MIA: An Effective and Robust Microarray Image Analysis System with Unstructured Information Management Architecture

Wei-Bang Chen and Chengcui Zhang
Dept. of Computer and Information Sciences, University of Alabama at Birmingham, Birmingham, AL 35294 USA
{wbc0522, zhang}@cis.uab.edu

Abstract

The proposed Microarray Image Analysis (MIA) system is designed to analyze microarray slide images in a fully automatic manner. This system identifies and rectifies tilted slides, discovers block boundaries, generates gridlines, recognizes spots, and finally extracts the accurate spot intensity values from the two image channels (red and green) in a microarray slide. The red-to-green intensity ratio of a spot represents the gene expression level in the specimen. Our experimental results demonstrate the effectiveness and robustness of the proposed system. Further, the MIA system is tightly integrated with the component-based Unstructured Information Management Architecture (UIMA) which is an open source platform for the analysis of unstructured data (e.g. images) and is developed by IBM. With UIMA, we can easily apply various analysis algorithms on data by simply plugging analysis components into the system. Further, the analysis results at each analyzing step are attached to the data object as its annotations. The major contribution of this paper is that we design a microarray image analysis system which provides users a convenient manner to automatically analyze slide images and acquire accurate gene expression data from microarray slides. Also, the proposed MIA system, which is based on UIMA, provides a flexible, scalable, and extensible environment for users to perform various analysis tasks on microarray slide images.

1. Introduction

Microarray technology is a chip-based device which performs a biological assay in a simultaneous manner. The central dogma of microarray technique is the binding, measurement, and analysis of specimen and its complementary probes [1]. In microarray experiments, probes are affixed on a solid surface as an array. A specimen is hybridized with the probes. The specimen and the probe form a specific binding if they are complementary. Later, the unbound specimen is washed off, and the measurement of the bound specimen is done by detecting the intensity value of the fluorescent dye on the specimen with a microarray scanner. Afterward, the scanned images are processed, and the obtained data are analyzed by computer software. Microarray technology enables biologists to perform a biological assay on huge amount of interested targets for profiling of complex diseases, analysis of drug effects, and many other applications within one experiment, showing its great potential in biological research and clinical diagnostics in the future.

Microarray technology is very powerful. However, there exist sources of variation that would have impacts on its data precision. The sources of variation come from (1) the biological variation which is intrinsic to all organisms, (2) the technical variation which is introduced during the operation of the experiment, and (3) measurement error which is associated with obtaining the fluorescent signals [2]. In our previous study [3], we reported an automated gridding and segmentation method for correcting the measurement error existent in cDNA microarray images.

However, the algorithm proposed in [3] cannot deal with tilted slide images since its assumption is based on the collimated property of ideal microarray images. However, microarray images often tilt (slightly) in the printing and scanning processes. Thus, there is a need to identify and rectify tilted slides before gridding. In this paper, we provide a solution for determining the tilt angle of a slide by using the Principal Components Analysis.

In addition, the layout of a typical microarray image consists of a group of blocks which are ideally of equal size and perfectly aligned in both vertical and horizontal directions. Each block contains an array of spots, and ideally the distance between two adjacent blocks is uniform. However, this is often not the case due to the technical variation introduced during the experiment. In this paper, we propose a new improved two-step automatic gridding method, based on our previous work [3]. Gridding is a process that registers a set of parallel and perpendicular lines with the image content representing a 2D array of spots. The proposed gridding method is fully automatic and consists of two major steps, including block boundary detection and automatic gridding (within each block). Several methods, such as Markov random field (MRF) [5], template matching and seeded region growing method [6], and the axis projections of image intensities [12], have been reported for automatic gridding. These approaches are not fully automatic because they generally require the user provide information such as numbers of rows and columns as mandatory input parameters. In addition, these methods process a microarray image as a whole and thus cannot deal with the errors/imperfections in block alignment.

Another issue discussed in this paper is related to the management of microarray data and its analysis modules.

1-4244-1500-4/07/$25.00 ©2007 IEEE

Microarray experiments generate results as images which contain abundant information. For example, a microarray slide also comes with a probe set which contains the information regarding to the specification and annotation of a probe (i.e. spot). The slide image and its related information are considered to be unstructured since computer cannot directly understand the contents and extract meanings from them. However, they contain enormous amount of direct and indirect evidence for discovering the useful knowledge. Therefore, there is a need to manage various types of unstructured information efficiently. In this paper, we introduce the use of *Unstructured Information Management Architecture* (UIMA) for microarray data management with a flexible, scalable, and extensible framework [4].

