Tool Path Optimization for Five-Axis Machining

Mud-Armeen Munlin

Faculty of Information Science and Technology, Mahanakorn University of Technology
140 M.1 Cheum-Sampan Rd., Nong Chok, Bangkok 10530 Thailand
Tel: 662-988-3655 ext 4112 Fax: 662-988-4027, mmunlin@gmail.com

ABSTRACT – This paper presents four algorithms to optimize a tool path of a five-axis milling machine. Algorithm 1 is based on the inverse kinematics of the machine and performing continuous rotation of the rotary table. Algorithm 2 further extends the inverse kinematics and performs optimal sequencing with regard to a set of feasible rotations by using the shortest path algorithm. Algorithm 3 employs uniform angular grid to insert addition points in a machine coordinate. Algorithm 4 combines and iterates algorithm 2 and 3 by injecting the points into large loops by equi distributing them with regard to the rotation angle having the largest variation. These algorithms are most efficient in the case of the rough cut characterized by large angle variations which produce considerable errors. The efficiency of the algorithm has been verified by a virtual machining as well as by real cutting on five-axis machine MAHO600E at the CIM Lab of Asian Institute of Technology of Thailand.

KEY WORDS – inverse kinematics, five-axis machining, tool path optimization, shortest path.

1. Introduction

The errors introduced during five-axis machining are common. Several physical phenomena, such as machine kinematics, thermal effects, static and dynamic loading and common-cause failures are errors that often affect the quality of the surfaces produced by five-axis machining. There are a number of error components such as thermal errors, dynamic errors, spindle errors, reference point errors, tool path generation errors, and inverse kinematics errors [1,2,3]. However, the particular effect of machine kinematics and geometric errors seems to be the most significant [4]. These errors, however, may be detected and minimized in advance before the real workpiece is being machined.

The ultimate goal of five-axis tool path optimization is simple: minimize the difference between the desired and the actual surface while producing the actual surface for a minimum time. However, mathematical formulations presented in the literature vary in terms of the error criteria and the set of optimized variables. The tool path is optimized with regard to the machining time, accuracy, the length of the tool path, the width of the machining strip, the volume of the removed material, the size of the remaining scallops, etc. [5,6]. Furthermore, the error analysis and optimization in the areas of large variations of the rotation angles have not been provided by commercial CAD/CAM systems such as Unigraphics, EdgeCam, Vericut, etc. Besides, only a few research papers deal with the subject. In [7] the authors analyze the sequence of rotations to minimize the number of the phase reverse steps at discontinuities of the first derivative of the surface (corners etc). However, kinematics errors in the case of the stationary points have not been analyzed. A method of avoiding singularities has been presented in [8]. However, it is not hard to give a counter-example of a surface on which such avoidance cannot be performed.

This paper presents four algorithms to optimize a tool path of a five-axis milling machine. The algorithms compute and estimate the inverse kinematics errors as well as construct a G-Code from a given tool path and simulate the milling operations of a five-axis milling machine. The G-Code is constructed based on the inverse kinematics of the machine. In particular, the system allows for an efficient simulation of inverse kinematics of multi–axis-milling machines. Moreover, it makes it possible to build a virtual milling environment that enables the user to interactively evaluate the kinematics of the mechanism and estimate the kinematics errors. The first algorithm reduces the errors by re-arrange the rotations to produce the continuous variation of the angles. The remaining errors can be further optimized by the second algorithm to reduce the errors near the stationary points. It is based on global

optimization with regard to feasible solutions of the inverse kinematics. The algorithm is based on the iterative shortest path scheme. The shortest path procedure can be applied to either the entire set of trajectories or to only the most inappropriate trajectories (the so-called loops) inside the work piece. The third algorithm is based on the equi distribution with regard to the rotation angles. The kinematics error near the stationary points depends on the variation of the rotation angles. The method allows inserting additional tool positions by finding numerically a grid of points equi distributed in the angular space. The proposed method requires 10-15 times less additional points than conventional schemes performed near the stationary points. Finally, the fourth algorithm combines and iterates the second and the third algorithms by re-injecting the points into large loops by equi distributing them with regard to the rotation angle having the largest variation. The efficiency of the algorithms has been verified by a five-axis machine MAHO600E at the CIM Lab of Asian Institute of Technology of Thailand.

2. Inverse Kinematics

Consider a typical configuration of the five-axis milling machine with the rotary axis on the table (Fig.1). The machine is guided by axial commands carrying the 3 spatial coordinates (*X, Y, Z*) of the tool-tip in the machine coordinate system *M* and the two rotation angles (*A, B*). The supporting CAM software generates a successive set of coordinates (*X, Y, Z, I, J, K,* called the cutter location points or CL-points) in the workpiece coordinate system *W*. Typically, the CAM software distributes the CL-points along a set of curves which constitutes the so-called zigzag or spiral pattern. An appropriate transformation into the *M*-system generates a set of the machine axial commands which provides the reference inputs for the servo-controllers of the milling robot. However the current CAD/CAM commercial software is not capable of optimizing the removal of the material. The tool path between two consecutive points on a five-axis machine is not a straight line. The real cutter contact (CC) point path is a space curve which needs to be compared with the reference surface. Only this operation produces a real geometric error.

Furthermore, the reference surface must be presented in the IGES format compatible with professional CAD/CAM systems. The tool geometry is a cylinder (flat end mill). Since we employ a parametric model of the surface, the corresponding equations produce the point coordinates and the normal vector. The CL point is positioned at the centerline of the tool along the surface normal. Such a representation must be changed to a CC point positioned in the tool tip plane and on the circle produced by intersection of the tool cylinder and the tool tip plane. This is shown in Fig. 2. However the input to the five-axis CNC machine is the sequence of CL-points transformed to the machine coordinate system by means of the inverse kinematics. The algorithm to compute this location must be integrated with tool path optimization software and with a solid model of the material removal.

