Int J Adv Manuf Technol (2010) 50:677–687
DOI 10.1007/s00170-010-2530-9

ORIGINAL ARTICLE

Quadtree-array-based workpiece geometric representation on three-axis milling process simulation

Jian Guang Li · Jian Ding · Dong Gao · Ying Xue Yao

Received: 6 October 2009 / Accepted: 7 January 2010 / Published online: 29 January 2010

Abstract A workpiece hybrid representation method based on quadtree-array is presented to improve the geometric simulation efficiency on three-axis milling process. The method takes the advantages both of Z-Map and quadtree in simulation model representation. The discrete points managed by using Z-Map algorithm is to represent the whole model, while the points used to represent the simulated surface detail are managed with quadtree-array. The method can reduce the levels of a quadtree without losing simulation accuracy. A dynamic optimization algorithm to the quadtree structure is highlighted to reduce its total nodes in simulation process. As a result, the simulation efficiency can be improved significantly. A three-axis milling process simulation system based on quadtree-array representation was developed and used to evaluate the performance of the presented method. The evaluated results show that quadtree-array-based hybrid representation method of workpiece can improve the simulation efficiency significantly, and reasonable division number of array cells is also recommended.

Keywords Quadtree-array · Geometric modeling · Simulation efficiency · Three-axis milling · Machining process simulation

1 Introduction

Machining process simulation, which simulates and evaluates the actual numerically controlled (NC) machining process by means of computer simulation, is an important technology in computer NC (CNC) machining process optimization. The geometric representation method, which directly influences much on the simulation accuracy and speed, is one of important research aspects in NC machining process simulation [1]. An appropriate workpiece representation in simulation is required not only reflecting the details on the simulated surface accurately but also performing the simulation efficiently.

Many researchers devoted their efforts to the geometric representation of workpiece, and several methods had been proposed. These methods can be classified into three categories. They are image space-based method, solid modeling-based method, and discrete method. The image space-based methods such as Z-buffer method [2] and Ray-casting method [3] utilize attribute setting of pixels in image space to represent workpiece and simulate machining process. The methods are easy to put into practice due to its simple algorithm and good visualization, but the view orientation is fixed during simulation, and the simulation accuracy is not high enough. The solid modeling-based method simulates the material removal process by Boolean subtraction operation between workpiece model and cutter swept volume [4]. The method is typically exampled by constructive solid geometry and boundary representation (B-Rep). Sungurtekin and Velcker [5] utilized the Boolean operation among primary bodies to simulate machining process. Fleisig and Spence [6] studied on the rough machining simulation of 2-1/2D cavity based on B-Rep. The methods are of higher simulation accuracy, but the simulation efficiency will be decreasing as the simulation goes, and a modeling kernel is needed.

Discrete method is a category of approximate methods to represent workpiece by discretizing it within a given accuracy. The method simplifies the complicated 3D

J. G. Li (✉) · J. Ding · D. Gao · Y. X. Yao
School of Mechatronics Engineering,
Harbin Institute of Technology,
Harbin 150001, China
e-mail: mejgli@hit.edu.cn

Boolean subtraction operation into simple 1D or 2D one in simulation. It has been focused on and widely applied due to their simple algorithm. Chappel [7] proposed discrete vector intersection method based on point-vector model to simulate material removal process. Hsu and Yang [8] put forward a simulation method which was suitable for three-axis milling by discretizing workpiece into a set of discrete points (i.e., Z-Map). Wastra et al. [9] proposed even space decomposition (i.e., voxel) to simulate multi-axis machining process. Navazo et al. [10] proposed an octree-based representation method by recursive division of 3D space, and Liu et al. [11] proposed a quadtree representation method similarly. Liu et al. [12] introduced the idea of level of detail (LOD) into three-axis milling simulation and put forward a sampling extraction algorithm based on even adaptive mesh represented with quadtree. LOD representation methods including octree and quadtree have gotten further research and application because they can represent the model in LOD and solve the contradiction between simulation accuracy and simulation speed to certain extent. Multi-axis machining process was simulated with octree representation to optimize the federate in reference [1]. Roy and Xu [13] proposed an extended octree to improve machining simulation speed and efficiency, and based on the extended octree, Li et al. [14] did their research works on cutting parameters optimization via machining process simulation. Wang et al. [15] simulated three-axis milling process by adopting adaptive dynamic quadtree algorithm to improve simulation accuracy. He and Bin [16] presented a geometric model called G-LODs for NC machining simulation. The G-LODs used a type of progressive mesh to construct the surface simulation grid. However, the above-mentioned references related to the discrete methods did not discuss the simulation efficiency but the methods. Karunakaran and Shringi [17] presented an algorithm for instantaneous workpiece conversion from octree (quadtrees) into B-Rep to unprocess the data traversal efficiency.

Three-axis NC machining technology is widely applied in mechanical manufacturing firms. To improve the simulation efficiency of three-axis NC machining, a quadtree-array-based hybrid representation of workpiece is presented in this paper, which takes advantages both of Z-Map and quadtree method to represent workpiece in three-axis milling process simulation.

2 Principle of quadtree-array representation

As a kind of hierarchical data structure, quadtree representation can dynamically control the density of discrete facets so that the complicated surface can be represented in an LOD manner. To improve the simulation efficiency based on quadtree further, quadtree-array-based representation of workpiece is presented. Firstly, the whole surface is evenly divided into a set of sub-surfaces (i.e., cells) then each sub-surface is represented with quadtree. For the whole surface, forest-like quadtree-array will be generated. The level number of each quadtree is decreased for the whole surface represented with quadtree-array compared to that only with a single quadtree. The shortened access path to leaf nodes results in the improvement of simulation efficiency.

2.1 Z-Map model

Z-Map is used to approximately represent the workpiece upper surface by sampling it into a set of discrete points according to simulation accuracy, and the height values of the discrete points are recorded and refreshed during simulation. In machining process simulation, the height values of the discrete points are modified, and then the simulated surface is constructed and rendered real-time to simulate material removal process. This simulation model is with particular advantages, such as simple data structure and rapid to access and modify the data. However, the discrete accuracy influences on simulation speed and accuracy greatly.

