International Journal of CAD/CAM *Vol. 4, No. 1, pp. 47~53 (2004)*

INTERNATIONAL JOURNAL OF CAD/CAM
www.ijcc.org

Parallel Generation of NC Tool Paths for Subdivision Surfaces

Junfu Dai, Huawei Wang* and Kaihuai Qin
Dept. of Computer Science & Technology, Tsinghua University, Beijing 100084, China

Abstract – The subdivision surface is the limit of recursively refined polyhedral mesh. It is quite intuitive that the multi-resolution feature can be utilized to simplify generation of NC (Numerical Control) tool paths for rough machining. In this paper, a new method of parallel NC tool path generation for subdivision surfaces is presented. The basic idea of the method includes two steps: first, extending G-Buffer to a strip buffer (called S-Buffer) by dividing the working area into strips to generate NC tool paths for objects of large size; second, generating NC tool paths by parallel implementation of S-Buffer based on MPI (Message Passing Interface). Moreover, the recursion depth of the surface can be estimated for a user-specified error tolerance, so we substitute the polyhedral mesh for the limit surface during rough machining. Furthermore, we exploit the locality of S-Buffer and develop a dynamic division and load-balanced strategy to effectively parallelize S-Buffer.

***Keywords*: Subdivision surface, NC tool path, S-Buffer**

1. Introduction

Free-form surfaces are standard in CAD/CAM systems, which are widely used in designs of electronic appliances, automobiles, and airplanes. Because control polyhedra of NURBS or B-spline surfaces are restricted to regular meshes, they can represent only limit surfaces of rectangular topology. Subdivision surfaces have emerged recently as an attractive technique in modeling surfaces of arbitrary topology [1, 2, 3, 4, 6]. Currently, there are numerous articles on NC machining of free-form surfaces, but few of them are focused on machining subdivision surfaces though subdivision surfaces have strong powers of modeling complicated surfaces.

In practice, a complete NC milling process usually consists of rough machining and fine machining. The main goal of fine machining is for accuracy of the resulted workpiece, while the most important target of rough machining is to reduce the machining time so as to improve efficiency. Most of existing NC tool path generation methods derive rough tool paths from fine ones. Some do address the special topic of rough machining and thus develop useful techniques for different applications. But it is difficult for them to be integrated with the methods used in fine machining.

Motivated by these problems, this paper takes full use of multi-resolution feature of a subdivision surface and the distance bound of the polyhedral mesh to its limit surface [7], and substitutes the polyhedral mesh within a user-specified tolerance for the limit surface during rough machining to make the rough paths simple. A new method called S-Buffer is presented by improving G-Buffer method [5] to seamlessly unify the rough and fine path generation, and a parallel method of S-Buffer is implemented using MPI.

*Corresponding author:
Tel: +86 (10) 6278 5592
Fax: +86 (10) 6277 1138
E-mail: whw9418@yahoo.com.cn

2. NC Tool Path Generation of Subdivision Surfaces

2.1. Subdivision surfaces

In CAD/CAM systems, NURBS, B-splines are widely used in free form surface modeling, but they can hardly represent complex shapes of arbitrary topology with just one surface. Subdivision surfaces seem to be more attractive in many fields by generalizing B-spline surfaces to meshes of arbitrary topologies.

2.1.1. Catmull-Clark surfaces

A Catmull-Clark surface is defined as the limit of a sequence of vertices of finer and finer controlled polyhedron generated in such a way that at each step the old vertices are updated and new vertices are introduced according to subdivision rules.

Assume $\boldsymbol{P}_0^n$ is a vertex with valence N after n times of subdivision. Other $2N$ vertices around $\boldsymbol{P}_0^n$ are labeled as shown in Fig. 1. In Catmull-Clark subdivision, the new vertices are computed as follows [1]:

$$\boldsymbol{P}_{2i}^{n+1}=\frac{1}{4}(\boldsymbol{P}_0^n+\boldsymbol{P}_{2i-1}^n+\boldsymbol{P}_{2i}^n+\boldsymbol{P}_{2(i\%\mathrm{N})+1}^n)$$

$$\boldsymbol{P}_{2i-1}^{n+1}=\frac{3}{8}(\boldsymbol{P}_0^n+\boldsymbol{P}_{2i-1}^n)$$

$$+\frac{1}{16}(\boldsymbol{P}_{2i}^n+\boldsymbol{P}_{2(i\%\mathrm{N})+1}^n+\boldsymbol{P}_{2(i-2\%N)+2}^n+\boldsymbol{P}_{2(i-2\%N)+1}^n)$$

