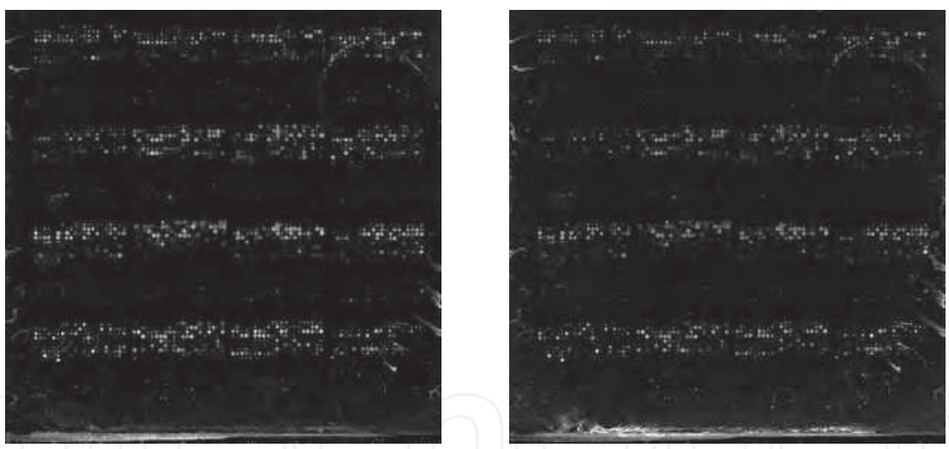
Compression of microarray images *

António J. R. Neves and Armando J. Pinho
Signal Processing Lab, DETI/IEETA, University of Aveiro
Portugal

1. Introduction

DNA microarrays have become a tool of paramount importance in the study of gene function, regulation, and interaction across large numbers of genes, and even entire genomes (Hegde et al., 2000; Moore, 2001). Microarray experiments generate pairs of 16 bits per pixel grayscale images (see Fig. 1, for an example). These images, which may require several tens of megabytes in order to be stored or transmitted, are analyzed by software tools that extract relevant information, such as the intensity of the spots and the background level. This information is then used for evaluating the expression level of individual genes (Hegde et al., 2000; Moore, 2001).



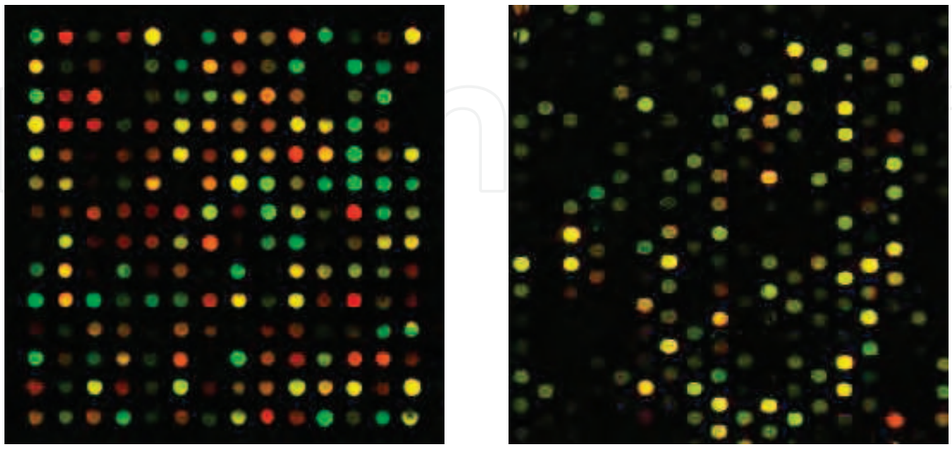
(a) Green channel (b) Red channel

Fig. 1. Example of a pair of images (1041 $\times$ 1044 pixels) that results from a microarray experiment.

The common approach for microarray compression has been based on image analysis for spot finding (griding followed by segmentation) with the aim of separating the microarray image data into different streams based on pixel similarities (Adjeroh et al., 2006; Faramarzpour and Shirani, 2004; Faramarzpour et al., 2003; Hua et al., 2003; 2002; Jörnsten et al., 2003; 2002a; Lonardi and Luo, 2004; Zhang et al., 2005). Once separated, the streams are compressed together with the segmentation information. A potential drawback of these segmentation based

*This work was supported in part by the FCT (Fundação para a Ciência e Tecnologia).

approaches is that different spot placements (e.g., non-rectangular) might compromise their performance. In fact, although initially the rectangular packing was the organization used for spot placement in microarrays, other non-rectangular packings have also been proposed (see Fig. 2).



(a) Rectangular packing (b) Orange packing

Fig. 2. Different spot packing: (a) Rectangular packing; (b) Orange packing (sample image from `http://microarray1k.aecom.yu.edu/`). Note that these images are not related. They serve just for illustrating different spot placements.

Although initially most of the specialized techniques for microarray image compression considered the lossy approach as a reasonable possibility (Faramarzpour and Shirani, 2004; Hua et al., 2003; 2002; Jörnsten et al., 2003; 2002a), the most recent methods address mainly reversible techniques (Faramarzpour et al., 2003; Lonardi and Luo, 2004; Zhang et al., 2005). Keeping the original images allows future re-analysis by possibly better algorithms. In fact, the analytic methods that are used for extracting information from the images are continuously being improved (Kothapalli et al., 2002; Leung and Cavalieri, 2003; Sasik et al., 2004). Also, as with other biomedical related data, legal issues might play a key role when choosing between maintaining or deleting the original data.

Recently, we have investigated methods for compressing microarray images that do not require spot segmentation. This new approach is based on arithmetic coding that is driven by image-dependent multi-bitplane finite-context models. Basically, the image is compressed on a bitplane basis, going from the most significant 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. To our knowledge, this technique is currently the best one available in terms of compression efficiency of microarray images (Neves and Pinho, 2009).

In this chapter, we start by describing the most important techniques for the lossless compression of microarray images that have been proposed in the literature. Then, we present a set of experiments that have been performed with the aim of providing a reference regarding the performance of standard image coding techniques, namely, lossless JPEG2000, JBIG and JPEG-LS, when applied to the lossless compression of microarray images. We proceed with the description of an image-independent multi-bitplane finite-context approach and we continue with the image-dependent version. Finally, we present experimental results that

illustrate the compression performance of the several approaches and we draw some conclusions.

2. Compression techniques for microarray images

In this section, we present the most important methods for compression of microarray images, namely, the works of Jörnsten et al. (2003), Hua et al. (2002), Faramarzpour et al. (2003), Lonardi and Luo (2004) and Zhang et al. (2005). Although all the methods presented in this section address the microarray compression problem using different approaches, some of the processing steps are common and similar to the ones depicted in Fig. 3.
All the methods start by segmenting the microarray images into *regions of interest* (ROIs) containing the spot and some surrounding background. Some methods go even further, separating the spot area from the background. However, the segmentation algorithm used in each method is different.

