The Visual Computer

Approximate methods for simulation and verification of numerically controlled machining programs

R.B. Jerard [1], S.Z. Hussaini [1], R.L. Drysdale [2], and B. Schaudt [2]

[1] Department of Mechanical Engineering, University of New Hampshire, Durham, NH 03824, USA
[2] Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755, USA

Algorithms for simulation and verification of Numerically Controlled (NC) machining programs are presented. Compared to NC simulation based on conventional solid modeling systems, these models are designed to give approximate results, but with a substantial decrease in computer time. The surfaces of the part are discretized into a Surface Point Set (SPS) with a point spacing dependent on cutting tool size and shape, local surface curvature and the desired accuracy of the approximate simulation. The surface-surface intersection calculations of the solid modeling approach are replaced by the intersection of the surface of the tool movement envelope with straight lines emanating from the surface points. The methods are applicable to both 3 and 5 axis machining. Samples test cases are presented, and implementation and efficiency issues are discussed.

Key words: Numerical control machining – Simulation – Verification

1 Introduction

An NC program consists of a series of cutter tool movements which remove material from a piece of raw stock to create a prototype part, mold or stamping die. In this process the production engineer uses a high level language like APT to define the geometry and cutter sequence, calculate cutter offsets and produce a Cutter Location Data (CLDATA) File. The CLDATA file must then be postprocessed into a Machine Control Data (MCD) file which contains the instructions to control a specific machine tool. The next step, an often time consuming and costly one, is the validation of the NC program to eliminate any errors. This step is often accomplished by machining plastic, wax or wooden models. Manufacturers, who are under constant pressure to shorten the product development process, could benefit greatly from a method which produced error free NC programs. It is the goal of our project to move the validation process into software and thereby eliminate errors before any machining is attempted.

In order to better understand the various aspects of the validation process we define three important terms: *simulation*, *verification* and *correction*. *Simulation* of the geometric aspects of material removal requires modeling of the swept volume of each tool movement in the CLDATA file and modification of a geometric model of the workpiece to keep track of the material removal process. The *verification* process requires a comparison between the final workpiece model and a geometric model of the part. If the two models are properly oriented then a boolean difference operation will yield the null set if they the same. Unfortunately, from the standpoint of manufacturing practice, exact correspondence is not really the issue. The real question is whether the machined part is "close enough" to the nominal part geometry, i.e. will it fall within the allowable tolerance zone? This is a much more difficult question to answer, but an answer is essential if the validation process is to have practical value. For example, in the sculptured surface die production which provided the impetus for our work, the nominal part and the machined part are almost *never* in exact correspondence. The machined part may, however, be perfectly acceptable if the deviations do not exceed the specified tolerance. Finally, *correction* of the CLDATA file can only be achieved if there is a means in the validation model to relate unacceptable deviations to a tool movement. Our method reveals which tool movements caused the error and thereby aids the NC programmer in fixing the error. We are also

The Visual Computer (1989) 5:329–348

working on methods which automatically correct the CLDATA file.

Some currently available commercial systems can do verification without simulation by detecting interference between the tool and the part model. However, there is no simulation of the cutting because there is no model of the workpiece geometry and therefore it is impossible to determine if there is excess unremoved material. It is also possible to perform simulation without verification. Some commercial NC controllers actually display, for 2 1/2 D parts, a graphic image of the part produced by the NC program. Since the system does not have access to nominal part geometry it can't make the required comparison to perform verification. Although the details of these methods are not in the public domain it is probably a variation of the view-based methods which will be described later in this paper.

Some desirable characteristics of an NC program validation system are summarized:

1. Capable of detecting both gouges and areas of excess material which deviate from the nominal part geometry by more than a specified tolerance.
2. Able to relate cutting errors to a specific tool movement for subsequent correction.
3. Sufficient precision to find errors even though the ratio of the nominal size to tolerance deviation can be greater than 10000:1.
4. Determine volume of material being removed at any given time in order to select optimum machining conditions.
5. Visual display of machined part with color coded display of errors.

2 Background

An early example of NC simulation and verification is the work by Gossard at MIT who developed a method based on set theory for turning operations (Gossard and Tsuchiya 1978). The use of solid modeling was first investigated by Voelcker and Hunt who did an exploratory study of the feasibility of using the PADL Constructive Solid Geometry (CSG) modeling system for simulation of NC programs (Voelcker and Hunt 1981; Hunt and Voelcker 1982). Solid geometry modeling systems offer the possibility of doing both simulation and verification (Wallis and Woodwark 1984). The simulation is achieved by boolean subtraction of the tool movement volume from the workpiece model. Verification is achieved by performing boolean differences between the model of the workpiece and the desired part. Fridshal at General Dynamics modified the TIPS solid modeling package to do NC simulation (Fridshal et al. 1982). The problem with using the solid modeling approach is that it is computationally expensive. The cost of simulation using the CSG approach is reported to be $O(N^4)$, where N is the number of tool movements (Voelcker 1981). A complex NC program might contain ten thousand movements, making the computation intractable.

In order to increase efficiency, a number of approximate simulation methods have been devised (Chappel 1983; Wang 1985; Wang and Wang 1986a, b; Oliver 1986; Oliver and Goodman 1986; Van Hook 1986; Jerard et al. 1986; Atherton et al. 1987). These approximate methods are $O(N)$; i.e. the simulation time grows only linearly with the number of tool movements. A more complete discussion of these methods may be found in a previous paper (Jerard et al. 1988). In all of these techniques machining simulation is approximated by finding the intersection of the tool movements with straight lines. For example, in Chappel's method the surface of the part is approximated by a set of points. Vectors are created normal to the surface at each point. A vector extends until it reaches the boundary of the original stock or intersects with another surface of the part. During the simulation the length of a vector is reduced if it intersects the tool movement envelope. An analogy can be made to mowing a field of grass. Each vector in the simulation corresponds to a blade of grass "growing" from the desired object. As the simulation progresses the blades are "mowed down". The final length of the vectors correspond to the amount of excess material (if above the surface) or the depth of the gouge (if below the surface) at that point.

A somewhat different approach is the "view based" methods characterized by the independent work of several researchers (Wang 1985; Wang and Wang 1986a, b; Van Hook 1986; Atherton et al. 1987). They use a variation of the standard Z depth buffer hidden surface algorithm. A vector normal to the plane of the graphics screen is drawn at each pixel. In Wang's approach, intersections of

these lines with tool path envelopes are calculated with a scan line algorithm. For each pixel an extended Z buffer is maintained that contains the Z depth of all the entries and exits of the workpiece. The workpiece Z buffer is modified by performing boolean difference operations with the Z buffer of the tool movement swept volume. In such image space methods, errors which are not visible in the chosen viewing direction are undetected and generating another view of the part requires rerunning the entire simulation. Furthermore, small machining errors (e.g. less than 0.1 mm) are unlikely to be detected by a visual inspection of the computer graphics image. As Oliver (1986) points out, the critical issue in machining verification is whether the machined part features fall within a specified tolerance zone. The very large ratio between the nominal dimensions and the tolerance dimensions (10000:1 is not unusual) makes it unlikely that an image based simulation will be adequate.

Despite their limitations the view based methods do have several advantages. They are the best approximate method for determining how much material is being removed by any given tool movement. This is important for setting optimum values for feedrates. Each pixel can represent a volume of material, and material removal can be approximated with an accuracy dependent on the number of pixels and the size of the object. Another advantage is that since the simulations can be run without any information about the model of the nominal part it is relatively easy to make a general system. Both Wang and Atherton have produced commercial products based on their methods. However, if nominal part information is not used then verification is not possible.

Of the validation methods discussed, the solid modeling approach is, in theory, the most desirable because it is an exact representation of the geometry of the machining. In practice, however, the process can introduce errors dependent on the internal model representation. If, for example, a faceted representation of a curved surface of the part or of the tool envelope is used then accuracy may be compromised. For asmall number of tool movements the solid modeling method may work well for simulation. However, the difference operation required to perform verification can be very expensive computationally and it still leaves unanswered the primary question of whether the workpiece falls within the acceptable tolerance zone.

3 Description of algorithms

The approaches of Chappel, Oliver and the authors provide the necessary elements for a simulation, verification and correction method which is both accurate and efficient. In these methods material removal is simulated by modification of the vector length and verification is trivially accomplished by a comparison between the final vector length and the allowable tolerance variation of the surface. The algorithms work in object space and graphical display is a post simulation process. Our approach (Jerard et al. 1986, 1988, 1989; Drysdale and Jerard 1987; Drysdale et al. 1989) shares characteristics of the methods of Chappel and Oliver, but it also contains several novel features that improve efficiency and allow the user to make informed tradeoffs between the accuracy of the approximate simulation and the CPU time.

In order to realize a practical sytem it was necessary to concentrate on three key aspects, *discretization*, *localization* and *intersections*. In the *discretization* phase, a Surface Points Set (SPS) is calculated to approximate the surface of the part with a spacing between points that depends on the size and shape of the cutting tool, local surface curvature and the desired accuracy of the simulation. The methods for surface discretization are described in detail in the above references and are not repeated here. Two alternative approaches for extending the method to five axis machining are also explained.

