IEIE Transactions on Smart Processing and Computing, vol. 4, no. 1, February 2015
http://dx.doi.org/10.5573/IEIESPC.2015.4.1.010

Accelerating Self-Similarity-Based Image Super-Resolution Using OpenCL

Jae-Hee Jun[1], Ji-Hoon Choi[1], Dae-Yeol Lee[2], Seyoon Jeong[2], Suk-Hee Cho[2], Hui-Yong Kim[2], and Jong-Ok Kim[1]

[1] School of Electrical Engineering, Korea University / Seoul, South Korea {indra, risty, jokim}@korea.ac.kr
[2] Electronics and Telecommunications Research Institute / Dae-jeon, South Korea {daelee711, jsy, shee, hykim5}@etri.re.kr

***** Corresponding Author: Jong-Ok Kim

Received April 5, 2014; Revised June 13, 2014; Accepted November 19, 2014; Published February 28, 2015

** Regular Paper*

***Abstract*:** This paper proposes the parallel implementation of a self-similarity based image SR (super-resolution) algorithm using OpenCL. The SR algorithm requires tremendous computations to search for a similar patch. This becomes a bottleneck for the real-time conversion from a FHD image to UHD. Therefore, it is imperative to accelerate the processing speed of SR algorithms. For parallelization, the SR process is divided into several kernels, and memory optimization is performed. In addition, two GPUs are used for further acceleration. The experimental results shows that a GPGPU implementation can speed up over 140 times compared to a single-core CPU. Furthermore, it was confirmed experimentally that utilizing two GPUs can speed up the execution time proportionally, up to 277 times.

***Keywords*:** Super-resolution, OpenCL, Parallelization, Multiple GPUs

1. Introduction

As the complexity of image processing algorithms increases, their running time on the CPU has been multiplied accordingly. This has led to an increase in the interest in GPGPUs (general-purpose computing on graphics processing units). GPGPU is a technique to utilize a GPU device for general computational applications, which is typically handled by a CPU.

Although a GPU has a much lower clock frequency than a CPU, it has a very large number of cores, and can accelerate complex computations significantly by parallelization. Researchers have recently examined the acceleration of image processing algorithms using GPGPU techniques [8, 9, 11, 12]. The GPGPU technology is categorized into two different techniques. One is CUDA, which was made public in 2007 by Nvidia, and the other is OpenCL, which was released by the Khronos group in 2008 [7]. Some studies have compared CUDA with OpenCL, such as [10]. OpenCL is an open, general-purpose parallel framework that is renowned for its good portability.

This paper proposes the parallel implementation of the self-similarity based image SR (super resolution) algorithm using OpenCL. The SR algorithm, which was proposed previously exploits the inherent self-similarity of an image [1, 2], but it requires tremendous computations to search for a similar patch. In addition, the SR technique is applied primarily to the resolution conversion from FHD (Full HD) to UHD (Ultra HD) which has just begun to be commercialized popularly. The amount of the UHD image data is 4 times that of FHD. Therefore, common software implementation makes it difficult to work in a real-time manner. Accordingly, accelerating the processing speed of SR algorithms is indispensable.

For parallelization, the SR process is divided into several kernels, and memory optimization is performed. In addition, two GPUs are utilized for further acceleration. The experimental results show that GPGPU implementation can speed up the SR process significantly compared to a single-core CPU.

The remainder of this paper is organized as follows. Section 2 introduces the super resolution algorithm that was implemented with OpenCL. Section 3 briefly explains the OpenCL implementation in detail, particularly focusing on memory optimization and support to the multiple GPU devices. Section 4 presents the experimental results. Section 5 concludes the paper.