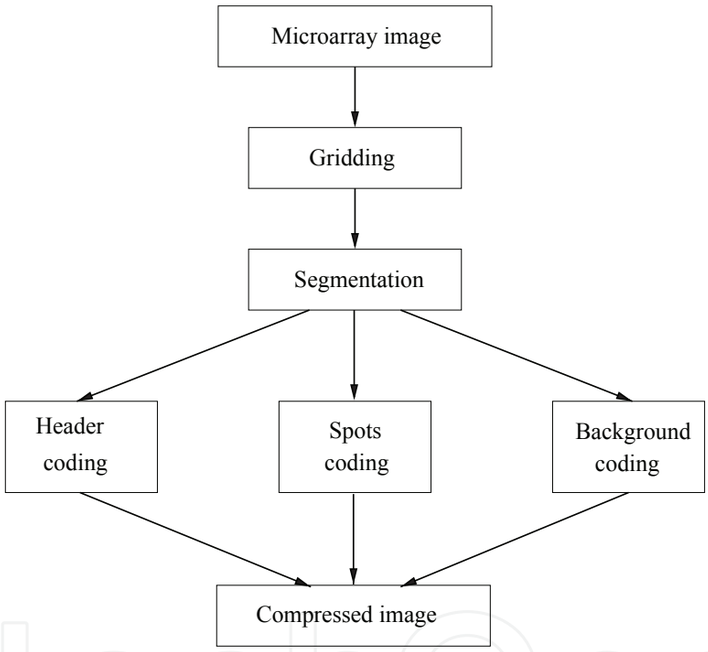


Fig. 3. The common processing steps of the compression methods presented in this section.

Through segmentation, it is possible to encode the spots and background separately. This is explicitly done in the works of Hua et al. (2003; 2002); Jörnsten et al. (2003); Jörnsten and Yu (2000; 2002); Jörnsten et al. (2002a;b); Lonardi and Luo (2004), and more implicilty in the work of Faramarzpour and Shirani (2004); Faramarzpour et al. (2003), because, in this case, the separation between the spot area and the background is performed only when the sequence is entropy encoded.
Almost all available methods have also a lossy compression version. These methods remove what is considered to be noise or redundant. Although this step sounds obvious, the question is "What should be considered noise or redundant?" Note that, in the context of microarray images, the background is very important for noise estimation, because the bias due to noise can be estimated and removed in the calculation of the gene expression level of each spot.

The technique proposed by Jörnsten et al. (2003) is characterized by a first stage devoted to griding 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 algorithm (LOw COmplexity LOssless COmpression for Images, the algorithm behind the JPEG-LS coding standard), named SLOCO. Besides lossy-to-lossless capability, Jörnsten's technique allows partial decoding by means of independently encoded image blocks.
Hua et al. (2002) 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 thresholding properties of the Mann-Whitney algorithm, the griding stage is avoided. Then, a modified EBCOT (Embedded Block Coding with Optimized Truncation) (Taubman and Marcellin, 2002) for handling arbitrarily shaped regions is used for encoding the spots and background separately, allowing lossy-to-lossless coding of background only (with the spots encoded in lossless mode) or both background and spots.
The compression method proposed by Faramarzpour et al. (2003) starts by locating and extracting the microarray spots, isolating each spot into an individual ROI. A spiral path is adjusted to each of these ROIs, such that its center coincides with the center of mass of the spot. The idea is to transform the ROI into an one-dimensional 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 and Luo (2004) proposed lossless and lossy compression algorithms for microarray images (MicroZip). The method uses a fully automatic griding 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 sub-streams and arithmetic encoded, with the option of being previously processed by a Burrows-Wheeler transform.
The method proposed by Adjeroh et al. (2006); Zhang et al. (2005) is based on PPAM (Prediction by Partial Approximate Matching). PPAM is an image compression algorithm which extends the PPM text compression algorithm, considering the special characteristics of natural images (Zhang et al., 2005). Initially, the microarray image is separated into background and foreground. Then, for each of these two components, the pixel representation is separated into its most significant and least significant parts. To compress the data, the most significant part is first processed by an error prediction scheme. The residuals are then encoded by the PPAM context model and encoder. The least significant part is encoded directly by the PPAM encoder and the segmentation information is saved without compression.

3. Standard image compression methods

JBIG, JPEG-LS and JPEG2000 are state-of-the-art standards for coding digital images. They have been developed with different goals in mind, being JBIG more focused on bi-level imagery, JPEG-LS dedicated to the lossless compression of continuous-tone images and JPEG2000 designed with the aim of providing a wide range of functionalities.
The JBIG standard (Joint Bi-level Image Experts Group) was issued in 1993 by ISO/IEC (International Organization for Standardization / International Electrotechnical Commission) and ITU-T (Telecommunication Standardization Sector of the International Telecommunication Union) for the progressive lossless compression of binary and low-precision gray-level images (typically, having less than 6 bits per pixel). The major advantages of JBIG over other

existing standards, such as FAX Group 3/4, are its capability of progressive encoding and its superior compression efficiency (Hampel et al., 1992; ISO/IEC, 1993; Netravali and Haskell, 1995; Salomon, 2000). The core of JBIG is an adaptive context-based arithmetic encoder, relying on 1024 contexts when operating in sequential mode or on low resolution layers of the progressive mode, or 4096 contexts when encoding high resolution layers. More recently, a new version, named JBIG2, has been published (ISO/IEC, 2000b), introducing additional functionalities to the standard, such as multipage document compression, two modes of progressive compression, lossy compression and differentiated compression methods for different regions of the image (e.g., text or halftones) (Salomon, 2000).
JPEG-LS was developed by the Joint Photographic Experts Group (JPEG) with the aim of providing a low complexity lossless image standard that could be able to offer better compression efficiency than lossless JPEG (ISO/IEC, 1999; Taubman and Marcellin, 2002; Weinberger et al., 2000). Part 1 of this standard was finalized in 1999. The core of JPEG-LS is based on the LOCO-I algorithm, that relies on prediction, residual modeling and context-based coding of the residuals. Most of the low complexity of this technique comes from the assumption that prediction residuals follow a two-sided geometric probability distribution and from the use of Golomb codes which are known to be optimal for this kind of distributions. Besides lossless compression, JPEG-LS also provides a lossy mode where the maximum absolute error can be controlled by the encoder. This is known as near-lossless compression or L_∞-constrained compression.
From the three image coding standards addressed in this section, JPEG2000 is the most recent one (ISO/IEC, 2000a; Taubman and Marcellin, 2002). Part 1 was published as an International Standard in the year 2000. It is based on wavelet technology and EBCOT coding of the wavelet coefficients, providing very good compression performance for a wide range of bitrates, including lossless coding. Moreover, JPEG2000 allows the generation of embedded code streams, meaning that from a higher bitrate stream it is possible to extract lower bitrate instances without the need for re-encoding. This property is of fundamental importance for progressive transmission, for example, over slow communication channels.
These three standard image encoders cover a great variety of coding approaches. In fact, whereas JPEG2000 is transform based, JPEG-LS relies on predictive coding, and JBIG relies on context-based arithmetic coding. This diversity in coding engines might be helpful for drawing conclusions regarding the appropriateness of each of these technologies for the case of microarray image compression.

3.1 Compression performance of the standards

Before trying to develop new compression methods, it is always useful to find out how existing compression standards behave on the class of images of interest. Therefore, for performing that assessment, we collected microarray images from three different publicly available sources: (1) 32 images that we refer to as the Apo AI set and which have been collected from `http://www.stat.berkeley.edu/users/terry/zarray/Html/index.html` (this set was previously used by Jörnsten et al. (2003); Jörnsten and Yu (2002)); (2) 14 images forming the ISREC set which have been collected from `http://www.isrec.isb-sib.ch/DEA/module8/P5_chip_image/images/`; (3) three images previously used to test MicroZip (Lonardi and Luo, 2004), which were collected from `http://www.cs.ucr.edu/~yuluo/MicroZip/`.

JBIG compression was obtained using version 1.6 of the JBIG Kit package[1], with sequential coding (-q flag). JPEG2000 lossless compression was obtained using version 5.1 of the JJ2000 codec with default parameters (lossless compression)[2]. JPEG-LS coding was obtained using version 2.2 of the SPMG JPEG-LS codec with default parameters[3]. For additional reference, we also give compression results using the popular compression tool GZIP (version 1.2.4).
Table 1 shows the compression results, in number of bits per pixel (bpp), where the first group of images corresponds to the Apo AI set, the second to the ISREC set and the third one to the MicroZip image set. Image size ranges from 1000×1000 to 5496×1956 pixels, i.e., from uncompressed sizes of about 2 megabytes to more than 20 megabytes (all images have 16 bits per pixel). The average results presented take 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.

