Motion and Virtual Cutting Simulation System for a Five-Axis Virtual Machine Tool

Rong-Shean Lee* and Ko-Jen Mei

Department of Mechanical Engineering, National Cheng Kung University, Taiwan

(**Received** 31 March 2011**; Published online** 1 September 2011)

*Corresponding author: mersl@mail.ncku.edu.tw

DOI: 10.5875/ausmt.v1i1.101

Abstract: Machine tools are an indispensable link in high-efficiency high-tech production. However, since five-axis machine tools are very costly and their use requires a high level of knowledge and expertise, a virtual machine tool must be used to simulate five-axis machine tool operation. Configuration code or a mechanism topology matrix must be used to describe a machine tool, and can be used as the framework for design of a virtual machine tool system. The first step is to isolate the basic motions of each element of a virtual machine tool and then establish their coordinate systems. The establishment of a node tree allows coordinate transformation matrices for virtual motion components to be derived, which are then used to simulate movements. The simulation of virtual cutting must take into consideration both accuracy and efficiency. While either a GPU or CPU can be used to perform calculations, there are currently restrictions on GPU memory use which results in relatively lower accuracy. In contrast, a CPU can perform calculations using an adaptive octree with voxels and multithreading to yield sufficient accuracy and efficiency. A five-axis virtual machine tool motion and virtual cutting simulation system was written in C/C++ with OpenGL and OpenMP, and can perform real-time cutting simulations.

Keywords: multi-axis; virtual machine tool; cutting simulation; octree.

Introduction

In the face of strong global competition, many domestic industries are pondering the questions of how to shorten product life cycle, quickly respond to the market's needs, and reduce production costs. In order to achieve these goals, many firms are upgrading their capabilities through the use of information technology, which is inevitably changing conventional product design and manufacturing processes. Although machine tools are costly and require a high threshold of knowledge, they are an indispensable link in high-efficiency high-tech production. Consequently, many types of machine tool evaluation and simulation software have emerged for the purpose of shortening processing time, boosting accuracy, and cutting costs. Attention must be paid to both machine tool hardware and software in order to achieve optimal performance; machine tool modeling technology in particular is extremely important to avoid large losses caused by incorrect processing.

Due to the growing need to process increasingly complex curved surfaces in recent years, multi-axis processing has been adopted as an important means of ensuring accuracy when processing complex workpieces. The motion behavior of four-axis and five-axis machine tools is more complex and difficult to model than that of three-axis machine tools; as a result, the assistance of a virtual machine tool is even more important when manufacturing with these multi-axis machine tools.

The simulation of cutting of workpieces with different geometries is an important issue in virtual machine tool-assisted manufacturing. Virtual cutting simulations can enable engineers to inspect the processing of a virtual workpiece, providing a valuable reference for process corrections. Virtual cutting simulation programs have high design thresholds; apart from visualization, they must also provide sufficient

computational efficiency and cutting accuracy. In addition, computational efficiency and cutting accuracy tend to be negatively correlated with one another.

Jang, Kim, and Jung [1] used a voxel model and Boolean computation to remove the overlap point between a virtual cutter and virtual workpiece; this system can be used for virtual cutting with a multi-axis machine tool. Ding *et al.* [2] employed an octree and oriented bounding box to find overlapping points in virtual space; this method can reduce memory use and increase computational efficiency. Tung [3], Yang [4], and Tsai [5] *et al.* also used an adaptive octree to establish a virtual machine tool cutting system.

A virtual multi-axis machine tool motion system

1. An explanation of the machine tool's mechanism

A three-axis machine tool possesses axes moving linearly in three directions, a four-axis machine tool adds a rotating axis, and a five-axis machine tool adds two rotating axes to a three-axis machine tool. Five-axis machine tools can be classified based on the mounting locations of the rotating shafts. When the rotating shafts are both mounted between the base and work table, the machine tool is termed a table-tilted machine tool. When the rotating shafts are both mounted between the base and spindle, it is termed a spindle-tilted machine tool. When one rotating shaft is mounted between the base and work table, and the other shaft is mounted between the base and spindle, it is termed a table/spindle -tilted machine tool.

