7-1994

NC Milling Error Assessment and Tool Path Correction

Yunching Huang
Iowa State University

James H. Oliver
Iowa State University, oliver@iastate.edu

NC Milling Error Assessment and Tool Path Correction

Abstract
A system of algorithms is presented for material removal simulation, dimensional error assessment and automated correction of Þve-axis numerically controlled (NC) milling tool paths. The methods are based on a spatial partitioning technique which incorporates incremental proximity calculations between milled and design surfaces. Hence, in addition to real-time animated Þve-axis milling simulation, milling errors are measured and displayed simultaneously. Using intermediate error assessment results, a reduction of intersection volume algorithm is developed to eliminate gouges on the workpiece via tool path correction. Finally, the view dependency typical of previous spatial partitioning-based NC simulation methods is overcome by a contour display technique which generates parallel planar contours to represent the workpiece, thus enabling dynamic viewing transformations without reconstruction of the entire data structure.

Keywords
Iowa Center for Emerging Manufacturing Technology, Virtual Reality Applications Center

Disciplines
Computer-Aided Engineering and Design | Graphics and Human Computer Interfaces

Comments
This is a manuscript of a conference proceeding from *Proceedings SIGGRAPH '94* (1994): 287, doi:10.1145/192161.192231. Posted with permission.

NC Milling Error Assessment and Tool Path Correction

Yunching Huang [†] and James H. Oliver [††]

Iowa Center for Emerging Manufacturing Technology
Department of Mechanical Engineering
Iowa State University
Ames, IA

ABSTRACT

A system of algorithms is presented for material removal simulation, dimensional error assessment and automated correction of five-axis numerically controlled (NC) milling tool paths. The methods are based on a spatial partitioning technique which incorporates incremental proximity calculations between milled and design surfaces. Hence, in addition to real-time animated five-axis milling simulation, milling errors are measured and displayed simultaneously. Using intermediate error assessment results, a reduction of intersection volume algorithm is developed to eliminate gouges on the workpiece via tool path correction. Finally, the view dependency typical of previous spatial partitioning-based NC simulation methods is overcome by a contour display technique which generates parallel planar contours to represent the workpiece, thus enabling dynamic viewing transformations without reconstruction of the entire data structure.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; J.6 [Computer-Aided Engineering]: Computer-Aided Manufacturing

1 INTRODUCTION

Numerically controlled (NC) milling technology is a production process that directs a cutter through a set of prescribed sequential trajectories to fabricate a desired part from raw stock. The technology is capable of producing free-form sculptured surfaces while maintaining tight milling error tolerance. NC milling technology is, therefore, widely used in the production of complicated, high precision, low quantity products such as molds, dies, aerospace parts, etc. These products, especially molds and dies, typically affect many other subsequent production processes. Thus the influence of NC milling on product development and quality control is significant.

In order to improve the accuracy and reliability of the NC milling process, verification methods are used prior to actual production, to check milling tool paths for potential problems such as milling error, collision, improper machining parameters, tool wear, etc. These problems typically produce undesirable consequences such as unqualified products, machine damage, and personnel injuries. Hence, NC milling verification is a critical procedure for actual production. Traditionally, milling verification is conducted by observing line drawings of tool paths and performing test milling on soft, inexpensive materials. Since these methods are time-consuming, expensive, and prone to error. They are gradually being replaced by analytical methods.

1.1 Analytical NC Milling Simulation and Verification Methods

Analytical methods of NC milling simulation and verification are generally distinct from techniques used to model milling phenomenon and formulate milling problems. These methods can be categorized into three approaches — direct solid modeling, discrete vector intersection, and spatial partitioning representation. Each of these approaches has been applied to five-axis NC verification with varying ranges of applicability and degrees of success. The following discussion summarizes the research underlying each approach.

1.1.1 Direct solid modeling approach

The direct solid modeling approach is typically implemented by using constructive solid geometry or boundary representation solid modeling systems [21]. Since regularized Boolean set operations are supported in these modeling systems, a milling simulation is implemented via a series of regularized Boolean difference operations to subtract successive tool swept volumes from the workpiece. The result is an explicit solid model of the milled workpiece.

The direct solid modeling approach is theoretically capable of providing accurate NC milling simulation and verification. However, its application remains limited by the complexity of five-axis

[†] 6 Chung-Hsing Street, Hsin-Chuang, Taipei, Taiwan 24209, R.O.C.
e-mail: yhuang@iastate.edu, phone: 011-886-2-991-5796

[††] 2078 Black Engineering Building, Ames, Iowa 50011-2160
e-mail: oliver@iastate.edu, phone: (515) 294-1745

SIGGRAPH '94, July 24-29, Orlando, Florida
 ISBN: 0-89791-667-0

tool swept volume formulation and the Boolean set operations. Although solid primitives combined by means of regularized Boolean set operators can be displayed via ray casting techniques without explicit evaluation [1,22], ray-intersecting five-axis swept volumes is still a time-consuming process [16].

1.1.2 Discrete vector intersection approach

The discrete vector intersection approach assesses milling error by computing distances between a set of pre-selected points on design surfaces and tool swept surfaces [4,10,17]. Each design surface point has an associated vector, typically the outward normal, as shown in Figure 1. Verification is performed by calculating the intersection between the point-vector pairs (rays) and the tool swept surfaces.

