NC simulation has been studied extensively in the literature, there are few published results of NC simulation for complex models.

The first comparison is done with a voxel based system that was published relatively recent (2009). This system uses octree-based voxel representation of the workpiece and extended marching cube algorithm for triangulation [51]. We selected the most complex model, a golf club head mold, from this paper and compare it with the Noh mask example as seen in Fig. 21. We have performed the simulation on the same type of machine, 3.2 GHz Intel Pentium 4 with 1 GB of DRAM. Both examples are similar in terms of workpiece dimensions, however our example is more complicated in terms of the number of cutter locations and tool path length. Table 3 gives a comparison of the simulation time and memory space for a NC simulation using octree based voxel representation and composite ADF representation. The first to the fourth rows present the tool path and milling information. The fifth and sixth rows give information about the simulation conditions, and the last two rows are the simulation times and the required

Fig. 21. Image of the finished golf club head mold [51].

memory space respectively. By comparison, the advantages of composite ADF based NC milling simulation are clear. A great reduction in time and space with better accuracy can be achieved by our approach.

The second comparison is done against the *B*-rep solid modeler based simulation system. In this comparison, a commercially available *B*-rep solid modeling engine is used to perform 3-axis milling simulation for ball-end mill, and then comparison is done for Dome workpiece shown in Fig. 18. All the milling stages of the model are machined by ball-end milling tool, and there are around 340,000 cutter locations. The simulated milling time using the composite ADFs for the all program is only 10.2 min with 25.4 MB simulation memory and 102 MB of application memory. However, the *B*-rep based system will take in the order of hours with increasing number of geometric entities(up to 230,000) and memory (up to 700 MB). The number of geometric entities (summation of boundary faces, edges and vertices), the required memory space and the simulation time are shown in Fig. 22 respectively. In this example, only the spherical part of the ball-end mill is in contact throughout the milling program. Therefore, the resulting surfaces are planar, cylindrical and spherical. As it can be seen from the results, the performance of *B*-rep method is very slow (simulation takes more than 9 h). The larger the tool path the slower it is. One of the reason for the slowness is that the in-process model becomes more and more complicated during the material removal process, therefore traversing the tree for Boolean operations will slow down the system. Compared to the other methods in the literature, our method robustly handles the Boolean operations and provides a very good error bound (1 μm) with a low memory requirement for complex models.

7.2. Future work

A high accuracy image is not always needed. Therefore, methods of indirect rendering are also being pursued. Generating and rendering a lower resolution triangle mesh (one triangle per cell) using Marching Cubes or Surface Nets will enable reasonable interaction rates during navigation with a high resolution image generated at completion.

In this paper, We showed our results for the most commons tools (flat-end mill and ball-end mill) for 3-axis milling. Other tools can be easily added to our simulation. We are currently in the process of developing a 5-axis milling simulation capability for general tools and complex motions.

Currently all editing and rendering is being performed using a single CPU core without use of a GPU. We are in the process of developing a multi-core version of the system and early indications are promising. Development of a GPU implementation is more challenging due to the nature of our current editing algorithms, but this is an area of ongoing effort. The GPU is better suited to

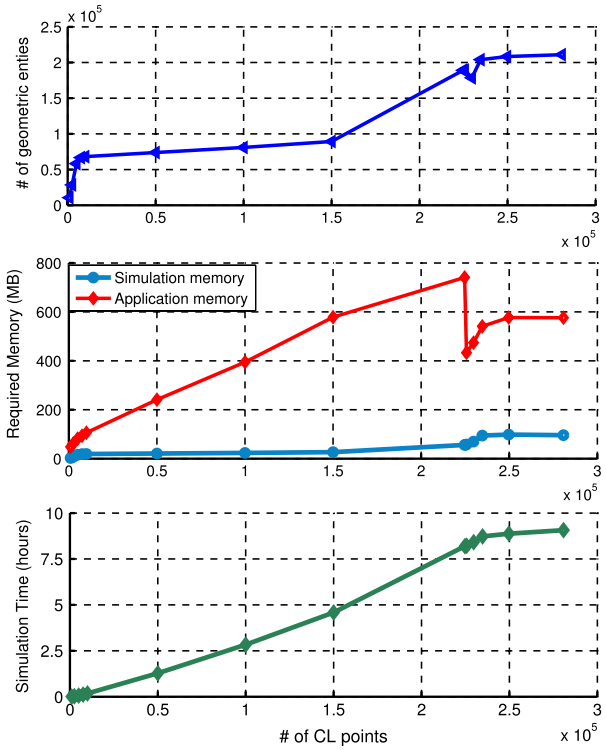

Fig. 22. The simulation results for Dome workpiece when using *B*-rep based milling simulation.

