Fingerprint Matching Using Bozorth3 Algorithm and Parallel Computation on NVIDIA Compute Unified Device Architecture

S Supatmi1*, I D Sumitra2
1Department of Computer System, Faculty of Engineering and Computer Science, Universitas Komputer Indonesia, Indonesia
2Departemen Magister Sistem Informasi, Universitas Komputer Indonesia, Indonesia

Email: * sri.supatmi@email.unikom.ac.id

Abstract. This paper studied fingerprint matching employing Bozorth3 Algorithm for matching fingerprint and parallel computation employing NVIDIA Compute Unified Device Architecture (NVIDIA CUDA). The objective of this study obtains the percentage and time processing of matching fingerprints. In this study, the fingerprint matching is done with parallel computing is applied to the GPU (Graphics Processing Unit). GPU device used in this study is the CUDA (Compute Unified Device Architecture), which is an Application Programming Interface (API) developed by NVIDIA. The development of applications with fingerprint matching serial computing on CPU and parallel computing on GPU can be applied to the CUDA API. The results from this study can be found in the performance process on the CPU and GPU. The results of this research are the process on CUDA execution time is better than the execution time on the CPU, the process is done at both the computation is to find a match in the fingerprint value.

1. Introduction
Fingerprint matching using Bozorth3 algorithm [1] and parallel computation on NVIDIA CUDA study is how to match the fingerprints and obtain the percentage and time processing of matching fingerprint. Bozorth3 algorithm used to match the minutiae fingerprint, and NVIDIA CUDA used to parallel computation to matching for one fingerprint to many fingerprints. Biometric technologies are automated methods of verifying or recognizing the identity of a living person based on physiological or behavioral characteristics [2-4]. One of the areas of biometrics is used as the object in this study is the physical form of a human fingerprint. Security system using fingerprints safer than pins or passwords because of humans’ fingerprints have different types and will not be the same from one person to another person that is not easy to imitate or be known by others. Fingerprint minutiae are one of the critical elements in the fingerprint matching process due to the number and location of these minutiae; the human fingerprint image can be recognized Process of fingerprint matching computation, which aims to determine the degree of match between a fingerprint with the other. Minutiae fingerprint used in this study is fingerprinted on a human thumb hand and minutia of data that has been extracted through the minutiae detection. From these minutiae, it can be calculated distances and angles between minutiae in a fingerprint that can be used for fingerprint matching process with the other.

The previous study showed on matching fingerprint employing the Bozorth algorithm and Boyer-Moore and obtained processing time of matching fingerprint in serial processing [5]. In paper [6] studied the proposed scheme for fingerprint recognition system using the bozorth3 algorithm and successful in
recognizing the fingerprint to paperless documentation. In paper [7] studied the fingerprint recognition system towards minimizing the effect of partial images and obtain the performance of the fingerprint recognition influenced by the excellent quality of the fingerprint image. Sudiro and Yuwono [8] studied the performance evaluation for single fingerprint extraction on valley structure using the Bozorth3 algorithm and obtained 12.37% of minutiae error detecting. Pablo et al. [9] Research in the fingerprint matching system for large databases employing minutia cylinder-code (MCC) based on GPU and obtained the GPU time for the matching process from 603.6ms to 105424.8ms. Ghafoor et al. [10] Studied the efficiency of matching fingerprint using the matcher, which integrated on GPU and obtained the excellent performance for large-scale applications of automated fingerprint image systems with the average matching speed from 26 seconds to 50 seconds.

The purposes of this research to obtain a better percentage of matching minutiae fingerprints and time processing of matching. The methods used in this research are the Bozorth3 algorithm and NVIDIA CUDA. Bozorth3 algorithm used to match the minutiae for fingerprint, and NVIDIA CUDA used to match the fingerprint on parallel computation in GPU.

2. Methods
2.1. Bozorth Algorithm
Bozorth3 algorithm is an algorithm that focuses on the process of its match minutiae points that have the location (x, y) and orientation angle theta (t) is represented as xyt files using the rotation and translation invariant. Bozorth3 algorithm consists of three critical steps, as follows:[5,6,10-12]

a. Construct Intra-Fingerprint Minutiae Comparison Tables
Creating a table containing the distance and angle values between minutiae in a fingerprint. One table for probe and one table for the gallery (shown in Figure 1).