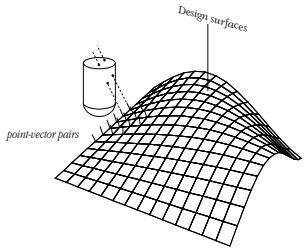

Figure 1 Illustration of discrete vector intersection approach

The discrete vector intersection approach is best described in terms of three sub-tasks: discretization, localization, and intersection [17]. The discretization task transforms the design surfaces into a sufficiently dense distribution of surface rays to approximate the original design surfaces. The localization process extracts a plausible subset of rays for each tool motion. Finally, the intersection calculation determines the directed distance (cut value) between each surface point in the subset and the tool swept surfaces, i.e., an indication of the milling error at the surface point. Graphical display of computed milling errors is typically presented by a color map depicting a range of cut values superimposed on the design surfaces. The severity of milling errors on the design surfaces can be visually depicted.

Since this technique addresses only the design surface, the discrete vector intersection approach is not capable of performing milling simulation or computing material removal rate. Also, computation of intersection between rays and the five-axis tool swept volumes is expensive.

1.1.3 Spatial partitioning representation approach

The primary disadvantage of the direct solid modeling method for NC milling verification, i.e., the computational complexity of regularized Boolean set operations, motivates the use of the spatial partitioning approach. This method decomposes a solid object into a collection of basic elemental components thus simplify the processes of regularized Boolean set operations to one-dimensional computations. Several proposed implementations differ in the type of basic element used to approximate the solid, including the dexel data structure [20], the G-buffer data structure [19], the octree data structure [3,11], and the ray-representation [13].

These spatial partitioning representation implementations share the advantage of efficient regularized Boolean set operations which facilities realistic milling simulation. Some even provide specialized hardware implementations to increase performance [13,20]. These advantages are very useful in visual detection of gross milling errors and in milling process animation. Furthermore, the volume removed at each tool motion can be easily calculated with user-specified accuracy depending on the size of the basic element. Despite these advantages, however, the spatial partitioning approach has failed to address dimensional milling verification capabilities of comparable complexity and accuracy as those provided by the discrete vector intersection approach.

1.2 Motivation

To maintain the advantages and overcome limitations of previous milling simulation and error assessment approaches, this research adopts the spatial partitioning approach as the basis for a comprehensive system which also incorporates the advantages of the discrete vector intersection approach. The goal is to develop a platform independent NC milling verification system that is capable of providing realistic simulation, dimensional error assessment and tool path correction. Finally, a contour display method is introduced to ameliorate the view dependency problem typically associated with application of the spatial partitioning representation. Thus dynamic viewing transformations of the milled and verified part is achieved.

2 DEXEL REPRESENTATION OF SOLID GEOMETRY

A dexel representation derived from the dexel data structure of Van Hook [20] is introduced to approximate free-form solid geometry as sets of rectangular solid elements. The dexel representation of a solid is constructed in a dexel coordinate system via ray intersection and is manipulated using dexel-based Boolean set operations. A major distinction between this approach and the original dexel data structure described by Van Hook is that the construction of dexels is not limited by the viewing vector. An independent dexel coordinate system is used to support dynamic viewing transformations. Furthermore, five-axis tool motions, dimensional error assessment, and tool path correction are implemented based on the dexel representation.

2.1 Dexel Coordinate System

The left-handed dexel coordinate system is defined by an origin point $\boldsymbol{O}$, a depth vector $\boldsymbol{v}_d$, and an orientation vector $\boldsymbol{v}_o$ in the world coordinate system. Basis vectors of the dexel coordinate system are given by,

$$\begin{bmatrix} \boldsymbol{v}_x \\ \boldsymbol{v}_y \\ \boldsymbol{v}_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_d \times \boldsymbol{v}_o \\ -\boldsymbol{v}_x \times \boldsymbol{v}_d \\ \boldsymbol{v}_d \end{bmatrix} \quad (1)$$

The vectors $\boldsymbol{v}_d$ and $\boldsymbol{v}_o$ are analogous to the vectors typically required to define a viewing transformation in computer graphics applications, i.e., the viewing direction and the view-up vector [7],

respectively. In Van Hook's dexel data structure, the depth vector $\boldsymbol{v}_d$ is limited to the viewing direction, thus the view is fixed once dexel data structure has been constructed. In the current implementation, the properties of each dexel are stored relative to an independent dexel coordinate system and can be transformed into either the world coordinate system or the screen coordinate system via coordinate transformations.

Dexel locations are referenced by a two-dimensional grid in the *xy*-plane of the dexel coordinate system, called the *dexel plane*. Each grid point is addressed by an integer pair, e.g., (I_x, I_y) as illustrated in Figure 2. Assuming the grid points are uniformly spaced along the *x*- and *y*-axis by distances w_x and w_y, respectively, the dexel coordinate values of each grid point are computed by ($I_x w_x$, $I_y w_y$). To simplify dexel operations and display tasks, in this implementation, w_x and w_y are assumed equal, and set to a value w.

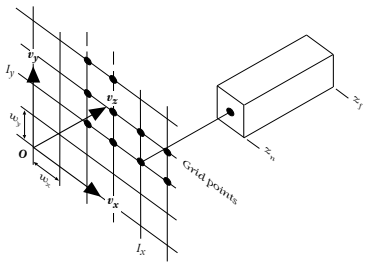

Figure 2 Dexel coordinate system

Each dexel is physically a rectangular solid located on a dexel plane grid point extending along the *z*-axis of the dexel coordinate system. The *x*- and *y*-dimension of each dexel is fixed and the length of a dexel is determined by a *z*-depth pair (z_n, z_f), where the subscripts *n* and *f* denote near and far values, respectively.