indirect rendering of the completed composite ADF by rendering a triangulation of the surface. In this case the cost of generation the triangle mesh is paid once and then highly interactive rendering of the alternate representation becomes possible.

8. Conclusions

In this paper, we have described a new shape representation, the *composite ADF*, and its application to NC milling simulation. The composite ADF representation provides micron accuracy with low memory requirements and enables simulation speeds up to 1000× faster than actual cutting time. In a composite ADF all surfaces are represented by analytic or procedural Euclidean distance fields that are combined using Boolean operations to implicitly represent the milled surface of the workpiece. The distance field functions are sampled within an octree bounding volume hierarchy that localizes the contribution of each distance field to the composite boundary and dramatically accelerates geometric operations.

The computation of the distance field of the swept volume of a milling tool is handled by an inverted trajectory approach where the problem is solved in the coordinate system of the moving tool. The effect of this transformation is that instead of computing the distance between a point and the complex envelope of the swept tool, we compute minimum distance between a curve and the piece-wise continuous surface of the stationary tool. For many tool geometries and 3-axis tool motions this reduces to a distance query to analytically defined geometries, however a numerical solution has to be applied for 5-axis tool motions. This approach can be generalized to more complex 3-axis and 5-axis tool motions by resorting to fast numerical optimization methods.

The high quality of the simulated workpiece is readily apparent when compared to the actual milled workpiece. The high accuracy provided by this approach to milling simulation will enable the detection and correction of very small milling errors in the production of free-form surface for mold and die applications thereby reducing waste and increasing productivity which is key to reducing both cost and time in the market for new designs. The *composite ADF* provides an exact correspondence between each portion of the composite surface and the milling instruction that generated it. This capability reveals the detailed origin of surface features and enables very rapid revisions to milling programs to be carried out.

