Send Orders for Reprints to reprints@benthamscience.ae

The Open Electrical & Electronic Engineering Journal, **2014**, *8*, 631-635

Interactive Pickup of Three-dimensional Fine Point Cloud Based on GPU

Ming Huang[1,*], Fang Yang[1], Yong Zhang[1] and Xinle Fu[2]

[1]*Beijing University of Civil Engineering Architecture, Key Laboratory for Urban Geomatics of National Administration of Surveying, Mapping and Geoinformation, Engineering Research Center of Representative Building and Architectural Heritage Database, The Ministry of Education, Beijing 100044, P.R. China*

[2] *GIS Application Development Department, Sichuan Remote Sensing Geomatics Institute, Chengdu 610100, P.R. China*

Abstract: Three-dimensional fine point cloud has gradually become a key data source of three-dimensional model. The large scale point cloud interactive quick pick up is a kind of important operation in the point cloud data processing and applications. Since the point cloud model is composed of massive points, the speed of ordinary picking method is limited. A GPU-based point cloud picking algorithm was thus presented to solve the problem. The basic idea of the algorithm is that by spatial transformation converting the point cloud to screen space, and then, the point was calculated which is the nearest to the mouse click point in screen space. The GPU's parallel computing capabilities were used to achieve spatial transformation and distance comparison by compute shader in this algorithm. So the speed of the pickup has been increased. The results show that compared with the CPU, the pickup method based on GPU has greater speed advantage. Especially for the point cloud over 4 million points, the speed of the pickup has been increased 2-3 times faster.

Keywords: Compute shader, GPU, pickup, screen space, three-dimensional point cloud.

1. INTRODUCTION

In an interactive computer graphics application, such as 3D games, virtual reality, CAD/CAM, *etc*, it requires the user to interact with the system *via* an input device. These interactions include rotation, translation, delete, view object status information *etc*. Interactions need to locate the object in the scene by picking up operation. Pickup technology has become one of the most basic and a very important component of these systems. Because there is such a wide range of pickup applications, many scholars have conducted in-depth research. Yao Jiquan and Wang Jian put forward that by adding a third dimension to the two-dimensional coordinates of pickup to inverter exchange for world space help calculate pick-rays according to the depth of the order of objects make-ray-object intersection judgment in the world space [1]. Zhu Mingliang put forward picking algorithm [2] based on viewport space. In recent years, graphics hardware has gradually become a powerful programmable function. There has been the extensive use of hardware (GPU) acceleration pickup algorithms [3] such as: Hanrahan and Haeberli raised WYSIWYG method [4]; Zhang Jiahua and other proposed GPU picking algorithms which use Geometry Shader to implement ray - geometry intersection operation [5]; and Zhao Hanli and Jin Xiaogang and other proposed FRMP method [6].

In recent years, the method of a point which is represented as a primitive object's surface has gradually become the focus of the study. On the one hand, due to the rapid development of three-dimensional laser scanning technology, super-large scale models can be acquired through laser scanning, such models are initially represented as a three-dimensional point discrete (point cloud). On the other hand, the computing power of the computer has reached a new level, at which level, the point such as drawing primitives become meaningful [7, 8]. However, the study agrees that pickup algorithms are meant for triangular elements, but not for point primitives. These algorithms are not suitable for point clouds, and because of the large point cloud data, the speed of these pickup methods is very slow.

In this paper, the pickup point cloud GPU-based algorithm was presented. The algorithm has the following characteristics: (a) it processes point cloud data, and is suitable for large-scale point cloud model pickup; (b) it is based on the screen space. The point is transformed from local space to screen space, and then compared with the distance and made consistent with the pipeline rendering order. It is not involved in the transformation matrix inversion. The algorithm is not restricted to the problem that the inverse matrix of the transformation matrix does not exist; (c) The algorithm is based on the GPU. DirectX 11 Shader (compute shader) was used to make point conversion and distance comparison operations thanks to the powerful GPU's parallel computing capabilities [9, 10]. Compared with the algorithm running on the CPU, Pick up speed is 2-3 times faster. The larger the point cloud, the more obvious the advantages of speed.