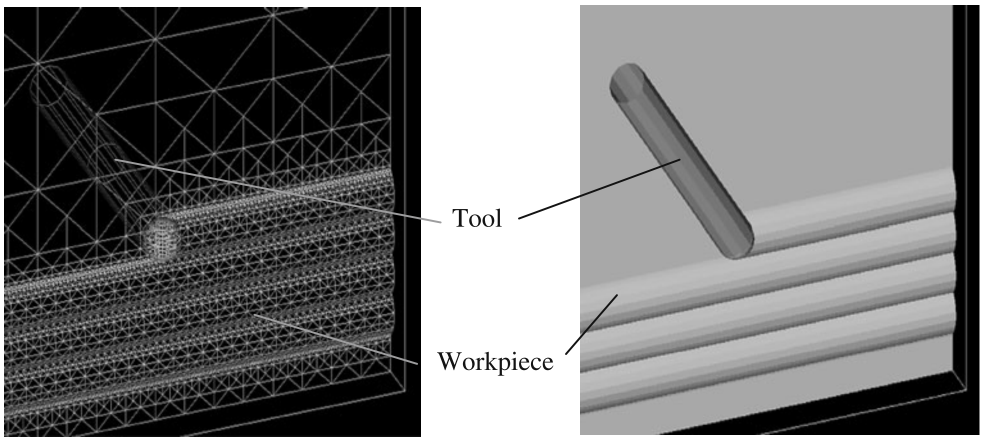


Fig. 1 Quadtree represented milling process simulation

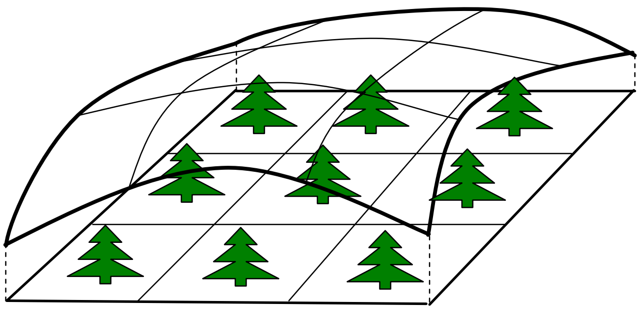


Fig. 2 Quadtree-array model

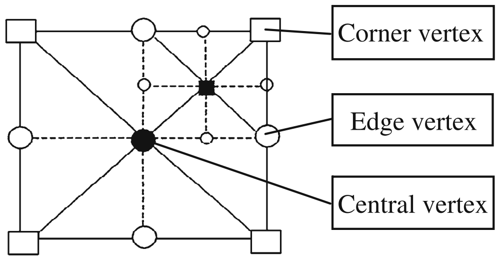


Fig. 4 Mesh model of a node (nine discrete points in total)

2.2 Quadtree model

To construct a quadtree, a square is evenly divided into four sub-squares corresponding to four child nodes, and each sub-square can be subdivided further in the same way till the simulation accuracy is reached. This recursive subdivision results in a quadtree. This model can represent a surface in different LOD with less deal of data dynamically. Figure 1 shows an example of milling simulation represented with quadtree. However, the access speed will be slowed down when the level number of a quadtree gets higher.

Two key aspects of quadtree structure should be considered when simulation is carried out. They are listed as follows:

1. The crack between meshes with different resolutions must be eliminated when the surface is subdivided (please refer to reference [12] for more details).
2. Over-refined redundant nodes of a quadtree must be merged dynamically. The operation is called optimization on quadtree. It will be discussed in a following section in detail.

2.3 Quadtree-array model

As shown in Fig. 2, the bounding square of the upper surface of workpiece is divided evenly into a set of sub-squares (i.e., cells) firstly. Each quadtree manages the discrete geometric data of a sub-surface corresponding to its sub-square. It is called quadtree-array representation in this paper. Before the simulation, each quadtree is initialized to represent the upper sub-surface of the raw stock. Many of the quadtrees will be refreshed to accurately represent the feature of sub-surfaces via recursive subdivisions as the simulation process is going. Meanwhile, the structure of a quadtree is optimized as it grows, and its data is modified. Then, each quadtree is traversed, and the represented sub-surface is displayed to present a visualized simulation result. A reasonable division number of array cells will not only reduce the levels of a quadtree but also decrease the data amount and time consuming on computation as well.

3 Structure of quadtree-array

The minimum bounding square S of a surface can be divided into several sub-squares S_i according to the division number of array cells, and then each sub-surface in sub-square S_i is represented with quadtree according to a given simulation accuracy. All of the quadtrees relevant to the sub-squares is managed in a form of array. In our research, the upper surface of the raw stock is evenly discretized and represented by a set of discrete points according to the given simulation accuracy, i.e., represented with Z-Map, and then the available discrete points are managed with quadtree to construct the detail feature of the surface. In order to dynamically optimize the structure of the quadtree during simulation process, two new data structures for quadtree and discrete points are built, respectively.

3.1 Management of discrete points

For a workpiece with given dimensions, the upper surface can be discretized into a set of discrete points according to simulation accuracy. The discrete points are managed with an array (e.g., *PointArray* in Fig. 3). The data of a discrete point in our research mainly includes a height value z and a label *Flag* (*Flag* is to indicate whether this discrete point is available to construct facet to present the surface detail). In machining process simulation, the data of a discrete point

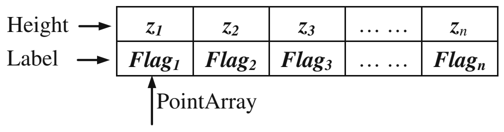


Fig. 3 Data structure of discrete points on curved surface

```
struct node
{
    int      m_pPt_Flag[9];
    int      m_pPt[9];
    int      m_uNo;
    struct node       *m_pChild[4];
    struct node       *m_pParent, *m_pPrev, *m_pNext;
};
```