![Figure 1. Minutiae Intra-fingerprint][11]

b. Construct an Inter-Fingerprint Compatibility Table
Comparing the distance and angle values that have been stored in each table on the probe and gallery (shown in Figure 2).

c. Traverse the Inter-Fingerprint Compatibility Table
Calculate and display the final value of the fingerprint matching process of step two.
2.2. Compute Unified Device Architecture (CUDA)

CUDA is a parallel computing platform and programming model invented by NVIDIA and implemented by the General Processing Unit (GPU) [14-25].

Examples of CUDA processing flow: [23–25]
1. Copy data from main memory to GPU memory
2. CPU instructs the process to GPU
3. GPU execute parallel in each core
4. Copy the result from GPU memory to main memory

The way to execution sequence of instructions in CUDA using a set of threads in a block is executed simultaneously. Here is an example of a program in parallel instruction execution in CUDA thread [23-25]. Figure 3 described the example code to execution the instructions on CUDA thread.

```c
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N], float C[N][N]) {
    int i = threadIdx.x;
    int j = threadIdx.y;
    C[i][j] = A[i][j] * B[i][j];
}

int main() {
    ...
    // Kernel invocation with one block of N * N * 1 threads
    int numBlocks = 1;
    dim3 threadsPerBlock(N, N);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

Figure 3. Execution of instructions on CUDA thread
3. Results and Discussion

The problem of this research is how to incorporate a fingerprint minutiae existing fingerprint minutia matching way by calculating the distance and angle of fingerprint minutiae, and parallel computing to process fingerprint matching. The final stage of this system will make the process of computing on NVIDIA CUDA.

The modeling for fingerprint matching is assumed that it will fingerprint is matched fingerprints right-hand thumb. The fingerprints will be matched minutiae points existing as well as the location has been changed in the form (x, y) and orientation theta t. Stages in the fingerprint matching process consist of three stages:

1. Started with the process of incorporating minutiae points \{x, y, t\} to the database in the probe (test data) and entered into a database gallery (data testers).
2. Started by calculating the distance and angle on both the fingerprint in the probe and the gallery.
3. Started by comparison of distances and angles fingerprints on the probe and gallery and calculate the value of the fit.

Software development on a parallel computing program is to send the value of \{x, y, t\} into shared memory so that the distance and the angle can be calculated. Values \{x, y, t\}, which has been sent to the shared memory, will speed up the process of access distance and angle values by a thread.

Each thread will do the process of parallel computing from the data input, which has been obtained from the stages or steps described in the previous section. The results of computation on each thread to one fingerprint are the distance, and angle values are displayed in a single array for one unit of time. The results are then sent back to the host in the form of an array. The process of computing the last of these, the host will compare the value of the array as a whole on the value in the probe and in the gallery and provide information on the percentage of the value matches the fingerprint comparison.

Stages of tests performed on computing fingerprint matching are done with two types of parallel computing systems and computing with the Serial system (host). The process of fingerprint matching is done on a parallel system using the NVIDIA CUDA. The results of this test are to get some output values to be analyzed, such as performance serial computing and parallel computing on CUDA C and variable research to analyze the performance of computing.

The explanation of the diagram block for Figure 4 described as follows. Fingerprint data already contains minutiae points \{x, y, t\} are used in this study derived from data from the National Institute of Standards.
and Technology (NIST) in the form of a fingerprint image namely file xyt of minutiae detection process (MINDTCT).

Parameters or variables of the study:
- File fingerprint minutiae with file formats *.xyt;
- Compliance calculation result data in serial and parallel computing;
- Execution Time on GPU and CPU.

Testing Steps:
