

Gridding and Compression of Microarray Images

Stefano Lonardi and Yu Luo

Department of Computer Science and Engineering
University of California, Riverside
{stelo|yuluo}@cs.ucr.edu

Abstract

With the recent explosion of interest in microarray technology, massive amounts of microarray images are currently being produced. The storage and the transmission of this type of data are becoming increasingly challenging. Here we propose lossless and lossy compression algorithms for microarray images originally digitized at 16 bpp (bits per pixels) that achieve an average of 9.5–11.5 bpp (lossless) and 4.6–6.7 bpp (lossy, with a PSNR of 63 dB). The lossy compression is applied only on the background of the image, thereby preserving the regions of interest. The methods are based on a completely automatic gridding procedure of the image.

1. Introduction

Microarray is a technology which allows biologists to potentially monitor the activity of all the genes of an organism. Microarrays are relatively new (papers [15, 4] are among the first published on the subject), but they are already extremely popular. Biologists and physicians have enthusiastically embraced this technology, and they are currently producing an unprecedented quantity of microarray data. As of November 2001, for example, the Stanford Microarray Database contained sets of images for about 18,000 experiments [16]. The database has been growing exponentially fast in the last five years.

The output of a single microarray experiment is a pair of 16 bits per pixels (bpp) digital images whose total size is typically in the tens of MB. The information extracted from images comes almost exclusively from the intensities of a small specific subset of the pixels, called *spots*. There is, however, still a debate in the scientific community on the analytic method that should be used to process the images. It is believed crucial, therefore, to keep the raw image data in some permanent storage medium. Simply discarding the raw data and repeating the experiment is not an option because of the high costs associated with this new technol-

ogy. Given the massive amount of data currently produced, and the need of long-term storage and efficient transmission, an *ad-hoc* compression method is becoming more and more needed.

A microarray experiment is based on an hybridization process with a two dye-tagged probes (e.g., the red-fluorescent dye Cy5 and the green-fluorescent dye Cy3) or other labeling methods [15, 4]. The microarray chip is then scanned to generate two digital images, each corresponding to one of the colors [19, 12]. As said, they are typically captured at 16 bpp containing as many as 20 millions pixels [7]. Microarray images are highly structured and appear like a field of spots arranged on a regular grid. The whole image is composed by a matrix of equally spaced blocks called *subgrids*, each of which consists of the same number of rows and columns. A microarray may have several subgrids. For example, Figure 1 shows a microarray with 16 subgrids for a total of about 9,000 spots. The spots in a subgrid are arranged in a relatively uniform spacing with each other. They have a roughly circular shape, though some show significant deviations from this shape due to the experimental variation of the spotting procedure.

The objective here is to design a fully automatic microarray image compression method that does not require any input and/or human intervention. Transform-based image compression methods do not perform well on microarray images because of the high content of high-frequency components (due to the presence of thousands of spots on a black background) [9]. In order to achieve better compression than general purpose compression methods, we need to exploit the geometric structure of the microarray image. The spots are the regions of interest (or the *foreground*) of the image. It is easy to realize that the pixels in the foreground have similar characteristics, which are quite different from that of the pixels in the background. The general strategy here is to separate pixels in set of classes, such that within each class the intensities of the pixels are similar. Currently, we compress the Cy5 and Cy3 image channels separately. We can expect that exploiting the strong correla-

