Development of computational methods for biomedical data analysis in software and hardware

Dimitris G. Bariamis[*]

Dept. of Informatics and Telecommunications, University of Athens

Abstract. The topic of this thesis is the development of novel methods of biomedical data analysis in software and hardware. A novel system has been developed for the extraction of second-order statistic texture features from images and video in software and hardware. Additionally, a novel FPGA-based architecture for the calculation of the logarithm function has been developed, which achieves fast operation, high accuracy and low resource utilization. Finally, a novel method for automatic microarray image gridding has been developed, based on the maximization of the margin between consecutive rows or columns of microarray spots.

1 Feature extraction system

Texture is an innate property of the natural objects and it is widely used for video content description. The utility of texture feature extraction from video extends to a wide range of advanced modern applications, such as segmentation of objects in image sequences, object recognition and tracking of moving objects.

The Gray Level Cooccurrence Matrix (GLCM) features [1] describe the textural image content by encoding the second order statistical properties of texture. These features have been successfully utilized in a number of applications including medical, remote sensing and industrial visual inspection applications.

GLCMs encode the gray level spatial dependence based on the estimation of the 2nd order joint-conditional probability density function, which is computed by counting all pairs of pixels of a video frame block at distance d having gray levels i and j at a given direction θ. The cooccurrence matrix can be regarded symmetric if the distribution between opposite directions is ignored, so the usual values of the angular displacement are $0°$, $45°$, $90°$ and $135°$. We have considered four GLCM features [1], namely angular second moment (f_1), correlation (f_2), inverse difference moment (f_3) and entropy (f_4), which have been shown to provide high discrimination accuracy.

In the following equations, p_{ij} is the ijth entry of the cooccurrence matrix, μ_x, μ_y, σ_x, and σ_y are the means and standard deviations of the marginal probabilities $P_x(i)$ and $P_y(j)$ obtained by summing up the rows or the columns of matrix p_{ij} respectively.

[*] Dissertation advisor: Dimitris Maroulis, Associate Professor

$$f_1 = \sum\sum p_{ij}^2 \qquad f_2 = \frac{\sum\sum i \cdot j \cdot p_{ij} - \mu_x \cdot \mu_y}{\sigma_x \cdot \sigma_y}$$
$$f_3 = \sum\sum \frac{1}{1+(i-j)^2} \cdot p_{ij} \qquad f_4 = -\sum\sum p_{ij} \cdot \log p_{ij} \tag{1}$$

The above calculations require floating point operations that would result in high FPGA area utilization and low operating frequencies. To implement the calculation of the features efficiently in hardware, we have reformulated the equations by extracting the following five expressions (V_1 to V_5), where c_{ij} is the ijth entry of the unnormalized cooccurrence matrix, $C_x(k)$ is obtained by summing up the rows of matrix c_{ij} and $\mathrm{L1}[i-j]$ is a small lookup table. In these equations, the operations needed to calculate V_1 to V_5 are performed exclusively in hardware using integer or fixed point arithmetic.

$$V_1 = \sum\sum c_{ij}^2 \qquad V_2 = \sum\sum i \cdot j \cdot c_{ij}$$
$$V_3 = \sum\sum c_{ij} \cdot \mathrm{L1}[i-j] \tag{2}$$
$$V_4 = \sum\sum c_{ij} \cdot \log_2 c_{ij} \qquad V_5 = \sum C_x^2(k)$$

1.1 Hardware implementation

The hardware implementation is based on a Xilinx XCV2000E-6 FPGA, programmed in VHDL. The architecture of the implemented hardware is illustrated in Fig. 1. The software iteratively feeds the FPGA board with four video frame blocks per iteration. The FPGA reads each block's pixels, calculates the GLCM of each block and their respective feature vectors for the four directions and a distance of one pixel, and stores them into memory bank 1 and 2.

The architecture consists of several units. The control unit coordinates the functions of the other units and handles the communication with the host, whereas the memory controllers handle the transactions between the FPGA and the memory banks. The circular buffers implements a buffering scheme that reads the pixels of the four blocks sequentially at a rate of four pixels per clock cycle and produces pairs of gray level intensities (i, j) for each block and direction considered, at a rate of 20 pixels per cycle.

