SICE Annual Conference 2008
August 20-22, 2008, The University Electro-Communications, Japan

High speed 3-D registration using GPU

Yasuo KITAAKI, Haruhisa OKUDA, Hiroshi KAGE, Kazuhiko SUMI
Mitsubishi Electric Corporation Advanced Technology R&D Center

Abstract

This paper describes high speed 3-D object recognition based on DAI(Depth Aspect Image) matching and M-ICP（Modified Iterative Closest Point). We regards GPU(Graphic Processing Units) as coprocessor which are capable of computation for general purpose. We proposed 3-D object recognition method which consists of two step pose estimation and positioning, i.e. the DAI matching for coarse step and HM-ICP (Hierarchical M-ICP) for fine one Our method on GPU which has remarkable performance for parallel computation. The experimental results show the effectiveness of our method. This method can process 2 or 3 times faster than the original one, although the calculation amount of this method is at least 20 times bigger than the original one. Additionally, its processing time is more stabler than original method.

1 Introduction

When we apply robots into the factory automation, the number of vision applications has been increasing and the needs for 3-D object recognition technology is especially strong. We proposed the 3-D object recognition technique which consists of two steps. The first step for the coarse positioning and pose estimation uses the DAI (Depth Aspect Image) matching[1]. The second step uses HM-ICP(Hierarchical M-ICP)[2] based on the ICP (Iterative Closest Point) algorithm [3] for precise 3-D registration after the 1st step. Through the experimental result, this paper shows GPU(Graphic Processing Units) application for each step to speed up. CPU is optimized for high performance on sequential task. On the other hand GPU has high performance of parallel computation because it is desined for specific graphic processing. We apply GPU as coprocessor to speed up of 3-D object recognition method.

Nvidia developed Cg as a high-level shading language. Microsoft developed HLSL for use with Microsoft Direct3D API. OpenGL ARB created GLSL to developers more direct control of graphics pipeline unless using assembly language or hardware-specific languages. Therefore, recent GPU can deal with not only graphic task but also general purpose calculation. For example, some reserchers are trying to apply the GPU to fluid dynamics simulator , rendering of 3-D object for madical image, and so on. But These methods are required knowledge about graphics hardware and higl level shader language because Cg, HLSL and GLSL are desined not for general purpose computation but for specific graphic task. As development environment for GPU computing, Nvidia has released CUDA technology[4]. CUDA solve complex coding problems, it's faster than Cg, HLSL and GLSL. This development environment is desined to have developers be capable of programing without any knowledge about specific graphics, GPU architecture and hardware. In this paper, we apply CUDA to shorten up the processing speed of our method. The paper consists of five sections. We introduce basic knowledge of CUDA architecture. Then, 3-D registration algorithm we proposed is explained in section 3, and experimental results are shown in section 4. Finally, we concludes in section 5.

2 Programming using CUDA

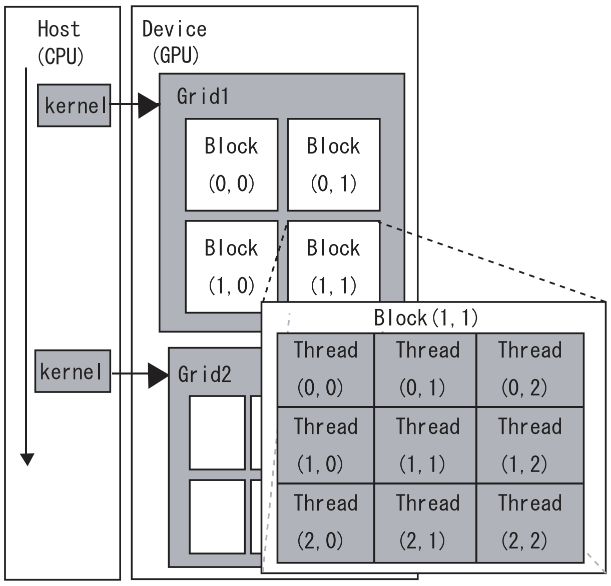


Figure 1: execution model

When we use CUDA programming, we can make programs much easier than traditional general purpose computation on GPU because it's the extension of the standard C language. For example, when developers use traditional development environment for GPGPU (General purpose computation on GPU)[5], the data must be inputed as texture data which are represented with R, G, B, α and the process must