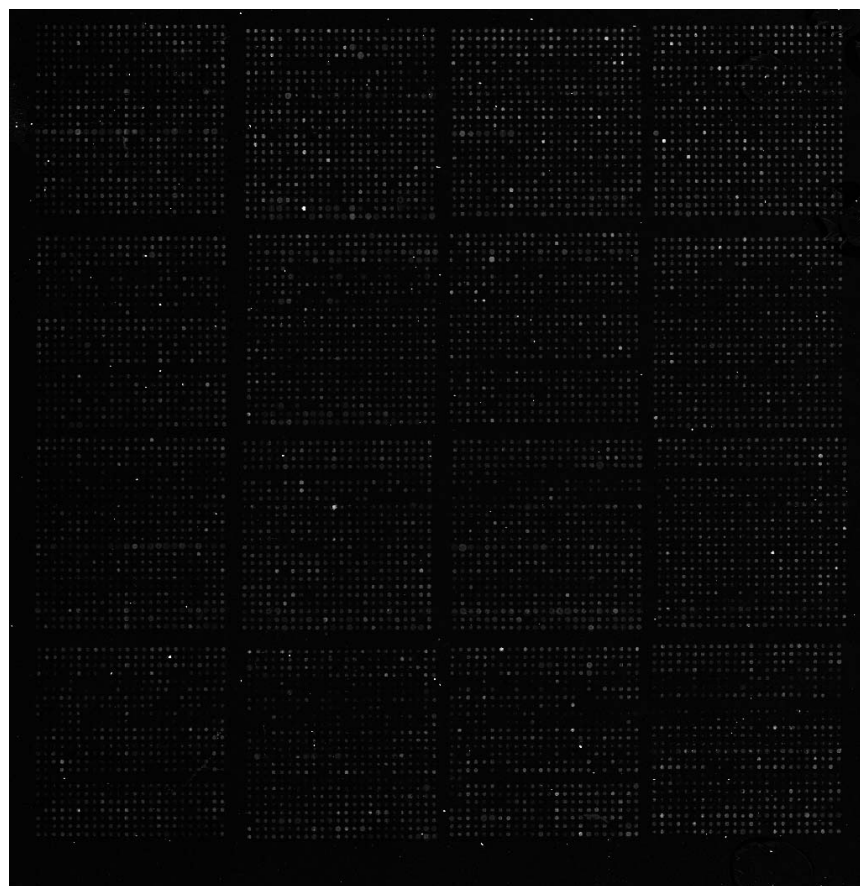


Figure 1. A typical microarray image, with 4×4 subgrids. Each subgrid is composed by a matrix 24×24 of spots. The resolution of this image is 1872×1916 pixels, 16 bpp

tion between the images, the compression will improve considerably.

The compression tool, called MICROZIP, works as follows (see Figure 2 for the flowchart). First, we determine the geometry of the microarray image, by solving the *gridding* or *spot finding* problem [1]. We find the number of subgrids and their location, as well as the number of spots in each subgrid and their location. The contribution of our algorithm is that we recover the grid geometry *without* any input from the user. All other microarray image analysis tools that we tested require the user to describe the parameters of the grid (i.e., number of rows and columns and the position of one of the corners of the grid). Once we have obtained the geometry, we separate the foreground (spots) from the background and then we compress the foreground and background separately (see e.g., [17]).

Because the information extracted from the image is obtained almost exclusively by processing the intensity of the spots, we propose an hybrid lossy/lossless approach. We compress the pixels of the spots without loss of data, and the pixels in the background within some controlled loss of

data. This has the effect of improving dramatically the compression, without affecting the region of interest of the image. In Section 3 we also report the results of purely lossless compression of microarray images.

2. Gridding and Spots Finding

Images from microarray experiments are highly structured since they are composed by high intensity spots located on a regular grid. The shape of the spots is roughly circular, although variations are possible. The ideal microarray image has the following properties [12]: (1) all the subgrids are of the same size; (2) the spacing between subgrids is regular; (3) the location of the spots is centered on the intersections of the lines of the subgrid; (4) the size and shape of the spots are perfectly circular and it is the same for all the spots; (5) the location of the grids is fixed in images for a given type of slides; (6) no dust or contamination is on the slide; (7) there is minimal and uniform background intensity across the image. It goes without saying that almost all real microarray images violate at least one of these these

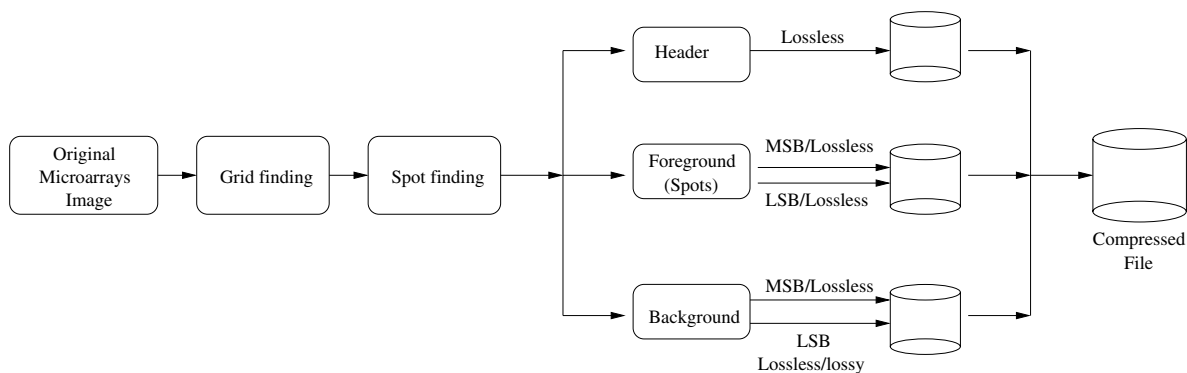


Figure 2. A flow chart of MICROZIP

conditions. In fact, we frequently observed variations on the spot position, irregularities on the spot shape and size, contamination, and global problems that affect multiple spots. In general, the shape and the size of the spots may fluctuate significantly across the array.