$i=1, 2, \ldots, N$, where "%" stands for the remainder, and

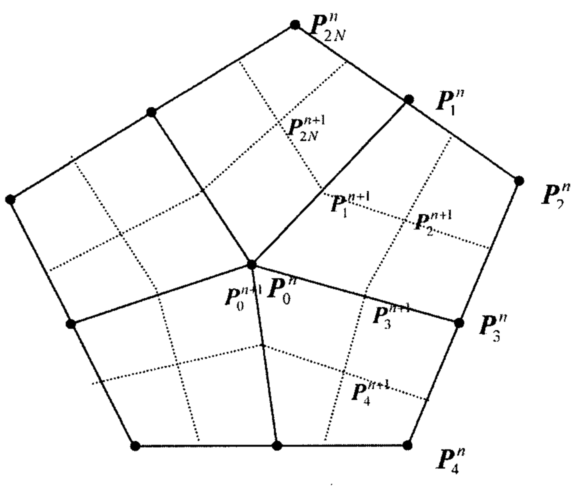


Fig. 1. Catmull-Clark subdivision.

$$P_0^{n+1}=\alpha_N P_0^n+\beta_N\left(\frac{1}{N}\sum_{j=1}^{N}P_{2j-1}^n\right)+\gamma_N\left(\frac{1}{N}\sum_{j=1}^{N}P_{2j}^n\right)$$

where α_N, β_N, $\gamma_N \geq 0$, $\alpha_N+\beta_N+\gamma_N=1$. For example,

$$\alpha_N=1-7/(4N),\ \beta_N=3/(2N),\ \gamma_N=1/(4N) \tag{1}$$

Let $C_n=(P_0^n, P_1^n, ..., P_{2N}^n)^T$, then $C_{n+1}=AC_n$, where A is the subdivision matrix [7].

2.1.2. Estimating error between a polyhedral mesh and its limit surface

After certain steps of subdivision, the polyhedral mesh becomes more and more approximate to the smooth limit surface. In order to ensure that the machined surfaces are accurate enough, it is very important to compute the depth of recursion within a user-specified error bound.

Define S_0^n as the set of control vertices created from certain initial vertices P_0^0 after n steps of recursive subdivision, $L(P)$ as the corresponding point in the limit surface for a control vertex P. Let Circle(P_0^n)= {P_0^n, P_1^n, ..., P_{2N}^n} denote the set of the vertices in the local structure of P_0^n, $C(P_0^n)=\frac{1}{2N}\sum_{j=1}^{2N}P_j^n$, and $abs(F)=(|f_{ij}|)$ for a matrix $F=(f_{ij})$. Then the distance of the control polyhedron to the limit surface after n steps of Catmull-Clark subdivision has the following bound [7]:

$$\max_{P\in S_0^n}|P-L(P)|_1 \leq \rho^n K \max_{P\in S^0}\max_{Q\in \mathrm{Circle}(P)}|C(P)-Q|_1 \tag{2}$$

where n is the depth of recursive subdivision,

$$K=\max_{3\leq N\leq M}\left\| U\cdot abs(V)diag\left(0,\left|\frac{\lambda_2}{\lambda_2}\right|^p,\ldots,\left|\frac{\lambda_{2N+1}}{\lambda_2}\right|^p\right)abs(V^{-1})\right\|_\infty$$

$$\rho=\max_{3\leq N\leq M}|\lambda_2|,\ U=diag(1,0,0,...,0),\ S^0=S_0^0$$

λ_i is the i-th eigenvalue of the subdivision matrix A, V is an invertible matrix whose columns are the corresponding eigenvectors of A, and p is a constant which is selected in advance as long as the resulting subdivision depth is not less than it.

2.2. Generating tool paths for subdivision surfaces

Tool path generation is the core task of an NC Machining system. A difficult task of path planning is detection and avoidance of interference and collision. Besides, it is necessary to take path verification and feed rate control into account during path generation.