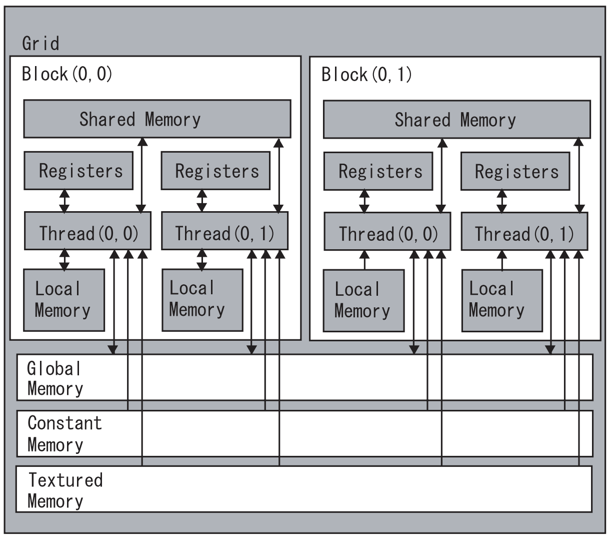


Figure 2: memory model

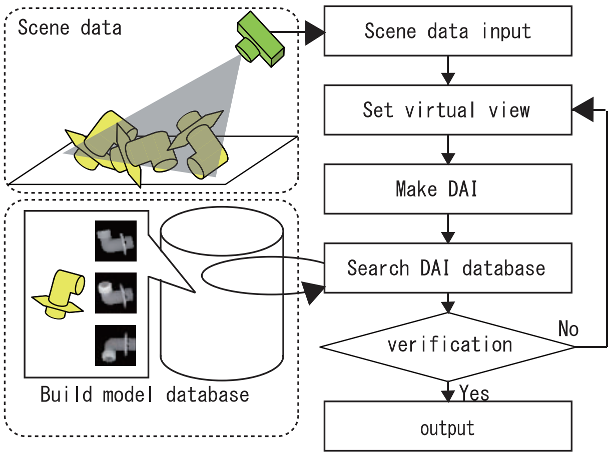


Figure 3: DAI matching algorithm

be concerned with graphic task like rendering. But when developers use CUDA, it enable them to programming without considering all these restriction.

2.1 Thread execution

A grid which consists of thread blocks is executed as illustrated in fig.1[4]. Each blocks consist of a batch of threads and have shared memory region among threads. Batched blocks which has same dimentionality can be regard as one grid of thread blocks. They are executed by a single kernel.

2.2 Memory model

Fig.2 shows CUDA memory model. They are divided into following three.

- register and local memory by each threads.
- shared memory by each blocks.
- global memory, constant memory and texture memory by each grids.

Processing time of applications developed with CUDA is depended on the use of these memory because each memory type are specialized in different purposes. When proper applied, processing time is much faster even if handling data size is larger than CPU's. Especially, shared memory is more important than others for faster parallel computing. Programmer should consider a restriction of shared memory size and the bandwidth of each memory. Access speed of shared memory is very fast, but the memory size is only 16 KB.

3 Object recognition algorithm

In this chapter, we explain 3-D object recognition method which consists of two steps pose estimation and positioning, i.e. the DAI matching for coarse step and the HM-ICP for the fine one.

3.1 DAI matching

The flow of the processing is shown in fig.3. First step, we build the 3-D view model database called DAI(Depth Aspect Image) which is generated from the virtual viewpoint. After building the database, we seek best matching position and pose of the target based on the correlation between the target view and the DAI database view.

3.2 HM-ICP

ICP algorithm is widely used for geometric alignment of 3-D models. ICP starts with two 3-D data and an initial guess for their relative rigid body transform, and refines the transform by generating pairs of corresponding points, minimizing distance of pairs and the update repeatedly. Modified ICP(M-ICP) by M-estimaor for the robustness was proposed[6]. We proposed Hierarchical M-ICP(HM-ICP) which was improved based on M-ICP by two techniques. One is hierarchical registration processing using multiple resolution data and the second is using partial regions which are selected based on the feature values for registration. We achieved more than 4-digits number reduction of the computational cost by these algorithm approaches.