The accuracy of a dexel representation of any object is determined by the dexel plane orientation and the dexel size w_0. Two methods are provided for specification of the dexel coordinate system. The first is via interactive selection, i.e., the user chooses a coordinate orientation in which essential areas of the design surfaces are exposed. The second approach automates this procedure by calculating the dexel coordinate system orientation that produces the maximum projected area of the design surfaces on the dexel plane [9].

The size of each dexel face is calculated from a user-specified approximation tolerance, *E*. In this application, the size of the milling tool determines the minimum feature size of the resulting part. Therefore, as shown in Figure 3, the dexel size is computed relative to tool radius, *R*,

$$w = 2R - 2\sqrt{R^2 - E^2} \qquad (2)$$

Finally, the ratio of *E* to *R* characterizes the relative accuracy of a tool-based dexel representation. In this implementation, an *E*/*R* value of 0.2 provides real-time performance at reasonable accuracy. This corresponds to a *w*/*R* value of 0.04.

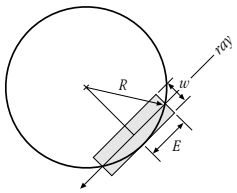

Figure 3 Maximum dexel approximation error of a circle

2.2 Ray Intersection of Dexel Models

A ray intersection process is used to convert solid objects into the dexel representation. Parallel rays are fired from a sub-set of grid points on the dexel plane to intersect solid objects, and dexels are formed from the segments of rays that are in the interior of the objects. Thus, a set of dexels is generated to represent the object.

The ray intersection procedure also generates the object's outward surface normal corresponding to each dexel face (grid point). The surface normals are used to determine the intensity of each dexel face by computing the scalar product of the normal vector and a light source vector. The light intensity is recorded in the data structure, hence each dexel face may be realistically shaded. In addition to the *z*-depth pair and light intensity, several auxiliary parameters are recorded, including dexel type, (e.g., cutter, workpiece, fixture, etc.) and a pointer to the next dexel, if any, at the grid address.

2.3 Regularized Boolean Set Operations on Dexel Models

Regularized Boolean set operations are simple to implement for dexel-based solid objects. Since dexel faces are uniform, the operations are reduced to one-dimensional depth comparisons. The union operation either merges two intersecting dexels if the depths overlap, or, otherwise, constructs a link between the two component dexels. The intersection operation generates a dexel from the overlapping range, or null if there is no intersection. The difference operation removes the intersecting portion. An example of the dexel-based Boolean set operations is shown in Figure 4. Such operations are used intensively for milling simulation.

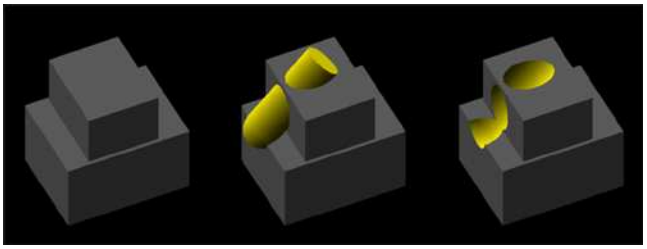

Figure 4 Shaded image illustration of Boolean operations

2.4 Display of Dexel Models

The image-space display method proposed by Van Hook [20] is very efficient for visualizing dexel-based objects. This method aligns the depth vector of the dexel coordinate system with the viewing vector of the screen coordinate system, so only the near face of each dexel is visible and each grid point on the dexel plane corresponds to a constant number of pixels on the display device. Figure 5 demonstrates a one-to-one mapping between the dexel plane and the viewing screen, the color index of the nearest dexel at each address is written directly to the frame buffer of the display device. However, since dexels are aligned with the viewing vector,

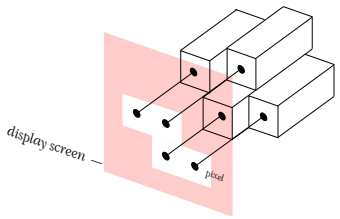

Figure 5 Image-space display method

only the front and the back views can be efficiently displayed. To display dexel-based objects in other viewing directions the entire dexel representation must be reconstructed which severely limits applications in manufacturing and engineering.

To overcome this limitation, a *contour display* method is developed to provide an efficient dynamic viewing capability. This method generates contours that connect dexel faces (center points) along constant x and y grid addresses. Thus two sets of equally spaced planar contours are displayed to represent dexel-based objects. Alternatively, the contour points could be used to construct a triangular mesh for a smoother rendering of dexel-based objects [6,14].

Assuming a constant x-contour is to be generated, contour generation proceeds by first selecting a starting dexel that has the smallest y coordinate value, then it sequentially traverses through all dexels with the same x address. The basic rule of the traversal is, from the current dexel point, step to the next higher grid point in the y-direction if the current dexel point is on the near side, otherwise, step to the next lower one. Several cases of dexel point connection, as illustrated in Figure 6, are classified in the contour generation process. In case 1, dexels A and B overlap and thus the next dexel point to be connected from A_n is B_n (the near dexel point at the next higher y). Case 4 is similar to case 1 except it handles the dexel points at the far side. In case 2, there is no dexel at the next higher grid point, thus the far dexel point is connected. Similarly, case 5 handles far dexel points. In case 3, dexel C and D are connected by dexel B, and there is no dexel above C, so the next dexel point to be connected from the far side of C is the near side of D. Case 6 addresses dexel near-to-far connection in a similar fashion.

