# ACCELERATING THE COMPUTATION OF GLCM AND HARALICK TEXTURE FEATURES ON RECONFIGURABLE HARDWARE

*M. A. Tahir, A. Bouridane, F. Kurugollu, and A. Amira*

School of Computer Science
The Queen's University of Befast
Belfast BT7 1NN, Northern Ireland
a.tahir@qub.ac.uk

## ABSTRACT

Grey Level Co-occurrence Matrix (GLCM), one of the best known tool for texture analysis, estimates image properties related to second-order statistics. These image properties commonly known as Haralick texture features can be used for image classification, image segmentation, and remote sensing applications. However, their computations are highly intensive especially for very large images such as medical ones. Therefore, methods to accelerate their computations are highly desired. This paper proposes the use of reconfigurable hardware to accelerate the calculation of GLCM and Haralick texture features. The performances of the proposed co-processor are then assessed and compared against a microprocessor based solution.

## 1. INTRODUCTION

Image processing applications usually require the processing of large amounts of data, especially after the introduction of multispectral and scanscope images [1, 2]. In multispectral images, instead of analysing conventional grey scale or RGB color images, multiple bands of an object are created using light from different parts of the spectrum [1]. Figure 1 shows two bands of a textured multispectral medical image. In scanscope images, for a $15mm * 15mm$ tissue section, approximately 2.5 gigabytes (GB) of data must be acquired. Much effort has therefore gone into designing high performance architectures tailored to image processing applications. This paper describes some results of a research programme which aims to develop an Field Programmable Gate Array (FPGA) co-processor for the classification of tissue patterns in prostatic cancer using multispectral medical images. Figure 2 shows the steps involved for prostate cancer classification [1]. The first step of the algorithm is to compute Grey Level Co-occurrence Matrix (GLCM) to extract Haralick texture features [3, 4, 5]. Step 2 involves the normalisation of GLCM. The third step of the algorithm is to compute texture features from normalised GLCM. Since

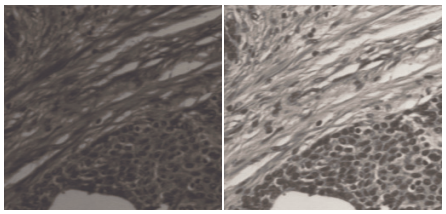

**Fig. 1**. Two bands of a multispectral image showing pathological section of a prostate biopsy.

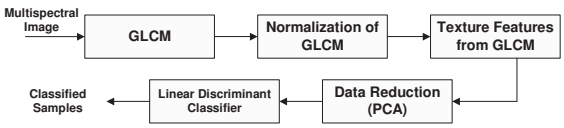


**Fig. 2**. Algorithm for the classification of prostate tissue cancer.

different regions in tissue section images can be classified as cancer or normal using texture features, therefore, these features are measured in a large number of sub-regions inside each band of the multispectral image. Thus, for each subregion, the computation of GLCM is required along with the computation of texture features. Step 4 and 5 involve the use of a classifier for classification [1].

The overall calculations for the computation of GLCM and texture features are computationally intensive. For an image of size 5000*5000 with 16 bands, the time required is 350 seconds using Pentium 4 machine running at 2400MHz. 75% of the total time spent is for the calculation of GLCM, 5% for the normalisation, 19% for the calculation of texture features while 1% is for the classification using linear discrimination. There are different images for each patient and even the number of patients can vary. Thus, overall timing requirements for the entire process are computationally intensive. The Von Neumann style of fetch-operate-writeback computation fails to exploit the inherent parallelism in computing these features. Therefore, a hardware intensive parallel implementation is needed.

The aim of this paper is to investigate the use of FP-

0-7803-8554-3/04/$20.00 ©2004 IEEE.

GAs to accelerate the computation of GLCM and texture features. The target hardware for this work is Celoxica RC1000-PP PCI based FPGA development board equipped with a Xilinx XCV2000E Virtex FPGA having 19,200 slices and 655,360 bits of block RAM, and four banks of static RAM with 2MB each [6, 7].

The paper is organised as follows. Section 2 reviews grey level co-occurrence matrix and texture features. Section 3 describes the proposed system model. Experiments and discussion are presented in Section 4. Section 5 concludes the paper.