Precision and efficiency are the two main considerations of NC machining. To reduce machining time without affecting precision, the rough machining should be treated specially, such as simplifying tool paths according to the shape of the original workpiece, slicing multilayer paths, and so on. Meanwhile these techniques should be seamlessly integrated with those used in the tool path generation for fine machining.

It is well known that subdivision surfaces are defined as the limit of recursively refined polyhedral meshes. The degree of approximation of a control mesh to the limit surface after any step during the subdivision process can be calculated [7]. So it is natural to make use of this multi-resolution feature to simplify the tool paths of rough machining to improve the efficiency. That is, we choose the control polyhedral meshes after n steps of subdivision as the substitution for the limit surface during the tool path computation of rough machining. The number n is dependent on the cutting amount required for rough cutting and can be easily estimated according to Eq. (2).

There are two basic considerations in generating tool paths for subdivision surfaces:

(1) Take the control mesh after enough steps of subdivision as a rough-machining workpiece;
(2) Adopt a unified method for computation of both rough tool paths (from polyhedral meshes) and fine tool paths (from smooth limit surfaces).

2.2.1. S-Buffer

G-Buffer is a very attractive method for NC tool path generation [5]. Based on the Z-Buffer method in computer graphics, G-Buffer can be used with any surfaces, either analytical, free-form, trimmed surfaces, or polyhedral faces.

Fig. 2 shows how to obtain tool path buffer from a workpiece buffer. Let Z and L be the buffers of the workpiece and the desired tool path, respectively, and $h(d)$ be the height of the tool-end at the distance d from the tool axis. When the tool-end touches the workpiece at (x_t, y_t), we have

$$L(x,y)=Z(x_t, y_t)-h(x-x_t, y-y_t)$$

Therefore, it is easy to get L as follows:

$$L(x, y) = \max_{i,j}(Z(x+i,\ y+j) - h(i,j)),\ (i^2+j^2<r^2) \quad (3)$$

where $h(i, j)$ depends on the shape of tool-end and r is the radius.

Now we can obtain different kinds of paths with different scanning strategies.

It is shown in Fig. 2 that G-Buffer can not only deal with different shapes of tools, but also create interference-free tool paths. It is also very easy to implement G-Buffer. In addition, it is robust to generate NC tool paths by G-Buffer. However, the requirement of memory is very large with G-Buffer. If we want to machine a workpiece with 1-meter long and 1-meter wide with

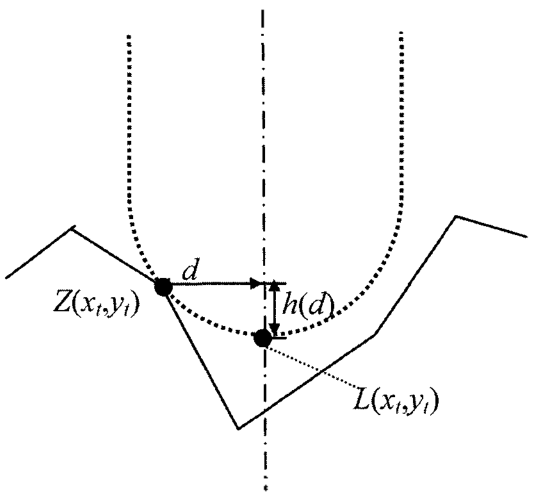


Fig. 2. Touch point of tool and wokpiece.

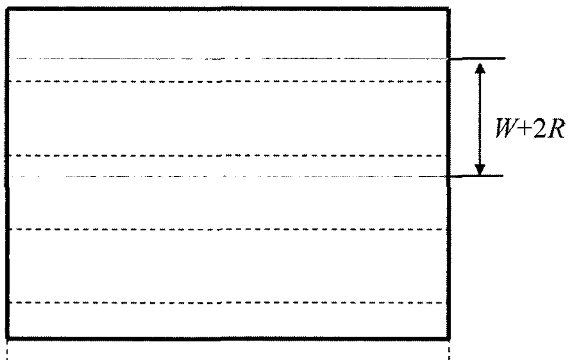


Fig. 3. Dividing working area into strips.

the error bound of 0.1 millimeters, and we only record the height field and the normal at each pixel in float format (32 bits), then 1.6G Bytes are needed! This is beyond the maximum size assigned for a single thread in Windows NT/2000.