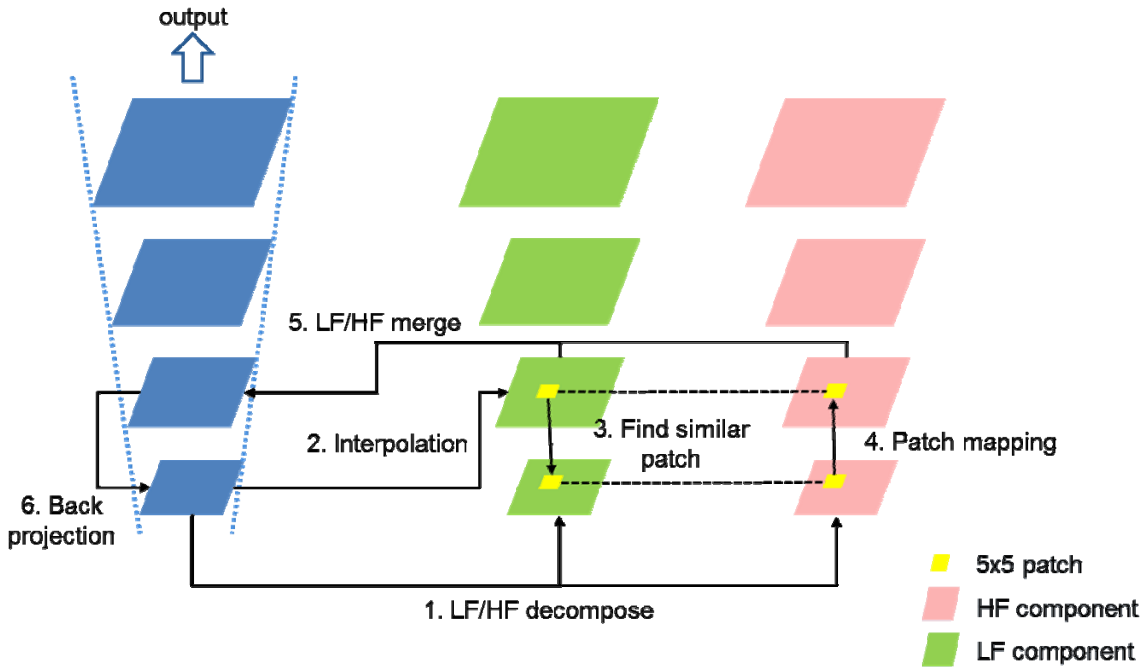


Fig. 1. Tasks of the self-similarity based SR algorithm.

2. Super-Resolution Algorithm

The goal of SR is to reconstruct a HR (high resolution) image from one or more observed LR (low resolution) images. Recently, a number of SR methods have been proposed to exploit the property of self-similarity in an image.

In natural images, small patches tend to recur redundantly across the scale as well as in-scale. This observation is referred to as self-similarity, and has been exploited popularly for image processing, such as de-noising, edge detection and retargeting [3]. In addition, it has been applied to learning-based image SR in recent years [6].

SR methods based on self-similarity can be classified into two categories, depending on the domain to which the self-similarity assumption is applied; spatial domain and LF (low frequency)-HF (high frequency) domain.

Self-similarity with the LF-HF domain is almost equal to the learning based method, such as [4]. Unlike the traditional example-based method, however, self-similarity based SR on the LF-HF domain does not require any prior databases. Therefore, this method can reduce the computational complexity and memory consumption, compared to the conventional learning based approach.

The self-similarity-based SR method consists of three parts overall, as follows: the generation of image pyramids for both LF and HF components, HF reconstruction-based on self-similarity, and back projection after combining the LF and HF components. First of all, an input LR image is decomposed into LF and HF components, which are given by

$$LF_{LR} = H(I_{LR}) \quad (1)$$

$$HF_{LR} = I_{LR} - LF_{LR} \quad (2)$$

where H is a blurring operator, LF_{LR} is the blurred image of an input image. After decomposition, an HR image is initially obtained by simply interpolating the input image. This HR image can be considered the LF component of the target HR image, which is denoted by LF_{HR}. Scaling is increased gradually to a target scale by a factor of 1.25. An incremental coarse-grained method using a small scale factor was reported to produce a more elaborate result [5].

For the reconstruction of HF_{HR}, for every patch p_i in the image LF_{HR}, the most similar patch p_j is searched on the LF_{LR} domain. The corresponding HF_{LR} patch is then copied to query the patch. By merging the LF_{HR} and HF_{HR} image, an HR image, I_{HR}, can be reproduced as follows.

$$HF_{HR}(p_i) = HF_{LR}(p_j) \quad (3)$$

$$I_{HR} = LF_{HR} + HF_{HR} \quad (4)$$

In this way, hierarchical HF reconstructions in the image pyramid are repeated until the target HR is achieved. Fig. 1 presents the overall processes of the SR algorithm.