UIMA is developed at IBM for unstructured information management. This emerging technology allows computer to extract useful information from the unstructured content and store these content as structured information. UIMA requires domain experts to build *Analysis Engines* (AEs) for information & knowledge discovery from unstructured contents. The analysis results are stored as annotations in a *Common Analysis Structure* (CAS) which is an object-based data structure for representing and sharing the structured information in the framework.

In addition, since UIMA is a component-based framework, it provides an easy and convenient way for user to plug in various analysis components for analyzing data. In this study, we take advantage of this flexible design and adopt it in managing microarray data and its related analysis modules. In particular, various microarray image analysis algorithms are implemented as Analysis Engines in the proposed system, and more than one algorithm can be implemented for the same analysis task (e.g., the spot segmentation task). The user is allowed to select via a convenient user interface a proper set of analysis engines for a specific microarray image analysis task. The analysis engines read and analyze a microarray image and then add the discovered information as annotations in the CAS. Finally, we store the structured CAS along with the raw data in a database for future data analysis and data mining.

In the rest of the paper, the proposed MIA system architecture is introduced in Section 2. Section 3 provides the experimental results. Section 4 concludes this paper.

2. MIA System Architecture Design

Our goal is to design a system for obtaining the accurate spot intensity values from the two image channels (red and green) in a microarray slide. The red-to-green intensity ratio of a spot represents the gene expression level in the specimen. To achieve this goal, we design a Microarray Image Analysis (MIA) system to analyze microarray slide images in a fully automatic manner. It detects and corrects slide tilt, discovers block boundaries, generates gridlines within each block, recognizes spots, and finally extracts the accurate spot intensity values for each spot. This process can be partitioned into four major modules: Slide Information, Slide Blocking, Slide Gridding, and Slide Segmentation.

The proposed MIA system is tightly integrated with the UIMA framework. The high-level architecture abstraction follows a commonly used Model-View-Controller (MVC) design pattern.

In the proposed system, we implemented a GUI which enables users to specify the locations of imported and exported data. Also, it allows users to choose appropriate analysis engines for performing certain analysis tasks.

The core of the proposed MIA system is designed and implemented as an Aggregate AE in UIMA framework. The four major modules contained in the MIA system are implemented as either a Primitive AE or an Aggregate AE. A primitive AE contains an Annotator and a Component Descriptor. An annotator is some code for analyzing unstructured contents, and a component descriptor is a XML file for describing the data structure, and input/output requirements of the annotator. An aggregate AE includes one or more primitive AEs, and it contains only a component descriptor for describing not only the data structure and input/output requirements, but also the flow of the primitive analysis engines applied on data. In our implementation, we developed a Slide Information Module, a Slide Blocking Module, and a Slide Segmentation Module as implemented them as primitive AEs. The Slide Gridding Module was implemented as an aggregate AE which is composed of two primitive AEs (Section 2.3). Each primitive AE has its own component descriptor and annotator. The raw data along with its analysis results are stored in a Common Analysis Structure (CAS) which provides a common representation

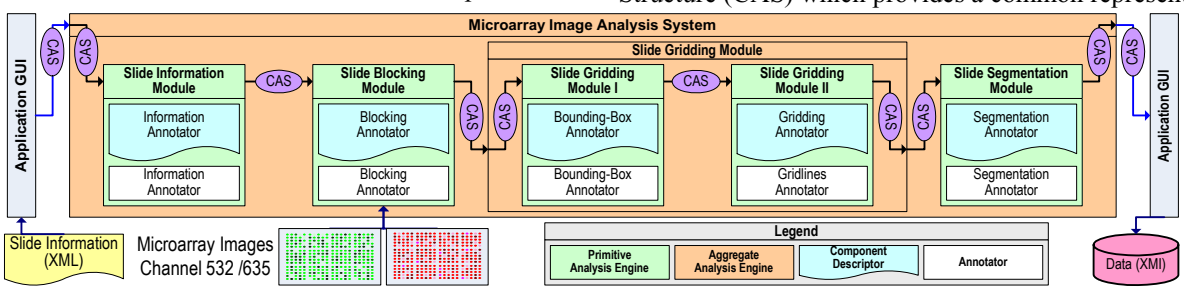

Figure 1. The architecture of the proposed Microarray Image Analysis (MIA) system

and mechanism for sharing data between UIMA components. The proposed architecture is illustrated in Figure 1.

