2017 International Conference on Computer Network, Electronic and Automation (ICCNEA 2017)

Research on Tool Path Planning Method of NURBS Surface Based on CPU - GPU Parallel Computing

Wujia Yu
School of Automation, Hangzhou University of Electronic Science and Technology HDU
Hangzhou, China
E-mail: yuwujia@163.com

Zhendong Li
School of Automation, Hangzhou University of Electronic Science and Technology HDU
Hangzhou, China
E-mail: 57815697@163.com

Yangqiang Bi*
School of Automation, Hangzhou University of Electronic Science and Technology HDU
Hangzhou, China
E-mail: byq_work@163.com
*The corresponding author

***Abstract*—In order to deal with the inefficiency of trational serial tool path algorithms and incompatibility issues on the heterogeneous hardware platforms, this paper suggests a tool path planning method based on CPU-GPU(Central Processing Unit-Graphic Processing Unit) heterogeneous parallel computing. The method contra poses NURBS(Non-Uniform Rational B-Splines) surface which is abstracted as a matrix multiplication on the principle of isoparametric line tool path planning method. Then a parallel algorithm in accordance with Open CL(Open Computing Language) specification is proposed. Adopting data parallel programming model, the method executes multiple work-items of the GPU on the core under control of the CPU logic, and reconstructs the isoparametric line method as parallel execution instead of traditional serial execution. Simulation results show that this algorithm takes less time to generate tool paths on the CPU-GPU heterogeneous platforms, reduced by 1.5 to 15.9 times compared with traditional serial algorithm and it is of great significance to the tool path planning's real-time or quasi real-time generation.**

Keywords-Component; Nurbs surfaces; OpenCL; Parallel computing; Tool path planning; CPU-GPU

I. INTRODUCTION

In recent years, the parallelization of CNC(Computer numerical control) system computing task has been developed rapidly. The traditional CNC software architecture and computing model did not consider about the problem of parallelization at the beginning of the design, especially in the tool path planning, the traditional methods are usually used in the design of serial computing[1-2]. And the introduction of parallel computing technology can enhance the performance of the tool path planning algorithm compared with the traditional serial computing method. [3] studied the problem of parallel processing CNC system core mandate and the establishment of a parallel evaluation model from a system perspective. [4] proposed the tool path generation algorithm based on parallel computing, and discussed the parallelization problem of tool path planning algorithm.

As the above is usually the software-level of parallel technology, real parallel computing is a hardware-level multi-threaded parallel computing and heterogeneous systems of parallel computing, it has never been involved in the field of NC[5]. In this paper, the use of OpenCL to the CPU and GPU which are different processor architectures, can be calculated synergistically, the CPU for logic control and serial computing, the GPU multiple work-items perform the same kernel program to solve the NURBS surface tool path, building up the parallel computing of heterogeneous system. OpenCL is an open, parallel computing standard for cross-platform, play a role in the different performance of the new hardware, so that the parallel algorithm has better openness and compatibility[6-7].

II. BASIC PRINCIPLE

A. Tool Path Planning Method

Tool path planning method can be divided into three categories: cross-section method, projection method, parametric line method[8]. This paper only using the parametric line method to carry on the parallelization research. The parametric line method uses one parameter direction of the surface as the direction of tool, the other parameter direction is the feed direction, and then the processing surface along the selected direction in the parameter domain within the parameters of subdivision, the formation of a number of parametric line information, and finally in accordance with certain rules connected parametric line nodes constitute the tool path.

For NURBS surface, which STEP(Standard for the Exchange of Product Model Data) defines it as the only mathematical method for industrial product geometry, it is assumed that it is a mesh surface consisting of a series of NURBS curves with constant u value and v value[9].

Therefore, the solution of NURBS surfaces can be decomposed into the grid point composed of NURBS curve corresponding to each *u* value and *v* value of node vector, which is consistent with isoparametric line tool path planning method. Therefore, for NURBS surface, the tool can select the direction along the surface (*u* or *v*) as the direction of tool, (*v* or *u*) direction as the direction of feed and tool path is generated by the isoparametric method. After subdividing the parameters *u*, *v*, we can get a series of *u* or *v* parameter line. Since each of the *u*-parameter line or the *v*-parameter line is not related, each of the *u*-parameter line or the *v*-parameter line corresponds to a series of coordinate values of the *v*-parameter or *u*-parameter subdivision, that is coordinate information of the grid points can be calculated by parallel calculation. In this paper, the *u* direction is taken as the feed direction, and the *v* direction is used as the tool direction and the tool path planning is carried out by the isoparametric method.