Based on G-Buffer, we divide the working area into strips and develop a new method, called Strip Buffer (*i.e.*, S-Buffer), as follows:

○ Dividing the working area

As illustrated in Fig. 3, we divide the whole area into many strips with the width of W. According to (3), we need to include two adjacent R-wide regions in adjacent strips, where $R = 2r$. So the width of the strip is actually $H = W+2R$. The value of constant W depends on the system configuration. In general, we assign an appropriate value to make full use of the physical memory and to reduce the data swapped between the memory and the hard disk of the computer system.

○ Generating NC tool paths with S-Buffer

For each strip, we create two buffers of both the workpiece and tool paths, then compute like G-Buffer to obtain tool paths of the strip.

○ Simplifying the tool paths

The resulted paths from the above steps are discretized on pixels. It is indispensable to fit the tool paths with longer line segments within the user-specified tolerance to improve machining efficiency, especially during rough cutting process. A scheme for the line fitting is presented as follows:

- First of all, define the joint points of the line segments of the final paths as "key points", and initially take the start point, end point and the extremum points in the tool path buffer as key points. Connect the adjacent key points to form a poly-line.
- Let S_i denote the set of $n=n(i)$ pixels ($\boldsymbol{P}_0$, $\boldsymbol{P}_1$, ..., $\boldsymbol{P}_{n-1}$) between the i-th pair of key points. Check the distance of each point of S_i to the line defined by the key point pair. If the distance is less than the user-specified tolerance, check next pair of the adjacent key points. Otherwise, break the line into two segments at $\boldsymbol{P}_{(n-1)/2}$, and check these two shorter line segments again (see Fig. 4). The joint point $\boldsymbol{P}_{(n-1)/2}$ is regarded as a new key point added. This "check-and-break" process is recursively done until the distance of each point to the corresponding

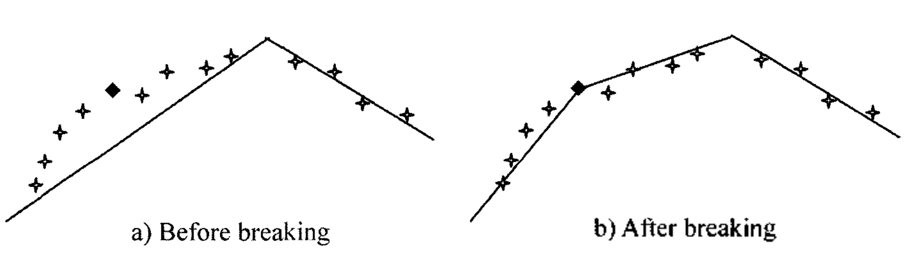


Fig. 4. Dividing path.

line defined by the two key points is less than the given tolerance.

○ Combining all tool paths into a complete path

After obtaining all strips paths, we combine them into a complete NC tool path by zigzag or spiral scanning.

3. Parallelizing S-Buffer

3.1. Parallel computing environment

With the rapid development of network technology and the ever-growing performances of PCs, clusters and other parallel architectures make it possible to constitute high performance computing environments with PCs connected by network. MPI is the standard for message passing among multi-computers and cluster. Using MPI, components of the virtual machine communicate and cooperate with each other by passing messages.

3.2. Program architecture

There are two kinds of patterns in MPI programming: peer-to-peer pattern and master/slave pattern. It is obvious that the master/slave pattern is more suitable for parallelizing S-Buffer. Master process runs on a local host, integrated in CAD/CAM systems, providing a visual interface to users. Slaves distribute on nodes connected to the local host, performing computation assigned by the master. In this architecture, the master's duty is to distribute tasks, control the slaves and generate the global tool paths after receiving results of all strips.

Parallelizing S-Buffer consists of three main phases: (1) the master calculates the subdivision depth, generates the corresponding polyhedral mesh and sends it to the slaves; (2) the slaves compute tool paths of the strips separately and return results to the master; (3) the master generates the global path.