Image set	Gzip	JPEG2000	JBIG	JPEG-LS
APO_AI	12.711	11.063	10.851	10.608
ISREC	12.464	11.366	10.925	11.145
Microzip	11.434	9.515	9.297	8.974
Average	**12.273**	**10.653**	**10.393**	**10.218**

Table 1. Compression results, in bits per pixel (bpp), using lossless JPEG2000, JBIG and JPEG-LS. For reference, results are also given for the popular compression tool GZIP.

The total average results show that gains of about 13.2%, 15.3% and 16.7%, in relation to GZIP compression, are attained respectively for lossless JPEG2000, JBIG and JPEG-LS, showing the superiority of image coding techniques over general purpose data compression methods in the task of compressing images. The average results by image set show that JPEG-LS provides the highest compression in the case of the Apo AI and MicroZip images, whereas JBIG gives the best results for the ISREC set. Lossless JPEG2000 is always slightly behind these two. It is interesting to note that the set for which JBIG gave the best results is also the one requiring more bits per pixel for encoding.

3.1.1 Sensitivity to noise

It has been noted by Jörnsten et al. (2003) that, in general, the eight least significant bitplanes of cDNA microarray images are close to random and, therefore, incompressible. Since this fact may result in some degradation in the compression performance of the encoders, we decided to address this problem and to study the effect of noisy bitplanes in the compression performance of the standards.
To perform this evaluation, we separated the images into a number p of most significant bitplanes and $16 - p$ least significant bitplanes. Whereas the p most significant bitplanes have been sent to the encoder, the $16 - p$ least significant bitplanes have been left uncompressed. This means that the bitrate of a given image is the sum of the bitrate generated by encoding the p most significant bitplanes plus the $16 - p$ bits concerning the bitplanes that have been left uncompressed.

[1] `http://www.cl.cam.ac.uk/~mgk25/jbigkit/`.
[2] `http://jj2000.epfl.ch`.
[3] The original website of this codec, `http://spmg.ece.ubc.ca`, is currently unavailable. However, it can be obtained from `ftp://www.ieeta.pt/~ap/codecs/jpeg_ls_v2.2.tar.gz`.

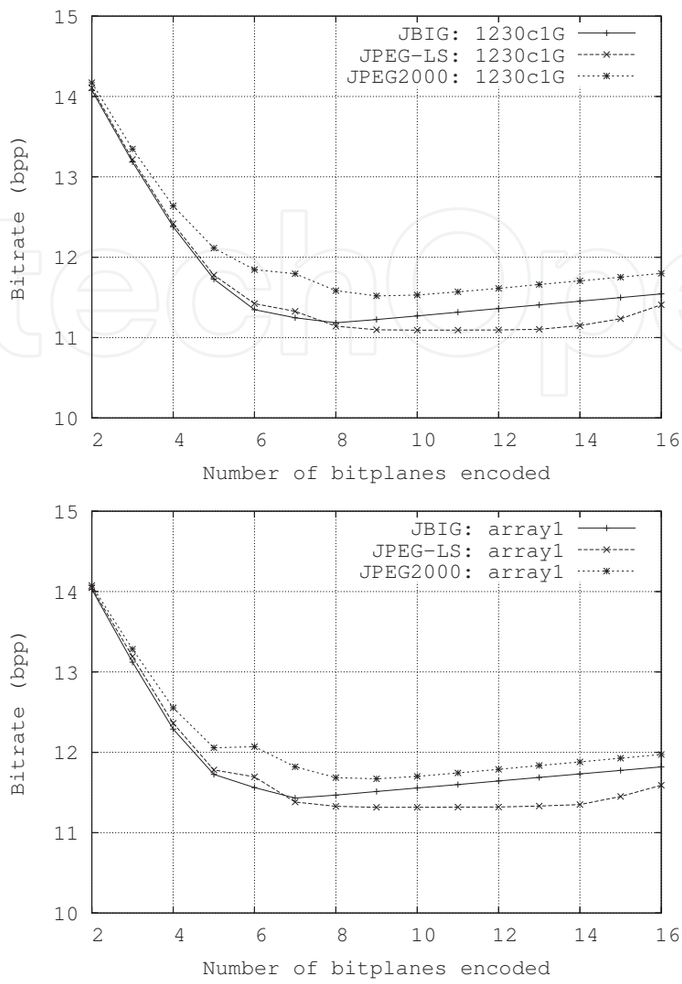


Fig. 4. Influence of noisy bitplanes in the performance of the standard encoding methods. The the curves indicate the bitrate obtained when only a given number p of the most significant bitplanes are sent to the encoder, whereas the other $16 - p$ bitplanes are left uncompressed.

Image set	JPEG2000		JBIG		JPEG-LS	
	8 bp	Best	8 bp	Best	8 bp	Best
Apo_AI	10.940	10.790	10.510	10.507	10.523	10.433
ISREC	11.100	10.954	10.607	10.583	10.838	10.713
MicroZip	9.918	9.321	9.506	9.030	9.588	8.912
Average	**10.661**	**10.376**	**10.224**	**10.073**	**10.302**	**10.026**

Table 2. Average compression results, in bits per pixel (bpp), when a number of bitplanes is left uncompressed. The columns labeled "8 bp" provide results for the case where only the 8 most significant bitplanes have been encoded and the 8 least significant bitplanes have been left uncompressed. The column named "Best" contains the results for the case where the separation of most and least significant bitplanes has been optimally found.

Figure 4 depicts bitrate curves, as a function of p, for two different images, "1230c1G" and "array1". As can be observed, the best bitrate is generally not met when compressing all 16 bitplanes, but instead when some of the least significant bitplanes are left uncompressed. However, the value of the optimum value of p, p_{opt}, varies not only from image to image, but also from one encoder to the other. In fact, for the Apo AI set, which is characterized by the most regular value of p_{opt}, JBIG is the encoder with the highest value of p_{opt} (around 8), then comes lossless JPEG2000 (around 10) and, finally, JPEG-LS (around 13). This result is not surprising, since JBIG encodes the bitplanes independently. Therefore, without being able to get information from other bitplanes, it is natural that JBIG starts considering bitplanes as "noise" earlier than the other encoders. Moreover, this can also be the justification for its better performance in the ISREC set, because it is the most noisy.
Table 2 compares average results for the three set of images regarding two situations: (1) the image is divided into the eight most significant bitplanes (which are encoded) and the eight least significant bitplanes (which are left uncompressed); (2) the optimum value of p is determined for each image. From this table, and comparing with the Table 1, we can see that, in fact, this splitting operation can provide some additional compression gains. The best results attained provided improvements of 3.1%, 2.6% and 1.9% respectively for JBIG, lossless JPEG2000 and JPEG-LS.
However, finding the right value for p may require as many as 16 iterations of the compression phase in order to find it. Moreover, from the results shown in Table 2, we can see that a simple separation of the bitplanes in an upper and lower half may improve the compression in some cases (Apo AI and ISREC image sets), but may also produce the opposite result (MicroZip image set).