Fig. 5 Data structure of quadtree node

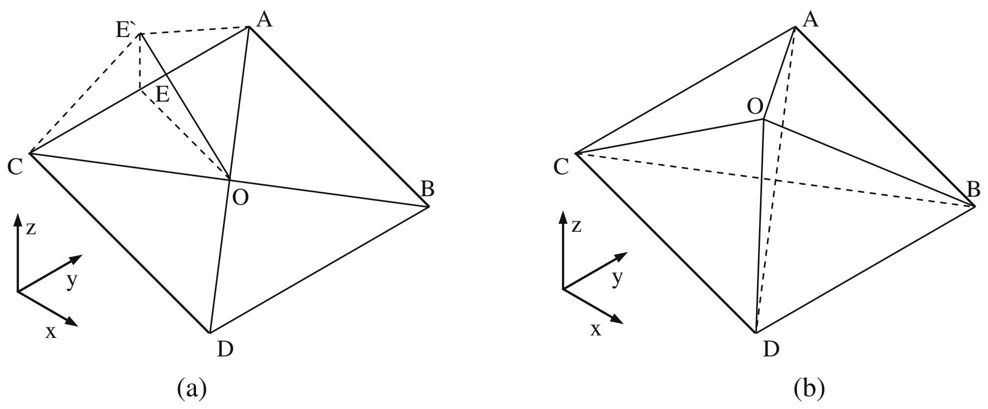


Fig. 6 Vertex evaluation in leaf node evaluation. **a** Evaluation on edge points. **b** Evaluation on central point

can be directly accessed from the array PointArray via the coordinates of the vertex of a quadtree node.

3.2 Structure of quadtree node

Figure 4 shows the mesh model of a quadtree node, and Fig. 5 is the corresponding data structure in C in our research. In Fig. 5, *m_pChild* points to its four child nodes of a node, *m_uNo* represents the sequential number of a node. *m_pPt* stores the index of nine discrete points (i.e., four corner vertexes, four edge vertexes, and a central point shown in Fig. 4) in *PointArray*, which are used to obtain the data of the points, and *m_pPt_Flag* stores the labels of the nine discrete points to identify whether the points are used to construct facets. To dynamically optimize the quadtree in simulation, its original hierarchy structure must be reconstructed into a list in an ascent order of the sequential number of each node. *m_pParent*, *m_pPrev*, and *m_pNext* link to the parent node, precursor, and successor, respectively. They are added to a node and used to construct a complicated bidirectional list from a hierarchal structure.

4 Dynamic optimization on quadtree in simulation

As the machining process simulation goes, large amount of nodes are inserted into the quadtree to describe the dynamically modeled details. However, part of the neighboring nodes represents the same geometric properties, which will decrease the efficiency of simulation. In order to represent the geometric details with the less number of nodes, some nodes and meshes should be merged and deleted after surfaces are subdivided in the given accuracy. By using the dynamic optimization through merging and deleting the redundant nodes, the data amount exponentially growing in simulation will be weakened effectively.

For a quadtree, the optimization is to merge and delete the redundant nodes of the modified quadtrees to improve the simulation efficiency. The following research on optimization takes single quadtree as an example. The nodes of a quadtree can be classified into three types, i.e., root node, intermediate node, and leaf node. In the optimization algorithm, the treatment method of the root node is similar to that of the intermediate node.

The optimization on quadtree is performed on geometric level and data structure level simultaneously. The optimization on geometric level is to evaluate the position deviations of the discrete points used by a node, and optimization on data structure level is to change the connection ship among the nodes. Three aspects should be considered emphatically when optimization on quadtree is carried out:

1. Evaluate the discrete points on the meshes according to simulation accuracy, and determine whether they are reserved or removed.

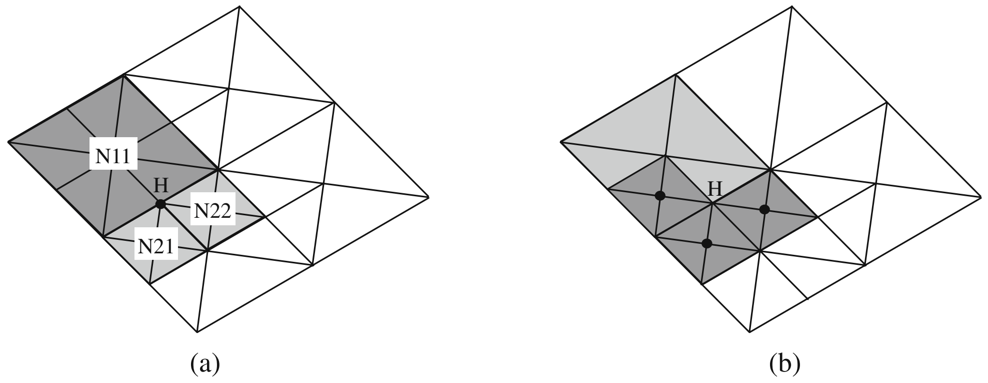


Fig. 7 Evaluation of edge vertex. **a** Estimate the sharing by the nodes on lower level. (**b**) Estimated the existence of the nodes on lower level

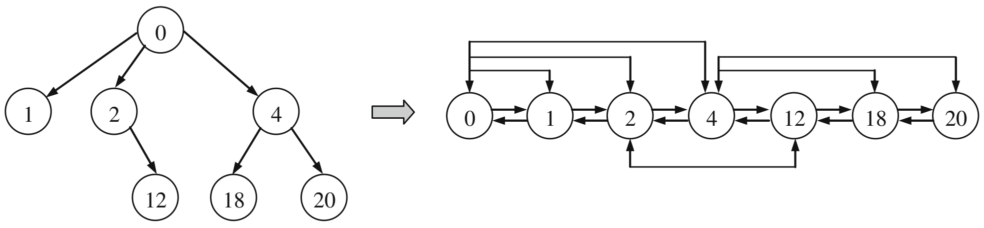


Fig. 8 Translation from quadtree to list

2. Determine whether the node is reserved or not.
3. Merge and delete the redundant nodes.

4.1 Evaluation on the leaf node

Leaf node only needs an evaluation on geometric level because it does not have any child node. The evaluation on geometric level consists of edge vertex evaluation and central vertex evaluation. Taken ideal point E on line CA shown in Fig. 6(a) as an example, the deviation (hence δ_E) between actual point E′ and ideal point E in Z direction should be calculated firstly. If δ_E is larger than a given threshold ε (e.g., simulation accuracy), which means that point E′ is an absolutely necessary point to represent the geometric detail of the surface, reserve the point E′. Otherwise, remove point E′ and line OE′ and merge triangular OAE and OEC into a larger triangular OAC. Since each discrete point (indicated by *m_pPt* in Fig. 5) has a label (the value is stored by *m_pPt_Flag* in Fig. 5) to indicate whether it's an available vertex of a node to construct triangular facets, the evaluation can be carried out on those vertexes with $Flag_i=1$.

