θ(1) TIME COMPLEXITY PARALLEL LOCAL NEIGHBOURHOOD DIFFERENCE PATTERN FEATURE EXTRACTOR ON A GRAPHICAL PROCESSING UNIT

Pranshul Goyal
Manipal Institute of Technology,

Manipal Academy of Higher Education,
Manipal, Karnataka State, India,
PIN-576104
pranshul20162002@gmail.com

Aashna Nitin Kunkolienker
Manipal Institute of Technology,

Manipal Academy of Higher Education,
Manipal, Karnataka State, India,
PIN-576104
aashnakunk@gmail.com

Ashwath Rao B
Manipal Institute of Technology,

Manipal Academy of Higher Education,
Manipal, Karnataka State, India,
PIN-576104
ashwath.rao@manipal.edu

***Abstract*—Local Neighbourhood Difference Pattern (LNDP) is a fairly new feature extraction method which has found its application in the field of facial recognition and medicine, such as building an automatic epilepsy detection framework. In such cases, the time taken to execute these tasks is very precious. Hence, to speed up the process, we have proposed a parallel approach involving the graphical processing unit. Through this work, we provide a θ(1) time complexity extraction algorithm for calculating the Local Neighbourhood Difference Pattern Features of an image. We tested this implementation on medical images from the LISS database.**

Keywords*—*Local Neighbourhood Difference Pattern, Parallel Programming, Graphical Processing Unit, CUDA, Medical image processing

I. INTRODUCTION

Texture is one of the predominant characteristics used in image analysis. Local Binary Pattern is a pilot local texture descriptor with considerably high efficacy. The initial approach for LBP was sequential. It took a lot of time to compute pixel values. Since the pixel value calculations are independent of each other, a parallel approach is much faster and efficient for this task. Many parallel approaches have been proposed for LBP algorithm. Some use the OpenCl framework, while others utilize Nvidia's CUDA framework. Compute Unifies Device Architecture (CUDA) is a parallel computing platform developed by Nvidia for general computing on its own graphical processing units.

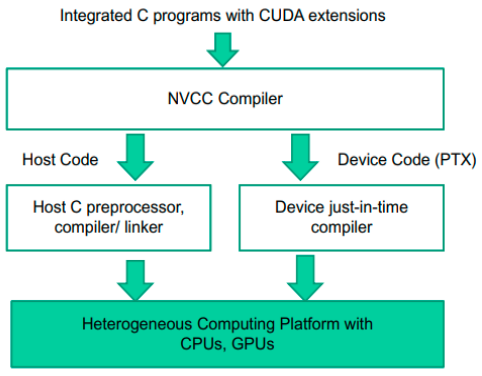

Fig. 1 CUDA program structure

Nowadays, many software applications operate on datasets which contain real-world phenomena. Data parallelism enables us to run these applications in a very efficient manner, since it ensures that multiple threads operate on different segments of the input data, thereby decreasing the end to end execution time. CUDA programs comprise of a host and multiple devices. The CPU is the host, whereas the GPUs are the devices. The chunk of code corresponding to the host is compiled with its standard C compiler. On the other hand, the GPU's code is compiled by the NVDIA CUDA compiler and executed on a GPU device.

The execution of a CUDA program starts with the host and then, a "kernel" function is called. This function is executed by a large number of threads on the device. A group of threads form a block. A group of blocks form a grid. Threads in a block run in parallel, each thread performing the same task.

Fig2. CUDA thread organisation

In feature extraction algorithms, operations on each pixel or a group of pixels are carried out by a thread, and all threads run in parallel. Upon the completion of operations

on each pixel of the input image, the final computed result corresponding to every pixel is sent back to the host.

"Local neighbourhood difference pattern" is a recently developed feature extraction method proposed by M. Verma and B. Raman [1]. Image feature extraction is used by content based image retrieval in order to search for images similar to an input image. LNDP extracts local features based on neighbourhood pixel differences and forms a binary pattern to represent each pixel in the image.