3.1.2 Lossy-to-lossless compression

From the point of view of compression efficiency, and taking into account the results presented in Table 1, JPEG-LS is the overall best lossless compression method, followed by JBIG and lossless JPEG2000. The difference between JPEG-LS and lossless JPEG2000 is about 4.1% and between JPEG-LS and JBIG is only 1.7%. However, the better compression performance provided by JPEG-LS can be overshadowed by a potentially important functionality provided by the other two standards, which is progressive, lossy-to-lossless, transmission.
In the case of lossless JPEG2000, this functionality is basically a by-product of the multiresolution wavelet technology used in its encoding engine and also due to a strategy of encoding the information in layers (Taubman and Marcellin, 2002). In the case of JBIG, this property comes from two different sources. On one hand, images with more that one bitplane are encoded using a bitplane-by-bitplane coding approach. This provides a kind of progressive transmission, from most to least significant bitplanes, where the precision of the pixels is improved for each added bitplane. Moreover, this technique produces a reduction of the L_∞ error by a factor of two for each additional bitplane. On the other hand, JBIG permits the progressive transmission of each bitplane by progressively increasing its spatial resolution (ISO/IEC, 1993; Salomon, 2000). However, the compression results that we present in Table 1 do not take into account the additional overhead implied by this encoding mode of JBIG (we used the -q flag of the encoder, which disables this mode).
In Fig. 5, we present rate-distortion curves for two images, "1230c1G" and "array1", obtained with the lossless JPEG2000 and JBIG coding standards, and according to two error metrics: L_2-norm (root mean squared error) and L_∞-norm (maximum absolute error). Regarding the L_2-norm, we observe that lossless JPEG2000 provides slightly better rate-distortion results for

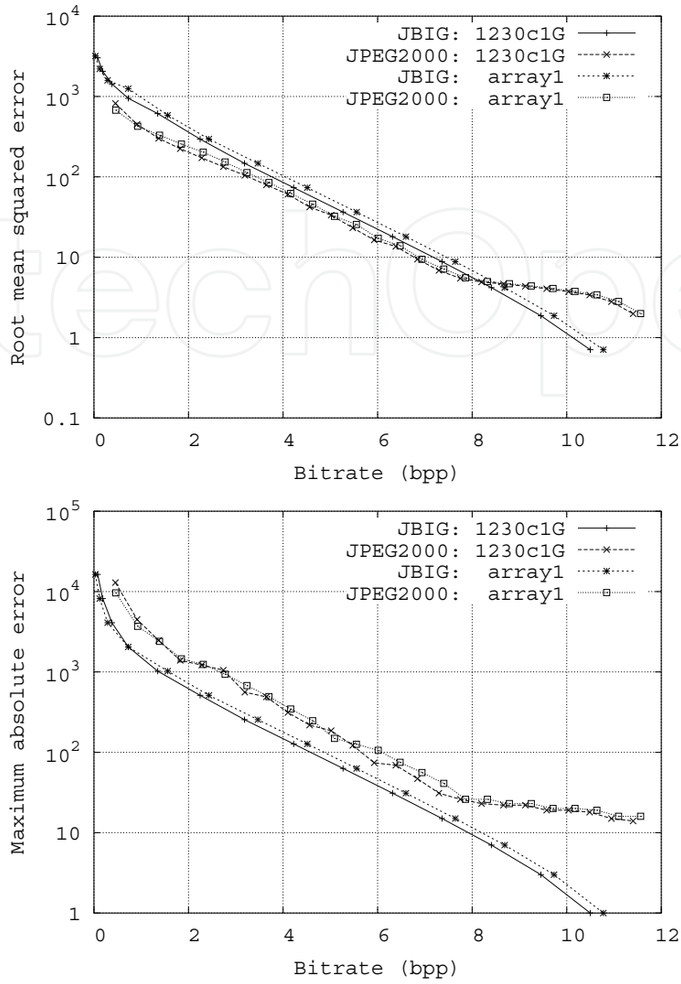


Fig. 5. Rate distortion curves showing the performance of lossless JPEG2000 and JBIG in a lossy-to-lossless mode of operation. Results are given both for the L_2 (root mean squared error) and L_∞ (maximum absolute error) norms.

bitrates less than 8 bpp. For higher bitrates, this codec exhibits a sudden degradation of the rate-distortion. We believe that this phenomenon is related to the default parameters used by the encoder, which might not be well suited for images having 16 bits per pixel, such as those of the microarrays. Moreover, we think that a careful setting of these parameters may lead to improvements in the rate-distortion of JPEG2000 for bitrates higher than 8 bpp, although we consider this tuning a problem that is beyond the scope of this work.

With respect to the L_∞-norm, we observe that JBIG is the one with the best rate-distortion performance. In fact, due to its bitplane-by-bitplane approach, it guarantees an exponential and upper bounded decrease of the maximum absolute error. The upper bound of the error is given by $2^{(16-p)} - 1$, where p is the number of bitplanes already decoded. Contrarily, lossless JPEG2000 cannot guarantee such bound, which may be a major drawback in some cases. Finally, we note that the sudden deviation of the lossless JPEG2000 curves around bitrates of 8 bpp is probably related to the same problem pointed out earlier for the case of the L_2-norm.

3.2 Conclusions

The main objective of this section was to provide a set of comprehensive results regarding the lossless compression of microarray images by state-of-the-art image coding standards, namely, lossless JPEG2000, JBIG and JPEG-LS. In order to facilitate future comparisons by other researchers, we collected a total of 49 microarray images available from the Internet. We believe that the development of specialized compression techniques should be supported by a preliminary study of the performance provided by well established methods and, particularly, by those that are standards. Only after making such study it is possible to be in a comfortable position for arguing about the relevance of some specialized technique.

From the experimental results obtained, we conclude that JPEG-LS gives the best lossless compression performance. However, it lacks lossy-to-lossless capability, which may be a decisive functionality if remote transmission over possibly slow links is a requirement. Complying to this requirement we find JBIG and lossless JPEG2000, lossless JPEG2000 being the best considering rate-distortion in the sense of the L_2-norm and JBIG the most efficient when considering the L_∞-norm. Moreover, JBIG is consistently better than lossless JPEG2000 regarding lossless compression ratios. Also, JBIG is the method that can benefit most from a correct separation of most significant bitplanes that are encoded and least significant bitplanes that are left uncompressed (it gained 3.1%), and it is also the coding technique that, due to the bitplane-by-bitplane coding, can search for the optimum point of separation on-the-fly. In fact, this can be done by monitoring the bitrate resulting from the compression of each bitplane, and stop doing compression when this value is over 1 bpp. As a final conclusion, and according to what we presented in this section, it is our opinion that the technology behind JBIG seems to be the most appropriate for microarray image coding.

4. Compression of microarray images using finite-context models and arithmetic coding

4.1 Finite-context models

The core of the methods proposed in the remainder of this chapter consists of an adaptive finite-context model followed by arithmetic coding. A finite-context model (see Fig. 6) of an information source assigns probability estimates to the symbols of an alphabet $\mathcal{A}$, according to a conditioning context computed over a finite and fixed number, M, of past outcomes (order-M finite-context model) (Rissanen, 1983; Rissanen and Langdon, Jr., 1981; Sayood, 2000). At time t, we represent these conditioning outcomes by $c^t = x_{t-M+1}, \ldots, x_{t-1}, x_t$. The number of conditioning states of the model is $|\mathcal{A}|^M$, dictating its complexity (or model cost). In our case, $\mathcal{A} = \{0, 1\}$ and, therefore, $|\mathcal{A}| = 2$.

In practice, the probability that the next outcome, x_{t+1}, is "0" is obtained using the estimator

$$P(x_{t+1} = 0|c^t) = \frac{n(0, c^t) + \delta}{n(0, c^t) + n(1, c^t) + 2\delta}, \tag{1}$$

where $n(s, c^t)$ represents the number of times that, in the past, the information source generated symbol $s \in \mathcal{A}$ having c^t as the conditioning context. The parameter $\delta > 0$, besides allowing fine tuning the estimator, avoids generating zero probabilities when a symbol is encoded for the first time. In our case, we used $\delta = 1$, 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.

