Virtual 5-Axis Milling Machine: Error Estimator

Mud-Armeen Munlin
Department of Information Technology,
Sirindhorn International Institute of Technology,
Tammasat University, Rangsit Campus,
Pathum Thani, 12121, Thailand,
amin@siit.tu.ac.th

Abstract

This paper presents the tool path optimization algorithms to compute and estimate the non-linear inverse kinematics errors of the 5-axis milling machine. The approach is based on a global approximation of the required surface by a virtual surface constructed from the tool trajectories. Errors are computed from the difference between the required surface and the virtual surface and displayed numerically and graphically through the virtual machine simulator. The simulator is based on 3D representation and employs the inverse kinematics approach to derive the corresponding rotational and translation movement of the mechanism. The simulator makes it possible to estimate the errors of a 3D tool-path based on a prescribed set of the cutter location (CL) points as well as a set of the cutter contact (CC) points with tool inclination angle. Errors, particularly near the vicinity of the large milling errors, are minimized using a discrete algorithm based on the shortest path strategy. Furthermore, the simulator can be used to simulate the milling process, verify the final cut of the actual tool-path before testing with the real machine. Thus, it reduces the cost of iterative trial and errors.

The error estimator has been verified by a number of cutting experiments performed by means of the proposed algorithms. It has been shown that the proposed graphical 3D software presents an efficient interactive approach to verification of the tool path optimization algorithms. The result of the simulation has been tested using the Maho600E 5-Axis Milling Machine at the Computer Integrated Manufacturing Laboratory in Asian Institute of Technology.

Keywords: Inverse kinematics, 5-axis CNC machines, Kinematics errors, Tool path simulation and optimization.

1.Introduction

The errors introduced during 5-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 5-axis machining. There are a number of error components such as thermal errors, dynamic errors, spindle errors, reference point errors, toolpath generation errors, and inverse kinematics errors [1,2,3,4,5]. However, the particular effect of machine kinematics and geometric errors seems to be the most significant [6,7,8]. These errors, however, may be detected and minimized in advance before the real workpiece is being machined. In order to compute and estimate such errors, the simulation of 5-axis milling is a must. The software must include tool-path tracing, dynamic display of all moving element combined with realistic solid modeling. This paper introduces software for calculating and estimating the inverse kinematics errors and constructing a G-Code from a given toolpath (or CL-points) and simulating the milling operations of a 5-axis milling machine. The G-Code is constructed based on the inverse kinematics of the machine. The inverse kinematics of the machine combined with the geometric constraints constitutes a basis to develop a solid modeling system for simulation, verification and optimization of the cutting operations. 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 environment that enables the user to interactively evaluate the kinematics of the mechanism and estimate the geometric and kinematics errors.

Consider a typical configuration of the 5-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 (Fig.2). 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.

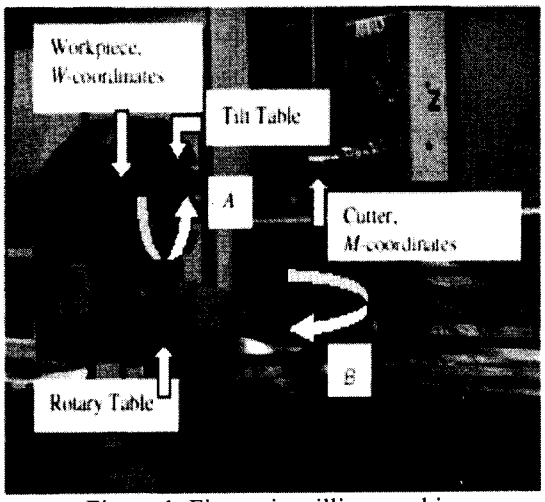
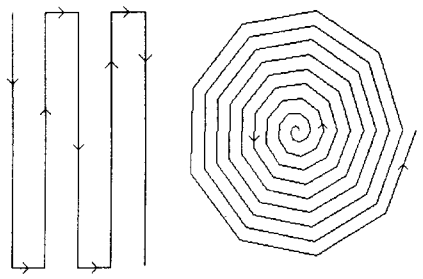