## 2. GLCM AND HARALICK TEXTURE FEATURES

GLCM, first introduced by Haralick [3], is a powerful technique for measuring texture features. The texture-context information is specified by the matrix of relative frequencies $P_{i,j}$ with two neighboring pixels separated by displacement d and an angle $\theta$. The GLCM is calculated with the following equation [5];

$$P(i,j,d,\theta) = \#\{(x_1,y_1)(x_2,y_2) | f(x_1,y_1) = i, \\ f(x_2,y_2) = j, |(x_1,y_1) - (x_2,y_2)| = d, \\ \angle((x_1,y_1),(x_2,y_2)) = \theta\} \quad (1)$$

where # is the number of occurrences inside the window sizes where the intensity level of a pixel pair changes from $i$ to $j$, the location of the first pixel is $(x_1, y_1)$ and that of the second pixel is $(x_2, y_2)$, $d$ is the distance between the pixel pair, $\theta$ is the angle between the two pixels. The co-occurrence matrix so defined is not symmetric. If the GLCM is calculated with symmetry, then only angles up to $180^0$ need to be considered. A symmetric co-occurrence matrix can be computed by the expression $P(i,j,d,\theta)^{'}$ = $(P(i,j,d,\theta) + P(i,j,d,\theta)^T)$ / 2 where $P(i,j,d,\theta)^T$ is the transpose of $P(i,j,d,\theta)$. Probability estimates are obtained by dividing each entry in $P(i,j,d,\theta)^{'}$ by the sum of all possible intensity changes with the distance $d$ and direction $\theta$.

### 2.1. Haralick Texture Features

Let $P(i,j,d,\theta)$ is a (normalised) frequency of occurrence of grey level pair (i,j) at distance $d$ and angle $\theta$ and $N_g$ be the number of gray levels.

$$f_1(d,\theta) = ASM = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P(i,j,d,\theta)^2 \quad (2)$$

$$f_2(d,\theta) = \mu = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} iP(i,j,d,\theta) \quad (3)$$

$$f_3(d,\theta) = \sigma = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}((i-\mu)^2 P(i,j,d,\theta))^{1/2} \quad (4)$$

$$f_{4(d,\theta)} = Corr = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\frac{(i-\mu)(j-\mu)P(i,j,d,\theta)^2}{\sigma^2} \quad (5)$$

$$f_{5(d,\theta)} = Cont = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-j)^2 P(i,j,d,\theta) \quad (6)$$

$$f_{6(d,\theta)} = Diss = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-j) P(i,j,d,\theta) \quad (7)$$

$$f_{7(d,\theta)} = Ent = -\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P(i,j,d,\theta)\log P(i,j,d,\theta) \quad (8)$$

where ASM, $\mu$, $\sigma$, Corr, Cont, Diss, Ent are angular second moment, mean, variance, correlation, contrast, dissimilarity, and entropy respectively.

## 3. PROPOSED SYSTEM MODEL

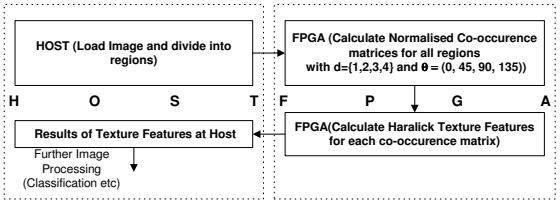


**Fig. 3**. System Model.

The proposed system model for the entire process as mentioned in Figure 2, is shown in Figure 3. At its most basic level, the programming model for our image processing machine is a host processor (typically a PC running at 2.4GHz Pentium 4-based system, programmed in C++). The host machine is working as a control unit that loads different inputs required for each stage in the external memory of the FPGA. In the first stage, the input to the FPGA consists of different bands of multispectral images. Each band *B* is divided into regions of size *N*N*. Experiments have shown that the choice of $N$ that equals to 128 exhibits the best trade off in terms of good localization and accurate measurements of texture features for the application at hand [1]. The input to the second stage are *B*R* co-occurrence matrices of size $N_g * N_g$ [1] with distance $d = \{1,2,3,4\}$ and $\theta = \{0^o, 45^o, 90^o, 135^o\}$ where $B$ is the number of bands in multispectral image and $R$ is the number of regions for each band.

### 3.1. Proposed Architecture for Calculating GLCM

The block diagram for calculating GLCMs on FPGA is shown in Figure 4. A novel architecture is proposed in which 16 co-occurrence matrices (d = {1,2,3,4} and $\theta = \{0, 45, 90, 135\}$