B. *CPU-GPU Heterogeneous System*

In the CPU-GPU heterogeneous system, CPU is a multi-instruction single-data stream architecture, the data processing is basically a single pipeline, which is very good at doing logical control, while the GPU is a typical single-instruction multiple-data architecture, which is specialized in data calculation[10]. Heterogeneous system architecture shown in Fig .1, CPU and GPU through the external bus interconnection, each with its own storage space, respectively, the main memory and video memory. The execution of the program can be roughly divided into three steps in the CPU-GPU heterogeneous system: first, the input data from the CPU main memory copy to the GPU video memory; then, call the GPU implementation; Finally, the calculation results from the GPU video memory copy back to the CPU main memory.

OpenCL as a new parallel computing technology, it can call all the computer computing resources, including CPU, GPU and other processors. In the execution model of OpenCL, the program is divided into two parts to execute, namely the main program and the kernel program, in which the main program runs on the host computer, the kernel program runs on the OpenCL device (computing device), and the main program manages the running of the kernel program.

III. DESIGN AND IMPLEMENTATION OF PARALLEL POOL PATH PLANNING ALGORITHM

A. *Algorithm Design*

The tool path of the NURBS surface is correct cutting tools, which is composed of a series of grid points of the coordinate information in accordance with the provisions of the tool direction and feed direction. The row vector **U**, which consists of all the u-subdivided values, are combined with a large matrix, and the column vector $\mathbf{V}^{\mathbf{T}}$ consists of all *v*-subdivisions are also represented by a large matrix, where each subdivision value corresponds to a parametric line, furthermore, the coefficient matrix is **M**, the control vertices are denoted by the matrix **G**, w_i denoted by the matrix **W**, so the NURBS surface equation can be regarded as a series of matrix operations, the denominator is a series of matrix multiplication, similarly, the numerator is also a number of columns matrix multiplication and then inverse operation, and finally multiplied by the denominator to get the final grid point coordinate information. As the parameters *u* and *v* is not related, and is independent, they can use the assigned computing unit and the processing unit. Each processing unit performs the same matrix multiplication kernel function in parallel to get the surface parameter grid point information.

B. *The OpenCL Implementation of the Algorithm*

According to the algorithm, the computational complexity of the coefficient matrix multiplied by the control vertex matrix is small, arranged on the CPU. However, the multiplication of the coefficient matrix and the parameter matrix is computationally large and can be calculated in parallel, so it is executed on the GPU.

The main program transforms the elements which are composed of the subdivision values *u*, *v* in the parameter matrix **U**, **V** into one-dimensional arrays U[i], V[i], storing them in the CPU memory, each subdivision *u*, *v* in the parameter matrix corresponds to a parametric line; The U[i], V[i] array in the CPU memory is then copied into the OpenCL memory object buffer of the applied memory space, which corresponds to the buffer context and the context associated with the device (GPU) is globally visible, and the buffer is set to a parallel access to the memory, which can reduce the access between different threads conflict or blocking, thereby reducing the data transmission and communication overhead; And the main program obtain data from the buffer to the kernel and then executes it on the CPU. The implementation of multiple work-items in parallel with the same kernel, the result is the information of all the parametric line, which the result calculated for each work-item corresponds to a row element in the matrix, that is the coordinate information of a parametric line; The final result is also written to the global result buffer, and then the data of the result buffer through a specific function mapped to the host memory available to other parts of the program.