4 Localization

Figure 1a shows a surface with a set of points and associated direction vectors normal to the surface. As each tool movement is processed the intersection of the direction vectors with the tool envelope can be calculated. It would be computationally expensive to calculate the intersection of all the direction vectors for each tool movement. It is therefore desirable to localize the calculations by eliminating from consideration the vectors which have no possibility of intersecting the tool movement envelope. If we restrict the generality of the method of the common case of three axis machining, and assume that the surface has only one z value for any given $x-y$ location (i.e., no undercuts) we can choose direction vectors to be parallel to the long axis (z axis) of the cutting tool (see Fig. 1b). Therefore, a vector can only be intersected if it lies directly

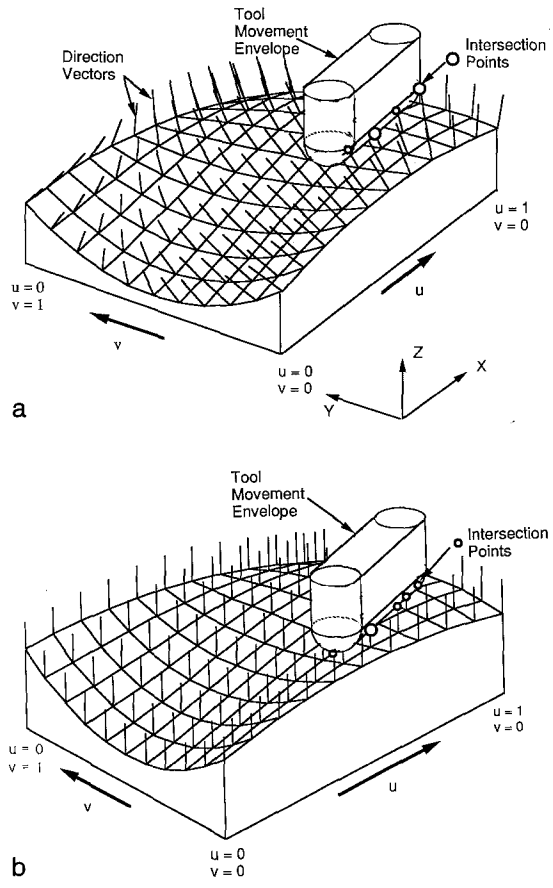


Fig. 1. a Tool path envelopes intersected with normal vectors. **b** Tool path envelopes intersected with z direction vectors

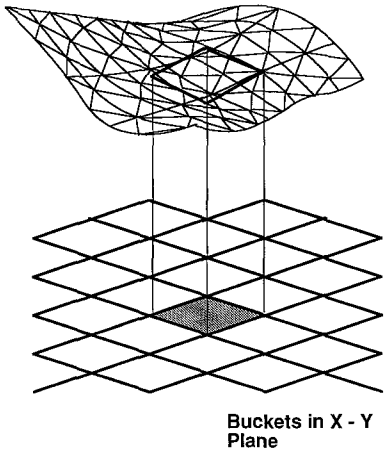


Fig. 2. Surface points can be hashed into buckets according to $x-y$ locations

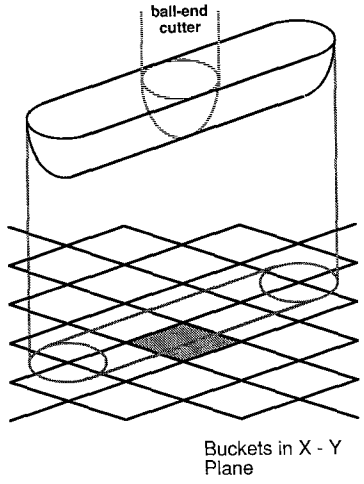


Fig. 3. Intersection calculations are performed only on points in buckets under the tool path shadow

under the tool path. This means that the points can be hashed into "buckets" as shown in Fig. 2. Localization is achieved by finding the set of buckets which lie under the shadow of the tool and examining only the points in those buckets (see Fig. 3). Only a small percentage of all the points are examined for each tool movement.

By choosing direction vectors in the z direction there is also an improvement in the efficiency of the intersection calculations (Jerard et al. 1988). The calculations are simplified by the fact that the x and y components of the direction vector are zero and also by the fact that we are only cutting on the bottom of the tool, so no intersection calculations need to be done for the sides of the tool. Another advantage of choosing vectors in the Z direction is that it is possible to simulate approximate material removal rates in a manner similar to the previously described view based methods.

5 Post simulation error analysis for three axis machining

The cutting error is related to the length of the direction vector after all tool paths have been processed. Choosing all vectors in the z direction introduces a potential problem whenever the surface normal deviates from the z direction. The problem becomes most apparent for nearly vertical surfaces

as shown in Fig. 4. The uncorrected estimate of the cutting error is the vertical distance between the surface point **P**, and the cut point **P'**. This effect causes the errors to be overestimated. No errors will be missed but points will be reported to be out of tolerance when in fact they are not. In order to remedy this problem it was necessary to implement a post simulation error analysis. We present two methods for addressing this problem: a quick but less accurate approach based on finding the component of the vertical error in the direction normal to the surface (the *Dot-Product* method) and the *Point-Triangle-Distance* (PTD) method which is slower but more accurate.

The question of how to esimate the actual error is not quite as simple as it might seem. The error could be estimated by measuring the distance: 1. From the machined point **P'** to the closest point on the design surface, 2. From the design surface point **P** to the closest point on the machined surface or, 3. From the design surface to the machined surface in the normal direction. These methods will yield different results under certain conditions. In practice, if the length of the trimmed direction vector is small (e.g. 0.1 mm) compared to the local radius of curvature (typically 10 to 1000 mm) then the surface may be considered to be essentially flat. Under these conditions it can be shown that there is little difference between these methods. The first option was the most straightforward one to implement with our simulation method. This method is accurate as long as the length of the trimmed direction vector does not greatly exceed the tolerance dimension. But under these conditions, the NC programmer knows that there is a problem needing correction, and the actual magnitude of the error is unimportant. Therefore care must be used only if the tolerances become significant relative to the dimensions of the object, a rather rare occurrence.

The *Dot-Product* method is illustrated in Fig. 5. Consider a point $\mathbf{P}(x, y, z)$ on the design surface. The corresponding point on the machined surface will be $\mathbf{P}'(x, y, z_{cut})$. If **N** is the normal to the design surface at **P**, the cutting error at this point is closely approximated by the length of the projection of the vector **PP'** on the normal. The vector **PP'** is the vector from the point on the design surface to its corresponding point on the machined surface.

This method works well for all points interior to a surface as long as the length of **PP'** is small relative the local radius of curvature as described above. This method will not work when:

1. Point **P** is very close to the boundary of a surface. As shown in Fig. 6, the closest distance from the machined point **P'** to the design surface point **P** is greater than the error estimated by the *dot-product* method.
2. Point **P'** is closer to some other part of the design surface as illustrated in Fig. 7.

The advantage of the *dot-product* method is that it is very fast, simple to implement, and in most cases, it results in a substantial improvement in accuracy over just using the vertical distance.

It would be more desirable to find the actual distance between each machined point **P'** and the design surface. For parametric surfaces this can be an extremely time consuming calculation requiring iterative methods. To simplify and accelerate the surface-to-point distance calculations the design surface is replaced by a polyhedral approximation. Inherent in our surface discretization scheme is the criterion that the deviations in this approximation be bounded by a tolerance (Drysdale et al. 1989). Thus the errors in surface-to-point distance calcu-

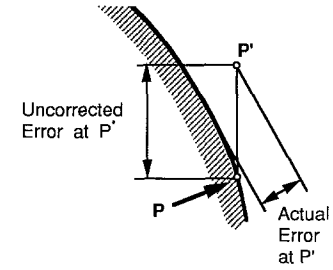


Fig. 4. Estimates of errors on nearly vertical surfaces are excessive

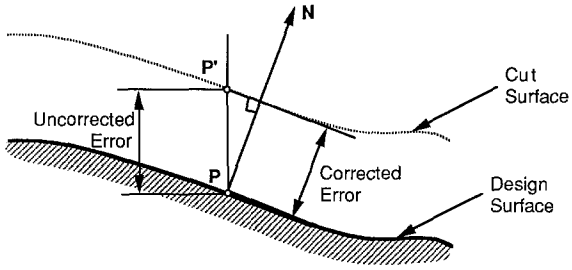


Fig. 5. Dot-Product method for vertical surface correction

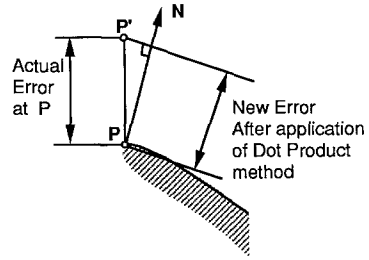


Fig. 6. Dot-Product method is inaccurate near surface boundaries

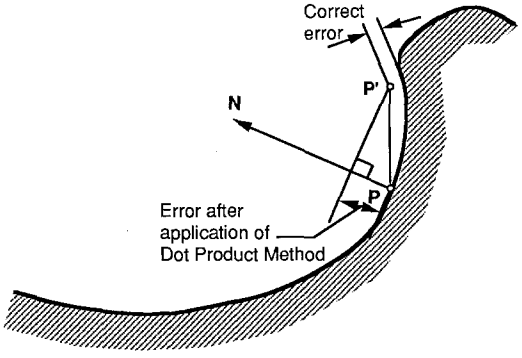


Fig. 7. Dot-Product method is incorrect if point is close to other surfaces

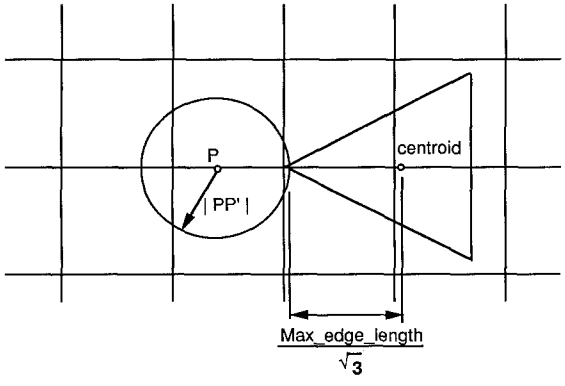


Fig. 8. Triangles close to point **P'** are examined for closest distance calculation

lations incurred by replacing the design surface with the polyhedral approximation are also bounded by this discretization tolerance. Since the polyhedron is described by triangular faces the problem is reduced to finding the distance from the machined point **P'** to the triangles of this surface.

The number of triangles on the surface is typically of the same order as the number of points. So, one cannot look at every triangle for every point **P'**. It would be too slow (for example, in our trials 139 cpu min compared to 2 min for the method described below). There are many ways one could select the triangles that are nearest to the point **P'**. Only those triangles less than the length $|\mathbf{PP'}|$ from **P'** need to be considered when determining the error at a given point.

Our approach consists of hashing all the triangles based on their centroid into an XY grid. After the hashing is completed, each hash table entry will be a linked list of triangles. The size of the XY grid is determined using the length of the longest edge on the surface (max_edge_length). Note that max_edge_length is the longest edge of any triangle in the entire polyhedral approximation to the surface. The surface discretization method that we use also attempts to keep the size of the edges of the triangles at about the same length, a feature which helps to make the hashing more efficient.