Ordinary multi-axis machine tools can be described using a configuration code [6], where each axis of motion is expressed using a letter, the directions of motion of the three linear motion axes are expressed as *x*, *y*, *z*, the directions of motion of the rotating axes are expressed as *a*, *b*, *c*, the base is expressed as the letter *o*, and each axis of motion is numbered in its sequential position between the work table end and the cutter end. In addition, division sign lines and symbols indicating the work table end and cutter end are added to the two ends. This is how configuration codes are expressed for machine tools. For instance, *T*/cbxyoz/*S*(*z*) indicates an table-tilted five-axis machine tool.

Machine tool mechanisms can also be described using mechanism topology matrices. A mechanism topology matrix can describe the structure of a mechanism [7]. It can describe the connection between a link and joint and the types of link and joint. A topology matrix is written using the following form:

$$\begin{bmatrix} K_{x1} & J_{y12} & J_{y13} \\ z_{21} & K_{x2} & J_{y23} \\ z_{31} & z_{32} & K_{x3} \end{bmatrix} \quad (1)$$

Here, K_{xn} is a link, x is the link type, and n is the link number; J_{ymn} is a joint, y is the joint type, m and n are the numbers of the links connected with that joint, and z is the joint number. If there is no joint connecting the two shafts, this z is expressed as 0. If there is a joint between the links, the junction of the upper triangular matrix is denoted as J_{ymn} and the junction of the lower triangular matrix is z_{mn}. Here, z is typically denoted by a lower-case letter.

A configuration codes is a succinct way of describing the mechanism of a machine tool. The advantage of using a topology matrix to describe a machine tool mechanism is that it can describe machine tools with special structures and allows virtual machine tool systems to be designed with a large degree of flexibility.

2. Coordinate system and node tree

A virtual machine tool can be divided into many virtual motion components on the basis of a configuration code or mechanism topology matrix. Each virtual component has a coordinate system; because of the linkage between different virtual motion components, these coordinate systems must be converted.

A tree-shaped chart can be drawn to represent the structure of a machine tool. In Figure 1(a), each node represents a coordinate system, and each branch is a coordinate transformation matrix. Each virtual component coordinate system in a virtual machine tool can be derived by sequentially multiplying the coordinate transformation matrices from the base to that component. As a consequence, controlling the coordinate transformation matrices between virtual components can drive the motion of the virtual machine tool.

Cutting Simulation involves obtaining the relative coordinates of the cutter and workpiece, and the conversion process is performed by multiplying each coordinate transformation matrix from the workpiece to the cutter (see Figure 1 (b)).

Rong-Shean Lee is a NCKU Distinguished Professor at Department of Mechanical Engineering, National Cheng Kung University, Taiwan. Professor Lee received his Ph.D. degree in mechanical engineering in 1982 from University of Leeds, U.K. He currently also serves as the Associate Director of the Engineering Technology Promotion Centre of National Science Council. He is a leading researcher in collaborative CAD/CAM and multi-axis virtual machine tools. His areas of interests include Computer-Aided Manufacturing, Intelligent Manufacturing, Applied Plasticity and Metal Forming System.

Ko-Jen Mei currently is a Ph. D. student at Department of Mechanical Engineering, National Cheng Kung University, Taiwan. He received Master degree in mechanical engineering with software engineering expertise in 2009 from National Cheng Kung University. His research focuses on CAD/CAM and virtual machine tool simulation technology.

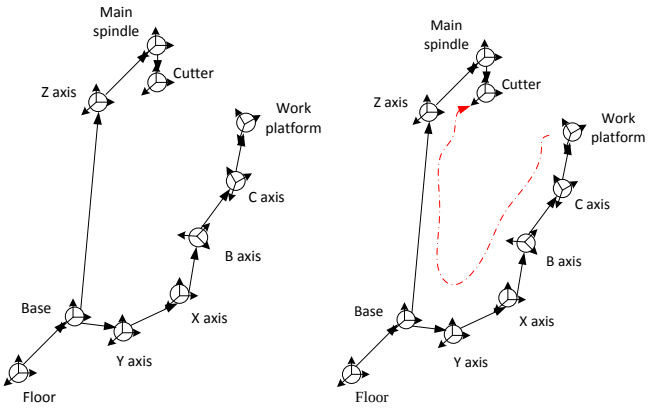


Figure 1. (a) Element coordinate systems in a virtual machine tool; (b) Coordinate conversion system for cutter relative to work piece.