[1] In this case $N_g = 32$

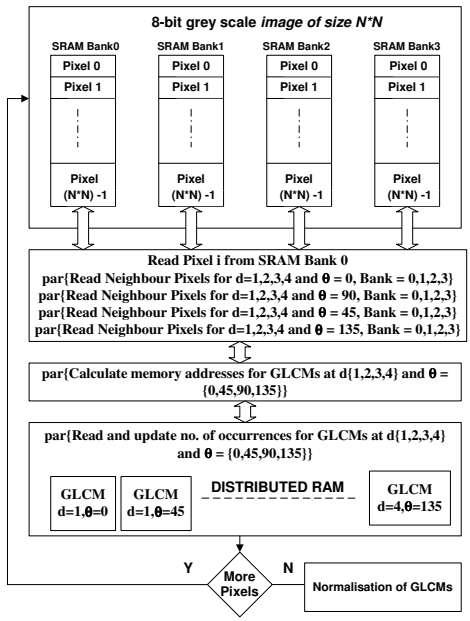


**Fig. 4**. Block diagram for calculating GLCM on FPGA.

i.e. 4*4 GLCMs) for a band *B* and region *R* are calculated in parallel. The input to the FPGA is the image region of size *N*N* loaded into memory bank 0 for band *B*. The first pixel *i* is read from memory bank 0. The same image region is duplicated into the other 3 banks of SRAM to access 4 neighbor pixels in parallel of pixel *i*. Thus, all 16 neighbor pixels are read in 4 clock cycles. The memory addresses for all 16 GLCMs are calculated in parallel followed by updating the number of occurrences of pixels in co-occurrence matrix. The process is repeated for each pixel of the entire image. For the application at hand, the normalisation of GLCMs is required. The flow diagram for the normalisation of GLCMs is shown in Figure 5. For simplicity, the normalisation of only one GLCM (for d=0 and $\theta$=0) is shown. All other GLCMs are also normalised simultaneously.

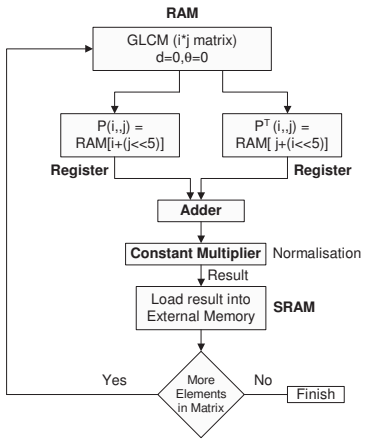


**Fig. 5**. Flow Chart for the normalisation of GLCMs on FPGA.

In order to normalize the GLCM, element $P(i,j,d,\theta)$ is first added with $P^T(i,j,d,\theta)$. The result is then divided by the expression (two times the sum of all possible intensity changes with distance $d$ and direction $\theta$) to compute the estimation of probability [5]. Since, the above expression is constant, a constant multiplier is used instead of divider by taking the reciprocal of the expression. The final result is stored in the external memory. If there are more elements in GLCM, the process is repeated. Other GLCMs are also normalised simultaneously. Thus, all GLCMs are calculated and normalised in parallel. The process of calculating and normalising GLCM is then looped for different bands and regions of a multispectral image.

### 3.2. Proposed Architecture for Haralick Texture Features Computation

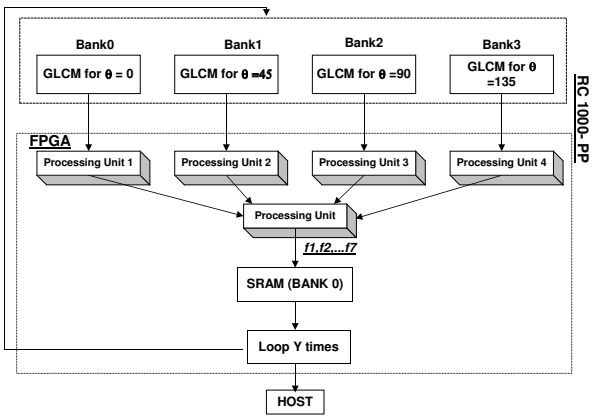


**Fig. 6**. Block diagram for extracting Haralick features in FPGA.

The block diagram of the proposed architecture for extracting texture features on FPGA is shown in Figure 6. Each processing unit consists of adders and multipliers as shown in Figure 7 and is used to calculate the features described in equations 2-8 for a particular angle $\theta$. Each processing unit operate in parallel. The final processing unit, which consists of adders and shift registers, is used to calculate the average of each feature computed at distance $d$ for different $\theta$. This will result in 7 features mentioned in section 2 at distance $d$ for region $R$ and band $B$. These features are stored in SRAM (Bank 0). This process is looped $Y$ times where $Y = 4 * R * B$.

Figure 7 shows the block diagram of the processing unit indicated in Figure 6. There are two stages in this block diagram. Mean, contrast, dissimilarity, and entropy are calculated in the first stage. Four Processing Elements (PEs) are used for the calculation. These PEs, consisting of parallel multipliers and adders, operate simultaneously. In order to speed up the computation, the constants $(i-j)$ and $(i-j)^2$ of equations 6 and 7 are pre-computed and stored in ROMS.