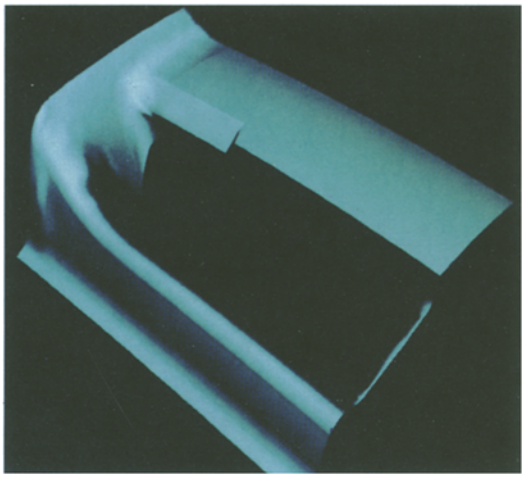
$$\text{minimum } \delta x \quad \text{or} \quad \delta y = \text{max\_edge\_length}/\sqrt{3}$$

For a given point $\mathbf{P}(x, y, z)$, the candidate triangles will be given by each of the hash boxes in the rectangular grid that lies under the circles around **P**, with radius $= |\mathbf{PP'}| + \text{max\_edge\_length}/\sqrt{3}$. This is illustrated in Fig. 8. It can be verified that every triangle that is close enough to be considered is selected. However, in an average, case, there are still many times more triangles selected than are really needed.

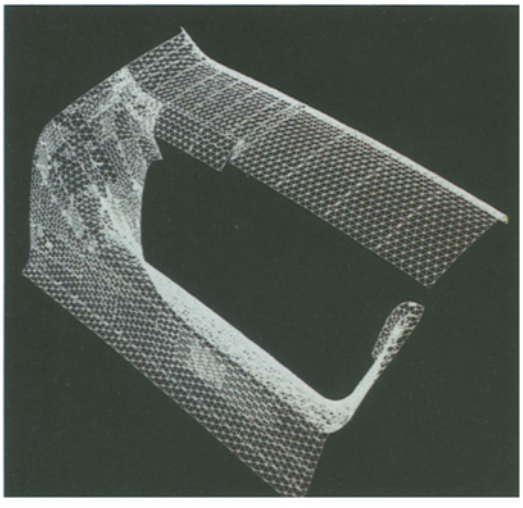
The structure of the algorithm is given below.

For each point **P** on the surface:
  If need to correct error at this point:
    Find the corresponding machined point **P'**
    Get list of triangles in the vicinity.
    Corrected Error at **P'** equals minimum distance from **P'** to any triangle in the vicinity

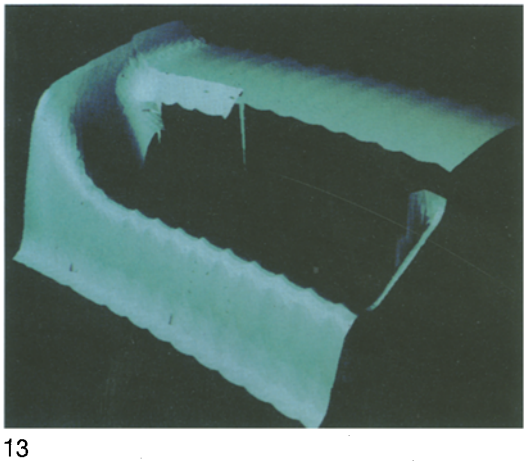
The details of the Point-Triangle-Distance calculations and Point-Edge calculations may be found in Appendix A, and in more detail in reference (Hussaini and Jerard 1988). Results from a test case are presented in Table 1. The table shows CPU times for simulations performed with and without error correction. The parabolic tool movements are



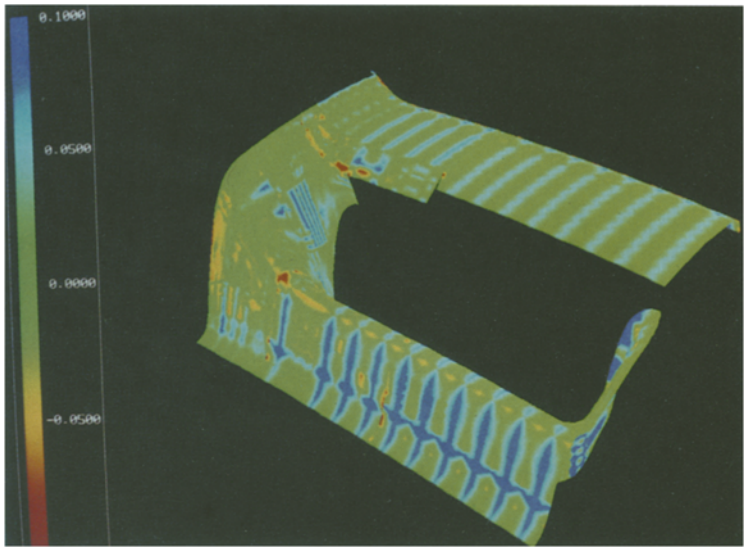
9



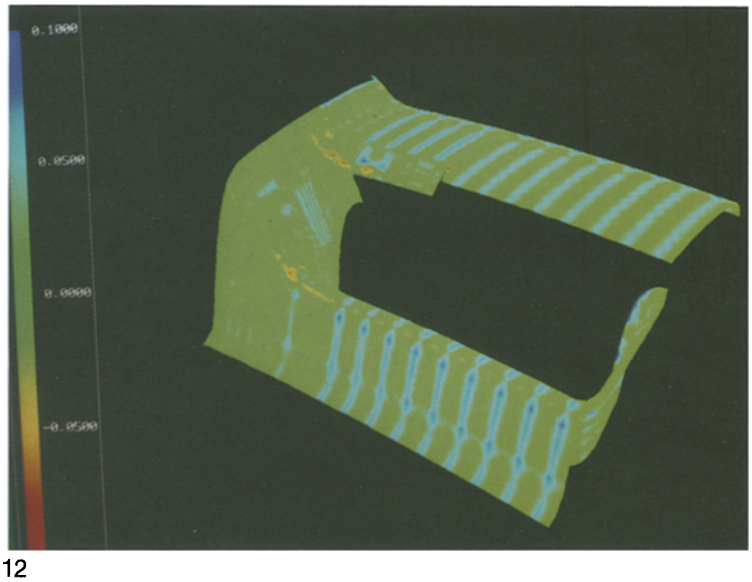
10



13



11



12

Fig. 9. Shaded image of air intake portion of a bumper

Fig. 10. Surface Point Set of bumper

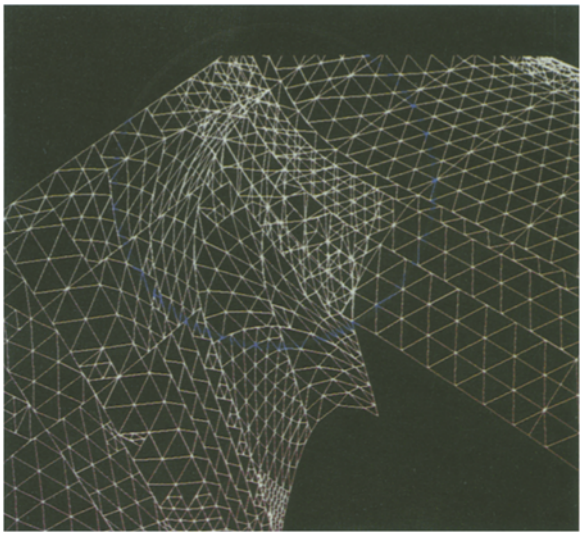
Fig. 11. Cutting errors without post simulation analysis shows excess material on vertical sides

Fig. 12. With post simulation correction. Note that overestimation of excess material on nearly vertical sides is corrected

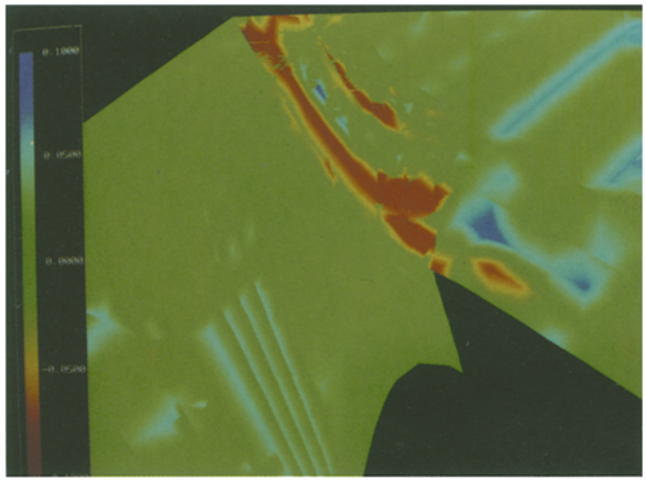
Fig. 13. Errors expanded by a factor of 50 show 0.1 mm cusps clearly

Table 1. Cutting simulation system performance for two test cases. Results show the number of CPU minutes on a DEC VAX 8650 for two different user-specified accuracies with three types of post simulation error analysis: *1.* no error correction; *2.* Dot product method; *3.* Point-Triangle-Distance method

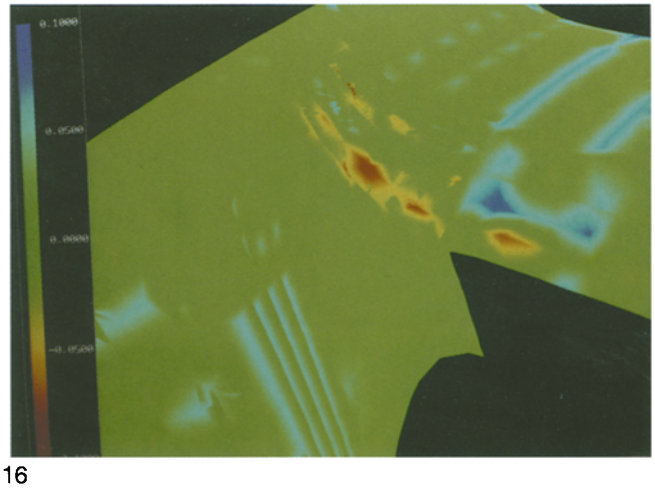
	Tool movements		Maximum simulation error – mm (in.)							
				2.5 (0.1)				0.75 (0.03)		
	Parabolic	Linear	Points	1	2	3	Points	1	2	3
Trunk	3,500	18,855	3,064	3.00	3.02	3.48	10,761	10.15	10.20	12.02
Bumper	8,100	45,411	7,348	5.43	5.47	7.58	16,789	11.53	11.58	17.16



14



15



16

Fig. 14. Wireframe picture of area where two surfaces overlap

Fig. 15. Errors are indicated in red on surfaces without post simulation correction

Fig. 16. Post simulation correction eliminates effect of overlap

broken up into the number of linear tool movements indicated. The maximum simulation error indicates the largest possible gouge that could possibly be missed. The Dot-Product method causes almost no discernible increase in CPU time while the PTD method causes an average increase of about 30%.