Virtual cutting simulation system

1. *Application of GPU and voxel method to cutting simulation*

The simulation of virtual cutting must take into consideration accuracy and efficiency, and either a CPU or GPU can be used to perform a cutting computation. A GPU is a computing device that allocates in a computer's graphics card; in contrast with a CPU, it can process large amounts of data, but it excels at generating simple and repetitive calculations. A GPU can therefore be used for large-scale coordinate conversion operations.

A virtual workpiece and virtual cutter can be expressed using voxels [1], which involves expressing a virtual geometry in terms of a points array. A GPU can rapidly perform coordinate conversion for large numbers of voxels, and can put the voxels of a virtual cutter and virtual workpiece in the same coordinate system, check whether there are any overlaps between points, and eliminate overlapping points from the virtual workpiece, which results in simulation of cutting.

Because GPU memory is more limited than that of a CPU, it cannot achieve very high levels of accuracy.

2. *Applying adaptive octrees in cutting simulations*

Because large amounts of memory must be used to fill a virtual workpiece with voxels, performing cutting calculations will expend a vast amount of computational resources. The adaptive octree concept [3-5] is therefore employed to reduce the quantity of voxels.

An octree is a data structure. As shown in Figure 2 (a), an octree is suitable for ordering and storing voxels, and the ordering of voxels can conserve search time during a cutting simulation. The tree data structure is composed of numerous nodes; apart from the branching nodes, each node has a parent node. The parent nodes of a voxel octree include the larger cubes consisting of

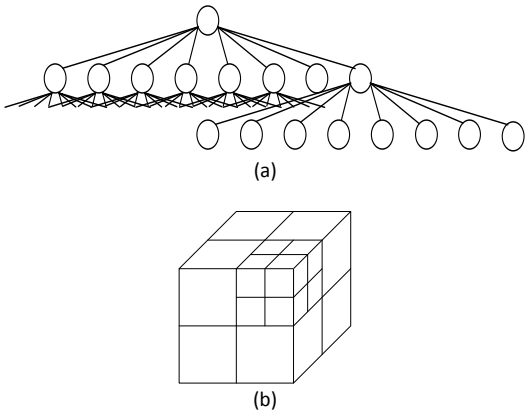


Figure 2. (a): Octree; (b) Geometric shape corresponding to a voxel octree.

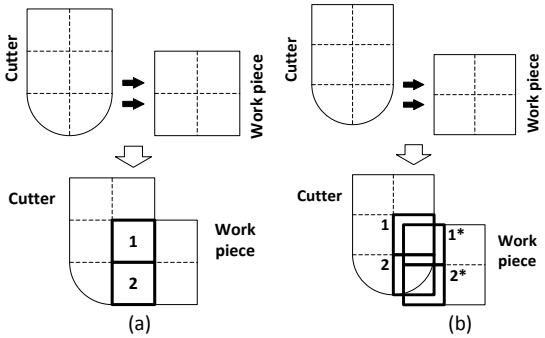


Figure 3. (a) Comparison of voxels using a parallel algorithm without interference; (b) Comparison of voxels using a parallel algorithm with interference.

the child node cubes, and can be divided into at most eight little cubes on the basis of its eight corners (see Figure 2 (b)).

The parent node of an octree filled with voxels can be simplified as a single node to form an adaptive octree. This kind of node is in a compressed state when not in use. Child nodes are released only when it is desired to use the existing child nodes within the parent node.

An STL file stores a triangular lattice including geometric graphic surfaces composed of normal vectors. This file can be used to first calculate the voxels on a geometric surface, and then fill in voxels in the direction of their normal vectors. If voxels are able to fill octree nodes during the filling process, those groups of voxels can immediately be compressed as the nodes, thereby conserving memory. After this filling process is completed, an adaptive octree is obtained.

3. *Use of CPU multithreading in cutting simulations with an adaptive octree*

Multithreading is a form of parallel computation where one computation element executes one thread. Virtual cutting of a virtual workpiece involves searching for overlapping voxels on two adaptive octrees. This means that parallel comparison of different blocks may

be performed. As in Figure 3(a), block 1 and block 2 can be subjected to voxel comparison in different threads without interference.

In the multithreading algorithm, and effort must be made to avoid mutual interference between threads. There are two situations in which serious interference may occur in multithreading cutting simulation employing an adaptive octree: The first is when an adaptive octree is decompressed, and the other is when cut voxels are eliminated from a virtual workpiece. As shown in Figure 3(b), each grid square is an adaptive octree node; the virtual cutter's node 1 is compared with the virtual workpiece's node 1^*, and the virtual cutter's node 2 is compared with the virtual workpiece's node 1^* and node 2^*. Here, interference occurs at node 2. Interference may result in node 1^* being decompressed twice, being repeatedly cut, or some other unexpected result.