Because of the variations on microarray images it is sometimes non-trivial to decide where is the border of a subgrid or the border of a spot. Several methods have been proposed (see, e.g., [3, 7, 19, 18]) and software tools have been developed. However, all the software systems we tested require human intervention. At the minimum, they require the user to specify the geometry of the array, such as the number of grids, number of rows and columns, etc. (see, for example, SPOT from UCSF [7], IMAGENE from BioDiscover [12] and DAPPLE from University of Washington [1]). Given that this is the first step for a data compression tool, we cannot rely on human intervention because it would be not only time-consuming but also quite impractical. To make the comparison more fair, we should remark that our gridding procedure does not need to be as accurate as the software tools mentioned above. In fact, if we happen to slightly misalign a grid, we may lose some compression performance, but the image will be perfectly reconstructed anyway.

As said, a distinguished feature of our algorithm is that of requiring only the image as an input. No other parameter is necessary. Currently, the method assumes the axis of grids to be parallel to the borders of the image. If a rotation of the grids was occurred during the digitization process, we may not be able to retrieve correctly the position of the grid. We are currently working on extending our technique to this case.

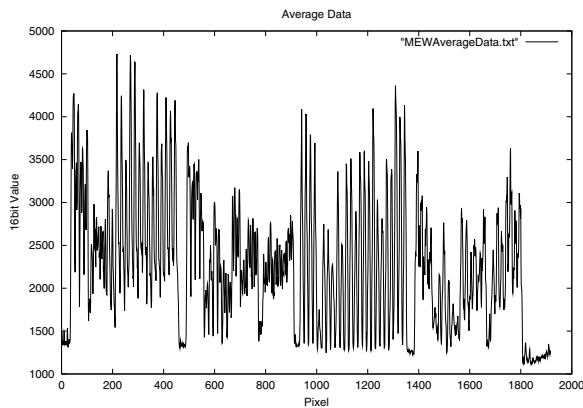
The algorithm is actually used twice. We first apply it to the whole image in order to find the positions of the subgrid. Then we use it again on each subgrid to find the position of the spots. To make the presentation consistent, we use the word *cell* to denote either the rectangle enclosing a spot or the rectangle enclosing a subgrid, and the word *line*

to denote the boundary of a collection of cells.

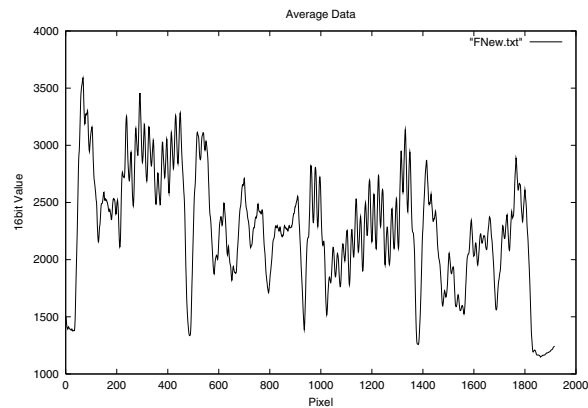
The first step is to compute the average intensities row-by-row and column-by-column on the whole image. We then apply a low-pass filter to remove the noise (see Figure 3). The smoothing window is 25 pixels when searching for the subgrids, 4 pixels when searching for the spots.

Because of the structure of the microarray image, we can assume that the distance between adjacent cells should be approximately equal. Initially we “guess” the positions P of the cells by finding the minima of the graph for the average intensity. We also compute the average value B of these minima, which will become the reference background. Note that these initial positions P may not form a regular grid of cells. Before we start refining the positions P , we temporarily remove the cell positions in first column, the last column, the first row and the last row of cells. The reason is that these rows and columns are very unreliable, which is due to the way microarray slides are spotted. In order to distinguish minima due to the grid boundaries from minima due to spot boundaries, we check the amount of variation of pixel intensities around the position of the minimum. If the intensities of the pixels in a window of 25 pixels centered around the position of the minimum shows little variation, we declare this minimum to be the position of a grid line. The refinement process which is then applied to P to correct possible mistakes is described next. Only after we get a regular grid, the first and last row/columns are added back to P .