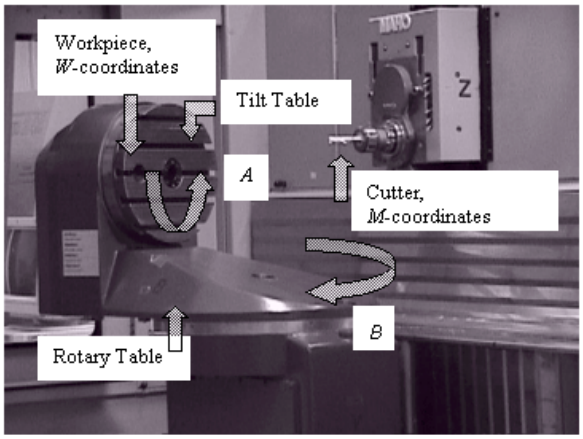

Fig.1: Five-axis milling machine MAHO600E

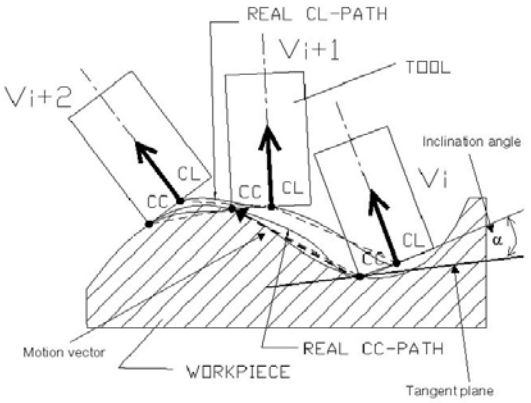

Fig.2: Tool Workpiece Contact Zone

Consider how the axial command translates the centers of rotation and simultaneously rotates the *W*-coordinates (Fig.3). Let $W_p \equiv (x_p, y_p, z_p)$ and $W_{p+1} \equiv (x_{p+1}, y_{p+1}, z_{p+1})$ be two successive spatial positions of the tool path and (a_p, b_p), (a_{p+1}, b_{p+1}) the corresponding rotation angles given by $a_p = \arctan(j_p / i_p)$, $b_p = -\arcsin(k_p)$, where (i_p, j_p, k_p) denotes the normal to the required surface at W_p. In order to calculate the tool trajectory between W_p and W_{p+1}, we first, invoke the inverse kinematics [7] to transform the part-surface coordinates into the machine coordinates $M_p \equiv (X_p, Y_p, Z_p)$ and $M_{p+1} \equiv (X_{p+1}, Y_{p+1}, Z_{p+1})$. Second, the rotation angles $a(t), b(t)$ and the machine coordinates $M \equiv M(t) \equiv$

$(X(t),Y(t),Z(t))$ are assumed to change linearly between the prescribed points, namely,

$$M(t) = tM_{p+1} + (1-t)M_p,$$
$$a(t) = ta_{p+1} + (1-t)a_p,$$
$$b(t) = tb_{p+1}t + (1-t)b_p,$$

(1)

where t is the fictitious time coordinate ($0 \le t \le 1$). Finally, invoking the transformation from M back to W yields $W(t) \equiv (x(t),y(t),z(t))$.

The kinematics are represented by the functions $A \equiv A(a(t)), B \equiv B(b(t))$ associated with the rotations around the primary (the rotary table) and the secondary (tilt table) axes respectively. They are specified by the structure of the machine. For the 5-axis machine in Fig 1, the kinematics involving two rotations and three translations are given by

$$M(t)=B(t)(A(t)(W(t)+R)+T)+C, \quad \textbf{(2)}$$

where R, T, and C are respectively the coordinates of the origin of the workpiece in the rotary table coordinates, coordinates of the origin of the rotary table coordinates in the tilt table coordinates and the origin of the tilt table coordinates in the cutter center coordinates. The general inverse kinematics are given by

$$W= (A^{-1}(B^{-1}(M-C)-T))-R. \quad \textbf{(3)}$$

The rotation angles are

$$A = \tan^{-1}(j/i),\ 0 \le A \le 2\pi; \quad \textbf{(4)}$$

$$B = -\sin^{-1}(k),\ 0 \le B \le \pi/2. \quad \textbf{(5)}$$

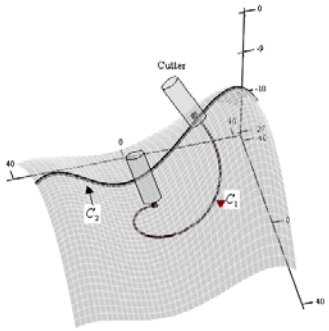

Fig.3: Non-linearity of the tool path in the workpiece coordinates

3. Surface Stationary Points

The kinematics of the machine depend on matrix-functions $A(a)$, $B(b)$ associated with the rotations a and b around the primary (the rotary table) and the secondary (the tilt table) axes shown in Fig.1. A simple analysis of the inverse kinematics reveals that a linear trajectory of the tool tip in the machine coordinates may produce a non-linear trajectory in the work piece coordinates (Fig.3). This effect is amplified when approaching a stationary point of the part surface since it involves sharp variations of the rotation angles. Note that a fine cut of a smooth surface employing small spatial and angular steps may not demonstrate the detrimental effects near the singularity points. However, a rough cut characterized by large gradients could produce considerable errors. The sharp angular jumps produce loop-like trajectories of the tool. Moving along such trajectories may destroy the work piece and even lead to a collision with the machine parts. For simplicity, assume that the tool is aligned along the surface normal. Then the rotation angles are given by (see also Fig.4):