3. Implementation Using OpenCL

This section present a parallel implementation of the SR algorithm. First, the SR process is divided into a couple of kernels, and the problem of the race condition, which actually occurs in searching similar patches, is then addressed. Next, memory optimization is performed based on the memory model. Finally, we deal with how to handle multiple GPU devices for further acceleration.

A kernel is a unit of code that represents a single

Table 1. Kernel design and computational time.

Procedure	Kernel name	Time(μs)
LF/HF subtract	Conv & Sub	6,237
Interpolation	imgScaler	3,611
Find similar patch	SSSR & Setzero	353,347
Patch mapping	Add & Div	2,897
LF/HF merge	Add	1,179
Back Projection	Conv & Add & Sub	4,834

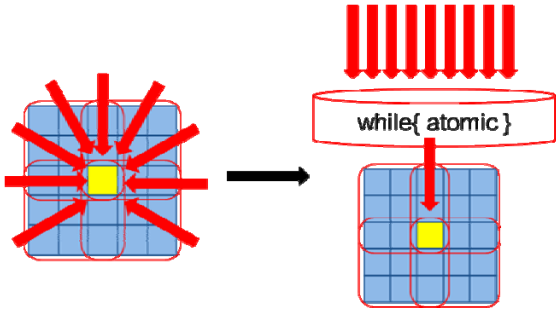


Fig. 2. Race condition and its solution.

executing instance in a GPU. This single kernel instance corresponds to a work item in OpenCL. When the work-item and work-group are designed, both GPU specification and image resolution should be considered simultaneously. 32x32 work-items are designated as a work-group. The kernels for the SR algorithm are designed, as listed in Table 1 according to tasks and synchronization points. For further optimization of the OpenCL implementation, the individual kernel running time is measured using performance profiler, as listed in Table 1.

The kernel used to find the best matching patch is the most time-consuming execution in the SR process. Hence, special optimization of this kernel is essential for better speed acceleration. This will be discussed in Section 3.2.

3.1 Race Condition

When the 'Find similar patch' kernel is running, two or more work-items may reference the address value of a single pixel at the same time. At this time, the race condition occurs, which indicates the situation where multiple work-items simultaneously accesses a certain pixel to be declared in the global memory. To overcome this conflict, a serialization method is adopted by the atom command provided by OpenCL. Note that the use of the atomic command ensures the correctness of local memory by serialization.

Fig. 2 illustrates the occurrence of the race condition and its solution.

3.2 Memory Optimization

Next, the memory optimization step is described. OpenCL defines a hierarchical memory model, as shown in Fig. 3. Local memory is manufactured by on chip, and it is much faster than global memory. In addition, OpenCL provides image memory that is specialized for accessing image data. Image memory requires fewer GPU cycles when data is read, so its access speed is much faster than global memory. In this paper, speed acceleration is optimized based on two memory model, which are image buffer and local memory.

At the 'Find similar patch' kernel, similar patches are searched in the surrounding local area of a target patch, and all pixel data in the search area are loaded into the local memory at one time. This can accelerate the memory access speed better than global memory.

The local memory is divided into two memory banks. To achieve high memory bandwidth (or high speed), all work-items should be split evenly into each bank to reduce the number of work-items per bank as much as possible. If many work-items are assigned to a certain bank unevenly, it takes more time due to serialization. This phenomenon is the so called bank conflict, and it should be avoided when local memory is used.