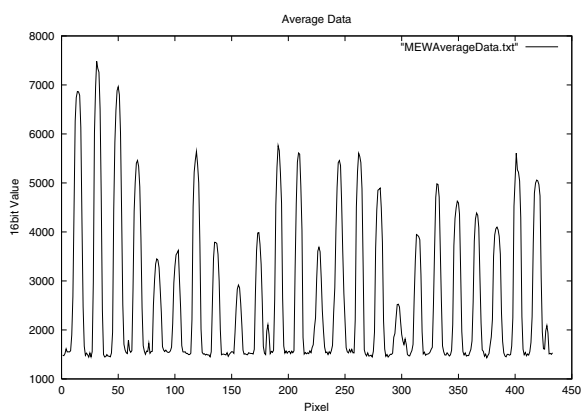
The method needs to be flexible enough to be able to adapt to the variety of situations that we may encounter when processing microarray images. Typically, the positions guessed initially may contain extra/missing lines. We can have an extra line because of noise or contamination in the microarray images that result in high intensities along a row or a column, and we can have a missing line when all the spots in a row/column happen to have low intensities. The iterative refinement tries to adjust the initial guess P to these situations. The first objective is to obtain the most



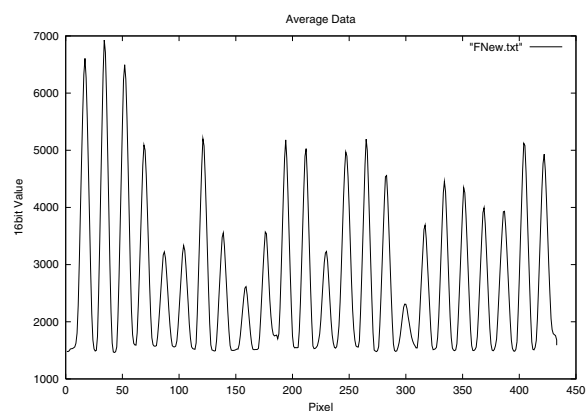
Average data of whole image



Smoothed average data of whole image



Raw average data of one subgrid



Smoothed average data of one subgrid

Figure 3. The effect of the smoothing step on average intensities

precise estimate of the correct distance between adjacent lines. In order to do this we run the iterative algorithm REFINES_GRID described in Figure 4.

The distance R computed by REFINES_GRID is finally used to insert missing lines or delete extra lines in the set of positions P . The value R is also used to adjust the position of the first row/column, and the last row/column. An example of the refinement process is shown in Figure 5. Figure 6 illustrates an example of the iterative procedure.

The method have been tested on several microarray images, and we have found that it is quite effective in handling noises and contaminations in the microarray images. The subgrid and spots positions are correct in the vast majority of the cases, even for poor quality images (see Figure 8 for an example).

Once the subgrid position has been computed we classify the pixels of the spot in each cell. This is done by comparing the intensity of a pixel against the the background reference B . We assign a pixel to the background when its intensity is below B , otherwise it is classified as foreground. Among all the pixels that we classify as a foreground, we find the cen-

ter (\bar{x}, \bar{y}) by computing the average of the x -coordinate and the y -coordinate of the pixels in the foreground.

We then search for the smallest enclosing circle of each spot. We fit each spot with a circle with growing diameter, checking at each choice whether the average intensity of the pixels on the circle goes below the value γB (see Figure 7-(a)). As soon as this happen, we fix the value of the diameter d . We finally extract all the pixels which belongs to disk centered in (\bar{x}, \bar{y}) with diameter d . The information (\bar{x}, \bar{y}, d) of each spot is saved in the header channel.

The identification of the borders of the spot can also be achieved using adaptive shape segmentation methods, such as seeded region growing. Although the shape that these methods can produce is much more accurate, one needs more bits to save it. We chose a circle to describe a spot because it can be described by only three parameters. We are planning to experiment with ellipses. A result of our gridding procedure is shown in Figure 7-(b) and Figure 7-(c). A complete example is shown in Figure 8.

```

real REFINE_GRID( $P$ )
1   $R_{new} \leftarrow$  average distance between adjacent cells on  $P$ 
2  repeat
3     $R \leftarrow R_{new}$ 
4     $P' \leftarrow P$ 
5    repeat
6       $d \leftarrow$  distance between a pair of adjacent lines
7      if  $d < \alpha R$  then remove a line from  $P'$ 
8      if  $d > \beta R$  then add a line to  $P'$ 
9    until all lines have been considered
10    $R_{new} \leftarrow$  average distance between adjacent cells on  $P'$ 
11  until  $|R_{new} - R| \leq \gamma R$ 
12  return  $R$ 

```

Figure 4. The iterative refinement algorithm (we use $\alpha = 0.9$, $\beta = 1.2$, and $\gamma = 0.01$)