$$a_{base} = \begin{cases} \arctan\left(\frac{I_y}{I_x}\right) & \text{if } I_x > 0 \text{ and } I_y > 0, \\ \arctan\left(\frac{I_y}{I_x}\right) + \pi & \text{if } I_x < 0, \\ \arctan\left(\frac{I_y}{I_x}\right) + 2\pi & \text{otherwise,} \end{cases} \quad \textbf{(6)}$$

$$b_{base} = -\arcsin I_z,$$

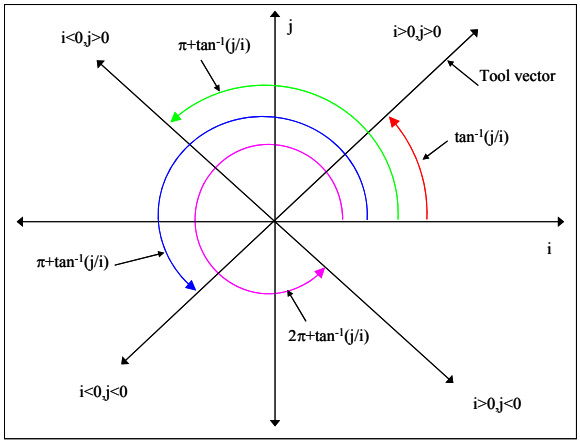

Fig.4: Computation of a_{base} in each quadrant

where $I = (I_x, I_y, I_z)$ is the tool orientation vector. It is not hard to demonstrate that the inverse kinematics admits four solutions given by:

$$\Lambda = \begin{cases} a_{base}, b_{base} \\ a_{base} - 2\pi, b_{base} \\ a_{base} - \pi, -b_{base} - \pi \\ a_{base} + \pi, -b_{base} - \pi. \end{cases} \quad \textbf{(7)}$$

Solutions similar to (7) can be established for other types of the machine kinematics such as, "one axis on the tool and one on the table" and "both axis on the tool". The shortest path from the feasible sequences must be identified in such a way that the kinematics error is minimized. In other words, the following problem must be solved

$$\underset{\Lambda}{\text{minimize}}\,(\varepsilon), \quad \textbf{(8)}$$

where

$\varepsilon = \sum_{p=1}^{N-1} \varepsilon_{p,p+1}$ is the total kinematics error and where

$$\varepsilon_{p,p+1} \equiv [\int_0^1 (W_{p,p+1}^D - W_{p,p+1})^2 \, dt]^{1/2} \quad \textbf{(9)}$$

is the kinematics error between cutter location points p and p+1 and $W_{p,p+1}^D(t), W_{p,p+1}(t)$ the desired and the actual tool trajectory. Note that in many cases problem (8) can be replaced by minimization of the total error variation as follows

$$\underset{\Lambda}{\text{minimize}}\,(w), \quad \textbf{(10)}$$

where

$$w = \frac{1}{N-1} \sum_{p=1}^{N-1} (a_p - a_{p+1})^2 + (b_p - b_{p+1})^2 \,,$$

and N is the total number of the cutter location points.

4. Tool Path Optimization Algorithms

The CAD software generates the CL points in the workpiece coordinate system whereas the CAM software generates the tool path or G-Code in the machine coordinate system from the prescribed CL points. The G-Code guides the cutting tool of the 5-axis machine to travel along a nonlinear trajectory to reach the required CL point. The nonlinear trajectories constitute a trajectory surface, which is slightly different from the actual surface. The error surface is estimated by computing the difference between the actual and trajectory surfaces.

Let four points *W1(i,j), W2(i+1,j), W3(i+1,j+1)* and *W4(i,j+1)* in Fig.4 be the grid $\{(u,v)_{i,j}\}$ which represents the actual surface *S(u,v)*. First, we apply the inverse kinematics (see section 2) to transform each point into the corresponding machine coordinate *M1, M2, M3,* and *M4* respectively. Then, we perform the linear interpolation procedure on the machine coordinate *M* and invoke the inverse transformation from *M* back to *W* (for every *t*) yields the tool path $W(t) \equiv (x(t), y(t), z(t))$. Next, the calculated tool path is mapped on the grid $\{(u,v)_{i,j}\}$. Finally, using the bilinear interpolation procedure on the grid $\{(u,v)_{i,j}\}$ yields an approximation of the machined surface *T(u,v).* The error surface is then calculated by

$$\omega_\tau(u,v) = |\, S(u,v) - T(u,v) \,|, \quad \textbf{(11)}$$

where $S(u,v) = (x(u,v), y(u,v), z(u,v))$ is the actual surface and $T(u,v) = (x_T(u,v), y_T(u,v), z_T(u,v))$, is the trajectory surface as shown in Fig.5. These errors are estimated and visualized graphically by the virtual machine. We propose four algorithms, namely 1) angle adjustment, 2) angle switching, 3) angle insertion and 4) iterative angle switching to optimize the tool path by minimizing such errors.