Figure 1. Five-axis milling machine



(a) (b)

Figure 2. Zigzag (a) and spiral (b) tool path

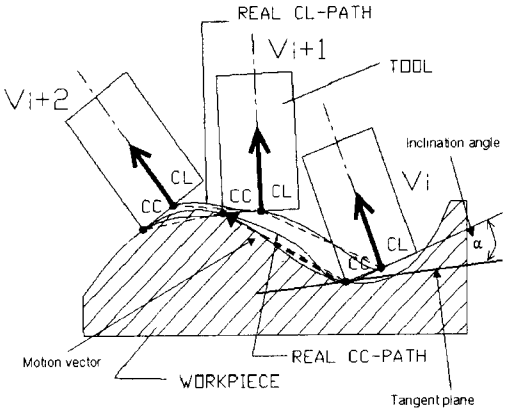
In order to ensure a prescribed tolerance, the standard CAM software "estimates" the local errors and incorporates additional cuts (if applicable) into a single G-Code block. However, such a strategy invokes a substantial increase of the CL-points and consequently a substantial increase of the machining time [7]. Therefore, recent papers have displayed a number of sophisticated methods to optimize a zigzag or spiral pattern combined with techniques dealing with the geometric complexity of the workpiece [8,9,10]. Besides, there exist a variety of off-line methods to generate a suitable non-uniform tool-path, for instance, the neural network modeling approach [11] and the Voronoi diagram technique [10]. Verification of this method requires an up to date software involving appropriate non-linear kinematics as well as a solid model of the milling machine.

2.Non-Linear Inverse Kinematics of a 5-Axis Milling Machine

A 5-axis machine designed for high quality manufacturing offers the highest degree of freedom with regard to the orientation of the workpiece and the cutting tool. However the current CAD/CAM commercial software is not capable of optimizing the removal of the material. We outline the following open problems: (i) Optimization of the tool angle at each cutter contact point (CC points with tool inclination) so that undercut/overcut are avoided and the geometric errors are minimized, (ii) Detection of the errors and collisions including side milling effect. The tool path between two consecutive points on a 5-axis machine is not a straight line. The real 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. 3. However the input to the 5-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.



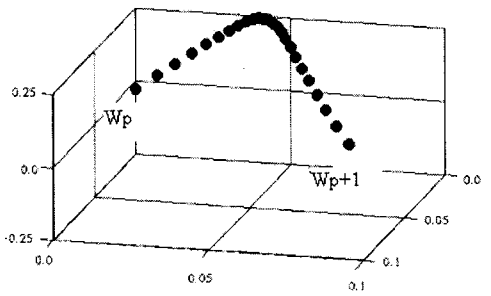


Figure 3. Tool Workpiece Contact Zone
Figure 4. Simulation of non-linear tool path

Consider how the axial command translates the centers of rotation and simultaneously rotates the W-coordinates (Fig.4). 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,

$$\begin{aligned} M(t) &= tM_{p+1} + (1-t)M_p, \\ a(t) &= ta_{p+1} + (1-t)a_p, \\ b(t) &= tb_{p+1} t + (1-t)b_p, \end{aligned} \qquad (1)$$

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

$$M(t) = B(t)(A(t)(W(t) + R) + T) + C, \qquad (2)$$

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

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

In this work, we use the following sequence of transformations to calculate the inverse kinematics of the 5-axis machine.

1. Transform a CL-point P_1 (x, y, z, i, j, k) into the local coordinate system (LCS) of the rotary table by means of the rotation matrix R_1 and the translation vector T_1. The new position is given by $P_2 = R_1(P_1 + T_1)$.
2. Transform P_2 into the LCS of the tilt table by translation vector T_2 and rotation R_2 and R_3. The new position given by is $P_3 = R_3 R_2 (P_2 + T_2)$.
3. Transform P_3 until it is coincident with the LCS of the tool tip by translation T_3. The final position is given by $P_4 = P_3 + T_3$.