If the labels of all of the four edge vertexes equal to 0, an evaluation on the central vertex is necessary. As shown in Fig. 6(b), the distances from point O to line AD and BC in Z direction are calculated, respectively, and take the bigger one as the deviation δ_C. Similar to the treatment method of the edge vertex, point O and the leaf node are removed if

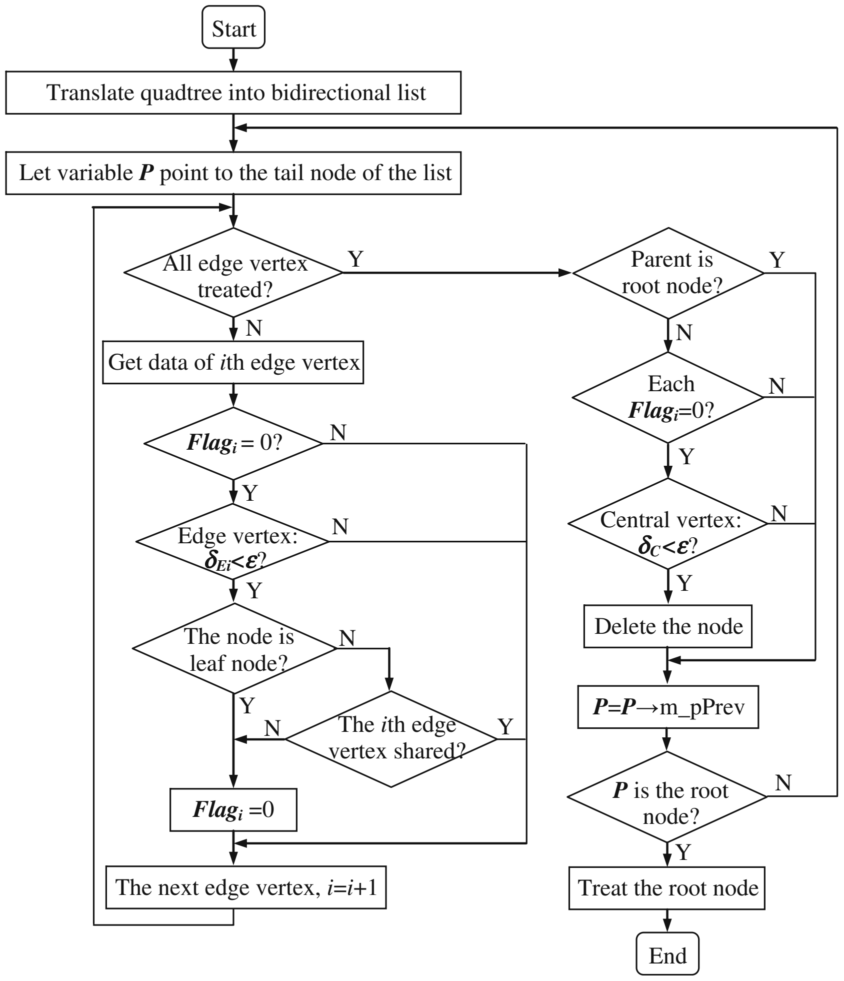


Fig. 9 Flow chart of dynamic optimization of quadtree structure

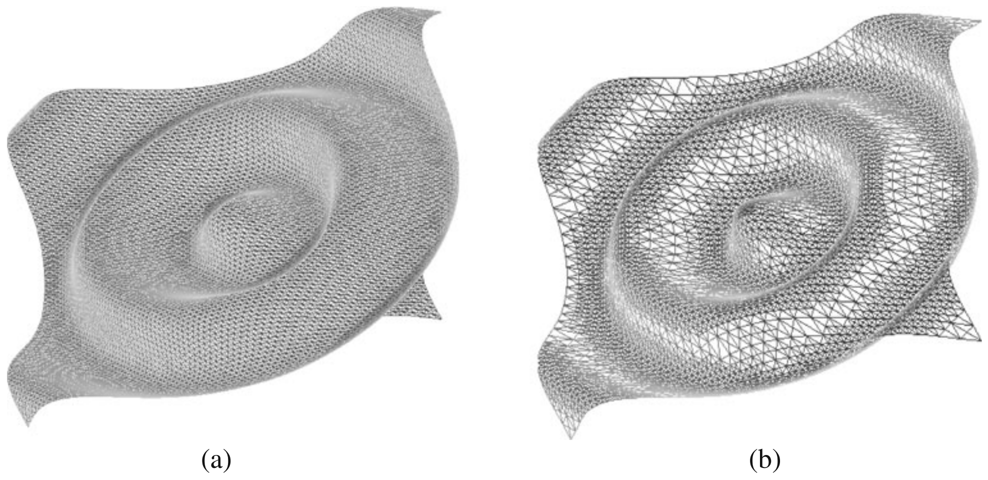


Fig. 10 Comparison of representation with and without optimization on quadtree. **a** Without optimization (21,845 nodes in total). **b** With optimization (4,168 nodes in total)

δ_C is smaller than the given threshold ε; otherwise, reserve them to enhance the detail of the surface.

4.2 Evaluation on the intermediate node

To avoid cracks during subdivision, the intermediate nodes which connect the nodes of upper level (with lower resolution) and lower level (with higher resolution) play a transitional role to guarantee the continuity and integrity of the meshes. An obvious feature distinguishing the intermediate node from the leaf node is that the edge vertex of the former may be shared as a corner vertex by the higher resolution nodes. As shown in Fig. 7(a), the edge vertex H of node N11 is shared by higher resolution nodes N21 and N22 as their corner vertex, respectively. The algorithm is begun with determining whether the edge vertex (e.g., point H) is shared. If the vertex H is shared, then reserve it and continue to examine the existence of the higher resolution nodes which share the vertex H; otherwise, evaluate the vertex H with the same method to the leaf node.

As shown in Fig. 7(b), edge vertex H is shared by more than one node. Existence estimation of the higher resolution nodes must be carried out to determine the vertex H is reserved or deleted. Note that the label of each node is set to 1 when the node is inserted into the quadtree. So, the problem of existence estimation of the higher resolution nodes is translated into an estimation problem of the central vertex according to its label. If one label of all central vertexes of the higher resolution nodes is at least 1, vertex H should be reserved; otherwise, treat the edge vertex H with the same method to the leaf node.