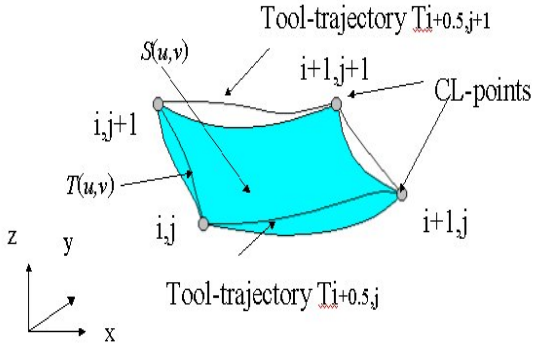

Fig.5: Error surface computation

4.1 Angle Adjustment

If the *a*-angle jumps unexpectedly from minimum to maximum e.g. from 5 degree to 355 degree or vice versa, this will produce a large angle variation and large error and may cause an expected collision among each machine's axis. Therefore, the angle adjustment algorithm is introduced to produce the

continuous variation of the angles. The sequence $\{a_p\}$ should be adjusted in order to minimize the difference between the successive angles. For instance if *a-angle* changes from 355 to 5 the algorithm must replace 5 by 365, etc. The following C++ program illustrates the angle adjustment algorithm.

```
for (i=0;i<N;i++)
{
   // get vertex
   a1 = a(i); a2 = a(i+1); ak = a2;
   // compare a(i) and a(i+1)
   if (a2 - a1  > PI )
   {
        // adjust the angle
        for(k=i+1; k<N; k++)
            ak = ak - 2*PI;
   }
   else if (a2 - a1 < -PI)
     {
         // adjust the angle
         for (k=i+1; k<N; k++)
             ak = ak + 2*PI;
     }
} // end i
```

4.2 Angle Switching

According to our initial setup of the 5-axis machine, the *a-angle* is given by a = tan $^{-1}$ (*j*/*i*), 0 ≤ a ≤ 2π. However, the normal vector *i and j* can be in any of the four quadrants as illustrate in equation **(6)**. Furthermore, there are four sets of *a-angle* and *b-angles* within the range $[0,2\pi]$ that can rotate the tool vector into the required orientation. The set of the *a-angles* is defined by $\{a_{base}, a_{base}-2\pi, a_{base}-\pi, a_{base}+\pi\}$. Note that, after the rotation by a_{base} or $a_{base}-2\pi$ the tool is positioned in the same quadrant with the original orientation whereas the rotation by $a_{base}-\pi, a_{base}+\pi$ corresponds to the tool being in opposite direction. The set of the feasible *b-angle* (b_{base},,$-\pi-b_{base}$) is required to transform the tool vector into the require orientation. It is not hard to demonstrate that a_{base} or $a_{base}-2\pi$ requires the rotation $b = b_{base} = -\sin^{-1}(k)$ whereas the rotation $a_{base}-\pi, a_{base}+\pi$ corresponds to another solution $b = -\pi - b_{base}$. Therefore, we have four paths represented by four pairs of feasible *(a, b)* to transform the tool vector p_i into the required location p_{i+1}. That is,

$$p_{i+1} = \begin{cases} x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1}, b_{i+1} \\ x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1}-2\pi, b_{i+1} \\ x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1}-\pi, -b_{i+1}-\pi \\ x_{i+1}, y_{i+1}, z_{i+1}, a_{i+1}+\pi, -b_{i+1}-\pi \end{cases} \quad \textbf{(12)}$$

We can determine the path by examining the minimum errors at each point along the tool path. According to our experiment, every line on the tool path near or cross a stationary point such as minimum or maximum or saddle generates the loop trajectories. These loops represent the larger errors than any other points due to the longer distance the tool has to travel to cross the stationary point. Therefore, we may perform the optimization with regards to this set. We will minimize the distance traveled by the tool in the space *(a, b)* with the Euclidean distance:

$$\text{distance}(p+1, p) = \frac{(a_{p+1}-a_p)^2 + (b_{p+1}-b_p)^2}{(x_{p+1}-x_p)^2 + (y_{p+1}-y_p)^2 + (z_{p+1}-z_p)^2} \quad \textbf{(13)}$$

The total distance traveled by the tool within the vicinity of the large errors is given by

$$\sum_{p=1}^{n} \frac{(a_{p+1}-a_p)^2 + (b_{p+1}-b_p)^2}{(x_{p+1}-x_p)^2 + (y_{p+1}-y_p)^2 + (z_{p+1}-z_p)^2} \quad \textbf{(14)}$$

Note that, we consider only the distant between *(a,b)* because the position *(x,y,z)* of p_i and p_{i+1} does not change. However, we have four set of feasible *(a,b)* between p_i and p_{i+1} and only one set of *(a,b)* with the smallest distant is selected to optimize the errors. The actual errors, however, are computed using equation **(11)** which takes into account both *(x,y,z)* and *(a,b).*

When the tool enters the zone of large milling errors, we apply the angle-switching algorithm to select the shortest distance for the tool to travel from the current point to the next. Note that a procedure to compute the rotation angles may become ill-conditioned in the regions $|i + j| < \varepsilon$ (ε is a small number). For instance for "flat" surfaces or near the points of maximum or minimum. As a matter of fact, small variations of the normal vector in the above

mentioned regions produce sharp variations of the rotation angles and numerical instabilities. However, the algorithm from section 4.1 only eliminates unwanted variations of more than 180 degree. We still face the problem of large variation of the angle between 0 to 180 degree producing the loops due to the non-linearity of the inverse kinematics of the machine. This phenomenon may cause an unexpected motion of the machine and even collisions with the tilt table. We, therefore, introduce an "Angle Switching" algorithm to select the shortest path from the feasible sequences $\{a_{i+1}\}$ in such a way that $\Sigma|a_{p+1}- a_p|+\Sigma|b_{p+1}- b_p|$ is minimized using possible equivalent angles. For instance, a position $\{a_p,b_p\}$ could be followed by either $\{a_{i+1},b_{i+1}\}$, calculated by means of the equation **(12)**.