3.3. Data division and parallel granularity

Data division strategy and parallel granularity have a great effect on speedup, which is the main index of the performance of a parallel algorithm. As described in 2.2, S-Buffer method divides working area into strips with the width of W, and introduces redundant computing of $2R$-wide adjacent region (for non-margin strips) for each strip. If W is too small, then the redundancy and communication loads will affect the parallel speedup. On the contrary, if we assign a too big value to W, then the degree of parallelism is not high enough, and thus the speedup decreases. Therefore, the value of M is dependent on the total working load, the number of nodes of the virtual machine and their computational capabilities, and the communication capability of the network connecting virtual machine.

As will be discussed in Section 3.4, assigning tasks of stationary size to different slaves is not appropriate in a distributed environment. Although we divide the whole working area into strips of the same size to make the task assignment strategy simple, the number of strips assigned to certain slave should be dependent on its capability. Thus, our data division strategy is adaptive and can contribute to a high speedup.

3.4. Performance index and dynamic task assignment

Subtask assignment and load balance strategy is a dominating factor of speedup. In certain circumstances where the performances of nodes and network are stable, we can assign the tasks statically. But in most cases, the virtual machine runs under a dynamic environment. The performances of the nodes and network vary from time to time, so it is necessary to take a dynamic strategy for the task assignment and load balancing.

To achieve a high speedup, the load assigned to a node should be decided by its capability. In the parallelization of S-Buffer, there is no need to communicate among slaves. So the computing performance of a node and the communication capability of the network can be comprehensively indicated by the response time, which is the time interval on the master from sending a slave commands to receiving results. Besides, the performance of a node varies dynamically. Therefore, the performance index of a node should reflect this change adaptively. We define weighted-average response time T_i (*i.e.*, the index of performance) as follows:

When receiving a result of a task from the i-th slave, the master refreshes T_i and calculates M_i, which is the number of strips that should be assigned to the slave this time. Let N_i be the total number of strips that have been dealt with by slave i, n_i the number of strips in this task, and t_i the response time of this task. Then,

$$T_i=\frac{T_i\times N_i}{N_i+n_i}+\frac{t_i}{n_i} \text{ and } N_i=N_i+n_i \text{ (initially } T_i=N_i=0);$$

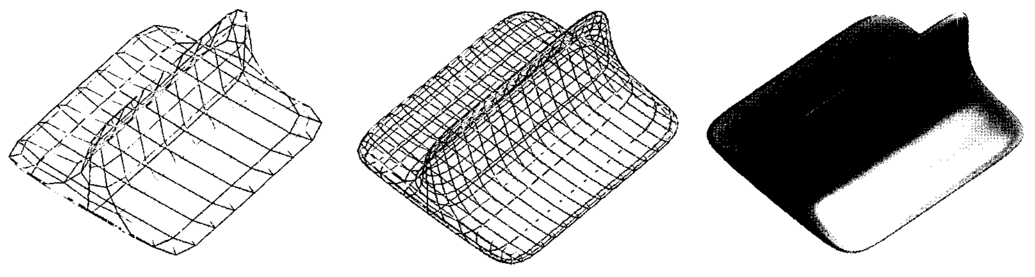
where N is the number of the nodes whose $N_i > 0$.

Our task assignment strategy is also preemptive. Let C be the number of the strips that have not been dealt with, E_u the time that has elapsed since the u-th strip was assigned, and a constant $k(0<k<1)$ "preemptive coefficient". Define "dealing strips" as strips that certain slaves are dealing with. When $M_i>C$, the master assigns all the C strips to slave i, then traverses all the "dealing strips". For each dealing strip (marked here by strip u), with which some slave (say, slave j) is dealing, if $T_i<k*(T_j-E_u)$, then the master deprives slave j of strip u and assign it to slave i, until the number of strips that have been assigned to slave i is equal to M_i.

4. Examples

The S-Buffer method has been implemented on a local 10M ethernet network of seven PCs. The software platform consists of Windows 2000 and the corresponding WMPI v1.5.