References

[1] Blackmore D, Samulyak R, Leu MC. A singularity theory approach to swept volumes. International Journal of Shape Modeling 2000;6:105–29.
[2] Wang WP, Wang KK. Geometric modeling for swept volume of moving solids. IEEE Computer Graphics and Applications 1986;6(12):8–17.
[3] Martin RR, Stephenson PC. Sweeping of three dimensional objects. Computer-Aided Design 1990;22(4):223–33.
[4] Weld John D, Leu Ming C. Geometric representation of swept volumes with application to polyhedral objects. International Journal of Robotics Research 1990;9(5):105–17.
[5] Abdel-Malek Karim, Yeh Harn-Jou, Othman Saeb. Swept volumes: void and boundary identification. Computer-Aided Design 1998;30(13):1009–18.
[6] Malek KA, Yeh HJ. Geometric representation of the swept volume using Jacobian rank-defficiency conditions. Computer-Aided Design 1997;29(6): 457–68.
[7] Blackmore D, Leu MC, Wang L. The sweep-envelope differential equation algorithm and its application to NC machining verification. Computer-Aided Design 1997;29:629–37.
[8] Blackmore D, Leu Ming C. A differential equation approach to swept volumes. Institute of Electrical and Electronics Engineers 1990;120:143–9.
[9] Gaemers S, Elsevier CJ, Bax A, Elber G, Kim M-S. Offsets, sweeps, and minkowski sums. Computer-Aided Design 1999;31(3): 163–163(1).
[10] William Schroeder, William Lorensen, Steve Linthicum. Implicit modeling of swept surfaces and volumes. In: VIS'94: proceedings of the conference on visualization'94. 1994. p. 40–5.
[11] Schmidt R, Wyvill B. Implicit sweep surfaces. Department of Computer Science. University of Calgary. 2005.
[12] Zhang Xinyu, Kim Young J, Manocha Dinesh. Reliable sweeps. In: 2009 SIAM/ACM joint conference on geometric and physical modeling. SPM'09, New York (NY, USA): ACM; 2009. p. 373–8.
[13] Juttler Bert, Wagner Michael G. Computer-aided design with spatial rational *b*-spline motions. ASME Journal of Mechanical Design 1996;118:118–93.
[14] Malek KA, Yang J, Blackmore D, Joy K. Swept volumes: foundations, perspectives, and applications. International Journal of Shape Modeling 2006; 12(1):87–127.
[15] Gupta Satyandra K, Saini Sunil K., Spranklin Brent W, Yao Zhiyang. Geometric algorithms for computing cutter engagement functions in 2.5d milling operations. Comput. Aided Des. 2005;37(14):1469–80.
[16] Imani BM, Sadeghi MH, Elbestawi MA. An improved process simulation system for ball-end milling of sculptured surfaces. International Journal of Machine Tools and Manufacture 1998;38(9):1089–107.
[17] Spence Allan D, Abrari Farid, Elbestawi MA. Integrated solid modeler based solutions for machining. In: SMA'99: proceedings of the fifth ACM symposium on solid modeling and applications. New York (NY, USA): ACM; 1999. p. 296–305.
[18] Spence AD, Altintas Y. A solid modeller based milling process simulation and planning system. Journal of Engineering for Industry 1994;116:61–9.
[19] Ferry W, Yip-Hoi D. Cutter-workpiece engagement calculations by parallel slicing for five-axis flank milling of jet engine impellers. Journal of Manufacturing Science and Engineering 2008;130:051011–2.
[20] Hoffmann Christoph M, Hopcroft John E. Geometric ambiguities in boundary representations. Computer-Aided Design 1987;19(3):141–7.
[21] Sambandan K, Wang KK. Five-axis swept volumes for graphic NC simulation and verification. In: Proceedings of the ASME design automation conference. ASME; 1989. p. 143–50.
[22] Hook Tim Van. Real-time shaded NC milling display. In: SIGGRAPH'86: proceedings of the 13th annual conference on computer graphics and interactive techniques. New York (NY, USA): ACM; 1986. p. 15–20.
[23] Saito Takafumi, Takahashi Tokiichiro. NC machining with *g*-buffer method. In: SIGGRAPH'91: proceedings of the 18th annual conference on computer graphics and interactive techniques. New York (NY, USA): ACM; 1991. p. 207–16.
[24] Huang Yunching, Oliver James H. NC milling error assessment and tool path correction. In: SIGGRAPH'94: proceedings of the 21st annual conference on computer graphics and interactive techniques. New York (NY, USA): ACM; 1994. p. 287–94.
[25] Hui KC. Solid sweeping in image space-application in NC simulation. The Visual Computer 1994;10:306–16.
[26] Muller Heinrich, Surmann Tobias, Stautner Marc, Albersmann Frank, Weinert Klaus. Online sculpting and visualization of multi-dexel volumes. In: SM'03: Proceedings of the eighth ACM symposium on solid modeling and applications. New York (NY, USA): ACM; 2003. p. 258–61.
[27] Kawashima Yasumasa, Itoh Kumiko, Ishida Tomotoshi, Nonaka Shiro, Ejiri Kazuhiko. A flexible quantitative method for NC machining verification using a space-division based solid model. The Visual Computer 1991;7(2–3): 149–57.