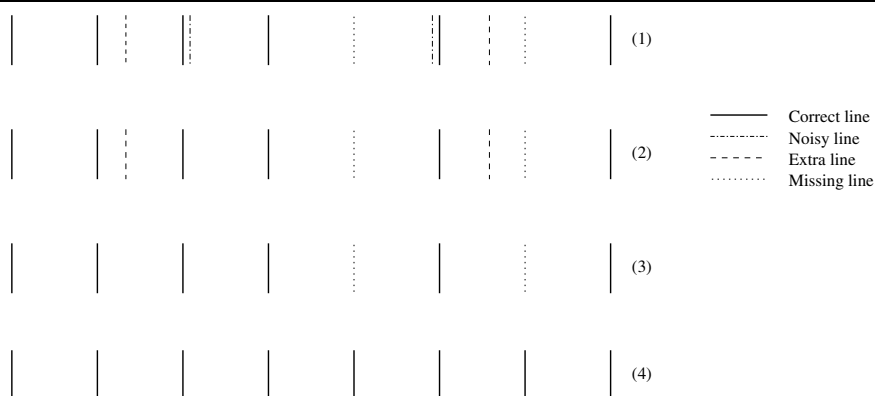


Figure 5. An example of refinement process. (1) the input of the problem; (2) we merge the lines which are very close to each other together; (3) we remove the incorrect lines; (4) we add back the missing lines

3. Compression and Experimental Results

Once the grid has been found, we assign the pixels to different channels depending on their class. If they belong to a spot, they are assigned to the foreground channel. If they do not, they go to the background channel. For our convenience, the background and foreground channels are actually divided in two subchannels: the most significant eight bits (MSB) and the least significant eight bits (LSB). A fifth channel is used to store the information about the geometry of the grid and its position as well as the position of the center and the diameter of each spot. The reference background intensities B for each subgrid is also saved in this latter channel.

We first employed an entropy-minimizing lossless coder to encode the five channels. Since adjacent pixels should be correlated we also considered encoding the differences, i.e., using *delta encoding*. Surprisingly, our experiments shows

that delta encoding does not improve compression. Good results were obtained using arithmetic coding directly on the stream of pixels, although better results were obtained using Burrows-Wheeler transform (BWT) [2]. The five compressed channels are finally combined together in the final compressed file. This format is also convenient for any downstream program that needs to extract information from the image, since a first step in the “parsing” of the image is already done.

The compression performance depends directly on the accuracy of the algorithm for finding the spots. The higher is the precision in classifying pixels as foreground or background, the better is the compression. We tested our software, called MICROZIP, on the image in Figure 1 which is a 1872×1916 pixels 16 bpp microarray image (called Array1) and two larger images (Array2 and Array3). The three images are available from <http://www.cs.ucr.edu/~yuluo/MicroZip/> to allow future com-

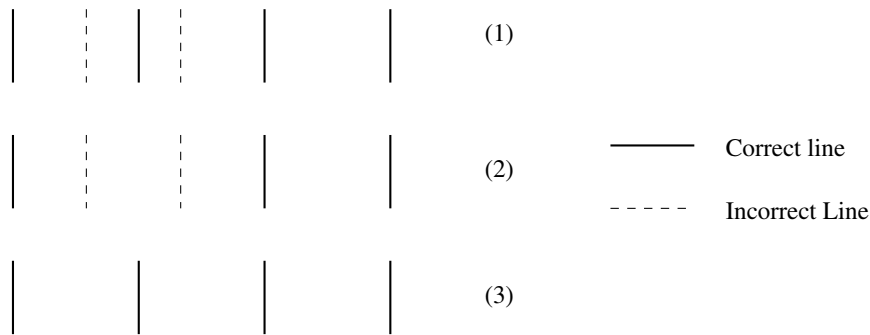


Figure 6. An example of the iterative method. (1) The input of the problem. First, we compute the average distance R between adjacent lines; (2) according to value of R (smaller than the "true" value), the algorithm may determine that the second correct line from the left is incorrect line and remove it; the new average distance R' will be bigger than R ; (3) according to the new value of R' , we look again at the original input and now the algorithm removes the two incorrect lines and keeps the second correct line. The iterative procedure stops when the difference between R and R' are less than δR .

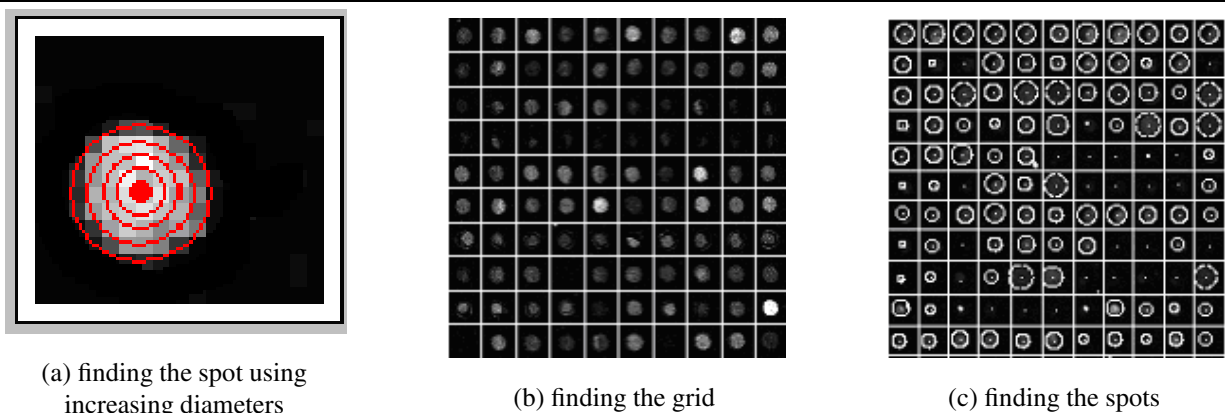


Figure 7. Finding the grid and the spots

parisons.