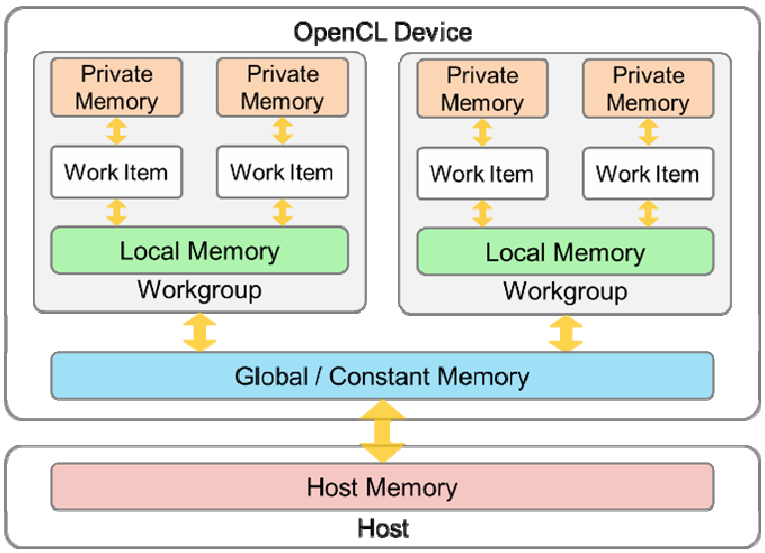


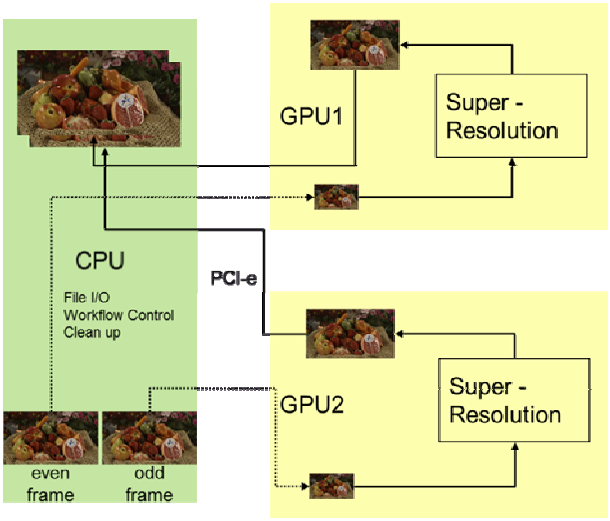
Fig. 3. OpenCL memory model.

3.3 Support to Multiple GPUs

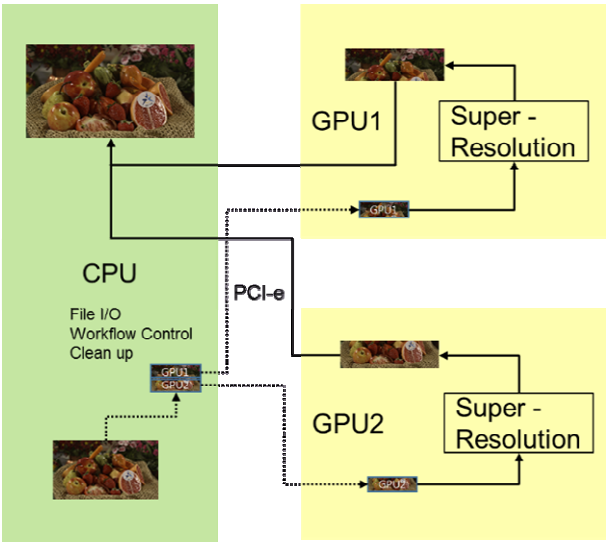
To utilize multiple GPU devices, a command queue should be prepared for each GPU individually. On the other hand, context can be shared among multiple command queues or each queue can have its own context. The former is a single context-based implementation. This is a typical way to work with multiple devices. When multiple devices use the same context, most of the OpenCL objects are shared (e.g., kernel, program, and memory objects). As mentioned above, one command queue should be generated per GPU; hence, multiple command queues are needed within the same context. This approach is suitable for the algorithm that uses less memory.

Another implementation approach is to define the different contexts per device separately. This is commonly used in computing on a heterogeneous device system (e.g., CPU and GPU). Data is handled independently among multiple contexts, and system crashes can be avoided.

Fig. 4 illustrates the actual implementation of the above-mentioned two approaches in the case of two GPUs. Suppose that video frames are super-resolved on two GPUs. As shown in Fig. 4(a), the image data is split into two devices at a frame level. In other words, the odd frames are processed into one GPU, and the even frames are processed into the other one. When a single context is adopted, a video frame is split equally into two segments, and each segment is processed independently of each GPU, as shown in Fig. 4(b).



(a) Multiple context



(b) Single context

Fig. 4. Two ways to utilize multiple devices in OpenCL.