The loop detection and elimination method is based on the following two ideas. 1) Since the loops usually occur near a minimum or a maximum of the concave or convex surface (stationary points), the angle switching algorithm must be applied only in the vicinity of a stationary points. 2) In order to obtain the optimal path, the graph–based shortest path algorithm must be employed. The angle switching algorithm for optimizing the tool path by computing the smallest distance within the vicinity of the stationary points is described by the following steps:

1. Locate the vicinity of the large milling errors to determine the source (*s*) and the destination (*t*) vertices. In this work, we demonstrate the tool path optimization based on the distant optimization using the saddle surface that contains both convex (maximum) and concave (minimum) areas. We use our virtual five-axis simulator [9] to graphically visualize the errors near the maximum and minimum of the saddle surface and determine the source and the destination vertices as well as the area of vicinity for optimization.

2. Construct the graph to represent four feasible paths from *s* to *t* using the adjacency list. The graph nodes represent the vertices and the arcs represent the distance between two adjacent nodes of all four paths as illustrated in Fig.6. We do not create the error graph because error computation is much more expensive than the distance computation. However, the average error from *s* to *t* is proportional to the average distance from *s* to *t*, such that distant optimization gives the result proportional to error optimization.

3. Apply the Dijkstra's shortest path algorithm [10] to compute the smallest distant from *s* to *t*, from the graph in Fig.6.

4. Update all vertices from *s* to *t* using the output shortest path from the previous step.

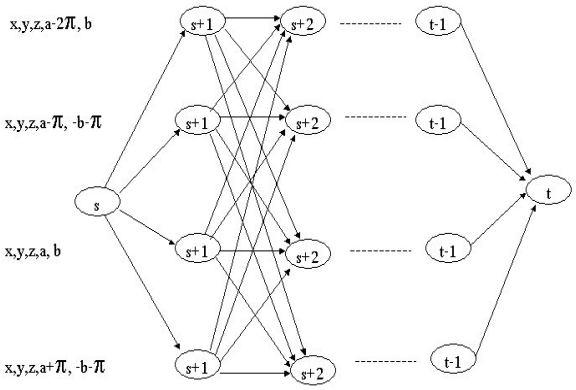

Fig.6: The graph represents four paths from s to t

4.3 Angle Insertion

The angle switching algorithm still leaves errors which can be further eliminated only by inserting additional CC points. In particular, when the tool path crosses the stationary point in the direction parallel (or nearly so) to the secondary rotary axis. The work piece often must be rotated by an angle close to π. Therefore, further optimization is required to reduce such angular jumps. If the trajectory lies far from singularities a conventional approach to insert spatially uniform grid of the CC points between the two given positions solves the problem. However, such interpolation does not remove the jumps near the singularities. In order to solve this problem, a special interpolation called the "Angle Insertion" is proposed to partition the large angular interval into a grid having equal angular increments. Unfortunately, we can not merely split the angular interval in the machine coordinates. The resulting angles do not correspond to the actual CC points and orientations. In other words the trajectory does not follow the surface. The correct set of the CC points and the tool orientations can only be calculated from the surface equation. Therefore, we propose the following angle insertion procedure.

1. Construct a uniform grid $a_i = a' + i\Delta a$ (where Δa is the angular step) near the singularity.

2. For each a_i find the CC point and the orientation that produce a_i by the bisection method as follows:

2.1 Compute a midpoint (x_m, y_m).

2.2 Invert the parametric equations $S_x(u,v) = x_m$, $S_y(u,v) = y_m$ to find the corresponding u_m, v_m and consequently z_m.

2.3 Calculate the orientation vector and obtain the corresponding a_m.

2.4 Compare a_m with the target a_i

The bisection procedure above runs until it converges within the prescribed accuracy.

4.4 Iterative Angle Switching

When it is necessary to insert additional points, there are many methods for such insertions. However, from the viewpoint of the kinematics error one of the most efficient methods near stationary points is angular insertion proposed in section 4.3. The method "injects" the points into large loops by equi distributing them with regard to the rotation angle having the largest variation. This complies with the idea of functional that the angle variation affects the kinematics error the most.

Consider the rotation angles (degree) before applying the angle switching algorithm, suppose a_i = 319, b_i = -79, a_{i+1} = 224, b_{i+1} = -80. The shortest path optimization produces a_i = 319, b_i = -79, $a_{i+1,new} = a_{i+1}$ + 180 = 404, $b_{i+1,new} = -b_{i+1} - 180$ = -100. Inserting a point in the middle yields a_{mid} = 267, b_{mid} = -82. Taking into that we have to perform the same modification as with a_i, b_i, we have the new pair of angles given by $a_{mid} = a_{mid} + 180$ = 447, $b_{mid} = - b_{mid}$ - 180 = -98, which is not actually in the middle between a_i = 319 and $a_{i+1,new}$ = 404. In other word, $a_{mid} \notin [a_i, a_{i+1}]$ anymore. This would produce a larger error. Therefore, a single additional point may destroy the integrity of a particular shortest path. The iterative angle switching algorithm is proposed to iteratively switch the angles when the additional points are inserted as follow:

1. Run the angle switching (AS) algorithm.
2. If the kinematics error is within the prescribed tolerance, quit, otherwise, find the trajectory with the kinematics error exceeding the tolerance.
3. Mark this trajectory.
4. Return to the original tool path (ORG) and using the angle insertion (AI) algorithm to insert a point inside the selected trajectory even though in the original path they do not produce large kinematics error.
5. Repeat step 1.

It is clear that the algorithm converges because in the worst case the additional points are inserted into every interval. However, in this case switching of the angles is no longer accomplishable. Therefore, we are interested in solutions when benefits of the angle switching are combined with error reduction produced by inserting some relatively small number of points. In other words, the question whether it is practical that a few points have been inserted to reduce the error and some switching is still present in the path. We also are interested whether the above mentioned angle insertion techniques still provides some benefits as the basic insertion method in this algorithm.