Figures 9–13 shows the results of simulation. The shaded image of the bumper test case is shown in Fig. 9. The surface point set used to perform the simulation is shown in Fig. 10. In Fig. 11 the uncorrected cutting errors are shown. Blue indicates excess material of 0.1 mm and red shows gouges of 0.1 mm with the color spectrum interpolating intermediate values. The corrected errors are shown in Fig. 12. The excess material was overestimated in Fig. 11 because of the nearly vertical surfaces, indicated by the darker blue lines. In Fig. 13 we magnified the errors by a factor of 50 to show qualitatively the cusps between the tool paths. The height of these cusps is a good indicator of the amount of expensive hand finishing required after machining. In addition to the post simulation graphical display it is possible to output a list of machined surface points which lie outside a given tolerance. We also output the tool paths which last cut the given point.

We also discovered an unexpected benefit when we implemented the Point-Triangle-Distance (PTD) method. The three dimensional complexity of sculptured surfaces makes it difficult to model them with current solid modeling systems. One problem with surface-based descriptions of parts is that topological consistency is not guaranteed (in contrast to solid modeling methods where it is). In the test cases provided by our industrial sponsor we occasionally find overlapping surfaces. The surfaces are actually some small distance from each other. Only one of the surfaces can be the drive surface for the NC program. This creates the possibility of large but irrelevant errors on the other surface. The points associated with one surface may be cut very accurately while the points on the other surface may be either severely gouged or show excess material. The simulation results can be quite confusing in cases like this. However, if we search for the closest distance between the machined surface point and all nearby surfaces (PTD method) then the inconsistency is eliminated and errors are only reported if the machined point is outside the tolerance zone of both surfaces. In Figures 14–16, the bumper case illustrates an area where two surfaces overlap. The simulation results shown in Fig. 15 indicate an area that has a 0.2 mm deep gouge. The size of the area is diminished significantly in Fig. 16 where the PTD method has been applied to eliminate the artifact produced by the two overlapping surfaces.

6 Five axis machining

In five axis machining the cutting tool has two additional degrees of freedom which control its angular orientation. The cutter can be placed at any cartesian coordinate with its direction cosines chosen to place it at a favorable orientation relative to the surface being machined. Typically, in sculptured surface face milling the tool is oriented normal to the surface with a tilt of a few degrees in the direction of movement. The large diameter flat-end or fillet-end cutter can remove material at a much faster rate than the three axis ball end cutter and smooth surfaces requiring a minimum of hand finishing can be machined efficiently. Despite the advantages of five axis machining it is not widely used because of the difficulty of programming the complex tool paths and avoiding interference between the cutting tool and the adjacent surfaces. Traditional methods for finding program errors, such as plotting the tool path, are not effective at showing both tool position and angular orientation. The development of a five axis simulation and verification capability would greatly aid in the effective utilization of five axis machining capacities.

During our investigation several important facts became evident:

1. The surface discretization method that we use is equally applicable to three or five axis simulation and no modifications to these algorithms are necessary, thus the inherent advantages of using a minimum point set with bounds on the simulation error is retained. However, in five axis machining the tool shape is usually a flat-end or fillet-end cutter, a factor which greatly increases the number of points required for a given level of user specified accuracy (Drysdale et al. 1989).
2. The localization method used for three axis simulation is no longer applicable. The versatility of five axis machining allows it to machine surfaces with multiple z values and the simplicity

of having all direction vectors pointing in the same direction is no longer possible.
3. The intersection calculations are complicated by both the more complex nature of the tool path envelopes and the fact that the surface vectors will not all be in the same direction.

7 Five axis localization

We have investigated two methods for localizing the calculations; the first is based on using surface normals of short length (short normal method), and the other on a concept of using average normal vectors (average normal method). With the first approach we limit the length of the direction vectors to a value which is small relative to the overall dimensions of the part but very large relative to the magnitude of the part tolerances. The long "blades of grass" which protruded from the surface in Fig. 1 have now become short "stubble". By limiting the length of the vectors it is possible to easily localize the calculations with a bucketing method which varies only slightly from the approach used for three axis localization. The length of the short normals is selected based on the surface tolerance. For example, if the surface tolerance is 0.1 mm, then a normal length of 1 mm is more than adequate to detect errors in the range of the interest. In our simulations the vector protruded 3 mm both into and out of the surface.

The points are still sorted into buckets but points in some buckets outside the shadow of the tool must also be examined as illustrated in Fig. 17. It is only necessary to expand the shadow of the tool envelope by the length of the direction vector to insure that all candidate points are examined. The points in the buckets are also sorted by Z value and therefore the Z extents of the tool movement are used to further localize the calculations. This method is similar in many respects to that used by (Oliver 1986; Oliver and Goodman 1986).

The alternate approach (average normal) consists of choosing the direction vector for each point from the predefined set of direction vectors. We have one bucket set for each direction vector and points are assigned to the bucket set whose direction vector most closely matches the point's normal vector. In general we would expect the direction vector for a given bucket set to be close to the average of all the normals of points in that set. If the angle between the average normal vector and any point

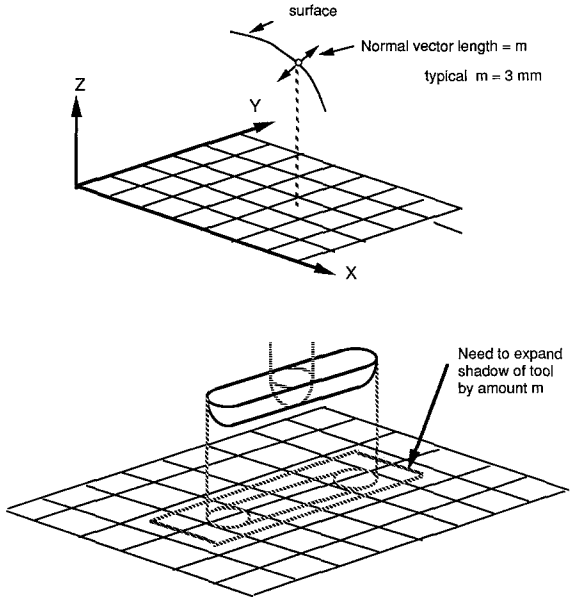


Fig. 17. Five axis localization using short normal approach

in its set is small then errors of the type discussed in Sect. 5 will also be small. Each of the bucket sets can be treated exactly the same as the three axis case.

In both of these approaches (short normal and average normal) the ability to simulate material removal has been lost. Based on our definitions of simulation and verification in Sect. 1 we are now doing verification without simulation. However, unlike most other verification methods we are able to detect not only gouging but also areas where excess material has been left.

As a simple example of a bucket set based on average normals imagine a cube with a direction vector associated with each face. Our predefined set of direction vectors consists of the vectors in the following directions; up, down, left, right, front, and rear. Thus when the normal vector of a point on the surface is generally to the left, we will use the left vector as our direction vector for that point and so on.

Thus in the example above, we can simulate the cutting six times. Note that, since each point only resides in one bucket set, this does not imply that there will be six times as many intersection calculations; indeed the number of intersection calculations should not be significantly larger than with

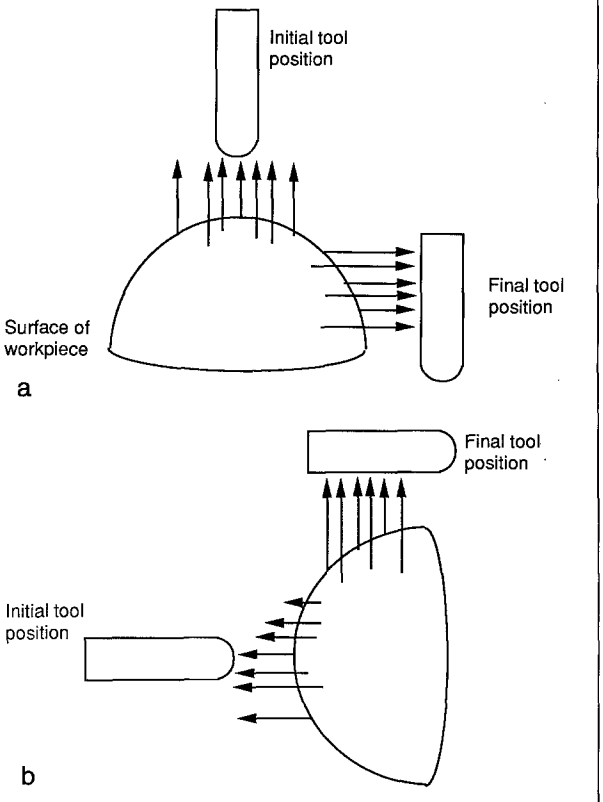


Fig. 18a, b. Five axis localization using average normal approach

the short approach. We can simulate cutting for the points whose direction vectors are up, as we did in the 3 axis machining (see Fig. 18a). For points whose direction vector is to the right, we rotate the workpiece so that the right direction vector is vertical (see Fig. 18b). Now for each tool movement, we rotate the tool positions by the same rotation used on the workpiece and simulate cutting on those points with a right direction vector. Notice that the direction vector associated with these points is now vertical.