Table 1 shows the compression achieved on Array1 by using arithmetic coding and BWT on the five channels. Note how the two channels for the least significant bit are the least compressible. This appears consistent with what the authors of [9] found. They report the least significant bits are "almost random", probably due to the noise introduced in the digitization process. They argue that it will be very unlikely to compress microarray images (without loss of data) more than 50%, i.e., less than 8 bpp.

After we combine the five compressed channels with the header containing the geometry of the grids, the final compression for Array1 is 73% of the original size for arithmetic coding and 72% for BWT. Images Array2 and Array3 can be compressed more. In Table 2 we compare this performance with some other general purpose com-

pression tools, like GZIP, lossless wavelets and JPEG-LS. JPEG-LS is the basis for new lossless/near-lossless compression standard for continuous-tone images incorporated in JPEG 2000. The standard is based on the LOCO-I algorithm (LOW COMPLEXITY LOSSLESS COMPRESSION for Images) [13]. JPEG-LS achieves 78% whereas GZIP only achieves 84% (GZIP is the tool currently used by the Stanford Microarray Database administrators [16]).

The running time of MICROZIP is quite reasonable. On an AMD Athlon 1.4GHz PC running Linux, MICROZIP compresses about 256 Kb of data each second, including I/O access. Since our tool does not require any human intervention, it can automatically process a batch of microarray images very easily and quickly.

As one can see from Table 2, there is a quite marginal advantage in using MICROZIP over off-the-shelf lossless

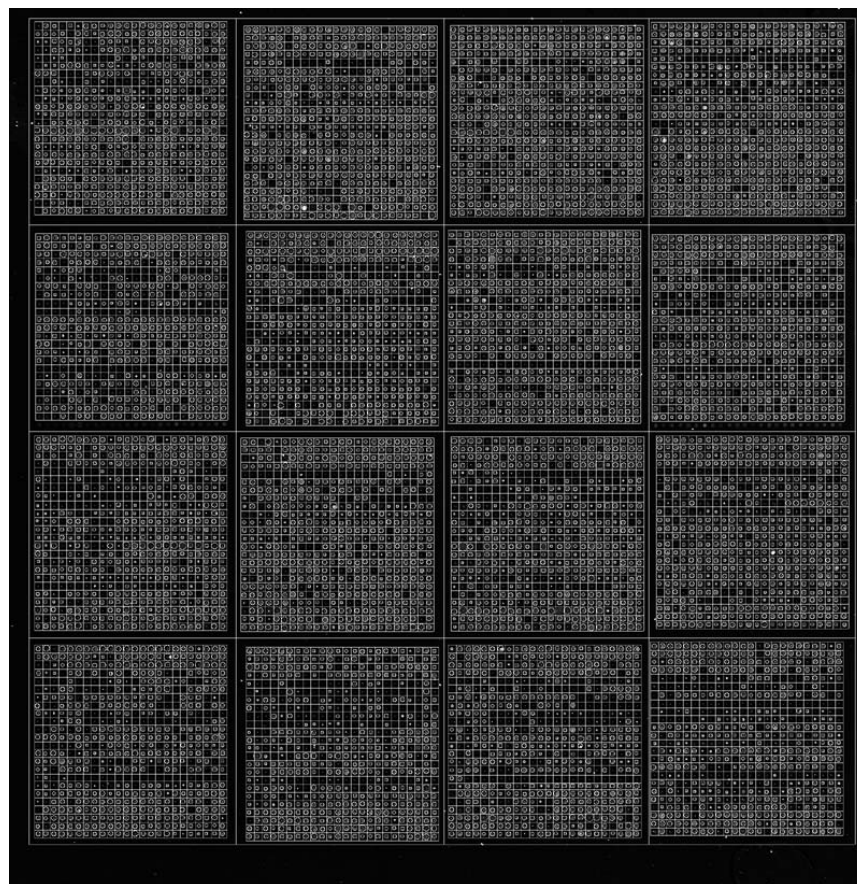


Figure 8. The image of Figure 1 with the grids and spots found by MICROZIP

channel	original size	BWT		Arithmetic coding	
		compressed size	%	compressed size	%
Background (MSB)	2,697,081	1,043,306	38.68%	921,589	34.17%
Background (LSB)	2,697,081	2,708,422	100.42%	2,699,930	100.42%
Foreground (MSB)	889,671	566,566	63.68%	606,705	68.19%
Foreground (LSB)	889,671	893,914	100.48%	890,646	100.11%
Header	47,218	28,193	59.71%	32,604	69.05%