The steps of using OpenCL to design the main program and kernel[11]:

1) Write the kernel function KERNEL(_kernel voidMatrixmultipli()) according to the matrix multiplication logic;

2) Call clGetPlatformIDs() to find OpenCL platform collection in the computer system;

3) Through the clCreateContextFromType() to establish context for the communication between the computing device , the memory object and the command queue;

4) clCreateProgramWithSource() creates a program object associated with context and use clBuildProgram() to build program object for the specified device;

5) Create CreateKernel() on the device, execute the kernel in parallel, the kernel is __kernel void Matrixmultipli ();

6) clCreateCommandQueue() to create a command queue, submit a command to the command queue to complete the specific operation of OpenCL;

7) Call clCreateBuffer() to create a buffer to facilitate the data read and write, through the clSetKernelArg() passes the kernel parameters and buffers to the kernel;

8) Call clEnqueueWriteBuffer() the data to be involved in the calculation of information written to the buffer, here is u, v subdivided one-dimensional array;

9) clEnqueueNDRangeKernel() added the kernel event to the command queue, ready to perform on the GPU, the function parameters set the size of the workgroup and work-items;

10) Call the clEnqueueMapBuffer() and memcpy() map the buffer data to the host;

Follow the steps above to design the program and execute it on the GPU. The CPU gets the coordinate information of all the parameter grid points.

IV. Experimental Results and Analysis

The speedup is one class of the criteria for measuring the performance and effectiveness of program parallelism. It is usually defined as the ratio of the serial program execution time to the parallel program execution time:

$$\text{Speedup Ratio} = \text{Serial Program Execution Time} / \text{Parallel Program Execution Time} \quad (1)$$

The test environment of this paper are the AMD(Advanced Micro Devices) Radeon (TM) R9 200 series GPU, video memory 2048MB, RAM(Random Access Memory) 2G, CPU using AMD Athlon (TM) 3.7GHz. In order to simulate the performance of the program in different computational scale, NUBRS surface is sampled at different sampling densities, that is through different *u*, *v* subdivision, statistics on the tool path generation time. In order to reduce the interference with other programs in the system to the operation of the algorithm, the expression (1) takes the method of repeat the three operations to take the arithmetic squared value as the experimental value, the numerical result retains one decimal place. The experimental results are shown in Table I.

From the experimental results, we can see that the execution efficiency of parallel program under heterogeneous platform is improved compared with traditional serial algorithm. Through the speedup ratio and the implementation of time we can conclude that when the number of points are small, the acceleration effect is not very obvious; when the points continue to increase, the parallel computing gradually reflected the advantages.

After the *u*, *v* subdivision is greater than 1000, the tool path is too dense and the effect is not obvious. So only the parameters *u*, *v* fineness 40 * 40,100 * 100,1000 * 1000 uniform sampling tool path simulation diagram. As the picture shows:

Through simulation that when the parameters are 40 * 40, the serial execution time of 0.625ms, parallel execution time of 0.352ms, the effect of the general. When the parameter is 100 * 100, the serial execution time is 2.241ms, the parallel execution time is 0.864ms, the effect is obvious. When the parameter is 1000 * 1000, the serial execution time is 198.517ms, the parallel execution time is 53.781ms, the effect is remarkable.

V. Conclusion

Based on NURBS surface characteristics, this paper uses OpenCL technology to establish a tool path planning method based on CPU-GPU heterogeneous parallel computing. Through the analysis of the experimental results, the parallelization of the calculation greatly enhance the performance of the isoparametric line planning method, and realize the cooperative parallel computing of different architectures.

TABLE I. NURBS SURFACE TWO KINDS OF PROGRAM EXECUTION TIME (UNIT: MS)

U fine fraction * V fine fraction	Serial program execution time	Parallel program execution time	Speedup ratio
40*40	0.6	0.4	1.5
100*100	2.2	0.9	2.4
1000*1000	198.6	53.8	3.7
2000*2000	768.4	96.7	7.9
3000*3000	1768.5	138.6	12.8
4000*4000	3026.8	198.4	15.2
5000*5000	4897.1	306.3	15.9

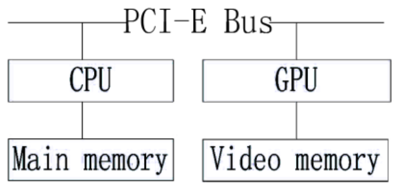

Figure 1. CPU-GPU heterogeneous architecture diagram

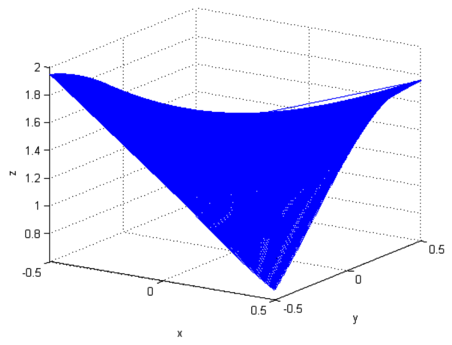

Figure 2. NURBS surface example.