*Address correspondence to this author at Beijing University of Civil Engineering Architecture, Beijing 100044, P.R. China; Tel: +8613391652980; E-mail: 595638440@qq.com

1874-1290/14

2. GPU-BASED ALGORITHM FOR POINT CLOUD PICKUP

2.1. Point Cloud Picking Algorithm Based Pn Screen Space

Currently in computer graphics, the most widely used algorithm is the classic Ray picking algorithm. The basic principle is that firstly users click on the screen to take a screen coordinate, then the coordinate is transformed into a viewport coordinate of display system. Next, add the depth value to the point. Through a series of coordinate space conversion, reversely calculate the coordinates of the pickup point in the world coordinate system. Lead a ray by viewpoint (camera position) to the pickup point. In the three-dimensional space, rays and objects conduct intersection. If they intersect, the object is picked up [11, 12]. Since the point cloud model is composed of a large number of points and the point has no radius or size, so it cannot be judged whether a point and the ray intersect each other. Therefore, ray picking algorithm does not apply to the pickup point cloud data. Point cloud is picked up through the method which is based on the screen space.

The basic idea of the algorithm is making a series of space conversion to the point of points cloud. Calculate projection coordinates on the screen. In screen coordinates, obtain point which is nearest to the points that user clicks on the screen. Specific steps are as follows:

a: Users click on the screen to get the screen coordinates of the point S (s_x, s_y).

b: Strike the screen coordinates of the points which are from point cloud. The coordinates of the point $P_i(x_i, y_i, z_i)$ from point cloud are known. The point is multiplied by the world matrix. And the point is transformed from model space to world space. Next, the point is multiplied by the view matrix M_{view} to be transformed from world space into the observation space. And then, it is multiplied by the projection matrix, the transformation is made from observation space to the projection space. Finally, it is multiplied by viewport transformation; the transformation is made from projection space to screen space. In the screen coordinate, system coordinates $P_i^{'}(x_i^{'}, y_i^{'}, z_i^{'})$,

$$P_i^{'} = P_i * M_{world} * M_{view} * M_{project} * M_{viewport} \quad (1)$$

Viewport transformation matrix:

$$M_{viewport} = \begin{bmatrix} w/2 & 0 & 0 & 0 \\ 0 & -h/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ w/2 & h/2 & 0 & 1 \end{bmatrix} \quad (2)$$

Where w is the width of viewport, h is the height of the viewport.

c: In screen coordinates, calculate the point $P_i^{'}(x_i^{'}, y_i^{'}, z_i^{'})$ (which does not participate in the operation) which is the nearest point $P^{'}(P_x^{'}, P_y^{'}, P_z^{'})$ from S(s_x, s_y). The corresponding point $P(p_x, p_y, p_z)$ in model space is the selected point.

2.2. The Processes of GPU-B Based Pickup Point Cloud Algorithm

The algorithm of pickup point cloud is based on the screen space. For large-scale point cloud (cloud point number greater than 15 million), the algorithm operations are performed on the CPU, and the speed of picking up is limited. This limitation can be solved by a powerful GPU parallel computing power. For the purpose, Microsoft's DirectX 11 compute shader newly added can be used to perform graphics unrelated general-purpose computing [13]. The above picking algorithms are ported to compute shader. By calculating shader's powerful parallel computing capabilities, you can solve the shortage of the pickup algorithm and improve pick up speed.

The basic idea of the GPU-based point cloud pick up algorithm is that the transformation of point and the distance calculation of clicking screen point are implemented in Direct3D 11 [14, 15]. Using GPU parallel computing to improve the speed of picking up, the whole process is shown through a flow chart in Fig. (**1**):

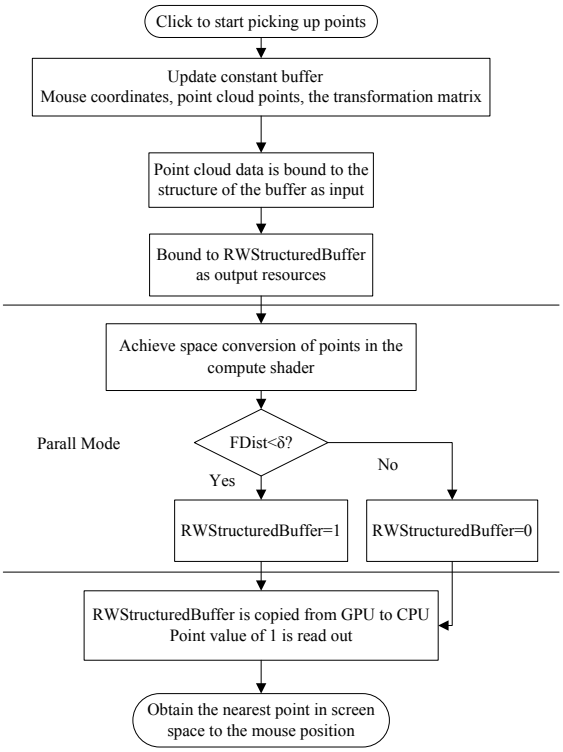

Fig. (1). GPU-based algorithm of pickup point cloud flowchart.