- Computing fingerprint matching is obtained from the value of fingerprint minutiae location \{x, y\} and orientation \theta;
- The process of testing computing one-to-one. Values obtained from the input of the fingerprint data that already exists;
- Perform a comparison of serial and parallel computing processes.

The performance test occurred in two categories, such as test one-to-one and test one-to-many. Tests performed on the set of minutiae from one fingerprint with the fingerprint else have calculated the value of the distance, the angle, and obtained the value of matching both percentage and processing time shown in Table 1 and Table 2.

### Table 1. Test Results of the one-to-one method of fingerprint matching

| No. | Testing minutiae | Matching value (%) | GPU time-Parallel Processing (ms) | CPU time-Serial Processing (ms) |
|-----|------------------|-------------------|---------------------------------|-------------------------------|
| 1   | 1.xyt to min1.xyt| 95                | 1.024                           | 6.93                          |
| 2   | 1.xyt to min100.xyt| 0            | 1.021                           | 6.91                          |
| 3   | 100.xyt to min100.xyt| 91         | 1.024                           | 6.93                          |
| 4   | 1.xyt to min200.xyt| 71.5             | 1.028                           | 6.95                          |
| 5   | 2.xyt to min2.xyt| 96.7             | 1.024                           | 6.93                          |

Table 1 described the results of matching fingerprints employing one by one file of minutiae of the fingerprint. The matching value obtained the higher percentage if the value of one minutiae file, which compared with other minutiae file has the same value. The fingerprints of the testing minutiae have a comparable value for the identic fingerprint on 95 to 96.7% and for the different fingerprint has the values from 0 to 71.5%. These results of identic or same fingerprint have error detected from 5 to 3.7%, and still better than the other research which using the same Bozorth3 algorithm on research [8] obtained 12.3% error detecting of the fingerprint matching.

### Table 2. Test Results of the one-to-many method of fingerprint matching

| No. | Testing Minutiae | Matching value (%) | GPU time-Parallel Processing (ms) | CPU time-Serial Processing (ms) |
|-----|------------------|-------------------|---------------------------------|-------------------------------|
| 1   | 1 to 100         | 99.2              | 10.23                           | 69.26                         |
| 2   | 1 to 200         | 0.5               | 16.73                           | 113.26                        |
| 3   | 1 to 300         | 92.9              | 23.23                           | 157.27                        |
| 4   | 1 to 400         | 73.5              | 29.73                           | 201.27                        |
| 5   | 1 to 500         | 91.6              | 36.23                           | 245.28                        |

Table 2 shows the results of matching fingerprints employing one file, then comparing it with many files of minutiae fingerprints. The matching value obtained the higher percentage if the value of one minutiae file, which compared with many minutiae files have the same value.

The processing time, both Table 1 and Table 2, obtained the GPU time faster than CPU time on this research and many files compared need more time to match the minutiae files. This research results
obtained a better performance both of CPU time and GPU time on matching fingerprint process than the other research in [9,10].

4. Conclusion

Based on the testing and analysis that has been done a few conclusions can be drawn as follows: the fingerprint matching model that has been used in the testing process has the same data with the serial matching process; results match fingerprints similar to testing a one-to-one match maximum percentage more than 90%; execution time on the GPU looks faster when compared to the execution time on the CPU. The use of threads in computing a significant impact on improving the performance of computing itself. Suggestions for the development of systems in this study are: the research can be continued by making fingerprint recognition system is directly connected to a device; research can be developed for fingerprint matching in parallel using the method of one-to-many on Bozorth3 algorithms; it is expected that the application of parallel computing can be applied to an institution such as the Bank or educational institutions, so that the security system more secure and faster execution time of the serial computing.

