Dedicated Hardware for Real-Time Computation of Second-Order Statistical Features for High Resolution Images

Dimitris Bariamis, Dimitris K. Iakovidis, and Dimitris Maroulis

Dept. of Informatics and Telecommunications, University of Athens,
Panepistimiopolis, Illisia, 15784 Athens, Greece
rtsimage@di.uoa.gr

Abstract. We present a novel dedicated hardware system for the extraction of second-order statistical features from high-resolution images. The selected features are based on gray level co-occurrence matrix analysis and are angular second moment, correlation, inverse difference moment and entropy. The proposed system was evaluated using input images with resolutions that range from 512×512 to 2048×2048 pixels. Each image is divided into blocks of user-defined size and a feature vector is extracted for each block. The system is implemented on a Xilinx VirtexE-2000 FPGA and uses integer arithmetic, a sparse co-occurrence matrix representation and a fast logarithm approximation to improve efficiency. It allows the parallel calculation of sixteen co-occurrence matrices and four feature vectors on the same FPGA core. The experimental results illustrate the feasibility of real-time feature extraction for input images of dimensions up to 2048×2048 pixels, where a performance of 32 images per second is achieved.

1 Introduction

The second-order statistical information present in an image relates to the human perception of texture. It has been successfully utilized in a variety of machine vision systems, including biomedical [1,2], remote sensing [3], quality control [4], and industrial defect detection systems [5].

A well established statistical tool that captures the second-order statistical information is the co-occurrence matrix [6]. The calculation of the co-occurrence matrix has a complexity of only $O(N^2)$ for an input image of $N \times N$-pixel dimensions, but the calculation of multiple matrices per time unit increases the processing power requirements. Using software co-occurrence matrix implementations running on conventional general-purpose processors does not enable real-time performance in a variety of applications, which require a high number of calculated matrices per time unit. Such demanding applications in the field of image processing include analysis of video streams [1,6], content-based image retrieval [7], real-time industrial applications [5] and high-resolution multispectral image analysis [2].

Field Programmable Gate Arrays (FPGAs) are high-density reconfigurable devices that can be hosted by conventional computer hardware [9]. They enable the rapid and

J. Blanc-Talon et al. (Eds.): ACIVS 2006, LNCS 4179, pp. 67 – 77, 2006.

low cost development of circuits that are adapted to specific applications and exploit the advantages of parallel architectures. A dedicated hardware system that efficiently computes co-occurrence matrices in parallel can meet the requirements for real-time image analysis applications. The Very Large Scale Integration (VLSI) architectures [10] provide an alternative to the FPGAs, but have drawbacks such as higher cost and time-consuming development. Furthermore, they cannot be reconfigured.

Within the first FPGA-based systems dedicated to co-occurrence matrix computations, was the one presented in [5,11]. It involves the computation of two statistical measures of the co-occurrence matrix. Moreover, the measures are extracted indirectly, without calculating the co-occurrence matrix itself. A later work by Tahir et al. [2] presents an FPGA architecture for the parallel computation of 16 co-occurrence matrices. The implementation exploits the symmetry, but not the sparseness of the matrices, resulting in a large FPGA area utilization. This leads to the need of a separate core for the feature calculation. Thus, the system is capable of processing high-resolution images, but does not achieve real time performance.

In this paper, we present a novel FPGA based system that allows the parallel computation of 16 co-occurrence matrices and 4 feature vectors. The dedicated hardware exploits both the symmetry and the sparseness of the co-occurrence matrix and uses an efficient approximation method for the logarithm, enabling real-time feature extraction for input images of dimensions up to 2048×2048 pixels. Furthermore, the system comprises of a single core for both the co-occurrence matrix and the feature calculation. Thus, no overhead is incurred by reprogramming cores onto the FPGA in order to calculate the feature vectors.

The paper is organized in five sections. Section 2 refers to the second-order statistical features and their integer arithmetic formulation. The architecture of the proposed system is described in Section 3. Section 4 presents the experimental results that demonstrate the system performance. The conclusions of this study are summarized in Section 5.