2.3. The Implementation of GPU-based Pickup Point Cloud Algorithm

The main process of the algorithm is implemented by a compute shader. Specific implementation process includes these steps: creating resources for compute shader, initializing resources and binding resources. setting compute shader, and calling dispatch execution.

2.3.1. Create, Initialize and Bind Resources for the Compute Shader

Compute shader requires the use of three resources: constant buffer, the buffer structure and RWS structured Buffer [16], as shown in Fig. (**2**).

Since the algorithm needs information of the screen-clicked point, the numbers of points and the transformation matrix, these values are stored with constant buffer. Before compute shader execution, the value of these variables needs to be passed to the constant buffer (Constant Buffer). Then start binding coordinate point cloud data into the structured buffer (Structured Buffer). When the compute shader's calculations are completed, the result of the calculation needs to be returned to the CPU by RW Structured Buffer. RW Structured Buffer in CPU is not only bound to buffer data for the GPU use but also can copy the buffer to the CPU for reading.

2.3.2. Compute Shader Setup and Execution

A certain number of thread groups can be opened by Dispatch (X, Y, Z) function in shader file. Specify the number of threads in a thread group containing Bynum threads (X, Y, Z) function. As shown in Fig. (**3**), a thread group may have a composition of n threads. But during the actual hardware implementation, these threads are divided into warps (Each warp is composed of 32 threads). Each wrap is processed by multiprocessor on the SIMD32. You can specify multiple threads within a thread group where the number is not 32. However, for performance reasons, the number of threads in the thread group should be the warp (32 threads) multiples. Program specifies a thread group has 256 threads composition. Because each thread handles one point, the program open (NumOfPoint + BLOCK_SIZE -1) / 256 thread group. NumOfPoint represents the total number of points. The number of BLOCK_SIZE is 256 to ensure that each point is processed. As shown in Fig. (**2**), for simplicity reasons, through numthreads (BLOCK_SIZE, 1, 1) and Dispatch ((UINT) (NumOfPoint + BLOCK_SIZE -1) / 256, 1, 1) the threads and thread groups are set to a dimension.

The number of threads which the compute shader opens, is the same as the number of point clouds, so a thread is responsible for operation of a point. As shown in Fig. (**4**), each thread corresponding to the index point will make space conversion. It is converted from local coordinate system to

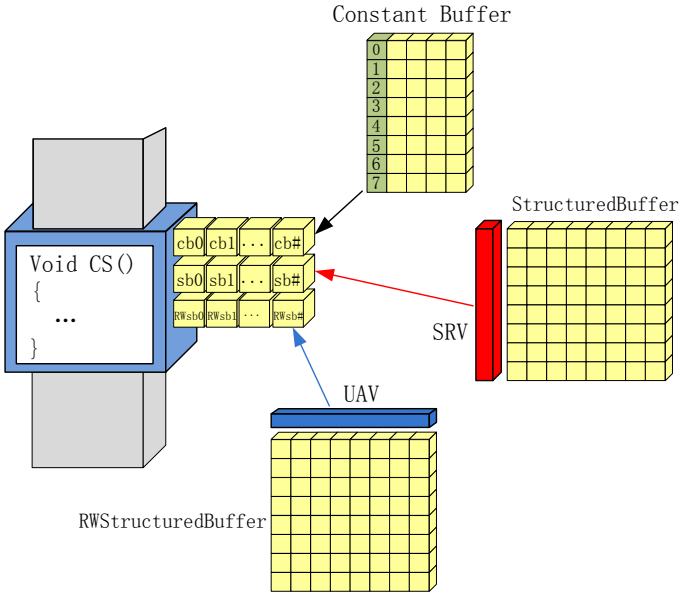

Fig. (2). Schematic diagram of compute shader binding resources.