Acknowledgment

We would like to thank Prof. Hou Rongtao for his support in this research. Besides, thank you to the Rector of Universitas Komputer Indonesia Prof. Dr. Ir. H. Eddy Soeryanto Soegoto, M.T., for the supporting funding and facilities for the accomplishment of this research.

References

[1] C. Watson et al., 2004 “User’s Guide to Export Controlled Distribution of NIST Biometric Image Software,” Control, pp. 14–21.
[2] J. Wayman, A. Jain, D. Maltoni, and D. Maio, 2005 “An Introduction to Biometric Authentication Systems,” Biometric Syst., pp. 1–20.
[3] E. Bowman, January, 2000 “Everything You Need to Know About Biometrics,” no..
[4] J. Wayman, January 2000, 2018 “Prepared under DoD Contract MDA904-97-C-03 and FAA Award DTFA0300P10092,” no..
[5] S. Supatmi and I. D. Sumitra, 2019 “Fingerprint Identification using Bozorth and Boyer-Moore Algorithm,” IOP Conf. Ser. Mater. Sci. Eng., vol. 662, no. 2.
[6] S. Zanwar, P. Nangare, V. Supe, and J. Yadav, 2017 “A PROPOSED SCHEME FOR FINGERPRINT RECOGNITION SYSTEM USING BOZORTH3, 4(05), pp. 269–270.
[7] I. Khurjekar, B. Garware, and A. Abhyankar, 2016 “Towards minimizing effect of partial fingerprint images on the performance of fingerprint recognition systems,” Proc. - IEEE Int. Conf. Inf. Process. ICIP 2015, pp. 839–842.
[8] S. A. Sudiro, M. Paindavoine, R. T. Yuwono, and L. Etp, 2008 “Performance evaluation of simple fingerprint minutiae extraction algorithm using crossing number on valley structure,” 2008 Int. Conf. Innov. Inf. Technol. IIT 2008, pp. 623–627.
[9] P. D. Gutierrez, M. Lastra, F. Herrera, and J. M. Benitez, 2014 “A high performance fingerprint matching system for large databases based on GPU,” IEEE Trans. Inf. Forensics Secur., 9(01) pp. 62–71.
[10] M. Ghafoor, S. Iqbal, S. A. Tariq, I. A. Taj, and N. M. Jafri, 2018 “Efficient fingerprint matching using GPU,” IET Image Process., 12(2), pp. 274–284.
[11] N. Fingerprint and I. 2001Software, “User ’ s Guide to NIST Fingerprint Image Software (NFIS),” vol. 6813.
[12] S. Maddala, J. S. Bartunek, and M. Nilsson, 2010 “IMPLEMENTATION AND EVALUATION OF NIST BIOMETRIC IMAGE SOFTWARE FOR FINGERPRINT RECOGNITION,” Int. Conf. Signal Image Process., pp. 207–211.
[13] S. Maddala, S. R. Tangellapally, J. S. Bartuněk, and M. Nilsson, 2011 “Implementation and evaluation of NIST Biometric Image Software for fingerprint recognition,” 2011 ISSNIP Biosignals Biorobotics Conf. Biosignals Robot. Better Safer Living, BRC 2011, pp. 18–22.
[14] S. Collange, D. Defour, and D. Parello, 2009 “Barra, a modular functional gpu simulator for gpgpu,” Simulation, no. February 2009.

[15] S. Cuomo, L. Marcellino, and G. Navarra, 2018 “A parallel implementation of the Hestenes-Jacobi-One-Sides method using GPU-CUDA,” 26th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. A, pp. 722–725.

[16] S. Datla and N. S. Gidijala, 2009 “Parallelizing motion JPEG 2000 with CUDA,” 2009 Int. Conf. Comput. Electr. Eng. ICCEE 2009, vol. 1, pp. 630–634.

[17] M. Fatica, “CUDA Toolkit and Libraries Closely Coupled CPU-GPU,” 2008 IEEE Hot Chips 20 Symp., pp. 1–22.

[18] L. Gang and L. Yujun, 2013 “CUDA based parallel wavelet algorithm in medical image fusion,” Proc. - 2013 2nd Int. Symp. Instrum. Meas. Sens. Netw. Autom. IMSNA 2013, pp. 198–201.

[19] H. Li, D. Yu, A. Kumar, and Y. C. Tu, 2015 “Performance modeling in CUDA streams - A means for high-throughput data processing,” Proc. - 2014 IEEE Int. Conf. Big Data, IEEE Big Data 2014, pp. 301–310.

[20] S. N. V, R. K, and N. N. Chiplunkar, 2017 “Performance Improvement of CUDA Applications by Reducing CPU-GPU Data Transfer Overhead,” Int. Conf. Inven. Commun. Comput. Technol. (ICICCT 2017) Perform., no. Icicct, pp. 211–215.

[21] M. Wezowicz, 2007 “On the cost of a general GPU framework -,” 2012 SC Companion High Perform. Comput. Netw. Storage Anal., pp. 5–6.

[22] E. Yalcin, H. Badem, and M. Gunes, 2015 “CUDA-based hybrid intuitionistic fuzzy edge detection algorithm,” IEEE Int. Conf. Fuzzy Syst., vol. 2015-Novem, no. 3.

[23] C. Zeller, 2011 “CUDA C / C ++ Basics What is CUDA ?,” NVIDIA Dev. Zo..

[24] “CUDA c programming guide,” no. April, 2018.

[25] D. B. Kirk and W. W. Hwu, 2013. Programming Massively Parallel Processors A Hands-on Approach 2nd.pdf.