4.3 Implementation of optimization

According to the optimization principle mentioned above, optimization is begun with the classification of the nodes of a quadtree, and the nodes are accessed from the leaf nodes to the root level by level. Considering that the optimization process and modeling process are carried out alternatively, the optimization algorithm is based on the incomplete quadtree structure, and a complicated bidirectional list is created to represent the quadtree so that the traversal of the quadtree can be realized easily.

In the data structure of quadtree node, pointers *m_pParent*, *m_pPrev*, and *m_pNext* are included. *m_pParent* is used to access its parent node, *m_pPrev* and *m_pNext* are used to access its precursor and successor in the list, respectively. In the process of node merging, the hierarchal structure of quadtree is translated into a list without losing the original connection ship via accessing to *m_pParent*, *m_pPrev*, and *m_pNext* (Fig. 8).

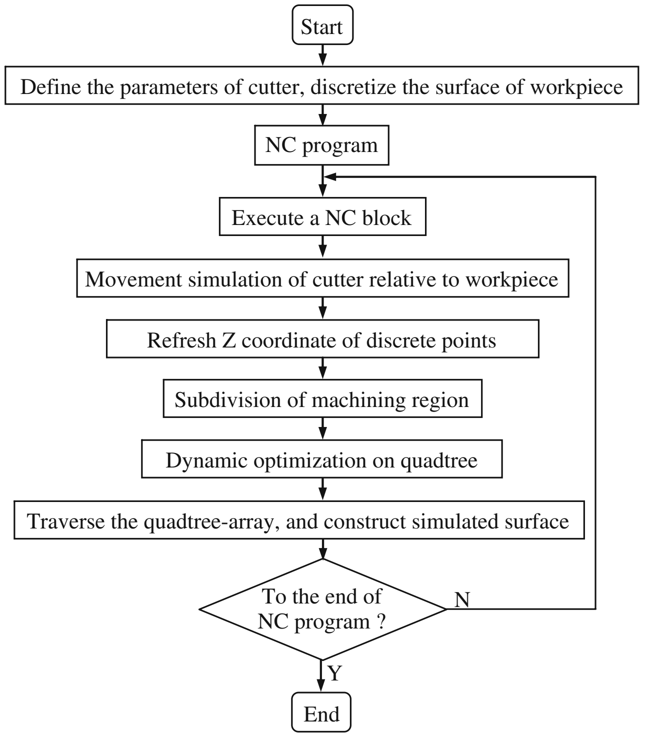
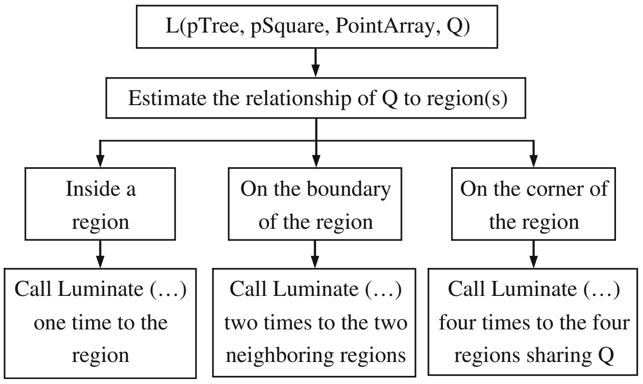

Start
Define the parameters of cutter, discretize the surface of workpiece
NC program
Execute a NC block
Movement simulation of cutter relative to workpiece
Refresh Z coordinate of discrete points
Subdivision of machining region
Dynamic optimization on quadtree
Traverse the quadtree-array, and construct simulated surface
To the end of NC program ?
N
Y
End

Fig. 11 Flow chart of milling simulation algorithm



L(pTree, pSquare, PointArray, Q)
Estimate the relationship of Q to region(s)
Inside a region
On the boundary of the region
On the corner of the region
Call Luminate (…) one time to the region
Call Luminate (…) two times to the two neighboring regions
Call Luminate (…) four times to the four regions sharing Q

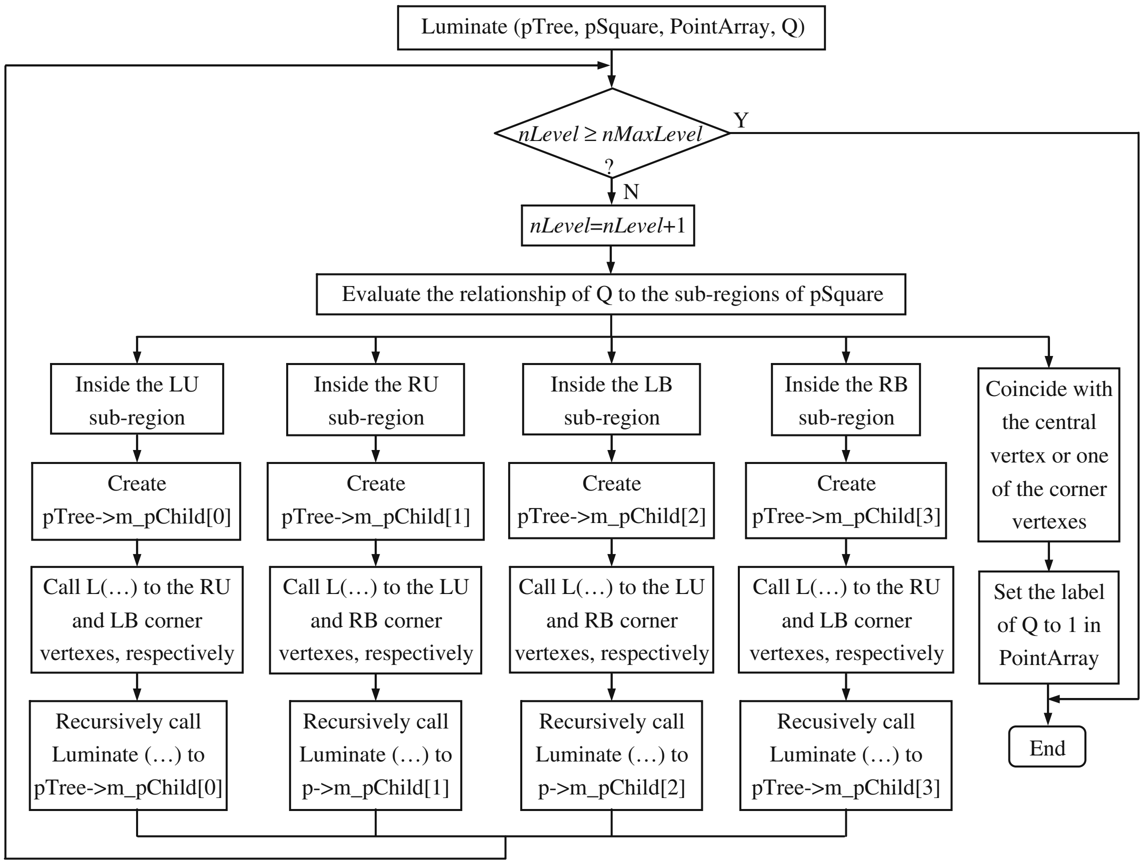
Fig. 12 Flow chart of function L(...)