As illustrated in Figure 6, only six cases can occur during contour generation, however, internal voids that do not connect to any outer contour are possible. Thus after each contour is generated, dexels are scanned to identify dexel points that are not included in the contours. Such dexel points are used as starting points for the contour generation process to create additional internal contours.

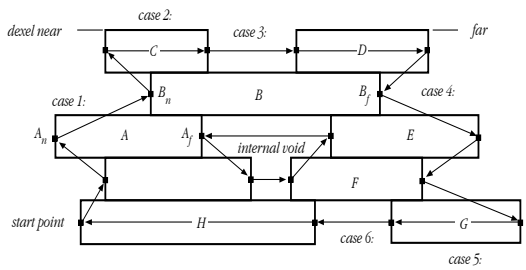

Figure 6 Cases of dexel point connection

A typical dexel representation of solid geometry can easily generate thousands of dexel points on the contours. However, not all dexel points are necessary for the contours, e.g., sequential dexel points that have the same color and are nearly colinear can be reduced to two end points. Thus intermediate dexel points are eliminated by a culling process that checks sequential dexel points, of the same color, against a linearity tolerance. Since the dexel model is displayed as a set of planar contours, dynamic graphical viewing of the model is achievable with commonly available graphics hardware. Figure 7 demonstrates an example of the contour display method in which 71% of the original 195,767 dexel points are eliminated, to facilitate efficient dynamic viewing.

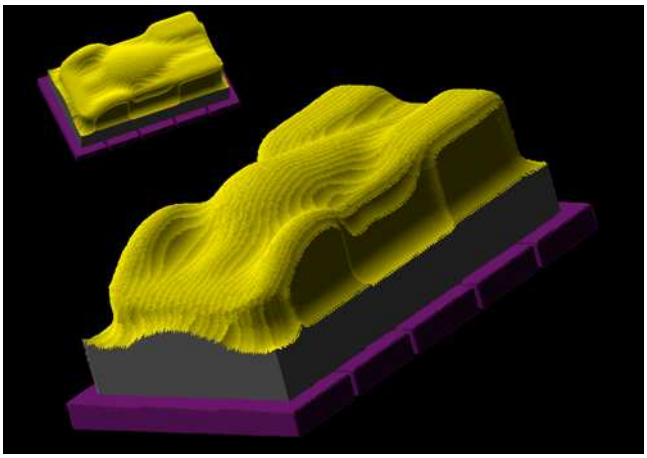

Figure 7 Contour display of dexel representation

3 NC MILLING SIMULATION AND DIMENSIONAL ERROR ASSESSMENT

The NC milling simulation algorithm utilizes the dexel representation of solid geometry to model cutters, workpiece stock and fixtures in a milling setup. It applies regularized Boolean difference operations to simulate the material removal process between a moving tool and workpiece during the milling process. A moving tool is represented by an *instances of motion* approach which successively updates the workpiece model. Thus the computational expense of ray intersecting a swept volume is eliminated without sacrificing accuracy, and realistic, real-time milling simulation is achieved.

3.1 NC Milling Simulation

The instances of motion approach approximates tool swept volumes to dexel resolution by a finite set of tool instances. Let the start and end cutter location (CL) points of a tool motion be denoted by $\boldsymbol{P}$ and $\boldsymbol{Q}$, respectively, and the corresponding unit tool axes by $\boldsymbol{u}$ and $\boldsymbol{v}$. Transforming the CL points and axes into the dexel coordinate system yields $\boldsymbol{P}$', $\boldsymbol{Q}$', $\boldsymbol{u}$', and $\boldsymbol{v}$', respectively, as shown in Figure 8. To model a tool swept volume using instances of motion, the maximum distance between adjacent instances must be less than the dexel size w, so the number of instances n between the CL points is given by,

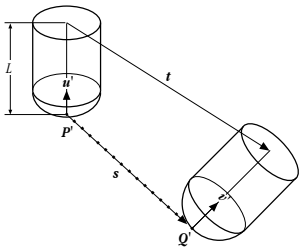

Figure 8 Computing instances of tool motion

$$n = \frac{max\left(|\boldsymbol{s}_x|, |\boldsymbol{s}_y|, |\boldsymbol{s}_z|, |\boldsymbol{t}_x|, |\boldsymbol{t}_y|, |\boldsymbol{t}_z|\right)}{w} \tag{3}$$

where $\boldsymbol{s}$ and $\boldsymbol{t}$ are linear sweeping vectors of the tool top and bottom center points in the dexel coordinate system, i.e.,

$$\begin{aligned} \boldsymbol{s} &= \boldsymbol{Q}' - \boldsymbol{P}' \\ \boldsymbol{t} &= \boldsymbol{s} + L\,(\boldsymbol{v}' - \boldsymbol{u}') \end{aligned} \tag{4}$$

where L is the length of the tool.