[28] Oliver JH, Goodman ED. Direct dimensional NC verification. Computer-Aided Design 1990;22(1):3–10.
[29] Jerard Robert B, Drysdale Robert L, Hauck Kenneth, Schaudt Barry, Magewick John. Methods for detecting errors in numerically controlled machining of sculptured surfaces. IEEE Computer Graphics and Applications 1989;9(1):26–39.
[30] Chappel Ian T. The use of vectors to simulate material removed by numerically controlled milling. Computer-Aided Design 1983;15(3):156–8.
[31] Inui Masatomo, Ohta Atsushi. Using a GPU to accelerate die and mold fabrication. IEEE Computer Graphics and Applications 2007;27:82–8.
[32] Inui Masatomo, Umezu Nobuyuki. GPU acceleration of 5-axis milling simulation in triple dexel representation. In: 2010 international symposium on flexible automation. Proceedings of 2010 ISFA. 2010.
[33] Bloomenthal Jules, Wyvill Brian, editors. Introduction to implicit surfaces. San Francisco (CA, USA): Morgan Kaufmann Publishers Inc.; 1997.
[34] Weinert Klaus, Du Shangjian, Damm Patrick, Stautner Marc. Swept volume generation for the simulation of machining processes. International Journal of Machine Tools and Manufacture 2004;44(6):617–28.
[35] Mann Stephen, Bedi Sanjeev. Generalization of the imprint method to general surfaces of revolution for NC machining. Computer-Aided Design 2002;34(5):373–8.
[36] Hu Zeng-Jia, Ling Zhi-Kui. Swept volumes generated by the natural quadric surfaces. Computers & Graphics 1996;20(2):263–74.
[37] Rossignac J, Kim JJ, Song SC, Suh KC, Joung CB. Boundary of the volume swept by a free-form solid in screw motion. Computer-Aided Design 2007;39(9):745–55.
[38] Erdim Hüseyin, Ilieş Horea T. Classifying points for sweeping solids. Computer-Aided Design 2008;40(9):987–98.
[39] Breen DavidE, Mauch Sean, Whitaker RossT. 3D scan conversion of CSG models into distance volumes. In: Proc. 1998 IEEE symposium on volume visualization. 1998. p. 7–14.
[40] Frisken Sarah F, Perry Ronald N, Rockwood Alyn P, Jones Thouis R. Adaptively sampled distance fields: a general representation of shape for computer graphics. In: SIGGRAPH'00: proceedings of the 27th annual conference on computer graphics and interactive techniques. New York (NY, USA): ACM Press, Addison-Wesley Publishing Co.; 2000. p. 249–54
[41] Frisken Sarah F, Perry Ronald N. Designing with distance fields. In: SIGGRAPH'06: ACM SIGGRAPH 2006 courses. New York, NY, USA: ACM; 2006. p. 60–6.
[42] Hoffmann Christoph M. Robustness in geometric computations. Journal of Computing and Information Science in Engineering 2001;1(2):143–55.
[43] Perry RonaldN, Frisken SarahF. Kizamu: a system for sculpting digital characters. In: SIGGRAPH'01: proceedings of the 28th annual conference on computer graphics and interactive techniques. 2001. p. 47–56.
[44] Frisken SF, Perry RN. Simple and efficient traversal methods for quadtrees and octrees. Journal of Graphics Tools 2002;7(3):1–12.
[45] Lorensen WilliamE, Cline HarveyE. Marching cubes: a high resolution 3D surface construction algorithm. In: Computer graphics. Proceedings of SIGGRAPH 87. vol. 21. July 1987. p. 163–9.
[46] Gibson Sarah F. Constrained elastic surface nets: generating smooth surfaces from binary segmented data. In: Proceedings of the first international conference on medical image computing and computer-assisted intervention. MICCAI'98, London (UK): Springer-Verlag; 1998. p. 888–98.
[47] Press William H, Teukolsky Saul A, Vetterling William T, Flannery Brian P. Numerical recipes in C (2nd ed.): the art of scientific computing. New York (NY, USA): Cambridge University Press; 1992.
[48] Hart JC. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer 1996;12(10):527–45.
[49] Hirai Tetu, Yamamoto Tsuyoshi. Hybrid volume ray tracing of multiple isosurfaces with arbitrary opacity values. IEICE Transactions on Information and Systems 1996;E79-D(7):965–72.
[50] Mora Benjamin, Jessel Jean-Pierre, Caubet Rene. A new object-order ray-casting algorithm. In: IEEE visualization 2002. 2002. p. 203–10.
[51] Yau Hong-Tzong, Tsou Lee-Sen. Efficient NC simulation for multi-axis solid machining with a universal APT cutter. Journal of Computing and Information Science in Engineering 2009;9(2):021001–0210010.