For each pixel we select a 3x3 block of pixels surrounding the pixel. Each neighbouring pixel is then compared to two most adjacent and appropriate pixels. We obtain two values for each of the neighbouring pixels using these two comparisons. The formulae for the same is mentioned below:

$$kn1 = I8 - In,\ kn2 = In+1 - In,\ \text{for } n = 1 \quad (1)$$

$$kn1 = In-1 - In,\ kn2 = In+1 - In,\ \forall\ n = 2, 3, .., 7 \quad (2)$$

$$kn1 = In-1 - In,\ kn2 = I1 - In,\ \text{for } n = 8 \quad (4)$$

Then, we obtain the final binary value for each pixel by applying XNOR operation on those values

$$F3(k1n,k2n) = \begin{cases} 1, & \text{if } k1n >= 0 \ \&\ k2n >= 0 \\ 1, & \text{if } k1n < 0 \ \&\ k2n < 0 \\ 0, & \text{if } k1n >= 0 \ \&\ k2n < 0 \\ 0, & \text{if } k1n < 0 \ \&\ k2n >= 0 \end{cases} \quad (5)$$

We get eight binary codes each having 1 or 0 and the code is converted into a byte by multiplying positional weights starting from 1 to 128. By this step, for each non-border pixel in an image we get a byte. This step is depicted in detail in figure 3.

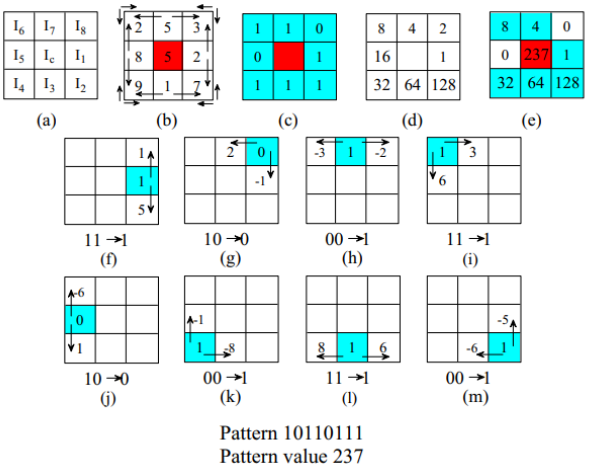

Fig. 3 LNDP calculation for center pixel (a) pixel index (b) example of pattern calculation for a window (c) computation of binary values (d) weights in their respective positions (e) weights multiplied by the corresponding pixel values

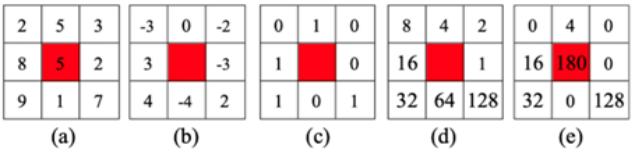

Fig. 4 LBP calculation (a) a 3×3 window (b) comparing the neighbouring pixel values with the centre pixel and computing the difference (c) Assigning binary values according to the threshold (d) weights in their respective positions (e) weights multiplied by the corresponding pixel values LNDP has been proposed to be used in association with the LBP feature extraction algorithm. Other methods are outperformed by this one, in terms of accuracy. Feature extraction is widely used in various fields. LBP is used in face identification [2 3 4], object recognition [5]. It has found diverse applications in the medical field too, like Multi-level Fusion in Ultrasound for Cancer Detection Multi-Level Fusion in Ultrasound for Cancer Detection [6], Age classification [7]. Another feature extraction method proposed for medical datasets is the parallel local Tri directional feature extraction method. [8] and the Local diagonal extrema pattern. [9]

LNDP is a relatively newer algorithm, and as of now, has found its usage in classifying epileptic EEG signals (building an automatic epilepsy detection framework) [10] and Recognition of Parkinson's Disease [11] .There has been no parallel implementation of the LNDP algorithm as yet. Through this paper, we aim to present a $\Theta(1)$ time complexity feature extraction method. We tested this algorithm on medical images from the publicly available LISS database.