A GLCM calculation unit (GCU) is used for the calculation of the GLCM of a single block for a particular direction. It consists of an n-way set associative array [2] with a capacity of N_c cells and the auxiliary circuitry needed for the calculation of the GLCM. Set associative arrays can be used for efficient storage and retrieval of sparse matrices, with a throughput of one operation per cycle. The set associative array uniquely maps an input pair of 6-bit gray-level intensities (i, j) to an address of the N_c-cell data array, which is implemented using FPGA Block RAMs.

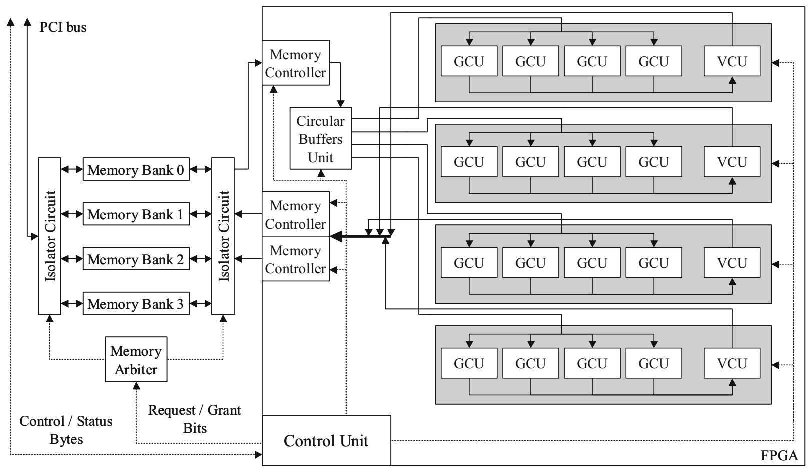

Fig. 1. Overview of the hardware architecture

Each vector calculation unit (VCU) receives a GLCM generated by a GCU and produces a vector $\overline{V} = [V_1, V_2, V_3, V_4, V_5]$, which is stored to the memory banks through the memory controllers. The calculation of V_1 to V_5 is implemented in five independent pipelined circuits, which include a number of computation stages and a final accumulation stage. These stages involve table lookup, logic or arithmetic operations such as multiplication, addition and subtraction. The logarithm function is approximated using an efficient piecewise linear method, described in depth in section 2.

1.2 Results

The performance of the proposed system was experimentally evaluated using standard test video clips, encoded in both CIF and QCIF formats. A high processing rate can only be achieved when all vectors $\overline{V}$ of a video frame sequence are calculated in hardware, which depends on the proper selection of the GCU parameters N_c and n. Small blocks ($W = 8$) can be handled entirely in hardware by set associative arrays of $n \geq 8$ with $N_c = 256$. Larger blocks ($W = 16$) require set-associative arrays of $n \geq 16$ and $N_c = 512$. Therefore, the configurations of the proposed system with $(N_c, n) = (256, 8)$ and $(512, 16)$ were implemented and evaluated.

The implementation results per configuration are presented in Table 1. The performance of the respective configurations in frames per second (fps) is presented in Table 2. The second table also includes the results of a Xilinx XC2V4000-4 FPGA, demonstrating the performance increase achieved by using a next generation FPGA. The software performance was measured on two workstations based on a 1GHz Athlon and on an Athlon XP 2800+ processor.

Table 1. Implementation results for Xilinx XCV2000E-6 FPGA

N_c	n	Slices	Area	Frequency	BlockRAMs
256	8	11625	60%	43.95 MHz	24
512	16	16158	84%	35.75 MHz	40

Table 2. Performance in frames per second

Video	W	XCV2000E-6 FPGA		XC2V4000-4 FPGA		Software	
		$n = 8$ $N_c = 256$	$n = 16$ $N_c = 512$	$n = 8$ $N_c = 256$	$n = 16$ $N_c = 512$	Athlon 1 GHz	Athlon XP 2800+
CIF	8	74.44	35.89	115.64	56.46	5.3	8.57
	16	-	133.01	-	209.24	21.2	32.38
QCIF	8	297.75	143.58	462.57	225.88	21.30	36.03
	16	-	532.02	-	836.95	84.80	131.21

These results illustrate that the proposed FPGA-based system outperforms general-purpose processors for GLCM feature extraction from video frame blocks. Even though general purpose processors have a significant frequency advantage, the parallel FPGA implementations result in higher overall performance. It is also worth noting that the VirtexE FPGA (XCV2000E-6) retains its performance advantage compared to the AthlonXP processor, even though the latter was released four years later.

