Omni-Directional Shortest Distance Algorithm by Complete Parallel-Processing Based on GPU Cores

Hiroshi Noborio[1*], Takahiro Kunii[2], Kiminori Mizushino[3]

[1] Osaka Electro-Communication University, Department of Computer Science, Kiyotaki 1130-70, 575-0063, Shijo-Nawate, Osaka, Japan.
[2] Kashina System Co. Hirata-Cho 116-22, 522-0041, Hikone, Shiga, Japan.
[3] Embedded Wings Co., Ine 5-2-3 562-0015, Minoh, Osaka, Japan.

* Corresponding author. Tel.: +81-72-876-5068; email: nobori@osakac.ac.jp
Manuscript submitted July 15, 2017; accepted August 28, 2017.
doi: 10.17706/ijbbb.2018.8.2.79-88

Abstract: We propose a new algorithm for calculating the 3D omni-directional minimum distance from the cavitron ultrasonic surgical aspirator (CUSA) tip to blood vessels. The distance is selected from many shorter distances calculated by the GPU cores for many pixels. First, we use z-buffering (depth buffering) as the classic matured function of the GPU to effectively obtain depths (the distances to blood vessels) corresponding to many pixels. Second, we calculate the Euclidean distance from the scalpel tip to the closest z-values of the depths by multiple GPU cores for all pixels. The many pixels are prepared within a cubic region overlapped by six rectangular parallelepipeds along the +X, -X, +Y, -Y, +Z, and -Z axes centered at the CUSA tip. In this algorithm, all cores are not distinguished within six cubes. Finally, we evaluate the algorithm performance with regard to calculation time and visual reality using an inexpensive GPU (GTX950). Our experimental results show that the calculation time is twice as fast, and the visual reality is also improved.

Key words: CUSA (cavitron ultrasonic surgical aspirator), DICOM (digital imaging and communication in medicine), GPGPU (general-purpose graphics processing unit), STL (STereo-lithographies), Z-buffering (depth buffering).

1. Introduction

Many researchers have proposed various accelerating algorithms based on a General-Purpose Graphics Processing Unit (GPGPU) used in computer vision, 3D structure modeling, 3D simulators, sorting, databases and so on [1]-[4]. Especially, concerning to shortest distance problem, GPU-based parallel calculation is frequently used for selecting the shortest path among discrete graph [5]-[8]. Regarding the calculation of the distance and/or intersection detection between a point and an object or multiple objects, a GPU has the following advantages: 1) quick digitalization of all objects by z-buffering, which is the classic matured function of a GPU; 2) fast calculation of the minimum distance and/or volume intersection between a point and an object or multiple objects using multi-cores of the GPU in parallel; and 3) free calculation and synchronous selection of the minimum distance from shorter distances along many directions from many viewpoints using multi-cores of the GPU in parallel.

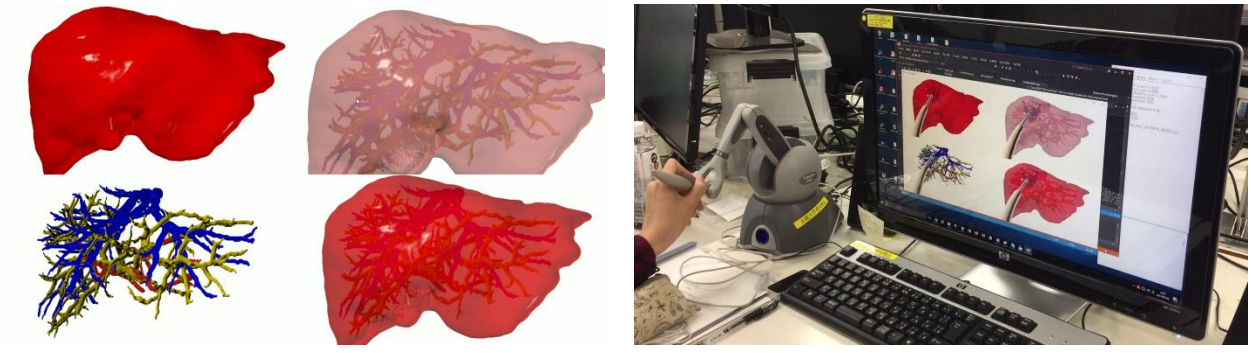
If each stereo-lithography (STL) is processed by the z-buffering of the GPU, a z-value reaching the boundary of the STL is individually calculated for each pixel. If the z-value at each pixel indicates that the