The translation vectors are given by $T_1 = [t_{1x}, t_{1y}, t_{1z}]$, $T_2 = [t_{2x}, t_{2y}, t_{2z}]$, $T_3 = [t_{3x}, t_{3y}, t_{3z}]$, where t_{ix}, t_{iy}, t_{iz} are the corresponding offsets which depend on the reference position of the machine (Fig 1).

The rotation matrices are given by

$$R_1 = \begin{pmatrix} \cos(A) & \sin(A) & 0 \\ -\sin(A) & \cos(A) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad (4)$$

$$R_2 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad (5)$$

$$R_3 = \begin{pmatrix} \cos(B) & 0 & -\sin(B) \\ 0 & 1 & 0 \\ \sin(B) & 0 & \cos(B) \end{pmatrix} \quad (6)$$

The rotation angles are

$A = \tan^{-1}(j/i)$, $0 \le A \le 2\pi$; (7)

$B = -\sin^{-1}(k)$, $0 \le B \le \pi/2$. (8)

Furthermore, given the M-coordinates of two consecutive CL-points $P_1=(x_1, y_1, z_1)$ and $P_2=(x_2, y_2, z_2)$ and the two rotation angles representing an orientation of the tool, the virtual machine will perform the required motion by means of the constrained-based algorithm [12]. At this stage, it may seem that the virtual machine performs nothing (does not yet show the movements of each machine's axis) more than the inverse transformation given by $R_1^{-1}(R_2^{-1}R_3^{-1}(P_4 - T_3) - T_2) - T_1$. Nevertheless, the results of the inverse kinematics can be visualized and used to inspect whether or not the virtual machine performs the correct actions. According to our experiment, all surfaces have been machining with satisfactory result in a single cut. Finally, it is worth noting that the results surprised the technician who had been running the machine for many years. He rarely witnessed machining in a single cut by means of other post-processing software.

3. Error Estimator

The CAD software generates the CL points in the workpiece coordinate system whereas the CAM software generates the 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 Figure 5 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 toolpath 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

$$\omega_T(u,v) = |S(u,v) - T(u,v)|, \quad (9)$$

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 Figure 5. These errors are estimated and visualized graphically by the virtual machine. We propose two algorithms, namely "angle adjustment" and "angle switching" to minimize such errors.