4 Parallel computation on GPU

In this chapter, the parallel computing technique on GPU is described.

4.1 Implementation of the DAI method

Fig4(a) shows the sequential computing model on CPU and fig4(b) shows parallel one on GPU for the DAI matching step. Main roop of this method implemented on CPU consists of following four parts.

(A) Set virtual view

(B) Make DAI

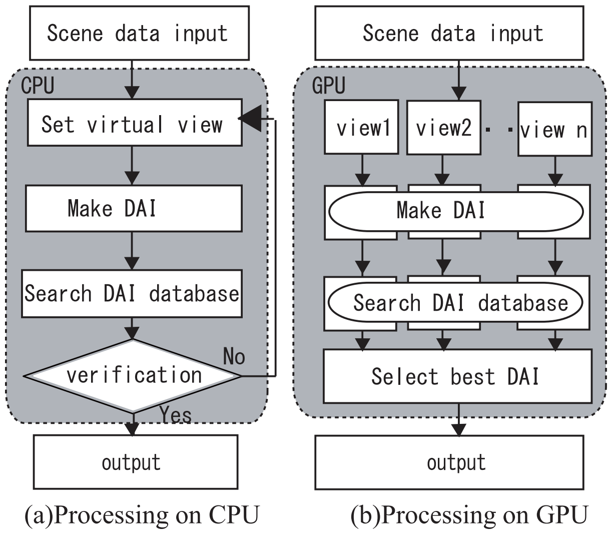


Figure 4: Computation model of DAI matching

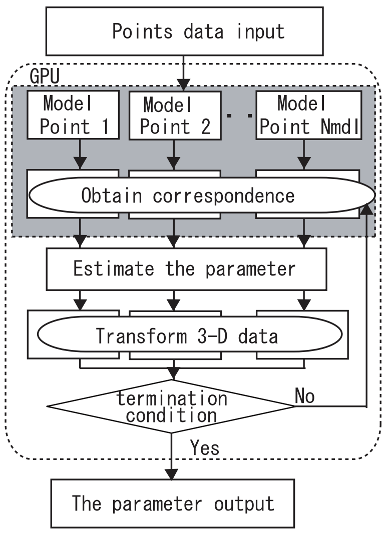


Figure 5: HM-ICP algorithm

(C) Search DAI database

(D) Verification

It's executed repeatedly until finding DAI which can pass the verification. Virtual view is set to scene data and corresponding scene DAI is created. The scene DAI is matched with model DAI database. In the case of that succeeds, it enables to estimate coarse position and pose from virtual views to create each DAIs and recognition phase will go to next step for verification. If the matching fails, next view is set in scene for created another DAI. In this method, the processing time is depended on a number of created DAI until the success.

We explain the process which was implemented on the development environment, CUDA. Before scene data measured, model DAI database is inputed into GPU memory. When scene inputed, it is executed with parallel that all processing which contains making DAI, searching for best matching DAI and calculating evaluation value. The bottleneck of GPGPU is the transfer time between GPU and CPU, especially the transfer GPU to CPU is much slower than CPU to GPU because GPU is designed for specific graphics originally. Therefore, we limited a number of times of the transfer between CPU and GPU.

4.2 Implementaition of the HM-ICP

We implemented only M-ICP algorithm on GPU because the part accounts for 90 percent of the computing time of HM-ICP. The ICP algorithm can be divided into three process -

(A) Obtain correspondence between scene and model.

(B) Estimate the parameter using a mean square error.

(C) Transform 3-D data by the estimated parameter.

The computational cost of (A) part is much larger than other's one. Therefore, we explain the implementation of (A) on GPU. The ICP algorithm estimates the corresponding of each data point between 3-D data sets A and B. Then, it calculates at the $N_a \times N_b$ time if supposing that the points number of data A and the points number of data B is N_b. When (A) processing on GPU, GPU can process points of data A in parallel because it is executed in the same way that searching for closest point of data B from each points of data A. This processing is very effective to reduce computational cost. We execute distance calculation for obtain correspondence from each points of data A to all the points of data B in parallel(fig.5). It handles only N_a points data in parallel for limitation of memory size. If memory size permits, it handles both points data in parallel. We limits the transfer between CPU and GPU, especially GPU to CPU. In our implementation, only seven parameter are returned to CPU after all processing which consist of (A), (B) and (C) on GPU.