On the data structure level, optimization process is carried out by accessing the nodes of the list from the tail to the head in an order of their sequential numbers. The sequence number of each list node is same to that of the corresponding quadtree node. Based on the list, optimizations on leaf node and intermediate node are carried out, respectively. Figure 9 shows the flow chart of dynamic optimization on quadtree structure.

Figure 10 shows a controlled comparison example of a quadtree-array represented cosine surface without and with dynamic optimization, respectively. For the represented surface without dynamic optimization, the total number of nodes is 21,845, and the meshes are very dense. It results in screen flicker when animation is performed. Contrasting to it, the total number of the quadtree nodes is reduced to 4,168 when the optimization is applied. The screen is refreshed smoothly in animation. A conclusion can be drawn that dynamic optimization on quadtree can reduce the data amount in simulation significantly. It is effective to improve simulation efficiency.

5 Implementation of milling process simulation based on quadtree-array

Machining process simulation is realized via Boolean subtraction operation of the cutter swept volume from the workpiece model driven by an NC program. Figure 11 shows the flow chart of algorithm of three-axis milling process simulation based on quadtree-array. In the algorithm, subdivision of represented region is the key module. The



Luminate (pTree, pSquare, PointArray, Q)
nLevel ≥ nMaxLevel ?
Y
N
nLevel=nLevel+1
Evaluate the relationship of Q to the sub-regions of pSquare
Inside the LU sub-region
Inside the RU sub-region
Inside the LB sub-region
Inside the RB sub-region
Coincide with the central vertex or one of the corner vertexes
Create pTree->m_pChild[0]
Create pTree->m_pChild[1]
Create pTree->m_pChild[2]
Create pTree->m_pChild[3]
Call L(…) to the RU and LB corner vertexes, respectively
Call L(…) to the LU and RB corner vertexes, respectively
Call L(…) to the LU and RB corner vertexes, respectively
Call L(…) to the RU and LB corner vertexes, respectively
Set the label of Q to 1 in PointArray
Recursively call Luminate (…) to pTree->m_pChild[0]
Recursively call Luminate (…) to p->m_pChild[1]
Recursively call Luminate (…) to p->m_pChild[2]
Recusively call Luminate (…) to pTree->m_pChild[3]
End

(LU: left-up RU: right-up LB: left-bottom RB: right-bottom)

Fig. 13 Flow chart of function Luminate(...)

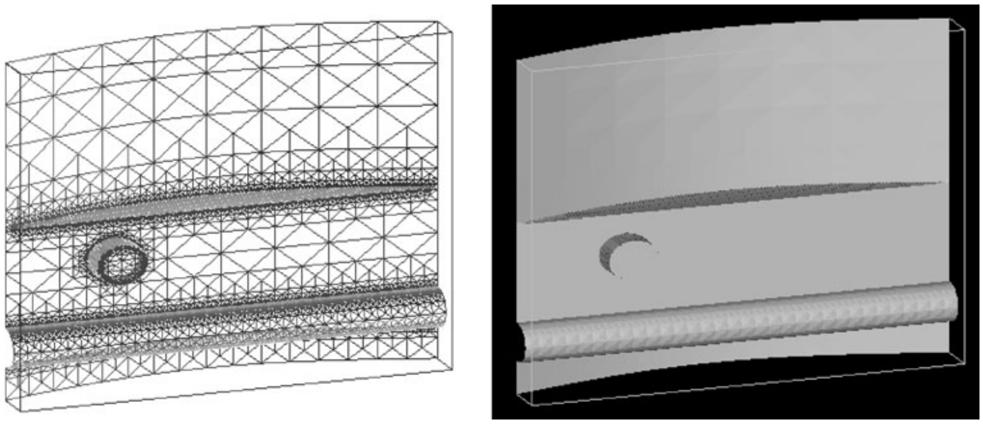


Fig. 14 Simulated surface of experiment 1

Table 1 Simulation results of experiment 1

Division number of array cells	Maximum number of quadtree levels	Total number of nodes	Total number of facets	Simulation time (s)	Frame rate (frames/s)		
					Minimum	Maximum	Average
1×1	9	6,605	28,024	3,310.55	1	39	1
2×2	8	6,604	28,024	929.19	2	63	3
4×4	7	6,660	28,024	237.28	7	62	9
8×8	6	6,660	28,068	75.50	24	62	29
16×16	5	6,656	28,408	54.84	47	63	51
32×32	4	7,108	30,462	40.83	49	63	54
64×64	3	9,294	39,944	49.50	40	62	44

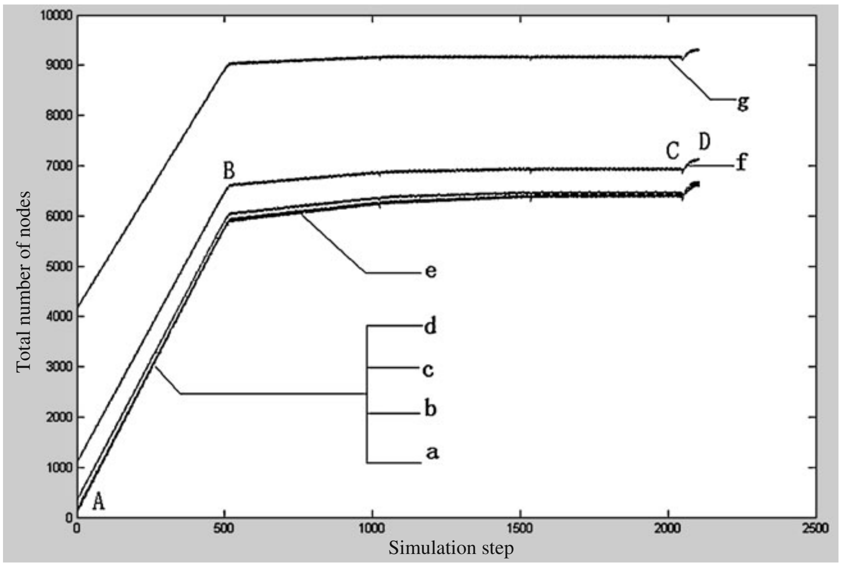


Fig. 15 Relationship of the total number of nodes to simulation step in experiment 1

