COMPRESSION OF MICROARRAY IMAGES USING MULTI-BITPLANE METHOD

A. Sharmila Agnal[1] and K. Mala[2]
Department of Computer Science and Engineering, Mepco Schlenk Engineering College, Tamil Nadu, India
E-mail: [1]sha_agnal@yahoo.com, [2]kmala@mepcoeng.ac.in

Abstract:
The development of microarray technology has been remarkable, and it is becoming a daily tool in many genomic research laboratories. The widespread adoption of this technology, coupled with the significant volume of data generated per experiment, in the form of images, has led to significant challenges in storage and query-retrieval. In this paper, we present a lossless bitplane based method for efficient compression of microarray images. This method is based on arithmetic coding driven by image-independent multi-bitplane finite-context models. It produces an embedded bitstream that allows progressive, lossy-to-lossless decoding. The compression results obtained by using a large set of images are compared with three image coding standards, namely, lossless JPEG2000, JBIG and JPEG-LS stated in the literature. The proposed method gives better results for all images of the test set.

Keywords:
Lossless Image Compression, Microarray Images, Arithmetic Coding, Finite-Context Modeling

1. INTRODUCTION

The DNA microarray technology has become an important tool in the study of gene function, regulation, and interaction across large numbers of genes, and even entire genomes. It allows the analysis of thousands of genes in a single experiment[1]-[2].The raw data of a microarray experiment consists of a pair of 16 bits per pixel grayscale images. The processes involved in the formation of microarray images are shown in Fig.1. The result is a set of two intensity images, one for the expression level of the reference (control) tissue/cell (the Cy3 or the Green channel), and the other for the sample (experimental) tissue/cell (the Cy5 or the Red channel). Depending on the spacing between the spots and the overall size of the microarray, this procedure allows for a potentially high density of spots on the array (hence larger images), making it possible to measure expression profiles for tens of thousands of genes simultaneously.

To capture the large range of possible expression levels, the intensities are usually represented as a 16-bit integer. With pixel spacing of about 2 microns per pixel, at 16 bits per pixel, image sizes of up to 50MB are common[8]. For genome-wide expression analysis, with say 20,000 genes under 5000 experimental conditions, we are looking at about 191MB per image per channel.

The common approach towards the compression of microarray images has been based on image analysis for spot finding (gridding followed by segmentation) with the aim of separating the microarray image data into different streams based on pixel similarities [3]-[4]. Once separated, the streams are compressed individually, together with the segmentation information [7]-[9].

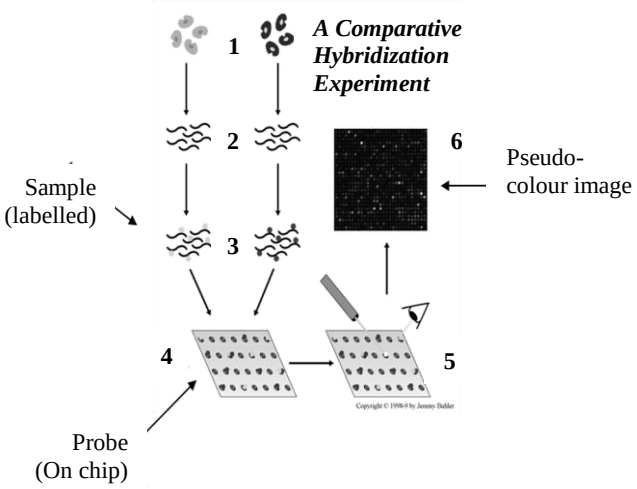

Fig.1. Formation of microarray images

The technique proposed in this paper is the best one currently available in terms of compression efficiency of microarray images. The method is based on arithmetic coding driven by image-independent multi-bitplane finite-context models. Basically, the image is compressed on a bitplane basis, going from the most significant bitplane to the least significant bitplane. The finite-context model used by the arithmetic encoder uses (causal) pixels from the bitplane under compression and also pixels from the bitplanes already encoded. The proposed method is compared with JPEG2000 [11], [12], JPEG-LS [13], [14], and JBIG [15], [16].

2. LITERATURE SURVEY

Some techniques have already been proposed for the lossy and lossless compression of microarray images.

Hua *et al.* [5] presented a transform-based coding technique. Initially, a segmentation is performed using the Mann–Whitney algorithm, and the segmentation information is encoded separately. Due to the threshold properties of the Mann–Whitney algorithm, the gridding stage is avoided. Then, a modified EBCOT (embedded block coding with optimized truncation) for handling arbitrarily shaped regions is used for encoding the spots and background separately, allowing lossy-to-lossless coding of background only or both background and spots.

The technique proposed by Jornsten *et al.* [6] is characterized by a first stage devoted to gridding and segmentation. Using the approximate center of each spot, a seeded region growing is performed for segmenting the spots. The segmentation map is encoded using chain-coding, whereas the interior of the regions are encoded using a modified version of the LOCO-I (LOw COmplexity LOssless COmpression for Images) algorithm (this is the algorithm behind the JPEG-LS coding standard), named SLOCO. Besides lossy-to-lossless capability, Jornsten's tech-

nique allows partial decoding, by means of independently encoded image blocks.

The compression method proposed by Faramarzpour *et al.* [8] starts by locating and extracting the microarray spots, isolating each spot into an individual region of interest (ROI). To each of these ROIs, a spiral path is adjusted such that its center coincides with the center of mass of the spot, with the idea of transforming the ROI into a 1-D signal with minimum entropy. Then, predictive coding is applied along this path, with a separation between residuals belonging to the spot area and those belonging to the background area.

Lonardi *et al.* [10] proposed lossless and lossy compression algorithms for microarray images (MicroZip). The method uses a fully automatic gridding procedure, similar to that of Faramarzpour's method, for separating spots from the background (which can be lossy compressed). Through segmentation, the image is split into two streams: foreground and background. Then, for entropy coding, each stream is divided into two 8 bit substreams and arithmetic encoded, with the option of being previously processed by a Burrows–Wheeler transform.

In [17], the compression performance of three Image coding standards in the context of microarray image compression: JPEG2000, JBIG, and JPEG-LS are studied. Since they rely on three different coding technologies, the performance cannot be evaluated on these standards, but also to collect hints regarding what might be the best coding technology regarding microarray image compression. In that study of three technologies evaluated (predictive coding in the case of JPEG-LS, transform coding in the case of JPEG2000 and context-based arithmetic coding in the case of JBIG), the technology behind JBIG seemed to be the most promising. In fact, JPEG-LS provided the highest compression, closely followed by JBIG. However, unlike JPEG2000 and JBIG, it does not provide lossy-to-lossless capabilities, a characteristic that might be of high interest, especially in the case where remote databases have to be accessed using transmission channels of reduced bandwidth.

Motivated by these observations, an efficient compression method is proposed for microarray images which is based on the same technology as JBIG but that, unlike JBIG, exploits inter-bitplane dependencies, providing coding gains in relation to JBIG. Designing contexts that gather information from more than one bitplane (multibitplane contexts) is not just a matter of joining more bits to the context, because for each new bit added the memory required doubles. Moreover, there is the danger of running into the context dilution problem, due to the lack of sufficient data for estimating the probabilities. Therefore, this extension to multibitplane contexts must be done carefully[20].

3. PROPOSED METHOD

3.1 IMAGE-INDEPENDENT CONTEXTS

In this paper, a lossless compression method for microarray images using arithmetic coding with a 3D context model is presented. This method was inspired on EIDAC [18],[19], which is a compression method used with success for coding simple images. The images are compressed on a bit-plane basis, starting from the most significant bit-plane (MSBP) and stopping at the least significant bit-plane (LSBP), or whenever a bit-plane requires more than one bit per pixel for encoding (the rest of the bit-planes are sent un-coded). The causal context model that drives the arithmetic encoder uses pixels both from the bit-plane currently being encoded (C_{intra}), in the order North, West, North-West, North-East (N, W, NW, NE) positions, Fig.2, and from the bit-planes already encoded(C_{inter}).

NW	N	NE
W	X	

Fig.2. The context configuration used in C_{intra}

When encoding the eight least significant bit-planes, the context model is only formed with pixels from the upper bitplanes. This procedure is done to avoid the degradation in compression due to, in general, the eight least significant bit-planes being close to random and, therefore, almost incompressible. As the method proceeds encoding the image, the average bit-rate obtained after encoding each bit-plane is monitored. If, for some bit-plane, the average bit-rate exceeds one bit per pixel, then we stop the encoding process, and the remaining bit-planes are saved without compression.

The context modeling part of EIDAC was designed mainly with the aim of compressing images with eight bitplanes or less, implying, at most, 19 bits of context. A straightforward extension to images with 16 bitplanes would require contexts of 27 bits. Essentially, the proposed technique differs from EIDAC in three aspects: 1) it was designed taking into account the specific nature of the images, keeping the size of the contexts limited to 21 bits; 2) it does not use the histogram packing procedure proposed for EIDAC because, generally, microarray images have dense intensity histograms; 3) it implements a rate-control mechanism that avoids producing average bitrates of more than one bit per pixel in bitplanes that are too noisy.

3.1.1 Algorithm for Finding Context bits:

Context model for Most Significant Bitplanes (BP15 to BP9)

For bitplane 15 to bitplane 9, the context bits are calculated from the previously encoded bitplanes by using the following procedure.

Step 1: Find the bitplane for the corresponding bit position by using bitget function.

Step 2: Find the context bits for each bit in the bitplane in row-scan manner.

i. Context positions falling outside the image at the image borders are considered as having zero value.

ii. The causal context model for the current bitplane is taken in the order of North, West, North-West, North-East (N, W, NW, NE) bit positions from the same bitplane.

Step 3: The causal contexts from previous bitplane are taken in the order North, West, North-West, North-East, X, East (N, W, NW, NE, X, E) bit positions, where X denotes the current bit position of bitplane.

Step 4: If the number of previous bitplanes > 1, proceed Step3 for the MSBP16 and the bitplane immediately previous to the current bitplane. Else proceed Step3 for the MSBP16.

Step 5: From the intermediate bitplanes, take the context bits as only the bit which is corresponding to current bit position of the considered bitplane.

Step 6: Repeat above Steps and find context bits for the bitplane until the last pixel is reached.

Context model for Least Significant Bitplane (BP8 to BP1)

For bitplane 8 to bitplane 1, the context bits are calculated from the previously encoded bitplanes by using the following procedure.

Step 1: Find the bitplane for the corresponding bit position by using bitget function.

Step 2: Find the context bits for each bit in the bitplane in row-scan manner.

Step 3: The causal contexts from the bitplane 16 are taken in the order of North, West, North-West, North-East, X, East (N, W, NW, NE, X, E) bit positions, where X denotes the current bit position of bitplane.

Step 4: From the intermediate bitplanes, take the context bits as only the bit which is corresponding to current bit position of the considered bitplane.

Step 5: Repeat above steps and find context bits for the bitplane until the last pixel is reached.

3.1.2 Context Length:

By applying this Image-independent context configuration procedure at five different compression stages, we get context bits of different length for each bitplane. (a) when encoding the most significant bitplane (four bits of context); (b) when encoding the second most significant bitplane (ten bits of context); (c) when encoding the third most significant bitplane (16 bits of context); (d) from the fourth until the eighth most significant bitplanes (17–21 bits of context); (e) the eight least significant bitplanes (13–20 bits of context).

3.2 FINITE-CONTEXT MODELS

The core of the proposed method consists of an adaptive finite-context model followed by arithmetic coding (Fig.3). A finite-context model of an information source assigns probability estimates to the symbols of an alphabet *A*, according to a conditioning context computed over a finite and fixed number *M*, of past outcomes (*M* order- finite-context model). In bitplane compression, $A = \{0, 1\}$ and $|A|=2$.In practice, the probability that the next outcome, x_{t+1} is "0" is obtained using the estimator given in Eq.(1)

$$P(x_{t+1}=0|c^t) = (n(0, c^t)+\partial) / (n(0, c^t)+ n(1, c^t)+2\partial), \quad (1)$$

where, $n(s, c^t)$ represents the number of times that, in the past, the information source generated symbol s having c^t as the conditioning context. The parameter $\partial > 0$, besides allowing fine tuning the estimator, avoids generating zero probabilities when a symbol is encoded for the first time. In our case, we used $\partial = 1$, which corresponds to Laplace's estimator (it can be seen as an initialization of all counters to one). The counters are updated each time a symbol is encoded. Since the context template is causal, the decoder is able to reproduce the same probability estimates without needing additional information.

3.2.1 Arithmetic Coding Algorithm with Scaling:

Step 1: Send the context bits and bits of corresponding pixel value to the encoder using cat function.

Step 2: Initially set the upper and lower limit of the tag interval as 0 and 1 respectively.

Step 3: Find the probability of occurrence of 0's and 1's in the sequence using probability density function.

Step 4: Send the bits of the sequence one by one into the encoder.

Step 5: Reset the upper and lower limits based on the current bit to be encoded.

There are 3 possibilities for new interval,

i. The interval is entirely confined to the lower half of unit interval [0, 0.5].

ii. The interval is entirely confined to the upper half of unit interval [0.5, 1.0].

iii. The interval straddles the midpoint of unit interval.

Step 6: If case 1 occurs, perform E_1 mapping and send 0 as scaling factor.

Step 7: If case 2 occurs, perform E_2 mapping and send 1 as scaling factor.

Step 8: If case 3 occurs, no need of rescaling.

Step 9: Repeat the above steps for entire bit sequence until the end of sequence is encountered.

3.3 MULTI-BITPLANE METHOD

The images are compressed on a bitplane basis, starting from the Most Significant Bitplane (MSBP) to the Least Significant Bitplane (LSBP) until the average bitrate becomes more than one bits per pixel. The compression procedure is given in Fig.4. First, the microarray input image is divided into 16 bitplanes. Then for each bitplane, find the best context configuration using image independent context models. The pixels in the bitplane are encoded using the arithmetic encoder by sending the calculated context bits. This process is repeated for bitplanes until the average bitrate exceeds 1 bpp and the remaining bitplanes are sent uncompressed.

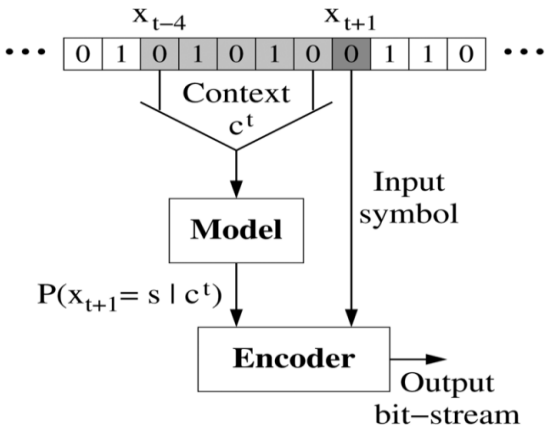

Fig.3. Finite-context model: the probability of the next outcome x_{t+1} is conditioned by the M last outcomes. In this example, M=5

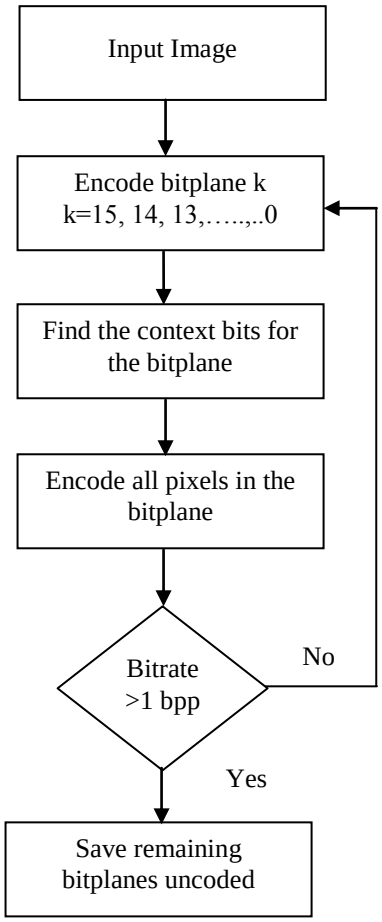

Fig.4. Encoding procedure of the proposed method

4. EXPERIMENTAL RESULTS

The compression method proposed in this paper was implemented in Matlab and evaluated using microarray images that have been collected from three different publicly available sources: 1). 32 images that we refer to as the APO_AI set; 2). 14 images from ISREC set; 3). three images from Micro-Zip image set[10]. Image size ranges from 1000 x 1000 to 5496 x 1956 pixels, i.e., from uncompressed sizes of about 2 MB to more than 20 MB (all images have 16 bits per pixel).

The average results obtained are tabulated in Table.1 and Table.2 which takes into account the different sizes of the images, i.e., they correspond to the total number of bits divided by the total number of image pixels.

Table.1 shows that the number of bits per pixel is reduced from 16 bits per pixel while encoding from most significant bit-planes to least significant bitplane of microarray images. The encoding process is stopped when average bit rate exceeds one.

Fig.5 represents the average number of bits per pixel required for encoding each bitplane of two images namely Def661cy5, Def665cy5 of ISREC image set of 1000 x 1000 pixels.The average bitrate exceeds one bpp at bitplane 2 and bitplane 4 for images Def661cy5 and Def665cy5 respectively

Table.2 shows that the number of bits per pixel is reduced from 16 bits per pixel while encoding most significant bit-planes of microarray images of Micro-Zip image set and the encoding process is stopped when average bit rate exceeds one.

Table.1. Encoding results of bit-planes of ISREC image set

ISREC / Bitplanes (bpp)	Def66Cy5.tif	Def665Cy5.tif
BP16	9.8	12.021
BP15	10.0808	12.8254
BP14	10.6584	13.521
BP13	10.7067	13.29
BP12	10.8889	12.6147
BP11	10.8271	12.8271
BP10	10.9526	12.145
BP9	10.5475	12.633

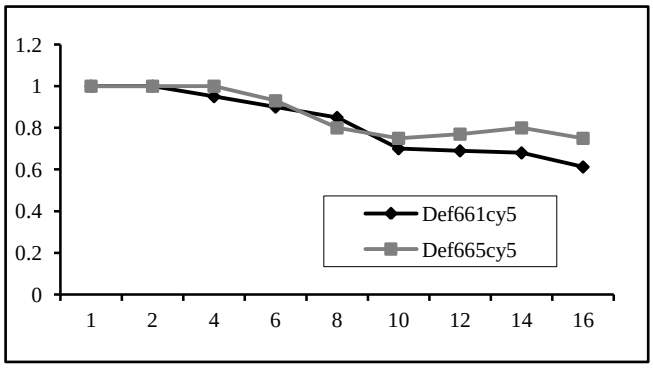

Fig.5. Bitplane vs bits per pixel

Table.2. Encoding results of bit-planes of Micro-Zip image set

MicroZip / Bitplanes (bpp)	Array1	Array2
BP16	10.256	6.525
BP15	10.545	7.091
BP14	11.756	8.587
BP13	12.789	8.524
BP12	11.897	7.765
BP11	12.241	8.490
BP10	11.567	8.786
BP9	11.125	8.640

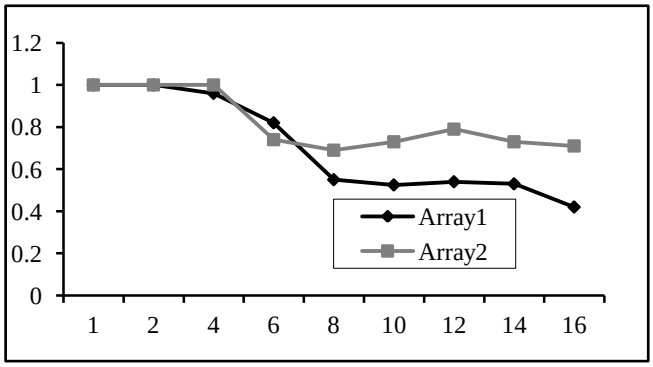

Fig.6. bitplane vs bits per pixel

Table.3. Comparison of Compression Results with Existing methods (Literature)

Image Set	JPEG2000(bpp)	JBIG (bpp)	JPEG-LS (bpp)	EIDAC (bpp)	Proposed method (bpp)	Compression ratio (%)
APO_AI	11.063	10.851	10.608	10.543	10.280	64
ISREC	11.366	10.925	11.145	10.446	10.199	63
MicroZip	9.515	9.297	8.974	8.837	8.840	55

Fig.6 depicts the average number of bits per pixel required for encoding each bitplane of two images namely array1, array of Micro-Zip image set. The average bitrate exceeds one bpp at bitplane 4 and bitplane 2 for images array1 and array2 respectively.

Table.3 tabulates the comparison of the compression results of the proposed method with the existing methods in the literature. The proposed method is 6.1% better than JBIG, 4.3% better than JPEG-LS and 8.2% better than lossless JPEG2000. The results of the existing methods namely JPEG-2000. JBIG, JPEG-LS and EIDAC are taken from the literature, for comparison.

5. CONCLUSION

In this paper, we presented an efficient method for lossless compression of microarray images, allowing progressive, lossy-to-lossless decoding .This method is based on bitplane compression using image-independent finite-context models and arithmetic coding. It does not require gridding and/or segmentation as most of the specialized methods that have been proposed so far. This may be an advantage if only compression is sought, since it reduces the complexity of the method. Moreover, since it does not require gridding, it is robust, for example, against layout changes in spot placement. Decoding is faster, because the decoder does not have to search for the best context.

The results obtained have been compared with other image coding standards in the literature like JBIG, JPEG2000, JPEG-LS, and EIDAC, based on image-independent context models. The results obtained shows that the proposed method has better compression performance in all three test sets.

REFERENCES

[1] S. K. Moore, "Making chips to probe genes," *IEEE Spectrum*, Vol. 38, No. 3, pp. 54–60, 2001.

[2] P. Hegde, R. Qi, K. Abernathy, C. Gay, S. Dharap, R. Gaspard, J. Earle-Hughes, E. Snesrud, N. Lee, and Q. John, "A concise guide to cDNA microarray analysis," *Biotechniques*, Vol. 29, No. 3, pp. 548–562, 2000.

[3] R. Jornsten, B. Yu, W. Wang, and K. Ramchandran, "Compression of cDNA and inkjet microarray images," *in Proc. IEEE Int. Conf. Image Proces., Rochester, NY*, Vol. 3, pp. 961–964, 2002.

[4] R. Jornsten, Y.Vardi, and C.-H. Zhang, "On the bitplane compression of microarray images", *Statistical Data Analysis based on the L1-norm and Related methods*, pp.415-425, 2002.

[5] J. Hua, Z. Xiong, Q. Wu, and K. Castleman, "Fast segmentation and lossy-to-lossless compression of DNA microarray images," *Proc. Workshop Genomic Signal Processing and Statistics (Gensips), Raleigh, NC*, 2002.

[6] R. Jornsten, W. Wang, B. Yu, and K. Ramchandran, "Microarray image compression: SLOCO and the effect of information loss," *Signal Processing*, Vol. 83, pp. 859–869, 2003.

[7] J. Hua, Z. Liu, Z. Xiong, Q.Wu, and K. Castleman, "Microarray basica: background adjustment, segmentation, image compression and analysis of microarray images" *Proc. IEEE Int. Conf. Image Process., Barcelona, Spain*, Vol. 1, pp. 585–588, 2003.

[8] N. Faramarzpour, S. Shirani, and J. Bondy, "Lossless DNA microarray image compression," *Proc. 37th Asilomar Conf. Signals, Syst., Comput.*, Vol. 2, pp. 1501–1504, 2003.

[9] N. Faramarzpour and S. Shirani, "Lossless and lossy compression of DNA microarray images," *Proc. Data Compression Conf., Snowbird, UT*, pp. 538–538, 2004.

[10] S. Lonardi and Y. Luo, "Gridding and compression of microarray images," *Proc. IEEE Computational Syst. Bioinformatics Conf., Stanford, CA*, pp. 122–130, 2004.

[11] Information Technology—JPEG 2000 Image Coding System ISO/IEC, 2000, ISO/IEC International Standard 15444-1, ITU-T Recommendation T.800.

[12] Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Process. Mag.*, Vol. 18, No. 5, pp. 36–58, 2001.

[13] Information Technology—Lossless and Near-Lossless Compression of Continuous-Tone Still Images ISO/IEC, 1999, ISO/IEC 14495-1 and ITU Recommendation T.87.

[14] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into jpegls," *IEEE Trans. Image Process.*, Vol. 9, No. 8, pp. 1309–1324, 2000.

[15] Information Technology – Coded Representation of Picture and Audio Information – Progressive Bi-Level Image Compression ISO/IEC, 1993, International Standard ISO/IEC 11544 and ITU-T Recommendation T.82.

[16] D. Salomon, "*Data Compression - The Complete Reference*", 2nd ed. New York: Springer, 2000.

[17] J. Pinho, A. R. C. Paiva, and A. J. R. Neves, "On the use of standards for microarray lossless image compression," *IEEE Trans. Biomed. Eng.*, Vol. 53, No. 3, pp.563–566, 2006.

[18] J. R. Neves and A. J. Pinho, "Lossless compression of microarray images," *Proc. IEEE Int. Conf. Image Process., Atlanta, GA*, pp. 2505–2508, 2006.

[19] Y. Yoo, Y. G. Kwon, and A. Ortega, "Embedded image-domain adaptive compression of simple images," *Proc. 32nd Asilomar Conf. Signals, Syst., Comput., Pacific Grove, CA*, Vol. 2, pp. 1256–1260, 1998.

[20] J. R. Neves and A. J. Pinho, "Lossless Compression of Microarray Images using Image Dependent Finite Context Model," *IEEE transactions on medical imaging*, Vol. 28, No. 2, pp.194 – 201, 2009.