The dexel-space tool location and non-normalized tool axis at each instance of motion, denoted by $\boldsymbol{I}^d$, is given by linear interpolation,

$$\boldsymbol{I}_i^d = \left[\, \boldsymbol{P}' + \frac{i}{n}\boldsymbol{s}\ ,\boldsymbol{u}' + \frac{i}{n}(\boldsymbol{v}' - \boldsymbol{u}') \,\right] \tag{5}$$

where $i = 1, \ldots, n$. Note that for three-axis motion, the tool axis is fixed so the tool axis interpolation portion of Equation (5) is omitted and the computation of n is simplified. During the simulation, a dexel representation of the tool model is generated at every instance to sequentially update the workpiece by Boolean difference operations thus simulating the material removal process.

An example of three-axis milling simulation representing a typical rough milling process is demonstrated in Figure 9. The computation time for this example is 47 seconds for 3324 instances, or 68.6 instances per second, on a Silicon Graphics (SGI) Indy with a 150MHz MIPS R44000 CPU. Note that the initial shape of the workpiece is not limited to blocks, more complicated parts can be constructed via ray intersection with quadric or sculptured surfaces.

The accuracy of the dexel-based instances of motion approach can be evaluated by comparing it with the results of an equivalent approach based on ray intersection with the actual tool swept volume. Assume that point $\boldsymbol{P}$ is generated from the intersection of a tool swept volume and a ray originating at a dexel grid point [16]. In this intersection process, the member (instance) of the family of tool positions on which $\boldsymbol{P}$ lies is computed. Note that the density of intersection points on the tool swept volume is limited by the density of grid points on the dexel plane. Since the instance of motion approach creates at least as many tool instances as the maximum number of grid points along the tool trajectory, as indicated in Equation (3), the resulting intersections are equivalent to those computed via the swept volume approach.

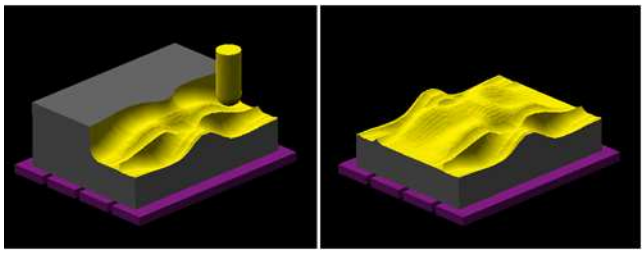

Figure 9 Three-axis milling simulation

3.2 Milling Error Assessment

An important property of the dexel representation is that dexel points lie on the surface of the represented object. This property results from the ray intersection process to convert objects into a dexel representation. Furthermore, since the dexel coordinate system is fixed for all dexel-based objects, this property also holds for dexel-based objects generated from regularized Boolean set operations. The milling error assessment algorithm exploits this property and is capable of computing the discrepancy between the dexel-based milled surfaces and actual design surfaces with high accuracy.

The algorithm is essentially an inverse formulation of the discrete vector intersection approach, i.e., instead of calculating milling errors from the design surfaces to the tool swept volume, this method computes the errors from the milled part to the design surfaces. Thus the localization and intersection sub-tasks of the former approach are replaced by a surface near point calculation between dexel points and design surfaces. The surface near point calculation is performed only on dexels that are updated during simulation, thus no additional localization effort is needed. Figure 10 illustrates an instance of the verification algorithm, in which a dexel is updated by a regularized Boolean difference operation and the new dexel point $\boldsymbol{C}$ is assessed for milling error by the surface near point calculation algorithm.

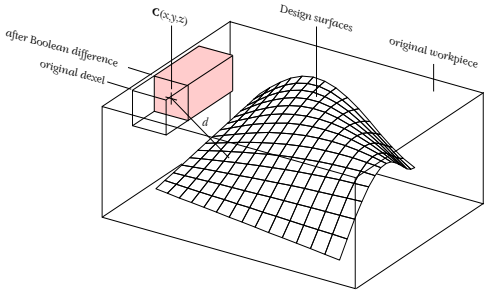

Figure 10 Milling error assessment algorithm

3.2.1 Surface near point calculation

The surface near point calculation algorithm computes the distance d, as shown in Figure 10, between each updated dexel point and the design surfaces. The sign of d is determined by the direction of corresponding surface normal vector. Since the design surfaces are not discretized for error assessment, any of a number of near point algorithms for sculptured surfaces can be employed [2,15,18]. These algorithms are based on the fact that the minimum distance between a space point $\boldsymbol{P}$ and a surface $\boldsymbol{S}$ occurs at a surface point $\boldsymbol{Q}$ at which the vector ($\boldsymbol{P}$-$\boldsymbol{Q}$) is perpendicular to the sur-

face tangent plane. This property forms a system of non-linear equations and is solved by using a Newton/Raphson search procedure [12].

Since the robustness and efficiency of the Newton/Raphson method depend on the choice of an initial point, a voxel data structure is implemented as a preprocessing step to supply candidate initial surface points for any given dexel point. The voxel data structure also provides for localization of candidate surfaces for each near point computation, i.e., only those surfaces nearest the given dexel point are stored in the voxel data structure. Hence the minimum distance is computed only among these surfaces.