Fig. 5 and 6 illustrate the rough tool paths of a T-


a) Initial mesh b) Divided mesh c) Realistic image

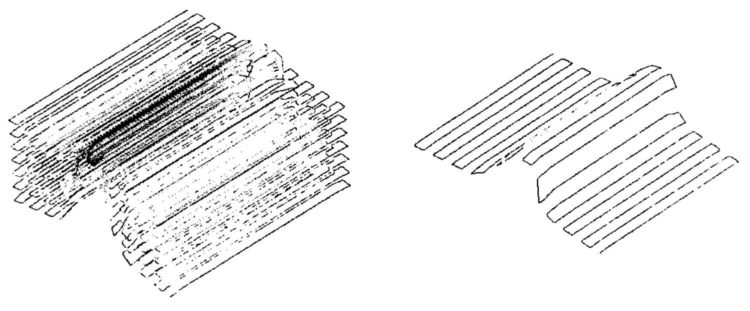
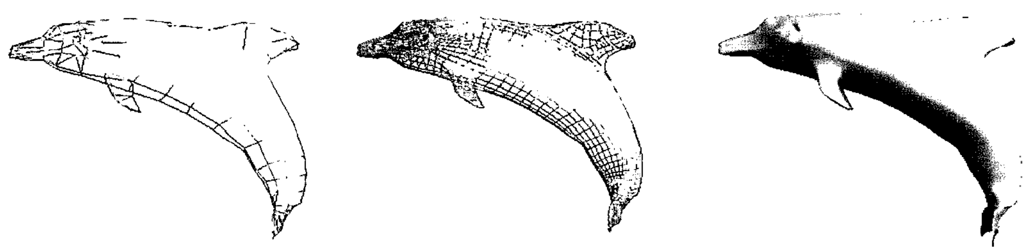

d) Tool paths of 7 layers e) Single path

Fig. 5. Machining a T-shaped object.


a) Initial mesh b) Divided mesh c) Realistic image

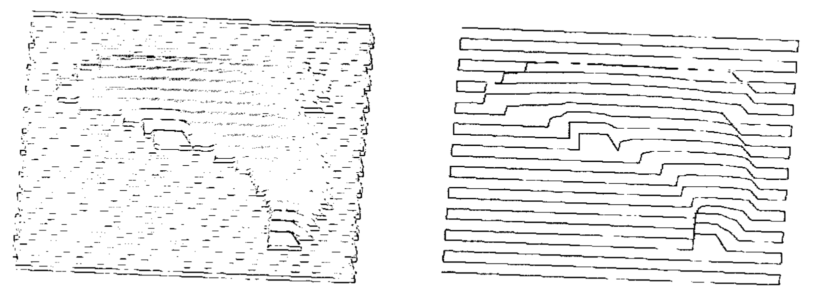

d) Multilayer paths e) Single path

Fig. 6. Machining a dolphin.

shaped object and a dolphin model, respectively.

We have tested the parallelized S-Buffer method with data as follows: The working area is 0.952-meter long and 1.99-meter wide, using a spherical milling tool with a radius of 8 millimeters, and the size of a pixel is 0.5 millimeters. Table 1 demonstrates the time costs for different *H*, and Fig. 7 illustrates the speedups for different numbers of nodes. It is shown in Table 1 that the value of *H* affects the time costs greatly, and the reason is that the ratio of the redundant computing time of all *R*-wide adjacent regions to the total computing time is highly dependent on the value of *H*. However, as shown in Fig. 7, the speedup is always equal to the number of nodes approximately for any *H*, because the

Table 1. Time costs (second) with different H

	1	2	3	4	5	6	7
H=100	671.35	336.11	224.48	168.64	134.95	112.56	96.56
H=200	414.73	207.06	138.32	103.95	83.25	69.42	59.68

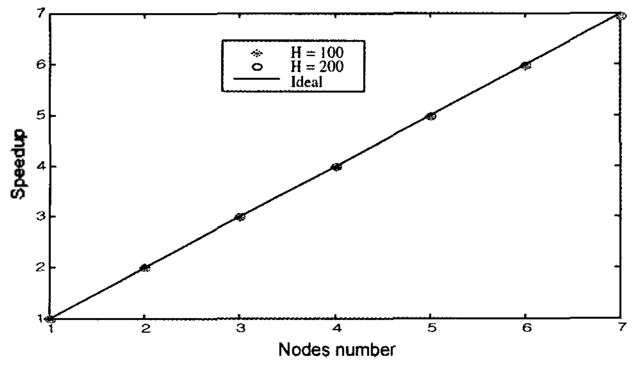


Fig. 7. Scalability of the virtual machine.

computation of S-Buffer is a time-consuming process (more than 400 seconds) so that the time of communication looks very tiny and accordingly can be ignored in comparison with the computing time. Therefore, the scalability of the parallel S-Buffer method is very satisfying.