STL's patch is closer to the viewer than the z-value in the z-buffer, the z-value memorized in the buffer is changed by the STL's value. Moreover, the z-value of the closest patch can be preserved through the GPU background removal function. As a result, a cuboid can be obtained with the width and pixilation calculated using the z-value of the surface and the reverse side of the polyhedron, resulting in a cuboid digital approximation of the polyhedra. Therefore, we can calculate the Euclidean distances from the tip of the scalpel to the rectangular parallelepipeds in parallel using multi-cores of the GPU, the smallest of which represents the shortest distance for one arbitrary direction [9].

The superimposition calculation of the liver, the three types of blood vessel cuboids, and the cavitron ultrasonic surgical aspirator (CUSA) scalpel cuboids can be performed instantly by the GPU, thus enabling a rapid calculation of the embedded distance and the embedded regions. From this embedding, for example, an artificial sense of touch is constructed with the Kelvin-Voigt model, which can be experienced through a tactile feedback device. Furthermore, the polyhedra can be rapidly transformed in response to the embedded region, allowing the concave region to become visible [10]-[12].

On the basis of the above mentioned pre-processing, we calculated the shortest distance from the CUSA tip to the three types of blood vessels (portal veins, arteries, and veins) along the CUSA's moving direction, having a single dimension [9]. In this omni-directional algorithm, the previously revised algorithm calculates the shorter Euclidean distances six times individually around the CUSA tip along the X, -X, Y, -Y, Z and -Z axes and then selects the shortest distance from the six shorter ones [13]. The algorithm's performance is dammed by the individual processing of six cubes. To overcome this defective point, we employ a completely parallel processing algorithm for all rectangular parallelepipeds within the six directional cubes. As a result, the universal algorithm always selects the omni-directional distance two times faster than the previously proposed algorithm.

Using the shortest distances to portal veins, arteries, vein blood vessels, and the liver tumor, a doctor can avoid cutting blood vessels as well as maintain a constant distance from the liver tumor using the CUSA, and consequently, they can protect the quality of postoperative life of the patient. This approach has already been incorporated into our navigation software used in liver surgeries [14]. In the future, we hope to superimpose real and virtual livers using several techniques of Mixed Reality.



(a) (b)

Fig. 1. (a) STL (stereo-lithography) liver with blood vessels converted from patient DICOM data, (b) a human operates a CUSA scalpel STL on the blood vessels using the Phantom Omni force-feedback device.

The rest of this paper is organized as follows: Section 2 describes several kinds of STLs such as a whole liver, blood vessels, and CUSA surgery scalpels. Also, we newly propose our GPU-based 3D-omindirectional algorithm in order to calculate the shortest distance from the CUSA tip to the three types of blood vessels. Our algorithm is written in pseudo language and includes a sequential bit count (selection of the minimum distance from many calculated distances for a lot of pixels) via many cores of the GPU working in parallel.

Because of the parallel processing, the computational time of the algorithm is always fixed, even when the surface number of STLs increases, thus making minor distance errors a possibility. In section 3, we evaluate our GPU-based 3D-omindirectional algorithm using an inexpensive GPU (GTX950). Finally, a summary of the study is provided in Section 4.

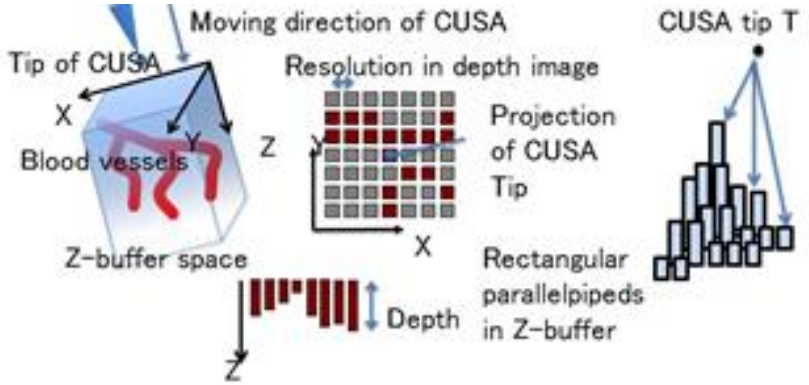


Fig. 2. The original concept used to calculate the one-directional shortest distance in the digitalized 3D space made via GPU's z-buffering.