Now localization can be achieved. When a direction vector is chosen for a point, the rotations required to make the direction vector vertical and the set of buckets that the point goes in are known. Thus when a direction vector is chosen, we can assign the point to a set of buckets, compute the rotated coordinates of the point, and store the point in a bucket within the chosen set of buckets based on the rotated x and y coordinates of the point. In other words, we are able to localize for each of our 6 directions.

In general, we may have more than the 6 predefined direction vectors in this example and increasing this number decreases the difference between the true normal at each point and the chosen direction vector, thus decreasing the magnitude of the errors of the type described in Sect. 5. This means that the average normal approach could also be used to simulate three axis machining and eliminate the need for the post simulation analysis described in that section.

8 Five axis intersection calculations

The intersection calculations were accomplished in almost the same manner for both the "short normal" and the "average normal" approach. The objective of these calculations is to determine the points of intersection between the tool path envelope and the direction vectors. When all the points have been assigned to a bucket using either the single bucket set of the "short normal" approach or into one of the multiple bucket sets for the "average normal" approach, we can proceed with the cutting simulation. In the case of the average normal method the rotated coordinates of each point have been precomputed, so for each tool movement, we rotate the tool and generate the tool envelope in the rotated coordinate system. Now that we have the tool envelope, we can determine which buckets to examine and proceed in a manner closely analogous to the method used for three axis machining. For the short normal method we determine the intersection between the tool path envelope and the direction vector associated with each surface point. For the average normal method the intersection calculations are similar, except the direction vector is always vertical (in the rotated coordinate system). Intersection calculations should determine the points where the direction vector enters and exits the tool envelope. The sign of the entry and exit distances indicate whether the intersections are outside (positive) or inside (negative) the surface.

Phantom gouges

For a five axis tool movement it is possible for the direction vector to have multiple sets of entry-

exit pairs. We treat each of these pairs independently and, except for one special case, the length of the direction vector is changed whenever the entry value is less than the previous length. However, if both the entry and exit distances of a particular entry-exit pair are negative we ignore that pair. Two things could have happened in this case. The point could have actually been gouged or the tool could have passed below the point but still be outside the surface. We label the latter case a "phantom gouge" since the tool has not really penetrated the surface and is simply machining on another surface which happens to lie below the point of interest. If the tool actually protruded far enough into the surface so that both entry and exit values are negative then our surface subdivision method guarantees that another point must have been gouged and the error will be detected even though this particular point will not report the error.

Tool movement envelopes

The difficulty of the intersection calculation depends on both the tool type and the complexity of the tool movement envelope. In general, there has been very little published research (Wang and Wang 1986b) on the problem of defining the exact mathematical description of envelopes of tool movements, and a complete discussion of this subject is beyond the scope of this paper. Three axis tool movements may be handled in a straightforward manner (Jerard et al. 1988) but five axis envelopes are much more difficult. The tip of the tool translates in a straight line while the angle between the long axis of the tool at the starting and end locations is linearly interpolated. The resulting envelope surfaces are quite complex. Our current approach approximates the envelope with a number of simpler movements in which the angular orientation of the cutting tool is assumed constant. This introduces both computational inefficiency and the possibility of additional simulation error. We are presently working to quantify the magnitude of this error and also develop more efficient representations of the envelope.

The intersection calculations for the envelope of a flat-end cutter with no rotational change are described in Appendix B. The flat-end cutter and fillet-end cutters are the most commonly used tool shapes for five axis machining. The derivation for a ball-end cutter is quite straightforward but of limited usefulness since this shape is not used very often in five axis cutting. The fillet-end cutter is commonly used but it is also more complex, and we are still working on good representations for this case. This problem reduces to finding the swept envelope of a torus translating and rotating through space; we believe this to be a rather challenging problem.

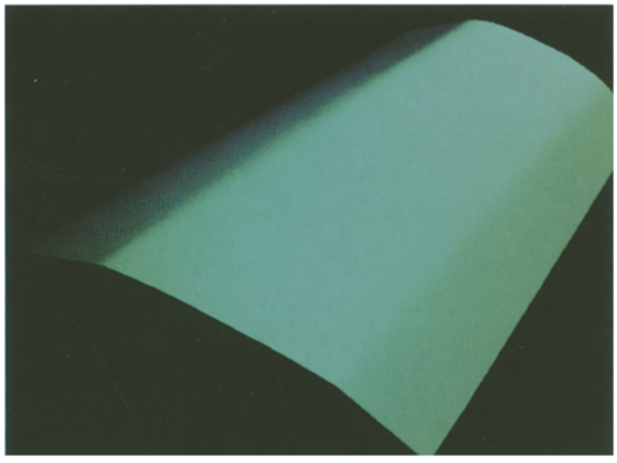
Results

The results of simulating the five axis machining of a stamping die for a door are shown in Fig. 19–21. The shaded image of the door is shown in Fig. 19, the discretized surface in Fig. 20, and a color coded image of the results of the simulation in Fig. 21. There was significant gouging of about 0.025 inches (0.6 mm – note that English units are used since the door units were in inches and the color scale on the figure is in inches). Table 2 shows the CPU times for twelve different simulations of the door using both the short normal and average normal localization methods.

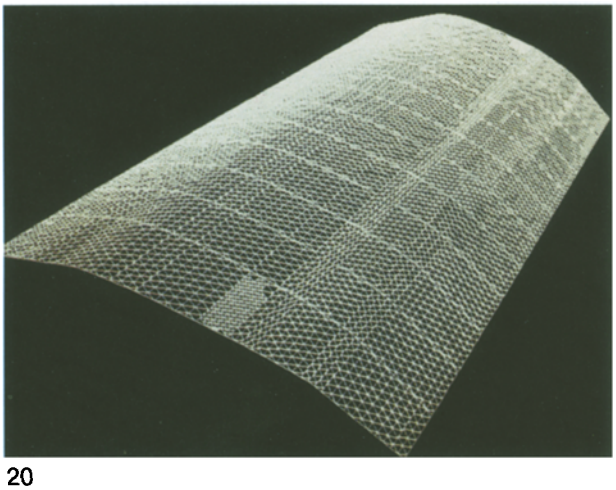
Using the methods described in our previous work (Drysdale et al. 1989) the maximum simulation error was determined to be 23.2 mm for the 3,035 point case, 11.4 mm for the 6,012 point case and

Table 2. CPU minutes on a DEC VAX 8650 for five axis cutting simulation performance for door die machining program. Short normal and averge normal localization methods were used for three different user specified levels of accuracy with 3,035, 6,012 and 12,047 points. Short normal vectors were 6 mm long. The original 38,964 five axis tool movements were subdivided into 85,786 and 97,275 movements such that the angle between the starting and the ending tool orientations was always less than 0.6 degrees and 0.4 degrees respectively

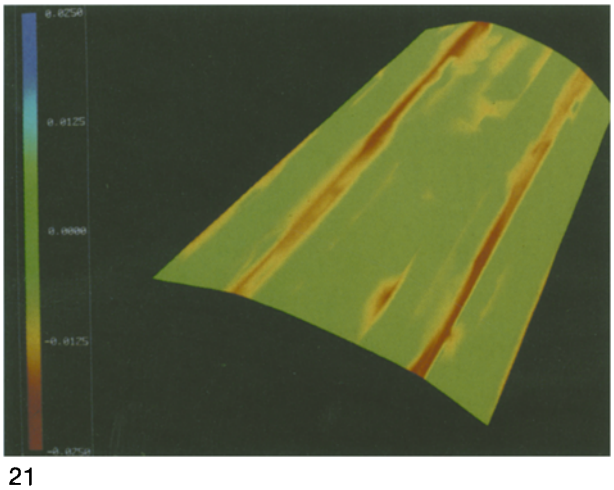
Angle tolerance	Divided movements	3,035 Points		6,012 Points		12,047 Points	
		Short	Average	Short	Average	Short	Average
0.6	85,786	35	46	63	68	120	115
0.4	97,275	40	50	69	78	136	136



19



20



21

Fig. 19. Shaded image of door

Fig. 20. Surface Points set (SPS) for door

Fig. 21. Machining from five axis NC simulation program

6.3 mm for the 12,047 point case. The maximum simulation error refers to the maximum depth gouge that could have been undetected by the simulation if the tool plunges in between the points in the SPS. Even though these numbers seem very large compared to the errors of interest it must be remembered that they represent worst case situations. As a practical matter all three test cases were equally effective at finding the NC program errors.

The results shown in Table 2 are preliminary and we anticipate that the times will decrease when better methods for representing the envelope are developed. We approximate the changing orientation of the tool by subdividing the tool movement into smaller movements in which the angular orientation does not change by more than a specified tolerance. This tolerance was set to 0.4 and 0.6 degrees for the test cases given in Table 2. The smaller tolerance results in more envelope intersection calculations and longer CPU times, but the error caused by approximating five axis tool movements with three axis movements is decreased.

9 Discussion

Methods for simulation and verification of Numerically Controlled machining have been presented. The methods are based on a discretization of the surface into a set of points. Cutting is simulated by calculating the intersection of vectors which pass through the surface points with tool path envelopes. Three axis machining can be simulated in a particularly simple manner by choosing the direction of the vectors to be the same everywhere, parallel to the long axis of the tool. Calculations are localized by hashing the points according to their $x-y$ coordinates into a set of "buckets". Localization for five axis machining was accomplished by two alternative methods, "short normals" and "average normals". In general, the short normal approach yielded somewhat faster CPU times but the advantage diminishes as the number of points increases, and in one of the cases the average normal approach is actually faster. This is true despite the fact that it was necessary to rotate both points and tool movements into the various coordinate systems. The amount of time necessary to perform these rotations seems to be quite small compared to the time spent doing the intersection calculations, and the intersection cal-

culations for the average normal case were simplified by taking advantage of the fact that the direction vectors are always the same once the rotation has taken place. It would appear that the amount of time spent on coordinate transformations becomes proportionately less when the number of points is increased.

The average normal approach has the advantage that the magnitude of errors of arbitrary size can be determined, while in the short normal approach only the magnitude of errors less than the length of the vectors can be discriminated. The existence of errors larger than the normal length will, of course, be detected but the actual magnitude of the errors will be unknown. In our specific implementation of the short normal approach the magnitude of any errors larger than 3.0mm was unknown. Since any errors larger than 0.1mm will require a revision of the NC program then any error larger than 3.0mm will also require revision.

