Texture Classification Using Local Binary Patterns on Modern Graphics Hardware

Joshua M. Leibstein, Andras P. Findt and Andre L. Nel
Department of Electrical and Electronic Engineering Science
University of Johannesburg, P.O. Box 524, Auckland Park, 2006
Tel: +27 83 6911172, Fax: +27 11 6723693
email: joshliebe@gmail.com; andren@uj.ac.za

Abstract- We explore a method for texture classification using the GPU. Tests on various Local Binary Pattern (LBP) algorithms were performed. These algorithms were implemented and parallelised for use on a GPU. Vast improvements in computation speed were achieved over traditional CPU-based algorithms. The focus of this paper is to provide a brief overview of these algorithms and to demonstrate the observed performance gains.

Index Terms— Local Binary Patterns, Algorithms, Texture Analysis, GPU.

I. INTRODUCTION

The CPU performs certain operations, especially image operations, in a linear order. GPUs have addressed this problem by allocating mathematical operations to multiple threads, in parallel. The intrinsic power of this processor has resulted in an exciting development called GPGPU (General Purpose on the GPU) programming, which allows a GPU to solve computing problems that are typically performed by a CPU. Our research is aimed at producing GPU-oriented algorithms for use in texture classification applications. In this paper we shall elaborate on how texture classification algorithms can be parallelised and processed by using image calculations on modern graphics hardware. The implementations of these texture analysis algorithms were initially prototyped in MATLAB before being rewritten and executed as GPU shaders. For performance evaluation purposes on the CPU, the algorithms were ported from MATLAB to C++.

II. OVERVIEW OF IMPLEMENTED ALGORITHMS

The LBP algorithm is among the most efficient descriptors used in texture analysis today. When compared to more sophisticated texture analysis algorithms, it is found that LBP descriptor is both computationally inexpensive and easily parallelisable. Although the LBP is already an efficient CPU algorithm, its performance is further enhanced by taking advantage of the additional computational power of graphics processing unit. We will now discuss the algorithms we have implemented for use in texture classification.

A. Local Binary Patterns

The LBP is an operator that was first introduced by Ojala *et al*. [1] and has been shown to be an effective descriptor in texture classification [2]. To create an LBP representation an input texture image must first be converted to greyscale before this operator is applied to each individual pixel within the image. A feature vector describing the textural properties of the image is then obtained from a histogram of the LBP values of the image. This feature vector acts as a "signature" for the texture to be used in texture classification. Figure 1 shows an example of a 3x3 LBP and resulting description value of pixel c = 11100001 = 225.

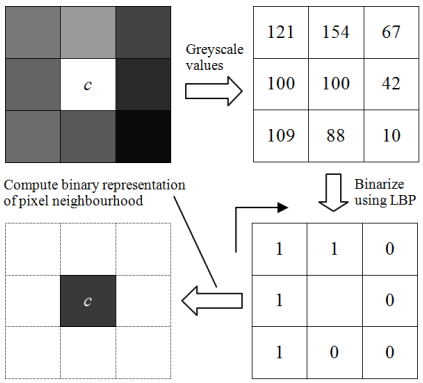

Figure 1: Process overview of a 3x3 LBP

As seen in Figure 2, it is observed that a mathematical explanation for this operation is described in accompanying equations (1) and (2).

t_1	t_2	t_3
t_0	c	t_4
t_7	t_6	t_5

$(t_0 = t_8)$

Figure 2: A 3x3 pixel neighbourhood

$$b_k = \begin{cases} 1, & \sum_{k=0}^{k=7} t_k \geq c \\ 0, & \sum_{k=0}^{k=7} t_k < c \end{cases} \qquad k \in \{0,7\} \tag{1}$$

$$c = \sum_{k=0}^{k=7} 2^k b_k \tag{2}$$

B. Multi-Block Local Binary Patterns

The MB-LBP (Multi-Block Local Binary Pattern) texture descriptor is an extension of the original LBP as proposed by Zhang, Chu, Xiang, Liao and Li [3]. MB-LBP are more robust than the original LBP descriptor as it can encode microstructures as well as macrostructures. For certain applications such as face recognition, experimental results indicate that MB-LBP out-perform other LBP algorithms [4]. The calculation of an MB-LBP is similar to a standard LBP except that in a MB-LBP t_0 to t_7 (Figure 3) are the average grey values of the pixels in each corresponding region. These regions are compared to the averaged central region. Each averaged region is of equal size but does not necessarily have to be square. In our evaluation 9x9 and 15x15 MB-LBP algorithms were used and tested.