2. Materials and Methods

Before using our algorithm, we captured the Digital Imaging and Communication in Medicine (DICOM) data of the liver of a patient through magnetic resonance imaging (MRI) or computed tomography (CT). We then converted the DICOM data into a polyhedron in STL format. The purpose of using STL is to ensure excellent visual quality, and to rapidly calculate a depth image using z-buffering of the GPU (Fig. 1(a)). In our liver surgical navigation study [13], we selected the tip motion of the CUSA by using a haptics (position/force-feedback) device (Fig. 1(b)).

A year ago, we first proposed the one-directional shortest Euclidean distance calculation algorithm [9]. We developed this GPU-based algorithm for calculating the distance and/or intersection detection between a point and an object or multiple objects to solve a significant problem with the classic CPU-based algorithm. The calculation time of the classic CPU-based algorithm proportionally depends on the number of object patches, i.e., the shape complexity of the target objects (polyhedrons) [15].

Contrasted with this, the calculation time of our GPU-based algorithm is inversely proportional to the core count. The calculation time of our GPU-based algorithm is always constant, and does not depend on the number of object patches, i.e., the shape complexity of the target objects (polyhedrons). The basic idea of our GPU-based algorithm is as follows: As shown in Fig. 2, all liver parts (portal veins, arteries, vein blood vessels, and the liver tumor) are independently digitized into a set of rectangular parallelepipeds. Then, all the Euclidean distances from the CUSA tip T to all the closest endpoints (z-values in the Z-buffer) of the rectangular parallelepipeds within the six digitized cubes and (x,y) values in depth image are calculated by the GPUs multi-cores in parallel.

Fig. 3 presents the application of the one-directional algorithm in an arbitrary surgical simulation and/or navigation in the 2D environment. This 3D version was tested in a virtual environment, including the CUSA and blood vessel STL [9]. We used a sequence of bits that corresponded to each digitized distance. Fig. 4 presents the application of the omni-directional algorithm in an arbitrary surgical simulation and/or navigation in the 2D environment. This 3D version was tested in a virtual environment, including the CUSA and blood vessel STL [13].

These algorithms use a sequence of bits that correspond to each digitized distance. Each GPU core sets an

arbitrary bit corresponding to its calculated Euclidean distance in parallel. First, all bits are initially set to 0, and bit setting is then simultaneously performed by all active cores. For this reason, we escaped exclusive control concerning the shortest distance selection by using multiple cores. In Fig. 5, all the bits are in common memory, and are renewed by all GPU cores in parallel processing.

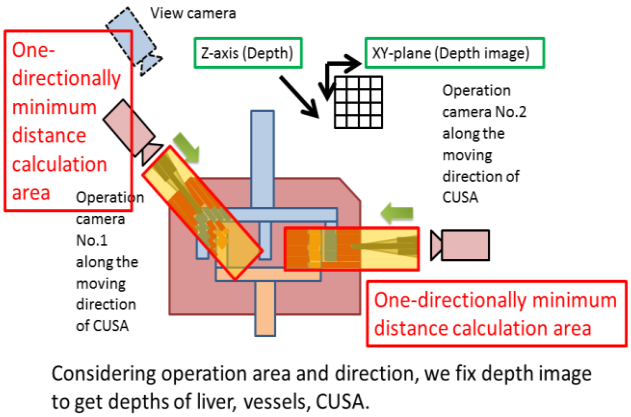


Fig. 3. In 2D space, we explain the one-directional shortest Euclidean distance calculation algorithm. We indicate the allocation of the CUSA scalpel, its moving direction, and a cube calculating different distances of all liver parts (portal veins, arteries, vein blood vessels, and liver tumor). In the distance-calculating cube, all liver parts are independently digitized into a set of rectangular parallelepipeds.

If the measuring range is defined within [0 mm, 50 mm], and the common memory is 50 bits, then each bit corresponds to 1 mm. Moreover, the precision of the calculated depth along the Z-axis is 50 mm/16*n (n being an arbitrarily selected value). If n = 1, the distance error along the Z-axis from this division would be 50 mm/16 = about 3 mm. If n = 16, then the distance error along the Z-axis from the division would be 50 mm/256 = about 0.2 mm.