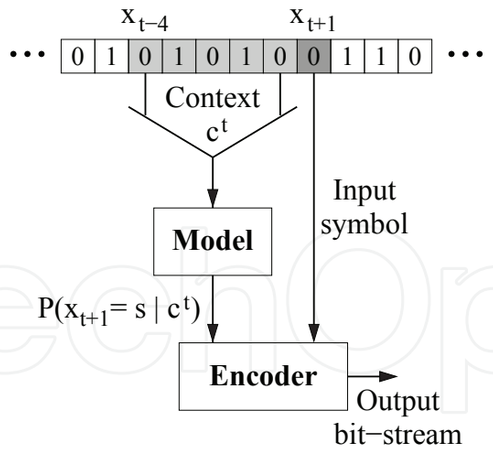


Fig. 6. Finite-context model: the probability of the next outcome, x_{t+1}, is conditioned by the M last outcomes. In this example, $M = 5$.

Context, c^t	$n(0, c^t)$	$n(1, c^t)$	$n(0, c^t) + n(1, c^t)$
00000	23	41	64
00001	16	6	22
00010	19	30	49
00011	34	42	76
00100	36	17	53
⋮	⋮	⋮	⋮
11111	8	2	10

Table 3. Simple example illustrating how finite-context models are implemented. The rows of the table represent a probability model at a given instant t. In this example, the particular model that is chosen for encoding a symbol depends on the last five encoded symbols (order-5 context).

Table 3 shows an example of how a finite-context is typically implemented. In this example, an order-5 finite-context model is presented. Each row represents a probability model that is used to encode a given symbol according to the last encoded symbols (five in this example). Therefore, if the last symbols were "00010", i.e., $c^t = 00010$, then the model communicates the following probability estimates to the arithmetic encoder: $P(0|00010) = 19/49$ and $P(1|00010) = 30/49$.

The block denoted "Encoder" in Fig. 6 is an arithmetic encoder. It is well known that practical arithmetic coding generates output bit-streams with average bitrates almost identical to the entropy of the model (Bell et al., 1990; Salomon, 2000; Sayood, 2000). In our case, the theoretical bitrate average (entropy) of the model after encoding N symbols is given by

$$H_N = -\frac{1}{N} \sum_{t=0}^{N-1} \log_2 P(x_{t+1} = s|c^t) \quad \text{bps}, \tag{2}$$

where "bps" stands for "bits per symbol". Since we are dealing with images, instead of using the generic "bps" measure we use "bpp", which stands for "bits per pixel". Recall that the

entropy of any sequence of two symbols is limited to 1 bps, a value that is achieved when the symbols are independent and equally likely.

4.2 Image-independent contexts

In Section 3, we presented a study of the compression performance of three image coding standards in the context of microarray image compression: JPEG2000, JBIG and JPEG-LS. Since they rely on three different coding technologies, we were able not only to evaluate the performance of each of these standards, but also to collect hints regarding what might be the best coding technology regarding microarray image compression. In that study, we concluded that from the 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, specially in the case where remote databases have to be accessed using transmission channels of reduced bandwidth. Moreover, with JBIG, the image bitplanes are compressed independently, suggesting the existence of some room for improvement.

Motivated by these observations, we developed a compression method 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 (Neves and Pinho, 2006). Designing contexts that gather information from more than one bitplane (multi-bitplane 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 multi-bitplane contexts must be done carefully.

The method proposed by Neves and Pinho (2006) was inspired by EIDAC (Yoo et al., 1998), a compression method that has been used with success for coding images with a reduced number of intensities (simple images). The images are compressed on a bitplane basis, from the most to the least significant bitplane. The causal finite-context model that drives the arithmetic encoder uses pixels both from the bitplane currently being encoded and from the bitplanes already encoded. As encoding proceeds, the average bitrate obtained after encoding each bitplane is monitored. If, for some bitplane, the average bitrate exceeds one bit per pixel, then the encoding process is stopped and the remaining bitplanes are saved without compression. The encoding procedure is outlined in Fig. 7.

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, i.e., at least $2 \times 2^{27} = 2^{28}$ counters. Essentially, the technique proposed by Neves and Pinho (2006) 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 (this is a common characteristic of the least significant bitplanes of microarray images (Jörnsten et al., 2003)).

As we mentioned before, choosing the context template for a multi-bitplane image is a critical task, requiring tradeoffs involving aspects such as the maximum size of the context, the problem of context dilution and the placement of the context bits such that the maximum informa-

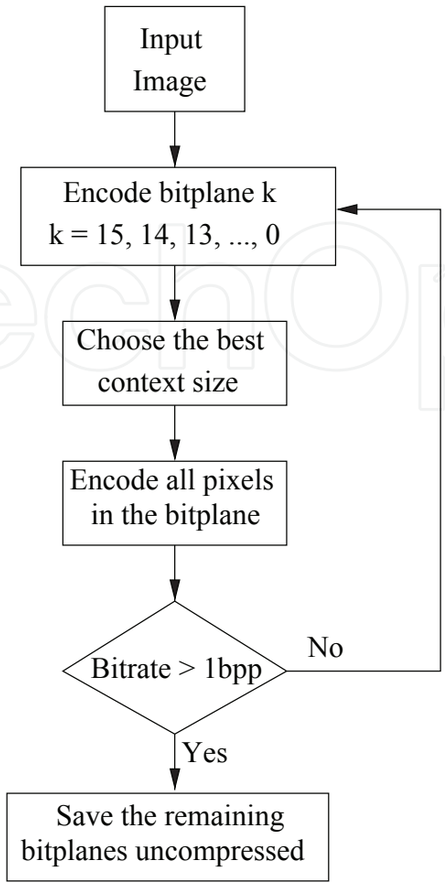


Fig. 7. Encoding procedure of the method proposed by Neves and Pinho (2006). The choice of the context shape is based on Fig. 8. Note that, being a bitplane based encoder, it is possible to monitor the bitrate used to encode each bitplane.

tion can be collected. This work was done in (Neves and Pinho, 2006) mainly using a trial and error procedure, leading to the image-independent context configuration displayed in Fig. 8. Note that, when encoding the eight least significant bitplanes, the finite-context model is only formed with pixels from the higher numbered bitplanes. This specific context configuration together with the rate-control mechanism avoids the degradation in compression rate when there are bitplanes that are close to random and, therefore, are almost incompressible.
Although being able to provide state-of-the-art compression results, the method proposed in (Neves and Pinho, 2006) could be improved. In fact, due to its image-independent nature, and despite being designed for a specific type of images (microarrays), the context configuration depicted in Fig. 8 resulted from a complicated process that tried to balance the inevitable particularities among the images. From the point of view of a single image, this context configuration might seem overkill, i.e., a smaller context might suffice. However, it is needed for satisfying the ensemble of images. This observation motivated the image-dependent context-modeling approach that we describe in the next section.