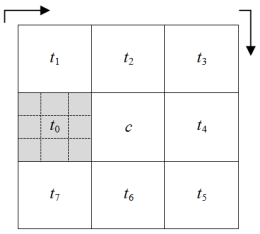

Figure 3: An example of a 9x9 MB-LBP

C. *Radial Local Binary Patterns*

We propose a new algorithm based on the ideas of Zolynski, Braun and Berns [5]. The suggested GPU implementation is a radial Local Binary Pattern (We will refer to it as the RLBP) and it uses largely the same principles as the standard LBP described above. The key difference is that the circular approach, as seen in Figure 4, uses a number of interpolated pixel values equally spaced along the circumference of a circle around each pixel to calculate the binary value. This provides the textural properties of an image at different scales resulting in a multiresolution representation. The number of points sampled is determined by the radius of the circle. For a circle with a 1 pixel radius, a total number of 8 sampled points are taken. The following rule is applied regarding the number of points:

$$p = 8 + floor\left(\frac{r-1}{1.5}\right) \times 4 \qquad (3)$$

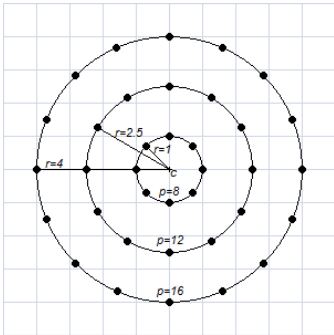

Figure 4: Circular pixel neighbourhoods

The algorithm we propose, the new RMB-LBP (Radial Multi-Block Local Binary Pattern) follows the same rules as the MB-LBP but as in the RLBP, it takes advantage of the low computational cost of interpolating the points of the circle on a GPU.

III. Experimental Setup

Experiments were carried out on a Windows 7 x64 system with the following hardware: 3GHz Athlon II X2 250 processor, 4GB RAM and a GeForce GTX 260 graphics card. The average execution time of each algorithm was recorded for 20 iterations. This was carried out for both CPU and GPU implementations. A 1024 x 1024 image was used to evaluate the algorithms.

IV. Preliminary Results

Tests indicated an improved execution time when processing was performed on the GPU as opposed to the CPU. As the algorithmic complexity increased the GPU did not suffer from the severe performance decrease exhibited by the CPU, as seen in Table 1. It is also seen that the RLBP reveals a similar performance to the GPU implemented LBP, whereas the RMB-LBP demonstrates the best performance of all implemented MB-LBP algorithms.

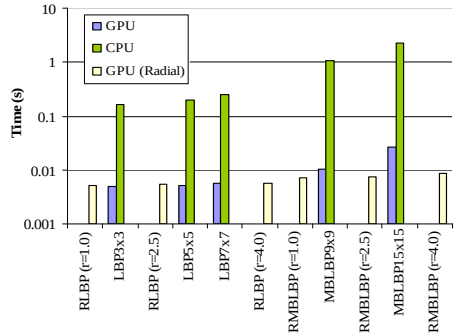

Table 1: GPU versus CPU

For the non-radial algorithms the GPU outperforms the CPU by a factor of between 30 and 100 when processing the same algorithms, as seen in Table 2. (The radial algorithms would be much slower due to the interpolation step required for the CPU implementation.)

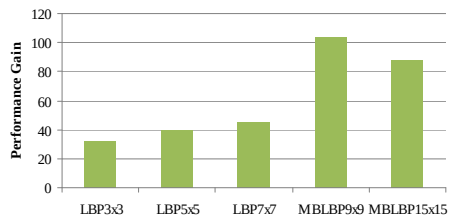

Table 2: GPU Performance

V. Conclusion

A GPU implementation of the rectangular LBPs outperforms the CPU version by at least 30 times while the presented RMB-LBP algorithm significantly out-performs other MB-LBP algorithms at increased scale factors. The GPU performance gains asymptote to roughly 5 for the radial versus rectangular forms of the MB-LBP.

References

[1] T. Ojala, M. Pietikäinen, and D. Harwood, "A Comparative Study of Texture Measures with Classification Based on Feature Distributions," *Pattern Recognition*, vol. 29, 1996, pp. 51-59.

[2] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, 2002, pp. 971-987.

[3] L. Zhang, R. Chu, S. Xiang, S. Liao, and S.Z. Li, "Face Detection Based on Multi-Block LBP Representation," *IAPR/IEEE International Conference on Biometrics*, 2007.

[4] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S.Z. Li, "Learning Multi-scale Block Local Binary Patterns for Face Recognition," *IAPR/IEEE International Conference on Biometrics*, 2007, pp. 828-837.

[5] G. Zolynski, T. Braun and K. Berns, "Local Binary Pattern Based Texture Analysis in Real Time using a Graphics Processing Unit", in Proceedings of Robotik 2008, vol. 2012, June 2008, pp. 321–325

Joshua Leibstein is an undergraduate student at the University of Johannesburg and is presently studying towards his BSc Degree in Information Technology. His research interests include texture analysis, GPGPU programming, image moments and face recognition.