2.1. Slide Information Module

The Slide Information Module is designed to analyze slide information saved in an XML document. It parses the XML document and retrieves slide information, such as slide ID and image filenames, and then stores the retrieved information in the CAS for further analysis. This module will collaborate with an agent-based automatic information retrieval module which can access various public database resources and retrieve the latest information regarding to any gene spotted on a slide. This mechanism allows users to update the gene annotations automatically, which can save a lot of time for users on this routine task. With the up-to-date information in hand, we can obtain more accurate analysis results in the subsequent data analysis steps and discover more interesting knowledge.

2.2. Slide Blocking Module

The design of the Slide Blocking Module is inspired by human recognition behavior. When processing a slide image, humans first identify and rectify tilted slides by eyes, and recognize blocks on a slide by discovering the patterns in block layout, i.e. two adjacent blocks are separated by a larger gap compared to a gap between two spots. Human eyes can easily recognize a gap since a gap is a region between spots with relatively low pixel intensity values. The proposed block recognition module is thus composed of three sub modules: 1. identify the foreground/background pixels, 2. detect tilted slides and correct their orientation, and 3. discover block boundaries.

During the first step, the pixels of the raw image are grouped into two classes – foreground and background pixels, and a binary mask is generated to record the class label for each pixel. We further analyze the binary mask and detect the angle of tilting (if any), which will be used for correcting slide orientation. The corrected slide is then used for locating horizontal and vertical block boundaries, which divide the entire slide into blocks for further analysis. The slide blocking module is thus composed of the signal/noise detector, the tilt detector, and the block boundary detector.

2.2.1. Signal/Noise (S/N) detector. A human operator can identify foreground/background pixels by comparing the pixel intensity value with that of its surrounding area (local) and the entire slide (global). This is especially important when a slide has an uneven background which is a common problem in microarray experiments. A sample slide with uneven background is shown in Figure 2. The S/N detector is designed to distinguish signal (foreground pixels) from noise (background pixels) by adopting a global-local thresholding technique as proposed in our previous work [3]. The goal is to preserve the real signal pixels by identifying the background pixels and removing the pixels that are most likely noise. The method proposed in [3] is robust to uneven background because it takes into consideration both the local and global intensity distributions, and is able to handle severe noise in the slide with a spatial analysis and voting method [3]. Our experiments have proved the effectiveness and robustness of this method. As a final step, a binary mask is generated with 1s representation the identified signal pixels and 0s for background/noise pixels. This mask will be used in the subsequent analysis.

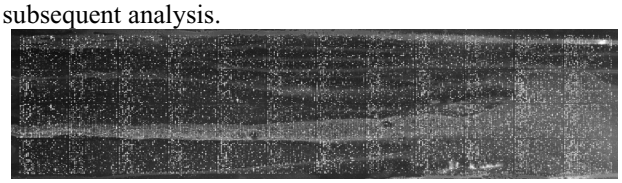

Figure 2. A slide with an uneven background

2.2.2 Tilt detector. In this step, our goal is to identify and correct a tiled slide by first determining the tilted angle, and then rotate the entire slide with affine transformation. Several approaches have been reported to detect the tilted angle. However, these approaches are not time efficient on large data set [7, 8]. In this paper, we proposed a simple yet effective method which adopts Principal Component Analysis (PCA) for detecting the tilted angle of the microarray image.

PCA is a numerical method that was originally devised by Karl Pearson in 1901 though it is more often attributed to Harold Hotelling who proposed it independently in 1933 [9, 10]. The algorithm linearly transforms the data to a new coordinate system and finds a vector, the principal component, on which the projection of the dataset has the greatest dispersion. This technique has been widely used in pattern recognition in high dimensional space and applied to fields such as face recognition and image compression.

After most of the background and noise pixels are identified and removed in the previous step, the slide will have a much cleaner look. We observed that the majority of the foreground pixels in the binary mask are distributed along two main directions which are perpendicular to each other. Ideally, these two directions are perfectly aligned with the horizontal and vertical axis, respectively. However, in a tilted slide, there is a small difference angle θ between the ideal direction and the real direction. In order to find the main direction for a given slide, we use PCA to find the vector on which the foreground pixels are best dispersed. In particular, with PCA, we first obtain two unit vectors v_1 and v_2, representing the two axes with the maximum (first principal) and the minimum dispersion, respectively. We then calculate the inner product of v_1 and v_0, where v_0 is a reference vector which can be either a horizontal vector or a vertical vector, as shown in Figure 3. The value of this inner product is

actually the cosine of the difference angle θ (tilted angle) between v_1 and v_0. Therefore, the actual value of θ can be obtained by computing the inverse cosine of the inner product. We illustrate this idea in Figure 3.