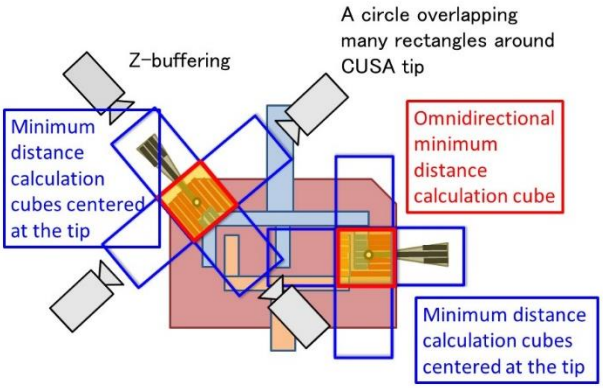


Fig. 4. In 2D space, we explain the omni-directional shortest Euclidean distance calculation algorithm. We indicate the allocation of the CUSA scalpel, its moving direction, and a cube calculating different distances of all liver parts (portal veins, arteries, vein blood vessels, and liver tumor). In the distance-calculating cube, all liver parts are independently digitized into a set of rectangular parallelepipeds.

In our simulation and navigation, we used cubes with sizes of 106 mm, 106 mm, and 213 mm along the X, Y, and Z-axes, respectively. In addition, the XY image resolutions of the z-buffer are 2048*2048 pixels. In the z-buffer, a 32- bit variable records the z-value. As a result, an error of about 0.05 nm (=213 mm/232) along the Z-axis and 52 nm (=106 mm/2048) in the XY image were generated. Consequently, a total error of 73.5

mm appeared in the XYZ space. This is a small issue in our GPU-based algorithm, but doctors have informed us that a distance error of a few mm is not a problem in actual surgical operations.

Recently, we extended the one-directional algorithm to an omni-directional algorithm whose six digitized cubes are independently processed [13]. By overlapping six digitized cubes that individually calculate the one-dimensional shortest Euclidean distance, we can obtain six shorter Euclidean distances, and select the minimum as the shortest Euclidean distance. In addition, we compared the computational costs of one-dimensional and omni-directional algorithms using the NVIDIA GeForce GTX 950, GTX 960, GTX 970, and GTX 980. As a result, the omni-directional algorithm was found to be ten times or more slower than the one-directional algorithm. In addition, the calculation time of the omni-directional algorithm was found to average 7, 6, 4, and 3.5 milli second, respectively. This means that the switching calculation cost between six digitized cubes is not negligible.

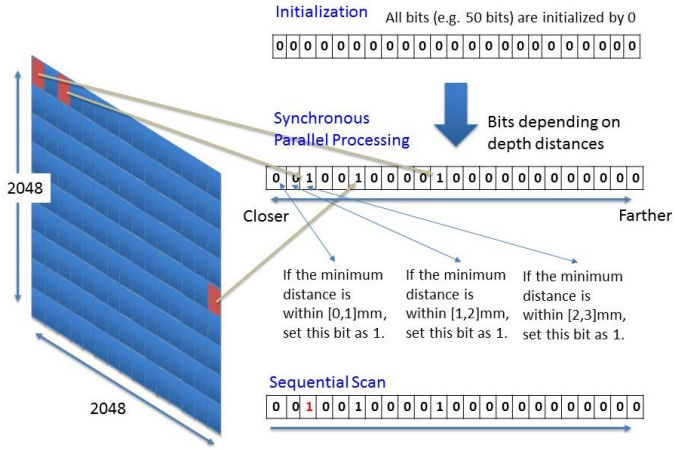


Fig. 5. A sequence of bits is prepared for all the distances of digitized depths.

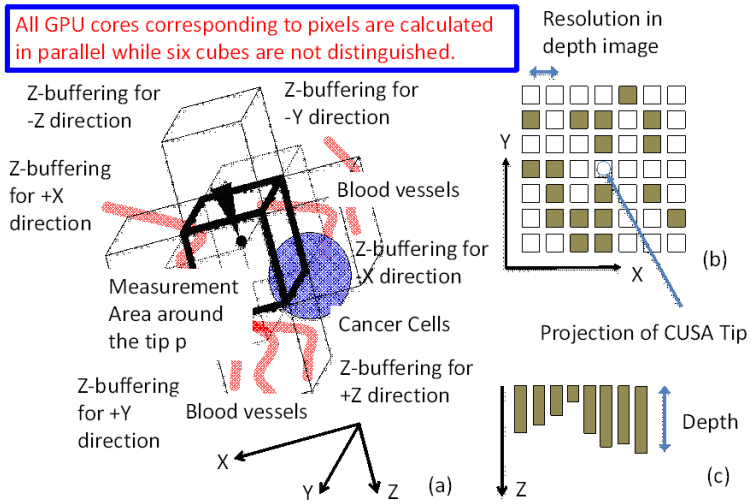


Fig. 6. The new algorithm that calculates the shortest distance in parallel from all the Euclidean distances generated by triangulation, based on all pixels and their corresponding depths in six digitized cubes at the same time by GPU-based multi-cores, whose number is about 1000