5. Conclusions

Subdivision surfaces are widely used in computer aided geometric design and modeling. We can substitute the polyhedral meshes of subdivision for the smooth limit surfaces during tool path generation for rough NC machining. The S-Buffer method presented can combine the rough and the fine machining seamlessly and generate NC tool paths for workpieces of large size in a parallel computer system or even on a single PC. In contrast to G-Buffer, S-Buffer does not handle the whole working area but divides the working area into strips, and accordingly avoids the demand for too huge memory. On the other hand, S-Buffer is very suitable to be implemented in parallel system, so the computing time is reduced greatly. In addition, our method also exploits advantages of subdivision surfaces to speedup the process for generating NC tool paths. It is shown that the parallel S-Buffer method is very efficient, scalable and robust.

Acknowledgements

This research was supported by the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20010003048) and the National Natural Science Foundation of China (No. 60273013).

References

[1] Catmull, E. and Clark, J. (1978), Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design*, **10**, 350-355.

[2] Qin, K. and Wang, H. (1999), Eigenanalysis and continuity of non-uniform Doo-Sabin surfaces, Proceedings of Pacific Graphics'99, SNU, Korea, Oct. 1999, 179-196.

[3] Qin, K. and Wang, H. (2000), Continuity of non-uniform recursive subdivision surfaces, *Science in China (Series E)*, **43**(5), 461-472.

[4] Reif, U. (1995), A unified approach to subdivision algorithms near extraordinary vertices, *Computer Aided Geometric Design*, **12**, 153-174.

[5] Saito, T. and Takahashi, T. (1991), NC machining with G-buffer method, *Computer Graphics*, **25**(4).

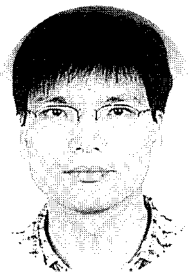
[6] Stam, J. (1998), Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values, *Computer Graphics (Proceedings of SIGGRAPH' 98)*.

[7] Wang, H. and Qin, K. (2004), Estimating subdivision depth of Catmull-Clark Surfaces, *Journal of Computer Science and Technology*, **19**(5), 657-664.

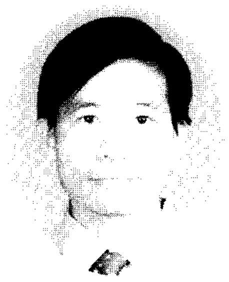
Junfu Dai is currently an Investment Consultant of Ningbo International Software Park, Ningbo, P. R. China. He received his master degree in Department of Computer Science and Technology from Tsinghua University in 2002. His research interests include computer graphics, computer aided geometric design, curves and surfaces, etc.

Huawei Wang is currently a postdoctor working in Institute of High Performance Computing of Tsinghua University, Beijing, P. R. China. He received his PhD and Meng degrees in Department of Computer Science and Technology from Tsinghua University in July, 2004, and his BSc and BEng degrees in Department of Applied Mathematics and Department of Computer Science and Technology, respectively, from Tsinghua University in July, 1998. His research interests include computer graphics, computer aided geometric design, curves and surfaces, physics-based geometric modeling, etc.

Kaihuai Qin is a Professor of Computer Science and Technology, at Tsinghua University. Dr. Qin was a Postdoctoral Fellow from 1990 to 1992, then joined the Department of Computer Science and Technology of Tsinghua University as an Associate Professor. He received his PhD and MEng from Huazhong University of Science and Technology in 1990 and 1984, and his BEng from South China University of Technology in 1982. He was a visiting scholar at SPL, BWH, Harvard Medical School, Harvard University, Boston, USA from 1999-2000. His research interests include computer graphics, CAGD, curves and surfaces, especially subdivision surfaces and NURBS modeling, physics-based geometric modeling, wavelets, medical visualization, surgical planning and simulation, virtual reality and intelligent and smart CAD/CAM.


Junfu Dai

Kaihuai Qin


Huawei Wang