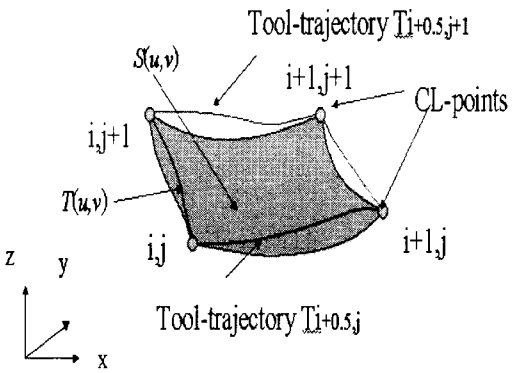


Figure 5. Error surface computation

3.1 Angle Adjustment

If the *a*-angle jumps unexpectedly from minimum to maximum (e.g. from 5 degree to 355 degree) or vice versa, this may cause an expected collision among each of the 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 values. 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
```

3.2 Angle Switching

According to our initial setup of the 5-axis machine, the *a-angle* is given by $a = \tan^{-1}(j/i)$, $0 \le a \le 2\pi$. However, the normal vector *i and j* can be in any of the four quadrants. Therefore, we can define

$$a_{base}=\begin{cases}\tan^{-1}(\frac{j}{i}), \text{if } i\ge 0 \text{and } j\ge 0,\\ \pi+\tan^{-1}(\frac{j}{i}), \text{if } (i<0 \text{ and } j>0) \text{ or } (i<0 \text{ and } j<0),\\ 2\pi+\tan^{-1}(\frac{j}{i}), \text{otherwise,}\end{cases}$$

(10)

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 required 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,

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

(11)

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 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 regard to this set. We will minimize the distance traveled by the tool in the space *(a, b)* with the Euclidean distance:

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

(12)

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

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

(13)

Note that, we consider only the distance 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 distance is selected to optimize the errors. The actual errors, however, are computed using equation (9) which takes into account both *(x,y,z)* and *(a,b)*.

When the tool enters the zones 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. Figure 6 illustrates the graphical result of this algorithm in which errors are clearly decreased. The errors displayed in Figure 6 enable the user to locate the "problem areas" where the loops exist or the errors are greater than the tolerance.

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 3.1 only eliminates unwanted variations of more than 180 degrees. We still face the problem of large variation of the angle between 0 to 180-degrees 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 algorithm called "Angle Switching" 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 formula above, or $\{a_{i+1} +\pi,-b_{i+1}-\pi)$ or $(a_{i+1}-\pi,-b_{i+1}-\pi)$. 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, the Angle Switching algorithm must be applied only in the vicinity of an extrema. 2) In order to obtain the optimal path, the graph–based shortest path algorithm must be employed.

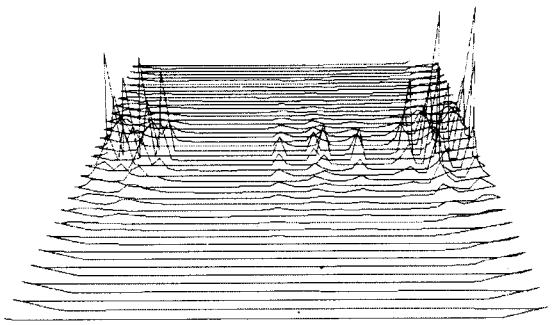


Figure 6a. Error surface without angle switching

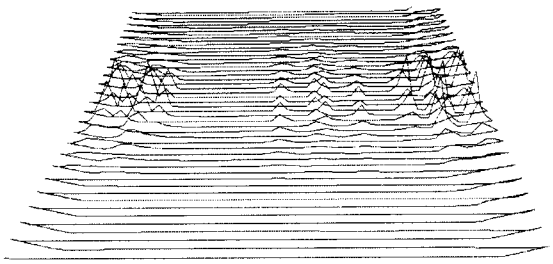


Figure 6b. Error surface with angle switching

The "Angle Switching" algorithms for optimizing the tool path by computing the smallest distance within the vicinity of the errors are 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 distance optimization using the saddle surface that contains both convex (maximum) and concave (minimum) areas. We use our virtual five-axis simulator 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. The large circle or loop (Fig 10a) represents the maximum error of the saddle surface.

