N A Hadi
Ibnun Sina Institute for Scientific and Industrial Research, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia
normi@fskm.uitm.edu.my

Abstract. This paper discusses on CPU-GPU collaboration in surface reconstruction process from CT scan image, in this case human head. This type of image consists hundreds of data points and needs a powerful processor to ensure the accurateness of result. Therefore, an architecture to segment the whole process to CPU and GPU is designed. Comparison of CPU and GPU is done based on execution time and speedup for total number of the data points up to 10^5. From the analysis, it is shown that the maximum speedup is more than 15 for a process.

1. Introduction
Surface reconstruction from multiple images is a significant method to be explored since it is cost effective in terms of space, price and time. The produced method can be implemented in any software, and on personal device independently which allows further analysis to be done off-machine [1]. Furthermore, reconstruction of human part such as head can be applied in many areas for example recognition, diagnosis, and animation [2].

However, a challenge in using personal devices in processing the multi-slice image is the memory limitation. A set of CT scan image contains about 180 slices, and each extracted outline consists of as many as 1500 data points. In some cases, some parts are cut out to reduce the number of data points since a CPU cannot afford the amount of data. Consequently, some important information of the reconstructed object are eliminated and lead to misinterpretation of the data [2].

The drawback of data points reduction has motivate some works on parallelization on image. [3] has segmented the surface points using dyadic segmentation. Each segment is run on several CPU simultaneously. Although the result shows the efficiency of parallel architecture, but the number of surface points been considered is small (64 points x 50 slices). Furthermore, the parallel process is only for surface fitting process. The same data are then tested on MATLAB pmode architecture [4]. In this work, only one CPU has been used but different workers. Thus, it cannot overcome the memory limitation problem.

Recently, the study on parallel programming using GPU is a very important area. In mathematical aspect, parallelization can be used to divide the mathematic equation into several parts and run it simultaneously. In differential equation field for example, parallelization has been applied in various
field such as in predicting the ice thickness [5], analysing the eye movement based on EEG signal [6],
and modelling a brain tumour growth [7]. In chemistry domain, parallelization has been applied in
studying the chemical properties of benzene [8]. However, in 3D image visualization focusing on
parametric surface fitting, lack of literatures has been found.

Thus, this project will employ CPU-GPU architecture so that more data points can be processed, and
more accurate result can be produced. The architecture is called as CPU and GPU collaboration or
cooperation [9] where CPU and GPU are combined intelligently to gain high computational paradigm
[10]. The software used is MATLAB with GPU Tesla K20c with compute capability 3.5.

This paper is structured such that Section 2 will discuss on pre-processing algorithm for CT scan
image. In the next section, the implementation of GPU on the algorithm is elaborated. This paper ends
with conclusion in Section 4.

2. Image pre-processing from CT scan images

Image processing consists of several main process as in figure 1 from raw image until surface points
extraction.

![Figure 1. Surface reconstruction flowchart.](image)

Step 1 is importing required image to the software. For CT scan image, it is greyscale form with
different intensity for each point. The imported image is read as a matrix for example in this case is 512
× 512 matrix. Following steps are done based in this matrix. To ease the process, the intensity is
converted to black and white only as in step 2, so that the region of interest (ROI) can be extracted as
black pixels in step 3. Since the surface will be reconstructed for face, only outline of ROI will be
considered. Therefore, outline extraction process is done in step 4, and arrangement of the extracted
points is done in step 5. Step 6 to 8 are towards the surface reconstruction. Since the number of points
in the outline of each slice are different, reparameterization process need to be carried out to produce
the same number of surface points. In step 6, spline curve, \( F(t) \) is fitted to points form step 5 as follows,
Curve equation in (1) is the standard spline equation for degree $n$ with control points $V_i$, and basis function $B_i(t)$.

After curve fitting process, the curves points $p_i = F(t_i)$ are extracted in step 6. Finally, surface points are extracted based on the extracted curves points. The number of surface points must be large enough to ensure the accuracy of the produces surface.

In step 8, there are several sub-processes as follows,

Step 8.1: Calculate the distance, $D_i$ between curve points, $p_i$ and $p_{i+1}$ where $D_i = ||p_i - p_{i+1}||$

Step 8.2: Calculate the cumulative distance, $\sum D_i$ for each curve points.

Step 8.3: Calculate the surface points by fitting the spline surface as,

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} V_{ij} B_i(u) B_j(v)$$ (2)

The produce images for steps in figure 1 are shown in figure 2.

3. Implementation of parallel algorithm for image pre-processing

By referring to figure 1, steps that involve computation are steps 6 and 8. Thus, these steps will be run on GPU to decrease the execution time. The architecture for parallelization is based on Single Instruction Multiple Thread (SIMT) as shown in figure 3 [11].
Figure 3. The CPU-GPU architecture for surface reconstruction.

In this architecture, some of the process can be written in the kernel and executed using GPU (device). Each kernel will employ several numbers of grids, blocks and threads. For this project, step 6 and steps 8.1 until 8.3 are executed in four (4) different kernels. Other steps are executed on CPU (host).

Since steps 1 to 5 are executed on CPU, the data must be passed to GPU for step 6. Then, GPU must return the processed data to CPU again, before it is processed in the CPU for step 7. Similar communication process between CPU-GPU is done for steps 8.1 to 8.3. In each kernel, the distribution of data depends on the number of blocks and threads [12]. The specification of CPU and GPU in this project is shown in table 1.
Table 1. Information of the employed CPU and GPU.

| Hardware | Specification                            |
|----------|------------------------------------------|
| CPU      | Intel (R) XEON (R) (2.10GHz) 2 processors |
| GPU      | NVIDIA Tesla K20c, MaxThreadBlockSize = [1024 1024 64] |
| OS       | Windows 10 64-bit                        |

The analysis of CPU and GPU performance is shown in the following section.

4. CPU-GPU performance

CPU versus GPU comparison is done based on execution time and speed up. In GPU analysis, the number of processor (threads) is increasing with the number of data.

4.1. Execution time

Execution time is the total time taken for execution. For GPU, the time taken includes the data passing from CPU to GPU, and back to GPU again. The comparison of execution time for CPU and GPU are shown in figure 5.

![Figure 4. Comparison of execution time for CPU-GPU.](image-url)

(a) Step 6.  
(b) Step 8.1  
(c) Step 8.2  
(d) Step 8.3  
(e) Curve points image rendering.  
(f) Surface points image rendering.
This execution time can be analyzed in terms of speedup as discussed in the section 4.2

4.2. Speedup

Speedup is calculated as the execution time for a processor (CPU) over multi-processor (GPU).

![Graphs showing speedup for each kernel](image)

**Figure 5.** Speedup for each kernel.

The speedup is increasing with number of data. It shows the load balancing of the data for each thread.

4.3. Discussion

Based on figure 5(a), the execution time for CPU is always lower than GPU for all considered number of data. This is because, these numbers are not big enough for GPU, and the amount of communication time is more than computation. Its speedup value in figure 6(a) has values less than 0.02. Therefore step 6 is not suitable to be executed in GPU for this amount of data.

For figure 5(b) to (d), the execution time for CPU is lower than GPU for the first few number of data points. However, GPU starts to give better execution time for $10^5$, $10^3$ and $10^2$ for figure 5(b),...
(c) and (d) respectively. This can be observed by its linear shaped speedup in figure 6(a) to (c). Additionally for step 8.3, CPU cannot afford the data for $10^5$ and above due to limitation of memory. Conversely for GPU, this step takes only 0.1671s to be executed.

For image rendering, both curve and surface shows that GPU has helped in reducing the execution time for all amount of data. However, the speedup for each amount of data is about the same with the difference less than 0.6. This is maybe because of no calculation involved in this process.

Overall analysis, execution time for CPU has been drastically increase after certain amount of data and lead to out of memory, but not for GPU. The execution time for GPU is increased less than 10% by the increment of data amount. This shows that GPU can handle more data within acceptable execution time.

5. Conclusion
In this paper, image processing steps for CT scan image of human head has been discussed. This type of image consists hundreds of data points per slice. These data points cannot be ignored since each of it carry significant interpretation of the image and object. However, limitation of CPU memory is the barrier to achieve accurate result. Therefore, CPU and GPU are collaborated in this project. The results show promising performance of the developed architecture in terms of execution time and speedup for all kernel except kernel 1 which is for curve points calculation. For better analysis, further comparison should be done such as computational complexity, granularity and standard deviation. Additionally, the distribution of data to thread needs further analysis to avoid time wastage for idle threads.

Acknowledgments
The author would like to express sincere gratitude to the Telkom University for the invitation for this conference. To Ministry of Education, Malaysia, Universiti Teknologi MARA and Universiti Teknologi Malaysia for funding this study.

References
[1] Hadi N A, Ibrahim A and Yahya F 2013 G 2 Beta-spline : An Ideal Surface Representation of Multi-Slice Medical Images IEEE 9th International Colloquium on Signal Processing and its Applications (Kuala Lumpur: IEEE) pp 8–10
[2] Hadi N A, Ibrahim A, Yahya F and Ali J M 2012 3-Dimensional beta-spline wireframe of human face contours 2012 IEEE 8th International Colloquium on Signal Processing and Its Applications, CSPA 2012 vol 0 (IEEE) pp 110–4
[3] Abdul Hadi N, Abd Halim M S, Ibrahim A, Sulaiman H, Yahya F and Md Ali J 2014 Dyadic Segmentation for Parallel Beta-Spline Surface Reconstruction from Multi-slice Images Fifth International Conference on Intelligent Systems, Modelling and Simulation pp 524–8
[4] Syawal M, Halim A, Hadi N A and Sulaiman H 2017 An Algorithm for Beta-Spline Surface Reconstruction from Multi Slice CT Scan Images using MATLAB pmode IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE) (IEEE) pp 1–6
[5] Saidi M M and Alias N 2016 High Speed Computing of Ice Thickness Equation for Ice Sheet Model J. Teknol. 78 143–9
[6] Alias N, Mohsin H M, Nadirah M, Mustaffa S H M S and Reyaz R 2016 PARALLEL ARTIFICIAL NEURAL NETWORK APPROACHES FOR DETECTING THE BEHAVIOUR OF EYE MOVEMENT USING CUDA SOFTWARE ON HETEROGENEOUS CPU-GPU SYSTEMS J. Teknol. 78 77–85
[7] Alias N, Said N M, Nur S, Khalid H, Sin D, Ching T and Ing P T 2008 High Performance Visualization of Human Tumor Growth Software VECPAR pp 591–9
[8] Alias N, Sahnoun R and Malyshtkin V 2017 HIGH-PERFORMANCE COMPUTING AND COMMUNICATION MODELS FOR SOLVING THE COMPLEX INTERDISCIPLINARY
PROBLEMS ON DPCS \textit{ARPN J. Eng. Appl. Sci.} 12

[9] Raju K and Chiplunkar N N 2018 A survey on techniques for cooperative CPU-GPU computing \textit{Sustain. Comput. Informatics Syst.} 19 72–85

[10] Mittal S and Vetter J S 2015 A Survey of CPU-GPU Heterogeneous Computing Techniques \textit{ACM Comput. Surv.} 47 1–35

[11] Alias N 2017 Introduction to Cuda, CPU-GPU Computing platform, Comparison and performance

[12] Alias N, Nofri N, Suhari Y, Farhah H, Saipol S, Dahawi A A, Saidi M M, Hamlan A, Rahim C and Teh C 2016 PARALLEL COMPUTING OF NUMERICAL SCHEMES AND BIG DATA ANALYTIC FOR SOLVING REAL LIFE APPLICATIONS \textit{J. Teknol.} 78 151–62

[13] Sahimi M S, Alias N and Sundararajan E \textit{The AGEB Algorithm for Solving the Heat Equation in Three Space Dimensions and Its Parallelization Using PVM}