The outline of the paper is as follows. Under section 2, we have thrown light on the methodology used by us to implement the algorithm. Under section 3, the setup used for our experiment and the observed results are discussed. In section 4, we have mentioned the conclusion and future scope of this work.

II. METHODOLOGY

CPUs are designed with a lesser number of processor cores whose clock speeds are higher, thus allowing a good performance on single threaded operations. But when we come to multi-threaded performance GPU outperforms a CPU due to its sheer number of processing elements. The major work in computing the LNDP values for pixels is mathematical computation which can be done in parallel as LNDP value of one pixel does not affect the LNDP value of another pixel. By executing massive number of threads at the same time all the pixel values can be computed at the same time given the GPU allows that many threads to execute at the same time. There have not been major changes in computing LNDP values over the years. So, in this work we present a method which will utilize the large memory present on the GPU to reduce the computational time complexity.

We propose a novel method where the LNDP is computed on the whole image at once. We create as many threads as the non-border elements in the image. If we assume the image size to be h*w pixels, then the feature vector size will be (h-2) *(w-2).

For the testing purpose we used medical images whose maximum resolution was 2048x2048 which is approximately 4 Megabytes. As evident from Table 1 and Table 2 modern GPUs can easily handle this data at the same time.

III. Experimental Setup and Results

The dataset we used was obtained from the publicly available LISS database. In order to process the input medical images (DICOM), we used the DTMCK library.

To execute the parallel code, we've used a laptop with intel iCore 7 8th Gen with NVIDIA RTX2060. The total memory available to the GPU is 6Gb @ 1750 MHz The maximum block size is (1024,1024,64).

Parallelisation of the code is done using CUDA architecture, instead of OpenCL, owing to CUDA's better performance.

We implemented the code with various block sizes, and the optimum performance was achieved when the block size was equal to the image size (only non-border elements), i.e., one thread computes the value for one pixel, and all threads run in parallel at the same time.

A. Results

In Table 4, the results of the sequential LNDP implementation are shown. As the input image size increases, execution time increases. There is a 3.3-fold increase as image size goes from 256 x 256 to 512 x 512 and a 12-fold increase as image size goes from 256 x 256 to 1024 x 1024 (a 16 fold increase in pixels).

Table 4. LNDP sequential execution time in milli seconds

Image Size	LNDP execution time (ms)	End to end execution time (ms)
256X256	11.384256	23.866528
512X512	64.319069	79.257828
1024X1024	266.733582	284.094147

We ran the parallel code on the same medical images, to get the following output. (Shown in Table 5)

Table 5. LNDP parallel execution

Image size	Thread block size	Memory loading time (ms)	Kernel execution time (ms)	Memory output time (ms)	End-end execution time (ms)
256 x 256	16 x 16	0.818688	0.840320	0.190336	16.1186
	32 x 32	0.863392	1.103552	0.133472	15.8446
	256 x 256	0.629248	0.002560	0.176768	14.1384
512 x 512	16 x 16	1.25738	2.820896	0.612608	21.4608
	32 x 32	0.948768	3.198976	1.0953	20.6165
	512 x 512	1.19654	0.002016	0.4632	18.5753
1024 x 1024	16 x 16	3.13901	13.474528	2.9953	42.3315
	32 x 32	1.49814	13.403456	3.10026	38.9026
	1024 x 1024	0.979968	0.004256	1.68659	20.5574

In our parallel implementation, as the image size increases four-fold from 256 x 256 to 512 x 512, kernel execution time increases by almost 4 times. As the image size increases 16-fold from 256 x 256 to 1024 x 1024, the kernel execution time increases 16-fold. Hence, we can come to a conclusion that when we use the GPU, parallel kernel execution time increases linearly to the increase in the number of pixels.

Table 6. Speedup of LNDP algorithm with respect to kernel execution time

Image size	Thread block size	Speedup
256 x 256	16 x 16	13.5475248
	32 x 32	10.3160123
	256 x 256	4446.975
512 x 512	16 x 16	22.8009359
	32 x 32	20.106143
	512 x 512	31904.3001
1024 x 1024	16 x 16	19.7953934
	32 x 32	19.9003587
	1024 x 1024	62672.3642

From Table 6, we can infer that the algorithm performs best when the image size and thread block size are the same. In other words, one thread will work on one pixel, and all the threads will run together in parallel.