Log tables, which are stored in Block RAMS, are used for the calculation of log function involved in the entropy computation. Angular second moment, variance and correlation are calculated in the second stage as variance and correlation depend upon the mean value.

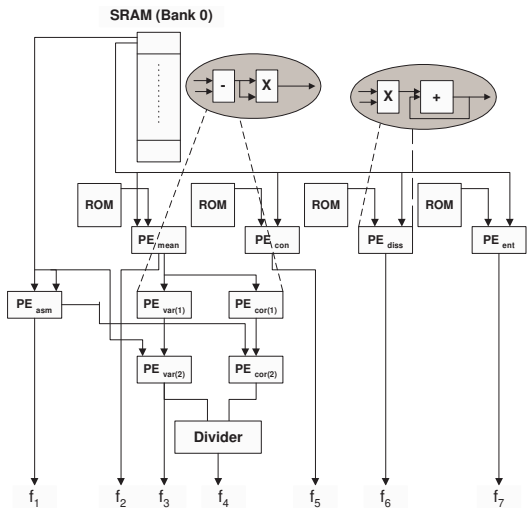


**Fig. 7**. Block diagram of Processing Unit 1 as shown in Figure 6. The other processing units 2-4 have the same architecture and are executed in parallel.

The proposed architecture has been implemented by using Handel-C [6]. Handel-C is a truly innovative C-like language for implementing algorithms in hardware.The output from Handel-C is a file that is used to create the configuration data for the FPGA.

## 4. EXPERIMENTS AND DISCUSSION

The original testing material consists of textured multispectral images taken at 16 spectral channels (from 500nm to 650nm) [1]. Each of the original images is divided into a region of size $N * N$. Fixed point number representation is used for the implementation.

Table 1 shows the execution time comparison between $\mu$P-based and FPGA-based implementation for various image sizes for the calculation of GLCM. The clock speed of the FPGA architecture for calculating GLCM is 50MHz and the area used (% of slices) is 59% while the clock speed for Pentium 4 is 2400MHz. The result shows that the performance of FPGA is approximately 5 times faster than that of Pentium 4 PC even though the PC platform has a clock speed which is 45 times faster. This performance improvement is mainly due to the calculation of different GLCMs in parallel.

Table 2 shows the execution time comparison between $\mu$P-based and FPGA-based implementation for various image sizes for the calculation of Haralick features. The result shows that the performance of FPGA is approximately 7 times faster than Pentium 4 PC. This is mainly due to the parallel architecture of FPGA that results in more multiplications and additions on every clock cycle than Pentium 4. The clock speed of this FPGA architecture is 45MHz and the area used (% of slices) is 85%.

**Table 1**. Execution Time in sec between $\mu$-P and FPGA implementation for calculating GLCM.

| Image Size (16 Bands) | $\mu$-P | FPGA | Speed-Up |
|---|---|---|---|
| 512*512 | 3.0 | 0.63 | 4.75 |
| 1024*1024 | 11.9 | 2.5 | 4.75 |
| 2048*2048 | 47.6 | 10.0 | 4.75 |

**Table 2**. Execution Time in sec between $\mu$-P and FPGA implementation for calculating Haralick Texture Features.

| Image Size | $\mu$-P based | FPGA based | Speed-Up |
|---|---|---|---|
| 512*512 | 0.710 | 0.097 | 7.3 |
| 1024*1024 | 2.840 | 0.388 | 7.3 |
| 2048*2048 | 11.360 | 1.550 | 7.3 |

## 5. CONCLUSION

Reconfigurable devices are useful in many application intensive problems such as image processing applications. This paper has described the use of FPGA as a co-processor to accelerate the computation of GLCMs and Haralick texture features. Efficient architecture is proposed for their implementation. Results show that the FPGA has superior speed performances when compared with a general-purpose processor for the computation of GLCM and Haralick texture features.

## 6. REFERENCES


[1] M. A. Roula et al., "A Multispectral Computer Vision System for Automatic Grading of Prostatic Neoplasia", *IEEE International Symposium on Biomedical Imaging*, 2002.

[2] URL: *http://www.scanscope.com*

[3] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification", *IEEE Trans. on Systems, Man, and Cybernetics*, 3(6), pp. 610-621, 1973.

[4] R.W. Conners and C.A. Harlow, "A Theoretical Comparison of Texture Algorithms", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(3), pp. 204-222, 1980.

[5] K. Wikantika, A.B. Harto, and R. Tateishi, "The use of spectral and textural features from Landsat TM image for land cover classification in mountainous area", *Proceedings of the IECL Japan workshop*, Tokyo, 2001.

[6] URL: *http://www.celoxica.com.*

[7] URL: *http://www.xilinx.com.*