Table 1. Arithmetic coding and BWT encoding of each individual channel for the image in Figure 1

methods. This is due to the difficulty in compressing the LSB channel for the background. Not only it cannot be compressed, but it accounts for more than one half of the final size. Luckily, it is also the least “important” portion of the image. If we lose some data in the least significant bit of the background, the image will be almost unaffected. In addition to this, we are guaranteed that the foreground will be perfectly preserved.

One should, in fact, keep in mind that microarray images are just an intermediate step in the long process of data collection for microarray experiments. These images are pro-

cessed by a combination of image analysis software and human interaction, to obtain a single real value for each spot. Whereas it may be hard to convince biologists to use lossy compression on the whole image, it would be much easier to argue about the advantages of lossy compression at least for the background.

In view of the considerations above, and in order to achieve a higher compression, we implemented a hybrid lossy/lossless compression. We use lossless compression on the pixels of the foreground (which contains the spots) and we use lossy compression on the LSB of background

lossless method	Array1			Array2		Array3	
	size	%	bpp	size	bpp	size	bpp
Original	7,173,504	100%	16	21,500,352	16	13,985,250	16
GZIP	6,001,931	83.67%	13.39	15,402,872	11.46	9,061,450	10.37
Arith. Coding (AC)	5,919,755	82.52%	13.20	15,344,285	11.42	9,092,298	10.40
JPEG-LS*	5,580,041	77.79%	12.45	14,399,970	10.72	7,704,829	8.81
MICROZIP (AC)	5,240,078	73.05%	11.69	13,107,348	9.75	7,269,855	8.32
MICROZIP (BWT)	5,150,642	71.80%	11.49	12,861,078	9.57	7,407,677	8.47

Table 2. Comparing the results of lossless MICROZIP with some general purpose lossless compression methods on microarray image in Figure 1 (Array1) and two other images (Array2 and Array3). * JPEG-LS results have been obtained by compression separately the LSB and the MSB of the image, due to limitations in the software

(BLSB channel for short). We tried several lossy methods for the BLSB channel, and we narrowed our search down to one.

The method, called SPIHT, is a very effective image compression method based on a partitioning of the hierarchical trees of the wavelets decomposition [14]. In order to use it, we had to convert the one-dimensional stream of data into a two-dimensional image. The dimensions x and y of the image can be arbitrarily chosen. Because wavelets-based methods work best when the image is a perfect square, MICROZIP searches for the values of x and y such that $|x - y|$ is the smallest possible. We used an open-source implementation of SPIHT that can be found at <http://www.cipr.rpi.edu/research/SPIHT/> (in progressive mode). Although this particular implementation claims to be able to handle directly 16 bpp images, the decompressor was not able to decompress the images correctly. We were forced to compress only the BLSB channel with SPIHT.

Table 3 and Table 4 show the results of using SPIHT on the BLSB channel, for increasing bit rates. As expected, while the bit rate increases, the root mean square error (RMS) and the peak noise-to-signal ratio (PNSR) decrease and the final size increases. The RMS and PNSR values have to be considered as a measure of quality for the whole image, because the other channels are lossless compressed with BWT. The tables also show the final size and the bpp for the entire compressed image. Unfortunately, SPIHT software currently cannot handle more compression higher than 8 bpp.

At the lowest quality, we obtain a compressed file that is 30%-40% of the original with 63 dB of PNSR. We believe that this quality, associated with the guarantee of perfect reconstruction of the foreground, should be satisfactory for the downstream analysis. If not, one could always choose a quality that matches the quality of the digitization process, as suggested in [9, 10].

4. Conclusions

In this paper, we investigate and experiment lossless and lossy compression system for microarray images. MICROZIP can automatically locate both grids and individual spots and compress the microarray image without any other input parameters and human intervention. Based on our tests, the tool is quite robust to images with variable quality.

We found that the compression performance of the lossless method is only marginally better than state-of-art general purpose lossless compression tools. However, by allowing lossy compression on the LSB of the background, we can obtain a significant compression. The price paid is the introduction a negligible amount of errors in the LSB of the background of the image. The quality of the compressed background can be adjusted to match the average error in the digitization process, thereby making the tool “logically lossy”.

There are several directions for future investigation. Future studies may include the design of more robust and accurate algorithm to locate the position of grids and spots, for example allowing rotations of the grids, along the lines of [3] and [11]. Higher compression could be also achieved by exploiting the strong correlations between the Cy3 and Cy5 images.