(a: 1*1 b: 2*2 c: 4*4 d: 8*8 e: 16*16 f: 32*32 g: 64*64)

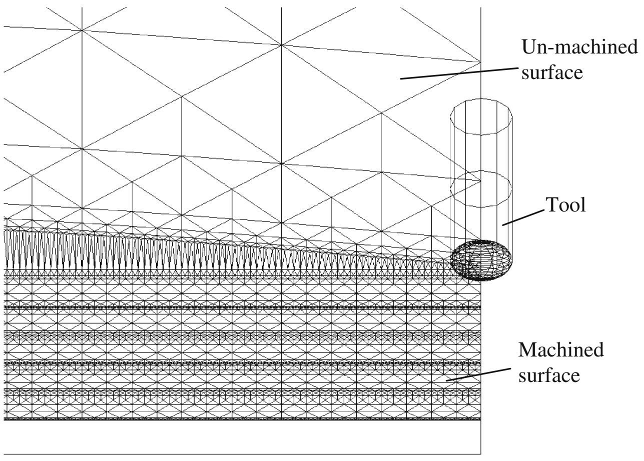


Fig. 16 Simulated surface in experiment 2

module is to determine whether subdivision and creation of new nodes should be carried out or not based on the estimation of the relationship between a discrete point and a quadtree node. The function of the module is implemented by two functions which called each other. In this paper, the two functions are expressed as Luminate(...) and L(...) for example, respectively.

Function L(...) is to determine whether a region needed to be subdivided according to the position of a discrete point related to the region. Function Luminate(...) is mainly to subdivide the region and create new quadtree nodes. To avoid cracks in construction of the simulated surface, other neighboring square regions should also be subdivided till the accuracy which the cracks can be eliminated is reached. The maximum levels of a quadtree, named nMaxLevel, are determined by the division number of array cells and simulation accuracy. The subdivision is stopped when the levels of the quadtree equals to nMaxLevel. Figures 12 and 13 show the flow chats of function L(...) and function Luminate(...), respectively. In the figures, pTree points to the current node of a quadtree, pSquare to the square region to be subdivided, PointArray is the array of discrete points, and Q is a discrete point to be evaluated. Before the subdivision begins, pTree points to the root node of a quadtree.

6 Performance evaluation of quadtree-array

A three-axis milling process simulation system based on quadtree-array representation was developed by using Microsoft Visual C++ 6.0. Graphic display is supported by OpenGL. Dynamic optimization on quadtree is also added to the system. The development and performance demonstrations of the system were conducted on a personal computer with a configuration of AMD Athlon 64 X2 Dual Core Processor 4000+ 2.11 GHz, memory 1.0 GB, and NVIDIA GPU GeForce 8500GT with 128bit/512 MB. The performance evaluations of quadtree-array were conducted through three milling simulation experiments on surfaces with different complexity and different division number of array cells.

6.1 Experiment 1

In this experiment, the dimension of the cylindrical surface is 512×512 mm in XY plane, simulation accuracy is 1.0 mm, and modeling error of cutter is 0.06 mm. The experiment is to simulate three-axis rough machining process. The surface with features such as ball slot, plane, and hole will be machined with a ball-end cutter, a flat-end cutter, and a slot cutter, respectively. Figure 14 shows the simulated surface, and Fig. 15 shows the relationship of total number of quadtree-array nodes to simulation step with different division number of array cells. The experimental results are tabulated in Table 1.

As shown in Fig. 15, each curve can be divided into three segments, i.e., AB, BC, and CD. Segment AB corresponds to the machining process simulation of slot feature with the ball-end cutter. In this process, the feature detail of the ball slot needs a great deal of small triangular facets to represent. Therefore, the total numbers of nodes and triangular facets are increased linearly as the simulation runs. Segment BC corresponds to the machining process simulation of the planar feature machined with the flat-end cutter. In this process, optimization on quadtree plays an important role. However, as new nodes with higher resolution are inserted to the quadtree in simulation, a lot

Table 2 Simulation results of experiment 2

Division number of array cells	Maximum number of quadtree levels	Total number of nodes	Total number of facets	Simulation time (s)	Frame rate (frames/s)		
					Minimum	Maximum	Average
1×1	9	21,813	91,792	9,565.37	2	62	3
8×8	6	21,792	91,792	648.64	13	63	31
16×16	5	21,728	91,792	585.34	16	62	34
32×32	4	21,505	91,875	587.88	16	62	34
64×64	3	20,737	92,612	616.45	13	61	31

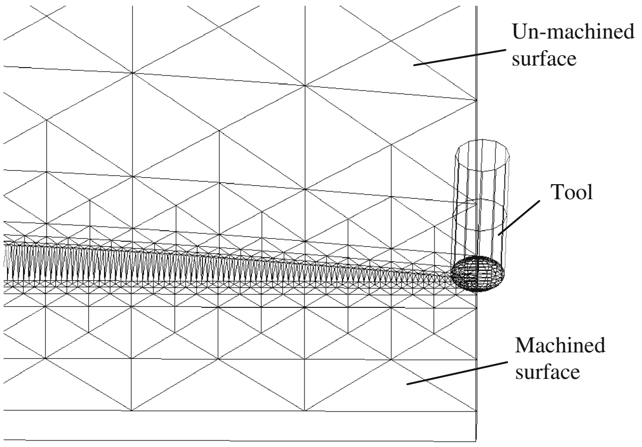


Fig. 17 Simulated surface in experiment 3

of nodes are been merged and deleted simultaneously in optimization. As a result, the total number of nodes tends to be constant. Section CD corresponds to the machining process simulation of hole feature with a drill. The detail of the hole feature should be represented by a lot of triangular facets. The number of nodes is increased.