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 Figure 7. 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.

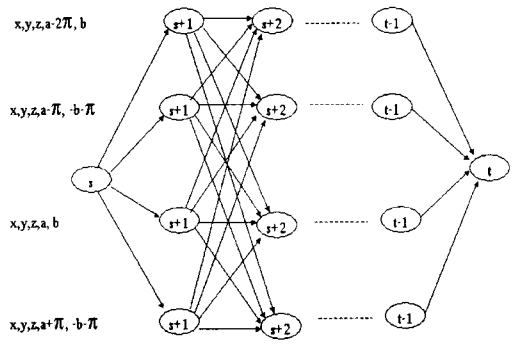


Figure 7. Distant graph represents four feasible paths from s to t

3. Apply the Dijkstra's shortest path algorithm [13] to compute the smallest distance from s to t, from the distance graph in Figure 7.

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

5. Increase the number of vertices from s to t until the total errors are minimized. Note that

the position of *s* and *t* on the surface is the same, although their new vertex indices are different. The new vertex indices are computed using the following equation:

$$new\_id = old\_id * \frac{(new\_size - 1)}{(old\_size - 1)} \tag{14}$$

4. Experimentation and Results

The objective of the pilot 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 designed for a telephone set, which contains both convex (maximum) and concave (minimum) regions is used as a case study to demonstrate the error estimation and minimization. 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 phone surface is given below.

$$P(u,v) = \begin{bmatrix} 20u - 10 \\ 20v - 10 \\ -20\left[(u-0.3)^2 + (v-0.3)^2\right] \\ e \end{bmatrix} - 20\left[(u-0.7)^2 + (v-0.7)^2\right] + e \tag{15}$$

The normal to the surface is computed using the derivative with regard to u and v.

$$N(u,v) = \frac{\partial P/\partial u \times \partial P/\partial v}{|\partial P/\partial u \times \partial P/\partial v|} \tag{16}$$

We then construct a zigzag tool path of 40x40 points.

$$n_i = 40, i = 0..39$$
$$n_j = 40, j = 0..39$$

The original CL points (X, Y, Z, I, J, K) are calculated from:

$$X_{i,j} = P(u_i, v_j)_0 \qquad I_{i,j} = N(u_i, v_j)_0$$
$$Y_{i,j} = P(u_i, v_j)_1 \qquad J_{i,j} = N(u_i, v_j)_1 \tag{17}$$
$$Z_{i,j} = P(u_i, v_j)_2 \qquad K_{i,j} = N(u_i, v_j)_2$$

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 8 shows the required saddle surface. Fig 9 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.

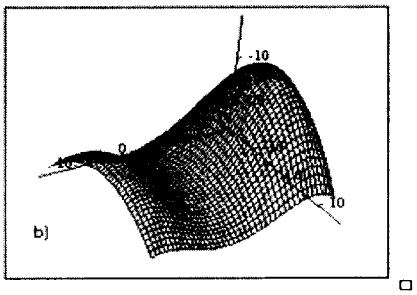


Figure 8: The saddle surface

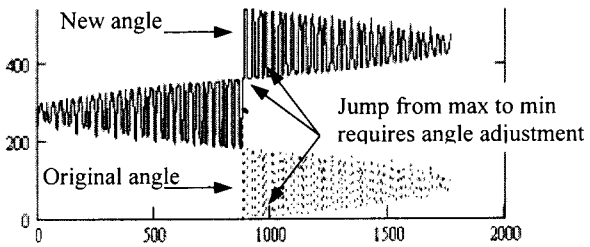


Figure 9: Angle adjustment for the saddle surface

Figure 10a illustrates the saddle surface produced by the virtual machine without angle switching. Figure 10b shows that the maximum error is at vertex number 272 and the virtual machine automatically locates the vicinity of the large milling errors to determine the source vertex as 268 and the destination vertices as 285. Note that, the large circle or loop represents the maximum error of the saddle surface at vertex number 272.

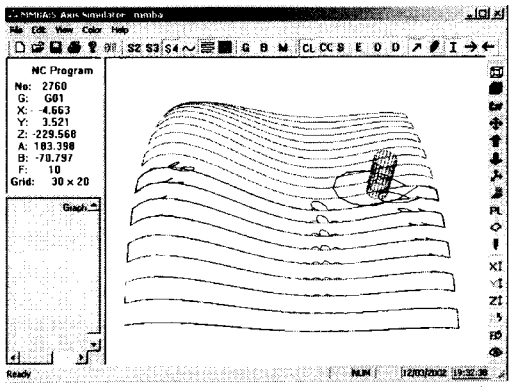


Figure 10a. Error identified by the virtual machine for the saddle surface

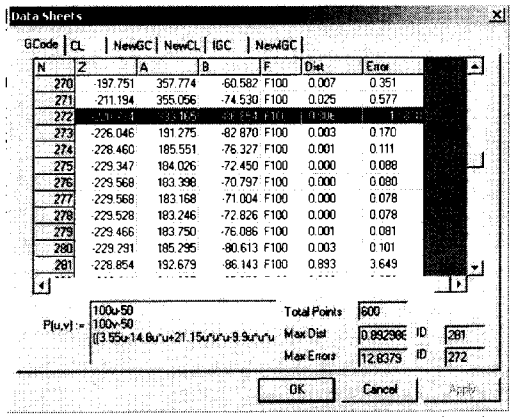


Figure 10b. Maximum error identified for the saddle surface

In order to minimize such errors, first, we apply the angle switching algorithm to the above mentioned regions by constructing the graph to represent four feasible paths from vertex 268 to 285 and apply the Dijkstra's shortest path algorithm to compute the smallest distant for this region. The result is given in Figure 11 and Table 1. The large circle that makes up maximum errors has disappeared. We can apply angle switching to other regions in a similar manner.