4. Experimental Results

The experiments are performed on an Nvidia GeForce GTX-780 device. The host machine is an Intel® Core™ i5-4590 CPU @ 3.30Ghz, 64 bits compatible. The OpenCL programs were developed using Visual Studio 2012 and OpenCL version 1.1. The test application was to super-resolve from a 1920 x 1080 FHD image to a 3840x2160 UHD using the self-similarity based SR method in Section 2. The performance was measured by the total execution time of all kernel functions in a GPU. The execution time was measured by *Nsight*, a profiling tool provided by Nvidia.

The parallel implementation in GPU speeds up the running time, as confirmed in Table 2. In particular, if the local memory based optimization is performed, the algorithm running speed can be accelerated further by approximately 2 times compared to the non-memory-optimization. When each GPU uses its own context separately, the execution speed can be made approximately 277 times faster than that of a single core CPU.

In a single context approach, an image is partitioned equally into two segments, and half of the image is processed into each GPU. When there are super-resolving pixels near the border line between two image segments, the pixel data of the other image segment is needed to find a similar patch. Therefore, for a single context approach, more than half an image is actually transferred to individual GPUs. This leads to more transfer time between the CPU and GPU.

Table 2. Running time comparisons.

# of device	Implementation method	Time(s)
1	CPU (1 core)	25.209
	GPU	0.388
	GPU (image buffer)	0.273
	GPU (local memory)	0.169
2	2 GPU (2 context)	0.091
	2 GPU (1 context)	0.098

5. Conclusion

This paper proposed a parallel implementation of a complex image SR algorithm using OpenCL for speed acceleration. Global memory synchronization was solved

using the race condition. In the SR algorithm, finding a similar patch was very time-consuming; its speed could be accelerated using the local memory. In addition, two GPUs were utilized for further acceleration. The experimental results showed that GPGPU implementation can speed up the process by more than 140 times compared to that using a single-core CPU. Furthermore, it was confirmed experimentally that the use of two GPUs can accelerate the execution time proportionally, by up to 277 times.

Acknowledgement

This work was supported by the ICT R&D program of MSIP/IITP. [13-912-02-002, Development of Cloud Computing Based Realistic Media Production technology]

References

[1] S.J. Park, O.Y. Lee, and J.O. Kim, "Self-similarity based image super-resolution on frequency domain," *Proc. of APSIPA ASC 2013*, pp. 1-4, Nov. 2013. Article (CrossRef Link)

[2] J.H. Choi, S.J. Park, D.Y. Lee, S.C. Lim, J.S. Choi, and J.O. Kim, "Adaptive self-similarity based image super-resolution using non local means," Proc. Of *APSIPA ASC 2014*, Dec. 2014. Article (CrossRef Link)

[3] A. Buades, B. Coll and J.M Morel, "A non-local algorithm for image denoising", *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. 60 – 65, June. 2005. Article (CrossRef Link)

[4] W.T. Freedman, T.R Jones, and E.C. Pasztor "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, pp. 56-65, Mar. 2002. Article (CrossRef Link)

[5] D. Glasner, S. Bagon, and M.Irani, "Super-resolution from a single image," *12th IEEE Int. Conf. on Computer Vision*, pp. 349 – 356, Sep. 2009. Article (CrossRef Link)

[6] G. Freeman, and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. on Graphics*, vol. 30, Article No. 12, Apr. 2011. Article (CrossRef Link)

[7] Khronos OpenCL Working Group. The OpenCL Specification Version 1.2. Khronos Group, 2012. http://www.khronos.org/opencl

[8] J. Zhang, J. F. Nezan, and J.-G. Cousin, "Implementation of motion estimation based on heterogeneous parallel computing system with OpenCL," *in Proc. IEEE Int. Conf. High Performance Computing and Commun.*, 2012, pp. 41–45. Article (CrossRef Link)

[9] N.-M. Cheung, X. Fan, O. C. Au, and M.-C. Kung, "Video coding on multicore graphics processors," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 79–89, 2010. Article (CrossRef Link)

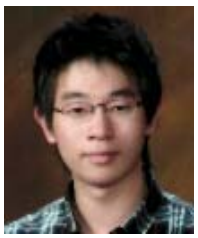
[10] J. Fang, A. L. Varbanescu, and H. Sips, "A Comprehensive Performance Comparison of CUDA and OpenCL," *in Proceedings of the International Conference on Parallel Processing, ICPP'11*, Sep. 2011. Article (CrossRef Link)