5 Experiments

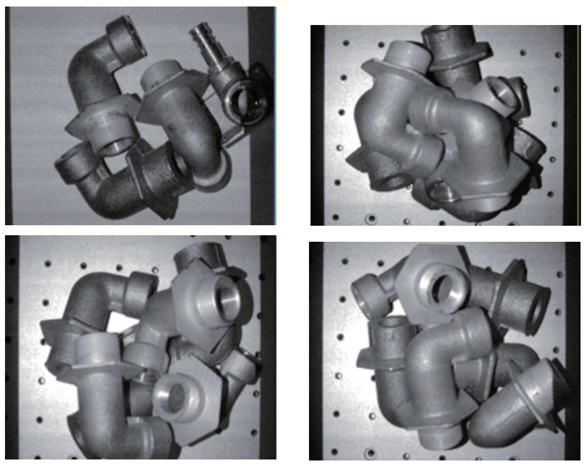
We experimented to verify the effectiveness of the our method which was described in the former section. We verify the processing time. The target object is shown in fig.**??**.

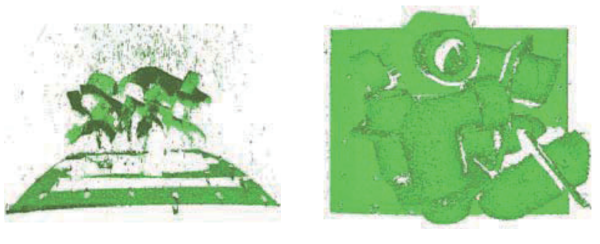
Table 1: PC Specification

CPU	Intel Core 2 Duo 2.6GHz
Memory	4GB
GPU	GeForce8800GTX(G80)
OS	MS-Windows XP
Compiler	MS-VC.net2005

5.1 DAI method

Table.1 shows our PC specification. Model data is measured by KONICA MINOLTA VIVID910. Fig.6 shows the model


(a)scene image sample


Side View Top View
(b)scene range data sample

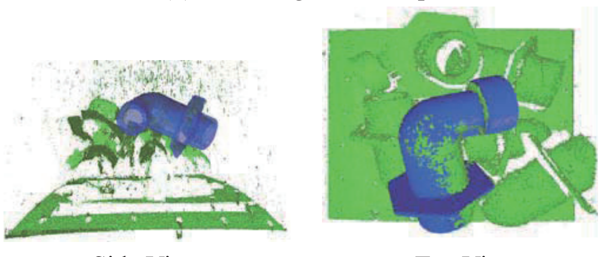

Side View Top View
(c)recongnition result sample

Figure 6: Scene and recongnition result sample

data in the experiments, which were created beforehand of the recognition. The point number of the model are 33558 points. Not only measured one but CAD format can be applied to our recognition method. The scene data sets are measured using our original 3-D range finder called "Micro3D". Resolution of this 3-D range finder is 307200(640 $\times$ 480) points. 17 scene data sets were prepared by using Micro3D. Fig.7 shows the experimental results of DAI method.

CPU processing time is depended on a number of created scene DAI. Usually, hundreds of scene DAI are created and compared with a model DAI in the database. These DAI are compared with the database in sequential. On the other hand, GPU processing time is faster and stabler than CPU because scene DAI are created from all virtual view which was limited by geometric conditions and matched with model DAI database in parallel.