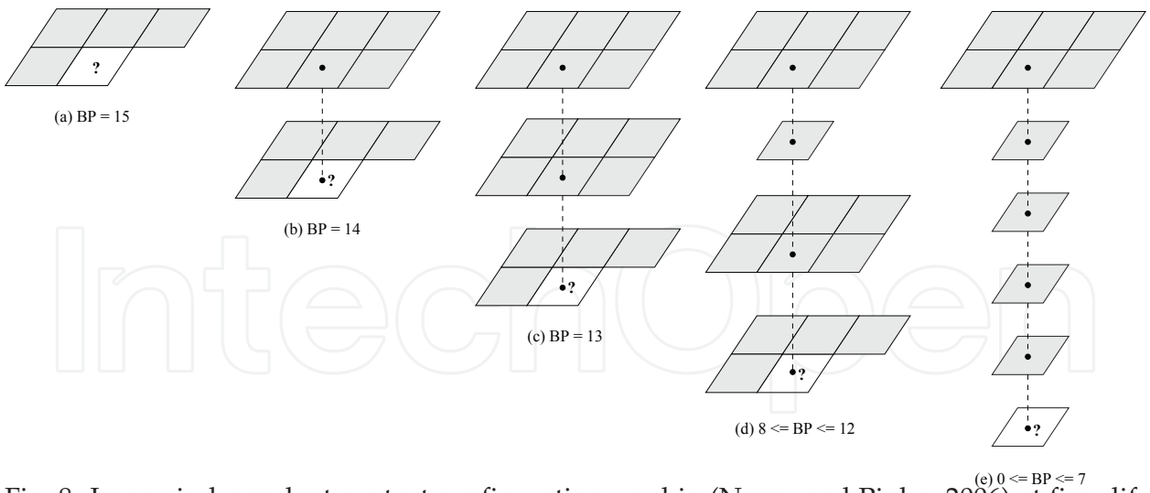
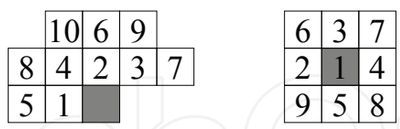


Fig. 8. Image-independent context configuration used in (Neves and Pinho, 2006) at five different compression stages: (a) when encoding the most significant bitplane (four bits of context); (b) when encoding the second most significant bitplane (ten bits of context); (c) when encoding the third most 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). Context positions falling outside the image at the image borders are considered as having zero value.

5. Image-dependent finite-context models

Instead of using the image-independent context model presented in Fig. 8, the algorithm that we describe in this section tries to find the "best" context configuration to encode the current bitplane, based on the templates depicted in Fig. 9. The test of all possible context configurations is a hard task, virtually impossible, due to the huge number of possibilities. To overcome this drawback, we developed a greedy approach that we explain next.



(a) (b)

Fig. 9. (a) The template used for growing the context at the level of the bitplane currently being encoded; (b) The template used for growing the context corresponding to the bitplanes already encoded.

Before encoding a bitplane, p, the algorithm constructs an appropriate context configuration through an iterative process (note that bitplanes are numbered from 0, the least significant bitplane, to 15, the most significant bitplane). In each iteration, an additional context bit is tested in each of the $16 - p$ possible locations, one in each of the $16 - p$ context bitplanes that are available. In a given context bitplane, the additional bit can only be inserted in position k of the corresponding template displayed in Fig. 9 if all positions $i < k$ belong already to the best context configuration found so far. This means that the part of the context belonging to a given bitplane can grow only according to the pixel numbering shown in Fig. 9. The template

in Fig. 9(a) is used for the context bitplane p, whereas the template in Fig. 9(b) applies to the remaining context bitplanes, i.e., from bitplane $p+1$ to bitplane 15.
After performing an iteration, the new context bit is assigned to the position where the largest improvement in the compression performance of bitplane p occurred. If none of the possible $16-p$ context bit positions were able to improve the compression, then the search stops and the context configuration found so far is used for encoding the bitplane p. Otherwise, a new context bit is tested. This iterative process proceeds while the new context bit is able to improve the compression performance of bitplane p or until the maximum context depth is reached. For the results presented in this section, we used a maximum of 20 context bits. Figure 10 presents an example of the context configuration obtained with this process for some of the bitplanes of the image "1230c1G" (APO_AI image set).
The configuration of the context bits for a particular bitplane, p, can be communicated to the decoder using approximately $4(16-p)$ bits. Note that the maximum number of context bits per context bitplane is less that 16 (see Fig. 9) and, therefore, can be represented in four bits. Hence, the total overhead regarding the image-dependent contexts is just some tens of bytes. The algorithm is outlined next.

```
bestCtx := 0-order context;
bestRate := rate for encoding bitplane
 with 0-order context;
do
  improved := FALSE;
  for p := bitplane to be encoded, 15
    if p = bitplane to be encoded
      add bit according to Fig. 9(a);
    else
      add bit according to Fig. 9(b);
    end
    rate := rate for encoding bitplane
     using current context;
    if rate < bestRate
      bestCtx := current context;
      bestRate := rate;
      improved := TRUE;
    end
    remove bit added above;
  end
while size of bestCtx < 20 AND improved
```

Being a greedy approach, it is not guaranteed that the optimum is found. In fact, as can be seen, for each context bitplane the context can only grow according to a predefined order which is given by the pixel numbering associated to the templates of Fig. 9. This limits the number of degrees of freedom of the search process, reducing the probability of finding the optimum configuration, but, on the other hand, also allowing running this procedure in a reasonable time.
In order to further accelerate the process of choosing these image-dependent contexts, and due to the highly structured nature of microarray images, we developed another version of the algorithm where only a small region of the image is used for constructing the contexts. Using this faster approach, the results obtained for a region of 256×256 pixels have been slightly worse. However, we verified a significant reduction in the time spent. In a 2 GHz Pentium 4 computer with 512 MBytes of memory, the MicroZip test set (three images totaling

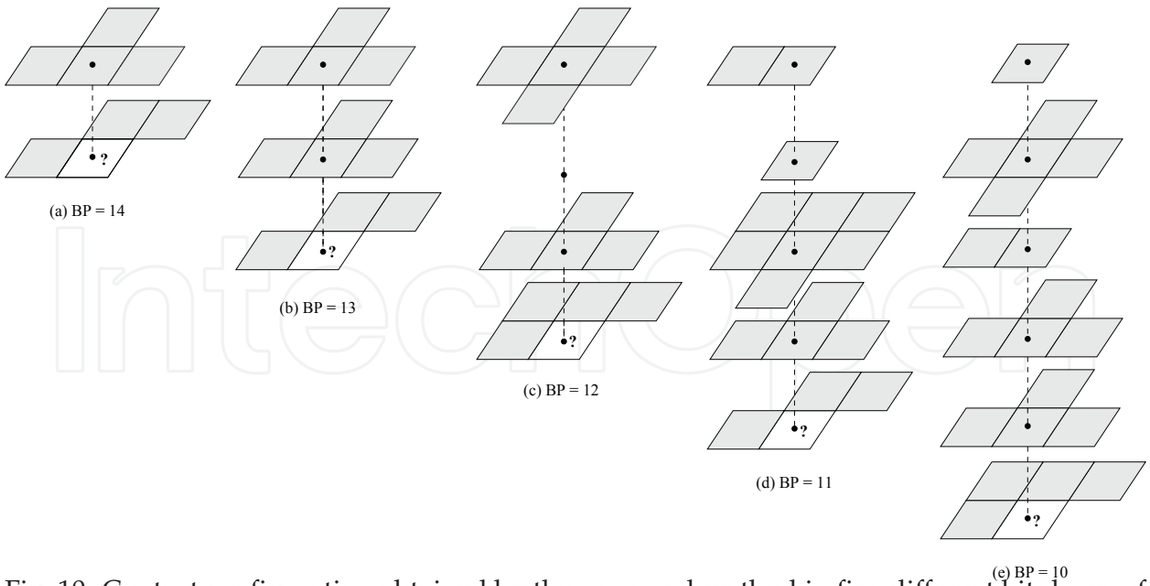


Fig. 10. Context configuration obtained by the proposed method in five different bitplanes of the image "1230c1G": (a) when encoding bitplane 14 (seven bits of context); (b) when encoding bitplane 13 (11 bits of context); (c) when encoding bitplane 12 (13 bits of context); (d) when encoding bitplane 11 (17 bits of context); (e) when encoding bitplane 10 (20 bits of context). Context positions falling outside the image at the image borders are considered as having zero value.

approximately 21 million pixels) required about 220 minutes to compress when the whole image was used to performed the search. When we used a region of 256×256 pixels, it required approximately 6 minutes to compress the MicroZip test set (about 2 minutes more than the image-independent approach). These three images have sizes of 1916×1872, 5496×1956 and 3625×1929 pixels. Decoding is faster, because the decoder does not have to search for the best context: that information is embedded in the bitstream.