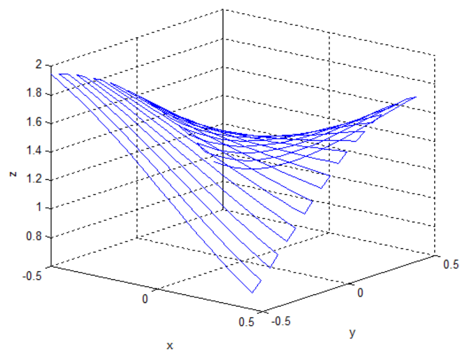

Figure 3. 40* 40 uniform sampling.

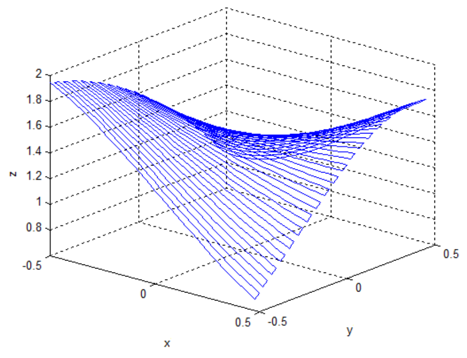

Figure 4. 100*100 uniform sampling.

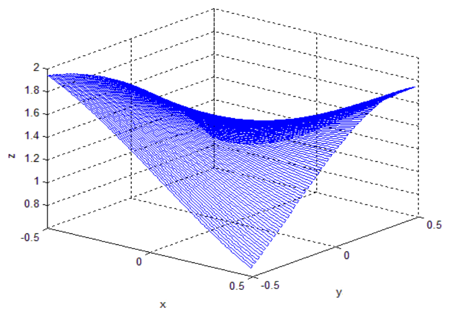

Figure 5. 1000*1000 uniform sampling.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (5140051146).

References

[1] Qin Hong-xin, Hua Rui. Research on Tool Path Planning for NC Machining of Freeform Surface[J]. Coal Technology 2013,30(3):40-41

[2] Li li, Fang li-jin, Wang Guo-xun. Research on Tool Path Planning of NURBS Surface Five - axis Machining[J]. Machinery 2014,52(2): 5-9.

[3] Zhang Xiang-li, Tang Xiao-qi, Chen Ji-hong. Parallel processing of computer numerical control system [J] .Computer Integrated Manufacturing System, 2008,14(8), 1603-1607.

[4] Yu Zhan-yue, Zhou Ruo-rong, Zhuang Hai-jun. A Parallel Algorithm for Tool Path Generation in NC Machining [J]. Mechanical Science and Technology, 2004,23 (3), 266-268.

[5] Yu Wu-jia. STEP-NC-based five-axis machining tool path planning method [D]. Doctoral dissertation, Zhejiang University.

[6] Stone J E, Gohara D, Shi G. OpenCL: A parallel programming standard for heterogeneous computing systems[J]. Computing in science & engineering, 2010, 12(3): 66-73.

[7] Du P, Weber R, Luszczek P, et al. From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming[J]. Parallel Computing, 2012, 38(8): 391-407

[8] WU Fu-zhong.Modeling Path Planning of Equal Spacing Tool for Free-form Surface Coordinate Processing [J] .Computer Integrated Manufacturing System, 2007, (10): 2064-2070.

[9] Starly B, Lau A, Sun W, et al. Direct slicing of STEP based NURBS models for layered manufacturing[J]. Computer-Aided Design, 2005, 37(4): 387-397.

[10] Research on Virtualization Technology Based on CPU / GPU Platform [D].Journal of Shanghai Jiaotong University

[11] Munshi A, Gaster B, Mattson T G, et al. OpenCL programming guide[M]. Pearson Education, 2011:25-73.