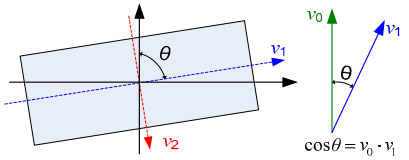

Figure 3. Tilted angle detection by PCA

It is worth noting that the noise cleaning step (see Section 2.2.1) is critical in detecting the tilted angle (if any) because severe noise can affect the spatial distribution of pixel intensities markedly and thus may introduce errors. However, even with the noise cleaning step, the foreground may still contain some aggregated noise pixels which cannot be removed by the previous step. To avoid noise pixels, we adopt a selection strategy, which is based on the observation that the intensity value of noise pixels is often significantly higher or lower than that of real signal pixels. Based on this observation, we select all pixels whose intensity values are between μ–0.6σ and μ+1.5σ as the dataset for computing the principal component, where μ and σ are the average and the standard deviation of the intensities of the selected foreground pixel. This selection strategy covers most of the real signal pixels and proves to be effective in computing the tilted angle within ±6º, as demonstrated in our experimental results (Section 3.1). Once the tilted angle is determined, we can apply the affine transformation on the slide with the rotation angle θ to recover the collimated property of the slide.

2.2.3 Block boundary detector. Given corrected slides, we are ready to locate block boundaries by detecting gaps between blocks. The proposed block boundary detector discovers the repeated block patterns in a slide by detecting gaps between blocks and generates the horizontal and vertical block boundaries. In the following discussion, we use 'gaps' to refer to the gaps between two adjacent spots, and (block) 'boundaries' to indicate the gaps between two adjacent blocks.

To detect block boundaries without knowing the number of blocks in each row/column, we scan the entire slide row by row and column by column at the pixel level. By counting the number of foreground pixels in each row and each column, we obtain the frequency distribution (FD) chart as shown in Figure 4.

We can observe from the FD charts that those rows and columns within the area of gaps or boundaries have relatively low counts of foreground pixels. The only difference between gaps and boundaries is the width value since the distance between two blocks is larger than that of two spots. Therefore, our proposed approach first screens for a set of candidate block boundary lines which includes all boundaries and some gaps, and then, identifies the true boundaries by checking their width. This is achieved by first selecting those rows/columns whose signal counts are lower than a threshold T. We determine the threshold T by calculating the mean and the standard deviation of each chart as shown in Formula 1.

$$T = \mu - k \times \sigma \qquad (1)$$

where μ and σ are the mean and the standard deviation of the number of foreground pixels, respectively, and k is a coefficient used to control the percent of rows/columns selected as block boundary candidates. In our case, we want to make sure that no less than 5% of the row/column lines are selected as candidates. Thus, the actual value of k is the maximum k that can satisfy this requirement. Although a smaller k will always satisfy this requirement, it could also cause the value of T too high to be appropriate.

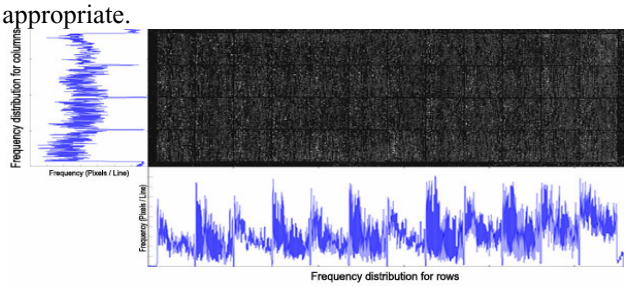

Figure 4. Frequency distribution charts

With this approach, the set of identified candidate lines contains a series of groups of adjacent rows or columns corresponding to the gaps/boundaries on the slide. With the assumption that a typical boundary should be wider than 5 pixels, we collect the width value for each group of adjacent rows/columns and exclude all those groups with their width values less than 5 pixels. Thus, the remaining groups of adjacent rows/columns are what we believe the areas corresponding to block boundaries. Next, we find the central line within each group as the boundary to separate adjacent blocks, as shown in Figure 5.

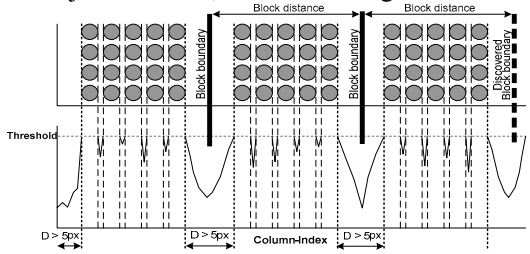

Figure 5. Block boundary detection

In some cases, due to the severe noise contamination, it happens that some block boundaries cannot be recognized with the above algorithm, not even by human eyes. To deal with this issue, we use the already identified block boundaries to estimate the location of missing block boundaries. In particular, we used identified boundaries to obtain the dominant distance between two adjacent block boundaries and use that as the estimated block size. This information is then used to check if there is any false

negative (missing block boundary) or false positive (false block boundary), and to interpolate missing boundaries or to remove false boundaries from the result. We illustrate how to use the estimated block size to discover missing block boundaries in Figure 5.