5. Experimental Results

The objective of the cutting experiments is the calibration of the parameters involved in the inverse kinematics. The inverse kinematics transforms the tool reference vector (*x, y, z, i, j, k*) fixed to the workpiece into the machine coordinates *X, Y, Z, A, B* fixed to the machine frame. A parametric saddle surface which contains both convex (maximum) and concave (minimum) regions is used as a case study to demonstrate the tool path optimization algorithms. The experiment constitutes the basic test of how our graphic simulation software would detect such kinematics errors, locate the problem areas, as well as minimize the errors. The parametric saddle surface is given below and illustrate in Fig.7.

$$S(u,v) = \begin{pmatrix} 100u - 50 \\ 100v - 50 \\ -80v(v-1)(3.55u - 14.8u^2 + 21.15u^3 - 9.9u^4) - 28 \end{pmatrix} \quad \textbf{(15)}$$

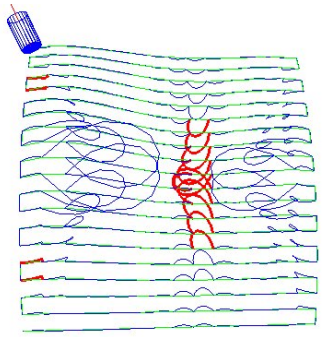

Fig.7: The original trajectories of the saddle surface (undercuts are bold)

5.1 Angle Adjustment

The *a-angle* and *b-angle* are computed using machine inverse kinematics described in section 2. A and B axis are rotating simultaneously ranging from 0 to 360 and from 0 to -90 respectively. Fig.7 shows the required saddle surface. Fig.8 shows the corresponding graphs of the *a-angles* and the angle adjustment. Note that the angle adjustment is required to eliminate sharp variations of the rotation angles near minimum or the maximum of the surface. Note that although the general case requires an adjustment of *B* as well, the case of the saddle surface implies that B $\in$[-90, 0]. Therefore the adjustment is not required.

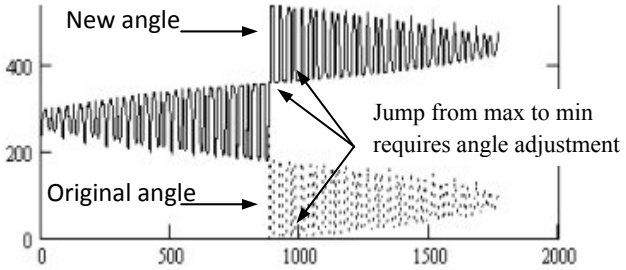

Fig.8: Angle adjustment for the saddle surface

5.2 Angle Switching

The initial setup and configuration of our five-axis machine in Fig.1 imply that the a-angle and b-angle are given by equation 6 and 7. The stationary position such as the minimum or the maximum or the saddle point may generate trajectories characterized by large deviations from the surface or so called "loops". Usually the loops represent the largest errors and appear because the large variations of the rotation angles a and b. Therefore, the optimization can be performed only in the vicinities of these points. The angle switching will select the shortest path from the feasible sequences of the angles in such a way that $\Sigma|a_{p+1} - a_p| + \Sigma|b_{p+1} - b_p|$ is minimized. The algorithm decreases the sharp rotation angle and turns the large loops (Fig.7) into the smaller ones (Fig.9).

It should be noted that the optimization may make sense only for the so-called rough cutting or for cuts characterized by sharp gradients. For instance, Table 1 presents the case of a rough cut when the number of the required cutter location points is not very large. Increasing the number of points along the cutting direction (see Table 1) shows that small angular steps make the optimization superfluous (see the path 20 x 130). When the angular step is small, switching between the feasible trajectories increases the step and therefore amplifies the error.

Note that the algorithm may also turn irreparable "bad loops" or undercut (bold trajectories in fig.7) which enter the surface into "good loops" or overcut which go above the surface (Fig.10). The "good loops" may also lead to considerable errors, however, such errors are reparable whereas the "bad loops" may destroy the workpiece.

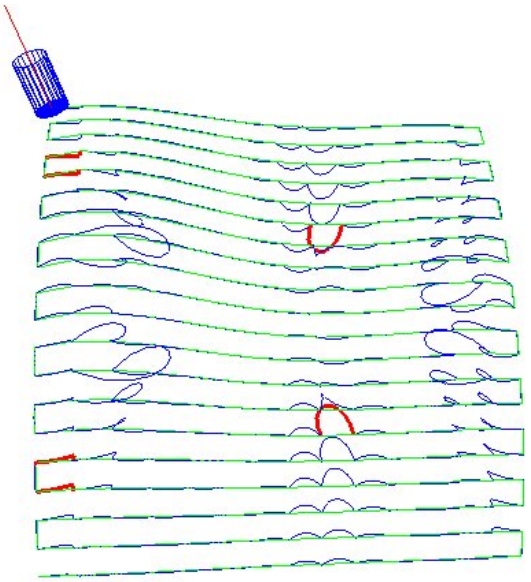

Fig.9: The repaired surface trajectory

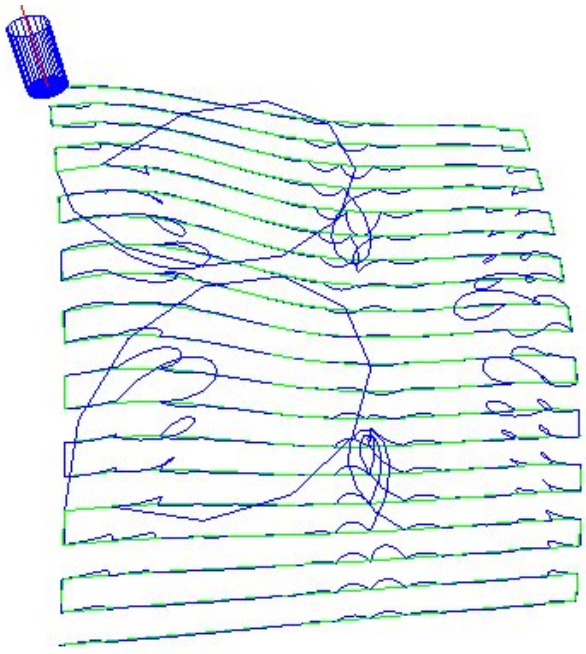

Fig.10: The repaired surface trajectory with regard to undercuts.