The initial surface near point lookup algorithm first discretizes each design surface into a set of N points based on a given chordal deviation tolerance [5,8]. These surface points form a triangular mesh that approximates the original surfaces. Each triangle of the mesh is then inserted in a voxel data structure [4,7] constructed in a bounding box of all design surfaces in the world coordinate system. Voxels that intersect with the surface triangular mesh are populated with the surface index and the (u, v) parameters of corresponding vertices. Since more than one vertex may be contained in a voxel, each voxel records the root index of a point list. Thus all surface points in a given voxel may be accessed via a linked list.

Two additional parameters complete the definition of the initial surface near point data structure: the dimension of a voxel and a range of projection. The voxel size affects the efficiency of surface near point computation, i.e., the larger the voxel, the more candidate initial surface points it contains. However, the accuracy of surface near point computation is typically not affected by the voxel size. The following heuristic relationship is generally effective for specification of voxel size d_v,

$$d_v = \frac{B_{max}}{\sqrt{N}} \tag{6}$$

where B_{max} is the maximum dimension of the design surface bounding box.

The range of projection parameter sets upper and lower offset bounds from the triangular mesh, so that only space points within this range are considered for surface near point calculation. To build the range of projection into the voxel data structure, additional voxels on both sides of each triangle are filled with the corresponding surface information if they lie within the range of projection (measured with respect to the triangle normal). Note that the range of projection must be larger than the chordal deviation tolerance to cover hills and valleys missed in surface discretization. An example of the voxel data structure is illustrated in Figure 11 which shows the original design surface and all filled voxels. Only voxels that are close to the surface are filled with surface near point data.

To lookup initial surface near points a space point is converted into integer voxel indices (I_x, I_y, I_z) to obtain the surface near point list of the corresponding voxel address. Surface near point calculations are then performed from each of the initial points in the voxel list, and the minimum of all candidate solutions is taken as the milling error.

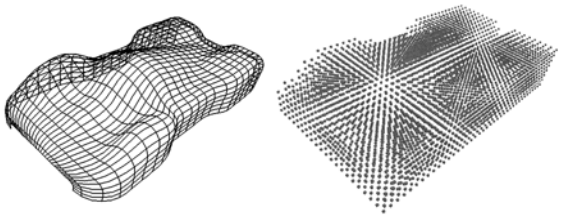

Figure 11 A design surface and associated voxel data

3.2.2 Graphical representation of milling errors

The results of milling error assessment are displayed by several hues depicting the depth of cut. A lookup table is prepared to interpret a cut value into a proper hue value for dexel display. The intensity of each hue is obtained from the dexel data structure, (based on normal and light source vectors described above). Figure 12 demonstrates an example of the graphical representation of milling error during a milling simulation process. In this figure, milling error information is encoded and displayed on the milled surface and the color bar shown at the left-hand side depicts the range of milling error. The computation time for this example running on the same SGI Indy is 606 seconds for 4624 instances of motion, or 7.6 instances per second.

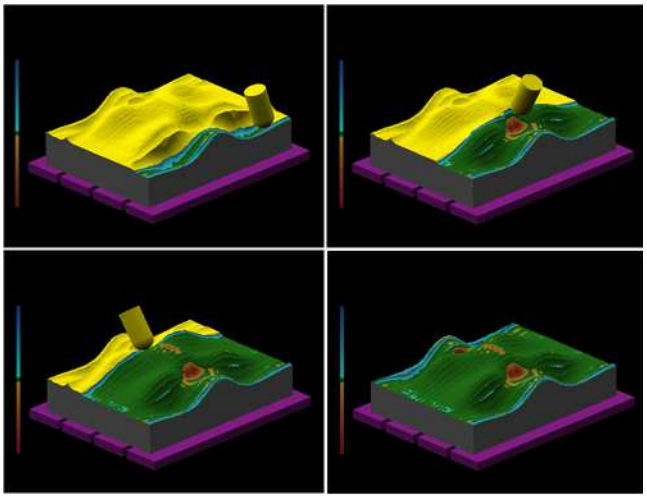

Figure 12 NC milling simulation with error assessment

Assuming a tolerance is given by $[t_l, t_b]$, a gouge is a cut deeper than the lower tolerance bound, and, an undercut is a cut above the higher tolerance bound. In Figure 12, the green color represents errors that are within the tolerance range (-0.01 to 0.01, in this example) relative to the nominal design surfaces; the upper blue colors represent the amount of undercut; and the warmer red colors represent gouge. The tool color (yellow) on the part indicates that the dexel points are outside of a range of interest [17]. The range of interest specifies the maximum and minimum magnitudes of distinguishable undercut and gouge (-0.04 to 0.04 in this case). Hence given a depth of cut, the corresponding color depicting its error is obtained from the color map. For cut values deeper than the lower bound of the range of interest, the color depicting the deepest gouge is used. For undercut larger than the highest bound, the milling tool color is displayed.

Note that the computation time for surface near point calculation is proportional to the number of updated dexels. Hence, a post-process error assessment that computes dimensional milling errors after milling simulation is completed is generally more efficient. The post-process milling error assessment scans through all dexels of the workpiece model and verifies dexel points that have been updated in the milling process. Hence it eliminates unnecessary surface near point computation during the simulation process. The reduction of computational cost can be significant. For example, a result identical to Figure 12 can be obtained in 160 seconds: 119 seconds for five-axis simulation (38.8 instances per second) and 41 seconds for the post-process verification task.