The most common CPUs currently in use have four or eight cores. Each core represents a thread that can be executed. As a consequence, not all operations can be executed in parallel, but must rather be executed in parallel in batches. To solve the problem of interference, the execution order of parallel operations must be arranged so that nodes do not repeatedly appear in a single parallel operation.

Approach of a virtual machine tool motion and cutting simulation system

The computational core of this study's virtual machine tool motion and cutting simulated program was written in C/C++, which enabled greater execution efficiency than most other ordinary high-level programming languages. Geometric imaging of the virtual machine tool was performed using OpenGL [8], and an STL file was used to import individual motion components and draw them after filling with triangular lattices (Figure 4). The virtual workpiece was drawn using a points array in the adaptive octree (Figure 5).

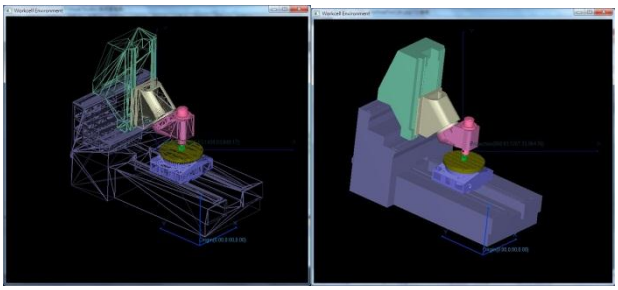


Figure 4. Use of a triangular lattice to display a virtual machine tool. (the image on the left is drawn directly using the triangular lattice; the image on the right is drawn after filling the triangular lattice).

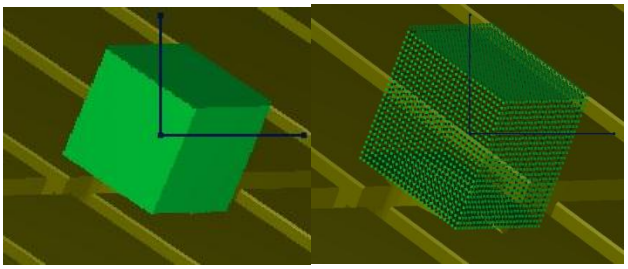
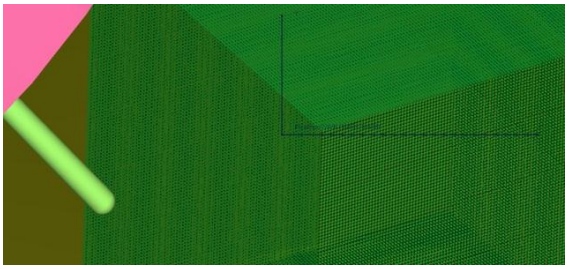
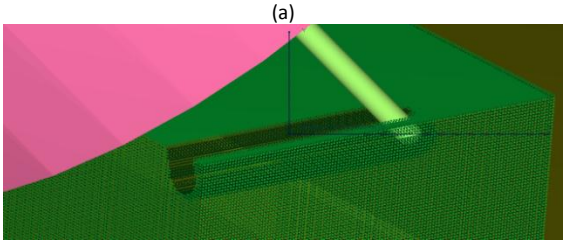


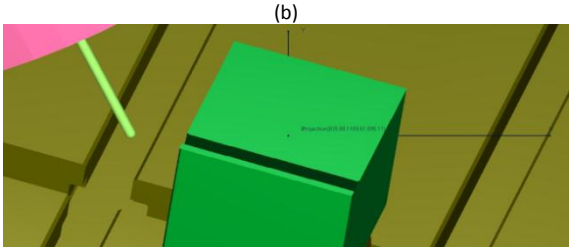
Figure 5. A virtual workpiece displayed as a points array (a distant view is shown on the left, and a close-up view on the right).



