Rapid determination of three-dimensional convex shapes by dispersion processing using Java RMI

SatoshiKodama[#1], ReiNakagawa[*2], YukaOzeki[*3]

[#]*Researcher, Research institute for Science and Technology, Tokyo University of Science*
2641 Yamazaki, Noda, ChibaPrefecture, Japan

[*]*Department of Information Sciences, Faculty of Science and Technology, Tokyo University of Science*
2641 Yamazaki, Noda, Chiba Prefecture, Japan

Abstract

To quickly determine the interior and exterior of a three-dimensional (3D) shape, one must apply shape recognition and contact determination algorithms. However, in general, a 3D figure largely differs from a two-dimensional figure, and is described by a large dataset. Consequently, the determination process is time intensive. To alleviate this problem, determination methods of 3D complex shapes are often based on solid angles, but this approach is inapplicable to many shapes unless the computer is equipped with a graphics processing unit. On the other hand, the use of embedded personal computers such as 3D printers and portable 3D scanners is increasing in modern data processing, and environments free of special devices are also required.

In this paper, we show that high-speed processing of convex object can be achieved by parallel computing using a plurality of relatively inexpensive Raspberry Pi3s.

Keywords — *Parallel computing, Java RMI, 3D Modeling, Inside/outside determination*

I. INTRODUCTION

Interior-exterior determination of three-dimensional (3D) shapes is an important technology not only in 3Dmodeling and contact detection, but also in shape recognition and region determination [1-3]. However, 3D shapes largely differ from two-dimension (2D) shapes, and are described by large datasets that greatly lengthen the time of the determination processing [4-6]. Therefore, the processing is usually performed using a graphics processing unit (GPU), but the algorithm must conform to the GPU after adding a device [7-9]. For this reason, methods for improving the overall processing speed have focused on the parallel connection of a plurality of relatively inexpensive small personal computers (PCs) [10-12]. In this case, a typical program must be supplemented by a network program that performs the parallel processing. Although the auxiliary program requires optimization, a device-aware system does not need to be constructed. In other words, the parallelism is easy to implement, is not restricted by the development environment, and can be programmed using ordinary control flow [13,14].

In this paper, we show that interior-exterior determination can be performed at high-speed by parallel processing of multiple Raspberry Pi 3 devices, which are widely used and have shown remarkable performance improvement with the development of Internet of Things (IoT).

II. RELATED RESEARCH

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

A. *The shape determination method*

The internal and external environments of a 3D shape can be determined by extending the 2D Crossing Number Algorithm or the Winding Number Algorithm. This section explains the 2D algorithms and their 3D extensions in detail.

a) Crossing Number Algorithm

The 2D Crossing Number Algorithm distinguishes the internal and external points of a 2D shape by counting the number of edge-crossings of a straight line passing through the shape from each point (Fig. 1).

Although this method determines the internal and external points at relatively high-speed, it can cause an erroneous judgment (Fig. 2) [5,15-17].

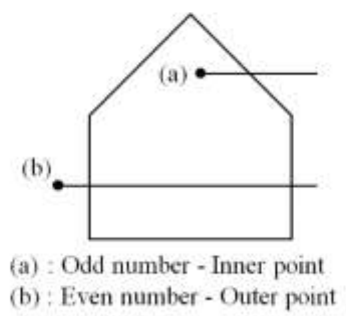

Fig. 1. Exterior-interior determination by the ray method

In three dimensions, the Crossing Number Algorithm extends a straight line in all directions and counts the number of intersections with an object. However, like its 2D equivalent, this approach can misjudge the situation of ambiguous points [18,19].

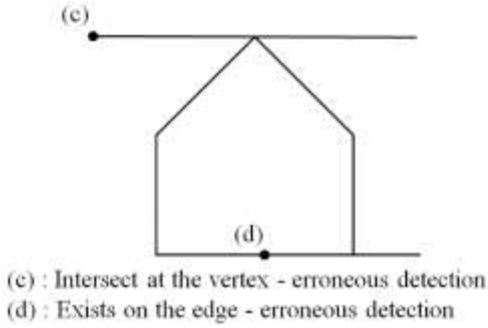

Fig. 2. Example of erroneous detection by the ray method

b) Winding Number Algorithm

The 2D Winding Number Algorithm detects the internal and external points of a shape using angles.

For example, as shown in Fig. 3 and 4, if one sequentially moves from point V_1 to point V_5 in the counterclockwise direction, the internal-external determination is performed by adding the angles subtended from the point of interest to successive pairs of vertices (in clockwise direction, the angles will be negative; see Fig. 5) [20]. If the angles sum to 2π or 0, the point is inside and outside the shape, respectively. By this analysis, the point in Fig.3 is discerned as internal, whereas that in Fig. 4 is external.

Summarizing the above results, the n-squared determination result is expressed as follows:

$$\sum_{i=1}^{n}\theta_i = \begin{cases}2\pi, & inner\\ 0, & outer\end{cases}$$

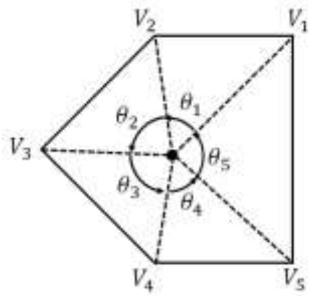

Fig. 3. Inner point discerned by the Winding Number Algorithm

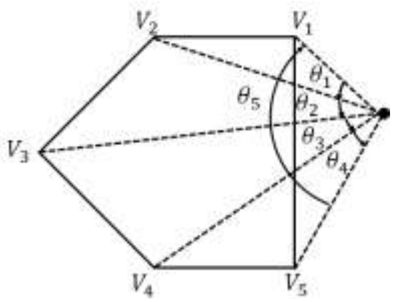

Fig. 4. Outer point discerned by the Winding Number Algorithm

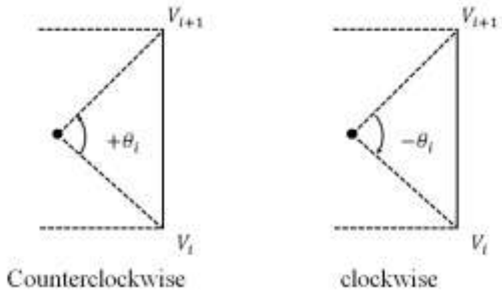

Fig. 5. Positive–negative determination by rotation direction

The 3D Winding Number Algorithm replaces the angle by the solid angle. When based on the solid angle, the determination algorithm must consider the surface area of a sphere (Fig. 6) [21]. The determination will cover the entire sphere of an internal point (Fig. 7), but only part of the sphere for an external point (Fig. 8) [20,21].

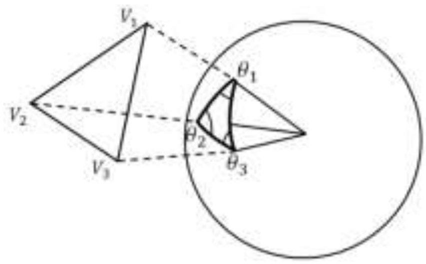

Fig. 6. Spherical trigonometry

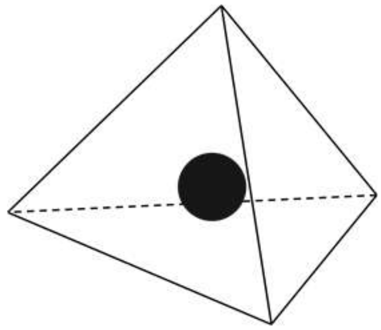

Fig. 7. Projection of an internal point

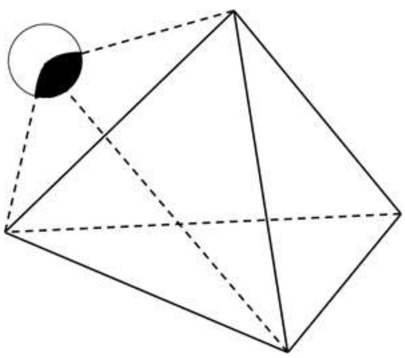

Fig. 8. Projection of an outside point

In general, the 3D Winding Number Algorithm performs the following calculation:

$$S_j = \left(\sum_{i=1}^{n} \theta_i - \pi \right) \; (j \geq 1)$$

$$S_{all} = \sum_{j=1}^{m} S_j = \begin{cases} 4\pi, & outer \\ 0, & inner \end{cases}$$

Here, S_{all} is the vertex size over all planes, m is the number of polygons comprising the object surface, and the sphere has unit radius [20-22].

Note that the solid angle determination is positive when viewed from the outside and negative when viewed from the inside, as shown in Fig. 9.This determination based on the input order of the data is also used in polygon-display algorithms, and is a general data structure of 3D objects [23-25].

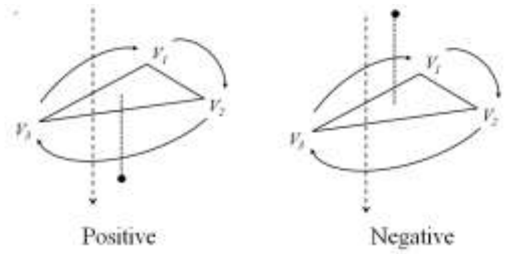

Fig. 9. Positive–negative determination by rotation direction in the 3D Winding Number Algorithm

The winding number determination correctly discerns the internal and external points even of complex shapes, but the numerous circular functions require a long processing time [16, 17].

B. GPU and parallel processing

Although it requires a special device, a method based on general purpose GPUs returns the determination results at high-speed by generating a huge number of threads [26-28]. However, unlike normal development methods, this approach needs a host-side program and a device-side program, which must usually be developed (Fig. 10) [26].

Furthermore, in the Single instruction, multiple thread (SIMT) format of such a program, the control flowis restricted for efficiency [26-29].

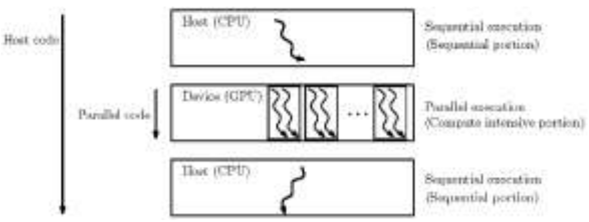

Fig. 10. Programming model and execution

C. Parallel processing via network

The authors of [12] achieved high-speed judgment with no special device by connecting multiple PCs in parallel via a network.When performing parallel computing over a network, inputs and outputs are commonly sequenced as bytes using a socket programming. The communication begins by specifying the IP address and the port number. Therefore, despite its versatility, this method must consider the data structures and network environments of the other PCs in the network.

To overcome this difficulty, rather than processing the transmission and reception as byte sequences, our method performs parallel processing on an object-by-object basis. The proposed method is demonstrated on a Java Remote Method Invocation (RMI) in the following example.

Java RMI is a middleware that enables distributed computing. A Java RMI application automates the processing between the server and the client, while both parties are unaware of the above conversion. Furthermore, as the object can be passed as an argument and a return value when calling a method, even complex data structures can be easily handled. In addition, programs can be called by all computers in the network, meaning that a program running on a remote computer can be called and used like a computer's own function (Fig. 11) [30-32].

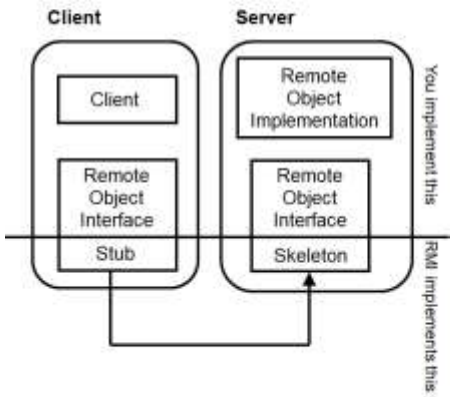

Fig. 11. How Java RMI works [32]

III. RESEARCH CONTENT

As the solid angle processing in the 3D Winding Number Algorithm is time-intensive, our internal-external determination method adopts a vector formulation. We also clarify whether the parallel distributed processing delivers high-speed results on PCs with relatively low performance, such as those used in IoT devices.

A. Internal-external determination by the vector-based method

The algorithm in the proposed method uses a vector rather than the solid angle.

First, all polygons are projected onto the plane as shown in Fig. 12. Next, the target polygon is specified by the determination processes shown in Fig. 13 and 14. The splitting function[33, 34] is given by

$$f_{AB}(x_1, x_2) = (b_2 - a_2)(x_1 - a_1) + (a_1 - b_1)(x_2 - a_2) \cdots \text{(A)} .$$

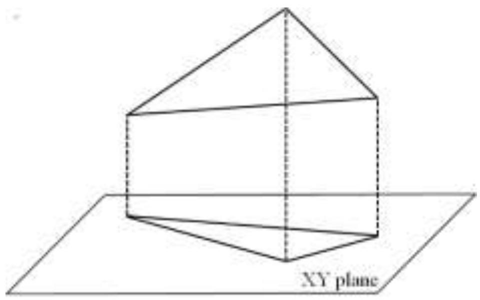

Fig. 12. Projection on the plane

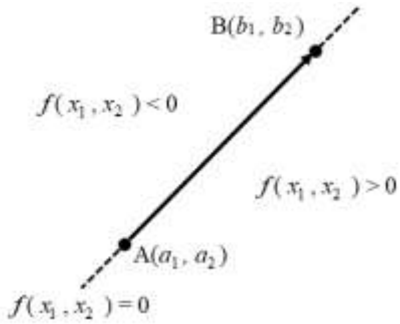

Fig. 13. Splitting function[33, 34]

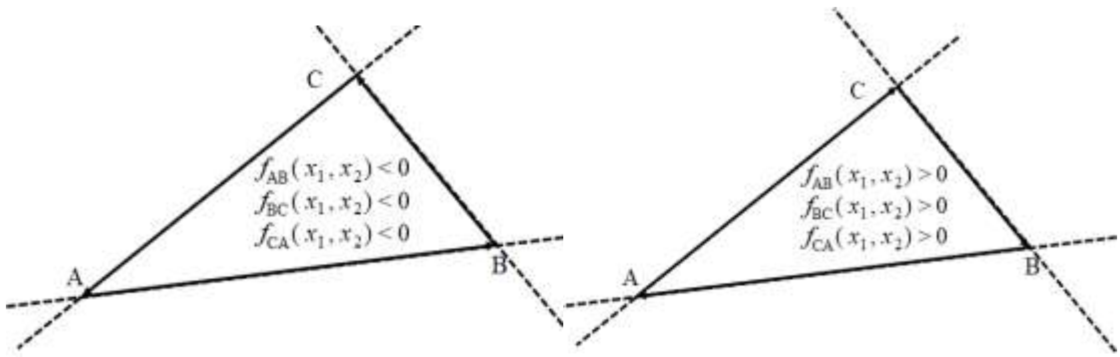

Fig. 14. Internal determination method (positive/negative matches inside the polygon)

If (A) lies on a straight line, the inside-outside determination is performed by changing the projection direction from the XY plane to the YZ plane, as shown in Fig. 15.

Based on the cross product and inner product, the algorithm decides whether the target coordinates belong to the upper part of the polygon. This judgment is made in order from the upper part (Fig. 16). Finally, the areas are merged to give the inside-outside judgments along the axis.

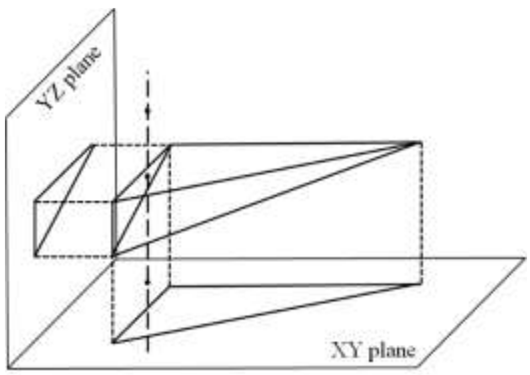

Fig. 15 If function (A) is = 0

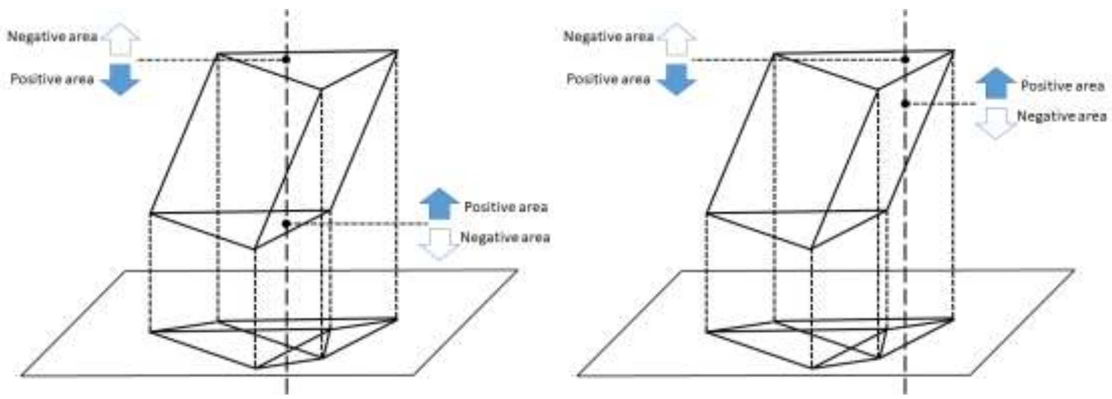

Fig. 16. Determining the sign of the area of each polygon

B. Determination processing by the server-client method

The proposed method was implemented in a Java RMI. In the server-client environment, the vector-based inside-outside determination algorithm is implemented in a plurality of servers, and each server processes one area of the determination (see Fig. 17).

The decision result of each server is finally merged on the client side to obtain the result (Fig. 18). Note that the number of servers depends on the number of constructions.

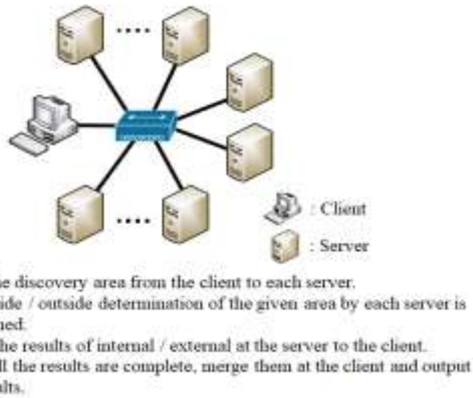

Fig. 17. Server–client relationship in the parallel implementation of the proposed method

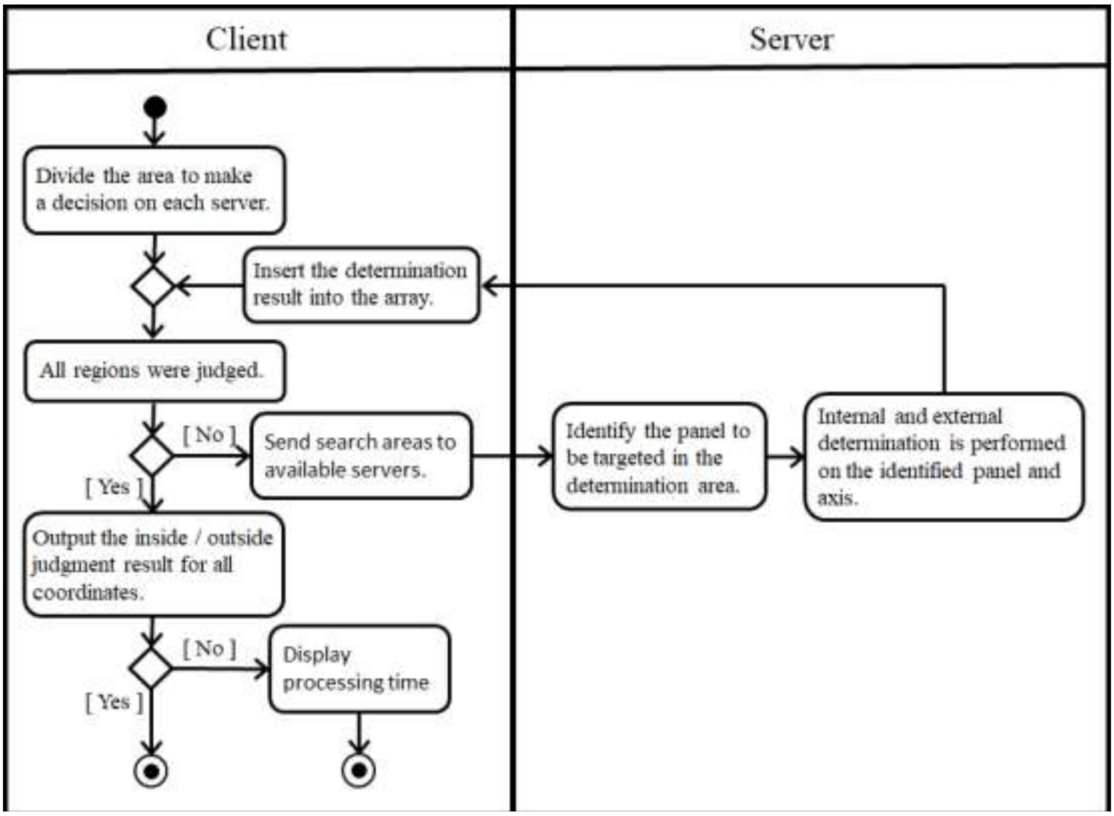

Fig. 18. Program flow of client and server

IV. EXPERIMENT

In the verification experiment, the calculation rage was set to −50≤x<50, −50≤y<50, −50≤z<50 (Integers), and the determination was performed on Figs. 19-22. The numbers of vertices and 2D polygons of the three-dimensional shape constituting each figure are shown in Table 1.

Each Raspberry Pi 3 judged the interiors and exteriors in regions comprising 5, 10, and 25 square pixels x × y and 101 pixels along the z direction. In addition, the number of Raspberry Pi 3 devices in the parallel calculations was varied as 1, 4, 8 and 16. The specifications of the devices are given in Tables 2-4.

The results are shown in Figs. 23-26, and the processing times of obtaining the determination results are listed in Table 5. Furthermore, Figs. 27-30 display the speed changes of processing each figure the number of Raspberry Pi 3 devices increased from 1 to 16.

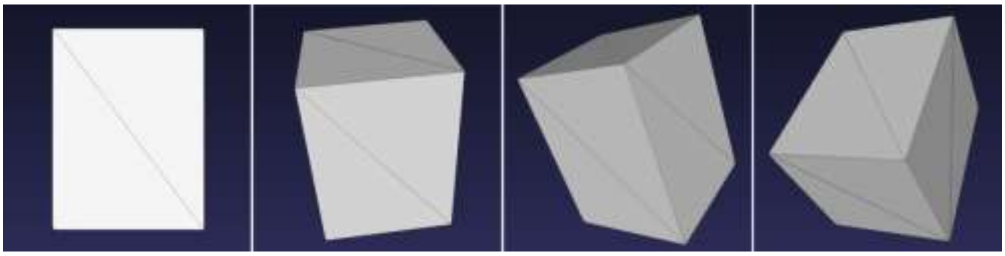

Fig. 19. Sample data 1

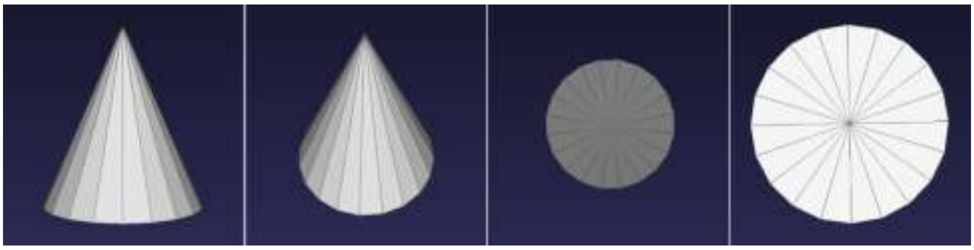

Fig. 20. Sample data 2

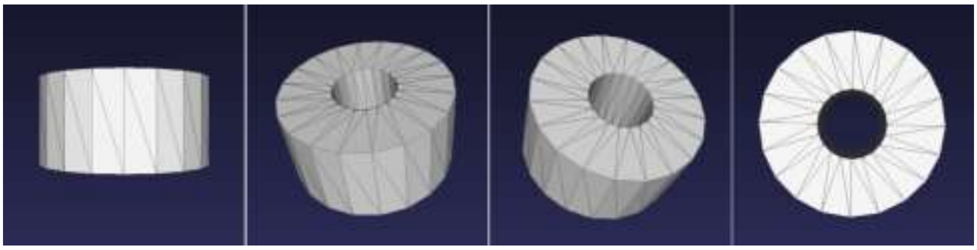

Fig. 21. Sample data 3

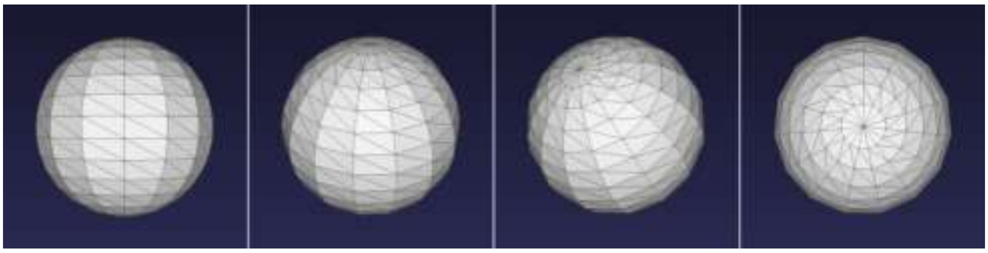

Fig. 22. Sample data 4

TABLE 1. Corresponding shape

	Number of vertices	Number of polygons
Sample data 1 (Fig. 19.)	8	12
Sample data 2 (Fig. 20.)	22	40
Sample data 3 (Fig. 21.)	80	160
Sample data 4 (Fig. 22.)	212	420

TABLE 2. Main specifications of Raspberry Pi 3B

	Specification
CPU	Broadcom BCM2837 1.2 GHz ARM Cortex-A53
RAM	1 GB LPDDR2 (900 MHz)
Ethernet	RJ-45 x1: 100BASE-TX 10BASE-T
Storage	MicroSD card slot
power supply	+5 V (2.5 A) micro USB socket

TABLE 3. Main specifications of Client

	Specification
CPU	Intel Core i7-7700HQ @ 2.80GHz
RAM	32 GB DDR4 (2400MHz)
Ethernet	RJ-45 x1: 1000 BASE-T
Storage	SSD 240GB

TABLE 4. Experimental environment

	Specification
Client	OS: CentOS Linux release 7.4.1708 Java Development Kit: javac 1.8.0_131
Server	OS: CentOS Linux release 7.2.1511 Java Development Kit: javac 1.8.0_191 Ethernet: 100 Mbps (as the theoretical value)

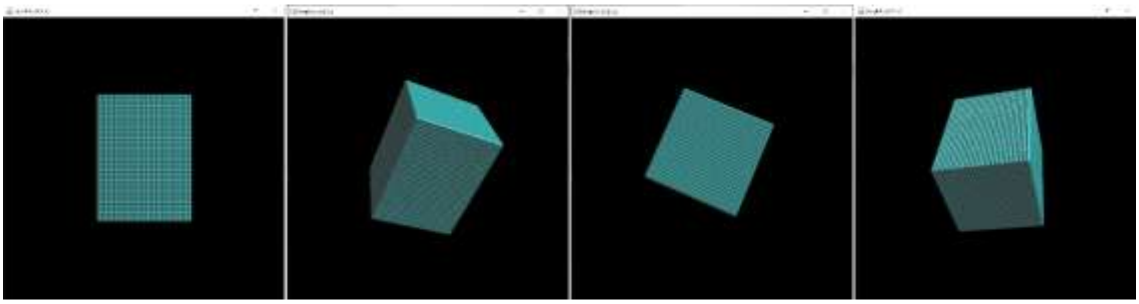

Fig. 23. Result (Sample data 1)

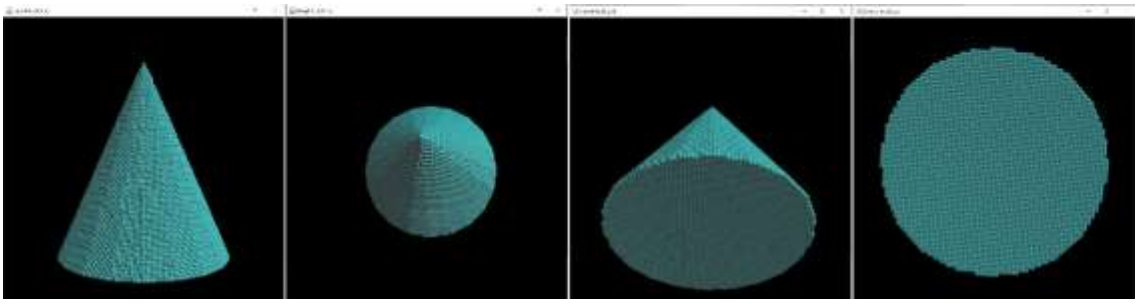

Fig. 24. Result (Sample data 2)

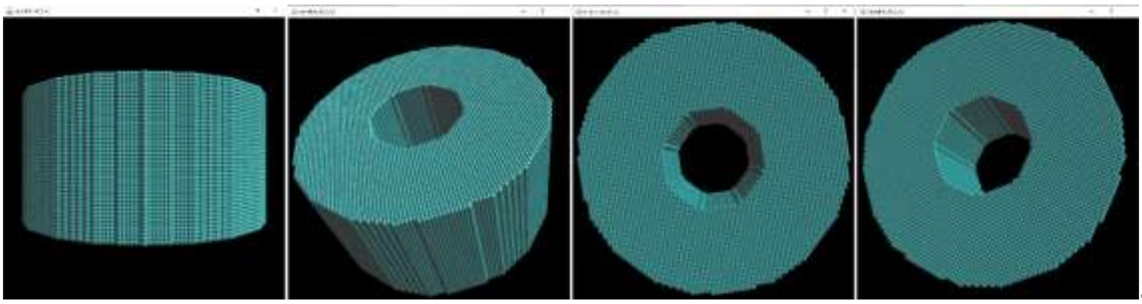

Fig. 25. Result (Sample data 3)

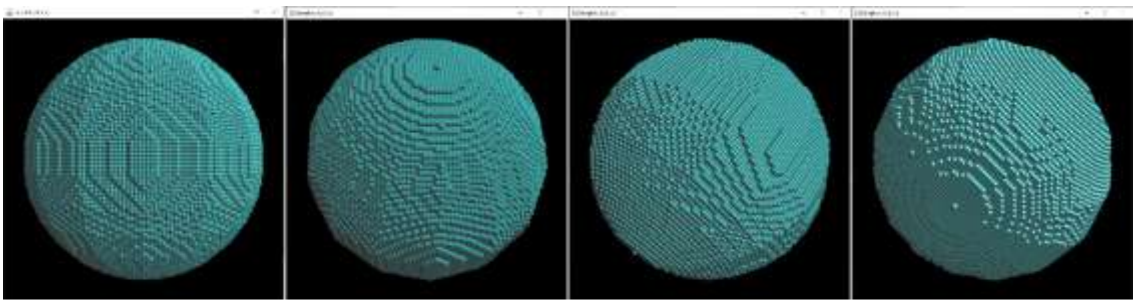

Fig. 26. Result (Sample data 4)

TABLE 5. Computation time (Sample data 1)

Number of nodes	Time [ms]				
	4 square	5 square	10 square	20 square	25 square
1	53883.8	51023.6	47440	46469.4	46515.4
4	13683.8	13149.6	12840.6	15681.6	12469.0
8	7104.0	6803.4	7417.6	13925.8	11946.4
16	3807.0	3703.2	4126.2	13295.0	11599.8

TABLE 6. Computation time (Sample data 2)

Number of nodes	Time [ms]				
	4 square	5 square	10 square	20 square	25 square
1	109494.0	106919.2	103369.8	102914.6	102612.8
4	27762.0	27103.2	26573.8	31268.0	27389.4
8	14257.2	13890.6	14334.2	19314.2	20430.0
16	7538.8	7484.8	7931.2	14192.0	20359.4

TABLE 7. Computation time (Sample data 3)

Number of nodes	Time [ms]				
	4 square	5 square	10 square	20 square	25 square
1	183240.8	180364.4	177244.6	176471.0	176460.2
4	45918.6	45303.0	44574.6	45558.0	55384.4
8	23256.0	22952.2	23702.0	30737.8	33526.0
16	11907.8	11903.2	15395.4	29354.8	31992.6

TABLE 8. Computation time (Sample data 4)

Number of nodes	Time [ms]				
	4 square	5 square	10 square	20 square	25 square
1	261480.2	257660.4	254542.0	254386.2	254348.0
4	65589.8	64859.6	64187.2	68765.0	67870.8
8	33094.2	32897.8	33994.2	36868.6	41042.0
16	17177.0	16863.0	18814.6	23316.6	27017.2

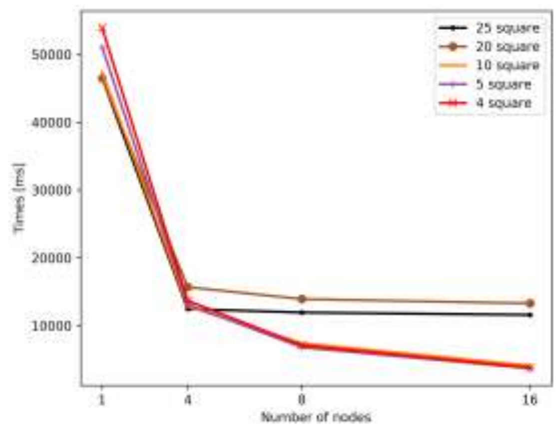

Fig. 27. Computation time (Sample data 1)

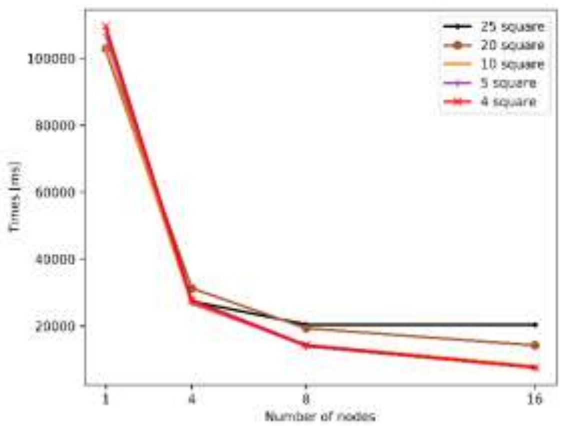

Fig. 28. Computation time (Sample data 2)

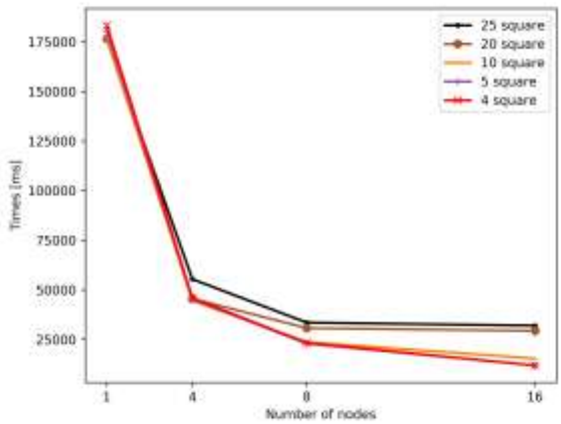

Fig. 29. Computation time (Sample data 3)

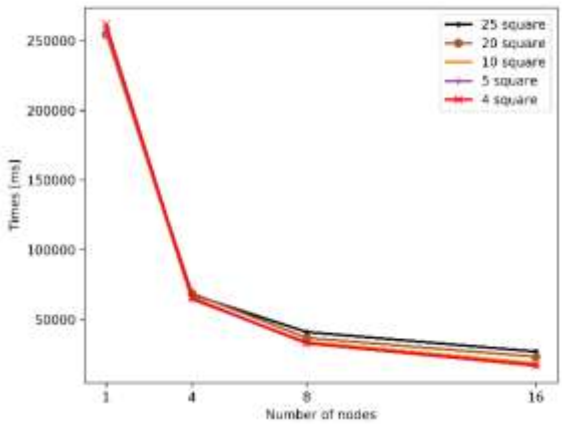

Fig. 30. Computation time(Sample data 4)

V. CONCLUSIONS

Determining the interior and exterior points of shapes and objects is important not only in fields such as 3D shape modeling and contact determination, but also in region detection and shape recognition. However, unlike 2D shapes, 3D shapes are described by huge amounts of data that seriously slow the determination process.

To reduce the processing time, shape data can often be processed in parallel. The present paper confirmed that parallel operations deliver high-speed result even on relatively low-performance devices such as Raspberry Pi 3 devices, which are used in the IoT field and are expected to be popularized in future.

If an algorithm solid angles is executed on the GPU, a large number of threads will guarantee high-speed results, but requires the creation of a program on the device side.

This study proposed that parallel processing can be implemented without the client being conscious of the network, and that the method implemented in each server can be called easily from the client side. In other words, various devices can be easily used without knowledge of the device side.

In future work, the allocation must consider the machine power of each server to boost the processing speed.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 19K14599.

REFERENCES

[1] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, Baoquan Chen, “Build-to-last: strength to weight 3D printed objects,” ACM Transactions on Graphics, DOI:10.1145/2601097.2601168, Vol. 33, No. 4, 2014.

[2] Munir Eragubi, “Slicing 3D CAD Model in STL Format and Laser Path Generation,” International Journal of Innovation, Management and Technology, Volume. 4, No. 4, 2013.

[3] Andrew Gleadall, Ian Ashcroft, JoelSegal, “VOLCO: A predictive model for 3D printed microarchitecture,” Additive Manufacturing, Volume 21, 2018.

[4] Abhijit Kundu, Yin Li, James M. Rehg, “3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare,” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3559-3568, 2018.

[5] S. Kodama, “Effectiveness of inside/outside determination in relation to 3D non-convex shapes using CUDA,” The Imaging Science Journal, DOI:10.1080/13682199.2018.1497251 Volume 66, Issue 7, 2018.

[6] Qian-Yi Zhou, Jaesik Park, Vladlen Koltun, "Open3D: A Modern Library for 3D Data Processing," arXiv:1801.09847, 2018.

[7] John Cheng, Max Grossman, Ty McKercher, Professional CUDA C Programming, Wrox Press Ltd., 9781118739327, 2014.

[8] Xun Gong, Rafael Ubal, David Kaeli, "Multi2Sim Kepler: A detailed architectural GPU simulator," 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), DOI: 10.1109/ISPASS.2017.7975298,pp. 269-278, 2017.

[9] Ilya V. Afanasyev, Vadim V. Voevodin, Vladimir V. Voevodin, Kazuhiko Komatsu, Hiroaki Kobayashi, "Analysis of Relationship Between SIMD-Processing Features Used in NVIDIA GPUs and NEC SX-Aurora TSUBASA Vector Processors," Parallel Computing Technologies, pp. 125-139, 978-3-030-25636-4, 2019.

[10] Scalable clusters make HPC R&D easy as Raspberry Pi, https://www.bitscope.com/cluster/bitscope-cluster-module-press-release-ZWLJ6PZ3.pdf.

[11] Dejan Vujičić, Dragana Mitrović, Siniša Ranđić, "Image Processing on Raspberry Pi Cluster," International Journal of Electrical Engineering and Computing, Vol.2, No 2, 2018.

[12] Vincent A. Cicirello, "Design, Configuration, Implementation, and Performance of a Simple 32 Core Raspberry Pi Cluster," arXiv:1708.05264, 2017.

[13] Dana A. Jacobsen, Julien C. Thibault, Inanc Senocak, "An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters," AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, DOI:10.2514/6.2010-522, 2010.

[14] J. Christy Jackson, V. Vijayakumar, Md. Abdul Quadir, C.Bharathi, "Survey on Programming Models and Environments for Cluster, Cloud, and Grid Computing that Defends Big Data," Procedia Computer Science, Volume 50, pp 517-523, 2015.

[15] Paul Heckbert, "Graphics Gems IV, Morgan Kaufmann," ISBN 978-0123361554, 1994.

[16] Kai Hormann, Alexander Agathos, "The point in polygon problem for arbitrary polygons," Computational Geometry, Volume 20, Issue 3, pp. 131-144, 2001.

[17] Dan Sunday, "Inclusion of a Point in a Polygon," http://geomalgorithms.com/a03-inclusion.html.

[18] Michael K. Reed, "Solid Model Acquisition from Range Imagery," Columbia University, http://crlab.cs.columbia.edu/files/solid-model-acquisition-from-range-imagery.pdf, 1998.

[19] Arsalan Malik, Benjamin Loriot, Youssef Bokhabrine, Patrick Gorria, Ralph Seulin, "A Simulation of Automatic 3D Acquisition and Post-processing Pipeline," Image Processing: Machine Vision Applications II, DOI: 10.1117/12.806148, Vol. 7251, 2009.

[20] Satoshi Kodama, Yuka Ozeki, Rei Nakagawa, "Internal and External Analysis Considering the Layers of Three-dimensional Shapes Using CUDA," International Journal of Computer Trends and Technology, Volume 67, Issue 6, DOI: 0.14445/22312803/IJCTT-V67I6P101, 2019.

[21] Atsushi Nakayama, Daisuke Kawakatsu, Ken-Ichi Kobori, Toshirou Kutsuwa, "A checking method for a point inside a polyhedron in grasping an object of VR," Information Processing Society of Japan, 48, pp. 297-298, 1994.

[22] Daisuke Kawakatsu, Atsushi Nakayama, Ken-Ichi Kobori, Toshirou Kutsuwa, "A Method of Selecting an Object in Virtual Reality," IPSJ SIG on CGVI, 110, 66-4, pp. 25-32, 1993.

[23] Adrian Kaehler, Gary Bradski, Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library, O'Reilly Media, 978-1491937990, 2017.

[24] Samuel D. Jaffee, Laura Marie Leventhal, Jordan Ringenberg, G. Michael Poor, "Interactive 3D Objects, Projections, and Touchscreens," Proceedings of the Technology, Mind, and Society, DOI: 10.1145/3183654.3183669, Article No. 18, 2018.

[25] Andrew Davison, "Pro Java 6 3D Game Development: Java 3D, JOGL, JInput and JOAL APIs," Apress, 978-1430211860, 2014.

[26] John Cheng, Max Grossman, Ty McKercher, "Professional CUDA C Programming," Wrox Press Ltd., 9781118739327, 2014.

[27] David B. Kirk, Wen-mei W. Hwu, "Programming Massively Parallel Processors," A Hands-on Approach 3rd Edition, Morgan Kaufmann, 9780128119860, 2016.

[28] Jason Sanders, Edward Kandrot, "CUDA by Example: An Introduction to General-Purpose GPU Programming," Addison-Wesley Professional, 978-0131387683, 2010.

[29] John Nickolls, GPU parallel computing architecture and CUDA programming model, IEEE Hot Chips 19 Symposium, DOI: 10.1109/HOTCHIPS.2007.7482491, 2007.

[30] Badr Benmammar, "Concurrent, Real-Time and Distributed Programming in Java: Threads, RTSJ and RMI," ISBN 978-1786302588, 2018.

[31] Paweł T. Wojciechowski, Konrad Siek, "Atomic RMI 2: distributed transactions for Java," Proceedings of the 6th International Workshop on Programming Based on Actors, Agents, and Decentralized Control, DOI: 10.1145/3001886.3001893, pp 61-69, 2016.

[32] University of Notre dame, "Java RMI",https://en.ppt-online.org/37724.

[33] Yutaro Hara, Satoshi Kodama, "A proposal for revising AR marker with infrared light," IEICE Tech. Rep., vol. 113, no. 299, pp. 53-56, 2013.

[34] Yuka Ozeki, Shinya Kameyama, Satoshi Kodama, Shigeo Akashi, A Proposal for the User Interface by Using Laser Devices Arranged in a Three Dimensional Space, The Institute of Electronics, Information and Communication Engineers and Information Processing Society of Japan, Volume 3, pp 385-388, 2016.