4 TOOL PATH CORRECTION

The NC milling error assessment system identifies potential problems in tool paths so that NC programmers can modify the paths to avoid errors. However, these problems require either manually changing the CL data of the tool paths or changing the design surface model and generating new tool paths. Such processes are generally time-consuming, inaccurate, and may introduce more problems. Therefore, to reduce the difficulty of tool path modification, a *reduction of intersecting volume* algorithm is developed to eliminate gouges.

4.1 Gouge Elimination

The objective of the reduction of intersection volume algorithm is to reduce the depth of cut to meet the lower limit of a specified tolerance range via tool path modification. The algorithm computes the intersection volume of the tool and workpiece using the regularized Boolean intersection operation, then checks the intersection volume for gouge using the milling error assessment algorithm. Detected gouges are removed by translating the tool position along a *guide vector*. The gouge elimination algorithm is employed recursively to ensure the new CL data is gouge-free.

Assuming m gouged dexel points are detected at an instance of tool motion, the guide vector $\boldsymbol{G}$, as illustrated in Figure 13, is computed by,

$$\boldsymbol{G} = \sum_{i=1}^{m} (d_i - t_l)\,\boldsymbol{n}_i \qquad (7)$$

where d_i is the depth of cut at a gouged dexel point, $\boldsymbol{n}_i$ is the outward normal vector generated from the surface near point algorithm at each dexel point, and t_l is the lower bound of the tolerance. If the length of $\boldsymbol{G}$ in Equation (7) is zero or within a range $[-\varepsilon,\varepsilon]$, where ε is a small value, then the tool axis is used as the guide vector. Finally, the magnitude of the guide vector is set to the value of the maximum gouge. The tool is translated by G and the process is repeated iteratively until the gouge is eliminated.

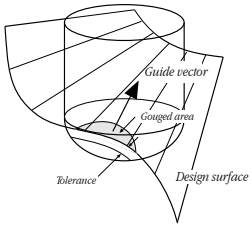

Figure 13 Gouge elimination using the guide vector

An illustration of the gouge elimination algorithm is shown in Figure 14. In this figure, a tool motion is defined by a pair of CL points ***P*** and ***Q***, and it is assumed that the start point ***P*** is gouge-free. The algorithm first iteratively eliminates the gouge at ***Q***, and hence generates a new CL point ***Q***'. The entire tool motion between ***P*** and ***Q***' is then evaluated and the first gouge point is discovered at ***S***. Let ***S***' be the CL point corresponding to ***S*** that avoids the gouge and denote the previous instance of motion as ***R*** (***R*** is gouge-free). The remaining tool motion is broken into two motions and the gouge elimination algorithm is applied recursively to each segment. Thus segments ***R*** through ***S***' and ***S***' through ***Q***' are recursively checked for gouges and subdivided, until the entire motion is gouge free.

An application of this algorithm is demonstrated in Figure 15. A Comparison with Figure 12, shows that the gouges are completely eliminated and a modified tool path has been generated to replace the original. The computation time for this example is 799 seconds for 4716 instances, or 5.9 instances per second. During the gouge elimination process, 138 gouge CL points are detected and 426 iterations are taken to correct them, or 3.1 iterations per gouge elimination. Note that the total instances of motion is increased (originally 4624 instances the original tool path) in the corrected tool path.

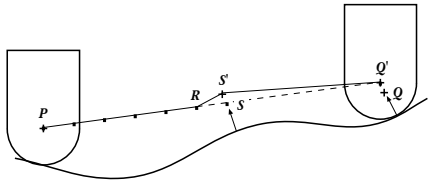

Figure 14 A 2D example of the gouge elimination algorithm

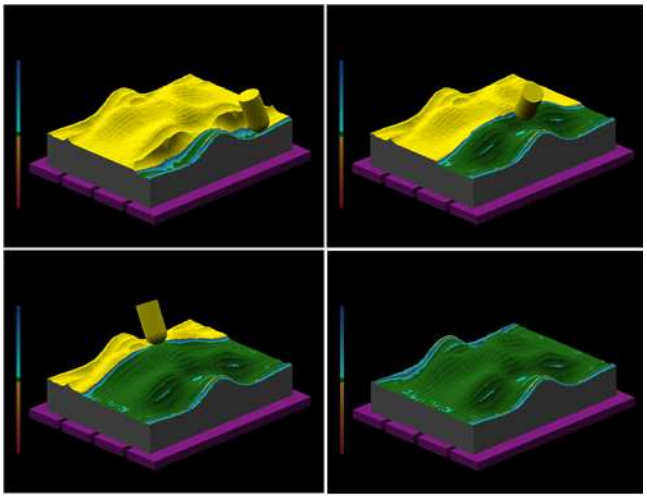

Figure 15 Gouge elimination during milling simulation

5 CONCLUSION AND FUTURE WORK

An integrated system is presented for material removal simulation, dimensional error assessment and automated correction of gen-

eral five-axis NC milling tool paths. The error assessment algorithm combines the advantages of the discrete vector intersection approach and the spatial partitioning representation method, and thus is capable of efficiently simulating and precisely verifying milling tool paths. The dexel-based geometry is constructed in a coordinate system that is independent of screen space, and a contour generation algorithm supports dynamic viewing transformations. Based on the milling error assessment results, a gouge elimination algorithm is developed.

One opportunity for future research is to incorporate an incremental triangulation algorithm [6,14] to generate a polygonal mesh over the emerging workpiece. This may facilitate a smoother rendering for dynamic viewing than the current contour display implementation. In addition, research is underway toward incorporation of tool feed rate with the volume and geometry of material removed from the workpiece to assess machine dynamic performance.