2 Second-Order Statistical Features

The co-occurrence matrix of an $N\times N$-pixel image block, encodes the gray-level spatial dependence based on the estimation of the second-order joint-conditional probability density function $P_{d,\theta}(i, j)$. It is computed by counting all pairs of pixels of an image block at distance d having gray-levels i and j at a given direction θ.

$$P_{d,\theta}(i,j) = \frac{C_{d,\theta}(i,j)}{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} C_{d,\theta}(i,j)} \tag{1}$$

where $C_{d,\theta}(i, j) = \#\{(m, n), (u, v) \in N\times N: f(m, n) = j, f(u, v) = i, |(m, n)-(u, v)| = d, \angle((m, n), (u, v)) = \theta\}$, # denotes the number of elements in the set, $f(m, n)$ and $f(u, v)$ correspond to the gray-levels of the pixel located at (m, n) and (u, v) respectively, and N_g is the total number of gray-levels in the image [6]. We choose $N_g = 32$ (5-bit representation).

The co-occurrence matrix can be regarded symmetric if the distribution between opposite directions is ignored. The symmetric co-occurrence matrix is derived as $P_{d,\theta}(i,j) = \left(P_{d,\theta}(i,j) + P_{d,\theta}(j,i)\right)/2$. Therefore, the co-occurrence matrix can be represented as a triangular structure without any information loss, and θ is chosen within the range of 0° to 180°. Common choices of θ include 0°, 45°, 90° and 135° [1,2,6,12].

Moreover, depending on the image dimensions, the co-occurrence matrix can be very sparse, as the number of gray-level transitions for any given distance and direction, is bounded by the number of image pixels.

Out of the 14 features originally proposed by Haralick et al. [6] we have considered four, namely angular second moment (f_1), correlation (f_2), inverse difference moment (f_3) and entropy (f_4):

$$f_1 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P_{d,\vartheta}^2(i,j) \tag{2}$$

$$f_2 = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} i \cdot j \cdot P_{d,\vartheta}(i,j) - \mu_x \cdot \mu_y}{\sigma_x \cdot \sigma_y} \tag{3}$$

$$f_3 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} \frac{1}{1+(i-j)^2} P_{d,\vartheta}(i,j) \tag{4}$$

$$f_4 = -\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} P_{d,\vartheta}(i,j) \cdot \log P_{d,\vartheta}(i,j) \tag{5}$$

where μ_x, μ_y, σ_x and σ_y are the means and the standard deviations of the marginal probabilities $P_x(i)$ and $P_y(j)$ obtained by summing the rows and columns of matrix $P_{d,\vartheta}(i,j)$ respectively. These four measures have been shown to provide high discrimination accuracy that can only be marginally increased by adding more features to the feature vector [1], [13].

The calculation of the four measures requires floating point operations that result in higher FPGA area utilization and lower operating frequencies. To implement the calculation of the measures efficiently in hardware, we have extracted five expressions that can be calculated using integer arithmetic:

$$V_1 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} C_{d,\vartheta}^2(i,j) \tag{6}$$

$$V_2 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} i \cdot j \cdot C_{d,\vartheta}(i,j) \tag{7}$$

$$V_3 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} C_{d,\vartheta}(i,j)\cdot IDMLUT[|i-j|] \tag{8}$$

$$V_4 = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} C_{d,\vartheta}(i,j)\cdot\left(2^{28}\cdot\log C_{d,\vartheta}(i,j)\right) \tag{9}$$

$$V_5 = \sum_{i=1}^{N_g} C_x^2(i) \tag{10}$$

$$C_x(i) = r\cdot P_x(i) \tag{11}$$

$$r = \sum\sum C_{d,\vartheta}(i,j) \tag{12}$$

$$IDMLUT[k] = \left\lfloor \frac{2^{31}}{1+k^2} \right\rfloor \tag{13}$$

The logarithm in Eq. (9) is approximated using the method described in Section 3.2, whereas *IDMLUT* is a 32×32-bit Look Up Table (LUT) used for the calculation of the Inverse Difference Moment. The result of the calculation in hardware is a vector $\overline{V} = [V_1, V_2, V_3, V_4, V_5]$ that is used for the calculation of the four Haralick features through the use of the following equations:

$$f_1 = \frac{V_1}{r^2} \tag{14}$$

$$f_2 = \frac{(N_g - 1)\cdot(r\cdot N_g^2\cdot V_2 - r^2)}{N_g^2\cdot V_5 - N_g\cdot r^2} \tag{15}$$

$$f_3 = \frac{V_3}{2^{31}\cdot r} \tag{16}$$

$$f_4 = r\log r - \frac{V_4}{2^{28}\cdot r} \tag{17}$$

Eqs. (14)-(17) are executed in software. The computation of these equations incurs a negligible overhead to the overall system performance.

3 System Architecture

The architecture of the proposed system was developed in Very High Speed Integrated Circuits Hardware Description Language (VHDL). It was implemented on a Xilinx Virtex-XCV2000E-6 FPGA, which is characterized by 80×120 Configurable

Logic Blocks (CLBs) providing 19,200 slices (1 CLB = 2 slices). The device includes 160 256×16-bit Block RAMs and can support up to 600kbit of distributed RAM. The host board, Celoxica RC-1000 has four 2MB static RAM banks. The FPGA and the host computer can access the RAM banks independently, whereas onboard arbitration and isolation circuits prohibit simultaneous access.

The system architecture is illustrated in Fig. 1. The FPGA implementation includes a control unit, four memory controllers (one for each memory bank), 16 Co-occurrence Matrix Computation Units (CMCUs) and four Vector Calculation Units (VCUs). Each input image is divided into blocks of user-specified dimensions and loaded into a corresponding RAM bank using a 25-bit per pixel representation. Each pixel is represented by a vector $\overline{a} = [a_p, a_0, a_{45}, a_{90}, a_{135}]$ that comprises of five 5-bit components, namely, the gray-level a_p of the pixel and the gray-levels a_0, a_{45}, a_{90} and a_{135} of its neighboring pixels at 0°, 45°, 90° and 135° directions.

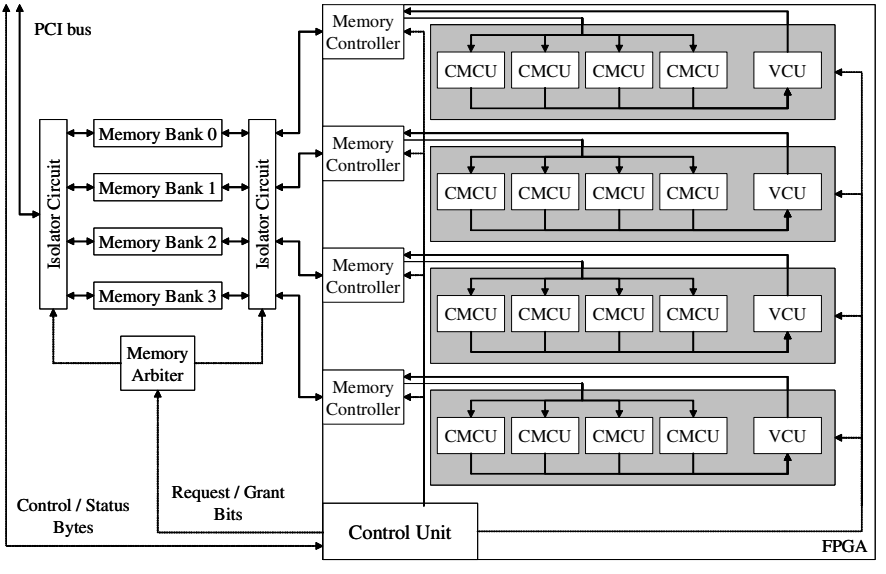


Fig. 1. Overview of the system architecture

All FPGA functions are coordinated by the control unit, which generates synchronization signals for the memory controllers, the CMCUs and the VCUs. The control unit also handles communication with the host, by exchanging control and status bytes, and requesting or releasing the ownership of the on-card memory banks. The system includes 16 CMCUs that are grouped in four quadruplets. Each CMCU in the quadruplet reads the vectors $\overline{a}$ that represent an image block from one of the memory controllers and computes the GLCM for a single direction. The 16 CMCU outputs of the four quadruplets are connected to the four VCUs that calculate the vectors $\overline{V}$ from the GLCMs. These vectors are written to the on-card memory through the memory controllers.

3.1 Co-occurrence Matrix Computation Units

Considering the requirements of the proposed application, the CMCU was developed to meet three main objectives: small FPGA area utilization, high throughput per clock cycle and high frequency potential. The small area utilization allows the implementation of the four VCUs on the same core, whereas the high throughput and frequency ensure the high efficiency of the design. To meet these three objectives we have considered various alternatives for the implementation of the CMCUs. These include the utilization of the existent FPGA BlockRAM arrays, the implementation of standard sparse array structures that store pairs of indices and values, and the implementation of set-associative [14] sparse arrays. The BlockRAM arrays and the standard sparse array structures would not suffice to meet all three objectives. The BlockRAM arrays would lead to larger area utilization, compared with the sparse implementations, whereas the standard sparse arrays would result in a lower throughput, compared with the other implementations, as the cycles needed to traverse the indices of the array are proportional to its length. In comparison, the set-associative arrays could be considered as a more flexible alternative that can be effectively used for achieving all three objectives.

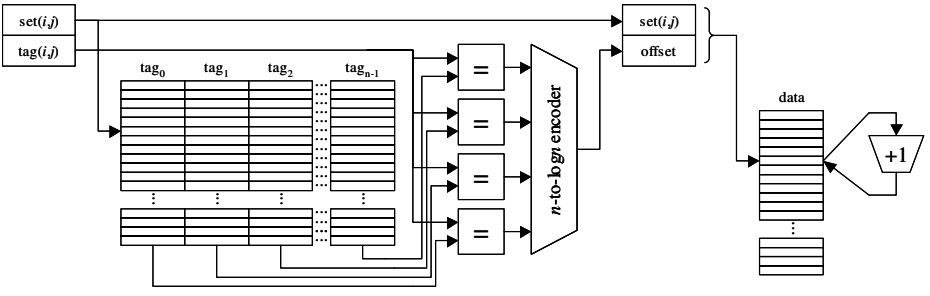


Fig. 2. Structure of the Co-occurrence Matrix Computation Unit

The internal structure of the CMCU is illustrated in Fig. 2. Every CMCU is implemented by means of an n-way set-associative array of N_c cells and auxiliary circuitry, which include n comparators, an n-to-$\log_2 n$ priority encoder and an adder. The set-associative arrays can be utilized for efficient storage and retrieval of sparse matrices, ensuring a throughput of one access per cycle with a latency of four cycles. An n-way set-associative array consists of n independent tag arrays ($tags_0$ - $tags_{n-1}$). The tag-arrays are implemented in the distributed RAM of the FPGA and each of them consists of N_c/n cells. The set-associative array uniquely maps an input pair of 5-bit gray-level intensities (i, j) into an address of the N_c-cell data array. The data arrays are implemented using FPGA Block RAMs, each of which can hold up to 256 co-occurrence matrix elements. The data array cells contain the number of occurrences of the respective (i, j) pairs.

The circuit is implemented as a four-stage pipeline. In the first two cycles the circuit reads an input pair of gray-level intensities (i, j) and maps it to the address of the BlockRAM cell that stores $C_{d,\theta}(i,j)$. In the next two cycles the value of $C_{d,\theta}(i,j)$ is retrieved from the *data* array and incremented by one. The necessary forwarding

circuits are implemented, ensuring a stall free operation of the pipeline regardless of the input data, thus guaranteeing a throughput of one update operation per cycle with a latency of four cycles. After all input pairs (i,j) have been read and the corresponding cells have been updated, the unit outputs the computed GLCM.

3.2 Vector Calculation Units

The Vector Calculation Unit receives a GLCM computed by a CMCU and calculates the $\overline{V} = [V_1, V_2, V_3, V_4, V_5]$ (Eqs. 6-10) vector. The resulting vector is written to a bank of the on-card memory through the corresponding memory controller. The calculation of V_1 to V_4 is implemented in four independent pipelined circuits. The pipeline stages for each circuit are a preprocessing stage, calculation stages, a postprocessing stage and an accumulation stage. The preprocessing and postprocessing stages facilitate the operations needed to calculate the V_1 to V_5 from the lower triangular representation of the GLCM. In the preprocessing stage, the elements of the diagonal of the GLCM are doubled and in the postprocessing stage the intermediate results of the computation are multiplied by two for all elements that do not belong to the diagonal. The calculation stages involve LUT access or arithmetic operations such as addition, multiplication and subtraction. The output of each postprocessing stage is accumulated in a register during the accumulation stage. The intermediate results of each operation are not truncated or rounded, ensuring a high accuracy of the final results. The width of the integers increases after each arithmetic operation. The values of i and j are 5 bits wide and their product $i \cdot j$ is 10 bits wide. The value of $C_{d,\theta}(i,j)$ is 16 bits wide and the product $i \cdot j \cdot C_{d,\theta}(i,j)$ is represented by 26 bits. The accumulators in the final stage of the computation are 64 bits wide.

The calculation of V_5 (Eq. 10) is implemented in two separate pipelined circuits. The first pipeline has two stages and uses a 64×16-bit BlockRAM for the storage of $C_x(i)$. At the first stage, the previous value of $C_x(i)$ is read from the BlockRAM and at the second stage it is incremented by $C_{d,\theta}(i,j)$ and stored back to the memory. A forwarding circuit ensures correct operation of the pipeline without stalls. The second pipeline is activated when all values $C_{d,\theta}(i,j)$ have been read and $C_x(i)$ has been calculated. It consists of three stages. At the first stage, $C_x(i)$ is retrieved from the BlockRAM, at the second it is squared and at the third it is accumulated into a register. The value of the accumulator after all $C_x(i)$ have been processed is the correct value of V_5.

Computation of the Logarithm. To support the calculation of V_4 (Eq. 9), we implemented a method for the efficient approximation of the base-2 logarithm of 16-bit integers. This method results in a 3-stage pipelined circuit that requires 123 slices (less than 0.7% of the total FPGA area) and achieves a maximum frequency of 121.5MHz. The stages of the circuit are:

1. The integer part of the logarithm $l_i = \lfloor \log_2 n \rfloor$ is calculated by means of a priority encoder. Its value is the position of the most significant bit of the input integer.

$$2^k \le n < 2^{k+1} \quad \Rightarrow \quad k \le \log_2 n < k+1 \quad \Rightarrow \quad l_i = k \tag{18}$$

2. The fractional part of the logarithm $l_f = \log_2 n - l_i$ is estimated from Eq. (19), as a linear approximation between the points (2^k, k) and (2^{k+1}, k+1). The value l_f can be

easily extracted from the binary representation of n, by removing its most significant bit and right shifting by k bits.

$$\frac{n-2^k}{2^{k+1}-2^k}=\frac{l_i+l_f-l_i}{k+1-k} \quad \Rightarrow \quad l_f=\frac{n}{2^k}-1 \tag{19}$$

3. A novel method has been devised to increase the accuracy of this linear approximation of the logarithm. The fractional part of the logarithm l_f is transformed by Eq. (20).

$$l_f^{'}=\begin{cases}(1+a)\cdot l_f & if \quad l_f \le 1/2\\ (1-a)\cdot l_f+a & if \quad l_f > 1/2\end{cases} \tag{20}$$

The optimal a is the one that minimizes the error E between $\log_2 n$ and the approximated logarithm ($l_i + l_f^{'}$), where

$$E=\frac{1}{65535}\sum_{n=1}^{65535}\frac{\left|\log_2 n-(l_i+l_f^{'})\right|}{\log_2 n}\ . \tag{21}$$

The error E as a function of a is illustrated in Fig. 3

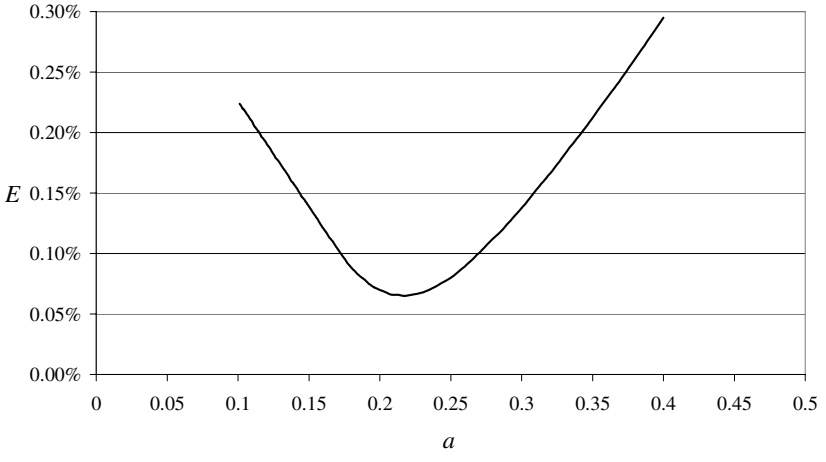


Fig. 3. Estimation of the error E for different values of a

Although the minimum error was achieved for a=0.22 (E=0.07%), we selected a=0.25 (E=0.08%) because it is implemented using simpler circuits (just a single shifter) and the error is only marginally higher.

4 Results

The proposed system was tested using natural raw images of 512×512, 1024×1024 and 2048×2048-pixel dimensions. The images were divided into blocks of 8×8,

16×16, 32×32, 64×64, 128×128 and 256×256-pixel dimensions and given as input to the system in order to evaluate its performance.

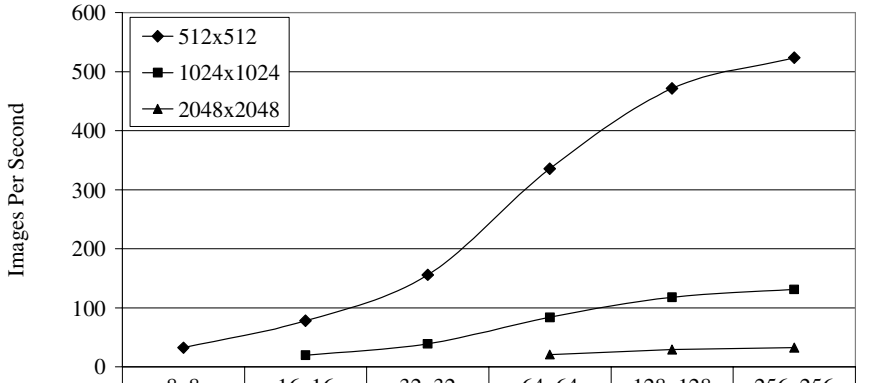
In the case of a 16×16-pixel or smaller input block, the triangular co-occurrence matrix for $N_g = 32$ is sparse, as the number of pixel pairs that can be considered for its computation, is smaller than the total number of co-occurrence matrix elements. Therefore, for input blocks of 8×8 and 16×16 pixels, N_c is set to the maximum possible value of 64 and 256 respectively.

In the case of a 32×32-pixel or a larger input block, the co-occurrence matrix is not considered sparse, as the number of all possible pixel pairs that take part in its computation is larger than the total number of its elements (i.e. 528). Therefore, N_c is set to 528.

By following a grid search approach for the determination of n, it was found that the sixteen-way set-associative arrays ($n = 16$) result in the optimal tradeoff between circuit complexity and time performance.

The proposed architecture, as implemented on the Xilinx Virtex-XCV2000E-6 FPGA, operates at 36.2MHz and 39.8MHz and utilizes 77% and 80% of the FPGA area for 8×8 and 16×16 input blocks respectively, by exploiting the sparseness of the co-occurrence matrices. The use of larger input blocks results in approximately the same operating frequency reaching 37.3MHz and an area utilization of 83%.

The image and block dimensions for which the proposed system achieves real-time performance are illustrated in Fig. 4.



	8x8	16x16	32x32	64x64	128x128	256x256
512x512	32.29	78.19	156.01	335.57	471.70	523.56
1024x1024		19.55	39.02	83.96	117.92	131.06
2048x2048				20.99	29.47	32.79

Block size

Fig. 4. Performance of the proposed system in images per second

The results show that as the dimensions of the block increases, the system performance is enhanced. It is worth noting that a real-time performance for video applications is reached using images of 2048×2048 pixels with a block size of 128×128 pixels or higher. Best time performance is achieved for 512×512-pixel images with a

block of 256×256 pixels, whereas marginal real-time performance is obtained for 1024×1024-pixel images with a block of 16×16 pixels, and for 2048×2048 with a block of 64×64 pixels.

5 Conclusion

We presented a novel dedicated hardware system for real-time extraction of second-order statistical features from high-resolution images. It is based on a parallel FPGA architecture, enabling the concurrent calculation of sixteen gray-level co-occurrence matrices and four feature vectors. The input images are divided into blocks which are loaded to the RAM banks of the FPGA board. The FPGA processes each block and writes back four feature vectors. The performance of the proposed system increases as the image blocks become larger and the number of calculated vectors decreases.

The experimental results showed that the proposed system can be used for high resolution video applications which require real-time performance, the analysis of multiple video streams, and other demanding image analysis tasks.

The proposed system displays several advantages over the system presented in [2], which are summarized in the following:

- It calculates both the co-occurrence matrix and the features in a single FPGA core, whereas in [2] they are calculated in two separate cores. This avoids the overhead introduced by reprogramming each core onto the FPGA.
- It is capable of producing multiple feature vectors for each image, whereas in [2] a single feature vector is produced for each image. This facilitates in more accurate localization of texture within the image.
- It uses 25 bits per pixel for the representation of the input images, whereas in [2] the input images are represented using 5 bits per pixel. This allows a read rate of 25 bits per clock cycle from each memory bank, which results in a 5 times larger input bandwidth.
- It uses set-associative arrays for the sparse representation of the co-occurrence matrices, which enable the inclusion of four vector calculation units in a single core.

The results advocate the feasibility of real-time feature extraction from high-resolution images, using an efficient hardware implementation.

Future perspectives of this work include:

- The implementation of more second-order statistical features in a single FPGA core.
- The implementation of Color Wavelet Covariance (CWC) features [1] or other features based on co-occurrence matrices.
- The classification of feature vectors in hardware.

Acknowledgement

This research was funded by the Operational Program for Education and Vocational Training (EPEAEK II) under the framework of the project “Pythagoras - Support of

University Research Groups" co-funded by 75% from the European Social Fund and by 25% from national funds.

References

1. Karkanis, S.A., Iakovidis, D.K., Maroulis, D.E., Karras, D.A., Tzivras, M.: Computer Aided Tumor Detection in Endoscopic Video Using Color Wavelet Features. IEEE Trans. Inf. Technol. Biomed. 7 (2003) 141-152
2. Tahir, M.A., Bouridane, A., Kurugollu, F.: An FPGA Based Coprocessor for GLCM and Haralick Texture Features and their Application in Prostate Cancer Classification. Anal. Int. Circ. Signal Process. 43 (2005) 205-215
3. Baraldi, A., and Parmiggiani, F.: An Investigation of the Textural Characteristics Associated with Gray Level Cooccurrence Matrix Statistical Parameters. IEEE Trans. Geosc. Rem. Sens. 33 (2) (1995) 293-304
4. Shiranita, K., Miyajima, T., Takiyama, R.: Determination of Meat Quality by Texture Analysis. Patt. Rec. Lett. 19 (1998) 1319-1324
5. Iivarinen, J., Heikkinen, K., Rauhamaa, J., Vuorimaa, P., Visa, A.: A Defect Detection Scheme for Web Surface Inspection. Int. J. Pat. Rec. Artif. Intell. (2000) 735-755
6. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural Features for Image Classification. IEEE Trans. Syst. Man Cybern. 3 (1973) 610-621
7. Iakovidis, D.K., Maroulis, D.E., Karkanis, S.A., Flaounas, I.N.: Color Texture Recognition in Video Sequences Using Wavelet Covariance Features and Support Vector Machines. Proc. 29th EUROMICRO, Sept. 2003, Antalya, Turkey, pp. 199-204
8. Wei, C.-H., Li, C.-T., Wilson, R.: A Content-Based Approach to Medical Image Database Retrieval, in Database Modeling for Industrial Data Management: Emerging Technologies and Applications. ed. by Z. Ma, Idea Group Publishing, 2005
9. York, T.A.: Survey of Field Programmable Logic Devices. Microprocessors and Microsystems. 17 (7) (1993) 371-381
10. Ba, M., Degrugillier, D., Berrou, C.: Digital VLSI Using Parallel Architecture for Co-occurrence Matrix Determination. Proc. Int. Conf. on Acoustics, Speech, and Signal Proc., 1989, Vol. 4, pp. 2556-2559
11. Heikkinen, K., Vuorimaa, P.: Computation of Two Texture Features in Hardware. Proc. 10th Int. Conf. Image Analysis and Processing, Sept. 1999, Venice, Italy, pp. 125-129
12. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, Academic Press, San Diego (1999)
13. Haralick R.M.: Texture Measures for Carpet Wear Assessment, IEEE Trans. Pattern Analysis and Machine Intelligence, 10 (1) (1988) 92-104
14. Hennesy J.L., Patterson D.A.: Computer Architecture, A Quantitative Approach, Morgan Kaufmann, 2002