6. Experimental results

Table 4 shows the average compression results, in bits per pixel, for the three sets of images described previously (see Section 3). In this table, we present experimental results of both the image-independent and the image-dependent approaches. We also include results obtained with SPIHT (Said and Pearlman, 1996)[4] and EIDAC (Yoo et al., 1998).

Comparing with the results presented in Table 1, we can see that the fast version of the image-dependent method (indicated as "256×256" in the table) is 6.3% better than JBIG, 4.7% better than JPEG-LS and 8.6% better than lossless JPEG2000. It is important to remember that JPEG-LS does not provide progressive decoding, a characteristic that is intrinsic to the image-dependent multi-bitplane finite-context method and also to JPEG2000 and JBIG. From the results presented in Table 4, it can also be seen that using an area of 256×256 pixels in the center of the image for finding the context, instead of the whole image, leads to a small degradation in the performance (about 0.3%), showing the appropriateness of this approach.

[4] SPIHT codec from `http://www.cipr.rpi.edu/research/SPIHT/` (version 8.01).

Image set	SPIHT	EIDAC	Image independent	Image-dependent	
				256×256	Full
APO_AI	10.812	10.543	10.280	10.225	10.194
ISREC	11.098	10.446	10.199	10.198	10.158
MicroZip	9.198	8.837	8.840	8.667	8.619
Average	**10.378**	**10.005**	**9.826**	**9.741**	**9.708**

Table 4. Average compression results, in bits per pixel, using SPIHT, EIDAC, the image-independent and the image-dependent methods. The "256 × 256" column indicates results obtained with a context model adjusted using only a square of 256 × 256 pixels at the center of the microarray image, whereas "Full" indicates that the search was performed in the whole image. The average results presented take 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 5 confirms the performance of the image-dependent method relatively to two recent specialized methods for compressing microarray images: MicroZip (Lonardi and Luo, 2004) and Zhang's method (Adjeroh et al., 2006; Zhang et al., 2005). As can be observed, the image-dependent multi-bitplane finite-context method provides compression gains of 9.1% relatively to MicroZip and 6.2% in relation to Zhang's method, on a set of test images that has been used by all these methods.

Images	MicroZip	Zhang	Image independent	Image-dependent	
				256×256	Full
array1	11.490	11.380	11.105	11.120	11.056
array2	9.570	9.260	8.628	8.470	8.423
array3	8.470	8.120	7.962	7.717	7.669
Average	**9.532**	**9.243**	**8.840**	**8.667**	**8.619**

Table 5. Compression results, in bits per pixel, using two specialized methods, MicroZip and Zhang's method, the image-independent method and the image-dependent method. The "256 × 256" column indicates results obtained with a context model adjusted using only a square of 256 × 256 pixels at the center of the microarray image, whereas "Full" indicates that the search was performed in the whole image.

Figure 11 shows, for three different images, the average number of bits per pixel that are needed for representing each bitplane. As expected, this value generally increases when going from most significant bitplanes to least significant bitplanes. For the case of images "`Def661Cy3`" and "`1230c1G`", it can be seen that the average number of bits per pixel required by the eight least significant bitplanes is close to one, as pointed out by Jörnsten et al. (2003). However, image "`array3`" shows a different behavior. Because this image is less noisy, the compression algorithm is able to exploit redundancies even in lower bitplanes. This is done without compromising the compression efficiency of noisy images, due to the mechanism that monitors and controls the average number of bits per pixel required for encoding each bitplane.

The maximum number of context bits that we allowed for building the contexts was limited to 20. Since the coding alphabet is binary, this implies, at most, $2 \times 2^{20} = 2\,097\,152$ counters that can be stored in approximately 8 MBytes of computer memory. In a 2 GHz Pentium 4

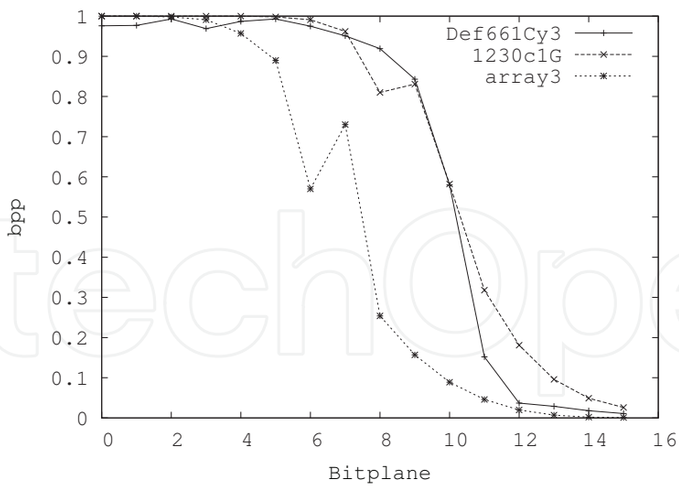


Fig. 11. Average number of bits per pixel required for encoding each bitplane of three different microarray images (one from each test set).

computer with 512 MBytes of memory, the image-dependent algorithm required about six minutes to compress the MicroZip test set (note that this compression time is only indicative, because the code has not been optimized for speed). Decoding is faster, because the decoder does not have to search for the best context. Just for comparison, the codecs of the compression standards took approximately one minute to encode the same set of images.

7. Conclusions

The use of microarray expression data in state-of-the-art biology has been well established. 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 work, we have studied the problem of coding this type of images.

We presented a set of comprehensive results regarding the lossless compression of microarray images by state-of-the-art image coding standards, namely, lossless JPEG2000, JBIG and JPEG-LS. From the experimental results obtained, we conclude that JPEG-LS gives the best lossless compression performance. However, it lacks lossy-to-lossless capability, which may be a decisive functionality if remote transmission over possibly slow links is a requirement. Complying to this requirement we find JBIG and lossless JPEG2000, lossless JPEG2000 being the best considering rate-distortion in the sense of the L_2-norm and JBIG the most efficient when considering the L_∞-norm. Moreover, JBIG is consistently better than lossless JPEG2000 regarding lossless compression ratios.

Motivated by these findings, we have developed efficient methods for lossless compression of microarray images, allowing progressive, lossy-to-lossless decoding. These methods are based on bitplane compression using image-independent or image-dependent finite-context models and arithmetic coding. They do not require griding and/or segmentation as most of the specialized methods that have been proposed do. This may be an advantage if only compression is sought, since it reduces the complexity of the method. Moreover, since they do not require griding, they are robust, for example, against layout changes in spot placement.

The results obtained by the multi-bitplane context-based methods have been compared with the three image coding standards and with two recent specialized methods: MicroZip and Zhang's method. The results obtained show that these new methods have better compression performance in all image test sets used.

8. References

Adjeroh, D., Y. Zhang, and R. Parthe (2006, February). On denoising and compression of DNA microarray images. *Pattern Recognition 39*, 2478–2493.

Bell, T. C., J. G. Cleary, and I. H. Witten (1990). *Text compression*. Prentice Hall.