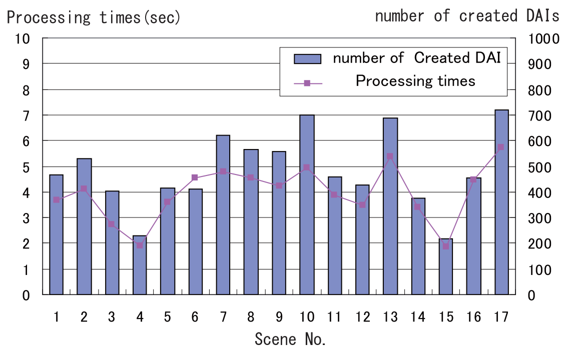


Figure 7: Result of DAI method

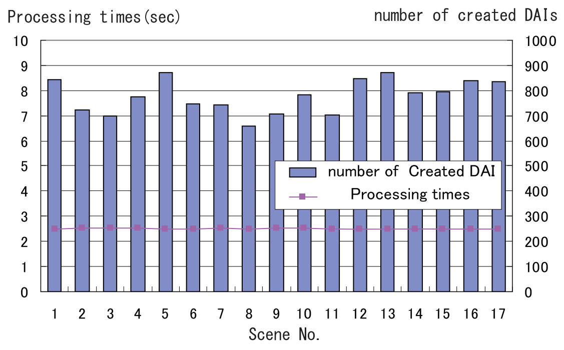


Figure 8: Result of HM-ICP

5.2 M-ICP

Fig.8 shows the the experimental result of HM-ICP. In CPU processing, even 3-D data size for registration are limited to partial regions to reduce computational cost, it takes more than 2 seconds. M-ICP implemented on GPU doesn't limit 3-D data to any region to reduce computational cost, and the processing time is within 0.5 seconds.

Total processing time is shown in table4. In this experiment, our approach is 2-3 times faster than CPU processing although it is executed by basic DAI matching and M-ICP on GPU without any speed-up algorithm.

Let us begin our analysis about calculation amount and processing time. We regard a number of created DAI as calculation amount of DAI method and compare CPU's one and GPU's one. When HM-ICP is executed, the calclation amount is defined as a number of distance calculation. In CPU processing, the calclation amount is reduced by algorithm approach. On the other hand, GPU's one is more enormous than CPU because the calclation amount is not reduced anyway. So, the processing speed on GPU is faster than 40 - 60 times on CPU because processing data size of on GPU is more than 20 times on CPU.

Table 2: Processing time of DAI method(sec)

	worst	best	average	variance
CPU	5.735	1.86	3.955	1.0973
GPU	2.512	2.507	2.509	0.00002

Table 3: Processing time of the HM-ICP(sec)

	worst	best	average	variance
CPU	3.250	2.016	2.858	0.1090
GPU	0.459	0.357	0.408	0.0006

Table 4: Total processing time (sec)

	worst	best	average	variance
CPU	8.188	4.766	6.813	0.9130
GPU	2.971	2.864	2.918	0.0007

6 Conclusion

This paper presents the speed-up technique of the 3-D object recongnition on GPU. Through basic experiments, we show the effectiveness of our approach. Even the processing on GPU is not optimized, the processing speed of proposed method is faster than 40 - 60 times on CPU. In the future, we plan to implement the optimization of parallel computing on GPU and modify the algorithm.

References

[1] Tomoyuki Takeguchi and Shun'ichi Kaneko Robust and Efficient Search of Multiple Objects in Cluttered Scene by Depth Aspect Image, IEEE-Trans. on Industry Electronics, vol.52, no.4, pp.1041-1049, August 2005.

[2] HM-ICP: Fast 3-D Registration Algorithm with Hierarchical and Region Selection Approach of M-ICP

[3] P.J.Besl and N.D.McKay: A Method for Registration of 3-D Shapes: IEEE Trans. on PAMI, vol.14, no.2, pp.239-256, 1992. Haruhisa Okuda, Yasuo Kitaaki, Manabu Hashimoto, and Shun'ichi Kaneko, Journal of Robotics and Mechatronics, Vol.18, No.6, pp. 765-771 2006

[4] NVIDIA CUDA Homepage http://developer.nvidia.com/object/cuda.html

[5] General-purpose computation using graphics hardware. http://www.gpgpu.org/

[6] S.Kaneko, T.Kondo and A.Miyamoto : Robust Matching of 3D Contours Using Iterative Closest Point Algorithm Improved by M-estimation, Pattern Recognition, vol.36,no.9, pp.2041-2047, 2003.