In our original three axis implementation we are able to calculate approximate volumes of material removal since all the direction vectors are in the z direction. This is not possible for either of our five axis methods. The view based extended Z buffer methods discussed in the text appear to be the best method for approximating material removal rates. On the other hand, the object space methods presented in our paper appear to offer the best method for determining if the machined object and the nominal part geometry are within a specified tolerance of each other.

Better methods for representing the tool movement envelopes should substantially decrease CPU times for both the three and five axis methods. In sculptured surface machining, the tool movements tend to be very short. For example, in our five axis simulations each tool movement was about 2–3 mm in length. Our current method of doing three axis approximations to the five axis movements requires that we do further subdivision. Since the tool diameter was around 100 mm the length of the motion is rather small compared to the size of the cutting tool extents. We always look at all points falling within the tool extents for each tool movement. This means that each point will be examined for a possible intersection many times before the tool has completely passed over it. We found that, on the average, each point's direction vector required 35 intersection calculations. Our current research is directed toward finding ways of geometrically modeling the surfaces of a series of five axis tool movements so that each point only needs to be examined once for those movements. The complexity of this problem is exacerbated by the need to find models that will work for a variety of cutting tool shapes. Our preliminary results using the envelope of a five axis flat end cutter decreased the total simulation times shown in Table 2 by over 50%.

One interesting question that was raised by reviewers of this paper was whether the methods used for five axis machining could be used for three axis verification and thereby eliminate the need for the error correction schemes described in Sect. 5. At the present time we have not done enough experimentation to make a fair comparison, but some preliminary work indicates that this is a fruitful area to investigate. We intend to report complete results in a future paper.

Acknowledgements. This research supported in part by the National Science Foundation under contracts DMC-851262, DMC-8704147 and by the Ford Motor Company. The programming efforts of Mr. Paul Leclerc and Mr. Ken Hauck are also gratefully acknowledged.

References

Atherton PR, Earl C, Fred C (1987) A graphical simulation system for dynamic five-axis NC verification. Proc Autofact, SME, Dearborn, MI (November 1987), pp 2-1, 2-12

Chappel IT (1983) The use of vectors to simulate material removed by numerically controlled milling. Computer Aided Design 15(3):156–158

Drysdale RL, Jerard RB (1987) Discrete simulation of NC machining. Proc 3rd Annual ACM Symposium on Computational Geometry (June 1987), pp 126–135

Drysdale RL, Jerard RB, Schaudt B, Hauck K (1989) Discrete simulation of NC machining. Algorithmica 4(1):33–60

Fridshal R, Cheng KP, Duncan D, Zucker W (1982) Numerical control part program verification system. Proc Conf CAD/CAM Technology in Mechanical Engineering (March 1982) MIT Press, pp 236–254

Gossard DC, Tsuchiya FS (1978) Application of set theory to the verification of NC tapes. Proc North American Metalworking Conf (April 1978)

Hunt WA, Voelcker HB (1982) An exploratory study of automatic verification of programs for numerically controlled machine tools. Production Automation Project Tech Memo No 34, Univ Rochester (January 1982)

Hussaini SZ, Jerard RB (1988) Post simulation error analysis of NC cutting simulation. Tech Memo 88-01, Dept Mechanical Engineering, Univ New Hampshire (August 1988)

Jerard RB, Hauck K, Drysdale RL (1986) Simulation of numerical control machining of sculptured surfaces. 15th Int Sym-

posium on Automotive Technology and Automation (ISATA), no 86057. Automotive Automation, Croydon, UK, Flims, Switzerland (October 1986)
Jerard RB, Drysdale RL, Hauck K (1988) Geometric simulation of numerical control machining, Proc ASME Int Computers in Engineering Conf, San Francisco 2:129–136
Jerard RB, Drysdale RL, Hauck K, Schaudt B, Magewick J (1989) Methods for detecting errors in sculptured surface machining. IEEE Comput Graph Appl 9(1):26–39
Oliver JH (1986) Graphical verification of numerically controlled milling programs for sculptured surface parts. Doctoral Dissertation, Michigan State Univ, E Lansing
Oliver JH, Goodman ED (1986) Color graphic verification of N/C milling programs for sculptured surface parts. First Symposium on Integrated Intelligent Manufacturing. ASME Winter Annual Meeting, Anaheim, California
Van Hook T (1986) Real-time shaded NC milling display. Comput Graph (Proc SIGGRAPH) 20(4):15–20
Voelcker HB, Hunt WA (1981) The role of solid modeling in machining – process modeling and NC verification. SAE Tech Paper 810195
Wallis AF, Woodwark JR (1984) Creating large solid models for NC toolpath verification. Proc CAD-84 Conf Brighton, UK, Butterworths, pp 455–460
Wang WP (1985) Integration of solid modeling for computerized process planning. ASME publication, PED 19:177–187
Wang WP, Wang KK (1986a) Real-time verification of multiaxis NC programs with raster graphics. IEEE Proc of 1986 Int Conf on Robotics and Automation, San Francisco, (April 1986), pp 166–171
Wang WP, Wang KK (1986b) Geometric modeling for swept volume of moving solids. IEEE Comput Graph Appl 6(12):8–17

Appendix A

Point-triangle and point-edge distance Calculations

To implement the PTD method described in Sect. 5 it is necessary to determine the distance of a point from triangle (see Fig. 22)

p1, **p2**, **p3** are the vertices of the triangle
n is the unit normal to this triangle
q is the point of intersection of the perpendicular from **p** with the plane containing the triangle
t is the parameter along the line joining **q** and **p**. $t=0$ at **p**
u is the parameter along the line joining **p2** and **p1**. $u=0$ at **p1**, $u=1$ at **p2**.
w is the parameter along the line joining **p3** and **p1**. $w=0$ at **p1**, $w=1$ at **p3**.

First step is to find **q** and t:

$$\mathbf{q}=\mathbf{p}-t\mathbf{n} \tag{A.1}$$

also

$$\mathbf{q}=\mathbf{p1}+u(\mathbf{p2}-\mathbf{p1})+w(\mathbf{p3}-\mathbf{p1}) \tag{A.2}$$

$$\Rightarrow \mathbf{p1}-\mathbf{p}=-t\mathbf{n}-u(\mathbf{p2}-\mathbf{p1})-w(\mathbf{p3}-\mathbf{p1}) \tag{A.3}$$

Take dot product with **n** on either side of (A.3):

$$t=(\mathbf{p}-\mathbf{p1})\cdot\mathbf{n} \tag{A.4}$$

Once t is determined we can use (A.1) to get **q**. From (A.2) we get:

$$u=\frac{(\mathbf{p3}-\mathbf{p1})\,\mathbf{x}\,(\mathbf{q}-\mathbf{p1})}{(\mathbf{p3}-\mathbf{p1})\,\mathbf{x}\,(\mathbf{p2}-\mathbf{p1})} \tag{A.5}$$

and

$$w=\frac{(\mathbf{p2}-\mathbf{p1})\,\mathbf{x}\,(\mathbf{q}-\mathbf{p1})}{(\mathbf{p2}-\mathbf{p1})\,\mathbf{x}\,(\mathbf{p3}-\mathbf{p1})} \tag{A.6}$$

To calculate u and w from above equations, we just consider either the x or y or z components (preferably the component with the largest magnitude in the denominator) of the cross products. Using u and w we can determine if **q** is inside the triangle or outside. If **q** is inside the triangle, then the distance of **p** from the triangle is given by t. Also, if $t>0$, the point **p** is *above* the triangle and the sign of the distance will be positive. If **q** is not inside the triangle, we determine on which side of the triangle it is. The closest distance of the point from the triangle is then given by the distance of the point from the nearest edge of vertex. Similar conditions from the nearest edge can be found.

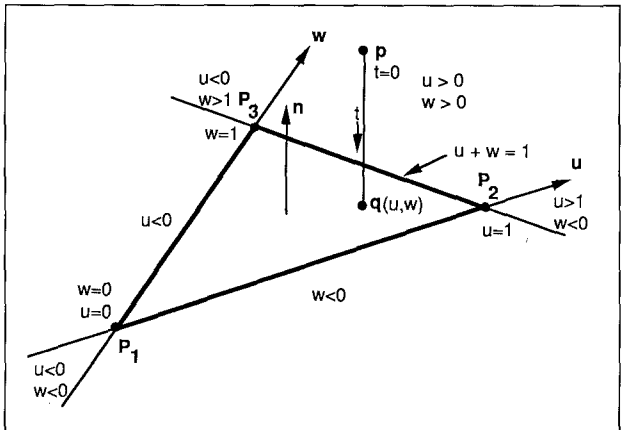


Fig. 22. Point-triangle distance calculation

q is inside if $u \geq 0, w \geq 0$, and $u+w \leq 1$
p is closest to **p1** if $u<0$, and $w<0$
p is closest to **p2** if $u>1$, and $w<0$
p is closest to **p3** if $u<0$, and $w>1$

If two vertices of the triangle coincide, the problem reduces to finding the distance from an edge. Similarly, if all three vertices coincide we find the distance of the point under consideration from one of the vertices.
For points which lie closest to the edge of a surface it is necessary to find the distance of a point **p** from an edge **e**. Refer to Fig. 23 for an illustration.

p1, p2 are the two end points of the edge **e**.
q is the point of the intersection of the perpendicular from **p** with the line joining **p2** and **p1**, that is, the edge **e**.
t is the parameter along the line joining **p2** and **p1**. $t=0$ at **p1**, and $t=1$ at **p2**.

$$\mathbf{q}=\mathbf{p1}+t(\mathbf{p2}-\mathbf{p1}) \tag{A.7}$$

$$(\mathbf{q}-\mathbf{p})\cdot(\mathbf{p2}-\mathbf{p1})=0 \tag{A.8}$$

substituting (A.7) in (A.8):

$$[(\mathbf{p1}\cdot\mathbf{p})+t(\mathbf{p2}-\mathbf{p1})]\cdot(\mathbf{p2}-\mathbf{p1})=0 \tag{A.9}$$

on expanding this equation we get:

$$t=\frac{(\mathbf{p}-\mathbf{p1})\cdot(\mathbf{p2}-\mathbf{p1})}{|\mathbf{p2}-\mathbf{p1}|^2} \tag{A.10}$$

If $t \leq 0$ distance = distance of **p** from **p1**.
else if $t<1$ distance = distance of **p** from **p2**.
else distance = distance of **p** from **q**, where **q** is given by Eq. A.7.