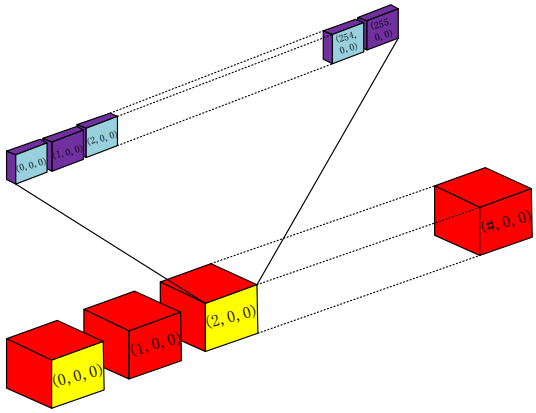

Fig. (3). Compute shader thread group and thread scheme.

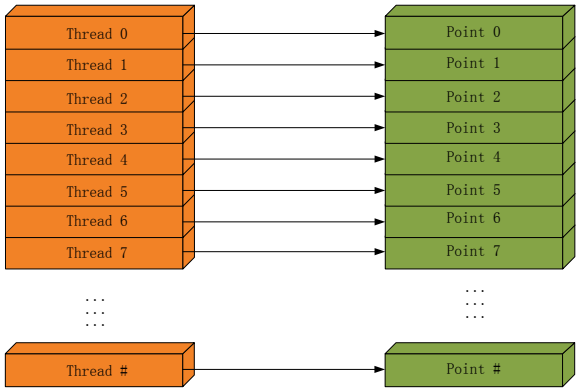

Fig. (4). Schematic diagram of each thread is responsible for a point of operation.

the screen coordinate system. Determine the distance from point to each screen-clicked point which is less than the set value.

2.3.3. Retrieve the Compute Shader Operation Result

After the calculation of compute shader is completed, RWStructuredBuffer needs to be copied from the GPU to the CPU, to be read by the CPU. For this, a buffer needs to eb created, in which the usage of buffer tag is D3D11_USAGE_STAGING, and CPU buffer access method is D3D11_CPU_ACCESS_READ. Use Copy-Resource method to copy the GPU resources to the system memory. Finally, we can use the Map and Uncap functions mapping buffer to read CPU resources. At last, a small amount is calculated by the CPU to obtain the selected point.

3. ANALYSIS OF EXPERIMENTAL RESULTS AND CONCLUSION

This paper is based on the following configuration of CPU: Intel Core 26600 and GPU: the NIVIDIA GT610 2G computer, and Based on Visual Studio 2010 environment, combination of using C ++ language and DirectX 11 programming. The picking algorithm which is running on the CPU is based on screen space. It is the combination of the method of picking-up point cloud based on CPU and the GPU-based point cloud pickup algorithms. The point cloud number in the experimental data is in the range of 130,000 to 18, 000, 000. By drawing crosshairs, the point selected to judge the accuracy of selection, is as shown in Fig. (**5**).

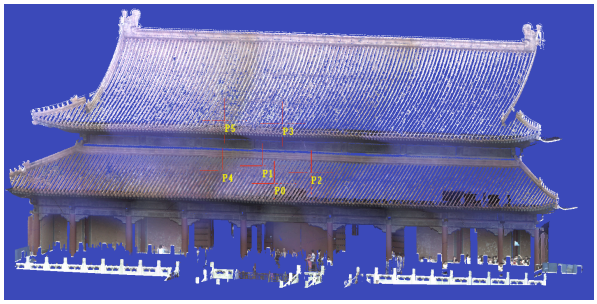

Fig. (5). Example of pickup point.

By testing different sizes of data, respectively, the amount of picking up time is obtained, which is either CPU-based or GPU-based. After statistical methods the statistics obtained are given in Table **1**.

For picking algorithm based on screen space, the main time-consuming task is to make space conversion and distance comparison. A major time-consuming task of GPU-based algorithm for picking is to copy the results from the GPU to the CPU. For the small amount of data of the point cloud (the number of points less than 2 million), while picking method based on GPU is faster, the difference in the time consumed for two methods is not large/significant. For large amounts of data point cloud (the number of point is greater

Table 1. Experimental statistics table.