2 Logarithm calculation

The utility of the logarithm in computer science spans a broad spectrum of applications, many of which require a vast number of logarithmic operations per second, while also requiring high accuracy. Hardware architectures that have been proposed for the approximation of the base-2 logarithm include power series and polynomial methods, high-radix algorithms, lookup tables, the CORDIC (Coordinate Rotation Digital Computer) algorithm [3] and linear approximation methods [4–8]. Most commonly, CORDIC has been the algorithm of choice for logarithm approximation in state of the art processing units. The piecewise linear approximation methods are based on Mitchell's method [4] and aim at crude, however fast approximation of the logarithm. The logarithm function is approximated by two [5], four [6, 7] or six [8] consecutive linear segments.

2.1 Generalized piecewise linear approximation

Combining both computational efficiency and approximation accuracy, we developed a generalized piecewise linear approximation of the base-2 logarithm in a fast and area-efficient way. This approach provides the versatility of using an arbitrary number of segments, rather than a small fixed number of segments used in the state of the art architectures. The achieved approximation accuracy depends on the number of segments used, which also affects the size of a ROM that is used for storing the parameters that control the computation. These parameters can be adapted to any data set, further enhancing the achieved accuracy. The implementation of the ROM using an FPGA BlockRAM allows the parameters to be updated without reconfiguration of the FPGA core, thus providing the advantage of data set adaptability.

The generalized piecewise linear approximation approach involves three steps:

1. The integral part of the logarithm, $l_i(x) = \lfloor log_2(x) \rfloor$, is determined by the position of the Most Significant Bit (MSB) of the input number x.
2. The fractional part of the logarithm, i.e. $l_f(x) = log_2(x) - l_i(x)$, is estimated by linear approximation between the points $(2^n, n)$ and $(2^{n+1}, n+1)$, of the function $log_2(x)$. The approximated fractional part is

$$l_f(x) = \frac{x - 2^{l_i(x)}}{2^{l_i(x)+1} - 2^{l_i(x)}} = \frac{x}{2^{l_i(x)}} - 1 \tag{3}$$

3. In the third step, a new approximation $l'_f(x)$ of the fractional part of the logarithm is derived as a piecewise linear function of $l_f(x)$, between the points $(2^n, n)$ and $(2^{n+1}, n+1)$ using s linear segments of equal length. The segment $seg(x)$ to which x belongs is determined by the fractional part of its logarithm: $seg(x) = \lfloor s \cdot l_f(x) \rfloor$. The new fractional part $l'_f(x)$ is:

$$l'_f(x) = l_f(x) + \left(c_{seg(x)+1} - c_{seg(x)}\right) \cdot \left(s \cdot l_f(x) - seg(x)\right) + c_{seg(x)} \tag{4}$$

The optimal parameters c_i can be determined upon the data set used in a particular application by minimizing the approximation error E (Eq. 5), which represents the weighted sum of the relative differences between the actual and approximated values of the logarithm. The weight $p(x)$ used for the calculation is the Probability Mass Function (PMF) of the data set.

$$E = \sum_x p(x) \cdot \left| \frac{\left(l_i(x) + l'_f(x)\right) - \log_2(x)}{\log_2(x)} \right| \tag{5}$$

The generalized logarithm approximation architecture is implemented as a fully pipelined circuit, in order to achieve a throughput of one result per clock cycle. It consists of 6 pipeline stages and one dual-ported ROM. The ROM stores the parameters c_i in a b-bit wide fixed point representation and is implemented on a dual-ported BlockRAM.