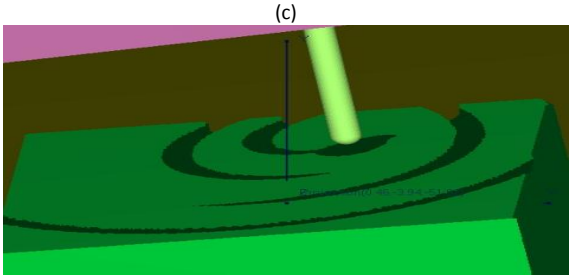
(a)



(b)



(c)



(d)

Figure 6. (a) Enlarged view of virtual workpiece; (b) Enlarged view of simulated cutting; (c) Reduced view of simulated cutting; (d) Simulated five-axis cutting.

Multithreading of virtual cutting was written using the OpenMP [9]. Figure 6 shows virtual cutting, Figure 6 (a) and Figure 6 (b) show enlarged views of the virtual workpiece. Because the voxels appear from the adaptive octree only at the time of cutting, hollow cubes are seen in Figure 6 (a), and voxels appear along the cutting path in Figure 6 (b). Figure 6 (c) shows a reduced view of the virtual workpiece after cutting, and Figure 6 (d) shows a simulation of five-axis cutting.

The dimensions of the virtual workpiece in Figure 6 are 50 x 50 x 50mm^3. When the preset cutting accuracy is 0.05 x 0.05 x 0.05mm^3, it is necessary to use about 4GB of memory. The computational efficiency is determined by the cutting depth and feed rate. Because the adaptive octree only appears during cutting, the size of the area cut is directly proportional to the computation time. When an Intel 2.8GHz 64bit 8-core CPU was used, cutting depth was approximately 3mm, and feed rate was 360mm/min, it was possible to complete one cycle of virtual cutting operations in 0.1s.

Conclusions and future outlook

The use of adaptive octrees can enhance the efficiency of virtual cutting while reducing memory loads. The expansion of an adaptive octree yields voxels, which enables the cut voxels to be derived; this in turn allows the depth of the cut to be calculated and engineering assessments to be performed, such as calculating the cutting force. Figure 7 shows cutting depth and cutting force information from the virtual cutting calculations. This cutting simulation system can avoid unnecessary processing losses, and can boost processing quality and efficiency by facilitating engineering assessments.

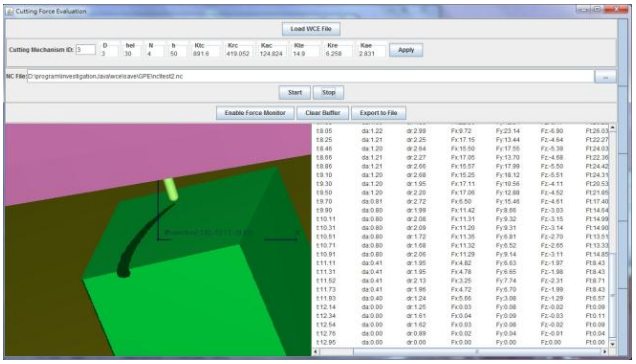


Figure 7. Simulated cutting force estimation for a virtual five-axis machine tool.

References

[1] D. Jang, K. Kim, and J. Jung, "Voxel-based virtual multi-axis machining," *The International Journal of Advanced Manufacturing Technology,* vol. 16, no. 10, pp. 709-713, 2000. doi: 10.1007/s001700070022

[2] S. Ding, M. A. Mannan, and A. N. Poo, "Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces," *Computer-Aided Design,* vol. 36, no. 13, pp. 1281-1294, 2004. doi: 10.1016/S0010-4485(03)00109-X

[3] Y. C. Tung, "Adaptive NC simulation for multi-axis solid machining," M.S. thesis, Graduate Institute of Mechanical Engineering, National Chung Cheng University, 2004.

[4] Y. J. Yang, "Cutting simulation for solid models on five-axis machine," M.S. thesis, Graduate Institute of Mechanical Engineering, National Taiwan University, 2008.

[5] M. J. Tsai, "Adaptive 3D solid modeling for multi-axis NC simulation and haptic device," M.S. thesis, Graduate Institute of Mechanical Engineering, National Chung Cheng University, 2006.

[6] Y. H. Lin, "Study on universal construction of five-axis. Virtual machine tool simulation system," M.S. thesis, Graduate Institute of Mechanical Engineering, National Chung Cheng University, 2004.

[7] H. S. Yan, *Creative design of mechanical devices*, 1st ed. Singapore: Springer, 1999.

[8] *OpenGL - the industry standard for high performance graphics*. Available: http://www.opengl.org/

[9] *OpenMP.org*. Available: http://openmp.org/wp/