Table 1: Kinematics error for the optimized and non-optimized tool path.

Number of CL points	Non-opt error (mm)	Opt error, (mm)	% error decrease
20 x 20	0.867	0.133	84.66
30 x 20	0.336	0.055	83.63
40 x 20	0.085	0.033	61.18
50 x 20	0.077	0.035	54.55
60 x 20	0.071	0.038	46.48
70 x 20	0.066	0.042	36.36
100 x 20	0.054	0.052	3.70
130 x 20	0.047	0.047	0.00

Although this algorithm does not entirely eliminates the need to insert additional CL points it substantially decreases the error. In particular, such optimization constitutes an efficient tool in the case of rough machining in the five-axis mode. The numerical experiments verified by practical machining demonstrate the accuracy increase ranging from 4 to 80 % in the case of rough cutting.

5.3 Angle Insertion

The angle insertion is proposed to partition the large angular interval into a grid having equal angular increments. The stationary point can be located by finding the largest loop in which the tool vector changes the sign of the *i* or the *j* component as shown in Fig.11.

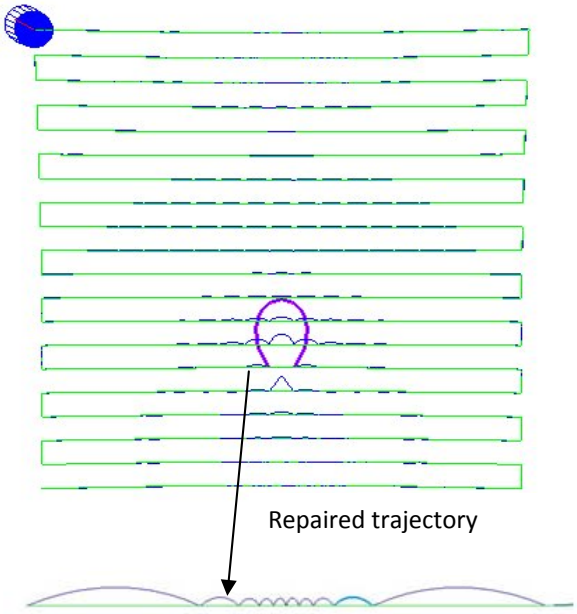

Fig.11: The original trajectory and the repaired trajectory using 7-points angle insertion

Fig.11 and Table 2 illustrate the procedure. The angular interval has been partitioned into the almost equal subintervals. Consequently the loops near the singularity have been significantly reduced. Note that usually only one of the two rotation angles changes sharply near the singularity. Therefore, the bisection should be applied in only one dimension. However, theoretically there exists a possibility when both angles must be bisected. In this case one should use either a two dimensional version of the above procedure or to bisect a linear combination of the angles. The singular point can be located by finding the largest loop in which the tool vector changes the sign of the *i* or the *j* component.

Table 2 displays the kinematics errors and the rotation angles before and after applying the proposed angle insertion. Clearly, the uniform angular grid allows us to substantially decrease the error, whereas the conventional point insertion or spatial grid is not efficient. Clearly, when a certain number of additional points have been inserted, the error decrease is approximately the same as for the space or angular insertion. Therefore, the method is applicable only for the rough cuts characterized by sharp variations of the rotation angles

Table 2: Max error versus number of inserted points

Number of inserted points	Max error (mm) Conventional	Max error (mm) Angular Grid
0	19.300	19.300
8	4.241	0.416
16	1.707	0.138
32	0.490	0.092
64	0.158	0.089
128	0.099	0.089

Table 3 illustrates the most impressive results obtained in the case of a rough cut on 20 by 20 grids. The error has been reduced in more than 20 times after the angle switching and in more than 40 times after the angle insertion. Moreover the angle insertion works almost 10 times better than a standard CC point insertion.

Table 3: Performance of the angle insertion algorithm

Grid 20 by 20	Max Error (mm)	% Error Decrease
Angle Adjustment	6.388	N/A
Angle switching	0.305	95.22
7 CC points insertion	1.209	81.07
7 points Angle insertion	0.150	97.65

5.4 Iterative Angle Switching

In many cases, inserting additional points destroy the integrity of the shortest path sequencing. The following is an example of such a drawback. Consider a saddle surface in Fig.9 with a tool path obtained by the angle switching optimization. The largest loop is on the right side of the surface. Inserting a point in the middle of the loop, produces a larger loop (Fig.12).

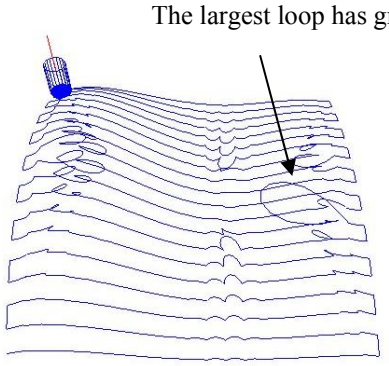

Fig.12: The trajectory surface after applying angle insertion to the surface in Fig.9

Fig.13 shows the saddle surface $S(u,v)$ from equation **(15)** subjected to the above mentioned procedures after the maximum error has been reduced to a certain prescribed value using iterative angle switching optimization methods. It has been proven experimentally that the proposed method requires 47.3 % less additional points than conventional schemes performed near the stationary points.