[11] G. Wang, Y. Xiong, J. Yun, and J. R. Cavallaro, "Accelerating computer vision algorithms using OpenCL framework on the mobile GPU – a case study", *in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May. 2013. Article (CrossRef Link)

[12] J.H. Jun, J.H. Choi, D.Y. Lee, S.Y. Jeong, J.S. Choi, J.O. Kim, "Implementation Acceleration of Self-Similarity Based Image Super Resolution Using OpenCL" , *in International Workshop on Advanced Image Technology (IWAIT) & International Forum on Medical Imaging in Asia (IFMIA)*, Jan. 2015.

Jae-Hee Jun received his B.S. degree in electronic engineering from Korea University, Seoul, Korea, in 2013, He is currently pursuing the M.S. degree in electronic engineering at Korea University, Seoul, Korea. His research interests are image signal processing and multimedia communications.

Ji-Hoon Choi received his B. S. degree in electronic engineering from Korea University, Seoul, Korea, in 2011, He is currently pursuing a Ph.D. degree in electronic engineering at Korea University, Seoul, Korea. His research interests are image signal processing and multimedia communications.



Dae-Yeol Lee received his BS and MEng degrees in electrical and computer engineering from Cornell University, United States, in 2012 and 2013 respectively. In August 2013, he joined Broadcasting and Telecommunications Media Research Department, ETRI, Korea, where he is currently a researcher. His research interests include image/video processing, video coding and computer vision.

Seyoon Jeong received his BS and MS degrees in electronics engineering from Inha University, Korea, in 1995 and 1997 respectively, and received hsi PhD degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2014. He joined ETRI, Daejeon, Rep. of Korea in 1996 as a senior member of the research staff and is now a principal researcher. His current research interests include video coding, video transmission and UHDTV.

Sukhee Cho received her BS and MS in computer science from Pukyong National University, Busan, Rep. of Korea, in 1993 and 1995, and her PhD in electrical and computer engineering from Yokohama National University, Yokohama, Kanagawa, Japan, in 1999. Since 1999, she has been with the Broadcasting and Telecommunications Media Research Department at ETRI, Daejeon, Rep. of Korea. Her current research interests include realistic media processing technologies, such as video coding and the systems of 3DTV and UHDTV.

Hui Yong Kim received his BS, MS, and Ph.D degrees from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1994, 1998, and 2004, respectively. From 2003 to 2005, he was the leader of Multimedia Research Team of AddPac Technology Co. Ltd. In November 2005, he joined the Broadcasting and Telecommunications Media Research Laboratory of Electronics and Telecommunications Research Institute (ETRI), Korea, and currently serves as the director of Visual Media Research Section. From 2006 to 2010, he was also an affiliate professor in University of Science and Technology (UST), Korea. From September 2013 to August 2014, he was a visiting scholar at Media Communications Lab of University of Southern California (USC), USA. He has made many contributions to the development of international standards, such as MPEG Multimedia Application Format (MAF) and JCT-VC High Efficiency Video Coding (HEVC), as an active technology contributor, editor, and Ad-hoc Group co-chair in many areas. His current research interest include image and video signal processing and compression for realistic media applications, such as UHDTV and 3DTV.

Jong-Ok Kim (S'05-M'06) received his B.S. and M.S. degrees in electronic engineering from Korea University, Seoul, Korea, in 1994 and 2000, respectively, and Ph.D. degree in information networking from Osaka University, Osaka, Japan, in 2006. From 1995 to 1998, he served as an officer in the Korean Air Force. From 2000 to 2003, he was with SK Telecom R&D Center and Mcubeworks Inc. in Korea, where he was involved in research and development on mobile multimedia systems. From 2006 to 2009, he was a researcher in ATR (Advanced Telecommunication Research Institute International), Kyoto, Japan. He joined Korea University, Seoul, Korea in 2009, and is currently an associate professor. His current research interests are in the areas of image processing and visual communications. Dr. Kim was a recipient of a Japanese Government Scholarship during 2003-2006.