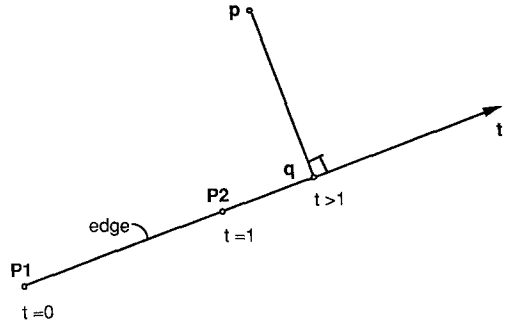


Fig. 23. Point-edge distance calculation

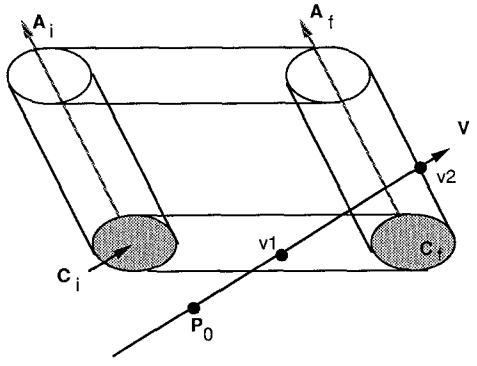


Fig. 24. Intersections with the tool path envelope generated by a translation of the tool

Appendix B

Five Axis intersection calculations

In this appendix we derive the intersection calculations for the special case of a five axis flat-end tool movement in which the angular orientation of the tool is constant.
In the Fig. 24, the tool is shown at its begining and end positions. $\mathbf{C}_i$ and $\mathbf{C}_f$ are the initial and final locations of the center of the tool bottom. $\mathbf{A}_i$ and $\mathbf{A}_f$ give the corresponding orientations of the axis of the tool. Since the tool undergoes no rotations $\mathbf{A}_i$ and $\mathbf{A}_f$ are parallel. $\mathbf{C}_i$, $\mathbf{A}_i$, $\mathbf{C}_f$, and $\mathbf{A}_f$ completely describe a tool path. $\mathbf{P}_0$ is the point on the design surface which has a vector **V** (which is normal to the surface at $\mathbf{P}_0$) projected from it. We need to determine the points of intersection between the vector **V** and the envelope generated by the tool path. Note that in the average normal case the equations that follow are simplified somewhat by the fact that **V** is always (0, 0, 1) resulting in slightly faster intersection calculations.
The envelope surface can be broken down to following parts:

(a) A cylindrical surface in the beginning of tool path,
(b) A flat plane in the front,
(c) The surface generated by sweeping a circle from $\mathbf{C}_i$ to $\mathbf{C}_f$,
(d) Another flat plane in the rear, and
(e) A cylindrical surface at the end of the tool path.

Intersecting the vector **V** with each of the above surfaces will yield a series of intersection points. If we use a parameter v along the vector **V** and

determine the v's for each of these points, the minimum and maximum values of v correspond to the entry and exit points of intersection of **V** with the envelope.

In the figures:
D gives the direction of tool movement
N is the normal to the plane parallel to tool movement
R is the radius of the tool
L is the length of the tool
x is the distance through which the tool moves
u is a parameter along **D**. $u=0$ at $\mathbf{C}_i$, $u=x$ at $\mathbf{C}_f$
t is a parameter along the tool axis. $t=0$ at the tool bottom, $t=L$ at the top.
v is the parameter along **V**. $\mathbf{v}=0$ at $\mathbf{P}_0$.
E is a point on the bottom edge of the envelope
F is a point on the top edge of the envelope.
p is any point on the surfaces mentioned earlier.
C is the center of the tool bottom at an intermediate position of tool
A is the axis of tool at **C**.

The following equations define the relationships between some of these parameters:

$$\mathbf{C}=\mathbf{C}_i+u\mathbf{D} \tag{B.1}$$

$$\mathbf{A}=\mathbf{A}_i=\mathbf{A}_f \tag{B.2}$$

$$x=|\mathbf{C}_f-\mathbf{C}_i| \tag{B.3}$$

$$\mathbf{D}=\|\mathbf{C}_f-\mathbf{C}_i\| \tag{B.4}$$

$$\mathbf{N}=\|\mathbf{D}\times\mathbf{A}_i\| \tag{B.5}$$

$\mathbf{A}$, $\mathbf{A}_i$, $\mathbf{A}_f$, $\mathbf{D}$, and $\mathbf{N}$ are unit vectors.

Intersection with front and rear planes

As the tool moves from $[\mathbf{C}_i, \mathbf{A}_i]$ to $[\mathbf{C}_f, \mathbf{A}_f]$, it sweeps out two flat surfaces at the front and rear. These surfaces are parallel to the plane of the Fig. 25, and lie as distances R and $-R$ from it. **E** is a point along the bottom edge of these surfaces, and **F** is on the top edge. Consider an intermediate position of the tool [**C**, **A**]. The corresponding point **E** at this position will be:

$$\mathbf{E}=\mathbf{C}\pm\mathbf{RN} \tag{B.6}$$

plus for the front plane, and *minus* for the rear plane.
A line at **E** which is parallel to **A** lies on the front on rear planes. This line is given by any point **p** such that:

$$\mathbf{p}=\mathbf{E}+t\mathbf{A} \tag{B.7}$$

Equations (B.1) through (B.7) completely define the

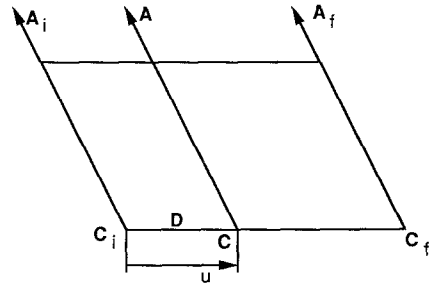


Fig. 25. Some of the tool path parameters

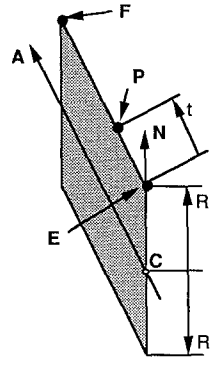


Fig. 26. Cross section of the envelope at [**C**, **A**]

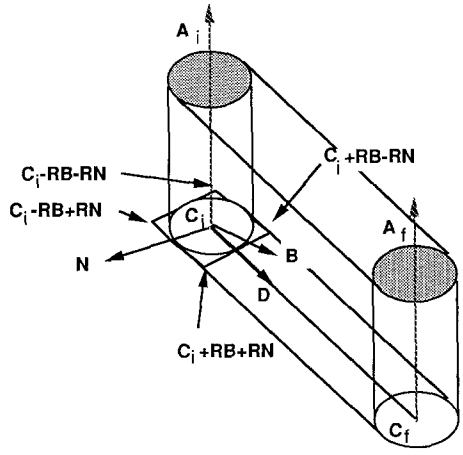


Fig. 27. Tool path extent calculation

two planes at the front and rear. To determine the intersections with **V** of these planes, we add another equation:

$$\mathbf{p}=\mathbf{P}_0+v\mathbf{V} \tag{B.8}$$

The above equations can be solved for v, u and t. The results are given below:

$$v=\frac{[\mathbf{C}_i-\mathbf{P}_0\pm R\mathbf{N}]\cdot[\mathbf{D}x\mathbf{A}_i]}{\mathbf{V}\cdot[\mathbf{D}x\mathbf{A}_i]} \tag{B.9}$$

$$u=-\frac{[\mathbf{C}_i-\mathbf{P}_0\pm R\mathbf{N}]\cdot[\mathbf{A}_i x\mathbf{V}]}{\mathbf{D}\cdot[\mathbf{A}_i x\mathbf{V}]} \tag{B.10}$$

$$t=-\frac{[\mathbf{C}_i-\mathbf{P}_0\pm R\mathbf{N}]\cdot[\mathbf{D}x\mathbf{V}]}{\mathbf{A}_i\cdot[\mathbf{D}x\mathbf{V}]} \tag{B.11}$$

It is a valid intersection only if $u\in[0, x]$ and $t\in[0, L]$. Any other intersections must be ignored.

Intersection with surface swept by tool bottom

The tool bottom is circular. The surface swept by the bottom is therefore the same as that obtained by sweeping a circle linearly through space. For arbitrary position [**C**, **A**] of the tool, a point **p** on the circular edge at the tool-bottom is defined by the following equations:

$$(\mathbf{p}-\mathbf{C})\cdot\mathbf{A}=0 \tag{B.12}$$

$$|\mathbf{p}-\mathbf{C}|=R \tag{B.13}$$

If the vector **V** intersects this circle:

$$\mathbf{p}=\mathbf{P}_0+v\mathbf{V} \tag{B.14}$$

Substituting for **C** and **A** in (B.12):

$$(\mathbf{P}_0-\mathbf{C}_i+v\mathbf{V}-u\mathbf{D})\cdot\mathbf{A}_i=0$$

or

$$(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{A}_i+v\mathbf{V}\cdot\mathbf{A}_i-u\mathbf{D}\cdot\mathbf{A}_i=0$$

$$\Rightarrow v=u\frac{\mathbf{D}\cdot\mathbf{A}_i}{\mathbf{V}\cdot\mathbf{A}_i}-\frac{(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{A}_i}{\mathbf{V}\cdot\mathbf{A}_i} \tag{B.15}$$

Similarly from Eq. (B.13) we obtain:

$$(\mathbf{p}-\mathbf{C})\cdot(\mathbf{p}-\mathbf{C})=R^2$$

i.e.

$$(\mathbf{P}_0-\mathbf{C}_i+v\mathbf{V}-u\mathbf{D})\cdot(\mathbf{P}_0-v\mathbf{V}-u\mathbf{D})=R^2$$

$$\Rightarrow(\mathbf{P}_0-\mathbf{C}_i)\cdot(\mathbf{P}_0-\mathbf{C}_i)+v^2+u^2+2v(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{V}$$

$$-2uv\mathbf{V}\cdot\mathbf{D}-2u(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{D}=R^2. \tag{B.16}$$

Applying the following substitutions:

$$a=|\mathbf{P}_0-\mathbf{C}_i| \qquad e=\mathbf{V}\cdot\mathbf{A}_i$$
$$b=(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{A}_i \qquad f=\mathbf{V}\cdot\mathbf{D}$$
$$c=(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{D} \qquad g=\mathbf{D}\cdot\mathbf{A}_i$$
$$d=(\mathbf{P}_0-\mathbf{C}_i)\cdot\mathbf{V}$$

(B.15) and (B.16) can be rewritten as:

$$v=g/e\,u-b/e \tag{B.17}$$

$$a^2+v^2+u^2+2dv-2fuv-2cu=R^2 \tag{B.18}$$

Substituting for v in (B.18) and rearranging terms we get:

$$h_1u^2-2h_2u+h_3=0$$

where

$$h_1=e^2+g^2-2efg$$
$$h_2=ce^2+bg-deg-bef$$
$$h_3=a^2e^2-R^2e^2+b^2-2bde$$

Solving for u:

$$u=\frac{h_2\pm\sqrt{(h_2^2-h_1h_3)}}{h_1} \tag{B.19}$$

We can determine v by substituting the magnitude of u in (B.17).