For example, for an interval [0 mm, 50 mm], we prepare 50 bits which each correspond to 1 mm (Fig. 5). In order to eliminate this switching cost, we propose a new algorithm that calculates the shortest distance in parallel from all the Euclidean distances generated by triangulation, based on all pixels and their

corresponding depths in six digitized cubes at the same time (Fig. 6). We precisely describe a new omni-directional shortest distance algorithm for parallel calculations of six digitized cubes by the following pseudo code:

First, the following procedure is simultaneously used for all (one in [9]) the rectangular parallelepipeds within the six digitized cubes around the CUSA tip along the X, -X, Y, -Y, Z and -Z axes:

An array z-buffer[x,y] initialized to the maximum value

begin /* We simultaneously process z-buffering for 6 direction planes */ do

A vector d-bits [6] [division-count+1] initialized to 0

/* We receive z-buffers of 6 directions, and process the following procedures in parallel for all pixels with (x,y) coordinates. */

(for each pixel (x,y) in 6 planes that intersects P calculate distance of P at (x,y) in plane of plane-number;

/* distance is normalized and digitized as bit-number within 0 - division-count */

calculate bit-number from distance;

d-bits[plane-number][bit-number] = 1

nearest-xy[plane-number][bit-number] = (x, y)

)

/* We determine the closest set of coordinates by comparing d-bits calculated by the GPU for 6 planes. */

nearest-distance =

(for each plane-number from 0 to 5

/* We sequentially search for one-bit within d-bits whose numbers are 0 - division-count, and then reset its index of one-bit as the bit-number*/

scan d-bits[plane-number][division-count+1] sequentially from left to right in order to find an initial bit

set bit-number to index of d-bits[plane-number][division-count+1] that found an initial bit;

/* According to the bit-number, we retrieve the distance. */

calculate distance from bit-number;

if distance nearest-distance then (nearest-distance = distance; nearest-xy = nearest-xy[plane-number][bit-number]; nearest-plane-number = plane-number;

)

)

if nearest-distance = then use nearest-distance, nearest-xy, nearest-plane-number;

Multi-cores for all pixels along six directions simultaneously calculate shorter distances and consequently reset specific bits, which establishes the existence of their distances (Fig. 7). If we recode not only the bit set, but also the closest XYZ value at the same time, we separately record multiple vectors expressed at the proximity point from the scalpel tip, which determines this distance. Because every bit is initialized for each motion, a bit value of 0 means no existence of its corresponding distance within the 3-D omni-directional cubic space.

If all arrays are initialized to 0 before beginning the calculation processes for all pixels in the six directions by the GPUs multi-cores, by searching the arrays from 0 after the parallel processing is finished, we omni-directionally convert the first 1's distance as the shortest distance. The equation is as follows: the shortest distance = the intermediate value of the distance range corresponding to the array number (Fig. 7). We also obtain the closest XYZ value around the blood vessels in addition to the shortest distance. Using the vector of the CUSA tip and its closest point for the three kinds of blood vessels or the liver tumor, a surgeon can view the optimal motion of the CUSA in our liver surgical navigation study [14].

