ICIC Express Letters
Volume **13**, Number **9**, September **2019**

 ISSN 1881-803X
pp. **867–874**

$\theta(1)$ TIME COMPLEXITY PARALLEL LOCAL BINARY PATTERN FEATURE EXTRACTOR ON A GRAPHICAL PROCESSING UNIT

Ashwath Rao Badanidiyoor and Gopalakrishna Kini Naravi

Department of Computer Science and Engineering
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal, Karnataka 576104, India
{ ashwath.rao; ng.kini }@manipal.edu

Received February 2019; accepted May 2019

Abstract. *Local Binary Pattern (LBP) feature is used widely as a texture feature in object recognition, face recognition, real-time recognitions, etc. in an image. LBP feature extraction time is crucial in many real-time applications. To accelerate feature extraction time, in many previous works researchers have used CPU-GPU combination. In this work, we provide a θ(1) time complexity implementation for determining the Local Binary Pattern (LBP) features of an image. This is possible by employing the full capability of a GPU. The implementation is tested on LISS medical images.*
Keywords: Local binary pattern, Medical image processing, Parallel algorithms, Graphical processing unit, CUDA

1. **Introduction.** Local binary pattern is a visual descriptor proposed in 1990 by Wang and He [1, 2]. Local binary pattern provides a distribution of intensity around a center pixel. It is a non-parametric visual descriptor helpful in describing a distribution around a center value. Since digital images are distributions of intensity, it is helpful in describing an image. The pattern is widely used in texture analysis, object recognition, and image description. The local binary pattern is used widely in real-time description, and analysis of objects in images and videos, owing to its computational efficiency and computational simplicity. It is a high performing feature in classifying objects. LBP is used in many domains namely medical image analysis and understanding, object recognition, biometrics, content-based image retrieval, remote sensing, industrial inspection, and document classification. In addition, LBP has been successfully applied in application areas such as dynamic texture recognition, fingerprint matching, visual inspection, image retrieval, and biomedical image analysis, facial image analysis, motion analysis, edge detection, and environment modeling. LBP is very useful because of its ease of implementation, invariance to monotonic changes and low computational complexity.

The local binary pattern feature is computed by finding the relative difference between a pixel and its eight neighbors. A code of 1 is assigned if the surrounding pixel is greater than the center pixel and zero if the center pixel is lesser than its neighbor. 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.

The detailed step is depicted in Figure 1. In a 3×3 image block, depending on the intensity of neighbors when compared with the center pixel intensity of the block, a 1 or 0 is coded.

DOI: 10.24507/icicel.13.09.867

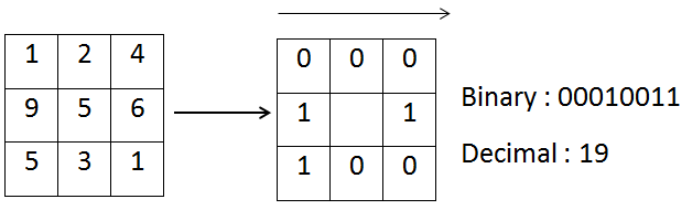

FIGURE 1. Local binary pattern feature computation

Since each center pixel is having eight neighbors, we get an 8-bit code of 1s and 0s. This eight-bit code is the LBP feature for the block. The LBP of the whole image is computed by applying the LBP operator on each 3×3 blocks surrounding each non-border pixel.

There are many variants of LBP. In general, the block size need not always be 3×3. It can be extended to any distance P along the radius R. With the positions of center pixel at (x_c, y_c), the sampling points $P(x_p, y_p)$ on a circle having radius R at a distance P is given by

$$\begin{aligned} x_p &= x_c + R\cos\left(\frac{2\pi p}{P}\right) \\ y_p &= y_c + R\sin\left(\frac{2\pi p}{P}\right) \end{aligned} \tag{1}$$

$$LBP_{r,p}(x_c) = \sum_{n=0}^{p-1} s(x_{r,p,n} - x_c)2^n \tag{2}$$

where

$$\begin{cases} s(x) = 1 & \text{if } x >= 0 \\ s(x) = 0 & \text{if } x < 0 \end{cases} \tag{3}$$

The LBP based feature has been used in diverse application areas. The application areas include face detection [3], face identification [4, 5, 6, 7, 8], facial expression recognition [9], facial micro-expression recognition [10], and facial emotion recognition [11]. In [3] LBP based face detection has been implemented on a GPU. Illumination invariance is achieved by Gamma correction and difference of Gaussian. Parallelization on a GPU is done using OpenCL. The researchers obtained a speedup of 5.45. LBP being a powerful descriptor is used in face identification. This is evident in [4]. A combination of LBP and discrete cosine transform on the grids of an image for face identification is developed [5]. Face verification of facial images with aging is studied in [6]. In this work, Adaboost has been used in enhancing the recognition rate. A parallel implementation of face identification on GPU using OpenCL is carried out in [7]. Parallel implementation shows a speedup of 30 when compared with the corresponding CPU based sequential implementation. Recognition of facial expression using LBP feature is carried out in [9]. A variant of LBP namely Volume-LBP and LBP on three orthogonal planes is used in facial micro-expression recognition [10]. A total of LBP and its 21 variants are assessed for their accuracy in facial emotion recognition [11].

LBP has been used in texture classification [12], texture analysis [13], peri-ocular biometric identification [14] and pedestrian detection in autonomous driving [15]. Variants of LBP namely complete LBP and multi-scale LBP are used in real-time texture classification on a GPU [12]. A speedup between 14 to 18 is obtained in this work. In another work, a speedup between 14 to 18 is obtained in real-time texture analysis on a GPU [13]. Peri-ocular biometric identification [14] is another task in which LBP based features are used. Parallelized human detection using LBP is carried out in [16]. A speedup of about 10 is obtained in this work. A speedup between 8 and 60 is obtained in pedestrian

detection for autonomous driving [15]. LBP and Histogram of Gradients (HOG) are used as the features.

LBP has been used in brain tumor detection from 3D MRI images [17]. In this work LBP on three orthognal planes has been calculated. In another work, LBP is used in the classification of breast cancer in mammography images [18].

In this paper, we present an efficient $\theta(1)$ time complexity extraction method for local binary pattern from an image. We have tested our technique on publicly available medical images from LISS database. This work will help reduce the time in extracting LBP features.

There have been a few parallel implementations of LBP feature extraction. Using SIMD instructions in the SSE extension of CPU a speedup of 6.5 is achieved. LBP feature extraction has been parallelized on a GPU [3, 7, 12, 15]. OpenCL and CUDA have been used as the language for implementation. The highest speedup of 60 is obtained [15].

The outline of this paper is as follows. In Section 2, the methodology is presented. In Section 3, the experimental setup and results are written. In Section 4, the conclusion and future scope are provided.

2. **Methodology.** Over the last two decades, graphical processing units have transformed from only graphics processors to support general purpose computing along with CPUs. This has been due to high power consumption in CPUs when the frequency is increased beyond a certain point. There has been a phenomenal growth in the memory size of GPUs. In the GPU industry, Nvidia and AMD are the major manufacturers of GPUs. Table 1 shows the memory size of GPUs with each release starting from beginning releases to till date [19]. Table 2 shows the memory sizes of AMD GPUs [20].

Table 1. Nvidia GPU models, memory size

Sl. No.	Desktop GPU	Year of first release	Min memory size in MB	Max memory size in MB
1	Pre-GeForce	1995	2	32
2	GeForce 256 series	1999	32	64
3	GeForce 2 series	2001	32	64
4	GeForce 3 series	2001	64	128
5	GeForce 4 series	2002	64	128
6	GeForce FX series	2003	64	256
7	GeForce 6 series	2005	128	512
8	GeForce 7 series	2007	128	1024
9	GeForce 8 series	2008	128	1024
10	GeForce 9 series	2008	256	1024
11	GeForce 100 series	2009	512	1024
12	GeForce 200 series	2009	512	2048
13	GeForce 300 series	2009	512	2048
14	GeForce 400 series	2011	512	2048
15	GeForce 500 series	2011	1024	3072
16	GeForce 600 series	2012	512	4096
17	GeForce 700 series	2014	512	12288
18	GeForce 900 series	2016	2048	12288
19	GeForce 10 series	2018	2048	12288
20	Volta series	2017	12288	32768
21	GeForce 20 series	2018	8192	11264

TABLE 2. AMD GPU models, memory size

Sl. No.	Desktop GPU	Year of first release	Min memory size in MB	Max memory size in MB
1	Wonder series	1986	0.0625	1
2	Mach series	1991	0.5	2
3	Rage series	1996	2	64
4	Radeon 7000 series	2001	32	128
5	Radeon 8000 & 9000 series	2001	64	256
6	Radeon R300 series	2002	64	256
7	Radeon X700 & X800 series	2005	128	256
8	Radeon X1000 series	2005	128	512
9	Radeon HD 2000 series	2007	256	1024
10	Radeon HD 3000 series	2009	256	1024
11	All-in-Wonder series	2000	32	512
12	Radeon HD 4000 series	2008	256	2048
13	Radeon HD 5000 series	2010	512	4096
14	Radeon HD 6000 series	2011	512	4096
15	Radeon HD 7000 series	2012	256	6144
16	Radeon HD 8000 series	2013	256	6144
17	Radeon R5/R7/R9 200 series	2013	256	8196
18	Radeon R5/R7/R9 300 series	2015	1024	8196
19	Radeon RX 400 series	2016	1024	8196
20	Radeon RX 500 series	2017	1024	8196
21	Radeon RX Vega series	2017	8196	8196

TABLE 3. Total LBP feature vector size

Image size	LBP feature vector size in bytes	Size in KB
256×256	64516	63
512×512	260100	254
1024×1024	1044484	1020
2048×2048	4186116	4088

As is evident in the above two table values, there has been a steady increase in memory size with each release of GPUs. So algorithm designers need to encash on this trend. There have not been major changes in the methods or techniques in the computational tasks involving the GPUs over the years. Most of the methods of LBP computation involve dividing the image into regions and computing the LBP on the regions and finally aggregating the LBPs from each region [3, 8, 12]. So in this work, we introduce a novel technique by which we make use of the large memory size available and we show that this has a direct effect on reducing computation time.

We propose a novel method where LBP is computed on the whole image at once. We create as many threads as the number of non-border elements in the image. Assume LBP has to be computed for an image of size $H \times W$. Then the image contains $(H-2) \times (W-2)$ non-border pixels. The LBP feature vector size becomes,

$$size\_LBP(H, W) = (H - 2) \times (W - 2) \text{ bytes} \tag{4}$$

We provide the feature vector size in Table 3.

Many of the medical images include the images in the LISS database, and the resolution varies between 512 and 1024. Even if the image is as large as 2048×2048, the memory

required to store LBP is only about 4 MB. Since all the present day GPUs have memory in the range of 8 GB, this is a practical and viable option.

3. **Experimental Setup and Results.** We have considered the publicly available LISS database [21] in our experiment. The images in the database are from CT modality and contain common imaging signs of lung diseases. We have used the DCMTK library for processing medical images [22].

We have written parallel LBP feature extraction using a GPU on a supercomputer. The supercomputer is powered by Intel(R) Xeon(R) CPU E5-2670 V3@2.30 GHz with 24 CPUs. The GPU is Tesla K40 with 2880 cores and 288 GB/s memory bandwidth with total board memory 12 GB.

We have used CUDA for our implementation as CUDA implementation has shown better results than OpenCL [23, 24]. The LBP computation is fastest if all the computation is performed at once by all threads. This is achieved by setting the block size equal to size of the non-border pixels in the image. Since Tesla K-40 GPU supports a maximum thread block size as $(1024, 1024, 64)$ we have made the block size equal to the number of non-border pixels along each axis. Hence, we are able to obtain the highest speedup in LBP computation.

3.1. **Results.** The sequential LBP computation with varying medical images is shown in Table 4. As shown in the table, the sequential execution time increases with the image size. We consider this time in speedup calculation.

Table 4. LBP sequential program execution time

Image size	LBP function execution time (ms)	End-to-end execution time (ms)
256×256	6.524992	22.577024
512×512	43.448418	63.505791
1024×1024	98.803490	119.640221

With the above settings, we have computed LBP parallely using a GPU. The parallel execution time is as shown in Table 5.

Table 5. LBP parallel program execution time

Image size	Thread block size	Memory transfer in time (ms)	Kernel execution time (ms)	Memory transfer out time (ms)	End-to-end execution time (ms)
256×256	16×16	0.665376	0.044672	0.044064	17.802752
	32×32	0.663712	0.044000	0.043328	17.845888
	1024×1024	0.668704	0.013376	0.044160	17.420704
512×512	16×16	1.198976	0.158048	0.480896	22.597530
	32×32	1.198496	0.127872	0.488544	22.333183
	1024×1024	1.202080	0.013920	0.487872	21.828608
1024×1024	16×16	0.949568	0.181760	0.692352	18.060896
	32×32	1.177312	0.156800	0.844896	23.542688
	1024×1024	1.178080	0.012224	0.842208	22.963743

The speedup of LBP feature extraction considering kernel execution time is shown in Table 6. As the image size increases by 256×256 to 512×512 a four-fold increase or 256×256 to 1024×1024 a sixteen-fold increase, sequential execution time increases linearly to the increase in number of pixels. However, in the parallel execution using GPU, either for a four-fold increase in the case of 256×256 to 512×512 or 256×256 to 1024×1024 a 16-fold increase in pixels, there is no change in kernel execution time.

TABLE 6. Speedup of LBP feature extraction considering kernel execution time

Image size	Thread block size	Speedup
256 × 256	16 × 16	146.0644
	32 × 32	148.2952
	1024 × 1024	487.8133
512 × 512	16 × 16	274.9060
	32 × 32	339.7805
	1024 × 1024	3121.2943
1024 × 1024	16 × 16	543.5931
	32 × 32	630.1242
	1024 × 1024	8082.7462

Hence, the kernel execution time remains constant. The time complexity is $\theta(1)$. This has been possible because in parallel execution using GPU, the utilization of a number of threads equals the number of non-border pixels in the image and hence the computation time remains constant. The highest speedup of 8082 is obtained with an image size of 1024×1024. This is the highest speedup achieved for LBP calculation in the literature. Hence, we are able to achieve $\theta(1)$ computation time in LBP feature extraction.

3.2. **Comparison with other parallel approaches.** There are many works on parallel LBP feature extraction using GPUs. The comparison of our approach with others is shown in Table 7. Few works are available on parallel LBP which has LBP feature extraction as part of other works and they do not mention the feature extraction time for LBP. Other works are in literature where they extract LBP features but they do not mention the LBP extraction time. Hence, such papers cannot be considered in comparison.

TABLE 7. Comparison of this work with other parallel works

Sl. No.	Paper #1	Paper #2	Paper #3	Our work
Authors, Year	C. Y. N. Dwith and G. N. Rathna, 2013 [12]	C. Y. N. Dwith and G. N. Rathna, 2012 [7]	N. Naik and G. N. Rathna, 2014 [3]	
End-to-end execution speedup	4.7228	–	5.45	6.22
Kernel speedup with 1024×1024 image	–	34.0757	–	8082.7462

As shown in Table 7, our approach is better than other similar works. With each new GPU release, the speed increases, owing to higher clock frequency. Hence, speed is not the only criterion for comparison. It is to be noted that, in LBP extraction, by making thread block size equal to the number of non-border elements in the image, the computation time becomes $\theta(1)$. This is the highest possible kernel execution time. Also, now GPUs can accommodate the LBP feature entirely in the global memory. Hence, splitting the image into sub-parts and then computing the LBP feature is not necessary.

4. **Conclusion and Future Scope.** The capabilities of graphical processing units have increased in memory size, the number of threads and speed. It is necessary to adapt algorithms to this changing trend. We have shown a $\theta(1)$ time complexity method for extracting local binary pattern features from a single medical image. As local binary

pattern feature is used in real-time applications, our work will help reduce the feature extraction time and thereby reduce response time.

There are many applications from various domains, which involves data parallel computations. All such computations can be carried out faster by utilizing the full capability of GPU. The initial techniques of repeated computation on each partition of data and then aggregating the results can be changed to obtain less time complexity.

REFERENCES

[1] L. Wang and D.-C. He, Texture classification using texture spectrum, *Pattern Recognition*, vol.23, no.8, pp.905-910, 1990.

[2] L. Wang, Texture unit, textural spectrum and texture analysis, *IEEE Trans. Geoscience and Remote Sensing*, vol.28, no.4, p.509, 1990.

[3] N. Naik and G. N. Rathna, Real time face detection on GPU using OpenCL, *Computer Science & Information Technology (CS & IT)*, pp.441-448, 2014.

[4] I. L. K. Beli and C. Guo, Enhancing face identification using local binary patterns and k-nearest neighbors, *Journal of Imaging*, vol.3, no.3, pp.1-12, 2017.

[5] Y. Zhang, T. Chai and C.-C. Hung, Local binary patterns for face recognition under varying variations, *Proc. of the 6th Annual Workshop on Cyber Security and Information Intelligence Research*, p.39, 2010.

[6] G. Mahalingam and C. Kambhamettu, Face verification with aging using AdaBoost and local binary patterns, *Proc. of the 7th Indian Conference on Computer Vision, Graphics and Image Processing – ICVGIP'10*, pp.101-108, 2010.

[7] C. Y. N. Dwith and G. N. Rathna, Parallel implementation of LBP based face recognition on GPU using OpenCL, *The 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp.755-760, 2012.

[8] S. C. Tek and M. Gökmen, CUDA accelerated face recognition using local binary patterns, *Threshold*, no.2, p.169, 2012.

[9] G. Zhao and M. Pietikainen, Dynamic texture recognition using local binary patterns with an application to facial expressions, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.29, no.6, pp.915-928, 2007.

[10] Y. Wang, J. See, R. C. W. Phan and Y. H. Oh, Efficient spatio-temporal local binary patterns for spontaneous facial micro-expression recognition, *PLoS ONE*, vol.10, no.5, pp.1-20, 2015.

[11] K. Slimani, M. Kas, Y. El. Merabet, R. Messoussi and Y. Ruichek, Facial emotion recognition: A comparative analysis using 22 LBP variants, *ACM International Conference Proceeding Series*, pp.88-94, 2018.

[12] C. Y. N. Dwith and G. N. Rathna, Parallel computing for accelerated texture classification with local binary pattern descriptors using OpenCL, *International Journal of Computer Applications*, vol.64, no.1, 2013.

[13] G. Zolynski, T. Braun and K. Berns, Local binary pattern based texture analysis in real time using a graphics processing unit, *VDIBERICHT*, p.321, 2008.

[14] J. Xu, M. Cha, J. L. Heyman, S. Venugopalan, R. Abiantun and M. Savvides, Robust local binary pattern feature sets for periocular biometric identification, *The 4th IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pp.1-8, 2010.

[15] V. Campmany, S. Silva, A. Espinosa, J. C. Moure, D. Vázquez and A. M. López, GPU-based pedestrian detection for autonomous driving, *Procedia Computer Science*, vol.80, pp.2377-2381, 2016.

[16] M. Farhadi, S. A. Motamedi and S. Sharifian, Efficient human detection based on parallel implementation of gradient and texture feature extraction methods, *The 7th Iranian Conference on Machine Vision and Image Processing*, pp.1-5, 2011.

[17] S. Abbasi and F. Tajeripour, Detection of brain tumor in 3D MRI images using local binary patterns and histogram orientation gradient, *Neurocomputing*, vol.219, pp.526-535, 2017.

[18] E. T. Pereira, S. P. Eleutério and J. M. Carvalho, Local binary patterns applied to breast cancer classification in mammographies, *Revista de Informática Teórica e Aplicada*, vol.21, no.2, pp.32-46, 2014.

[19] *List of NVIDIA Graphics Processing Units*, https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units, accessed on 19th September, 2018.

[20] *List of AMD Graphics Processing Unit*, https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units, accessed on 19th September, 2018.

[21] G. Han, X. Liu, F. Han, I. N. T. Santika, Y. Zhao, X. Zhao and C. Zhou, The LISS – A public database of common imaging signs of lung diseases for computer-aided detection and diagnosis research and medical education, *IEEE Trans. Biomedical Engineering*, vol.62, no.2, pp.648-656, 2015.
[22] Kuratorium OFFIS, *DCMTK: DICOM-toolkit*, 2005.
[23] J. Fang, A. L. Varbanescu and H. Sips, A comprehensive performance comparison of CUDA and OpenCL, *The International Conference on Parallel Processing*, pp.216-225, 2011.
[24] K. Karimi, N. G. Dickson and F. Hamze, A performance comparison of CUDA and OpenCL, *arXiv Preprint arXiv: 1005.2581*, 2010.