Intersection with a cylinder

The only remaining tool-path envelope surfaces are the cylindrical surfaces at the beginning and end of the tool path. Following is a discussion of the analysis of intersections with a cylindrical surface. The cylinder bottom is centered at **C** and the axis is oriented along the unit vector **A**. Substitute [$\mathbf{C}_i$, $\mathbf{A}_i$] for [**C**, **A**] to intersect with the initial tool position and [$\mathbf{C}_f$, $\mathbf{A}_f$] for the final position.

The cylindrical surface of the tool can be generated by rotating a line (which is at a distance R from **C**, and parallel to **A**) through 360° about **A**. A point **p** on this line is given by:

$$\mathbf{p}=\mathbf{E}+t\mathbf{A} \tag{B.20}$$

where **E** is a point on the bottom edge of the cylinder. **E** is defined by the following set of equations:

$$|\mathbf{E}-\mathbf{C}|=R \tag{B.21}$$

$$(\mathbf{E}-\mathbf{C})\cdot\mathbf{A}=0 \tag{B.22}$$

For a valid intersection, $t\in[0, L]$. R is the radius of the cylinder and L its length. If there is an inter-

section, the point **p** is also on the vector **V**:

$$p = \mathbf{P}_0 + v\mathbf{V} \tag{B.23}$$

From (B.20) and (B.23) we get:

$$E = \mathbf{P}_0 + v\mathbf{V} - t\mathbf{A} \tag{B.24}$$

Substituting for E in (B.22):

$$(\mathbf{P}_0 - \mathbf{C} + v\mathbf{V} - t\mathbf{A}) \cdot \mathbf{A} = 0$$
$$\Rightarrow t = (\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{A} + v\mathbf{V} \cdot \mathbf{A} \tag{B.25}$$

Similarly from (B.21):

$$(\mathbf{P}_0 - \mathbf{C}) \cdot (\mathbf{P}_0 - \mathbf{C}) + v^2 + t^2 + 2v(\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{V} - 2vt\mathbf{V} \cdot \mathbf{A} - 2t(\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{A} = R^2 \tag{B.26}$$

Making the following substitutions:

$$a = |\mathbf{P}_0 - \mathbf{C}|$$
$$b = (\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{A}$$
$$c = (\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{V}$$
$$d = \mathbf{V} \cdot \mathbf{A}$$

we get:

$$t = b + vd \tag{B.27}$$
$$a^2 + v^2 + t^2 + 2cv - 2dvt - 2bt = R^2 \tag{B.28}$$

Substituting for t in (B.27):

$$(1-d^2)v^2 - 2(bd-c)v + (a^2 - b^2 - R^2) = 0 \Rightarrow v = \frac{(bd-c) \pm \sqrt{[(bd-c)^2 - (1-d^2)(a^2-b^2-R^2)]}}{(1-d^2)} \tag{B.29}$$

For either v determine t; make sure $0 \leq t \leq L$; the lower (in magnitude) value of v gives the intersection closer to $\mathbf{P}_0$.

If the two values of t happen to lie on either side of zero such that the value corresponding to lower v is negative, then the vector **V** also intersects the bottom face of the cylinder. We need to determine this intersection. The next section deals with such a case.

If $(1-d^2)=0$, it implies that $d=+1$, that is, **V** is parallel to **A**. The intersection can only be with the bottom face of the cylinder. The distance from $\mathbf{P}_0$ of the point of intersection on the plane of bottom face is given by:

$$v = -b/d$$

This is a valid intersection only if this point occurs within the circular patch of radius R around **C**. i.e.

$$|\mathbf{p} - \mathbf{C}| \leq R^2$$

or

$$a^2 + v^2 + 2cv \leq R^2$$

Intersections with a circular patch

Consider a plane circular patch of radius R centered at **C**. **M** is a unit vector perpendicular to the plane of the patch. Every point in this circular patch is defined by:

$$|\mathbf{p} - \mathbf{C}| \leq R^2 \tag{B.30}$$

and

$$(\mathbf{p} - \mathbf{C}) \cdot \mathbf{M} = 0 \tag{B.31}$$

If there is an intersection with **V**:

$$\mathbf{p} = \mathbf{P}_0 + v\mathbf{V} \tag{B.32}$$

On solving for v, we obtain:

$$v = -\frac{(\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{M}}{\mathbf{V} \cdot \mathbf{M}} \tag{B.33}$$

We have a special situation for $\mathbf{V} \cdot \mathbf{M} = 0$, i.e. **V** is parallel to the plane containing the circular patch. Intersections exist only if the point $\mathbf{P}_0$ is in the plane:

$$(\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{M} = 0 \tag{B.34}$$

Also the intersections will only be at the boundary of the patch, which is a circle. So:

$$|\mathbf{p} - \mathbf{C}| = R \tag{B.35}$$

where **p** is a point of intersection.

Solving (B.32), (B.34), and (B.35) simultaneously for v, we obtain:

$$v = -b \pm \sqrt{(b^2 - a^2 + R^2)} \tag{B.36}$$

where

$a = |\mathbf{P}_0 - \mathbf{C}|$ and

$b = (\mathbf{P}_0 - \mathbf{C}) \cdot \mathbf{V}$

Tool path extents

In order to find the buckets which fall under the shadow of the tool path it is necessary to find the extents of the tool path envelope.

For a given tool path $[\mathbf{C}_i, \mathbf{A}_i] \rightarrow [\mathbf{C}_f, \mathbf{A}_f]$, calculate the three unit vectors **D**, **N**, and **B**.
D is a unit vector from $\mathbf{C}_i$ to $\mathbf{C}_f$:

$$\mathbf{D} = \| \mathbf{C}_f - \mathbf{C}_i \|$$

N is perpendicular to the plane containing **D**, $\mathbf{A}_i$, and $\mathbf{A}_f$:

$$\mathbf{N} = \| \mathbf{D} x \mathbf{A}_i \|$$

B is perpendicular to $\mathbf{A}_i$ and **N**.

$$\mathbf{B} = \mathbf{A}_i x \mathbf{N}$$

Now consider $\mathbf{C}_i$, the center of tool bottom at the initial position. Extents of the circular patch at $\mathbf{C}_i$ can be stated in terms of the four points:

$$\mathbf{C}_i - \mathbf{RB} - \mathbf{RN}$$
$$\mathbf{C}_i - \mathbf{RB} + \mathbf{RN}$$
$$\mathbf{C}_i + \mathbf{RB} - \mathbf{RN}$$

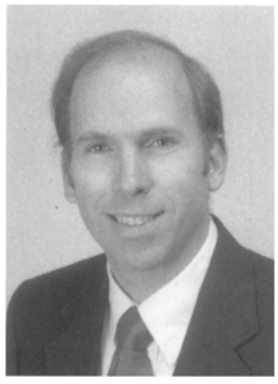
and

$$\mathbf{C}_i + \mathbf{RB} + \mathbf{RN}$$

In a similar way we can find four points defining the extents of the circular face of tool at the top. Center of the top in the initial position can be given as:

$$\mathbf{C}_i, \text{top} = \mathbf{C}_i + L\mathbf{A}_i$$

where L is the length of the tool.
Determine the corresponding extent points for the final position of the tool. Look at the x, y, z coordinates of these 16 points to determine the x min, x max, y min, and y max for the entire tool path.



ROBERT B. JERARD is an associate professor of mechanical engineering at the University of New Hampshire. His research interests are in simulation and automatic generation of numerically controlled machining programs. He has also held faculty positions at Dartmouth College, the University of Connecticut, and Boston University's overseas program in Ramstein, West Germany.

Jerard has a BS from the University of Vermont, SM from MIT, and PhD from the University of Utah. He is a member of IEEE Computer Society, ASME, SME, and ASEE.

ROBERT L. (SCOT) DRYSDALE, III is an associate professor of computer science and mathematics at Dartmouth College, where he has taught since 1978. His primary research area has been algorithms, with special emphasis on computational geometry.

Drysdale received a BA in mathematics from Knox College in 1973 and MS and PhD degrees in computer science from Stanford in 1975 and 1979. He is a member of IEEE Computer Society, ACM and SIGACT, SIGGRAPH, and SIGCSE.

BARRY SCHAUDT is a graduate student in computer science at Dartmouth College. His research interests include the design and analysis of algorithms, data structures, and computational geometry.

Schaudt received an MS and BS in mathematics from the University of Michigan. He is a member of ACM.

SYEO (ZAFAR) HUSSAINI is a graduate student in mechanical engineering at the University of New Hampshire. His research interests are in computer aided design and manufacturing, mechanical design and computer graphics.

Hussaini received his BS in mechanical engineering from the Indian Institute of Technology in Madras and a masters from Dartmouth College. He is a member of ASME.