The number of points from point cloud (Ten thousand)	Time consuming of the method of picking-up point cloud based on CPU (A millisecond)	Time consuming of the method of picking-up point cloud based on GPU (A millisecond)
13.1516	11.806	9.345
28.2361	21.554	16.342
57.1459	42.917	25.429
79.0590	58.134	34.365
139.1901	105.674	72.547
228.2259	203.952	105.149
440.0833	384.890	180.702
608.3357	556.826	240.427
836.0005	706.340	313.959
965.2038	816.199	372.684
1791.3302	1296.009	560.421

than five million), GPU-based picking method is more than two times the speed of the methods which is based on screen space. And with the increasing amount of data, the speed advantage of GPU-based method of picking becomes more and more obvious. Thus, for modern high-performance of GPU, it is suitable for performing a large number of general purpose computing. For large point cloud, using the GPU for pickup has a great speed advantage.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This paper was supported by plan projects National Administration of Surveying, Mapping and Geoinformation of China (Grant NO.2013CH-15, special public welfare industry research 201512009) and by the National Natural Science Foundation of China (Grant No.41301429). Project also supported by the Special Scientific Research Fund of Surveying Public Welfare Profession of China (Grant No. 201512009).

REFERENCES

[1] J. Q. Yao and X. X. Li, "Research on 3-dimension pick-up of human-computer interaction in computer graphics". *Journal of Engineering Design*, vol. 13, no. 2, pp. 116-120, 2006.

[2] M. L. Zhu, B. Dong, Y. Wang, and B. Y. Xie, "Algorithm for Picking in 3D Scenes Based on Viewport Space", *Journal of Engineering Graphics,* vol. 2, pp. 84-97, 2008.

[3] A. F. Abate, M. Nappi, and S. Ricciardi, "GPU accelerated 3d face registration/recognition", *Lecture Notes in Computer Science, Advances in Biometrics,* vol. 4642, pp. 938-947, 2007.

[4] P. Hanrahan and P. Haeberli, "Direct WYSIWYG painting and texturing on 3D shapes", *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 4, pp. 215-223, 1990.

[5] J. H. Zhang, C. Liang, and G. Q. Li, "3D Primitive Picking on GPU", *Journal of Engineering Graphics,* vol. 1, pp. 46-52, 2009.

[6] H. Zhao, X. Jin, and J. Shen, "Fast and Reliable Mouse Picking Using Graphics Hardware", *International Journal of Computer Games Technology*, vol. 9, pp. 11-18, 2009.

[7] T. Akenine-moller and E. Haines, "*Real-Time Rendering*", Third Edition, USA：A K Peters, 2008.

[8] J. Bosch, P. Goswami, and R. Pajarola, "Simple and efficient terrain rendering on the GPU", In: *Proceedings of EUROGRAPHICS*, pp. 35-42, 2009.

[9] A. Moslah, V. Valles-Such, S. Guitteny, Couvet, and S. Philipp-Foliguet, "Acceleratedmulti-view stereo using parallel processing capababilities of the GPUs", 3DTV Conference, pp. 1-4, May 2009.

[10] D. Bhattacharya and S. Roychowdhury, "A Constrained Cost Minimizing Redundancy Allocation Problem in Coherent Systems with Non-overlapping Subsystems", *Advances in Industrial Engineering and Management*, vol. 3 no. 3, pp. 1-6, 2014. doi:10.7508/AIEM-V3-N3-1-6

[11] Y. X. Guo, T. P. Hou, Y. Y. Du, "Picking Entities in 3D Scene Based on DirectX", *Journal of Liaoning University of Petroleum & Chemical Technology,* vol. 29, no. 3, pp. 77-80, 2009.

[12] H. Nie, "Interactive mapping of pixel pickup", *Journal of Northwest Institute of Textile Science And Technology*, vol. 11, no. 4, pp. 329-332, 1997.

[13] Microsoft Corporation. DirectX 11 SDK, USA: Microsoft Corporation, 2010.

[14] D. Frank D, "*Introduction to 3D Game Programming with DirectX 11*", USA: Mercury Learning & Information, 2012.

[15] J. Zink, M. Pettineo, and J. Hoxley, "*Practical Rendering and Computation with Direct3D 11*" USA：A K Peters, 2011.

[16] C. Boyd, "*Direct3D 11 Compute shader: More generality for advanced techniques*" Gamefest USA, 2008.

Received: September 18, 2014 Revised: December 22, 2014 Accepted: December 31, 2014