2.2 Results

Experiments were conducted to evaluate the performance of the proposed architecture with different datasets. There are several applications where the PMF of the input number x is not uniform, such as the analysis of medical or biological

Table 3. Error E for piecewise linear methods for data with a uniform PMF

Method	Segments	Error E
Mitchell [4]	1	$3.992{\cdot}10^{-3}$
SanGregory et al. [5]	2	$1.888{\cdot}10^{-3}$
Combet et al. [6]	4	$1.106{\cdot}10^{-3}$
Hall et al. [7]	4	$1.650{\cdot}10^{-4}$
Abed et al. [8]	2	$1.106{\cdot}10^{-3}$
	3	$6.748{\cdot}10^{-4}$
	6	$2.277{\cdot}10^{-4}$
Proposed method	2	$6.539{\cdot}10^{-4}$
	16	$6.826{\cdot}10^{-6}$
	128	$1.001{\cdot}10^{-7}$
	1024	$1.829{\cdot}10^{-9}$

Table 4. Comparative implementation results for data with a uniform PMF

Method	E	s	b	Slices	Frequency MHz
Hall et al. [7]	$1.65{\cdot}10^{-4}$	4	-	202	322.79
Proposed method	$3.55{\cdot}10^{-4}$	4	2	249	401.78
	$3.02{\cdot}10^{-5}$	8	8	254	318.02
	$6.83{\cdot}10^{-6}$	16	16	295	302.57
	$4.08{\cdot}10^{-7}$	64	16	285	302.57
	$8.95{\cdot}10^{-8}$	512	16	293	302.57
	$1.83{\cdot}10^{-9}$	1024	24	329	288.27
CORDIC	$1.36{\cdot}10^{-4}$	-	-	445	358.42
	$5.35{\cdot}10^{-7}$	-	-	1363	329.83
	$2.07{\cdot}10^{-9}$	-	-	2587	310.89

data. We have considered two such data sets, namely the logarithmic normalization of the intensities of more than nine million microarray spots (Data Set 1) and the estimation of the entropy of grey level co-occurrence matrices, involving more than one billion logarithm calculations (Data Set 2). We have also considered several synthetic data sets based on the Gauss-Kuzmin, the uniform and a number of normal distributions. The optimal set of the parameters c_i for these PMFs was calculated and the parameters were then loaded into the ROM of the architecture.

The architecture was implemented on a Xilinx Virtex-5 FPGA (LX110T-3) and was tested for several different configurations, with the number of linear segments s ranging between 2 and 1024 and b ranging from 2 to 24 bits.

Table 3 shows the error E of the proposed method when adapted to data with a uniform PMF. When using eight or more segments, the proposed method significantly outperforms the other methods for all data sets evaluated. Table 4 shows the most characteristic of the implemented configurations of the proposed architecture and the CORDIC core, sorted by the obtained approximation error when used on the uniform PMF data set. It also includes the implementation results for the simple linear approach proposed by Hall et al. [7], which has a low slice utilization but can only achieve an approximation error in the order of 10^{-4}.

It is worth noting that the approximation error for the proposed architecture when adapted to Data Set 2 is up to 200 times smaller that the uniform PMF data set, illustrating the significant gains in accuracy due to the data set adaptability. Moreover, the proposed architecture can be implemented in significantly fewer slices than the CORDIC core for the same approximation error, while operating at a similar frequency. The proposed architecture only requires 329 slices when $E = 1.83 \cdot 10^{-9}$, compared to 2587 slices for the CORDIC core for a slightly higher error of $E = 2.07 \cdot 10^{-9}$.

3 Microarray gridding

The goal of a microarray experiment is the quantification of the expression of the genes in a test sample compared to that of a reference sample. Such an experiment involves a lab process that produces a high resolution greyscale digital image, which consists of a matrix of blocks. Each block contains a number of rows and columns of spots. The grey level intensity of each spot indicates the expression level of the respective gene. The gene expression levels are extracted from microarray images in three steps, namely gridding, segmentation and intensity extraction. Gridding involves the construction of a grid covering the microarray image so that it isolates each spot into a distinct cell enabling the localization of each spot. Being the first step in the microarray image processing, the process of gridding has a significant impact on the following steps, and consequently on the accuracy of a microarray experiment.

A gridding algorithm should be able to automatically grid images that include spots of various shapes, sizes and intensities, while being robust to noise and artefacts introduced at a microarray preparation stage, as well as rotation due to misplacements. The automatic operation prevents any user intervention from affecting the microarray experiment, while also enabling higher processing throughput by analyzing large amounts of microarray images. Only a few state of the art methods have been proposed as providing fully automatic gridding, but they have specific drawbacks, such as a requirement for prior knowledge about the microarray image, specific image alignment or abundance of highly expressed spots. The most accurate of these methods [9] provides a near optimal gridding based on a genetic optimization process, however these optimization processes require long processing times to converge.