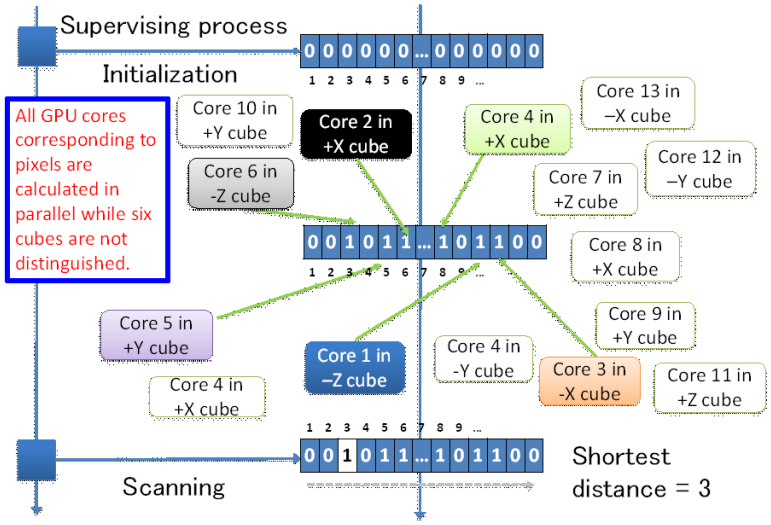


Fig. 7. After the initialization of bit sequence, multi-cores for all pixels along six directions calculate shorter distances in parallel and then reset specific bits, which establishes the existence of their distances. After that, by scanning the bit sequence from right to left, we can find the first 1's distance as the shortest distance.

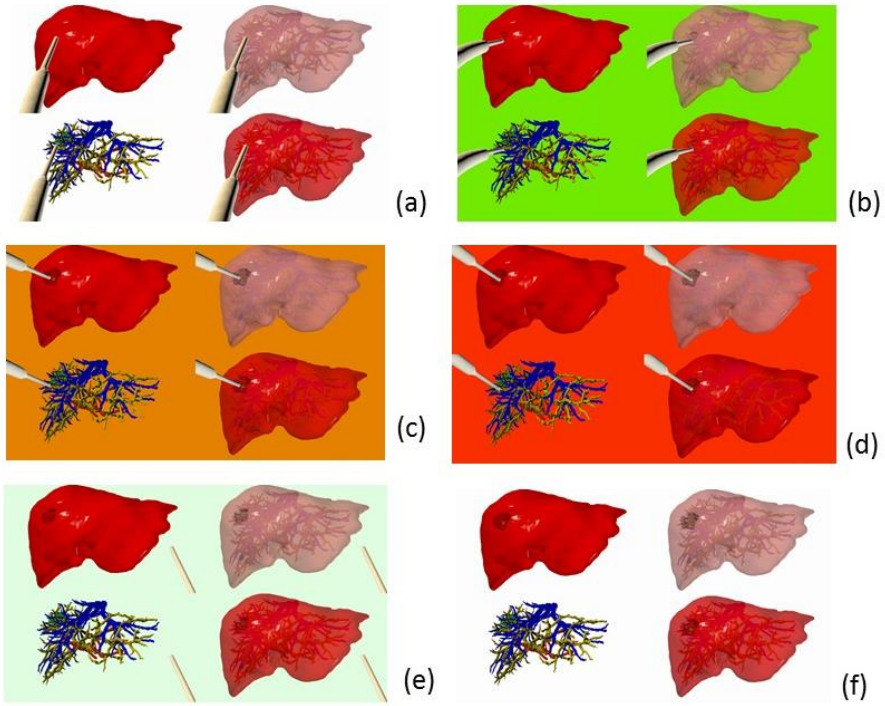


Fig. 8. We show a sequence of CUSA motions for cutting a liver tumor by the CUSA from (a) to (f) successively. This is prepared for evaluating the 1D and 3D shortest distance algorithms using the NVIDIA GeForce GTX950.

3. Results

For the evaluation, we used a smart sequence of CUSA motions for cutting a virtual liver in real time. The virtual liver was represented by plastic material without deformation. When a surgeon cut a plastic liver, the vertices around the cutting area were moved without adding any patch in the virtual liver. In Fig. 8, we show the six successive stroboscope shots. For this surgical operation sequence, we obtained the

calculation results using the NVIDIA GeForce GTX950. As shown in Fig. 9, the calculation speed was two times or more fast than our previous omni-directional shortest distance algorithm. These results were achieved by two trials as follows: (1) the shortest Euclidean distance was calculated in parallel from all the rectangular parallelepipeds within six digitized worlds until the GPU cores were completely used up; (2) we changed M and N by trial and error: the thread count was M * M and the block count was width/M height/M. When the image resolution was N * N, a parallel calculation was performed with a thread count of M M and a block count of N/M N/M. Finally, the following fundamental issues were still left: 1) memory transfer time and transfer time from the GPU to the CPU (parameters and results), and 2) timing; GPU screen update,

This calculation takes 3 milli seconds at most in all scenarios of surgical navigation (Fig. 9). Therefore, a surgeon can always identify useful information. In surgical navigation, portal veins, arteries, vein blood vessels, and the liver tumor are distinguished by different colors. The minimum distance to portal veins, arteries, and vein blood vessels is used for avoiding an incorrect cut. A patient's life is protected by this function during the operation.