From Table 1, the bigger the division number of array cells is, the smaller the number of quadtree levels will be. When the division number is quite small, the path to access the data of leaf nodes of a quadtree level by level is prolongated, which leads to a great amount of intermediate data and computation of subdivision in simulation, and simulation efficiency is decreased. The path to access the data of leaf nodes of a quadtree is shortened when the division number is increased so that the amount of data and computation of subdivision is reduced and simulation efficiency is improved remarkably. However, the total number of nodes will be increased as the division number is increased over a reasonable value. The reason is that the effect of optimization on node mergence is weakened in simulation.

6.2 Experiment 2

This experiment aims to evaluate the performance of quadtree-array to represent the complicated surface in milling process simulation. In the experiment, a surface is "machined" with a ball-end cutter along zigzag tool-paths. The diameter of the ball-end cutter is 20 mm, and the step-over is 10 mm. Figure 16 shows the simulated surface and the experimental results are listed in Table 2.

From Table 2, different division number of array cells has little effect on the total number of quadtree nodes, but much on the simulation efficiency. The simulation efficiency is improved as the division number of array cells is increased. In the experiment, the total number of quadtree-array nodes keeps nearly constant because the total number of leaf nodes, which represents the details of the simulated surface, is prominent over that of other nodes. The simulation efficiency is better when the division number of array cells is set to 16× 16 or 32×32. The same conclusion can be drawn in experiment 1.

6.3 Experiment 3

This experiment is to evaluate the quadtree-array representation in simulation of milling planar surface with a flat-end cutter. Figure 17 shows the simulated surface, and the simulation results are listed in Table 3.

From Table 3, different division number of array cells has great influence on the total number of quadtree nodes and the simulation efficiency. As the division number of array cells is increased, the total number of nodes is also increased. It is because the increased array cells become an obstacle to the node emerging in optimization. However, the simulation efficiency is also improved due to the reduction of the levels as the increase on the division number of array cells, but there is a reasonable value. In this experiment, better simulation performance is also achieved when the division number of array cells is set to 16×16 or 32×32.

7 Conclusions

The representation of workpiece in machining process simulation plays a more important role in simulation

Table 3 Simulation results of experiment 3

Division number of array cells	Maximum number of quadtree levels	Total number of nodes		Simulation time (s)	Frame rate (frames/s)		
		Minimum	Maximum		Minimum	Maximum	Average
1×1	9	86	948	6,352.33	2	62	3
8×8	6	90	952	233.11	60	99	71
16×16	5	280	1,094	202.70	60	164	97
32×32	4	1,045	1,764	212.14	60	132	86
64×64	3	4,113	4,640	259.42	19	70	62

accuracy and simulation efficiency. Up to now, it has still been highlighted. To improve geometric simulation efficiency of three-axis milling process, a novel workpiece representation based on quadtree-array is presented and discussed in detail in this paper. Several conclusions are drawn from the evaluation experiments and listed as follows:

1. Compared to single quadtree, workpiece representation based on quadtree-array in machining process simulation can improve simulation efficiency significantly.
2. The effect of dynamic optimization of quadtree structure on simulation efficiency is obvious in simulation.
3. There are several reasonable division numbers of array cells which will lead to better simulation performance. The recommended values are 16×16 and 32×32 according to the experimental results.

Acknowledgements The authors deeply thank the 111 Project (B07018), China, for financial support.

References

1. Karunakaran KP, Shringi R (2008) A solid model-based off-line adaptive controller for feed rate scheduling for milling process. J Mater Process Technol 204(1–3):384–396
2. Hook TV (1986) Real-time shaded NC milling display. Comput Graph 20(4):15–20
3. Kedem G, Ellis JL, Bartsch KE, Subrahmanyan R (1986) Custom ICS for the ray-casting machine. In: Proceedings of the IEEE 1986 Custom Integrated Circuits Conference. Rochester, NY, USA, pp 182–185
4. Yang JZ, Karim AM (2006) Verification of NC machining processes using swept volumes. Int J Adv Manuf Technol 28(1–2):82–91
5. Sungurtekin UA, Velcker HB (1986) Graphical simulation and automatic verification of NC machining programs. In: Proceedings of 1986 IEEE International Conference on Robotics and Automation. San Francisco, CA, USA, pp 156–165
6. Fleisig RV, Spence AD (2005) Techniques for accelerating B-rep based parallel machining simulation. Comput Aided Des 37:1229–1240
7. Chappel IT (1983) The use of vectors to simulate material removed by numerically controlled milling. Comput Aided Des 15(3):156–158
8. Hsu PH, Yang WT (1993) Real-time 3D simulation of 3-axis milling using isometric projection. Comput Aided Des 25(4):215–224
9. Wastra WH, Bronsvoort WF, Vergeest JSM (2005) Interactive simulation of robot milling for rapid shape prototyping. Comput Graph 18(6):861–871
10. Navazo I, Ayana AD, Brunet PA (1986) Geometric modeler based on the exact octree representation of polyhedral. Comput Graph Forum 5(2):91–104
11. Liu CL, Esterling DM, Fontdecaba J, Mosel E (1996) Dimensional verification of NC machining profiles using extended quadtrees. Comput Aided Des 28:845–852
12. Liu SQ, Ong SK, Chen YP, Nee AYC (2006) Real-time, dynamic level-of-detail management for three-axis NC milling simulation. Comput Aided Des 38:378–391
13. Roy U, Xu YX (1998) 3-D object decomposition with extended octree model and its application in geometric simulation of NC machining. Robot Comput-Integr Manuf 14:317–327
14. Li JG, Yao YX, Xia PJ, Liu CQ, Wu CG (2008) Extended octree for cutting force prediction. Int J Adv Manuf Tech 39(9–10):866–873
15. Wang WJ, Wang TY, Fan SB, Wang WY (2008) Research on material removal algorithm model in virtual milling process based on adaptive dynamic quadtrees algorithm. Mech Mater 10–12:822–827
16. He W, Bin HZ (2007) Simulation model for CNC machining of sculptured surface allowing different levels of detail. Int J Adv Manuf Technol 33(11–12):1173–1179
17. Karunakaran KP, Shringi R (2007) Octree-to-BRep conversion for volumetric NC simulation. Int J Adv Manuf Technol 32(1–2):116–131