Next, we increase the number of vertices for the saddle surface until the total errors are minimized. Note that the source and destination vertex for each region on the surface is the same, although their new vertex indices are different. For example, one of the zigzag tool paths in Table 1 is 20x20, the maximum distance is located between the vertex 181 and 182. The physical vertices 176 to 187 determine the vicinity of errors such that the maximum distant vertex is in the middle of this vicinity. If we increase the size of the tool path to 40 x 20, the distant from the same region of the tool path should be mapped to the new vertex indices of 361 to 383, e.g. 361 = 176 * 39 / 19.

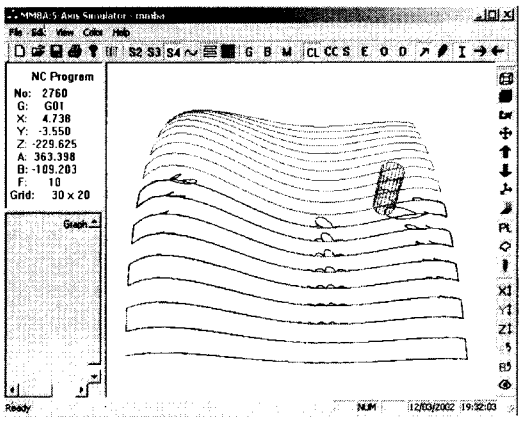


Figure 11. Optimized tool path with minimum error of the saddle surface

The numerical results of our experiment are given in Table 1, which compares the distances and errors (in mm unit) between non-optimized (without angle switching), and our shortest path optimized tool path (with angle switching). Errors and distances are computed using equation 9 and 12 respectively. Different tool path produces different distances and errors depending on the zone of large milling errors depending and number of points in this zone. However, the angle-switching algorithm always reduces the distances and errors. Our algorithms have demonstrated up to 85 percent error reduction. Moreover, the optimum number of points for each surface can also be determined based on a prescribed tolerance. Note that when we increase the number of points to a certain limit, both optimized and non-optimized give the same result in which optimization is not required. For the saddle surface, we can conclude that the shortest path optimization is very efficient with less number of points and when we increase the number of points to 150 x 20, the optimization is no longer required. If we define a prescribed tolerance to be 0.150 mm, the optimum number of points in this case would be 50x20. Figure 12 shows the finished part of the saddle surface using 100x100-mm workpiece and 6-mm tool size. The result is a good surface quality, with an average undercut.

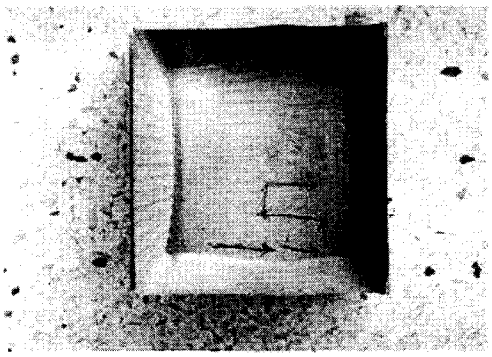


Figure 12. The final cut of the phone surface

5. Conclusions

We developed 3D interactive error estimator software for a 5-axis milling machine. The software evaluates a correct position of the tool in the 5-dimensional space, i.e. the location and the orientation of the tool and computes the kinematics errors for an arbitrary sculptured surface. The software is also capable of visualizing errors graphically and numerically. This makes it easy to locate the "problem areas" where the loops exist or the error is greater than the tolerance. Thus, large milling errors are detected and can be corrected beforehand. Each CL-point may be analyzed and edited manually. It is also possible to reduce the error by modifying the CL-points manually. Besides, two algorithms called "Angle Adjustment" and "Angle Switching" are capable of minimizing such errors during the runtime simulation. The software has been verified by surfaces having the saddle type regions, multiple extreme, steep regions with a high curvature etc.