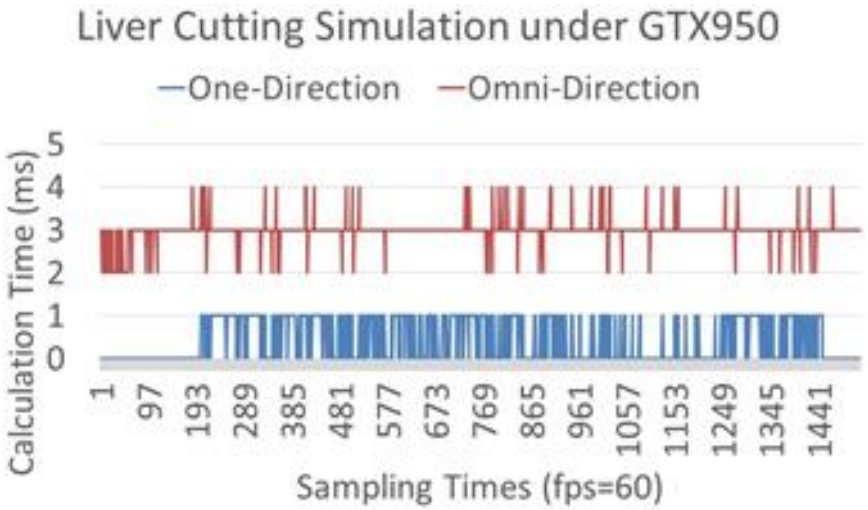


Fig. 9. Comparing calculation costs of one-dimensional and omni-directional, completely parallel algorithms using the NVIDIA GeForce GTX 950.

4. Conclusion

In this paper, we revised the previous omni-directional shortest distance algorithm for serial calculations of six digitized cubes and consequently constructed a new omni-directional shortest distance algorithm for parallel calculations of six digitized cubes. We then compared the calculation costs of the one-directional and omni-directional algorithms using the NVIDIA GeForce GTX 950, GTX 960, GTX 970, and GTX 980. We found that the omni-directional, completely parallel algorithm is always three times slower than the one-directional algorithm. If six cube divisions appear, the difference becomes six times or more. Our revised algorithm uses the computation power of the NVIDIA GeForce GTX series by completely eliminating the six cube divisions. The calculation time of the omni- directional, completely parallel algorithm is 3 milli seconds using the NVIDIA GeForce GTX 950. This is two times faster than the previous omni-directional, partially parallel algorithm using the GTX 950. This means that the omni-directional, completely parallel algorithm thoroughly uses the full power of the GPU.

Acknowledgment

This research has been partially supported by the Collaborative Research Fund for Graduate Schools (A)

of the Osaka Electro-Communication University, and a Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology (Research Project Number: JP26289069).

References

[1] Miura, M., Fudano, K., Ito, K., Aoki, T., Takizawa, H., & Kobayashi, H. (2013). Performance evaluation of phase-based correspondence matching on GPUs. *Proceedings of the SPIE 8856, Applications of Digital Image Processing*.

[2] Green, O., McCol,l R., & Bader, D. A. (2012). GPU merge path - A GPU merging algorithm. *Proceedings of the 26th ACM International Conference on Supercomputing* (pp. 331 - 340). San Servolo Island, Venice.

[3] Taylor, Z. A., Cheng, M., & Ourselin, S. (2012). Neurosurgery simulation using non-linear finite element modeling and haptic interaction. *Proceedings of the SPIE Int. Soc. Opt. Eng*.

[4] Smith, T. F., & Waterman, M. S. (2008). High-speed nonlinear finite element analysis for sur- gical simulation using graphics processing units. *IEEE Trans Med Imaging, 27(5)*, 650-663.

[5] Mintal, M. (2012). Accelerating distance matrix calculations utilizing GPU. *Journal of Information, Control and Management Systems*, *10(1)*, 71-80.

[6] Chapuis, G., Djidjev, H., Andonov, R., Thulasidasan, S., & Lavenier, D. (2014). Efficient multi-GPU algorithm for all-pairs shortest paths. *Proceedings of the 28th IEEE International Parallel & Distributed Processing Symposium*.

[7] Okuyama, T., Ino, F., & Hagihara, K. (2012). A task parallel algorithm for finding all-pairs shortest paths using the GPU. *International Journal of High Performance Computing and Networking*, *7(2)*, 87-98.

[8] Matsumoto, K., Nakasato, N., & Sedukhin, S. G. (2012). Blocked united algorithm for the all-pairs shortest paths problem on hybrid CPU-GPU systems. *IEICE Trans. on Information and Systems*, *95(12)*, 2759-2768.

[9] Noborio, H., Kunii, T., & Mizushino, K. (2016). Comparison of GPU-based and CPU-based algorithms for determining the minimum distance between a CUSA scalper and blood vessels. *The SCITEPRESS Digital Library,* 128-136.

[10] Noborio, H., Onishi, K., Koeda, M., Mizushino, K., Kunii, T., Kaibori, M., Kon, M., & Chen, Y.-W. (2016). Fast surgical algorithm for cutting with liver standard triangulation language format using z-buffers in graphics processing unit, masakatsu fujie (Ed.), *Computer Aided Surgery,* 127-140.

[11] Onishi, K., Mizushino, K., Noborio, H., & Koeda, M. (2014). Haptic AR dental simulator using Z-buffer for object deformation. *Universal Access in Human-Computer Interaction. Aging and Assistive Environments,* 342-348.

[12] Onishi, K., Noborio, H., Koeda, M., Watanabe, K., Mizushino, K., Kunii, T., Kaibori, M., Matsui, K., & Kon, M. (2015). Virtual liver surgical simulator by using Z-buffer for object deformation. *Universal Access in Human-Computer Interaction,* 345-351.

[13] Noborio, H., Kunii, T., & Mizushino, K. (2016). CPU-based omni-directional shortest distance algorithm and Its evaluation by changing GPU cores. *Proceedings of the 13th International Conference of Computational Intelligence methods for Bioinformatics and Biostatistics* (pp.76-81).

[14] Noborio, H., Aoki, K., Kunii, T., & Mizushino, K. (2016). A potential function-based scalpel navigation method that avoids blood vessel groups during excision of cancerous tissue. *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (pp.6106-6112).

[15] Lin, M., & Canny, J. A. (1991). Fast algorithm for incremental distance calculation. *Proceedings of the 1991 IEEE Robotics and Automation* (pp. 1008-1014).

Hiroshi Noborio was born in Osaka, Japan, November 4, 1958. He graduated at the Department of Computer Science, Shizuoka University, Hamamatsu, Japan, and received the Dr.Eng. degree from the Department of Mechanical Engineering, Osaka University, Toyonaka, Japan.

From 1987 to 1988, he was an assistant professor at Osaka University. From 1988, he joined a lecturer at Osaka Electro-Communication University. From 2009 to 2012, he was the dean of the Faculty of Information Science and Arts in OECU. Now, he is a professor at Department of Computer Science now.

Prof. Noborio is recently interested in surgical simulation and navigation in the medical and dental areas. Professor Noborio is member of Japanese Society for Medical Virtual Reality, the Japan Society of Computer Aided Surgery, the Virtual Reality Society of Japan, Robotics Society of Japan, the Society of Instrument and Control Engineers, Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers and IEEE.

Kiminori Mizushino was born in Osaka, Japan, October 27, 1986. He graduated at the Department of Computer Science, Osaka Electro-Communication University, Shijo-Nawate, Japan, and received the master engineering degree from the Division of Computer Science, Faculty of Information Science and Arts, Osaka Electro-Communication University, Shijo-Nawate, Japan.

After graduated, he established the Embedded System Co. and now he is the president & CEO of Embedded System Co. Also, he is a part-time lecturer of the Department of Computer Science, Osaka Electro-Communication University.

Mr. Mizushino is recently interested in several kinds of computer systems, software, educations including surgical simulation and navigation in the medical and dental areas.

Takahiro Kunii was born in Kagawa, Japan, March 6, 1966. He graduated at the Department of Precision Engineering, Faculty of Engineering, Osaka Electro-Communication University, Neyagawa, Japan.

From 1990 to 1997, he joined TECHNICAL GROUP LABORATORY, INC. After that, he joined Kashina System Co., Hikone, Shiga, Japan, and now he is vice-president & CEO of Kashina System Co.

Mr. Kunii is recently interested in several kinds of programming languages, computer software including surgical simulation and navigation in the medical and dental areas.