Acknowledgments. This project was supported in part by NSF grant DBI-0321756.

References

- [1] J. Buhler, T. Ideker, and D. Haynor. Dapple: Improved techniques for finding spots on DNA microarrays. Technical Report CSE Technical Report UWTR 2000-08-05, University of Washington, 2000.
- [2] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipments Corporation, May 1994.

Array1	Background LSB		Entire image				
	<i>compressed size</i>	<i>%</i>	<i>compressed size</i>	<i>%</i>	<i>bpp</i>	<i>RMS</i>	<i>PSNR</i>
SPIHT (low)	489,860	18.16%	3,021,839	42.13%	6.74	37.23	64.91
SPIHT (avg)	732,913	27.17%	3,264,892	45.51%	7.28	22.46	69.30
SPIHT (high)	979,714	36.32%	3,511,693	48.95%	7.83	13.71	73.59

Table 3. The results of lossless/lossy MICROZIP compression on Array1 (original size: 7,173,504 bytes). Lossy compression is used in the LSB of the background (using three increasing choices of bit rate) and BWT lossless compression for the other channels.

Array2	Background LSB		Entire image				
	<i>compressed size</i>	<i>%</i>	<i>compressed size</i>	<i>%</i>	<i>bpp</i>	<i>RMS</i>	<i>PSNR</i>
SPIHT (low)	1,507,257	18.14%	6,206,568	28.87%	4.62	38.92	64.53
SPIHT (avg)	2,262,440	27.22%	6,961,751	32.38%	5.18	23.98	68.73
SPIHT (high)	3,016,587	36.30%	7,715,898	35.89%	5.74	14.34	73.20

Table 4. The results of lossless/lossy MICROZIP compression on Array2 (original size: 21,500,352 bytes). Lossy compression is used in the LSB of the background (using three increasing choices of bit rate) and BWT lossless compression for the other channels.

- [3] J. M. Carstensen. An active lattice model in a Bayesian framework. *Computer Vision and Image Understanding*, 63:380–387, 1996.
- [4] J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metablobic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.
- [5] N. Faramarzpour and S. Shirani. Lossless and lossy compression of dna microarray images. In J. A. Storer and M. Cohn, editors, *Data Compression Conference*, page 538, Snowbird, Utah, 2004. IEEE Computer Society Press, TCC.
- [6] J. Hua, Z. Liu, Z. Xiong, Q. Wu, and K. R. Castleman. Microarray BASICA: Background adjustment, segmentation, image compression and analysis of microarray images. In *EURASIP Journal on Applied Signal Processing: Special Issue on Genomic Signal Processing*, pages 92–107, 2004.
- [7] A. N. Jain, T. A. Tokuyasu, A. M. Snijders, R. Segraves, D. G. Albertson, and D. Pinkel. Fully automatic quantification of microarray image data. *Genome Research*, 12(2):325–332, 2002.
- [8] R. Jornsten, W. Wang, B. Yu, and K. Ramchandran. Microarray image compression: SLOCO and the effects of information loss. *Signal Processing Journal (Special Issue on Genomic Signal Processing)*, 2002.
- [9] R. Jornsten and B. Yu. “Comprestimation”: Microarray images in abundance. In *Proceedings of Conference on Information Science and Systems*, Princeton, March 14–17, 2000.
- [10] R. Jornsten and B. Yu. Compression of cDNA microarray images. Technical report, Department of Statistics, UC Berkeley, 2001.
- [11] M. Katzer, F. Kummert, and G. Sagerer. A Markov random field model of microarray gridding. In *Proceedings of the ACM Symposium on Applied Computing*, pages 72–77, Melbourne, FL, March 2003.
- [12] A. Kuklin. Laboratory automation in microarray image processing. *American Laboratory*, pages 64–67, May 2000.
- [13] G. S. M. Weinberger, G. Seroussi. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Trans. Image Processing*, 9:1309–1324, 2000.
- [14] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.
- [15] M. Schena, D. Shalom, R. Davis, and P. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.
- [16] G. Sherlock. Personal communication, 2001.
- [17] P. Simard, H. Malvar, J. Rinker, and E. Renshaw. A foreground/background separation algorithm for image compression. In J. A. Storer and M. Cohn, editors, *Data Compression Conference*, pages 498–507, Snowbird, Utah, 2004. IEEE Computer Society Press, TCC.
- [18] Y. H. Yang, M. J. Buckley, and T. P. Speed. Analysis of cDNA microarray images. *Briefings in Bioinformatics*, 2(4):341–349, 2001.
- [19] S. D. Y.H. Yang, M. J. Buckley and T. P. Speed. Comparison of methods for image analysis on cDNA microarray data. *Journal of Computational and Graphical Statistics*, 2002.