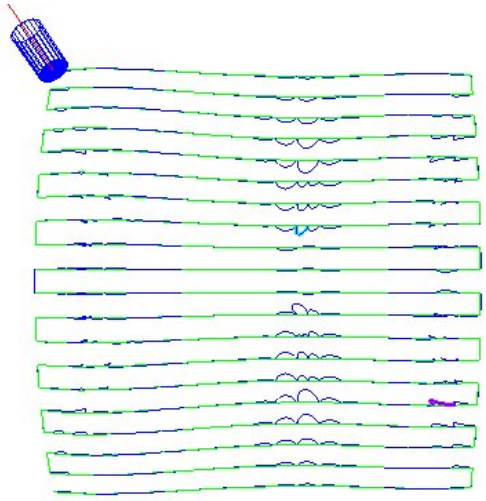

Fig.13: The trajectory surface after applying iterative angle switching to the surface in Fig.9

Table 4 compares the proposed iterative angle switching algorithm using conventional spatial grid (IT/PI) and uniform angular grid (IT/AI) for inserting additional CL points. PI stands for inserting equi-spaced points along the tool trajectory in the physical space (x,y,z), whereas AI stands for points equi distributed in the angular space (a,b) along the angular arc. The maximum allowable error is 2 mm. Clearly the iterative angle switching algorithm combined with the angular insertion requires the smallest number of points. This is because the method presents an appropriate combination of angle switching and inserting, whereas AS and AI use only a single technique.

Table 4: Comparison of the inserted points with several methods on the entire saddle surface

Methods	**AS**	**PI**	**AI**	**AS-IT/PI**	**AS-IT/AI**
#inserted points	0	74	48	56	39
Max error (mm)	7.45	1.96	1.96	1.96	1.96

The preliminary results show a 20% improvement over the pure angular insertion scheme and about 50% improvement with the reference to spatially equi-distributed points.

6. Conclusion

A tool path thought as a trajectory in the five-dimensional space is a subject of optimization on the set of solutions of the inverse kinematics equations with regard to the required rotations. Minimization of the total angle variation for rough cuts leads to a substantial accuracy increase ranging up to 80 %. The optimization can be formulated in terms of the total error as well as in terms of the undercut and overcut error. Optimization of the undercut error ensures against removal of the excess material during the rough cut. Further improvement of the accuracy can be achieved by constructing the uniform grid in the angular space around the CC points characterized by large angle variation. The methods are most efficient in the case of the rough cut characterized by large angle variations which produce considerable errors. A fine cut of a smooth surface employing small spatial and angular steps may not display the detrimental effects near the stationary points. In this case the methods do not lead to a substantial accuracy increase.

Four algorithms to optimize the tool path for five-axis machining have been presented. 1) Angle

Adjustment produces the continuous variation of the angles. The sequence of rotation angles is adjusted in order to minimize the difference between the successive values. 2) Angle Switching is capable of repairing the bad trajectories or minimizing the large milling error by adjusting the rotation angles in such a way that the kinematics error is minimized. The algorithm employs the shortest path scheme to optimize tool path. 3) Angle Insertion is based on the equi distribution of the rotation angles near stationary position. A special interpolation combined with the bisection methods is employed to insert additional points equally in angular space to obtain a uniform minimal kinematics error. 4) Iterative Angle Switching based on the shortest path algorithm combined with the equi distribution of the cutter location points in the angular space to improve the efficiency of five-axis machining.

The efficiency of the algorithm has been verified by a practical machining (Fig.14) using five-axis machine MAHO600E at the CIM Lab of Asian Institute of Technology of Thailand. It has been also verified by the virtual milling machine simulator [9] developed by the author.

Fig.14: The real cut of the saddle surface using the proposed algorithms

References

[1] Srivastava AK, Veldhuis SC, Elbestawit MA, "Modelling geometric and thermal errors in a five-axis CNC machine tool", Int J Mach Tools Manufact, 1995, V35, N9, pp. 1321-1337.
[2] Mu YH, Ngoi KA, "Dynamic error compensation of coordinate measuring machines for high-speed measurement", Int J Adv Manuf Technol, 1999, V15, pp. 810-814.
[3] Aekambaram R, Raman S, "Improved toolpath generation, error measures and analysis for sculptured surface machining", Int J Prod Research, V37, N2, 1999, pp. 413-431.
[4] Y. Koren, "Five-Axis Interpolators", Annals of CIPR, V.44, N1, 1995, pp.379-382.
[5] Taejung Kim and Sanjay E. Sarma, "Tool path generation along directions of maximum kinematics performance; a first cut at machine-optimum paths", Computer Aided Design 34, 2002, 453-468.
[6] Chuang-Jang Chiou and Yuan-Shin Lee, "A machining potential field approach to tool path generation for multi-axis sculpture surface machining", Computer Aided Design 34, 2002, 357-371.
[7] Y.H. Jung, D.W. Lee, J.S. Kim and H.S. Mok, "NC post-processor for 5 axis milling machine of table-rotating/tilting type", Materials Processing Technology, 130-131, 2002, 641-646.
[8] Affouard, E. Duc, C. Lartigue, J.M. Langeron, and P. Bourdet, "Avoiding 5-axis singularities using tool path deformation", International Journal of Machine Tools and Manufacture, 44(4), 2004, 415-425.
[9] M. Munlin, "Tool Path Simulation Using a Virtual 5-Axis Milling Machine", Proc. Of 2002 IEEE International Conference on Industrial Technology, Bangkok, 11-14 December 2002, pp. 193-198.
[10] M. A. Weiss, "Data structures and algorithm analysis in C", Addison Wesley, 1997.