ACKNOWLEDGEMENT

Research support provided by the Office of Naval Research (Grant No. N00014-92-J-4092) is gratefully acknowledged. The authors are also grateful to the Iowa Center for Emerging Manufacturing Technology and the Iowa State University Research Foundation[1]. Special thanks are due to Professors James Bernard, Martin Vanderploeg, and Lin-Lin Chen for their constructive comments and advice. The authors also thank Patrick Bergan, James Schlosser, and Kevin Renze for their support during paper preparation.

REFERENCES

1. Atherton, P. R., "A Scan-Line Hidden Surface Removal Procedure for Constructive Solid Geometry," Proceedings of SIGGRAPH '83. In *Computer Graphics,* 17, 3 (July 1983), pp. 73-82
2. Barnhill, R. E. and Kersey, S. N., "A Marching Method for Parametric Surface/Surface Intersection," *Computer Aided Geometric Design*, 7, 1990, pp. 257-280
3. Brunet P. and Navazo, I, "Solid Representation and Operation Using Extended Octrees," *ACM Transactions on Graphics*, 9, 2, April 1990, pp. 170-197
4. Chang, K. Y. and Goodman, E. D., "A Method for NC Tool Path Interference Detection for A Multi-Axis Milling System," *ASME Control of Manufacturing Process*, DSC-Vol. 28/PED-Vol. 52, 1991, pp. 23-30
5. Drysdale, R. L. and Jerard R. B., "Discrete Simulation of NC Machining," *Algorithmica, Special Issue on Computational Geometry* 4, 1, 1989, pp. 33-60
6. Ekoule, A. B., Peyrin, F. C., and Odet, C. L., "A Triangulation Algorithm From Arbitrary Shaped Multiple Planar Contours," *ACM Transactions on Graphics*, 10, 2, April 1991, pp. 182-199
7. Foley, J. D., Van Dam, A., Feiner, S. K. and Hughes, J. F., *Computer Graphics Principles and Practice*, 1990, Addison-Wesley Publishing Company, New York, NY
8. Huang, Y. and Oliver, J. H., "Non-Constant Parameter NC Tool Path Generation on Sculptured Surfaces," *Proceedings of ASME International Computers in Engineering*, 1992, pp. 411-419
9. Huang, Y. *Dimensional verification and correction of five-axis numerically controlled tool paths*, Ph.D. Dissertation, 1993, Iowa State University
10. Jerard, R. B., Hussaini, S. Z., Drysdale, R. L., and Schaudt, B., "Approximate Methods for Simulation and Verification of Numerically Controlled Machining Programs," *The Visual Computer*, 4, 1989, pp. 329-348
11. Kawashima, Y., Itoh, K., Ishida, T, Nonaka, S., and Ejiri, K, "A Flexible Quantitative Method for NC Machining Verification Using a Space-Division Based Solid Model," *The Visual Computer*, 7, 1991, pp. 149-157
12. Kincaid, D. and Cheney, W., *Numerical Analysis, Mathematics of Scientific Computing*, 1991, Brooks/Cole Publishing Company, Pacific Grove, CA
13. Menon, J. P. and Robinson, D. M., "Advanced NC Verification Via Massively Parallel Raycasting: Extensions to New Phenomena and Geometric Domains," *ASME Manufacturing Review*, 6, 2, June 1993, pp. 141-154
14. Meyers, D., Skinner, S., and Sloan, K., "Surfaces from Contours," *ACM Transactions on Graphics*, 11, 3, July 1992, pp. 228-258
15. Mortenson, M. E., *Geometric Modeling*, 1985, John Wiley & Sons Inc., New York, NY
16. Narvekar, A., Huang, Y. and Oliver, J. H., "Intersection of Rays with Parametric Envelope Surfaces Representing Five-Axis NC Milling Tool Swept Volumes," *Proceedings of ASME Advances in Design Automation* Vol 2, 1992, pp. 223-230
17. Oliver, J. H. and Goodman, E. D., "Direct Dimensional NC Verification," *Computer Aided Design*, 22, 1, 1990, pp. 3-10
18. Pegna, J. and Wolter, F.-E., "Designing and Mapping Trimming Curves on Surfaces Using Orthogonal Projection," *Proceedings of ASME Advances in Design Automation*, Computer Aided and Computational Design, DE-Vol. 23-1, 1990, pp. 235-245
19. Saito, T. and Takahashi, T., "NC Machining with G-buffer Method," Proceedings of SIGGRAPH '91. In *Computer Graphic,* 25, 4 (July 1991), pp. 207-216
20. Van Hook, T., "Real-Time Shaded NC Milling Display," Proceedings of SIGGRAPH '86. In *Computer Graphics* 20, 4 (August 1986), pp. 15-20
21. Voelcker, H. B. and Hunt, W. A., "The Role of Solid Modeling in Machine-Process Modeling and NC Verification," SAE Technical Paper No810195, Feb. 1981
22. Wang, W.P. and Wang K.K., "Geometric Modeling for Swept Volume of Moving Solids," *IEEE Computer graphics and Applications*, 6, 12, 1986, pp. 8-17

1. The ISU Research Foundation has applied for a US patent on this technology and has licensed it to Arete Software Company.