6. Acknowledgement

The author wishes to thank the National Electronics and Computer Technology Center, National Science and Technology Development Agency of Thailand for their financial support. I also would like to thank Dr. S.S. Makhanov for his helpful ideas and suggestions.

7. References

[1] Srivastava AK., Veldhuis SC. and Elbestawit MA., *Modelling Geometric and Thermal Errors in a Five-axis CNC MachineTtool*, Int J Mach Tools Manufact, Vol.35, No.9, pp. 1321-1337, 1995.

[2] Mu YH. And Ngoi KA., *Dynamic Error Compensation of Coordinate Measuring Machines for High-Speed Measurement*, Int J Adv Manuf Technol, Vol.15, pp. 810-814, 1999.

[3] Tu JF., Bossmanns B. And Hung SCC., *Modeling and Error Analysis for Assessing Spindle Radial Error Motions*, Precision Engineering, Vol.21, No.2, pp. 90-101, 1997.

[4] Bhat V. and De Meter EC., *An Analysis of the Effect of Datum Establishment Methods on the Geometric Errors of Machined Features*, Int J Mach Tools Manufact, Vol.40, No.13, pp. 1951-1975, 2000.

[5] Aekambaram R. And Raman S., *Improved Toolpath Generation, Error Measures and Analysis for Sculptured Surface Machining*, Int J Prod Research, Vol.37, No.2, pp. 413-431, 1999.

[6] Bohez ELJ., *Compensating for Systematic Errors in 5-axis NC Machining*, CAD, Vol.34, 2002, pp. 391-403.

[7] Koren Y., *Five-axis Interpolators*, Annals of CIPR, Vol.44, No.1, 1995, pp.379-382.

[8] Lin R. And Koren Y., *Real Time Interpolator for Machining Ruled Surfaces*, Proc. ASME Annual Meeting, DSC-Vol.55-2, pp.951-960, 1994.

[9] Altan T., Lilly B., Kruth J.P., Konig W., Tonshoff H.K., Vanluttervelt C.A. and Khairt A.B., *Advanced Techniques for Die and Mold Manufacturing*, Annals of CIPR, Vol.42, No.2, pp.707-716, 1993.

[10] Jeong J. And Kim K., *Generation of Tool Paths for Machining Free-Form Pockets With Islands Using Distance Maps*, Advanced Manufacturing Technology, 15, pp.3111-316, 1999.

[11] Suh S.-H. And Shin Y.-S., *Neural Network Modeling for Tool Path Planning of the Rough Cut in Complex Pocket Milling*, Journal of Manufacturing Systems, Vol.15, No.5, pp.295-324, 1996.

[12] Mud-Armeen Munlin., *Interactive Constraint-Based Assembly Modeling*, Thammasat International Journal of Science and Technology, Vol.6, No.1, pp 36-45, 2001.

[13] Evans James R. and Edward Mineka., *Optimization Algorithms for Networks and Graphs*, Marcel Dekker Inc., 1992.

Tool Path Grid Array (X x Y)	Maximum Errors Vertices	Vicinity of Maximum Errors	No Optimization (mm)		Shortest Path Optimization (mm)		% of error reduction
			Distance	Errors	Distance	Errors	
20 x 20	181-182	176-187	0.067	2.922	0.009	0.446	84.74
30 x 20	272-273	268-285	0.103	1.166	0.010	0.185	84.13
40 x 20	375-376	361-383	0.109	0.653	0.050	0.163	75.04
50 x 20	469-470	453-482	0.129	0.532	0.035	0.135	74.62
60 x 20	563-564	546-580	0.077	0.153	0.021	0.124	18.95
70 x 20	657-658	639-679	0.080	0.135	0.031	0.115	14.81
100 x 20	943-944	917-974	0.087	0.108	0.072	0.101	6.48
130 x 20	1229-1230	1195-1270	0.088	0.096	0.088	0.095	1.04
150 x 20	1419-1420	1380-1466	0.090	0.092	0.090	0.092	0

Table 1. Results of the shortest path tool path optimization