We can observe a ~4500, ~32000, ~63000 speedup when the aforementioned condition is satisfied. Maximum performance (highest speedup) is achieved when the image size as well as thread block size is 1024 x 1024.

The time complexity for our LNDP algorithm is $\Theta(1)$, because each thread works on each non border element of our input image simultaneously.

IV. Conclusion and future scope

We incorporated parallel programming into the LNDP algorithm in order to reduce our code execution time, thereby enabling real-time applications for feature extraction to be faster. Today, image feature extraction finds its application in numerous domains. The high throughput of the GPU is evident as we got the parallel code to execute a maximum speed-up of ~63000. A lesser time complexity is obtained, thus making such parallel algorithms highly efficient. There is a huge future scope for this parallel algorithm, since the faster a feature extraction algorithm executes, the better it is. For example, if our algorithm is used in cancer detection using LNDP features, we'll be able to obtain results faster on a larger dataset, hence speeding up the detection process.

References

[1] Verma, M., Raman, B. Local neighborhood difference pattern: A new feature descriptor for natural and texture image retrieval. Multimed Tools Appl 77, 11843–11866 (2018). https://doi.org/10.1007/s11042-017-4834-3

[2] Karanwal, Shekhar, and Manoj Diwakar. "Neighborhood and center difference-based-LBP for face recognition." Pattern Analysis and Applications 24.2 (2021): 741-761.

[3] Jin, Hongliang, et al. "Face detection using improved LBP under Bayesian framework." Third International Conference on Image and Graphics (ICIG'04). IEEE, 2004.

[4] Ren, Jianfeng, Xudong Jiang, and Junsong Yuan. "Quantized fuzzy LBP for face recognition." 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015.

[5] A. Satpathy, X. Jiang and H. Eng, "LBP-Based Edge-Texture Features for Object Recognition," in IEEE Transactions on Image Processing, vol. 23, no. 5, pp. 1953-1964, May 2014, doi: 10.1109/TIP.2014.2310123.

[6] Zeebaree, D. Qader, et al. "Multi-level fusion in ultrasound for cancer detection based on uniform LBP features." Computers, Materials & Continua 66.3 (2021): 3363-3382.

[7] A. Gunay and V. V. Nabiyev, "Automatic age classification with LBP," 2008 23rd International Symposium on Computer and Information Sciences, 2008, pp. 1-4, doi: 10.1109/ISCIS.2008.4717926.

[8] Rao, B.A., Kini, G.N., Aithal, P.K., Vaishnavi, K., Kamath, U.N. (2022). Parallel Local Tridirectional Feature Extraction Using GPU. In: Dua, M., Jain, A.K., Yadav, A., Kumar, N., Siarry, P. (eds) Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-16-5747-4_37

[9] Rao, B.A., Kini, N.G. (2021). Parallelization of Local Diagonal Extrema Pattern Using a Graphical Processing Unit and Its Optimization. In: Singh, V.K., Sergeyev, Y.D., Fischer, A. (eds) Recent Trends in Mathematical Modeling and High Performance Computing. Trends in Mathematics. Birkhäuser, Cham. https://doi.org/10.1007/978-3-030-68281-1_20

[10] Abeg Kumar Jaiswal, Haider Banka,Local pattern transformation based feature extraction techniques for classification of epileptic EEG signals,Biomedical Signal Processing and Control,Volume 34,2017,Pages 81-92,ISSN 1746-8094,https://doi.org/10.1016/j.bspc.2017.01.005.

[11] Priya, S. Jeba, et al. "Local pattern transformation based feature extraction for recognition of Parkinson's disease based on gait signals." Diagnostics 11.8 (2021): 1395.

[12] W. Chen and W. Li, "Definition and Usage of Texture Feature for Biological Sequence," in IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 18, no. 2, pp. 773-776, 1 March-April 2021, doi: 10.1109/TCBB.2020.2973084.