3.1 Maximum margin gridding method

A novel method for automatic cDNA microarray gridding has been developed based on a computationally efficient optimization approach. It is based on the maximization of the margin between the consecutive rows and columns of the microarray spots, which is implemented by training a linear maximum margin classifier with an automatically detected subset of spots on the microarray image. It is named *Maximum Margin Microarray Gridding* (M^{3}G) and consists of the following steps:

Rotation estimation The microarray image (Fig. 2a) is analyzed by the Radon transform, which is applied to estimate the image rotation angle. In the transformed image illustrated in Fig. 2b, the rotation angle θ of the microarray

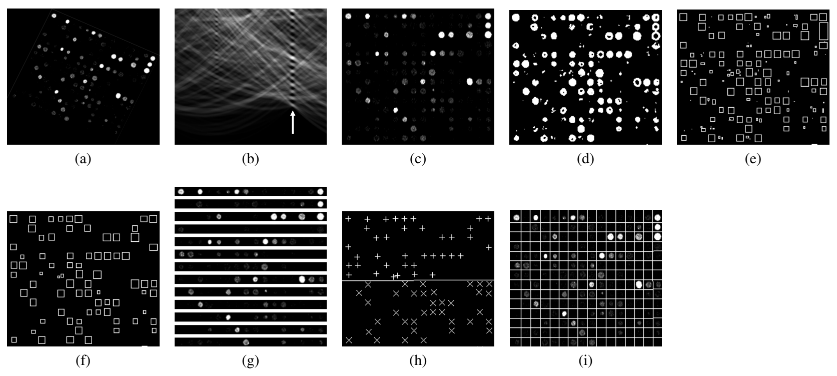

Fig. 2. Steps of the maximum margin microarray gridding method

image is estimated by locating the column with the highest mean brightness, which is denoted by the arrow. The image is subsequently counter-rotated by angle θ as illustrated by Fig. 2c.

Preprocessing The image is linearly normalized and quantized to 256 gray levels. The edges of the spots are detected by the application of the Sobel operator [10] on the normalized image. A threshold is determined automatically using the Otsu method [11] in order to binarize the image and isolate the sharpest edges that correspond to spots, as illustrated in Fig. 2d.

Spot detection All groups of consecutive white pixels are considered as residing on the same spot edge, therefore they are located and represented by a rectangle that circumscribes a pixel group (Fig. 2e).

Spot selection The spots are filtered based on their aspect ratio and their size, in order to remove false spots introduced by noise and artefacts. More specifically, the bounds s_{min} and s_{max} of the valid spot sizes are determined, so as to maximize the resemblance of the spot size distribution to the normal distribution. The spots that have sizes out of these bounds are considered false and discarded, reducing the effect of noise, artifacts and multiple merged spots on the subsequent steps. The selected spots are illustrated in Fig. 2f.

Distance estimation The optimal distance between spot rows is calculated by segmenting the input microarray image into horizontal stripes with a height of d_r pixels (Fig. 2g), which are then averaged. If d_r is selected so that it is equal to the distance between the rows, the spots of all rows will be in the same relative positions in the horizontal stripes, therefore they will be highly overlapping in the resulting average stripe. Thus, the average stripe will contain well defined spot areas, which result in a high standard deviation of the pixel intensities. For a suboptimal value of d_r, the spots will not significantly overlap, resulting in a lower standard deviation. The optimal value of d_r is thus selected by maximizing the standard deviation. The distance between columns d_c is likewise determined.

Maximum margin gridding Each selected spot is represented by a vector $\bar{x_i}$ that consists of the coordinates of the spot centre. These vectors are assigned into distinct rows and columns, based on the distances d_r and d_c. Each pair of consecutive rows or columns of spots can now be separated by a single separating line, as shown in Fig. 2h. The optimal separating line is positioned so as to maximize the margin between the rows or columns of the spots. For a pair of rows numbered k and $k+1$, the vectors that belong to row k or to any row above it are assigned a class label $c_i = +1$ and the vectors that belong to row $k+1$ or to any row below it are assigned a class label $c_i = -1$. These vectors $\bar{x_i}$, along with their respective class labels c_i are provided as a training set to a linear Support Vector Machine (SVM) classifier [12], which produces the maximum margin grid line.

The soft-margin variant of the SVM is employed, in order to diminish the effects of misdetected spots that result from artefacts or noise. A cost parameter C adjusts the effect of outliers, namely large values of C result in a grid line that is mostly determined by any outliers, whereas for smaller values of C any outliers are virtually ignored. A small fraction of these outliers might have a shape and size similar to valid spots and could therefore pass through the selection step without being discarded. The soft-margin SVM ensures that such outliers will not have an impact on the produced grid lines. The resulting grid is determined by finding the optimal grid line for each pair of consecutive rows and columns of spots, as shown in Fig. 2i.

3.2 Results

Data set and evaluation method The data set used for the evaluation of the developed method consists of 54 cDNA microarray images, from the Stanford Microarray Database [13]. The images are TIFF files with a resolution of 1900×5500 pixels and 16-bit grey level depth. Each image includes 48 blocks of 870 spots each, resulting in a total of 2,255,040 spots in the data set. These images have been produced for the study of the gene expression profiles of 54 specimens of BCR-ABL-positive and -negative acute lymphoblastic leukemia [14]. This data set is a superset of the one used by the genetic algorithm approach proposed in [9] and is accompanied by ground truth annotations regarding the positions and the sizes of the spots. In order to produce directly comparable results with the aforementioned methods, the same statistical analysis is performed. Each spot was evaluated as being perfectly, marginally or incorrectly gridded when the percentage of its pixels contained within its grid cell is 100%, more than 80%, or less than 80% respectively.

Comparative evaluation The evaluation results of M^{3}G and [9] are shown in Table 5. Out of more than two million spots present in the data set, 98.3% of the spots were perfectly gridded, while only 1.5% and 0.2% were marginally and incorrectly gridded respectively. These results show that M^{3}G achieves a percentage of marginally and incorrectly gridded spots that is more than three

Table 5. Comparative evaluation of the gridding methods

Method	Perfect	Marginal	Incorrect
M^3G	98.3%	1.5%	0.2%
Zacharia et al. [9]	94.6%	4.8%	0.6%

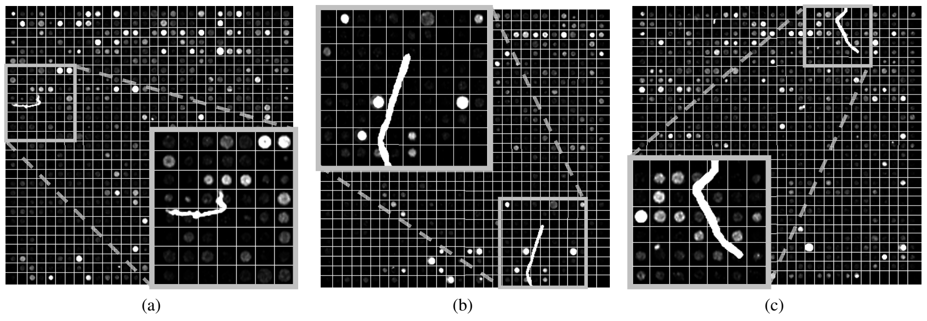

Fig. 3. Details of successful gridding in images with bright artefacts

times smaller than the one achieved by the state of the art genetic algorithm method. Additionally, M^3G is nearly one order of magnitude faster than the aforementioned genetic algorithm approach.

Figure 3 illustrates the gridding that results from the application of M^3G on three microarray images that include large and bright artefacts and includes a detailed view of the area around each artefact. Even in the vicinity of the artefacts, the gridding is not affected by their presence.

4 Conclusions

4.1 Feature extraction

A novel system capable of performing real-time texture analysis of video frames has been developed. It is capable of calculating a total of 64 features comprising of four 16-dimensional GLCM feature vectors from four video frame blocks in parallel. The hardware is based on an FPGA and it is capable of performing fast integer and fixed point operations, which include the computation of many GLCMs in parallel and the computation of GLCM features. An algorithm for the approximation of the logarithm, which is required for the computation of the entropy feature, has been included within the hardware architecture. A buffering scheme ensures a high processing throughput, while maintaining low memory bandwidth requirements. The software supports the hardware by managing the video frame transfers from/to the hardware and by performing only supplementary floating point operations.

The proposed system was tested on standard test video clips encoded in CIF and QCIF formats, and demonstrated real-time performance for video texture analysis. The evaluation procedure showed that the proposed system is capable of performing GLCM feature computations much faster than software running on modern workstations, thereby making it suitable for replacing software implementations in systems requiring real time extraction of GLCM features from video frames.

4.2 Logarithm calculation

A novel architecture for fast and area-efficient approximation of the base-2 logarithm on FPGA devices has been developed. The novel features of this architecture are the implementation of a generalized piecewise linear approximation of the logarithm function using an arbitrary number of linear segments, the adaptability to different data sets without requiring the reconfiguration of the FPGA core and the exploitation of the available FPGA resources, such as BlockRAMs and multipliers. It is thus an architecture that can be used in a variety of FPGA designs as a generic component. It is also suitable for applications operating with datasets of different probability mass functions, especially if timely critical alterations of the datasets are involved.

Other logarithm approximation architectures implementing piecewise linear approximation [5–8] use up to a maximum of 6 segments, whereas they cannot be implemented using a larger number of segments, or adapted to the dataset of each particular application. Compared with the CORDIC architecture, which is used by state of the art processing systems for logarithm approximation, the proposed architecture requires significantly less FPGA area to achieve the same or higher accuracy. Moreover, it operates at a similar frequency to the CORDIC core, while providing a throughput of one result per cycle. Thus, it can be embedded into any FPGA-based application that requires fast and accurate logarithm approximation.

4.3 Microarray gridding

A novel method has been presented for gridding cDNA microarray images without user intervention, based on the maximization of the margin between consecutive rows and columns of spots. The proposed method involves several preprocessing steps, including a Radon-based rotation estimation for the microarray image, as well as spot detection and selection. The distance between rows and columns of spots is then estimated and the positions of the selected spots are used to train a set of linear soft-margin Support Vector Machine classifiers. The proposed method achieves successful automatic gridding of cDNA microarray images in the presence of irregular spots, noise and artefacts, as well as image rotation. The experimental results on reference DNA microarray images containing more than two million spots showed that the proposed method outperforms the most accurate state of the art method, providing the potential of achieving perfect gridding for the vast majority of the spots.

Bibliography

[1] Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics **3**(6) (1973) 610–621

[2] Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach. Morgan Kaufmann (May 2002)

[3] Volder, J.E.: The CORDIC trigonometric computing technique. IRE Transactions on Electronic Computers (8) (1959) 330–334

[4] Mitchell, Jr., J.N.: Computer multiplication and division using binary logarithms. IRE Transactions on Electronic Computers **11** (Aug 1962) 512–517

[5] SanGregory, S., Siferd, R., Brother, C., Gallagher, D.: A fast, low-power logarith approximation with CMOS VLSI implementation. In: Proceedings of the Midwest Symposium on Circuits and Systems, IEEE (Aug 1999)

[6] Combet, M., Zonneveld, H., Verbeek, L.: Computation of the base two logarithm of binary numbers. IEEE Transactions on Electronic Computers **14** (Dec 1965) 863–867

[7] Hall, E.L., Lynch, D.D., Dwyer, S.J.: Generation of products and quotients using approximate binary logarithms for digital filtering applications. IEEE Transactions on Computers **19**(2) (1970) 97–105

[8] Abed, K.H., Siferd, R.E.: CMOS VLSI implementation of a low-power logarithmic converter. IEEE Transactions on Computers **52**(11) (2003) 1421–1433

[9] Zacharia, E., Maroulis, D.: An original genetic approach to the fully automatic gridding of microarray images. IEEE Transactions on Medical Imaging **27**(6) (2008) 805–813

[10] Gonzalez, R.C., Woods, R.E.: Digital Image Processing. 3rd edn. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2006)

[11] Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics **9**(1) (Jan 1979) 62–66

[12] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297

[13] Stanford Microarray Database: (2007)

[14] Juric, D., Lacayo, N.J., Ramsey, M.C., Racevskis, J., Wiernik, P.H., Rowe, J.M., Goldstone, A.H., O'Dwyer, P.J., Paietta, E., Sikic, B.I.: Differential gene expression patterns and interaction networks in BCR-ABL-positive and -negative adult acute lymphoblastic leukemias. Journal of Clinical Oncology **25**(11) (Apr 2007) 1341–1349