Faramarzpour, N. and S. Shirani (2004, March). Lossless and lossy compression of DNA microarray images. In *Proc. of the Data Compression Conf., DCC-2004*, Snowbird, Utah, pp. 538.

Faramarzpour, N., S. Shirani, and J. Bondy (2003, November). Lossless DNA microarray image compression. In *Proc. of the 37th Asilomar Conf. on Signals, Systems, and Computers, 2003*, Volume 2, pp. 1501–1504.

Hampel, H., R. B. Arps, C. Chamzas, D. Dellert, D. L. Duttweiler, T. Endoh, W. Equitz, F. Ono, R. Pasco, I. Sebestyen, C. J. Starkey, S. J. Urban, Y. Yamazaki, and T. Yoshida (1992, April). Technical features of the JBIG standard for progressive bi-level image compression. *Signal Processing: Image Communication 4*(2), 103–111.

Hegde, P., R. Qi, K. Abernathy, C. Gay, S. Dharap, R. Gaspard, J. Earle-Hughes, E. Snesrud, N. Lee, and J. Q. (2000, September). A concise guide to cDNA microarray analysis. *Biotechniques 29*(3), 548–562.

Hua, J., Z. Liu, Z. Xiong, Q. Wu, and K. Castleman (2003, September). Microarray BASICA: background adjustment, segmentation, image compression and analysis of microarray images. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2003*, Volume 1, Barcelona, Spain, pp. 585–588.

Hua, J., Z. Xiong, Q. Wu, and K. Castleman (2002, October). Fast segmentation and lossy-to-lossless compression of DNA microarray images. In *Proc. of the Workshop on Genomic Signal Processing and Statistics, GENSIPS*, Raleigh, NC.

ISO/IEC (1993, March). *Information technology - Coded representation of picture and audio information - progressive bi-level image compression*. International Standard ISO/IEC 11544 and ITU-T Recommendation T.82.

ISO/IEC (1999). *Information technology - Lossless and near-lossless compression of continuous-tone still images*. ISO/IEC 14495–1 and ITU Recommendation T.87.

ISO/IEC (2000a). *Information technology - JPEG 2000 image coding system*. ISO/IEC International Standard 15444–1, ITU-T Recommendation T.800.

ISO/IEC (2000b). *JBIG2 bi-level image compression standard*. International Standard ISO/IEC 14492 and ITU-T Recommendation T.88.

Jörnsten, R., W. Wang, B. Yu, and K. Ramchandran (2003). Microarray image compression: SLOCO and the effect of information loss. *Signal Processing 83*, 859–869.

Jörnsten, R. and B. Yu (2000, March). Comprestimation: microarray images in abundance. In *Proc. of the Conf. on Information Sciences*, Princeton, NJ.

Jörnsten, R. and B. Yu (2002, July). Compression of cDNA microarray images. In *Proc. of the IEEE Int. Symposium on Biomedical Imaging, ISBI-2002*, Washington, DC, pp. 38–41.

Jörnsten, R., B. Yu, W. Wang, and K. Ramchandran (2002a, September). Compression of cDNA and inkjet microarray images. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2002*, Volume 3, Rochester, NY, pp. 961–964.

Jörnsten, R., B. Yu, W. Wang, and K. Ramchandran (2002b, October). Microarray image compression and the effect of compression loss. In *Proc. of the Workshop on Genomic Signal Processing and Statistics, GENSIPS*, Raleigh, NC.

Kothapalli, R., S. J. Yoder, S. Mane, and T. P. L. Jr (2002). Microarray results: how accurate are they? *BMC Bioinformatics 3*.

Leung, Y. F. and D. Cavalieri (2003, November). Fundamentals of cDNA microarray data analysis. *Trends on Genetics 19*(11), 649–659.

Lonardi, S. and Y. Luo (2004, August). Gridding and compression of microarray images. In *Proc. of the IEEE Computational Systems Bioinformatics Conference, CSB-2004*, Stanford, CA.

Moore, S. K. (2001, March). Making chips to probe genes. *IEEE Spectrum 38*(3), 54–60.

Netravali, A. N. and B. G. Haskell (1995). *Digital pictures: representation, compression and standards* (2nd ed.). New York: Plenum.

Neves, A. J. R. and A. J. Pinho (2006, October). Lossless compression of microarray images. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2006*, Atlanta, GA, pp. 2505–2508.

Neves, A. J. R. and A. J. Pinho (2009, February). Lossless compression of microarray images using image-dependent finite-context models. *IEEE Trans. on Medical Imaging 28*(2), 194–201.

Pinho, A. J. and A. J. R. Neves (2006, October). Lossy-to-lossless compression of images based on binary tree decomposition. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2006*, Atlanta, GA, pp. 2257–2260.

Rissanen, J. (1983, September). A universal data compression system. *IEEE Trans. on Information Theory 29*(5), 656–664.

Rissanen, J. and G. G. Langdon, Jr. (1981, January). Universal modeling and coding. *IEEE Trans. on Information Theory 27*(1), 12–23.

Said, A. and W. A. Pearlman (1996, June). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology 6*(3), 243–250.

Salomon, D. (2000). *Data compression - The complete reference* (2nd ed.). Springer.

Sasik, R., C. H. Woelk, and J. Corbeil (2004, August). Microarray truths and consequences. *Journal of Molecular Endocrinology 33*(1), 1–9.

Sayood, K. (2000). *Introduction to data compression* (2nd ed.). Morgan Kaufmann.

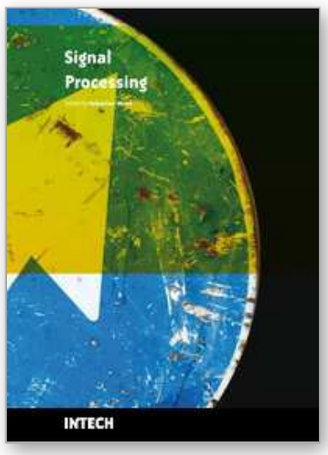
Skodras, A., C. Christopoulos, and T. Ebrahimi (2001, September). The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine 18*(5), 36–58.

Taubman, D. S. and M. W. Marcellin (2002). *JPEG 2000: image compression fundamentals, standards and practice*. Kluwer Academic Publishers.

Weinberger, M. J., G. Seroussi, and G. Sapiro (2000, August). The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *IEEE Trans. on Image Processing 9*(8), 1309–1324.

Yoo, Y., Y. G. Kwon, and A. Ortega (1998, November). Embedded image-domain adaptive compression of simple images. In *Proc. of the 32nd Asilomar Conf. on Signals, Systems, and Computers*, Volume 2, Pacific Grove, CA, pp. 1256–1260.

Zhang, Y., R. Parthe, and D. Adjeroh (2005, August). Lossless compression of DNA microarray images. In *Proc. of the IEEE Computational Systems Bioinformatics Conference, CSB-2005*, Stanford, CA.



Signal Processing

Edited by Sebastian Miron

ISBN 978-953-7619-91-6
Hard cover, 528 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

This book intends to provide highlights of the current research in signal processing area and to offer a snapshot of the recent advances in this field. This work is mainly destined to researchers in the signal processing related areas but it is also accessible to anyone with a scientific background desiring to have an up-to-date overview of this domain. The twenty-five chapters present methodological advances and recent applications of signal processing algorithms in various domains as telecommunications, array processing, biology, cryptography, image and speech processing. The methodologies illustrated in this book, such as sparse signal recovery, are hot topics in the signal processing community at this moment. The editor would like to thank all the authors for their excellent contributions in different areas of signal processing and hopes that this book will be of valuable help to the readers.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen