(IJNCAA)

ISSN 2220-9085(ONLINE)
ISSN 2412-3587(PRINT)

INTERNATIONAL JOURNAL OF

New Computer Architectures and their Applications

Volume 8, Issue 1
2018

SDIWC

www.sdiwc.net

International Journal of
NEW COMPUTER ARCHITECTURES AND THEIR APPLICATIONS

Overview

The SDIWC International Journal of New Computer Architectures and Their Applications (IJNCAA) is a refereed online journal designed to address the following topics: new computer architectures, digital resources, and mobile devices, including cell phones. In our opinion, cell phones in their current state are really computers, and the gap between these devices and the capabilities of the computers will soon disappear. Original unpublished manuscripts are solicited in the areas such as computer architectures, parallel and distributed systems, microprocessors and microsystems, storage management, communications management, reliability, and VLSI.

One of the most important aims of this journal is to increase the usage and impact of knowledge as well as increasing the visibility and ease of use of scientific materials, IJNCAA does NOT CHARGE authors for any publication fee for online publishing of their materials in the journal and does NOT CHARGE readers or their institutions for accessing the published materials.

Publisher

The Society of Digital Information and Wireless Communications
20/F, Tower 5, China Hong Kong City, 33 Canton Road, Tsim Sha Tsui, Kowloon, Hong Kong

Further Information

Website: http://sdiwc.net/ijncaa, Email: ijncaa@sdiwc.net,
Tel.: (202)-657-4603 - Inside USA; 001(202)-657-4603 - Outside USA.

The issue date is March 2018.

IJNCAA

ISSN 2220-9085 (Online)
ISSN 2412-3587 (Print)

Volume 8, Issue No. 1 *2018*

CONTENTS

ORIGINAL ARTICLES

International Journal of New Computer Architectures and their Applications (IJNCAA) 8(1): 1-10
The Society of Digital Information and Wireless Communications, 2018 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Acceleration of Canny Edge Detection Algorithm Using Parallel Clusters

Njood S.Alassmi and Soha S. Zaghloul, PhD.
College of Computer & Information Sciences, Department of Computer Science, King Saud University, Riyadh
435920199@student.ksu.edu.sa, smekki@ksu.edu.sa .

ABSTRACT

Image processing is a computational operation that requires many CPU cycles for simple image transformation. It takes every pixel of an image to perform a transformation to a new image. The image can be divided into smaller chunks, with same transformation operations being implemented on each. Thus, image processing is a good candidate for running on a parallel processor to improve the speed of computation when there are multiple images to be processed. In fact, this research focuses on Canny edge detection as a case study of probing parallelism. This work presents the design and implementation of sequential and parallel edge detection algorithms that are capable of producing high-quality results and performing at high speed. Therefore, this research aims to improve the Canny edge detection algorithm in terms of speed and scalability with different sizes of images. The algorithm is implemented using parallel clusters on KACST's SANAM supercomputer. It is found that there is a valuable gained speedup with respect to the sequential version.In addition, it is found that more parallelism is explored in larger image sizes with Canny edge detector.

KEYWORDS

Canny Edge Detection, Sequential Implementation, Parallel Processing, Parallel Cluster, SANAM.

1 INTRODUCTION

The importance of image detection algorithms is increasing with the advance of technology due to the recent expansion of images in industry. The tools that acquire images became handy to an ample number of users. These are interested in assessing the quality of the acquired images. Image edge detection is one of the most essential methods in image processing for image quality assessment, image analysis, image pattern recognition, and computer vision. Actually, the importance of image detection comes from the fact that it is the first step to be conducted on images.

In the literature, many filters are used to conduct the image edge detection. These include Prewitt, Canny, Roberts, and Sobel. In [1], the researchers proved that although the Sobel operator is the simplest amongst the previously mentioned ones; however, it is sensitive to noise. On the other hand, the Canny operator applies Gaussian filter to remove any noise in the image. Therefore, better results are guaranteed. In addition, [2] found that almost 90% of the edges are correctly detected using Canny operator.

However, the Canny edge detection algorithm suffers from complex computations and therefore, is time consuming. Given an image of size mxn, the time complexity of the Canny algorithm is found to be O(mn log mn). In order to make the algorithm produce its results in real time, parallelism is needed.

There are two main parallel programming models; namely, the shared memory processor (SMP) and the parallel clusters - also known as message-passing processors (MPP). This classification is based on the way the memory is shared between the multiprocessors [3]. In SMP, the processors communicate with each other through a common memory. On the other hand, MPP is a distributed memory parallel architecture. Communication between processors is conducted through message passing. A hybrid architecture of both SMP and MPP is also available.

The aim of this research is tomaximize the speedup, the efficiency, and the scalability of the parallel program. Also, the accuracy of the detected edges should be at least the same as the sequential version. The layout of this research is as follows: Section 2 exposes similar work in the literature. Section 3 and Section 4 detail the sequential and parallel versions of Canny algorithm respectively. Section 5 explains the performance metrics to be measured in the experiments. Results are shown and analyzed in Section 6. Section 7 concludes the research and gives a hint about the future work.

2 REVIEW OF RELATED STUDIES

This research focuses on Canny edge detection as a case study to implement the parallelism concept. In order to implement the Canny edge detection algorithm, a series of steps must be followed. The first is to filter out any noise in the original image before attempting to detect any edges. This can be done by applying the Gaussian filter that can be computed using a simple mask. After smoothing the image and removing the noise, the next step is to find the edge strength by measuring the gradient of the image. Subsequently, the direction of the edge should be computed using the gradient in the x and y directions. After the edge directions are known, non-maximum suppression should be applied, which is used to trace along the edge in the edge direction and suppress any pixel value. This results in a thin line in the output image. The final stage is edge tracking by hysteresis; the final edges are determined by suppressing all edges that are not associated with the definite edge [4]. Y. Kong et al. [5] implemented the Canny edge detection method and found that almost 90% of the edges are correctly detected. However, disadvantages include intensive and complex computations, ensuring that it is time consuming. The overall time complexity is found to be O (m*n log m*n), where m and n are the width and length of the image in terms of pixels.

With the prevalence of parallelism, image edge detection algorithms are implemented in various parallel environments, including GPUs, SMPs, and MPPs. In [6] and [7], the authors proved the efficacy of the GPU in conducting millions of pixel calculations involved in image processing in parallel. In addition, [8] examined the edge detection in both sequential and parallel algorithms in order to measure the speedup. The results indicate that the parallel implementation is about 262 times faster than the sequential implementation.

In [9], the authors proposed a parallel Canny algorithm for the embedded CPU and GPU heterogeneous. The results showed that the proposed parallel Canny algorithm achieved nearly 50 times speedup on the embedded systems. They used differently sized images for the tests. For 512×512 images, the runtime of a Canny algorithm in sequential implementation is found to be over 1,5898 milliseconds. Otherwise, Canny improved the runtime to a level of 3,310 milliseconds by using embedded CPUs and GPUs.

In [10], a parallel version of the Canny algorithm is applied on a SMP environment. The maximum gained speedup is around 1.7 when using three threads for an image size of 2000x2000 pixels.

In [11]-[13], the edge detection algorithm is applied on parallel clusters. In [11], the authors used 4 parallel clusters, and achieved a gained speedup mostly above 0.7 for a 2Kx2K image. In [12], the authors applied K-means clustering. It is found that the algorithm is more accurate in edge recognition; however, it takes more time (3.32 seconds as opposed to 1.28 seconds for Canny). Finally, in [13], the authors used 10 parallel clusters and the execution time is found to be 0.0311 seconds for an image size of 2.5K x 2.5 K.

3 SEQUENTIAL ALGORITHM

The Canny edge detection algorithm is proved to provide the optimal edge detector [14].The algorithm goes through four steps before reaching the final result. The input image is read into an array of size m×n where m and n are the width and length of the image respectively in terms of number of pixels. The algorithm starts by storing the 2-D

image in a 1-D array to simplify processing in later phases. Then, the algorithm visits every pixel of the image and makes the modifications as detailed below. The algorithm goes through a series of steps [15]:

1. **Noise Removal:** The first step is to filter any noise in the original image before attempting to detect edges. This is done using Gaussian filter computed by a simple mask and exclusively used in the Canny algorithm.
2. **Gradients Calculation:** After eliminating the noise and smoothing the image, gradients of the image are taken for each edge, gradient magnitude (M) represents the edge intensity.
3. **Thresholding:**
   Two thresholds, upper and lower, are used to detect image edges. This allows for greater flexibility than using a single threshold approach.
4. **Edges Classification:**
   The thresholding is used to detect the final edges in an image by using the two thresholding values of v_low and v_high. In this step, the pixels are classified into three categories in terms of the value of the magnitude M. If a pixel gradient is higher than the high threshold (M >v_high), the pixel is accepted as a strong edge. If (v_low ≤ M <v_high), the pixel is classified as a weak edge and waits for thejudgement of the edge points. If (M <v_low), the pixel is classified as a non-edge, and therefore, rejected.The diagram in Fig.1 shows the main steps of the approach for Canny edge detection.

The diagram in Fig.1 depicts the previously detailed steps of the approach for Canny edge detection. A major disadvantage of Canny is its intensive and complex computations making it time consuming. In order to measure the overall program complexity, the complexity of each step is first calculated independently. Steps 1 to 3 are implemented by image convolution. For an *mxn* image size, the time complexity is therefore *O (mn log mn)* for the first three steps. The final step of thresholding is implemented by the final step of thresholding is implemented by selecting all the high values and removing the low values. This can be done in time *O (mn)* [15]. Therefore, by taking the highest complexity, it is found that the overall time complexity is *O (mn log mn)*.

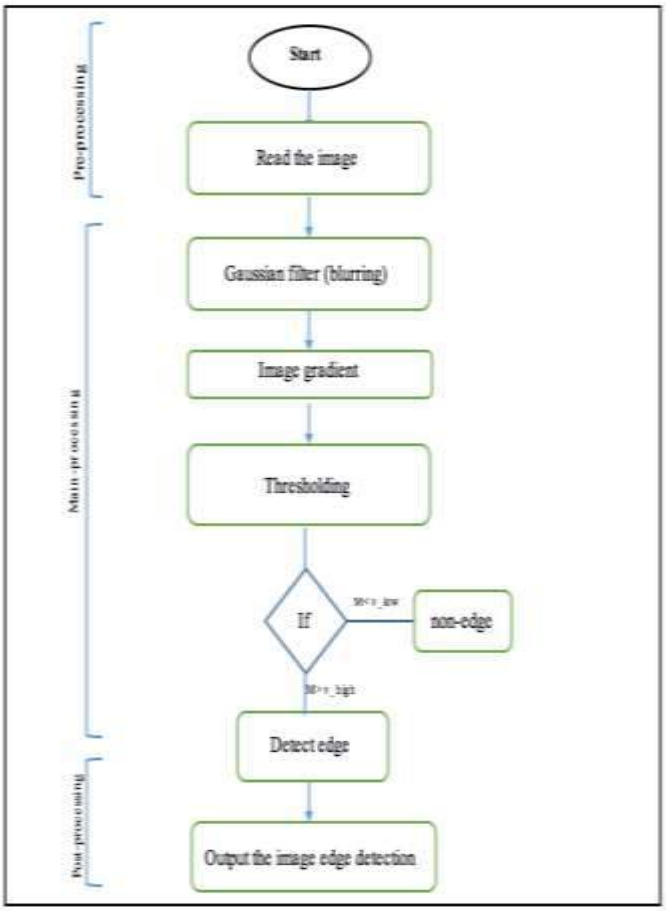


Figure 1. The flow chart of the sequential version for Canny Edge Detection steps.

4PARALLEL ALGORITHM ON CLUSTERS

Clustering is a technique to configure multiple machines belonging to a general network for parallel purposes. A cluster consists therefore of a set of nodes interconnected by high technology network.

The algorithm for parallel clusters is divided into three main parts: the partitioning of the image, the implementation of the Canny filter, and finally, the merging of sub-images.

- Image Partitioning

The first step to make use of data parallelism on clusters, is to partition the image. This is explained as follows: the image is divided into chunks; each chunk is to be processed independently. In this experiment, the partitioning is conducted more than once to conclude the relationship between the execution and the number of partitions. The aim is to find the number of partitions that yield the highest speed-up.

Fig.2 explains the image partitioning by deciding the chunk size in terms of number of rows and columns. The image is partitioned to equal-sized chunks. It is assumed that the size of the image is divisible by the number of chunks, and therefore, there are no left-overs. For example, if the numbers of rows and columns are set to three, the algorithm divided the original image into nine chunks. Then, each sub-image is saved and assigned to a node.The partitioning process is conducted in parallel. In other words, all sub-images are produced and distributed simultaneously.

Canny edge detection is then applied on each sub-image independently and simultaneously. Thus, all sub-images are processed in parallel.

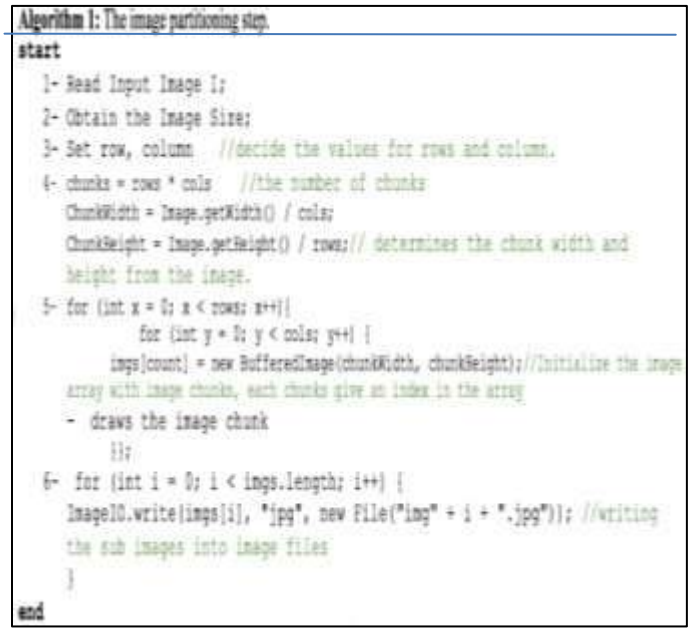


Figure 2. The Pseudo-code of the the image partitioning.

-Image Merging

As soon as a sub-image is processed; it is sent to a specified node for merging. This results in the original image after being subject to the Canny operator. Fig. 3 illustrates the pseudo-code for merging the sub-images. First, the same chunk size - in terms of numbers of rows and columns - as previously determined in the partitioning step is used. Then, all sub-image files are sent to a specific node that is responsible to execute the merging process. Finally, all sub-images are read in order, to get the final original image after applying the Canny edge detection.

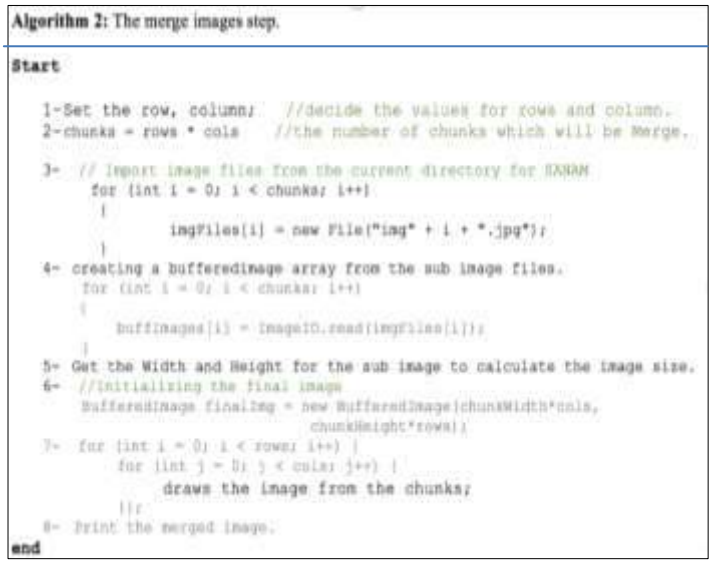
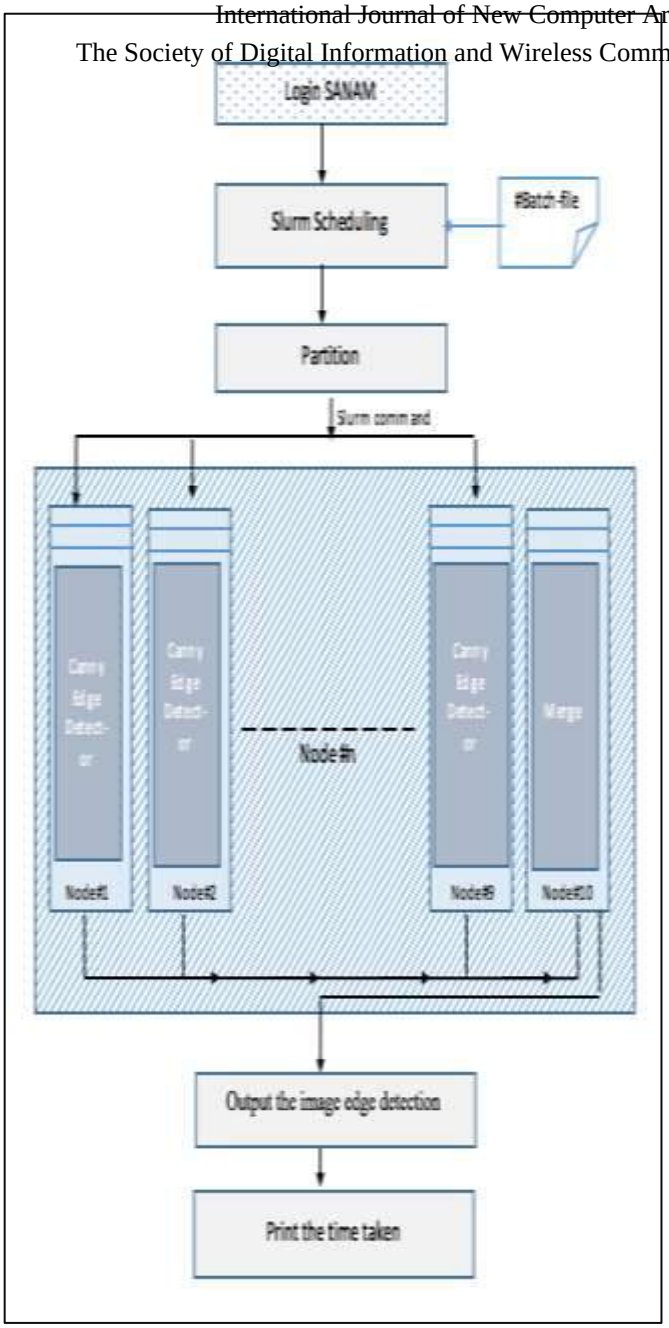


Figure 3. The Pseudo-code of the image Merging partitioning.

- Applying Canny Filter

The parallel edge detection algorithm follows the data parallelism which is most suitable to almost all image processing applications [1]. Parallel image processing is suitable for the clusters environment if there are little sequential dependencies. Consequently, there is minimum message passing involved between the computing processors [16]. Each node applies the sequential version of the Canny algorithm on different partitions simultaneously. The diagram in Fig. 4 depicts the previously detailed steps of the approach for edge detection cluster.

Figure 4. The architecture of parallel cluster.



5 PERFORMANCE METRICS

The performance of the parallel algorithm is measured by comparing it to the sequential algorithm. Three main metrics are used to measure such improvement; namely, the speedup, the efficiency, and the scalability. In addition, the experiments target to find the number of partitions that give the highest speed up in the parallel clusters environment. In addition, the accuracy of the edge detection is of major concern. These are detailed in the following sections

- Speed up

The speedup (S) is defined as the ratio of the execution time of the serial implementation (*Tseq)* over the execution time of the parallel implementation*(Tpar)*. Let *T(N, 1)* be the time required for the best serial algorithm to solve a problem of size *N* on one processor; and *T(N, K)* be the time for a given parallel algorithm to solve the same problem of the same size *N* on *K* processors. Thus, speedup is defined as [12]:

$$S(N, K) = \frac{T\,seq\,(N,1)}{T\,par\,(N,K)} \quad (1)$$

-Efficiency

The Efficiency (*E*) is a metric that identifies how close a program is to the ideal speedup. In other words, it measures the extent to which the program uses the hardware resources efficiently. *E* is a fraction between 0 and 1. A program whose *E* is closer to 1 makes better resources utilization. As the program's efficiency is closer to **1**, as it is closer to the ideal status. Efficiency is therefore measured using the following formula [13]:

$$E(N, K) = \frac{\text{Actual Speedup}}{\text{Ideal Speedup}} = \frac{\text{Speedup (N,K)}}{K} \quad (2)$$

-Scalability

The scalability of a program measures its ability to handle larger amounts of data. A program's size-up is the size of the parallel version running on *K* processors relative to the size of the sequential version running on one processor for a given running time *T*. This is expressed in the following formula [13]:

$$\text{Size-up}\,(T, K) = \frac{N\,par\,(T,K)}{N\,seq\,(T,1)} \quad (3)$$

6 EXPERIMENTAL RESULTS

In this section, we undergo a set of experiments on the parallel version of the Canny filter. The experiments' environment is detailed in Section 6.1. Section 6.2 explains the followed methodology in conducting all experiments. Section 6.3 investigates the optimum number of partitions that gives the

least execution time. Section 6.4 explores the speedup of the parallel program. Section 6.5 and Section 6.6 examine the efficiency and scalability respectively. Section 6.7 demonstrates the validity of the parallel program in edge detection. Finally, Section 6.8 compares our results with previous similar works.

6.1 Experiment Environment

In this research, the Canny image edge detector is implemented in two versions: the sequential, the and the parallel clusters. Both versions are implemented on the Saudi SANAM supercomputer.

SANAM belongs to King Abdel-Aziz City for Science and Technology (KACST). It is ranked second in the Green500 worldwide list of the most energy-efficient computers, as per the listing of November 2012. The KACST's supercomputer comprises standard servers connected via the high-speed network InfiniBand. It consists of 210 servers with 3,360 processor kernels [17]. In terms of computing speed, SANAM is about 40% faster than the German supercomputer "LOEWE-CSC". In addition, it requires only one-third of the power per computing operation. This is achieved by using a larger number of high-performance graphic chips in conjunction with software optimizations, as well as by using energy-efficient storage chips.

SANAM works under the Linux operating system. On the other hand, it uses Slurm: a highly scalable cluster workload manager and job scheduling system. It is available as an open-source basis and can deliver fault-tolerant cluster workload management for Linux clusters of various sizes [19]. A server includes two Intel Xeon E5−2650 CPUs (8 Hyper-Threading-enabled cores per processor). Each of the 256 server nodes has two AMD FirePro S10000 dual GPUs. Every node contains 128 GiB of main memory and has an on-board InfiniBand HCA [18].

6.2 Methodology

The measurement of the run time is a basic step in this research for both sequential and parallel codes. This is essential since the speedup is measured in terms of the execution times of both versions. However, practically speaking, the execution time of a program is rarely the same for multiple successive runs. This is because the underlying operating system is active all the time. Since it is impossible to control the OS activity, then a program is run multiple times (from 7 to 10 times), and the minimum value is recorded for the experiment. Taking the minimum value corresponds to the least interference of the operating system activities during the program run.

6.3 Optimum Number of Partitions

On the other hand, Table 1 depicts the execution time of the parallel version run on 10 nodes with different number of partitions. The parallel version is run on the same images used with the sequential version.

The table proves experimentally that for each image size, the execution time decreases with the increase of the number of partition. However, there is a different turning point for each image size: these are 50 partitions for the 1024x1024 image, and 60 partitions to the other two image sizes. At these turning points, the execution time begins to increase again. Fig. 5 depicts this relationship.

This is due to the increasing efforts to partition/merge larger number of sub-images. In addition, the limitation of the available hardware resources with respect to the number of partitions affects the execution time.

It is also noticed that the execution time increases with the increase of the image size for the same number of partitions. This is explained to the fact that the chunk sizes are equal for the three sizes; therefore, the number of sub-images increases with the increase of the image size.

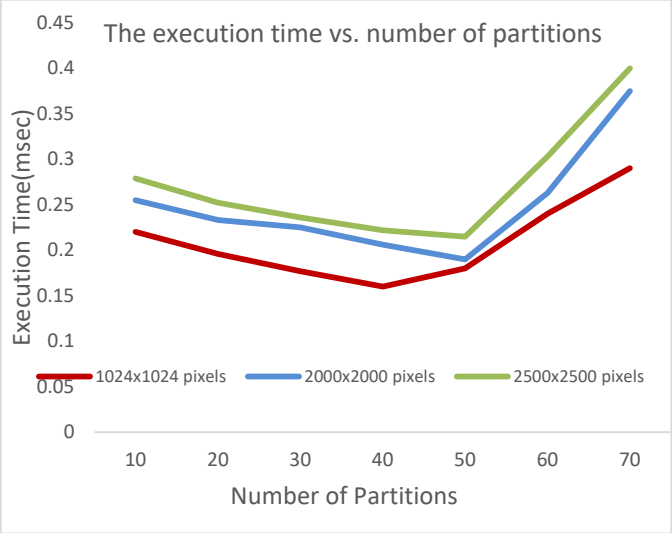


Figure 5.Relation of Execution Time with Number of Partitions.

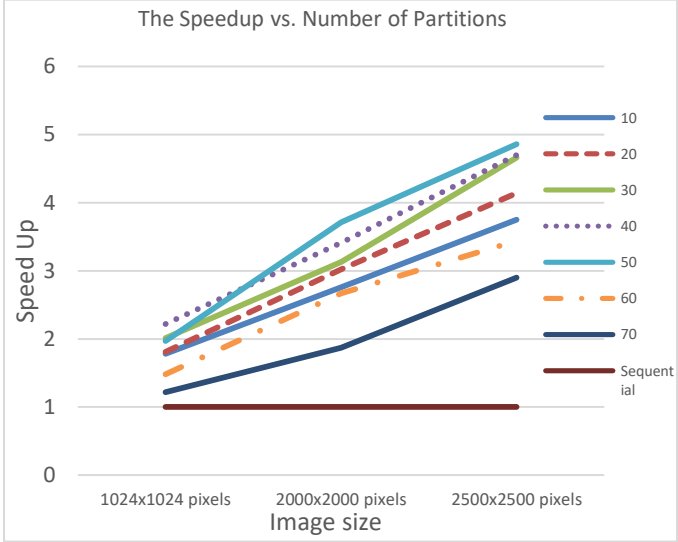


Figure 6. The speedup of SANAM Canny edge detection for the different number of partitioning.

6.4 Measuring Speedup

In this set of experiments, our aim is to test the performance of the parallel algorithm on Clusters. The sequential program is run on a single node; whereas the parallel version is destined to run on 10 nodes. In the parallel version, the same program runs with different sets of data on each node. Experiments are applied on three image sizes to measure the running time according to the methodology previously explained in section 6.2. The results of the sequential run are shown in Table 2.

It is noticed from Table 2, that the execution time is directly proportional with the image size. These results are expected since more processing is needed for larger images.

The minimum execution times for the parallel version are taken for the three image sizes. These correspond to number of partitions equal to 40, 50 and 50 for 1024x1024, 2000x2000 and 2500x2500 pixel images respectively. By applying formula (1), the speedup is found to be 1.97, 3.71, and 4.87 respectively for the three image sizes. More comprehensive results on all numbers of partitions are illustrated in Fig. 6.

From the graph, it may be concluded that the parallel version reduces the execution time of the sequential Canny filter. In addition, it is noticed that it is more beneficial to use the parallel version in case of larger images since it yields the maximum speed-up.

6.5 Efficiency

Table 3 and Fig. 7 depict the results of the efficiency experiments numerically and graphically respectively. Obviously, the efficiency increases with the increase of the number of partitions till a peak at which it starts to decrease. The maximum obtained efficiencies are 0.22, 0.37 and 0.49 for the three images respectively.

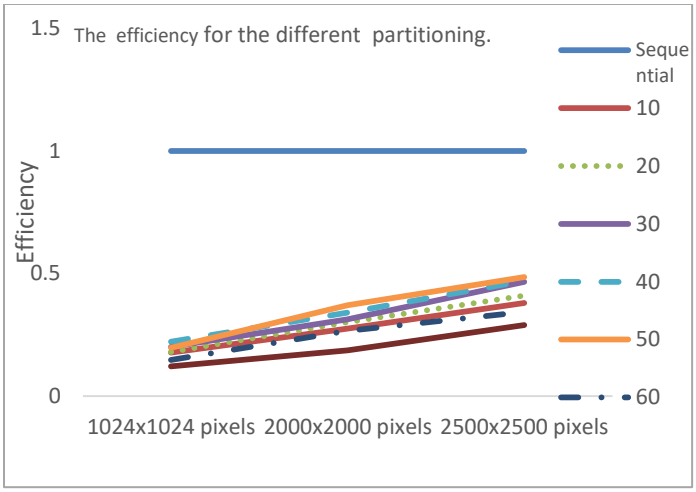


Figure 7. Relationship of Eefficiency of the parallel Canny with Number of Partitions.

6.6 Scalability Measurement

The scalability of a parallel program is performed as follows: The sequential program is run many times with differenct image sizes till the run crashes. The last size before the crash is recorded as *Nseq*. This represents the largest problem size that can be handled by the sequential program. The same step is repeated for the parallel program. *Npar* represents the largest size that can be handled by the program. Then, the scalability factor, or the sizeup, is calculated according to formula (3) above.

Since the sequential program runs on a single node, and the parallel version on ten nodes, *Nseq* and *Npar* are denoted as *N1* and *N10* respectively.

Fig. 8displays the execution time of different image sizes till the program crashes. The blue and red lines in the graph represent the results of the sequential and the parallel versions respectively.

From the graph, the program crashes at sizes equal 5Kx5K and 9Kx9K for the sequential and parallel programs respectively. Therefore, the sizeup is calculated as follows:

$$\text{Scalability factor (10 nodes)} = \frac{N10}{N1} = \frac{9000}{5000} = 1.8$$

A shared memory processor (SMP) environment lacks scalability [10]. However, this limitation is when using SANAM clusters, which can deal with the images of larger sizes.

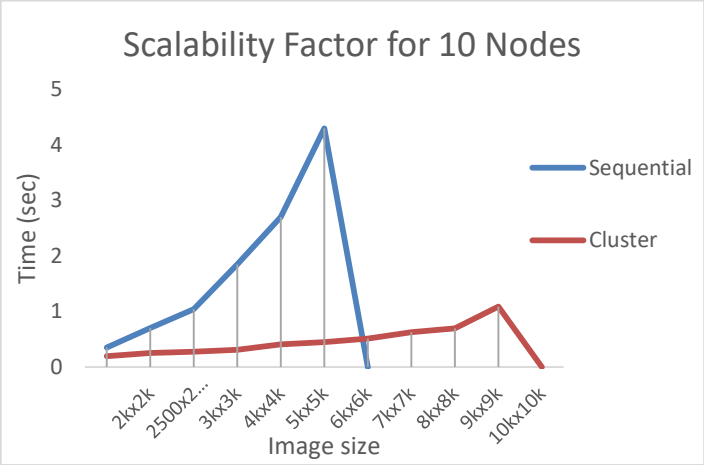
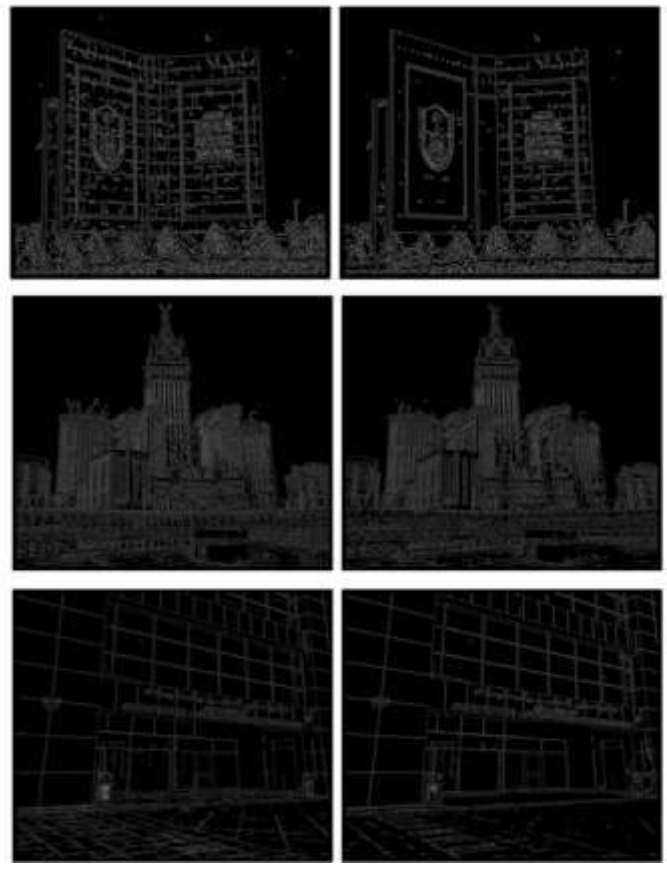


Figure 8. Relationship of Exectution time with Image Size.

6.7 Edges Accuracy

In this section, the accuracy of the parallel program is investigated. The importance of such experiment is to ensure the validity of the parallel program. Fig. 9 shows the original images in (a); the resulting detected edges using sequential program and the Cluster programs are shown in (b) and (c) respectively. Obviously, the edges are more precise in the images resulting from the parallel program compared to their counterparts produced from the sequential program.

(a)



(b) **(c)**

Figure 9. The edge detection results of the three images by using sequential and Cluster versions.

6.8 Comparison with Similar Work

In this section, the speedup results are compared to previous similar work. Our comparison is limited to those studies that used the same sizes of images as our experiments for more accurate conclusions.

In [11], they used an image 1024x1024 pixels in their experiment by using only 4 clusters. Our program outperforms by 81.98%. In [12], the experience is conducted on an image of size 2000x2000 pixels. SANAM gave a better result rate by 81.13%. Finally, in [13], they applied the approximation experiment using 10 clusters. The results are almost the same, slightly in favor of SANAM's result with a percentage close to 0.81% The comparisons are summarized in Fig. 10.

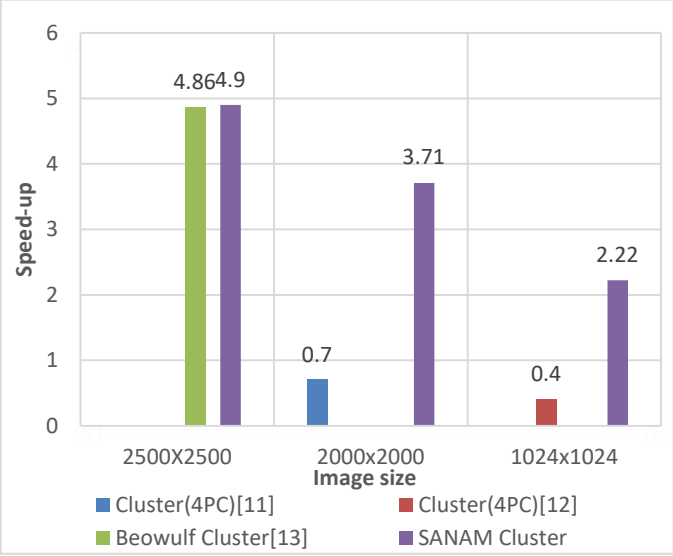


Figure 10. Comparison with Similar Work.

7 CONCLUSION

This researchimplements the Canny edge detection algorithm on parallel clusters. Ten nodes are used on KACST's SANAM supercomputer.

The image is partitioned, the Canny filter is applied on each sub-image simultaneously, and then the sub-images are re-collected and merged to give the edges of the big image.

Experiments target to find the optimum number of partitions. Also, the speedup, the efficiency, and the scalability of the parallel program are investigated. In addition, the accuracy of the detected images is examined. It is found that the speedup, the efficiency, and the scalability are drastically increased. In addition, the edges are more precise as compared to the results of the sequential program.

Our work is compared to previous similar work, and it is found that it gives the highest speedup.

ACKNOWLEDMENT

We would like to extend our gratitude to King Abdel-Aziz City for Science and Technology (KACST) in Riyadh for allowing us to use their SANAM supercomputer at all times. Not only this, but the staff was also of great support and helpful; replying to our posed questions promptly albeit their heavy duty load.

REFERENCES

[1] M. Raman R., and H. Aggarwal. "Study and comparison of various image edge detection techniques". International Journal of Image Processing (IJIP), Vol. 3, No. 1, 2009.

[2] A.-A.-N., Y. Kong, and M. N. Hasan, "Performance analysis of Canny's edge detection method for modified threshold algorithms," 2015 International Conference on Electrical & Electronic Engineering (ICEEE), 2015.

[3] B. Parhami. "Introduction to Parallel Processing: Algorithms and Architectures". Boston, MA: Springer US, 2002.

[4] M. Raman R., and H. Aggarwal. "Study and comparison of various image edge detection techniques," International journal of image processing (IJIP) Vol.3 no.1, 2009.

[5] A.-A.-N., Y. Kong, and M. N. Hasan, "Performance analysis of Canny's edge detection method for modified threshold algorithms," 2015 International Conference on Electrical & Electronic Engineering (ICEEE), 2015.

[6] M. R. Vahidi, M. M. RiahiKashani and A. Bagheri, "An efficient gradient based algorithm for improving performance of image edge detection," International Journal of Computer Applications, 103(4), 2014.

[7] A.-A.-N., Y. Kong, and M. N. Hasan, "Performance analysis of Canny's edge detection method for modified threshold algorithms," 2015 International Conference on Electrical & Electronic Engineering (ICEEE), 2015.

[8] A. Jain, A. Namdev and M. Chawla, "Parallel edge detection by SOBEL algorithm using CUDA C," 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-6, 2016.
[9] D. Yuefan. "Applied Parallel Computing," Singapore, US: Imperial College Press, 2012.
[10] NjoodAlAssemi, Soha S. Zaghloul, Phd. Speeding Up Canny Edge Detection Using Shared Memory Processing. International Journal of New Computer Architectures and their Applications (IJNCAA). Vol. 7, No. 3, September 2017, pp. 68 – 76.
[11] N. E. A. Khalid, N. M. Noor, S. A. Ahmad, M. H. Rosli, and M. N. Taib, "Parallel Laplacian Edge Detection Performance Analysis on Green Cluster Architecture," Communications in Computer and Information Science Digital Enterprise and Information Systems, pp. 308–316, 2011.
[12] W. Ju, J. Liu and S. Jin, "An improved clustering based on edge detection method," 2016 35th Chinese Control Conference (CCC), Chengdu, pp. 4026-4030, 2016.
[13] Haron N., Amir R., Aziz I.A., Jung L.T., Shukri S.R., "Parallelization of Edge Detection Algorithm using MPI on Beowulf Cluster," Innovations in Computing Sciences and Software Engineering, Springer, 2010.
[14] A. Muntasa, "Hybrid Method Based Retinal Optic Disc Detection," International Journal of New Computer Architectures and their Applications, vol. 5, no. 3, pp. 102–106, 2015.
[15] G. M. H. Amer and A. M. Abushaala, "Edge detection methods," 2015 2nd World Symposium on Web Applications and Networking (WSWAN), Sousse, pp. 1-7, 2015.
[16] A. Kaminsky, "BIG CPU, BIG DATA: Solving the World's Toughest Computational Problems with Parallel Computing" 2nd ed, Boston, MA: Course Technology, Cengage Learning, 2015.
[17] KACST The Saudi Supercomputer "SANAM" is the World's 2nd Leader in Energy Efficiency. [Online]. Available: https://www.kacst.edu.sa/eng/about/news/Pages/news3841117-3854.aspx . [Accessed: 28-Feb-2017].
[18] D. Rohr ndS.Kalcher., "An Energy-Efficient Multi-GPU Supercomputer," 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), Paris, pp. 42-45, 2014.
[19] "Slurm Workload Manager," Slurm Workload Manager. [Online]. Available: https://slurm.schedmd.com/. [Accessed: 05-Dec-2017].

Appendix

Table 1: Results of parallel execution time in SANAM.

Number of Nodes = 10 Nodes, Time= milliseconds.			
Image size / Partition	**1024x1024 pixels**	**2000x2000 pixels**	**2500x2500 pixels**
10	200	255	279
20	196	233	252
30	177	225	236
40	160	206	222
50	180	190	215
60	240	263	303
70	290	375	360

Table 2: Results of sequential execution time in SANAM.

Number of Nodes = 1 Node			
Image size	**1024x1024 pixels**	**2000x2000 pixels**	**2500x2500 pixels**
Time (milliseconds)	355	704	1045

Table 3: Efficiency of Parallel Program for Various Number of Partitions.

Number of Nodes = 10 Nodes			
Image size / Partition	**1024x1024 pixels**	**2000x2000 pixels**	**2500x2500 pixels**
10	0.178	0.276	0.375
20	0.181	0.302	0.414
30	0.201	0.313	0.466
40	0.222	0.341	0.470
50	0.197	0.371	0.490
60	0.148	0.267	0.344
70	0.122	0.187	0.290

International Journal of New Computer Architectures and their Applications (IJNCAA) 8(1): 11-20
The Society of Digital Information and Wireless Communications, 2018 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Measuring the Performance of Data Placement Structures for MapReduce-based Data Warehousing Systems

S. Kami Makki and M. Rakibul Hasan
Department of Computer Science
Lamar University
Texas, USA
kmakki@lamar.edu, mhasan4@lamar.edu

Abstract **-** The exponential growth of data requires systems that are able to provide a scalable and fault-tolerant infrastructure for storage and processing of vast amount of data efficiently. Hive is a MapReduce-based data warehouse for data aggregation and query analysis. This data warehousing system can arrange millions of rows of data into tables, and its data placement structures play a significant role for increasing the performance of this data warehouse. Hive also provides SQL-like language called HiveQL, which is able to compile MapReduce jobs into queries on Hadoop. In this paper, we measure the efficiency of these data placement structures (Record Columnar File (RCFile) and Optimize Record Columnar File (ORCFile)) in terms of data loading, storage and query processing using MapReduce framework. The experimental results showed the effectiveness of these data placement structures for Hive data warehousing systems.

Index Terms **- Big Data; Hive; MapReduce; ORCFile; RCFile**

1. INTRODUCTION

We are in the phase of fast technological advancement, where every sectors of any business is generating an unprecedented amount of data through its daily processes. Digitalization of every device and escalating the number of Internet users help to grow the data exponentially which makes the traditional data processing technology obsolete. According to ACI in 2012, 2.5 Exabyte's of data were generated in every day [3], and the total volume of data in the world is doubled every two years [10]. Statistics show that two billion people connected to the Internet in 2015, which is 100 times more than 1999 [8].

Therefore, the stunning growth in the number of users as well as variety of different devices that easily can connect to Internet generate huge and complex data or Big data. The authors in [1] defined the characteristics of Big data by the volume, variety, velocity and value. The Volume indicates the amount of data which is generated by the users, where this data has unknown value and low density. The Velocity refers to the rate of speed at which the data are generated and evaluated in real time manner to meet the users' demand and challenges. The Variety refers to the massive amount of data which is accumulated with huge speed, and can be of different types and nature such as structured, semi-structured and unstructured data. The Veracity indicates the quality of captured data, which totally depends on the source of data.

MapReduce framework provides a fault-tolerant infrastructure for processing of Big Data on large clusters. The MapReduce-based warehouse systems (such as Hive) are playing an important role for the effectiveness of web service providers and performance of social network websites. Hive stores large amount of data using distributed clusters [5], and efficient data placement structures are much needed factor for proper data organization. Hive uses two types of file format structures to store the data, such as RCFile (Record Columnar File) and ORCFile (Optimize Record Columnar File).

In this research, we investigate the effectiveness of RCFile and ORCFile of Hive data warehousing system. The goal of this research was to find out the most useful data placement structures that satisfy the requirements such as fast data loading, fast query processing, highly efficient storage space utilization, and strong adaptability to highly dynamic workload patterns. The rest of the paper has been organized as follow. Section 2, provides the background information on Big Data Technology, the Hadoop system architecture and Hive. In Section 3 we explore different data placement structures for Hive data warehousing system. Section 4 presents the performance evaluation of the RCFile and ORCFile for data loading, query processing and data storage. Finally Section 5 presents the conclusion.

2. BIG DATA TECHNOLOGY

2.1 *Hadoop*

Hadoop is a Java based open source system developed by Apache Software Foundation, which can store, process, and analyze a large volume of datasets in parallel on clusters of computers. Hadoop can scale up easily from a single server to hundred servers, where each server provides local computation and storage capability. Hadoop has four core modules: MapReduce, HDFS (Distributed storage), Yarn framework, and common utilities.

MapReduce is a programming model to process a large volume of datasets in parallel on clusters of computers [13]. MapReduce uses two methods: Map and Reduce. The Map method takes the raw data as input and breaks the data into numbers of smaller data sets. Each data set in a Map method receives a key/value pair and produces intermediate key/value pairs, and stores the output of a temporary storage system for further processing. The Reduce method combines all the intermediate key/value pairs based on the intermediate key and generates new sets of output. Besides Map and Reduce methods, shuffling is another process to transfer the data from Map processes to Reducers. The MapReduce tasks are executed in the local disk to avoid the network congestions, and the results will be sent to the appropriate servers [16, 17].

Hadoop has a file system named Hadoop Distributed File System (HDFS). The HDFS has master-slave architecture, which introduces master as NameNode, and slaves as a number of DataNodes [16]. NameNode controls the file system namespace and stores the metadata across the clusters. Metadata contains the information about where and which DataNode has stored which data files. The data files are broken into multiple pieces of blocks, and the size of each block is 128 MB by default. NameNode is responsible for namespace operation such as opening, closing, renaming files directories, and mapping blocks to DataNodes [4]. It does not control the block operation since DataNodes arrange the blocks whenever the system starts. Hadoop provides two mechanisms to make a NameNode consistent and protect it from the single point of failure. The first one is creating backup files of metadata to multiple file systems, and the other one maintains a secondary NameNode in a different machine. The secondary NameNode periodically merges the namespace images and keeps an updated copy in its own spaces. It provides a backup NameNode when original NameNode fails. Figure 1 shows the architecture of HDFS. A DataNode also stores data in different blocks in HDFS and allows read /write operations by clients [4].

DataNodes are responsible for performing block creation, deletion and replication according to the instructions by the NameNode. The NameNode and DataNodes are communicating by heartbeat messages every 3 seconds. A DataNode is considered to

be dead if it does not receive a message within a few seconds.

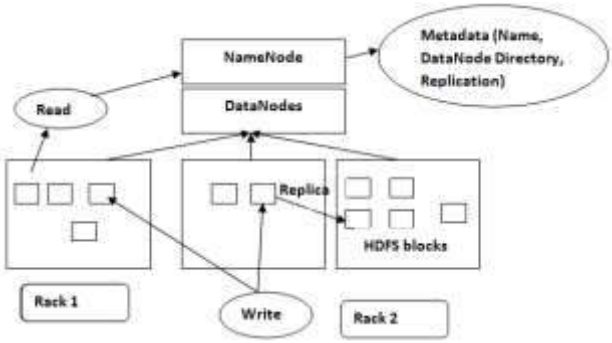


Figure 1. HDFS architecture [2]

2.2 Hive Data Warehousing System

The Apache Hive was developed by Facebook to manage its growing volume of data that Facebook produces everyday from its social networking activities [16]. Hive is included in Hadoop as a data warehouse. Hive supports SQL like scripts called HiveQL, to run a query on a large volume of data using MapReduce. Hive maintains metastore to contain schema and statistics for data exploration, query optimization and query compilation.

In Hive, data are organized in three formats, which are tables, partitions, and buckets. The tables are much like to the relational databases system. The second format is the partitions, which divides the data tables into subdirectories, which are defined by the data type characteristics. The last of data format is buckets, which can store data in both partitions and table's directory that depends on whether the table is partitioned or not [12,14]. Figure 2 shows the architecture and integration of Hive and Hadoop.

The External Interfaces supports different types of interfaces to initiate the works between users and HDFS, such as Command Line Interfaces (CLI), Web Interfaces, and programming interfaces (JDBC, ODBC).The Thrift Server supports cross-language services, which works with clients API to execute query statement. Metastore helps Hive to store the system catalog and metadata, where they contain details information about tables, columns, and partitions and so on [12]. Driver maintains sessions and statistics of HiveQL statement, which moves to Hadoop through Hive. Query Compiler compiles user defined HiveQLs into MapReduce tasks. Execution Engine is responsible for executing the tasks produced by compiler and MapReduce, which maintains the dependency order to communicate with the Hadoop modules.

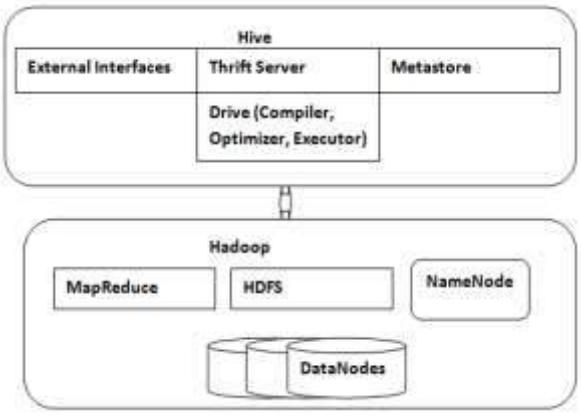


Figure 2. Architecture and integration of Hive and Hadoop

3. DATA PLACEMENT STRUCTURES

3.1 Record Columnar File (RCFile)

RCFile (Record Columnar File) is a data placement structure for MapReduce-based data warehouse systems (HIVE) which can organize a large volume of data on HDFS clusters. The RCFile is a combination of multiple features such as data storage format, data compression and data optimization techniques [2]. In data storage format, tables are stored first horizontally, and then vertically to organize each column independently in the clusters. RCFile supports column-wise data compression technique (lazy decompression) within each row group that helps to avoid unnecessary column reads

during query execution. Furthermore, RCFile allows to select flexible row group size and arrange the same row data in the same node that increases the performance of data compression and query execution [5]. Figure 3 shows the layout strucure of a RCFile.

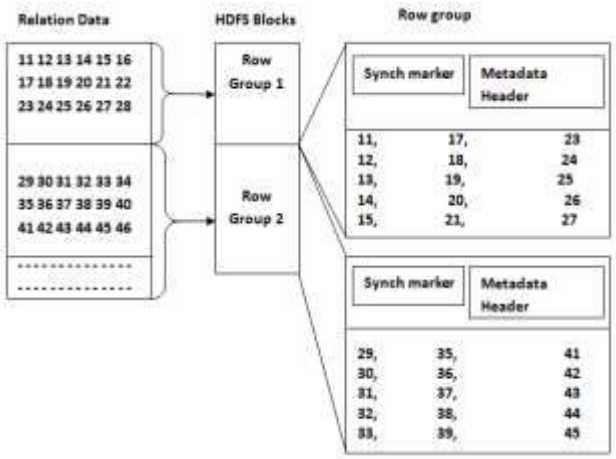


Figure 3. RCFile layout structure

A row group in RCFile has three components, which are synch marker, metadata header and column store. Synch marker is the beginning placement and helps to isolate the two-row groups in an HDFS block. Metadata stores the information of all row groups, such as how many row groups are placed, the size of each column as well as the size of each field in a column. The last one is column store, which helps to arrange all the fields in the same column together.

3.1.1 RCFile Data Compression

In RCFile, metadata header and table data sections are compressed separately. The metadata header section uses RLE (Run Length Encoding) algorithm to compress as a whole unit and the data table section compress each column independently [5].The RLE can find long run repeated values because all the values in a column are stored in continuously. For each column, RCFile supports separate algorithm to compress data in each table section. RCFile does not support random writing operation rather than it provides an interface for appending to the end of the file. RCFile maintains a memory column holder for writing data in each column. Before writing data to the disks, RCFile uses two parameters to control the memory buffer. The first parameter is to control the number of records, and the second parameter is to control the size of the memory buffer.

3.1.2 RCFile Lazy Decompression

In MapReduce framework for each HDFS block, a mapper is worked sequentially to process each row group. For reading, RCFile does not read the whole file rather than it reads only metadata and the corresponding columns to avoid the reading of unnecessary columns in the row group. Then the metadata header and compressed columns are loaded into the memory for decompression. RCFile uses lazy decompression technique that remains in memory until the other row groups are processed. Lazy decompression becomes useful when there is a `where` condition in a query, because if some row groups do not satisfy the `where` condition then those row groups do not need to be decompressed. For example, consider a table (T) with the following columns (`col1, col2, col3, col4,col5, …`) and if there is a query such as "`select col1, col3 from T where col2 = 2`". Then the RCFile only reads the metadata header in the row group and decompresses only those row groups that match the `where` condition (i.e. `col2 = 2`) of the above query, not other row groups that do not match the `where` condition, and therefore it saves time.

3.2 Optimize Record Columnar File (ORCFile)

3.2.1 ORCFile Structure

ORCFile is a data placement structure similar to RCFile for Apache Hive for

organizing and storing large data. ORCFile maintains one file for collections of rows, which is arranged in a columnar format that allows parallel processing of row collections in clusters [11]. ORCFile structure has three parts; these are stripes, file footer and postscripts. Figure 4 shows the structure of ORCFile [9].

Stripes hold the groups of row data and file footer maintains a list of stripes in the file, which contains the information of a number of rows per stripe and column's data type that includes aggregate functions such as count, min, max and sum. Each stripe size is 256 MB by default, which is suitable for a sequential read on HDFS. The larger block sizes will reduce the load of NameNode because the users can read more data from a single file.The last part is postscripts, which maintains compression parameter and the size of the compressed footer.

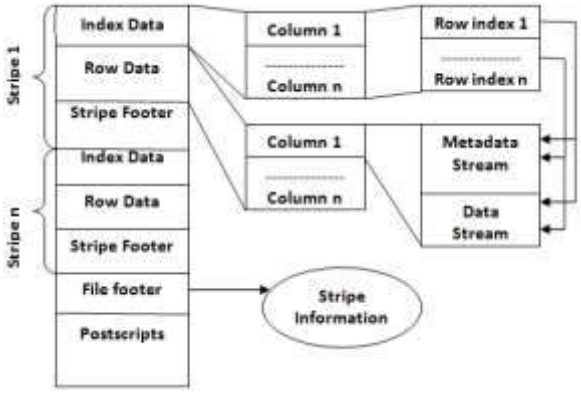


Figure 4. ORCFile structure

The stripes have divided further into three parts, which are index data, row data and stripe footer. Index data maintains the information of min, max values and the row positions for each column. These row positions are very efficient to find the specific compression and decompression blocks by providing the offset of indexes, where indexes are used to select the stripes and row groups. The row data stores multiple streams per column independently and uses them for table scans. Stripe footer provides directory services such as encoding types and stream locations.

3.2.2 Data Write and Compression

The ORCFile writer does not shrink the tables or whole stripes at a time rather than it applies data encoding and compression techniques [7]. To write data into HDFS, ORC uses memory manager to buffer the whole stripe in the memory. Due to large stripe size, ORCFile uses multiple writers concurrently to write data in a single MapReduce task, where memory manager controls the memory consumption of writers [6].

It supports two types of compression techniques. The first one is automatically used as type-specific encoding methods for columns with various data types and the second one is optional compression codecs set by users called generic compression. A column in a stripe can contain multiple streams, where each stream can be divided into four primitive types. These primitive types are bytes, run length bytes, integers, and bit field streams. Each stream uses the own encoding technique, which depends on the streams' data types. For example, integer columns data type are serialized into two streams, which are bit stream and data stream. For one bit or small integers, the variable length encoding is used, and for the long data streams of integers, the run length encoding technique is used. Besides using these type-specific encoding, users can also compress an ORCFile by general purpose codecs such as ZLIB, Snappy, and LZO [9] or others.

3.2.3 Data Read, Lazy Decompression and Lazy Decoding

In an ORCFile, the performance of data read is enhanced by lazy decompression technique [15]. Without lazy decompression and lazy decoding, a query seeks all the

stripes to bring out a specific column, which will take a long time to finish the MapReduce tasks. This decoding technique is used index stride that already existed in ORCFile format. The index stride helps the reader to skip unnecessary stripes and only decompress and decodes the target columns needed by the query. The footer in the ORCFile has all the stripes that contain the streams location. Thus, the users query only read the stripe lists to find the appropriate stripe location.

4. Performance Evaluation of RCFile and ORCFile

4.1 Experimental Setup

The effectiveness of distributed systems depends on how one can perform the read and write operations, where the format of stored data is an important metric for completing these operations successfully. In this paper, we choose the following three aspects (data storage space, data loading time, and query execution time) to determine the best data structure between RCFile and ORCFile.

In this work, we have set up a virtual Hadoop cluster, which consists of three nodes. This cluster works as a master-slave architecture, where master node maintains the workspace to distribute, store and replicate data to slave nodes. We have installed virtual box software in each machine to configure Hadoop environment. The operating system in each virtual box was Ubuntu 14.04.2 LTS 64-bit. The host windows machine has 8 GB memory with 3 GHz Intel Core i7 CPU, where each node in the virtual box has shared 8 GB memory with 60 GB disk.

In our experiments, we used Hadoop 2.7.1 and Hive data warehouse 1.2.1. The MapReduce was chosen as an execution engine since it is the default data processing engine used by Hive. The HDFS block size was set to 128 MB, and its replication factor was set to three. For these experiments, we have used 6 GB dataset, which contained movie reviews (5 million reviews) from Amazon. The dataset consisted of eight columns: Productid, Userid, Profilename, Helpfulness, Score, Time, Summary, and Text. All of them were string data type.

4.2 Performance Analysis

4.2.1 Data Storage Space

We have configured Hive with RCFile and ORCFile to measure the compression efficiency. Figure 5 shows their storage sizes with compression and without compression. We have used LZ4 compression technique to compress the data using RCFile and ORCFile. The figure shows both file formats have reduced the size of data set significantly. The RCFile reduces the data size from 6 GB to 3.29 GB, where ORCFile reduces even more to 2.01 GB. That is because ORCFile uses larger data blocks than RCFile.

Therefore, each block can arrange more data in column format which allows compressing each column independently. Without compression, there is not much difference between a text file and RCFile, but ORCFile can decrease the file size significantly compare with other file formats because ORCFile uses a default compression technique (ZLIB). So, the ORCFile provides better storage efficiency than RCFile, whether using compression technique or not.

4.2.2 Data Loading Time

To demonstrate the data loading time of RCFile and ORCFile, we have measured both MapReduce time and the total time to finish the job. Figure 6 shows the data loading time after compression, where it shows that ORCFile take more time to load data than RCFile. We have taken another result shown in Figure 7, which shows the loading time for these two files before using compression technique.

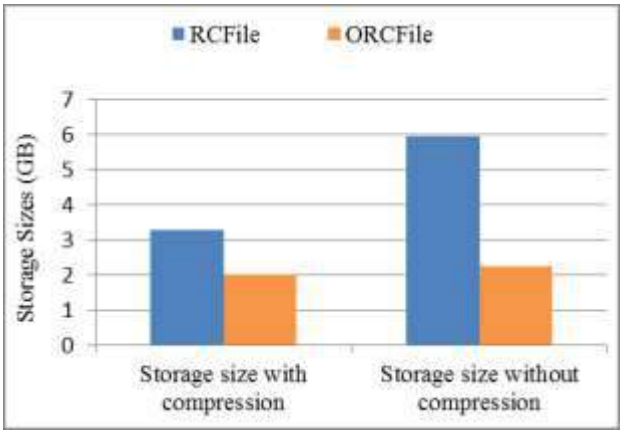


Figure 5. Data storage space of RCFile and ORCFile

4.2.3 Data Loading Time

To demonstrate the data loading time of RCFile and ORCFile, we have measured both MapReduce time and the total time to finish the job. Figure 6 shows the data loading time after compression, where it shows that ORCFile take more time to load data than RCFile. We have taken another result shown in Figure 7, which shows the loading time for these two files before using compression technique.

Figure 6. Data loading time with compression

4.2.4 Query Execution Time

For measuring the query execution time we used four queries as follow:

Query 1: `select productid from movie_rc;`

Query 2: `select reviewsummary from movie_rc where Profilename = "review/profileName: Jessica Lux" and userid = "review/userId: A2EBLL2OYEQJN9";`

Query 3: `select productid, userid, profilename from movie_rc where Score = "review/score: 50";`

Query 4: `select t1.productid, t2.userid from movie_rct1 right outer join movie_rc1 t2 on (t1.productid = t2.productid) where t1.profilename = "review/profileName: Jessica Lux";`

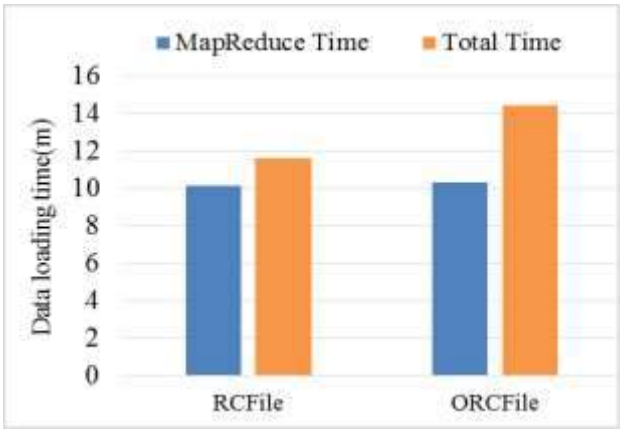


Figure 7. Data loading time without compression

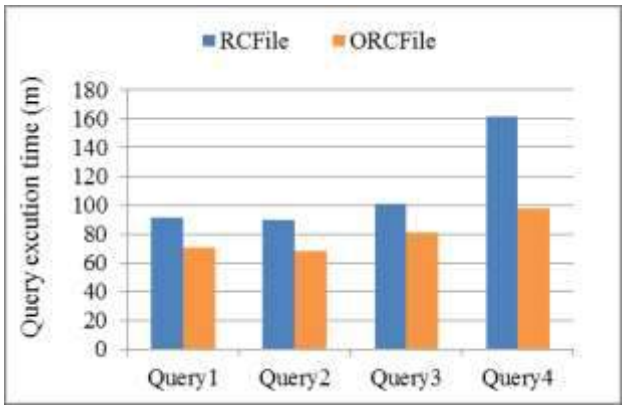


Figure 8. Query execution time with compression

Figure 8 shows the query execution time after data is compressed and Figure 9 shows the query execution time where data compression technique have been used. In both cases, ORCFile outperforms the RCFile significantly because the lazy decompression technique, which helps the ORCFile to skip a larger block of data if it does not match a query.

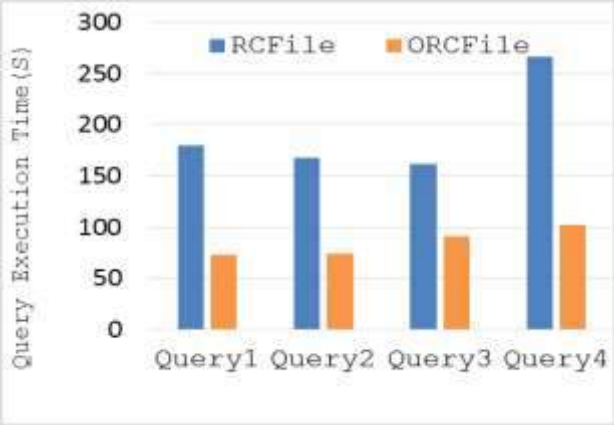


Figure 9. Query execution time without compression

4.3 RCFile and ORCFile with Different Row Group Size

Both RCFile and ORCFile allow the user to set flexible data block sizes because large data block can have better compression efficiency than a small one, where small data block may have better read or query performance than a large one. Furthermore, a large data block consumes more memory and can affect MapReduce tasks. In this experiment, we have used the same movie review database as above, and size of the dataset is 1.2 GB.

4.3.1 Data Storage Space

In this section, we have used different row group sizes for RCFile and different stripe sizes for ORCFile to demonstrate how they affect the storage space. Figure 10 shows the data storage efficiency of RCFile of different row group sizes (from 512 KB to 48 MB). Figure 11 demonstrates the data storage efficiency of ORCFile of different stripe sizes (from 4 MB to 256 MB).

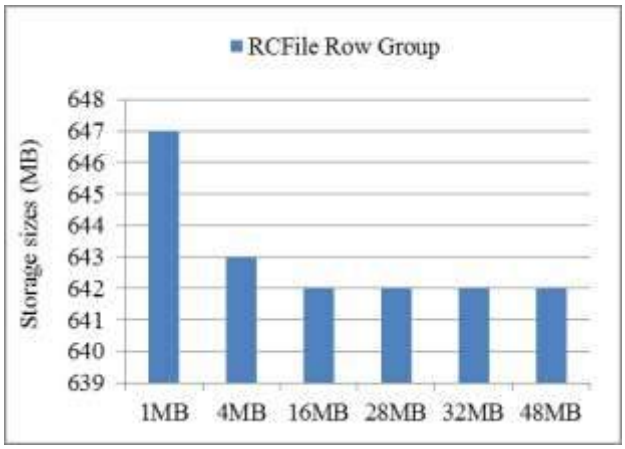


Figure 10. RCFile storage space with different row group sizes

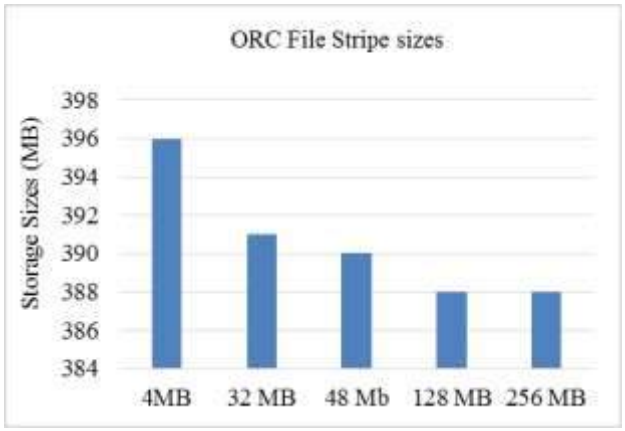


Figure 11. ORCFile storage space with different stripe sizes.

Figures 10 and 11 show that the larger block sizes can be compressed more than smaller block sizes. As a result, this reduces the storage space for both file formats. However, in the RCFile, when the row group size is larger than 4 MB, the storage space stays almost constant. As shown in Figure 11, compression is more in ORCFile when the stripe sizes are larger than 48 MB. So, the

ORCFile provides better storage space than the RCFile with larger data blocks.

4.3.2 Query Execution Time

In this experiment, we have evaluated the performance of lazy decompression technique for both RCFile and ORCFile. Figure 12 and 13 show the query execution time for RCFile and ORCFile. We have designed a query as below with different characteristic according to the `where` condition of a query.

```
Query 1: select review summary from movie_rc where PROFILENAME = "review/profileName: Jessica Lux";
```

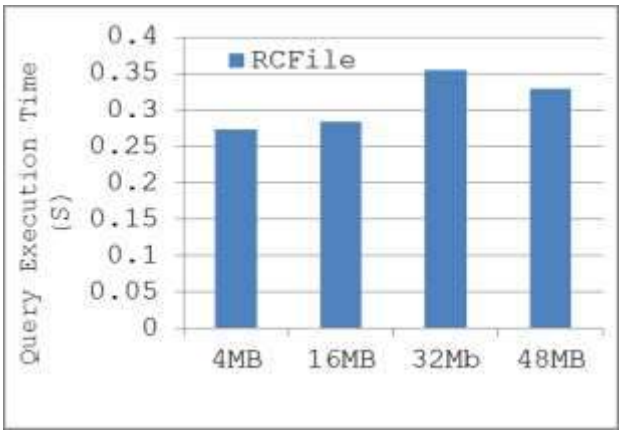


Figure 12. Query execution times of different data block sizes of RCFile

Figure 12 and 13 demonstrate that, when row group sizes are large, RCFile shows lower performance than ORCFile for query execution. The ORCFile has the better query performance when data block size is large (256 MB), because the ORCFile can skip large data block if it does not match the query based on lazy decompression technique.

5. CONCLUSION

There are four essential requirements for data placement structures, which are: to reduce the data loading time and storage space as well as enhancing the query performance and adaptivity of dynamic workload pattern. Our experimental findings showed that both file formats have significant characteristics which satisfy all four requirements of data placement structures mentioned above.

Between RCFile and ORCFile, the RCFile has a major inherent advantage in data loading time over ORCFiles. Since the RCFile has small row-group size than the ORCFile, which effectively reduces the data loading time. However, in the case of storage space and query execution time, the ORCFile outperform the RCFile. Though both data placement structures (i.e. RCFile and ORCFile) use column-wise compression, the large row-group size inside each stripe in ORCFile can hold and compress more data at a single time and reduces more storage space for ORCFile than RCFile. We have also observed that ORCFile can skip large numbers of unnecessary columns during a query, which significantly improves query performance. Thus, the ORCFile contains most of the performance benefits features and will play important role for data placement structures in MapReduce-based data warehouse systems on Hadoop.

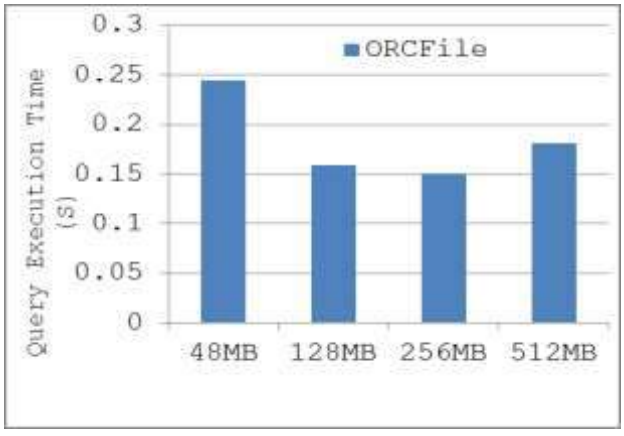


Figure 13. Query execution times of different data block sizes of ORCFile.

6. REFERENCES

[1] An Enterprise Architect ' s Guide to Big Data.

ORACLE ENTERPRISE ARCHITECTURE WHITE PAPER.

[2] RC/ORC File Format.http://datametica.com/ rcorc-file-format/

[3] The Data Explosion in 2014 Minute by Minute – Infographic. Retrieved January 8, 2017, from http://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/

[4] HDFS Architecturehttps://hadoop.apache.org/ docs/ stable/hadoop-project-dist/hadoophdfs/ HdfsDesign. html

[5] He, Y., Lee, R., Huai, Y., Shao, Z., Jain, N., Zhang, X., & Xu, Z. (2011). RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. *Proceedings - International Conference on Data Engineering*, 1199–1208. http://doi.org/10.1109/ICDE.2011.57679 33

[6] Huai, Y., Chauhan, A., Gates, A., Hagleitner, G., Hanson, E. N., Malley, O. O., … Zhang, X. (2014). Major Technical Advancements in Apache Hive. *SIGMOD '14 - Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 1235–1246. http://doi.org/10.1145/2588555. 2595630

[7] Huai, Y., Ma, S., Lee, R., O'Malley, O., & Zhang, X. (2013). Understanding insights into the basic structure and essential issues of table placement methods in clusters. *Proceedings of the VLDB Endowment*, *6*(14), 1750–1761. http://doi.org/10.147 78/2556549.2556559

[8] Internet live user. http://www.internetlivestats. com/internet-users/

[9] LanguageManual ORC https://cwiki.apache. org/confluence/display/Hive/Language Manu a l+ ORC#LanguageManualORC-orc-spec

[10] Big Data Solutions http://www.nttdata.com/ global/ en/services/bds/index.html

[11] ORC: An Intelligent Big Data file format for Hadoop and Hive http://www.semantikoz. com/blog/orc-intelligentbig-data-file-format-hadoop-hive/

[12] Thusoo, A., Sarma, J., & Jain, Zheng S., Prasad C., Zhang N., Antony S., Liu H, and Murthy R., (2010). Hive-a petabyte scale data warehouse using hadoop. IEEE 26th International Conference on *Data Engineering (ICDE 2010), pp.* 996–1005. http://doi. org/10.1109/ICDE.2010.5447738

[13] Hadoop - MapReduce. http://www.tutorial spoint. com/hadoop/hadoop_mapreduce.htm

[14] Hive Introduction http://www.tutorialspoint. com/hive/hive_introduction.htm

[15] Vagata, P., & Wilfong, K. (2014). Scaling the Facebook data warehouse to 300 PB. Retrieved January 11, 2016, from https://code.facebook.com/posts/ 229861827208629/scaling-the-facebook-data-warehouse-to-300 pb/?attachment_canonical_url= https%3A%2F%2Fcode.facebook.com%2Fposts%2F229861827208629%2Fscaling-the-facebook-data-warehouse-to-300-pb%2F

[16] White, T. (2015). *Hadoop: The definitive guide* (Vol. 54). http://doi.org/citeulike-article-id:4882841

[17] Apache Hadoop https://en.wikipedia.org/ wiki/ Apache_Hadoop

International Journal of New Computer Architectures and their Applications (IJNCAA) 8(1): 21-32
The Society of Digital Information and Wireless Communications, 2018 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

Comparison of Parallel Simulated Annealing on SMP and Parallel Clusters for Planning a Drone's Route for Military Image Acquisition

Eman Alsafi and Soha S. Zaghloul, PhD
King Saud University
435204448@student..ksu.edu.sa
smekki@ksu.edu.sa

ABSTRACT

Drones are vastly used in many civil and military applications. However, there are many factors to be highly considered in military applications. Such factors should ensure the sensitivity and the secrecy of the mission. In order to send a military drone with the aim of acquiring images from multiple sites, the mission time should be the least possible. Therefore, the minimum route plan is required. Simulated annealing (SA) algorithm is one of the metaheuristics selected to generate a feasible solution to solve this problem. However, the time complexity is too high. Therefore, this research exploits the parallelism in the simulated annealing with the aim of accelerating the time to find a suitable solution. Parallel programming divides the problem into smaller independent tasks, and then executes the sub-tasks simultaneously. Two parallel versions are therefore developed on different environment: synchronous SA on SMP, and asynchronous SA Complete Search Space (CSS) on parallel clusters. Experiments are conducted on the parallel clusters environment of the SANAM supercomputer. This research details the CSS, and compares it with the SMP SA developed in our previous study. Comparison is made in terms of speedup, efficiency, scalability, and quality of solution.

KEYWORDS

Parallel processing; Simulated annealing; Parallel Simulated Annealing; Shared-memory Processor; Parallel Cluster; SANAM

1 INTRODUCTION

Recently, drones or Unmanned Aircraft Vehicles (UAV) became very popular. This refers to their ability to undergo dangerous missions without exposing human beings' lives to any type of danger. Drones are associated with sensors and devices such as cameras, computing units, communication tools, and others. They are remotely controlled [1,2].

Drones are utilized in diverse military and civilian applications. Examples include, but are not limited to, aerial surveillance, image acquisition, remote sensing, and scientific research [2]. In addition to saving human lives, drones complete missions quickly with minimum cost [1, 3]. On the other hand, the main restriction imposed on a drone is its limited energy; and therefore, flight time. Consequently, one of the main challenges when dealing with drones is to find an effective route plan in the minimum possible amount of time [4].

As drones usually follow preloaded instructions without human intervention, the route plan may be generated either online during the flight, or offline before taking off. Moreover, drones route planning becomes more challenging when there are several geographical locations to be visited that are dispersed apart; these are called waypoints [2]. This research targets for finding a route plan that allows drones to acquire images from predefined waypoints in the least possible amount of time. Each waypoint is to be visited exactly once. Obviously, this is analogous to the well-known Travelling Salesman Problem (TSP). Finding a near-optimum route plan is necessary to minimize the drone's power consumption during the flight to cover the largest possible geographical area; and therefore, visit the largest number of targeted waypoints. In addition, achieving the mission in the minimum possible time ensures its secrecy.

However, solving TSP problem using a brute-force approach requires a significant amount of time to try every possible solution [2] . This approach is not suitable for the problem in-hands, since time and secrecy are both important factors in military applications. Therefore, a metaheuristics algorithm, the simulated

annealing (SA) algorithm is used. SA is capable of finding an acceptable local optimum route plan [5]. Although SA is used to solve several complex problems, but it requires significant processing time to find a suitable solution [6]. Therefore, parallel computing is expected to positively contribute in the solution of this problem. Parallelism may minimize the execution time to fulfill the requirements of the military mission. In addition, it may increase the chance to provide a better-quality plan. However, the SA is inherently sequential as each new solution depends on the previous one. Therefore, this imposes one of the challenges associated with parallelizing SA. The improvement of the parallel computational power can overcome this challenge. In this research, we aim to study the parallel SA on SANAM supercomputer.

The performance of the parallel program is measured in terms of three metrics; namely, the speedup, the efficiency, and the scalability [7].

Therefore, the aim of this research is to design a parallel SA implementation with the purpose of generating a route plan for military drones emitted with the intention of acquiring images at multiple sites. Therefore, the program speedup, efficiency, and scalability are to be maximized; while the final distance should be at its minimum.

The layout of this paper is as follows: Section 2 exposes similar work in the literature. Section 3 explains the design of the asynchronous CSS. Section 4 reports the experiments' results. The paper is then concluded in Section 5 with a hint to our plan to future work.

2 RELATED WORK

The drone route planning problem in this research is analogous to the TSP. There are several algorithms that provide a solution for the TSP, such as LS, BB, EAs, ACO, and hybrid algorithms [7]. This chapter exposes the sequential and parallel solutions to the TSP in general, and emphasizes on the drone route planning.

Many sequential algorithms are proposed to solve the TSP, some of which are based on the ACO algorithm. The proposed solution in [8] provides a modification of the traditional ACO method; this is known as the High Performance ACO. The traditional ACO algorithm involves a single ant randomly looking for the path; whereas the updated algorithm applies the TSP on a group of ants. The authors provide a comparison between their proposed algorithm and the ant colony system algorithm on various number of nodes. They found that the proposed algorithm completes the task in less time.

Also, Local search algorithms are widely used to solve the TSP. The research in [7] provides an experimental study to test the performance of the Lin-Kernighan and the Multi-Neighborhood Search. Results show that the Lin-Kernighan provides better results than the Multi-Neighborhood Search in terms of runtime.

On the other hand, several parallel solutions are proposed in the literature to solve the TSP using diverse parallel programming platforms. In [9], the experiment is performed on a standard multicore CPU. The reported results indicate that a gained speedup of 7.3 on 8 cores. Thus, the usage of PSO algorithm is more suitable for real-time planning for the drone. Moreover, the experiments also proved that the performance of the GA is better than the PSO. The same authors improved their results in [10] by proposing a parallel hybrid algorithm that exploits the advantages of both the PSO and the GA to generate a suitable path plan for fixed-wing drones. It is found that the gained speedup is 10.7 on a 12-core SMP.

In [2], the authors planned the drone's path using parallel ACO solution on both GPU and CUDA platforms. The generated path guides the drone in disseminating keys and collecting data from wireless sensors, which are previously deployed at minimum cost. The drone launches from a station, visits all sensors in a limited period of time, then returns back to the same station it is emitted from. In their experiment, they compared the sequential performance with the parallel implementation performance. They showed that the speedup is higher when using GPU platform.

In [5], the authors generate multiple route paths for several drones simultaneously using synchronous parallel SA on the GPU. Experiments' results prove that the processing

time is reduced, and a better solution is acquired, as compared to the CPU implementation.

3 DESIGN AND IMPLEMENTATION

This section discusses the general design of SA. Then, the implementation of the asynchronous CSS is then discussed.

3.1 Simulated Annealing Design

Two concepts are to be defined when it comes to designing an iterative metaheuristics algorithm; namely, the solution representation and the objective function. Since SA is classified as a single-solution based metaheuristics, it requires the definition of the neighborhood. These are detailed in the following subsections [11].

3.1.1 Solution Representation

The solution representation of the drone route planning is a permutation of size *n*, where *n* is the number of the waypoints to be visited exactly once. Each permutation represents one solution as shown in Figure 4 as a sequence of nodes, where each node represents a waypoint and its index represents the corresponding order. The number of all permutations that represent the solution space taking into consideration the fixed point of start (ground station) is *(n-1)!*.

Figure 1. The permutation representation of the drone route plan problem

3.1.2 Neighborhood Solution

The neighborhood of a solution is found by performing a move operator which leads to a tiny perturbations to the solution *S* [11]. As the drone route plan is represented by a permutation, a neighborhood is generated by the swap operator between two elements in the solution. This is illustrated in Figure 5.

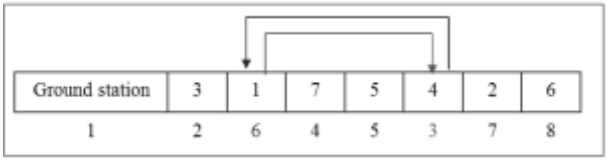


Figure 2. Neighbourhood solution generated by swapping the order of two waypoints

3.1.3 Objective function

The objective function is used to define the goal to be achieved by the SA. The goal of the problem in-hands is looking for the shortest route plan for a drone such that it visits each waypoint exactly once. As previously mentioned, this is similar to the TSP and has a similar objective function which is shown below:

$$f(\pi) = Min \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \quad (1)$$

where:
- π is a permutation representing a tour of the drone;
- *n* is the number of waypoints.

3.2 Sequential simulated annealing

IAs previously mentioned, the SA, like other single-solution based metaheuristics, includes two main steps. The first step is to generate the initial solution, which is constructed by using a greedy heuristic, such as the nearest neighbor algorithm or randomly. In the design of the sequential algorithm of this research, the random method is used because the greedy heuristics produces a solution in local optimum, which may not be able to provide an improved local optimum solution at the end [11].

The second step, which is the solution improvement, the design uses the swap operator between two points to generate a neighbor solution from the current solution.

In fact, the SA algorithm imitates the process of the solid hardening, which depends on the initial temperature value and the cooling rate. Therefore, the SA implementation consists of two main loops to provide a suitable solution. The outer loop, known as the cooling loop, is responsible for managing the temperature value. On the other hand, the inner loop, known as the equilibrium state loop, is responsible for constructing a neighbor solution from the current one, evaluating it, and computing the probability of the acceptance using the following formula:

$$e\ (-\Delta/T) \qquad (2)$$

where:

- Δ is the difference between the cost of the old and the new solution;
- T is the current temperature

Accordingly, the main parameters that are to be defined during SA implementation are the initial temperature, the cooling rate, and the stopping condition. The latter might be the minimum temperature or a specific number of iterations. In this research, the stopping condition is taken based on a minimum temperature. The other parameters are determined after several experiments [6]. The flowchart of the sequential algorithm is shown in Figure 3.

3.3 Parallel Simulated Annealing

Metaheuristic algorithms are sequential by nature; SA is no exception. Consequently, parallelizing the SA entails a challenging problem [7]. Many approaches are proposed to parallelize SA algorithm [12]:

- Decompose the search space into smaller parts, then assign each part to a processor to find the minimum cost and share its result with other processors.
- Apply the synchronous approach, where each processor uses the same initial solution and performs parallel improvements within the same temperature. Then at each temperature value the best solution is shared between the processors to perform parallel improvement until the end. Figure 4 illustrates the synchronous approach.
- Apply the asynchronous approach, where each processor executes SA independently. The initial solution may be the same or different across the processors. Finally, compute a reduce operation to get the best solution among them. As illustrated in Figure 5.

The synchronous parallel SA on shared-memory processor (SMP) is previously studied in [13]. In this paper, the asynchronous parallel SA Complete Search Space (CSS) is investigated. The CSS algorithm starts with different initial solutions for the complete search space. The idea is illustrated in Figure 5.

3.3.1 Parallel SA approach on cluster

On the other hand, using the approach in [14] to implement SA on parallel clusters will increase the overhead of communication between nodes. This is explained by the fact that in synchronous

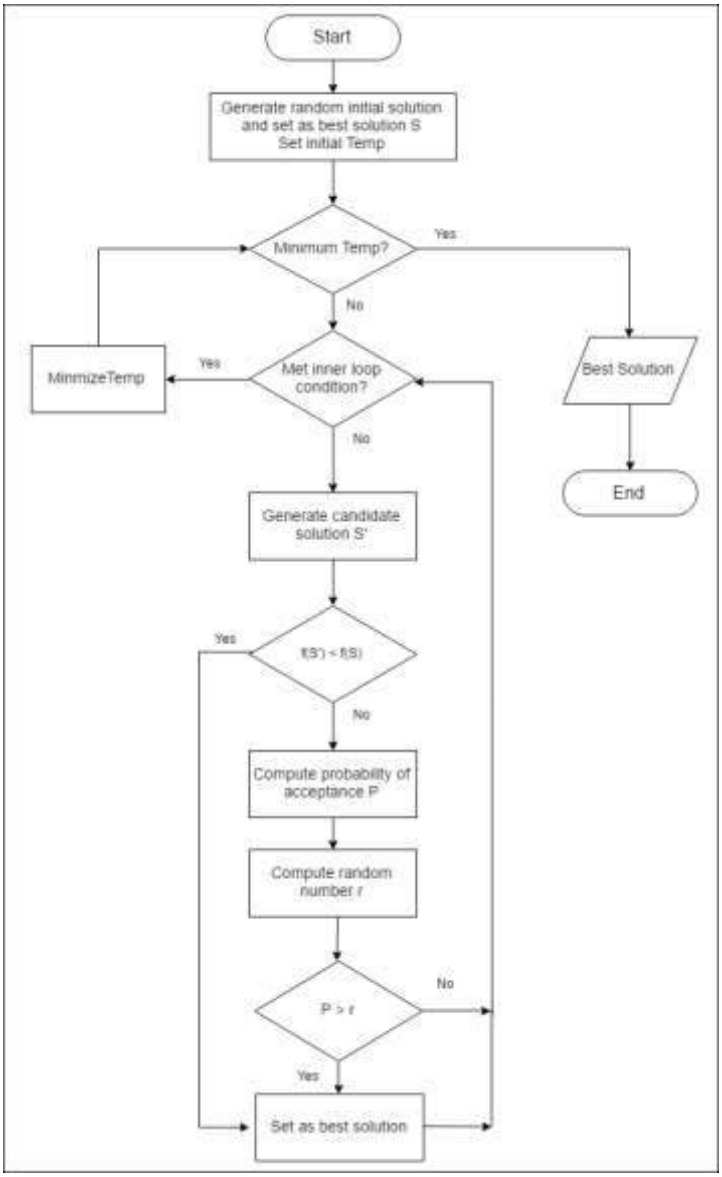


Figure 3. Flow chart of the SA algorithm

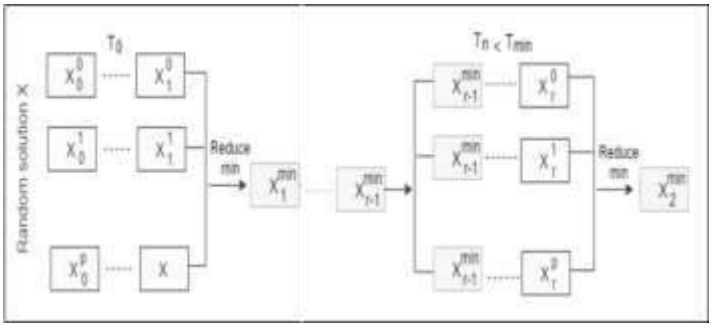


Figure 4. Synchronous SA parallel approach

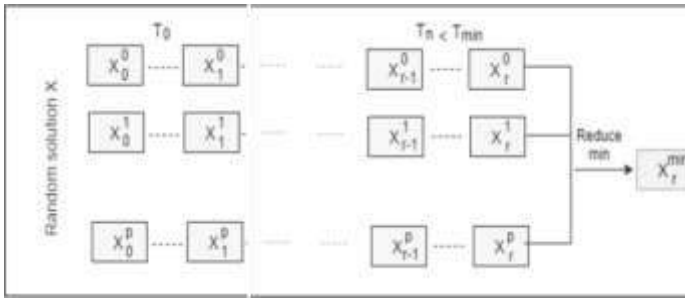


Figure 5. Asynchronous SA parallel approach

parallel SA, the processors frequently communicate with each other. Shared memory processor environment is suitable for such solution. However, in parallel clusters, communication is needed between processors after each inner loop. This is performed through message passing on parallel clusters. Message passing imposes an overhead on the program; and therefore, increases the speedup.

The asynchronous parallel SA CSS is therefore suggested to minimize the communication overhead. In this algorithm, each cluster node works on the complete search space (CSS) as illustrated in Figure 6. In the start, several initial solutions are generated and distributed over the nodes. Then each node applies the sequential SA. However, data parallelism is applied. At the end, all nodes send the produced route together with the final distance. The minimum distance with its corresponding route are then selected to be the best route plan.

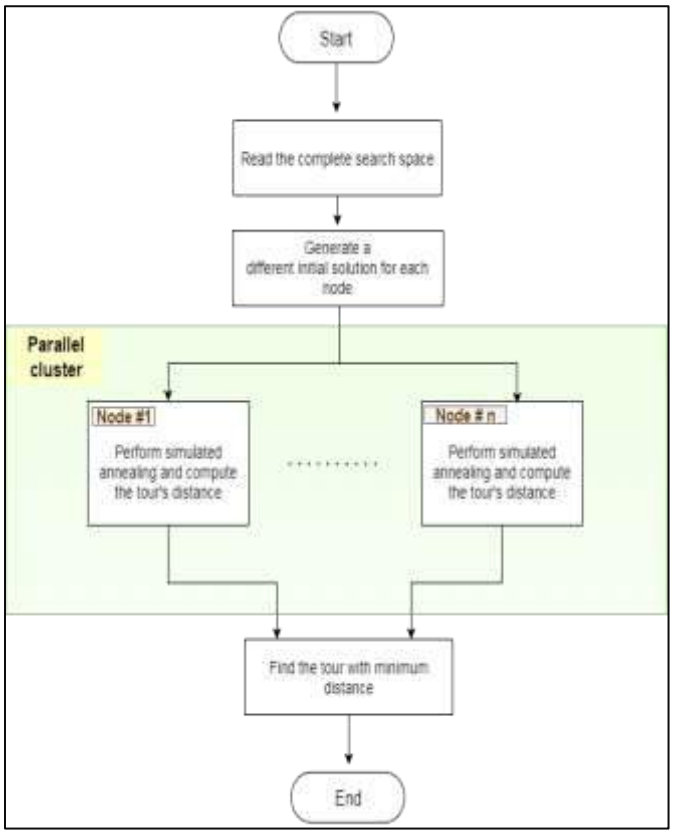


Figure 6. Flowchart of asynchronous parallel SA for complete search space approach

3.4 Handling the Drone Energy Constraints

After generating the route plan using SA algorithm at the ground station, the route is evaluated in terms of the energy required to complete the planned mission. If the energy level is above a predefined threshold, then the drone is emitted according to the planned route. Otherwise, the mission is divided into multiple journeys. The drone is re-charged after the end of each trip, before starting a new one.

In fact, the drone's energy is expressed in terms of its enduring lifetime L. Therefore, the time *T* needed to travel the final distance *D*, as planned by the SA, is calculated as follows:

$$T = D / S \qquad (3)$$

where:

- *T* is the time required to make the complete calculated tour, including the return trip to the ground station;
- *D* is the final distance as calculated by the SA algorithm;
- *S* is the drone's speed as specified in its hardware specifications

If the calculated time *T* is less than the drone's enduring lifetime *L*, then the drone is safely launched. Otherwise, the journey is broken into multiple trips. Figure 7 illustrates the previously mentioned steps. Therefore, the applied predefined threshold is the enduring lifetime of the emitted drone.

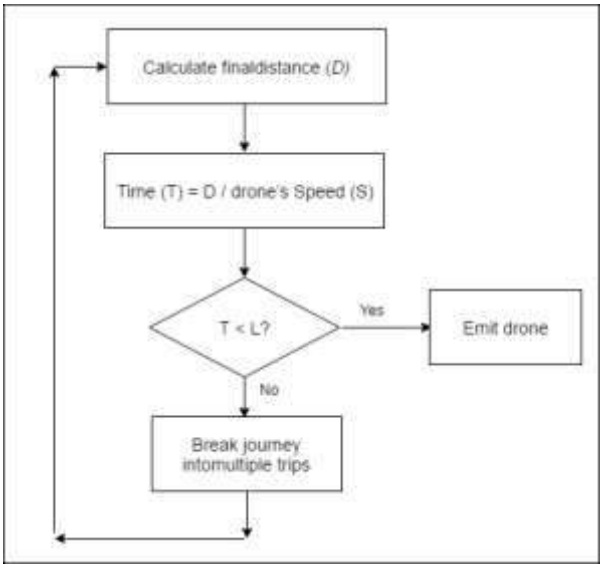


Figure 7. Checking drone's **energy**

4 EXPERIMENTAL RESULTS

This section details the methodology used in the conducted experiments. In addition, the obtained results are discussed, analyzed, and represented graphically. The first subsection reveals the deployed environment in terms of software and hardware specifications. Subsection 4.2 explains the performance metrics used to measure the effectiveness of the developed algorithm. Subsection 4.3 details the general methodology used to collect the figures of the experiments. Finally, subsection 4.4 details the results of both the sequential and parallel CSS algorithm.

4.1 Software and Hardware Specification

This research is implemented on KACT's Saudi supercomputer SANAM. KACST is King Abdel-Aziz City for Science and Technology in Riyadh, Kingdom of Saudi Arabia. SANAM includes Intel Xeon E5−2650 CPUs, with 12 cores. The access to SANAM is available through a group of interactive login nodes, which are connected to KACST network and Internet [14, 15]. The program is coded under Linux Ubuntu 16, with JAVA using the pj2 library for threads management [7], and NetBeans as programming tool. In addition, Simple Linux Utility for Resource Management (SLURM) is used for Linux clusters management in SANAM. SLURM performs three main tasks: First, it is responsible for nodes allocation, management, execution, and monitoring reserved nodes. Second, it manages waiting work queues and finally, resolves conflicting resource orders [16].
In this research, ten nodes are used. This is the maximum allowed by KACST to the external users.

4.2 Performance Measurements

The main objectives of this research are to minimize the execution time, increase resources utilization, and increase scalability. In addition, the final distance is to be minimized. Therefore, speedup, efficiency, scalability, and final distance are used to evaluate the performance of the parallel program [7].

4.2.1 Speed up:

Speed up is used to measure the extent of the time reduction gained from the parallel implementation as compared to its sequential counterpart. The gained speed up is calculated as the ratio of the execution time of the sequential program *Tseq* to that of the parallel program *Tpar* [7]:

$$\text{Speed up} = \frac{\text{Tseq}}{\text{Epar}} \qquad (4)$$

4.2.2 Efficiency

The efficiency (*E*) is used to measure how a program is close to the ideal speed up. In other words, it indicates the effectiveness of the parallel program to use the available resources. The ideal program efficiency is equal to 1. However, the actual efficiency is between 0 and 1. As the efficiency is closer to 1, as the program is making better use of the available hardware resources. Efficiency is computed as the ratio of the actual speed up of the parallel program *Tpar* to the number of processors *K* that are used to

run the parallel program. This is expressed in the following formula [7] :

$$\text{fficiency} = \frac{\text{Tpar}}{\text{K}} \qquad (5)$$

4.2.3 Scalability

Scalability is the ability of a program to adapt to the increasing amount of problem size. In order to measure the scalability of a parallel program, the sequential version is run multiple times; each time the problem size is increased. When the program crashes, and cannot hold any more the given problem size, the last recorded size is taken. This is *Nseq*. The same experiment is repeated with the parallel version to get *Npar*. The scalability is therefore calculated according to the following formula [7]:

$$\text{Scalability} = \frac{\text{Npar}}{\text{Nseq}} \qquad (6)$$

It is expected that the problem size increases with the increase of the number of processors.

4.3 Methodology

The execution time of a parallel program is hardly the same when run multiple times successively. This is because the operating system is conducting its own activities at the same time as the program runs. Since these activities differ from a run to another, the resulting execution time is directly affected. The interference of the operating system always increases the resulting execution time. Therefore, to measure the execution time of a program as accurate as possible, it is run multiple times – from 7 to 10 times – and the execution time is recorded after each run. Then, the minimum of these recordings is taken since this represents the less interference from the part of the operating system.

4.4 Experimental results

The results of both versions of the SA algorithm are reported in the following subsections. The parallel platform is an SMP; with a number of threads ranging from 2 to 8. A set of experiments is conducted according to the previously detailed methodology detailed with the aim of measuring the three main performance metrics: speedup, efficiency, and scalability in addition to the quality of the solution. These are exposed in the following subsections.

4.4.1 Speedup

The first set of experiments aims to explore the impact of the number of waypoints on the execution time. Therefore, the program is run multiple times with various number of waypoints; namely, 50, 100, 150, and 200. The experiment is done only for these four problems sizes as the increments in the execution time is linear.

Table 1 shows the minimum execution time for the sequential, SMP -as performed in [14]-, and CSS. Note that all the parameters of SA; namely, initial temperature, the cooling rate, and the stopping condition, are fixed. Worth to mention, the parameters are chosen after several experiments to ensure the quality of the final solution. Figure 8 shows the impact of the number of waypoints on the execution time.

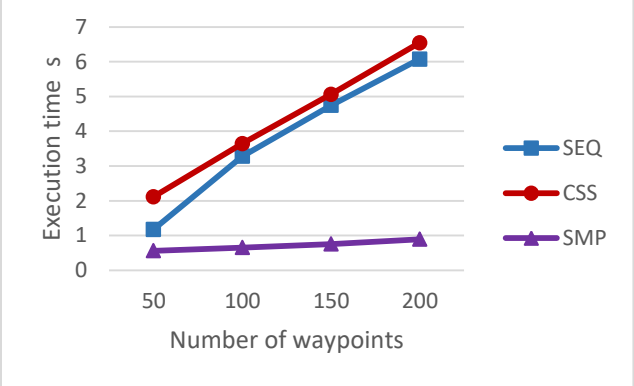


Figure 8. Relationship of Execution Time with Number of Waypoints

As seen in the graph depicted in Figure 8, the execution time of sequential and parallel SA versions is proportional to the problem size. It is noted that the execution time of the parallel cluster CSS is the highest when compared to the sequential and SMP. This is due to the fact that in CSS, each node works on the complete search space. In addition, several initial solutions are to be produced and propagated to all nodes at the beginning of the program. Therefore, the

communication between the nodes imposes an overhead on the execution time.
On the other hand, the execution time of the parallel SMP is the best. This is explained to the lack of communication overhead, since data is shared in the main memory.

The gained speedup is calculated from the values recorded in Table 1 according to formula 4 The results are depicted in Figure 9. The highest speedup is achieved by the SMP version; it is equal to 6.81for 200 waypoints as compared to a speedup of 0.57 for the CSS.

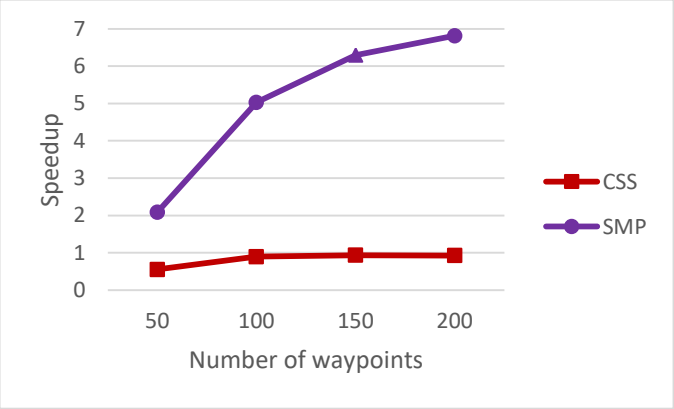


Figure 9. Relationship between Speedup and Number of Waypoints

4.4.2 Efficiency Measurement

The efficiency is calculated for both SMP and CSS versions according to formula 5 The results are displayed in Table 2. The corresponding graph is depicted in Figure 10. It is noticed that the efficiency is the best with parallel SMP where all threads in the nodes are utilized.

Although the efficiency is equal to 1 for the sequential program, but this does not imply that the program makes full use of the available hardware resources. However, the actual speed of the sequential program and the number of cores are both equal to one.

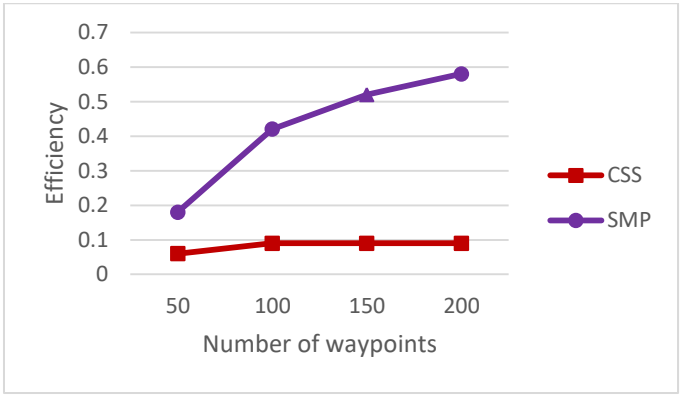


Figure 10. Relationship of Efficiency with Number of Waypoints for SMP and CSS

4.4.3 Scalability Measurement

Here, the largest number of waypoints that can be handled by each parallel SA version is divided by that handled by the sequential program. The results are reported in Table 3, with calculations deduced from formula 6 Again, the scalability in parallel SMP is too much better than that of the CSS. This is explained by the fact that all threads in the node are utilized in SMP. Thus, increasing the ability to handle larger problem sizes than sequential and parallel CSS. Moreover, the parallel CSS does not provide any improvement on the sequential SA.

4.4.4 The quality of the route plan

The final distance values are depicted in Table 1 with various SA algorithm versions and waypoints. The corresponding graph is shown in Figure 11. It is noticed that the quality of the route plan is the best with the SMP SA version as compared to the other program versions. However, the CSS SA produces very close distances as compared to the SMP SA. On the other hand, it gives better distance than the sequential. This is because the CSS SA uses more nodes working on different initial solutions; thus, increasing the chance of improving the distance.

5 CONCLUSION AND FUTURE WORK

This research targets for generating a minimum route plan distance for a single drone emitted by a military organism for image acquisition. The area in concern may be a sensitive site, a defense

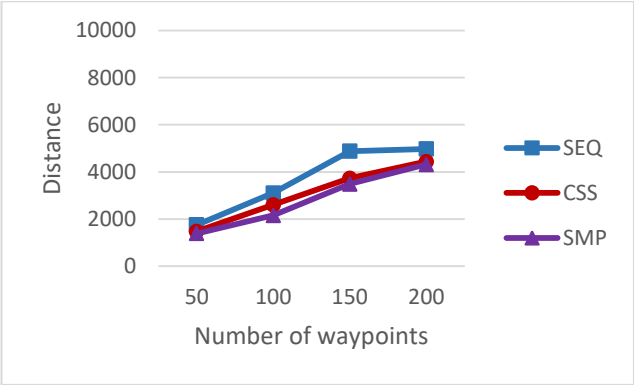


Figure 11. Quality of the final distance on parallel SA

and/or attack war front, or an enemy's territory. Therefore, the route path should be completed in the least amount of time.

The SA algorithm is implemented to solve this problem, which is analogous to the TSP. Since metaheuristics require an extensively long time for execution, parallel computing is deployed to accelerate the SA algorithm.

Therefore, several parallel versions of the SA are developed. First, the parallel SA is previously developed [14]. In this paper, another parallel version is implemented on SANAM supercomputer. Ten nodes are used to implement the program using Java programming language under Linux on SANAM supercomputer.

The two parallel versions are compared in terms of speedup, efficiency, scalability, and the final distance.

The reported results prove that the synchronized parallel SA on SMP outperforms the CSS for all number of waypoints in terms of gained speedup, efficiency, and scalability. On the other hand, the CSS outperforms the SMP SA in terms of quality of solution.

In the future, more methods to parallelize SA are to be investigated.

ACKNOWLEDMENT

We would like to extend our gratitude to King Abdel-Aziz City for Science and Technology (KACST) in Riyadh for allowing us to use their SANAM supercomputer at all times. Not only this, but the staff was also of great support and helpful; replying to our posed questions promptly albeit their heavy duty load.

REFERENCES

[1] N. Özalp and O. K. Sahingoz, "Optimal UAV path planning in a 3D threat environment by using parallel evolutionary algorithms," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference*, 2013, pp. 308-317.

[2] M. O. UgurCekmez, "A UAV Path planning with parallel ACO algorithm on CUDA platform," presented at the IEEE Unmanned Aircraft Systems (ICUAS), FL, USA, 2014.

[3] M. Coeckelbergh, "Drones, information technology, and distance: mapping the moral epistemology of remote fighting," *Ethics and information technology*, vol. 15, pp. 87-98, Jun 2013.

[4] X.-f. Liu, Z.-w. Guan, Y.-q. Song, and D.-s. Chen, "An optimization model of UAV route planning for road segment surveillance," *Journal of Central South University*, vol. 21, pp. 2501-2510, Jun 2014.

[5] T. Turker, G. Yilmaz, and O. K. Sahingoz, "GPU-Accelerated Flight Route Planning for Multi-UAV Systems Using Simulated Annealing," in *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, 2016, pp. 279-288.

[6] M. Sanjabi, A. Jahanian, S. Amanollahi, and N. Miralaei, "ParSA: parallel simulated annealing placement algorithm for multi-core systems," in *Computer Architecture and Digital Systems (CADS), 2012 16th CSI International Symposium*, 2012, pp. 19-24.

[7] A. Kaminsky, "BIG CPU, BIG DATA: Solving the World's Toughest Computational Problems with Parallel Computing," 2016.

[8] *KACST , The Saudi Supercomputer "SANAM" is the World's 2nd Leader in Energy Efficiency", KACST, 2012. [Online]. Available: https://www.kacst.edu.sa/eng/about/news/Pages/news3841117-3854.aspx. [Accessed: 25- Apr- 2017].*

[9] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 132-141, May. 2013.

[10] V. Roberge, M. Tarbouchi, and F. ALLAIRE, "Parallel hybrid metaheuristic on shared memory system for real-time UAV path planning," *International Journal of Computational Intelligence and Applications*, vol. 13, p. 1450008, Jun. 2014.

[11] E.-G. Talbi, *Metaheuristics: from design to implementation* vol. 74 ,pp 126-133: John Wiley & Sons, 2009.

[12] A. Ferreiro, J. García, J. G. López-Salas, and C. Vázquez.:An efficient implementation of parallel simulated annealing algorithm in GPUs, *Journal of Global Optimization*, vol. 57, pp. 863-890, Nov. 2013.

[13] S. Zaghloul and E. Alsafi, "Drone route planning for military image acquisition using parallel simulated annealing", International Journal of New Computer Architectures and their Applications (IJNCAA), vol. 7, no. 3, Sep, 2017.

[14] D. Rohr, S. Kalcher, M. Bach, A. A. Alaqeeliy, H. M. Alzaidy, D. Eschweiler, V. Lindenstruth, S. B. Alkhereyfy, A. Alharthiy, and A. Almubaraky, "An energy-efficient multi-GPU supercomputer," in *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on*, 2014, pp. 42-45.

[15] "Intel® ARK (Product Specs). (2017). Intel® Xeon® Processor E5-2650 v4 (30M Cache, 2.20 GHz) Product Specifications. [online] Available at: https://ark.intel.com/products/91767/Intel-Xeon-Processor-E5-2650-v4-30M-Cache-2_20-GHz [Accessed 13 Dec. 2017].".

[16] Slurm.schedmd.com. (2017). Slurm Workload Manager. [online] Available at: https://slurm.schedmd.com/quickstart.html [Accessed 14 Dec. 2017]."

[17] T. Rauber and G. Rünger, *Parallel programming: For multicore and cluster systems*: Springer Science & Business Media, 2013.

[18] A. Kaminsky, "Building Parallel Programs: SMPs, Clusters, and Java. Cengage Course Technology (2010)," ISBN 1-4239-0198-3.

Table 1 Relationship between the execution time and number of waypoints with the corresponding output distance

#waypoints	Sa algorithm version	Execution time (s)	Distance
50	Sequential	1.1711	1762
	Parallel cluster CSS	2.113	1483
	Parallel SMP	0.561	1389
100	Sequential	3.276	3110
	Parallel cluster CSS	3.644	2610
	Parallel SMP	0.651	2162
150	Sequential	4.741	4880
	Parallel cluster CSS	5.059	3733
	Parallel SMP	0.753	3487
200	Sequential	6.072	4976
	Parallel cluster CSS	6.544	4439
	Parallel SMP	0.891	4320

Table 2 Efficiency of sequential, SMP, and CSS versions

#waypoints	Sa algorithm version	Execution time (s)	Speed up	Efficiency
50	Sequential	1.1711	1	1
	Parallel cluster CSS	2.113	0.55	0.055
	Parallel SMP	0.561	2.09	0.17
100	Sequential	3.276	1	1
	Parallel cluster CSS	3.644	0.9	0.09
	Parallel SMP	0.651	5.03	0.42
150	Sequential	4.741	1	1
	Parallel cluster CSS	5.059	0.94	0.09
	Parallel SMP	0.753	6.3	0.52
200	Sequential	6.072	1	1
	Parallel cluster CSS	6.544	0.93	0.093
	Parallel SMP	0.891	6.81	0.57

Table 3 Measure scalability of parallel SA

SA algorithm version	Size up	Scalability
Sequential	3100	--
Parallel cluster CSS	3000	0.97
Parallel SMP	23000	7.4

International Journal of New Computer Architectures and their Applications (IJNCAA) 8(1): 33-40
The Society of Digital Information and Wireless Communications, 2018 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

A Finger-Mounted Haptic Device with Plane Interface

Makoto Yoda and Hiroki Imamura
Department of Information System Science, Graduate School of Engineering, Soka University
Mailing Address: 1-236, Tangi-machi, Hachioji-shi, Tokyo, Japan, 192-8577
E-mail: e16m5223@soka-u.jp, imamura@soka.ac.jp

ABSTRACT

Recently, several researches of haptic devices have been conducted. Haptic devices provide users with sense of touching virtual objects such as Computer Graphics (CG) by force feedback. Since they provide force feedback from a single point on an object surface where users touched, users touch it by point contact. However, they cannot provide a sense such as humans touching an object with a finger pad because humans do not touch by point contact but surface contact. We focused on this characteristic and developed surface contact haptic device. To touch a CG object in surface contact, we use a plane interface. It provides force feedback to the normal direction by being approximated to tangent plane on a CG object surface where users touched and the sense of grabbing it. In the evaluation experiments, twelve users evaluated this haptic device. From results, we see that users could feel sense of touching in surface contact and grabbing a CG object.

KEYWORDS

Haptic device, Surface contact, Plane interface, Force feedback, Augmented Reality

1 INTRODUCTION

In recent years, researches of human interface using Augmented Reality (AR) have been conducted [1]. AR is new technology which can overlay the information or computer generated virtual object such as Computer Graphics (CG) into real world [2]. With the advances in AR interface technology, haptic devices have been developed and attracting attentions of researchers. Haptic devices provide users with sense of touching a CG object by processing force feedback. Therefore, haptic devices are expected to be used in applications such as a virtual surgery system [3], a virtual experience system [4] or a remote control of robots [5].

Examples of conventional haptic devices include Falcon [6], PHANToM [7] and Dexmo [8]. Falcon and PHANToM are classified into a grounded type. This type can provide accurate force feedback because its fulcrum is fixed on the table. The operating range of a grounded type is limited. Dexmo is classified into a finger-mounted type. Users can feel sense of grabbing CG objects easily by using this type. In addition, the operating range of a finger-mounted type is not limited. These haptic devices have been developed as a point contact haptic device. A point contact type haptic device provides force feedback from a single point on a CG object surface where users touched. In case of perception of an object shape, humans perceive an object shape from force feedback to the normal direction on touching area [9]. However, in point contact, it is difficult to perceive an object shape because the direction of force feedback changes according to a finger direction as shown in Figure 1. To perceive CG object shape in point contact, users must trace the surface. To perceive force feedback to the normal direction, users must be provided force feedback from a tangent plane on touching area (surface contact) as shown in Figure 2.

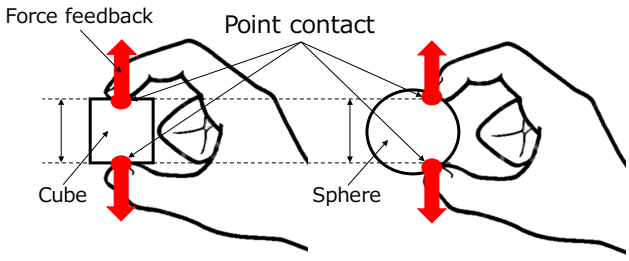


Figure 1. Point contact

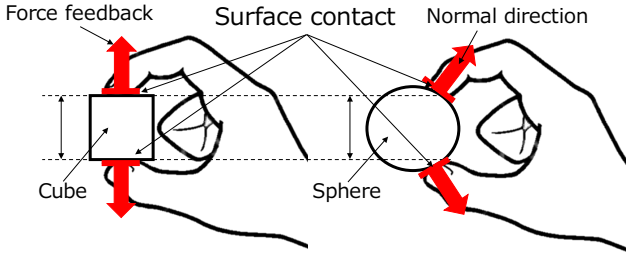


Figure 2. Surface contact

We focused on this characteristic and have developed a finger mounted type haptic device using surface contact [10] that is shown in Figure 3. This haptic device has a plane interface having four movable points. Figure 4 shows the outline of this haptic device. Four movable points of the plane interface operate up and down separately. In the initial state, the user is not touching a CG object. In the operating state, the user is touching a CG object. The plane interface provides a finger pad with force feedback to the normal direction by four movable points operating up and down and being approximated to tangent plane on the CG object surface where a finger pad touched. Thus, users can perceive the shape without tracing surface.

Figure 3. The developed finger-mounted haptic device

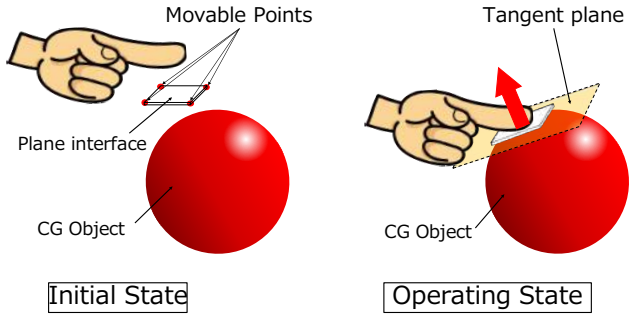


Figure 4. Outline of the developed haptic device

In this paper, we propose an improvement of this haptic device. This haptic device uses AR marker to detect a finger position and posture. However, if a part of the AR marker is hidden, a finger position and posture are not detected. In addition, there is the case that the plane interface cannot provide accurate slope. Therefore, we redesign controlling a plane interface mechanism to provide accurate plane interface slope and propose a novel finger-mounted haptic device using a method of detecting a finger position and posture without AR marker.

2 PROPOSED SYSTEM

2.1 Hardware Construction

Figure 5 shows the hardware construction of the proposed haptic device. This haptic device is glove type to grab a CG object and composed of Arduino Uno, two servo motors, acceleration sensor that are shown in Figure 6, 7 and 8, respectively, four springs and a plane interface for each finger. Figure 9 shows the appearance of the proposed haptic device. We developed it by using 3D printer.

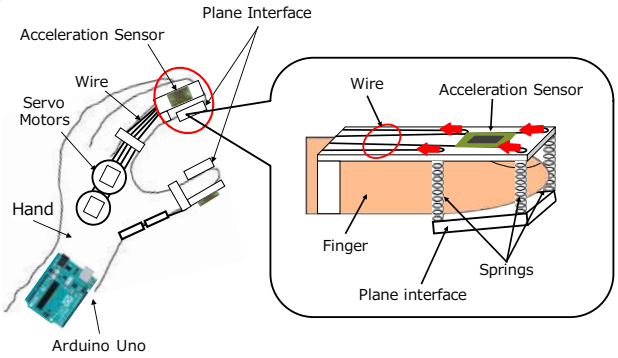


Figure 5. Hardware construction

Figure 6. Arduino

Figure 7. Servo motor

Figure 8. Acceleration sensor

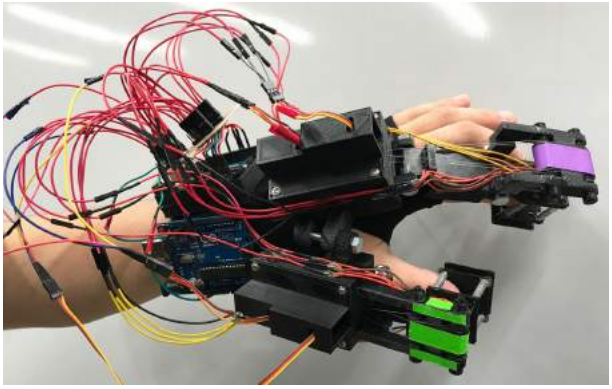


Figure 9. Appearance of the proposed haptic device

Two servo motors are mounted back of the hand as shown in Figure 10. We developed wire winding mechanism using pulleys to provide accurate a plane interface slope. A pulley is connected to two movable points of plane interface with wires. Arduino Uno is connected to each servo motor and controls them to operate each pulley. These pulleys pull up and down each movable point. A plane interface is controlled and provides a finger pad with a slope. Each spring adheres to each movable point as shown in Figure 11.

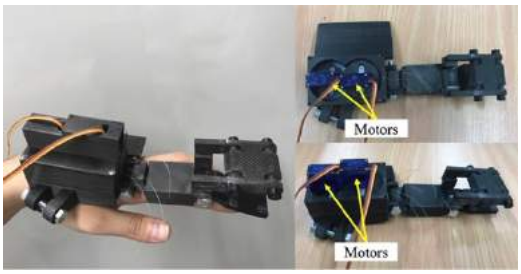


Figure 10. Back of the proposed haptic device

Figure 11. Fingertip part

2.2 System Overview

In this system, users touch a CG object in the display with the proposed haptic device and feel force feedback. Figure 12 shows system overview. We use PC, display, Creative Senz3D and the proposed haptic device. Users wear the proposed haptic device and touch a CG object. Figure 13 shows Creative Senz3D. Creative Senz3D is RGB-D camera whose range of depth sensor is 0.15~1[m]. We use it to draw CG objects and to detect 3-dementional finger position from it.

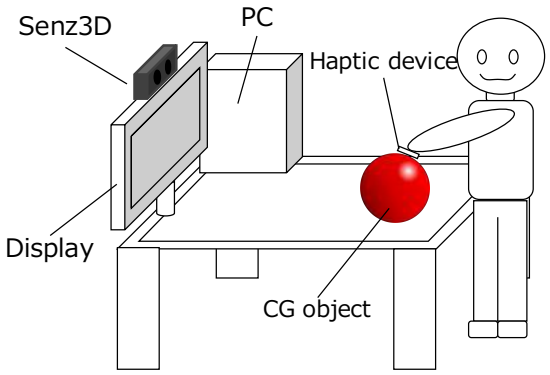


Figure 12. System overview

Figure 13. Creative Senz3D

2.3 System Flowchart

Figure 14 shows the system flowchart. This system draws a CG object on the desk and calculates a finger position. When a finger is close to a CG object, this system calculates a tangent plane slope angles on the surface which the finger is close to. By using tangent plane slope angles and finger slope angles, this system calculates motor rotation angles and send these angles to Arduino Uno by serial communication. Arduino Uno controls motors. A plane interface is controlled and provide a finger pad with force feedback. Users feel touching a CG object. The followings are the explanation of the processing in this system.

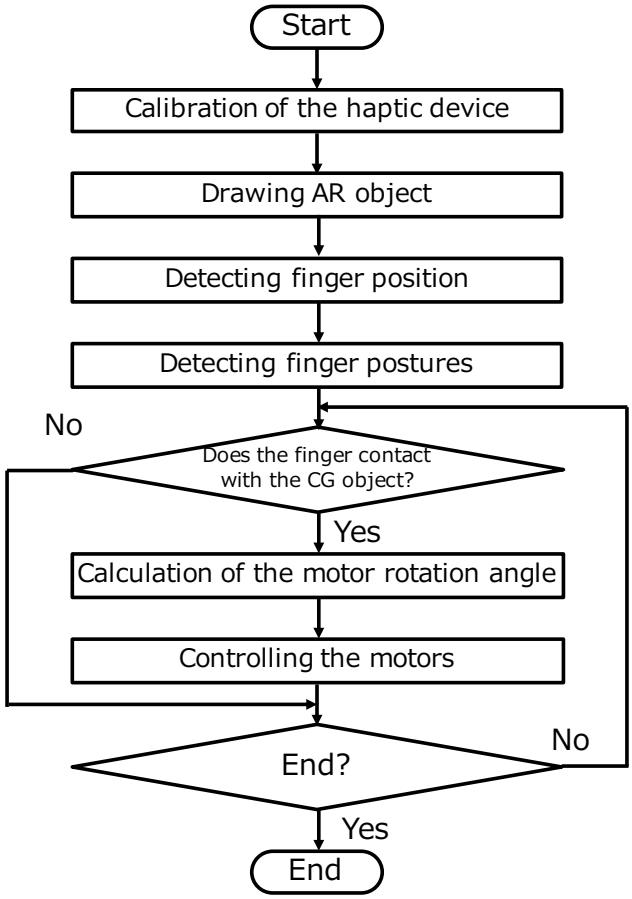


Figure 14. Flowchart

In the calibration of the haptic device, Arduino Uno controls motors to make a plane interface initial position. In this time, a finger pad is not touching a plane interface.

In the drawing AR object, this system draws a CG object on the desk. To draw a CG object, this system detects a plane by using depth information from Senz3D. Figure 15 is the image including a plane (desk). Users set a point as an origin on the plane using mouse cursor. This system sets other two points on the plane and calculates vectors $\boldsymbol{A}(A_x, A_y, A_z)$ and $\boldsymbol{B}(B_x, B_y, B_z)$ from a set point to other points. By using these vectors, this system calculates the normal vector $\boldsymbol{N}(N_x, N_y, N_z)$ from outer product

$$\boldsymbol{B}\times\boldsymbol{A} = \begin{pmatrix} B_yA_z - B_zA_y \\ B_zA_x - B_xA_z \\ B_xA_y - B_yA_x \end{pmatrix} = \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix}. \quad (1)$$

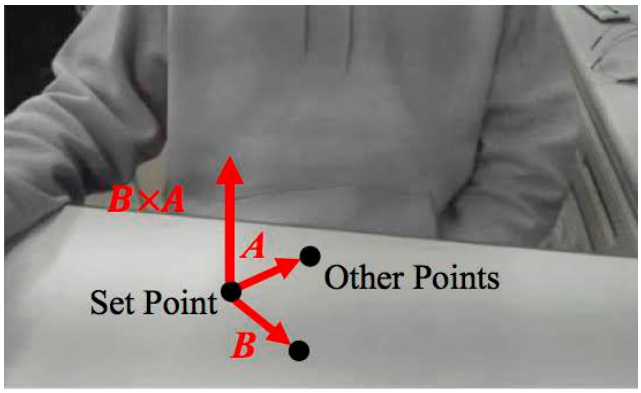


Figure 15. The image including a plane

Next, this system calculates a transformation matrix to convert the coordinates system whose origin is a set point to Senz3D coordinates system. By using the z-axis direction vector of Senz3D $\boldsymbol{S}(0, 0, 1)$ and the normal vector $\boldsymbol{N}$, this system calculates a rotation angle θ and a rotation axis $\boldsymbol{n}(n_x, n_y, n_z)$ from inner product and outer product

$$\theta = \cos^{-1}\left(\frac{\boldsymbol{S}\cdot\boldsymbol{N}}{|\boldsymbol{S}||\boldsymbol{N}|}\right), \quad (2)$$

$$\boldsymbol{n} = \boldsymbol{S}\times\boldsymbol{N} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}. \quad (3)$$

By using θ and $\boldsymbol{n}$, this system calculates quaternion $\boldsymbol{q}$ and a transformation matrix $\boldsymbol{R}$

$$\boldsymbol{q} = \begin{pmatrix} q_x \\ q_y \\ q_z \\ t \end{pmatrix} = \begin{pmatrix} \sin\left(\frac{\theta}{2}\right)n_x \\ \sin\left(\frac{\theta}{2}\right)n_y \\ \sin\left(\frac{\theta}{2}\right)n_z \\ \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \quad (4)$$

$$\boldsymbol{R} = \begin{pmatrix} 1-2{q_y}^2-2{q_z}^2 & 2q_xq_y-2tq_z & 2q_xq_z-2tq_y & p_x \\ 2q_xq_y-2tq_z & 1-2{q_x}^2-2{q_z}^2 & 2q_yq_z-2tq_x & p_y \\ 2q_xq_z-2tq_y & 2q_yq_z-2tq_x & 1-2{q_x}^2-2{q_y}^2 & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5)$$

where p_x, p_y and p_z are coordinates of a set point from Senz3D. This system draws CG objects (Sphere or Sin-Cos curve) on the coordinate system whose origin is a set point as shown in Fig. 16 and 17 by using this transformation matrix R and OpenGL.

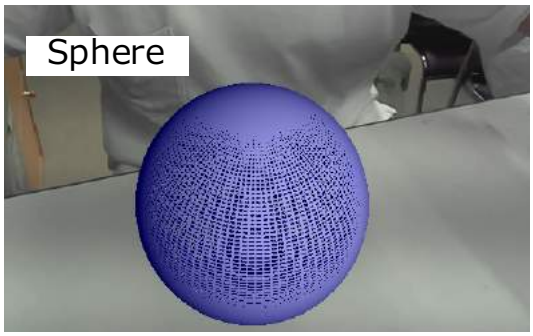


Figure 16. Sphere CG object

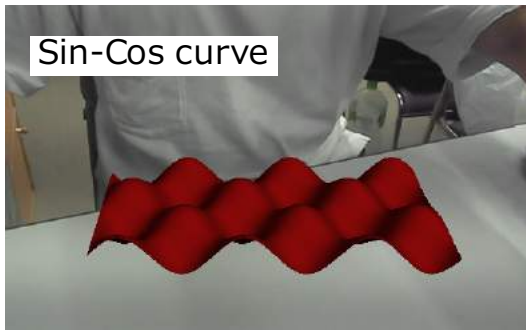


Figure 17. Sin-Cos curve CG object

In the detecting a finger position, this system uses a color information on the image. We make the fingertip a specific color as shown in Figure 18. This system extracts a specific color area and calculates three-dimensional coordinates (S_x, S_y, S_z) of the extraction region from Senz3D. By using this coordinates and $\boldsymbol{R}$, this system calculates a finger position $\boldsymbol{F}(F_x, F_y, F_z)$ from a set point

$$\boldsymbol{F} = \begin{pmatrix} F_x \\ F_y \\ F_z \\ 1 \end{pmatrix} = \boldsymbol{R}^{-1}\begin{pmatrix} S_x \\ S_y \\ S_z \\ 1 \end{pmatrix}. \quad (6)$$

Figure 18. Extraction color information

In the detecting a finger posture, this system calculates F_r that denotes the roll angle and F_p that denotes the pitch angle of each finger by using acceleration sensor. Figure 19 shows vectors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ that are gravity accelerations of X, Y and Z-axis respectively. From these vectors, this system calculates

$$F_r = \tan^{-1}\left(\frac{|\boldsymbol{A}|}{|\boldsymbol{B}|}\right), \tag{7}$$

$$F_p = \tan^{-1}\left(\frac{|\boldsymbol{C}|}{|\boldsymbol{B}|}\right), \tag{8}$$

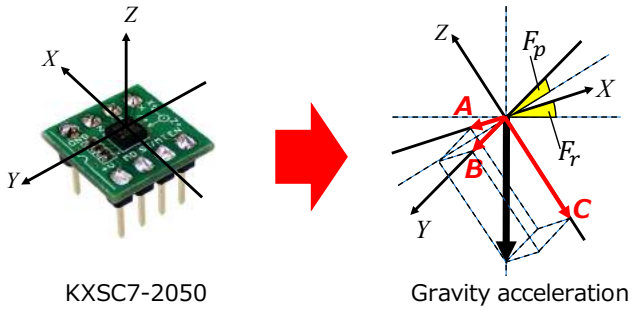


Figure 19. Detecting a finger position

In the judgement of contact, this system calculates z-coordinate on a CG object surface. The following is equation of Sphere and Sin-Cos curve.

$$x^2 + y^2 + z^2 = r^2, \tag{9}$$

$$A \sin x \cos y = z \tag{10}$$

where r is radius of Sphere CG object and A is an amplitude of Sin-Cos curve CG object. Using (9), (10), F_x and F_y, this system calculates z-coordinate on CG object surface. When F_z is under this z-coordinate on a CG object surface, judgement is contact and this system defines the finger position as a contact point.

In the calculation of the motor rotation angle, this system calculates the normal vector on the contact point in the same way as the process of drawing AR object. As shown in Figure 20, by using the contact point and other two points on a CG object surface, this system calculates the normal vector

$$\boldsymbol{N} = \boldsymbol{A} \times \boldsymbol{B} = \begin{pmatrix} N_x \\ N_y \\ N_z \end{pmatrix} \tag{11}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the vectors from the contact to other points

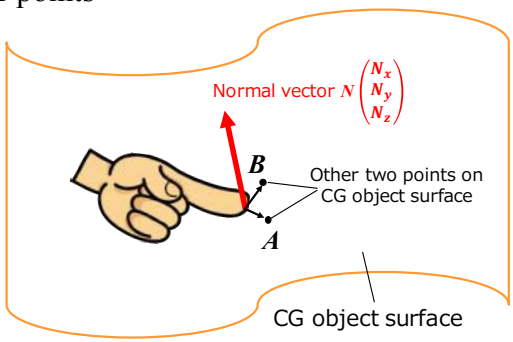


Figure 20. Calculation of the normal vector

From the normal vectors, to calculate the angles of tangent plane slope on the contact point, this system calculates

$$\theta_x = \cos^{-1}\left(\frac{|N_x|}{\sqrt{{N_x}^2 + {N_z}^2}}\right), \tag{12}$$

$$\theta_y = \cos^{-1}\left(\frac{|N_y|}{\sqrt{{N_y}^2 + {N_z}^2}}\right), \tag{13}$$

where θ_x is the angle between x-axis and the normal vector in X-Z plane, θ_y is the angle between y-axis and the normal vector in Y-Z plane that is shown in Figure 21.

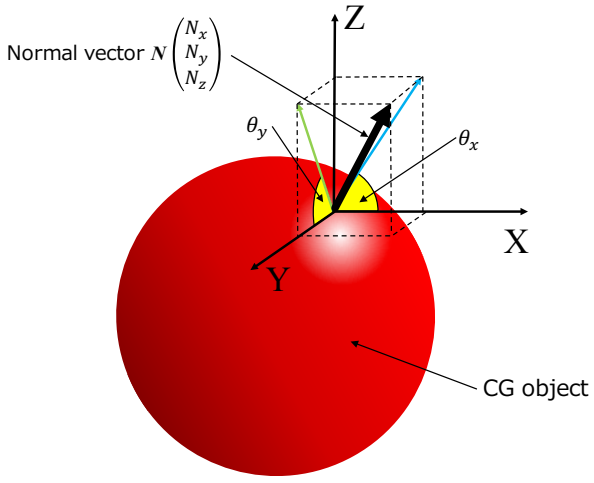


Figure 21. Calculation of θ_x and θ_y

Figure 22 shows the calculation of the angle of tangent plane slope on touching point in X-Z plane and Y-Z plane, respectively. Since the normal vector is vertical to the tangent plane, this system calculates the roll angle T_r and the pitch angle T_p of tangent plane slope

$$T_p = 90° - \theta_y, \tag{14}$$

$$T_r = 90° - \theta_x. \tag{15}$$

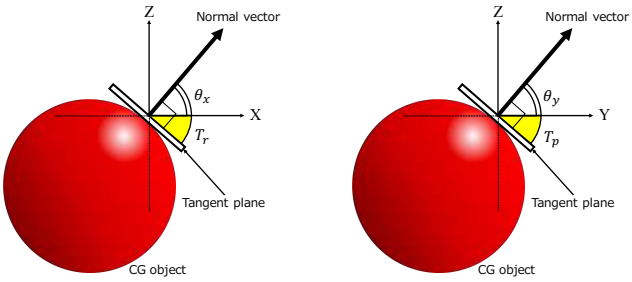


Figure 22. Calculation of the tangent plane slope angles

Using the angle of tangent plane slope and the angle of a finger, the system calculates the difference roll angle D_r and the pitch angle D_p between a finger and tangent plane

$$D_p = |T_p - F_p|, \tag{16}$$

$$D_r = |T_r - F_r|. \tag{17}$$

These difference angles are the roll and pitch angle of a plane interface slope as shown in Figure 23.

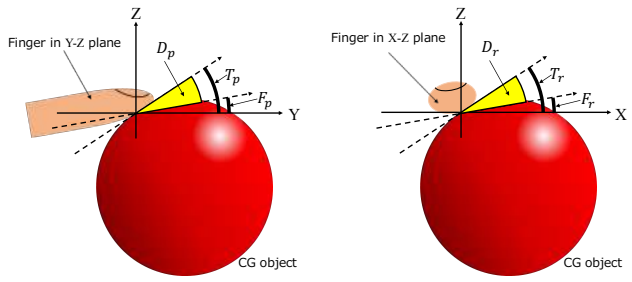


Figure 23. Calculation of plane interface slope angles

Figure 24 shows the state transition of a plane interface. A plane interface operates as shown in this figure. Using a plane interface slope angles, the system calculates the operation length *L1* and *L2* of each movable point

$$L1 = \frac{A\left(2\sin\left(\left|\frac{D_p}{2}\right|\right)+\sin(|D_r|)\right)}{2\sin\left(\tan^{-1}\left(\frac{2\sin\left(\left|\frac{D_p}{2}\right|\right)+\sin(|D_r|)}{1-\cos(|D_r|)}\right)\right)}, \tag{18}$$

$$L2 = \begin{cases} \dfrac{A\left(2\sin\left(\left|\frac{D_p}{2}\right|\right)-\sin(|D_r|)\right)}{2\sin\left(\tan^{-1}\left(\frac{2\sin\left(\left|\frac{D_p}{2}\right|\right)-\sin(|D_r|)}{1-\cos(|D_r|)}\right)\right)} & (|D_p| \geq |D_r|) \\ -\dfrac{A\left(2\sin\left(\left|\frac{D_p}{2}\right|\right)-\sin(|D_r|)\right)}{2\sin\left(\tan^{-1}\left(-\frac{2\sin\left(\left|\frac{D_p}{2}\right|\right)-\sin(|D_r|)}{1-\cos(|D_r|)}\right)\right)} & (|D_p| < |D_r|) \end{cases} \tag{19}$$

where A is the length of one side on a plane interface as operation length of each movable point. Figure 25 shows *L1* and *L2*.

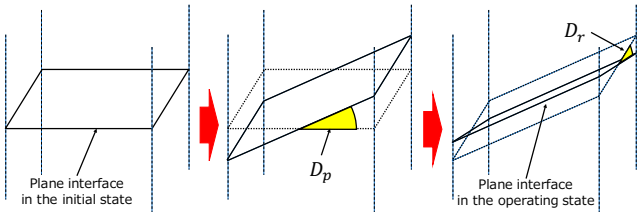


Figure 24. The state transition of a plane interface

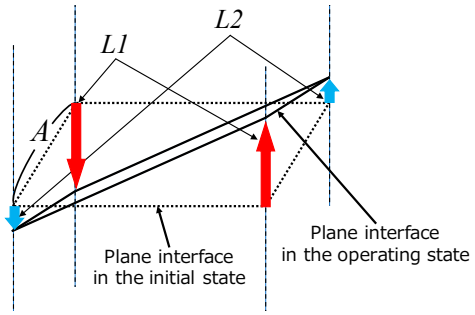


Figure 25. Operation length

Using the operation length of movable point *L1* and *L2*, the system calculates motor rotation angles

$$A1 = 2\sin^{-1}\left(\frac{L1}{2R}\right), \tag{20}$$

$$A2 = 2\sin^{-1}\left(\frac{L2}{2R}\right) \tag{21}$$

where R is the radius of a pulley. As shown in Figure 26, this system calculates the motor rotation angles so that the length of circular arc is equivalent to the operation length. Each pulley is connected to two movable points with wires. This system sends this motor rotation angles to Arduino Uno by Serial communication.

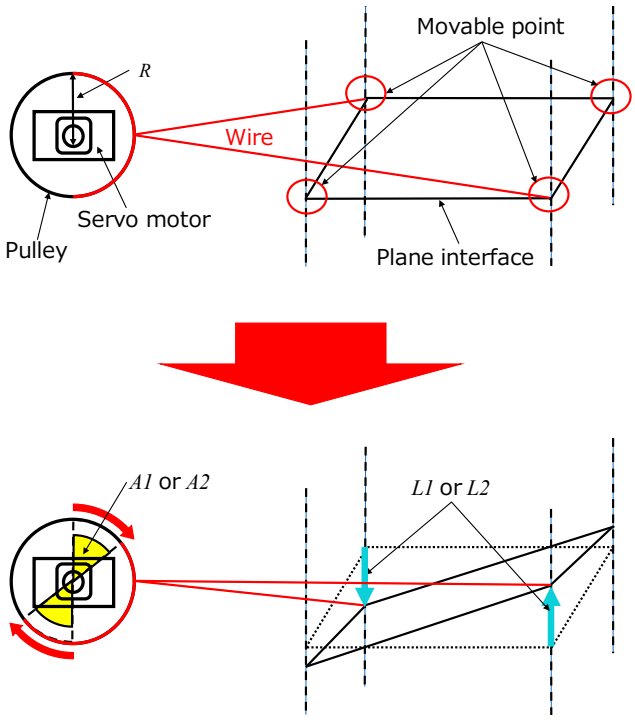


Figure 26. Calculation of the motor rotation angles

In the controlling the motors, Arduino Uno receives motor rotation angles and controls motors. Motors controls each pulley. These pulleys pull up each movable point with wires. A plane interface provides a finger pad with force feedback. Figure 27 shows that a plane interface is providing a finger pad with force feedback when users touched Sphere CG object in Y-Z plane and X-Z plane. Figure 28 shows that a plane interface is providing a finger pad with force feedback when users touched Sin-cos curve CG object in Y-Z plane and X-Z plane.

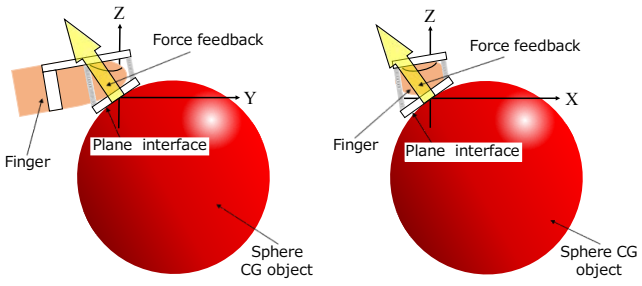


Figure 27. Touching Sphere CG object

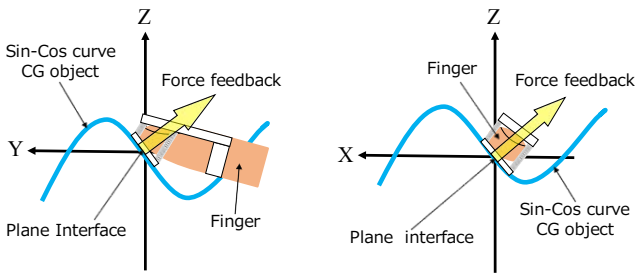


Figure 28. Touching Sin-Cos curve CG object

3 EVALUATION EXPERIMENTS

3.1 Overview of Experiments

We had evaluation experiments to compare previous haptic device and the proposed haptic device. Sphere CG object is used to evaluate whether users can perceive sense of touching and grabbing a CG object or not. Twelve people used previous haptic device and the proposed haptic device. After that, they evaluated following items with a 5-grade score.

Q1. You feel sense of touching a CG object.
Q2. You feel partial shape of a CG object surface.
Q3. You feel sense of touching a CG object with two fingers.
Q4. This system can detect a finger position naturally.

Evaluation values are from 1 to 5 (1: “Strongly Disagree”, 2: “Disagree”, 3: “Neutral”, 4: “Agree”, 5: “Strongly Agree”).

3.2 Results

Table 1 shows the results. This result shows average scores and standard deviations. All standard deviations are lower than 1.0. Thus, we consider that there is no variance of responding values. In addition, all average scores of the proposed haptic device are higher than that of the previous haptic device.

Table 1. Results

	Average score		Standard deviation	
No.	Previous	Proposed	Previous	Proposed
Q1	4.17	4.67	0.90	0.62
Q2	2.92	4.58	0.64	0.49
Q3	2.92	4.58	0.95	0.49
Q4	2.50	4.58	0.65	0.49

3.3 Discussions

From the results of Q1, we see that the both haptic devices can provide users sense of touching a CG object because the both average scores are higher than 4.0. In Q2, the average score of the previous haptic device is lower than 3.0 because there is the case that the plane interface cannot provide accurate slope. Thus, we see that the previous haptic device cannot provide sense of the partial shape of a CG object well. From the results of Q3 and Q4, we see that the previous haptic device cannot provide sense of touching a CG object with index finger and thumb well because two AR markers cannot be recognized at the same time. On the

other hand, from the results of Q1 to Q4, we see that the proposed haptic device can provide sense of touching a CG object and a partial shape of it more naturally than the previous haptic device because these average scores are higher than 4.5. From these results, we see that the plane interface can provide accurate slope by using redesigned wire winding mechanism. In addition, we see that the finger recognition ratio is greatly improved by using a color and a depth information.

4 CONCLUSION and FUTURE WORKS

The purpose of this paper is an improvement of a finger-mounted haptic device using surface contact. This haptic device has a plane interface having four movable points. The plane interface provides a finger pad with force feedback to the normal direction by being approximated to tangent plane on a CG object surface where finger pads touched. To draw AR object, we use depth information of Senz3D. To detect a finger position without AR marker, we use a color information and depth information of Senz3D. After the evaluation experiments, we see that the proposed haptic device can provide force feedback to the normal direction on the touching area of a CG object surface and grabbing a CG object. In addition, from the comparison results, we see that the performance of the proposed haptic device has been greatly improved than the previous haptic device. In the future, we want to develop a haptic device to deal with non-rigid object.

REFERENCES

1. Jinha Lee and Cati Boulanger, "Direct, Spatial, and Dexterous Interaction with See-through 3D Desktop", In Proceedings of ACM SIGGRAPH 2012 posters, Emerging Technologies (2012)
2. R. Silva, J. C. Oliveira and G. A. Giraldi, "Introduction to Augmented Reality", In National Laboratory for Scientific Communication, LNCC Research Report, 25 (2003)
3. Naoki SUZUKI, Asaki HATTORI, Takeshi EZUMI, Takahiro KU- MANO, Akio IKEMOTO, Yoshitaka ADACHI, and Akihiro TAKATSU, "Development of virtual surgery system with sense of touch", In Transaction of the Virtual Reality Society of Japan, Vol. 3, No. 4, pp. 237- 243 (1998)
4. Kevin Huang, Ellen Yi-Luen Do, and Thad Sterner, "Piano Touch: A Wearable Haptic Piano Instruction System for Passive Learning of Piano Skills", In Proceedings of 12th IEEE Symposium on Wearable Computers (2008)
5. I. Ivanisevic, and V. J. Lumelsky, "Configuration space as a means for augmenting human performance in teleoperation tasks", In IEEE Trans. On SMC, Part B, Vol. 30, No. 3, pp. 471-484 (2000)
6. Novint Technologies, Inc., "Haptic Device Abstraction Layer programmer's guide", In Version 1.1.9 Beta (2007)
7. J. K. Salisbury, and M. A. Srinivasan, "Phantom-Based Haptic Interaction with Virtual Objects", In IEEE Computer Graphics and Applications, Vol. 17, No. 5, pp. 6-10 (1997)
8. Xiaochi Gu, Yifei Zhang, Weize Sun, Yuanzhe Bian, Dao Zhou and Per Ola Kristensson, "Dexmo: An Inexpensive and Lightweight Mechanical Exoskeleton for Motion Capture and Force Feedback in VR", In Proceedings of 2016 CHI Conference on Human Factors in Computing Systems (2016)
9. Lederman S. J, "Tactile roughness of grooved surfaces: The touching process and effects of macro- and micro surface structure", In Perception & Psychophysics, Vol. 16, No. 2, pp. 385-395 (1974)
10. Makoto Yoda and Hiroki Imamura, "Development of A Finger Mounted Type Haptic Device Using A Plane Approximated to Tangent Plane", In Proceedings of 9th International Conference on Advances in Computer-Human Interactions (2016)

International Journal of New Computer Architectures and their Applications (IJNCAA) 8(1): 41-52
The Society of Digital Information and Wireless Communications, 2018 ISSN 2220-9085 (Online); ISSN 2412-3587 (Print)

A 3-Dimensional Object Recognition Method Using Relationships between Feature Points, and Invariance of Local Hue Histogram

Tomohiro Kanda, Kazuo Ikeshiro and Hiroki Imamura
Department of Information Systems Science, Graduate School of Engineering, Soka University
1-236, Tangi-machi, Hachiouji-shi, Tokyo, Japan 192-8577
e16m5206@soka-u.jp, ikeshiro@soka.ac.jp, imamura@soka.ac.jp

ABSTRACT

In recent years, there is concern about the lack of labor power such as household chores and nursing care at home due to advancement of the declining birthrate and an aging population. Accordingly, the Life Support Robots aiming at livelihood support of people are attracting attention. In the Life Support Robot, the process of identifying objects is one of the most important tasks, and a lot of studies have been performed in order to solve it. However, there are still many subjects that must be solved for practical use. Therefore, this paper presents a novel technology focusing on the recognition process for Life Support Robot. This technology describes useful features in 3D object by using relationships between feature points, and invariance of local hue histogram, furthermore this technology performs matching by them. Thereby, this technology is demonstrated to improve the recognition accuracy in challenging object recognition scenarios such as occlusion.

KEYWORDS

Life Support Robot, Cognitive system, 3D object, List matching, 2-dimensional projection, Hue histogram

1 INTRODUCTION

1.1 Background of This Study

In recent years, the declining birthrate and an aging population are developing, and there is concern about the lack of labor power such as household chores and nursing care at home. Accordingly, application of robot technology to the living field is expected. In the living field, robots that support the lives of people are collectively referred to as Life Support Robots [1][2][3]. This robot is required to perform various tasks to support the human. Especially, objects recognition task is important when people request the robots to transport and rearrange objects. We consider that there are six necessary properties to recognize in domestic environment as follows.

1. Robustness against occlusion
2. Fast recognition
3. Pose estimation with high accuracy
4. Coping with erroneous correspondences
5. Recognizing objects in a noisy environment
6. Recognizing objects which have same shape but have different texture

Firstly, the robots need the robust recognition for occlusion because occlusion frequently occurs between different objects in domestic environment. Secondly, the robots need to recognize a target object fast to achieve required tasks fast. Thirdly, the robots need to estimate a pose of a target object with high accuracy to manipulate a target object. Fourthly, the robots need to cope with erroneous correspondence to accurately recognize objects which have the same feature in a local region but which are not same. For example, a cube and a rectangular parallelepiped they both have same feature points in their vertex, however aspect ratio is totally different. Fifthly, a target object contains some noises with high probability when recognizing from cameras and sensors, so the robots need the robust recognition for noise. Finally, the robots need to accurately recognize objects which have same shape but which have different texture.

As conventional object recognition method using 3-dimensional information, there is model-based recognition method such as the previous research by Kudo et al [4]. The previous research uses SHOT (Signature of Histogram of Orientations) descriptor as feature descriptor [5]. SHOT descriptor is expressed by a histogram with 352 dimensions which is described by the relationships between the reference point and surrounding points. As above, SHOT enables highly accurate pose estimation, and highly

accurate object recognition in noisy environment by high dimensional feature description. Furthermore, the previous research uses some matched points by SHOT descriptor as feature points. Then the previous research generates a list by listing relationships of distances and angles between feature points, and matches lists. Thereby, the previous research can cope with erroneous correspondences.

However, the previous research erroneously recognizes objects which have same shape but which have different textures as shown in Figure 1 because the previous research uses only shape information of an object.

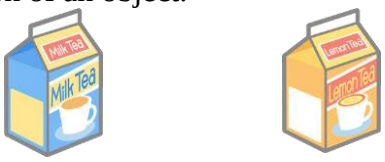
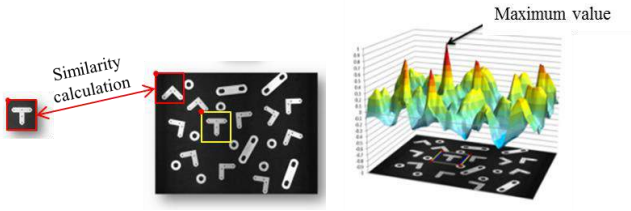


Figure 1. An example of objects which have same shape but which have different texture.

Table 1 shows properties of the previous research. As we mentioned, the previous research does not satisfy all the properties.

To satisfy all the properties for recognition, it is necessary to use not only shape information of an object but also texture information of an object.

As general object recognition method using texture information, Template Matching is widely known [6]. Template Matching calculates whether a pattern similar to the template exists in the image region by comparing pixels as shown in Figure 2.

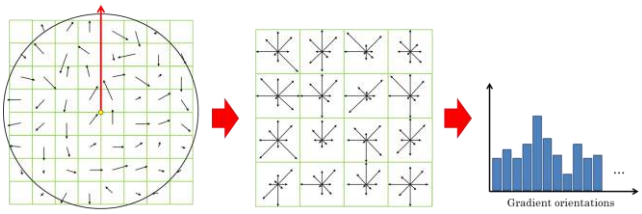


(a) Template image and input image (b) Mat of similarity

Figure 2. An example of matching by using Template Matching.

Therefore, Template Matching may misrecognize objects when changes such as the scale change and the rotation change are applied to a target object. Since recognition environment is unspecified when recognizing an object in domestic environment, changes are applied to a target object with high probability. Therefore, we think that a recognition method which has robustness against those changes is indispensable in this research.

As object recognition method which has robustness against above changes, SIFT (Scale Invariant Feature Transform) is widely known [7]. SIFT uses the points which have extreme values in DoG image as feature points. Furthermore, SIFT descriptor is expressed by a gradient histogram based on the direction of the feature point as shown in Figure 3.



(a) Image gradients (b) Kyepoint descriptor (c) Histogram (128 bins)

Figure 3. SIFT feature description.

Thereby, SIFT can correctly recognize objects even when the scale change, the rotation change and the illumination change occurs. However, SIFT descriptor is easily effected because perspective projection adds distortion to an image. Furthermore, objects which have few textures have hardly a local luminance gradient, therefore SIFT is difficult to describe features.

On the other hand, there is Color Indexing as a robust method for perspective projection [8]. Color Indexing uses 3-dimensional color histogram based on the RGB values in an image as feature descriptor. Figure 4 shows an example of 3-dimensional color histogram. As shown in Figure 4, a size of the square in 3-dimensional color histogram expresses the frequency of each color.

Table 1. Properties of the previous research and the proposed method.

	Robustness against occlusion	Fast recognition	Pose estimation with high accuracy	Coping with erroneous correspondences	Recognizing objects in a noisy environment	Recognizing objects which have same shape but have different texture
Previous Research	○	○	○	○	○	×
Proposed Method	○	○	○	○	○	○

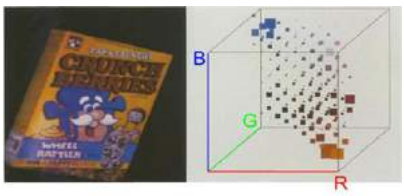


Figure 4. An example of 3-D color histogram.

The values of 3-dimensional color histogram has a characteristic which is hardly effected from the scale change, the rotation change and perspective projection. Therefore, Color Indexing can correctly recognize objects even when the scale change, the rotation change and perspective projection occur. However, the RGB color system which is used for Color Indexing is easily affected by lighting.

Table 2 shows properties of these methods. As we mentioned, two of the method do not satisfy all the properties.

Therefore, to compensate for the defect of SIFT and Color Indexing, we have developed the previous research using texture information for the object recognition [9]. The previous research using texture information focuses on the invariance of the positions of the unevenness of the hue histogram. Figure 5 shows the invariance of the positions of the unevenness of the hue histogram for each change.

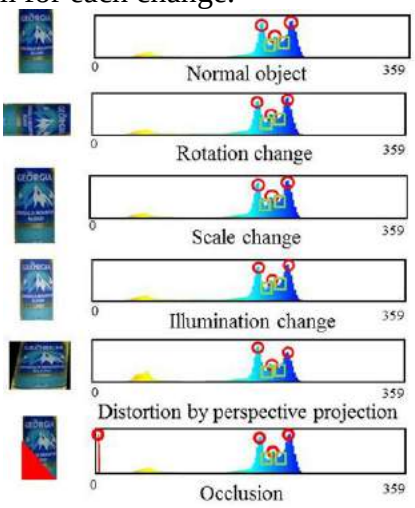


Figure 5. The invariance of the positions of the unevenness of the hue histogram for each change.

As shown in Figure 5, the positions of the unevenness of the hue histogram have a characteristic, which they do not change even when the scale change, the rotation change, the illumination change and the perspective projection occur. Furthermore, even when the occlusion occurs, the unevenness is seen in other places, but the positions of the original unevenness do not change. For these reasons, the previous research using texture information uses the positions of the unevenness of the hue histogram as feature descriptor. In addition, the previous research using texture information divides an image into plural regions and generates a hue histogram for each divided region. Thereby, even if objects have similar hue values, the previous research using texture information can accurately recognize ones. From above, we adopt the previous research using texture information as an object recognition method using texture information.

1.2 Purpose of This Study

To satisfy the six properties for recognition as shown in Table 1, we propose a 3-dimensional object recognition method using relationships between feature points, and invariance of local hue histogram. As our approaches, firstly, the proposed method extracts correspondence points by matching lists which consist of relationships of distances and angles between feature points. Secondly, the proposed method estimates a pose of a target object with high accuracy and performs registration between objects. Thirdly, the proposed method projects objects after registration to 2-dimensional plane. Fourthly, the proposed method divides the 2-dimensional planes into plural regions and generates a hue histogram for each divided region. Finally, the proposed method extracts the position of unevenness from the generated hue histograms as invariant feature, and matches based on

Table 2. Properties of conventional methods and the previous research using textures.

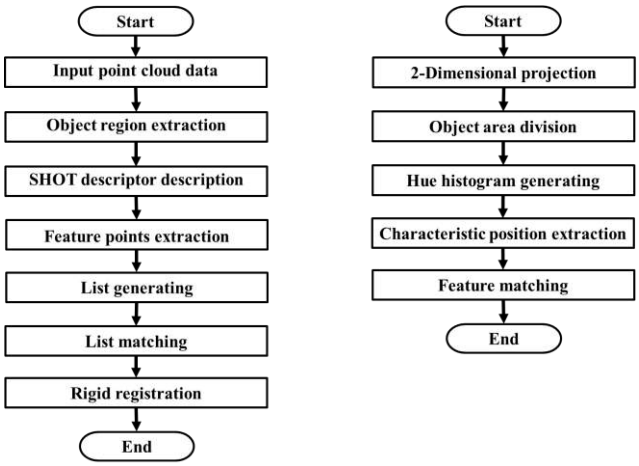
	Rotation change	Scale change	Illumination change	Distortion by perspective projection	Occlusion	An object with few textures
SIFT	○	○	○	×	○	×
Color Indexing	○	○	×	○	○	○
Previous research	○	○	○	○	○	○

extracted invariant features. Thereby, the proposed method can accurately recognize objects which have same shape but which have different textures.

2 PROPOSED METHOD

2.1 Flow of the Proposed Method

In this section, we describe about an overview of the proposed method based on its processing flow. Figure 6 shows the flow of the proposed method.



(a) Candidate regions extraction process (b) Recognition process

Figure 6. Flow of the proposed method.

As shown in Figure 6, the proposed method consists of candidate regions extraction process and recognition process. We represent each process in the next sections.

2.2 Input Point Cloud Data

Firstly, the proposed method inputs a teaching data and a scene data as shown in Figure 7. Here, the teaching data has shape information and texture information for the entire circumference of the object.

(a) Teaching data (b) Scene data

Figure 7. Overview of the input data.

2.3 Object Region Extraction

To extract useful information from a huge amount of 3-dimensional data, the proposed method segments object regions in scene data. In this paper, we assume that the target object is on a table or a floor in domestic environment as shown in Figure 7 (b). Therefore, the proposed method firstly detects the plane region by applying a plane detection method using RANSAC [10] and excludes it. Secondly, the proposed method uses the clustering method to cluster each object as shown in Figure 8.

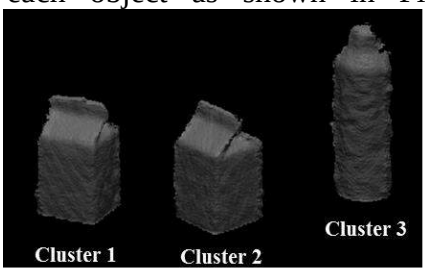
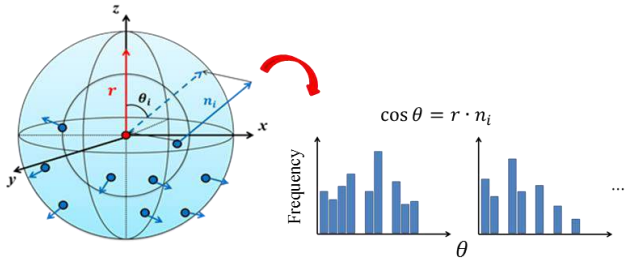


Figure 8. The result of deleting a plane and classifying each object.

2.4 SHOT Descriptor Description

To extract feature points, the proposed method uses SHOT (Signature of Histogram of Orientations) descriptor as feature descriptor. SHOT descriptor is expressed by a histogram with 352 dimensions which is described by the relationships between the reference point and surrounding points. Therefore, the surface features of the three-dimensional model can be described with unique and repeatability by using SHOT descriptor. In this section, we explain about how to describe the SHOT descriptor according to Figure 9.



(a) Signature structure for SHOT (b) Histogram (352 bins)

Figure 9. SHOT feature description.

As shown in Figure 9, SHOT descriptor is defined by the normal direction histogram of the peripheral point group. Firstly, surround the reference point with a sphere. This sphere is the

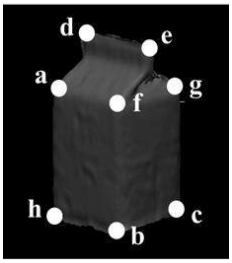
range of points used for description. Furthermore, this sphere is divided into 32 rooms by dividing it into 2 rooms in the z axis direction, 2 rooms spherically on the center and outside, and 8 rooms for xy plane. Finally, in each room, the inner product of the normal n_i of the point existing in the room and the norm r of the reference point is calculated. If the normal is normalized, the inner product can be expressed by $\cos\theta$ of r and n_i. Since $\cos\theta$ takes value from 0 to 1 ($-90 \leq \theta \leq 90$), it is divided into bin number and converted into a histogram.

2.5 Feature Points Extraction

The SHOT descriptor is represented by a vector of high dimensions. KNN search is used to match this feature. If the ratio of the distance to the first node and the distance to the second node obtained by KNN search is equal to or greater than a certain value, we consider that it is available for discrimination as a feature quantity and save the matching. Conversely, if there is not much difference between the distances of the first node and the second node, it is considered to be unstable to use for matching, and it is excluded. Furthermore, the most desirable matching point is searched out from the saved point group. In this research, the matched points are registered as feature points.

2.6 List Generating

In the list generating process, the proposed method generates the list of distances and angles between extracted feature points as relationships of these points. To generate the list of relationships, the proposed method firstly sorts the extracted feature points in descending order of the dispersion of SHOT descriptor as shown in Figure 10.



Feature point	Dispersion
a	$\sigma^2{}_a$
b	$\sigma^2{}_b$
c	$\sigma^2{}_c$
d	$\sigma^2{}_d$
e	$\sigma^2{}_e$
f	$\sigma^2{}_f$
g	$\sigma^2{}_g$
h	$\sigma^2{}_h$

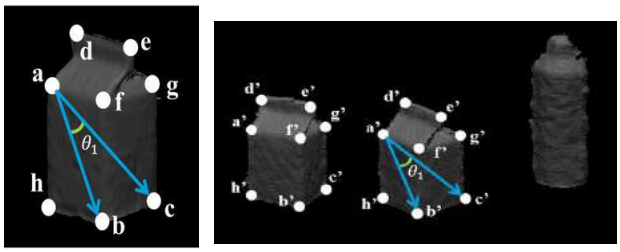
Figure 10. An example of sorted feature points in descending order on dispersion of SHOT.

The dispersion is calculated by the following equation.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}) \quad (1)$$

Where, σ^2 is the dispersion. n is the number of data. x_i is each data. $\bar{x}$ is the average value.

Secondly, the proposed method extracts the combination of three points based on the order of the aligned feature points as much as possible, and describes the relationships of them in a list as shown in Figure 11, Table 3 and Table 4.



(a) Teaching data (b) Scene data

Figure 11. An example of combination of three points.

Table 3. The list of the teaching data.

number	Corresponding point ①	Corresponding point ②	Corresponding point ③	Distance between point ① and ②	Distance between point ③ and ③	angle
1	a	b	c	l_{ab}	l_{ac}	θ_1
2	a	b	d	l_{ab}	l_{ad}	θ_2
3	a	b	e	l_{ab}	l_{ae}	θ_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
336	h	g	f	l_{hg}	l_{hf}	θ_{336}

Table 4. The list of each cluster data.

number	Corresponding point ①	Corresponding point ②	Corresponding point ③	Distance between point ① and ②	Distance between point ③ and ③	angle
1	a'	b'	c'	$l_{a'b'}$	$l_{a'c'}$	θ'_1
2	a'	b'	d'	$l_{a'b'}$	$l_{a'd'}$	θ'_2
3	a'	b'	e'	$l_{a'b'}$	$l_{a'e'}$	θ'_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
336	h '	g'	f'	$l_{h'g'}$	$l_{h'f'}$	θ'_{336}

2.7 List Matching

In the list matching process, the proposed method matches the list of the teaching data and the list of each cluster data. As shown in Figure 11, Table 3 and Table 4, the lists have distances between the feature points, and an angle as element. Then, the proposed method matches between list number 1 of the teaching data and all the lists of each cluster data. Furthermore, in the proposed method, if the sum of the difference of distances between corresponding point ① and corresponding point ②, and the difference of

distances between corresponding point ① and corresponding point ③, and the difference of the angle between the those vectors is minimum and less than the threshold, a list having it is registered as corresponding list. At this time, the feature points of each element of these lists are associated. Thereby, the proposed method can eliminate mismatched points which are occurred while the matching is conducted by SHOT descriptor.

2.8 Rigid Registration

To estimate the pose of the target object in the scene data, the proposed method applies the rigid registration to the teaching data as shown in Figure 12. Firstly, the proposed method fits the teaching data to each cluster data in the scene by calculating the optimum rotation matrix R and the translation vector t from associated feature points. Secondly, the proposed method calculates a corresponding rate M between a fitted teaching data and each cluster data by using

$$Score = \sum_{i=1}^{N} f\left(\min\{dist_{ij} | 1 \le j \le L\}\right),$$

$$f(x) = \begin{cases} 1 & (x \le th_c) \\ 0 & (x > th_c) \end{cases}, \quad (2)$$

$$dist_{ij} = \|p_i - q_j\|,$$

$$M = \frac{Score}{L} \cdot 100. \quad (3)$$

Where, N is the number of points of the teaching data. L is the number of points of each cluster data. p_i is matched point of the fitted teaching data. q_j is matched point of each cluster data in the scene. The proposed method counts a number of p_i which are within a threshold th_c which is 1 [mm] of q_j by the equation (2) as a score. And then, the proposed method calculates the corresponding rate M based on the score by equation (3). Finally, the proposed method selects objects which have the corresponding rate higher than the threshold value in scene data as candidate regions as shown in Figure 13.

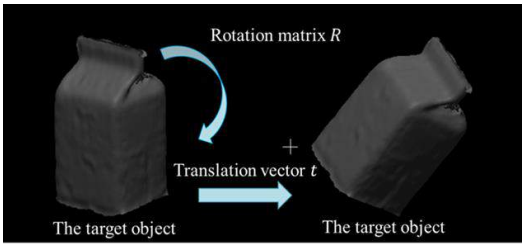


Figure 12. Illustration of the rigid registration.

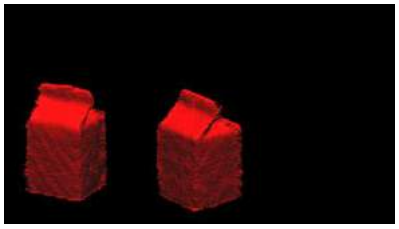


Figure 13. The result of candidate regions extraction.

2.9 2-Dimensional Projection

The proposed method projects the fitted teaching data and each candidate region onto a 2-dimensional plane respectively to extract texture information. It is not necessary to take into consideration problems related to rotation change and scale change since the positions of the teaching data and each candidate region are matched in rigid registration processing. Therefore, we use parallel projection as a 2-dimensional projection method. In Parallel Projection, a 3-dimensional point cloud of an object is projected to 2-dimensional plane by using

$$x' = x$$
$$y' = y \quad (4)$$
$$z' = z = const$$

As shown in Figure 14. Where, (x, y, z) is point before conversion. (x', y', z') is after conversion.

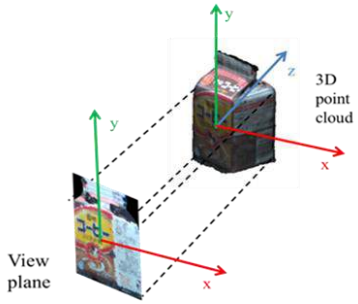


Figure 14. Parallel Projection.

Figure 15 shows the result of projecting the teaching data and a part of the candidate region to 2-dimensional plane.

(a) Plane of teaching data. (b) Plane of candidate region.

Figure 15. The result of 2D projection process.

2.10 Object Region Division

To generate the local hue histogram, the proposed method divides the planes of the teaching data and each candidate regions into plural regions. Here, we define the number of division as four as example.

2.11 Hue Histogram Generation

To generate the hue histogram, the proposed method extracts hue from each divided regions of the teaching data and each candidate regions. Then, the proposed method generates local hue histograms. The hue value of the generated histogram is represented from 0 to 359. Figure 16 shows the divided regions and local hue histograms. We define the vertical axis as the frequency, and the horizontal axis as the hue value.

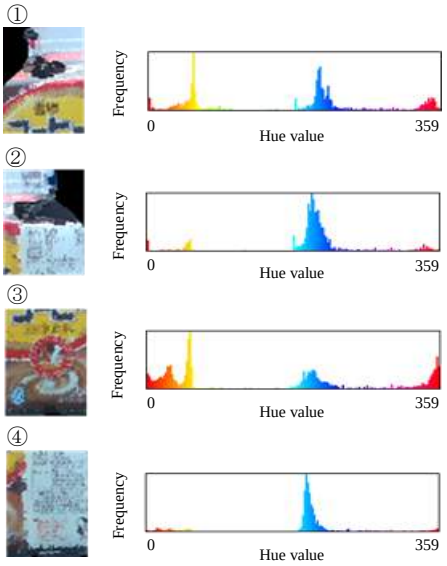
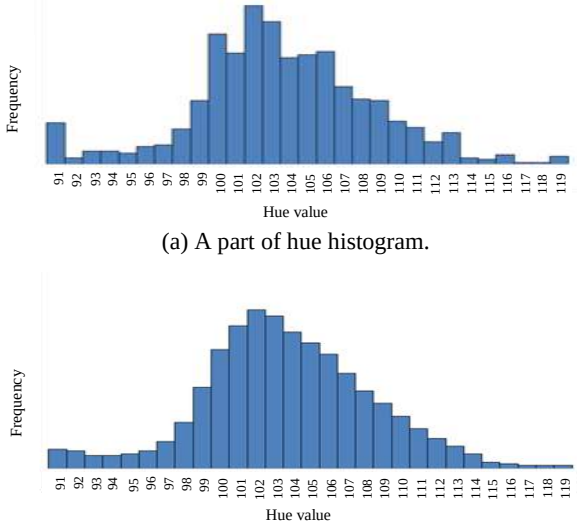


Figure 16. An example of divided regions and local hue histograms.

Here, we focus item ② of Figure 16. Figure 17 (a) shows an expanded hue histogram of item ② of Figure 16. In Figure 17 (a), because there are small irregularities at 100 and 102 of hue value, the feature descriptor becomes unstable by extracting the positions of peak and trough of the hue histogram in this state. Therefore, to eliminate those small irregularities, the proposed method smooths the hue histogram by using Gaussian Filter. Figure 17 (b) shows a smoothed hue histogram of Figure 17 (a). In Figure 17 (b), we can see small irregularities of the hue histogram are omitted, and the characteristic positions of peak and trough of the hue histogram are remained.



(a) A part of hue histogram.

(b) A part of smoothed hue histogram.

Figure 17. Smoothing processing.

2.12 Characteristic Position Extraction

In characteristic position extracting process, the proposed method registers the positions of peak and trough of the hue histogram as feature descriptor. The proposed method extracts the positions of peak from the smoothed hue histograms of teaching data and each candidate regions by using

$$(H_{x-1} < H_x) \wedge (H_x > H_{x+1}), \quad (5)$$

and the positions of trough from the smoothed hue histograms of teaching data and each candidate regions by using

$$(H_{x-1} > H_x) \wedge (H_x < H_{x+1}), \quad (6)$$

and registers them as feature descriptor. Where, H_x is the hue value which is focused on. H_{x-1} is the hue value before one of H_x. H_{x+1} is the hue value after one of H_x. And then, the extracted positions of peak and trough of teaching data are expressed by using

$$\left\{ {p_1}_{(a)}{}^{(\alpha)}, {p_2}_{(a)}{}^{(\alpha)}, \cdots \right\} \in {P_{(a)}}^{(\alpha)}, \quad (7)$$

$$\left\{ {t_1}_{(a)}{}^{(\alpha)}, {t_2}_{(a)}{}^{(\alpha)}, \cdots \right\} \in {T_{(a)}}^{(\alpha)}, \quad (8)$$

Where, ${P_{(a)}}^{(\alpha)}$ is a set of peak position of a smoothed hue histogram in divided region α of a. ${p_1}_{(a)}{}^{(\alpha)}, {p_2}_{(a)}{}^{(\alpha)}, \cdots$ which are each peak position of a smoothed hue histogram in divided region α of a. ${T_{(a)}}^{(\alpha)}$ is a set of trough position of a smoothed hue histogram in a divided area α of a. ${t_1}_{(a)}{}^{(\alpha)}, {t_2}_{(a)}{}^{(\alpha)}, \cdots$ which are each trough position of a smoothed hue histogram in divided

region α of a. In addition, the extracted positions of peak and trough of candidate regions are expressed by using

$$\{hp_1{}^{(\beta)}, hp_2{}^{(\beta)}, \cdots\} \in hP^{(\beta)}, \quad (9)$$

$$\{ht_1{}^{(\beta)}, ht_2{}^{(\beta)}, \cdots\} \in hT^{(\beta)}, \quad (10)$$

Where, $hP^{(\beta)}$ is a set of peak position of a smoothed hue histogram in a divided region β. $hp_1{}^{(\beta)}, hp_2{}^{(\beta)}, \cdots$ which are each peak position of a smoothed hue histogram in a divided region β. $hT^{(\beta)}$ is a set of trough position of a smoothed hue histogram in a divided region β. $ht_1{}^{(\beta)}, ht_2{}^{(\beta)}, \cdots$ which are each trough position of a smoothed hue histogram in a divided region β.

2.13 Feature Matching

To recognize teaching data from candidate regions, the proposed method performs matching by the positions of peak and trough of the local hue histograms between teaching data and each candidate region. Firstly, the proposed method calculates the difference values between feature descriptor of teaching data and feature descriptor of the each candidate region by using

$$Dp_1 = \min_{\substack{1\le y\le n \\ 1\le \alpha,\beta\le i \\ 1\le a\le m}} \left| p_{1_{(a)}}{}^{(\alpha)} - hp_y{}^{(\beta)} \right|, \quad (11)$$

$$Dp_2 = \min_{\substack{1\le y\le n \\ 1\le \alpha,\beta\le i \\ 1\le a\le m}} \left| p_{2_{(a)}}{}^{(\alpha)} - hp_y{}^{(\beta)} \right|, \quad (12)$$

$$\vdots$$

$$Dt_1 = \min_{\substack{1\le z\le l \\ 1\le \alpha,\beta\le i \\ 1\le a\le m}} \left| t_{1_{(a)}}{}^{(\alpha)} - ht_z{}^{(\beta)} \right|, \quad (13)$$

$$Dt_2 = \min_{\substack{1\le z\le l \\ 1\le \alpha,\beta\le i \\ 1\le a\le m}} \left| t_{2_{(a)}}{}^{(\alpha)} - ht_z{}^{(\beta)} \right|, \quad (14)$$

$$\vdots$$

$$DP = \sum_{\alpha=1}^{f} Dp_\alpha, \quad (15)$$

$$DT = \sum_{b=1}^{k} Dt_b, \quad (16)$$

$$D = DP + DT, \quad (17)$$

Where, n is the number of peak of a smoothed hue histogram in a divided region β. i is the number of division. m is the number of teaching data. l is the number of trough of a smoothed hue histogram in a divided region β. $Dp_1, Dp_2, \cdots$ which are the smallest difference values between $hP^{(\beta)}$ to $p_{1_{(a)}}{}^{(\alpha)}, p_{2_{(a)}}{}^{(\alpha)}, \cdots$. $p_{1_{(a)}}{}^{(\alpha)}, p_{2_{(a)}}{}^{(\alpha)}, \cdots$ which are each peak position of a smoothed hue histogram in a divided region α of a. $Dt_1, Dt_2, \cdots$ are the smallest difference values between $hT^{(\beta)}$ to $t_{1_{(a)}}{}^{(\alpha)}, t_{2_{(a)}}{}^{(\alpha)}, \cdots$. $t_{1_{(a)}}{}^{(\alpha)}, t_{2_{(a)}}{}^{(\alpha)}, \cdots$ which are each trough position of a smoothed hue histogram in a divided region α of a. f is the number of peak of a smoothed hue histogram in a divided region α of a. k is the number of trough of a smoothed hue histogram in a divided region α of a. DP is the total value of difference value of peak position. DT is the total value of difference value of trough position. D is the total difference value. As an example, Figure 18 shows the matching based on the positions of peak and trough of the hue histogram.

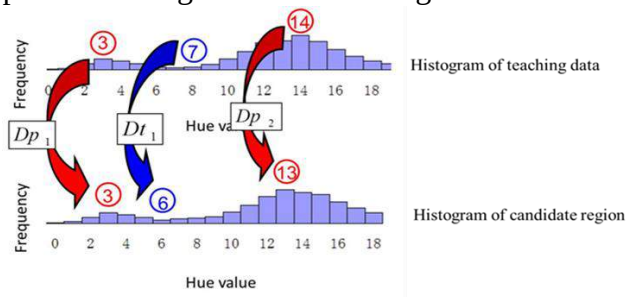


Figure 18. An example of matching based on the positions of peak and trough of the hue histogram.

As shown in Figure 18, the proposed method compares the positions of peak and trough of a smoothed hue histogram of teaching data with the positions of peak and trough of a smoothed hue histogram of candidate region, and calculates difference values. Furthermore, the proposed method registers a peak and trough having the smallest difference value as the nearest peak and trough. Finally, the proposed method recognizes the object with the smallest D in scene data as target object.

3 EXPERIMENT

In this section, to evaluate effectiveness of the proposed method, we carry out a quantitative comparison against the previous research about six properties mentioned in section 1 as follows

1. Robustness against occlusion
2. Fast recognition
3. Pose estimation with high accuracy
4. Coping with erroneous correspondences
5. Recognizing objects in a noisy environment
6. Recognizing objects which have same shape but have different texture

For the properties 1 to 5, the effectiveness of the previous research is shown in the paper relating to the previous research [4]. The proposed method adopts the previous research in order to extract candidate regions. Therefore, we only show the effectiveness for property 6 of the proposed method in this paper.

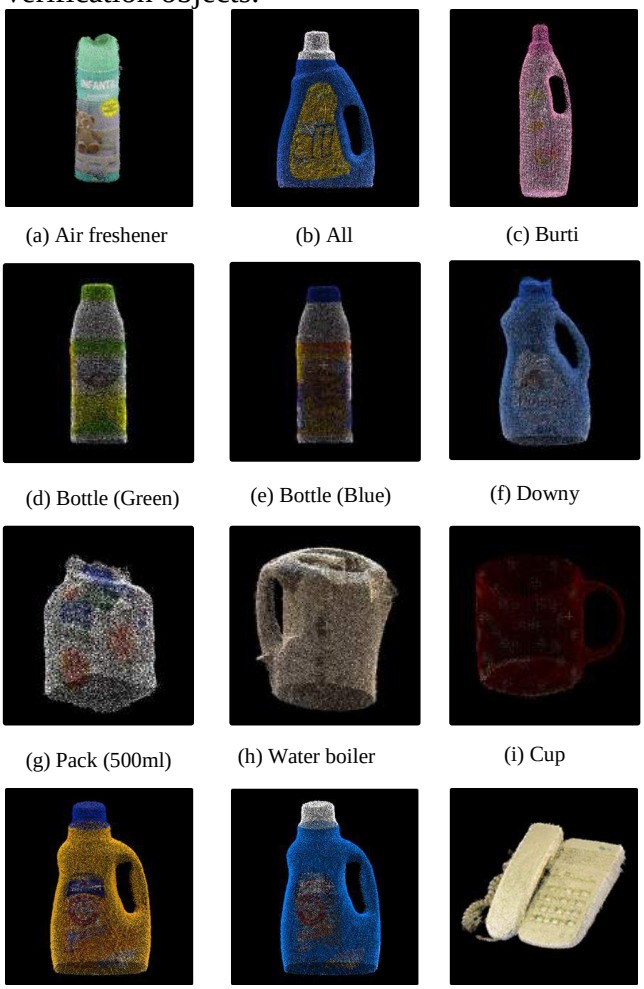
3.1 Quantitative Comparison Experiment Relating to Effectiveness

3.1.1 Experimental Overview

In this experiment, we compared the proposed method with the previous research quantitatively to evaluate about a property as follows

6. Recognizing objects which have same shape but have different texture

We selected 15 objects frequently used in domestic environment from TUW Object Instance Recognition Dataset [11] as verification objects. Figure 19 and Figure 20 show verification objects.



(a) Air freshener (b) All (c) Burti

(d) Bottle (Green) (e) Bottle (Blue) (f) Downy

(g) Pack (500ml) (h) Water boiler (i) Cup

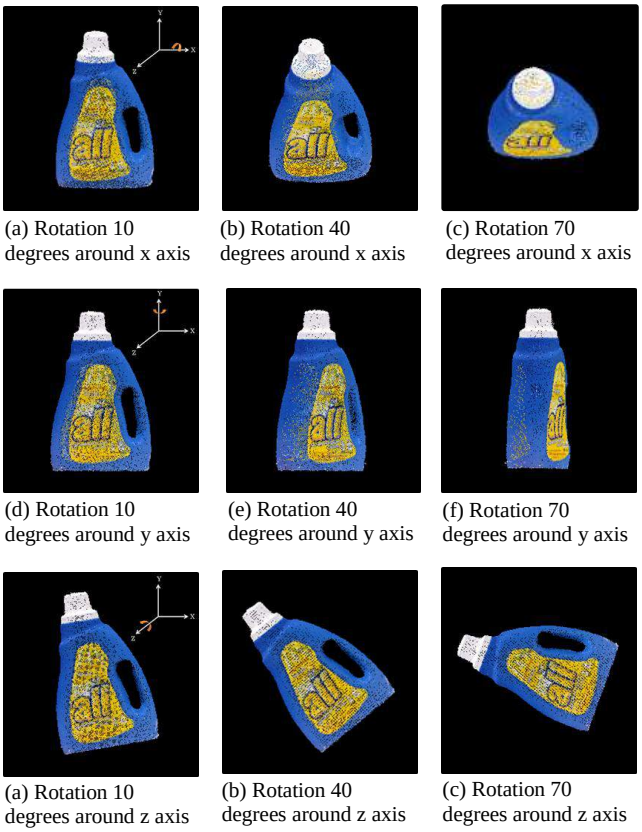
(j) Arm & Hammer (Yellow) (k) Arm & Hammer (Blue) (l) Telephone

Figure 19. Verification objects.

(m) Coca cola (350ml) (n) Doll (o) Toy car

Figure 20. Verification objects.

Object (d) and (j) each have the same shape and different texture as object (e) and (k), respectively. These verification objects have shape information and texture information for the entire circumference of an object. Thereby, even if rotation is added to a recognition target existing in the scene data, it is expected that pose estimation with high accuracy can be performed. Therefore, in this experiment, in addition to the above evaluation for the property 6, we evaluated the correspondence rate of pose estimation at each rotation angle. To generate rotation scene, we rotate each 3-dimensional object data by 10 degrees up to 90 degrees around each axis (X axis, Y axis and Z axis) as shown in Figure 21.



(a) Rotation 10 degrees around x axis (b) Rotation 40 degrees around x axis (c) Rotation 70 degrees around x axis

(d) Rotation 10 degrees around y axis (e) Rotation 40 degrees around y axis (f) Rotation 70 degrees around y axis

(a) Rotation 10 degrees around z axis (b) Rotation 40 degrees around z axis (c) Rotation 70 degrees around z axis

Figure 21. The example of rotation scene of the all.

To evaluate effectiveness of each method, we calculate the recognition rate by using

$$A = \frac{c}{z} \times 100\ [\%] \qquad (18)$$

Where, A is the recognition rate. c is the number which the each method could correctly recognize objects. z is the number of the verification object. A method of finding the recognition rate is that each method extract the most similar object from scene data which is each verification object with 2.5-dimensional in each verification object, and when it is equal to the teaching data, it is counted as correct recognition c. Detailed settings for recognizing a target object of each method are as follows. In the previous research, to evaluate pose estimation accuracy of a target object, we use the corresponding rate M mentioned in the rigid registration process (section 2.8). The corresponding rate M is calculated by using the equation (2) and (3) with th_c which is 1 [mm]. In case that, the corresponding rate M is the highest and is 70 percent or more, the object having that rate is recognized as a target object. In the proposed method, to evaluate pose estimation accuracy of a target object, we use the corresponding rate M as in the previous research. And then, in case that, the corresponding rate M is the highest and is 70 percent or more, an object having it is recognized as candidate object. Furthermore, to accurately recognize a target object from the candidate objects, we use the difference value D mentioned in the feature matching process (section 2.13). The difference value D is calculated by cumulating the difference value (Dp_i, Dt_j) for each divided region. We define the number of division as four in this experiment. In case that, the difference value D is the lowest, an object having it is recognized as a target object. Also, the reported processing time is obtained using Intel(R) Core(TM) i7 2.20GHz with 8.00 GB of main memory.

3.1.1 Experimental Overview

Figure 22 shows the recognition rate of the previous research and the proposed method for each rotation angle around x axis as experimental result.

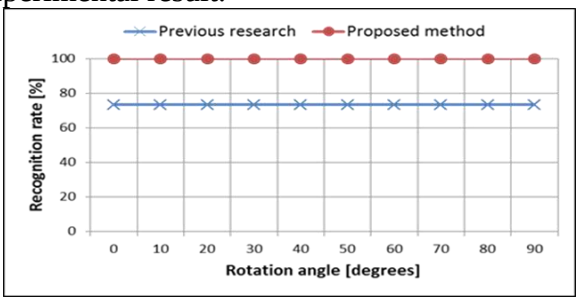


Figure 22. The recognition rate for each rotation angle around x axis.

As shown in Figure 22, the proposed method obtained a high recognition rate which is 100 percent at each rotation angle, whereas the data of the previous research is a slightly lower recognition rate than the proposed method's rate. The same can be said for Figure 23 and Figure 24.

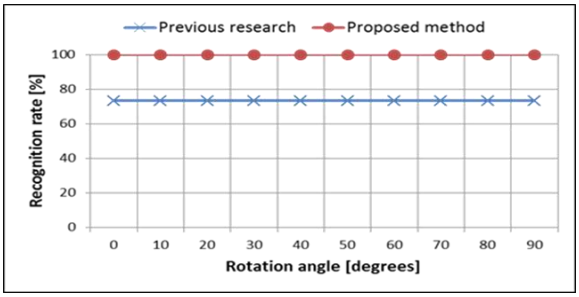


Figure 23. The recognition rate for each rotation angle around y axis.

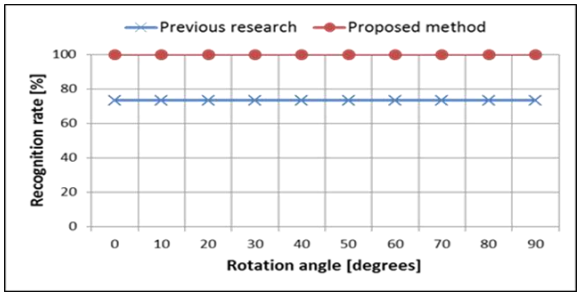


Figure 24. The recognition rate for each rotation angle around z axis.

Furthermore, Table 5 shows a processing time of the previous research and the proposed method in each verification object.

Table 5. The average processing time in each verification object.

	Average processing time [ms]														
	Air freshener	All	Burti	Bottle (Green)	Bottle (Blue)	Downy	Pack	Water boiler	Cup	Arm & Hammer (Yellow)	Arm & Hammer (Blue)	Telephone	Coca cola	Doll	Toy car
	410834 points	48268 points	38444 points	19346 points	19346 points	39996 points	13679 points	66109 points	24933 points	52381 points	52381 points	785093 points	15026 points	628812 points	16641 points
Previous research	122.82	52.19	7.46	134.28	134.28	24.10	5.32	123.51	7.40	83.80	83.80	248.80	4.78	219.90	6.29
Proposed method	157.09	87.17	42.44	167.56	167.84	57.55	38.46	159.25	38.01	118.00	118.55	281.25	38.48	256.88	38.26

As shown in Table 5, the difference between the processing time of the proposed method and the processing time of the conventional research was not as large as 30 ms to 40 ms.

3.1.1 Discussion

In this section, we discuss the results presented in experimental results (section 3.1.2).
As shown in Figure 22, Figure 23 and Figure 24, it can be seen that the recognition rate of the previous research maintains 73 percent at all rotation angles around each axis. Therefore, we confirm which object the recognition rate of the previous research is decreasing. In this experiment, the recognition rate of the previous research decreased at the object (d), (e), (j) and (k). As an example, the corresponding rates for each scene data when the object (j) is used as teaching data are shown as follow.
As shown in Table 6, the corresponding rate of object (j) and (k) were equal. This is because the previous research uses only shape information of an object, it is impossible to accurately distinguish objects having same shape and different texture like object (j) and (k). On the other hand, since the proposed method uses not only shape information of an object but also texture information of an object, it was able to perform a robust recognition at all verification objects as shown Figure 22, Figure 23 and Figure 24. From these results, the effectiveness of the proposed method for the property 6 was shown.

4 CONCLUSION

In this paper, we proposed the 3-dimensional object recognition method using relationships between feature points, and invariance of local hue histogram for the purpose of improving the recognition technology for the life support robot. The proposed method has effectiveness for six properties necessary as follows for recognition in the domestic environment.

1. Robustness against occlusion
2. Fast recognition
3. Pose estimation with high accuracy
4. Coping with erroneous correspondences
5. Recognizing objects in a noisy environment
6. Recognizing objects which have same shape but have different texture

From experimental results of quantitative comparison experiment relating to effectiveness, we considered that the proposed method can accurately distinguish objects which have same shape but have different texture. In addition, since the difference between the processing time of the proposed method and the processing time of the previous research is 30 ms to 40 ms which is not very large, we considered that the proposed method can perform recognition with high speed and high accuracy.
However, since SHOT descriptor used by the proposed method is difficult to describe features for objects including a lot of planes like box, pose estimation on ones may not be performed well. Therefore, when describing features of an object, we describe not only the shape feature but also the texture feature, and estimate the pose by using them. Thereby, we improve pose estimation accuracy of the proposed method in a feature work.

REFERENCES

1. S. Sugano, T. Sugaiwa, and H. Iwata, “Vision System for Life Support Human-Symbiotic-Robot,” The Robotics Society of Japan, 27(6), pp. 596-599, 2009.
2. T. Odashima, M. Onishi, K.Tahara, T. Mukai, S. Hirano, Z. W. Luo, and S. Hosoe, “Development and Evaluation of a Human-interactive Robot Platform “RI-MAN”,” The Robotics Society of Japan, 25(4), pp. 554-565, 2007
3. Y. Jia, H. Wang, P. Sturmer, and N. Xi, “Human/robot interaction for human support system by using a mobile manipulator,” Robotics and Biomimetics (ROBIO), pp. 190-195, 2010.
4. H. Kudo, K. Ikeshiro, and H. Imamura, “A 3-Dimensional Object Recognition Method Using SHOT and Relationship of Distances and Angles in Feature Points,” International Journal of New Computer Architectures and their Applications

Table 6. The corresponding rate for each scene data at 0 degrees.

	Corresponding rate [%]														
	Air freshener	All	Burti	Bottle (Green)	Bottle (Blue)	Downy	Pack	Water boiler	Cup	Arm & Hammer (Yellow)	Arm & Hammer (Blue)	Telephone	Coca cola	Doll	Toy car
Arm & Hammer (Yellow)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	74.86	74.86	0.0	0.0	0.0	0.0

(IJNCAA), 7(4), pp. 149-155, 2017.
5. F. Tombari, S. Salti, and L. D. Stefano, "Unique signatures of histograms for local surface description," European conference on computer vision (ECCV), pp. 356-369, 2010.
6. INTELLIGENT SENSING LABORATORY, http://isl.sist.chukyo-u.ac.jp/Archives/tm.html
7. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60(2), pp. 91-110, 2004.
8. M. J. Swain, and D. H. Ballard, "Color Indexing," International Journal of Computer Vision, 7(1), pp. 11-32, 1991.
9. T. Kanda, K. Ikeshiro, and H. Imamura, "An Object Detection Method Using Invariant Feature Based on Local Hue Histogram in Divided Areas of an Object," International Journal of New Computer Architectures and their Applications (IJNCAA), 7(4), pp. 112-122, 2017.
10. M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Comm. of the ACM, 24(6), pp. 381-395, 1981.
11. AUTOMATION & CONTROL INSTITUTE (ACIN), https://repo.acin.tuwien.ac.at/tmp/permanent/dataset_index.php

International Journal of
NEW COMPUTER ARCHITECTURES AND THEIR APPLICATIONS

The *International Journal of New Computer Architectures and Their Applications* aims to provide a forum for scientists, engineers, and practitioners to present their latest research results, ideas, developments and applications in the field of computer architectures, information technology, and mobile technologies. The IJNCAA is published four times a year and accepts three types of papers as follows:

1. **Research papers**: that are presenting and discussing the latest, and the most profound research results in the scope of IJNCAA. Papers should describe new contributions in the scope of IJNCAA and support claims of novelty with citations to the relevant literature.
2. **Technical papers**: that are establishing meaningful forum between practitioners and researchers with useful solutions in various fields of digital security and forensics. It includes all kinds of practical applications, which covers principles, projects, missions, techniques, tools, methods, processes etc.
3. **Review papers**: that are critically analyzing past and current research trends in the field.

Manuscripts submitted to IJNCAA **should not be previously published or be under review** by any other publication. Plagiarism is a serious academic offense and will not be tolerated in any sort! Any case of plagiarism would lead to life-time abundance of all authors for publishing in any of our journals or conferences.

Original unpublished manuscripts are solicited in the following areas including but not limited to:

- Computer Architectures
- Parallel and Distributed Systems
- Storage Management
- Microprocessors and Microsystems
- Communications Management
- Reliability
- VLSI