2.3. Slide Gridding Module

Gridding is the process of identifying the dividing line between each two consecutive rows/columns of spots, such that the position of each spot within the grid can be determined. This Slide Gridding Module generates a grid within each block for separating spots, i.e. a cell in the grid contains only one spot. In our first attempt, we used the idea of FD chart from the previous section but found it did not work well. The reason for the poor performance is that the typical width of a gap (between spots) is only a couple of pixels, and the noise existent in that regions often interfere with the foreground pixel counts. Hence, we adopt a different strategy in which the first step is to generate a minimum bounding box for each group of connected foreground pixels. Each of such connected group (ideally) corresponds to a single spot in the slide, with a few exceptions which constitute only a small fraction of all the groups (see Figure 6). Therefore, the average size (length and width) of a spot can be estimated. Then, all the bounding boxes whose size is close to the average spot size are selected, and the centroids of their bounding boxes are used to locate the central lines passing through the center of each row/column of spots, which are shown as dotted horizontal/vertical lines in Figure 6. Afterwards, we can use the middle point of two adjacent central lines to generate grids.

Figure 6. Detect central lines

The Slide Gridding Module was implemented as an aggregate analysis engine which is composed of two primitive analysis engines (one for bounding box generation and another for grid line detection). Each primitive analysis engine has its own component descriptor and annotator. The output of this module is a set of grids for the blocks on the slide, which will be stored as a type of microarry image annotation data in a Common Analysis Structure (CAS).

2.4. Slide Segmentation Module

Although the spot segmentation is not the major contribution of this study, it is briefly presented here for the completeness of the MIA system. When the Slide Gridding Module generates grid for each block, it also generates the locations of spot centers and spot bounding boxes. We are now ready to perform spot segmentation on each cell in the grid and extract the real signal pixels from each spot. In our previous study, we proposed a simple yet effective segmentation method that utilizes Otsu's threshold algorithm [11] in a progressive manner. With Otsu's algorithm, for each cell area (a spot region), a local threshold is automatically chosen to minimize the intra-class variance and the inter-class variance of the thresholded foreground and background pixels. The threshold *th* is positioned midway between the means of the two classes ('foreground' and 'background') [3]. The Otsu's algorithm is applied to the remaining foreground pixels in an iterative manner with which noise due to scratches and donut effects can be gradually removed from the foreground. In this study, this spot segmentation algorithm [3] is implemented as a primitive analysis engine in the MIA system.

3. Experiments

3.1. Tilted angle detection results

To evaluate the performance of our proposed method in detecting the tilted angle, we test our approach on 5 real microarray images and 32 manually rotated artificial microarray slides, including 20 clockwise and 12 counterclockwise rotated images with tilted angles from 0.0 to 6.0 degrees.

The actual and the detected tilted angles of the 32 artificial microarray images are shown in Table 1. We use 'CW' and 'CCW' to represent clockwise angles and counterclockwise angles, respectively. The accuracy of the proposed method on these slides is 100%.

Table 1 Tilted angle detection on artificial slides

Tilted Angle	Detected Angle	Tilted Angle	Detected Angle
CW 0.0	0.0	-	-
CW 0.1	0.1	CCW 0.1	-0.1
CW 0.2	0.2	-	-
CW 0.3	0.3	-	-
CW 0.4	0.4	-	-
CW 0.5	0.5	CCW 0.5	-0.5
CW 0.6	0.6	-	-
CW 0.7	0.7	-	-
CW 0.8	0.8	-	-
CW 0.9	0.9	-	-
CW 1.0	1.0	CCW 1.0	-1.0
CW 1.5	1.5	CCW 1.5	-1.5
CW 2.0	2.0	CCW 2.0	-2.0
CW 2.5	2.5	CCW 2.5	-2.5
CW 3.0	3.0	CCW 3.0	-3.0
CW 3.5	3.5	CCW 3.5	-3.5
CW 4.0	4.0	CCW 4.0	-4.0
CW 4.5	4.5	CCW 4.5	-4.5
CW 5.0	5.0	CCW 5.0	-5.0
CW 6.0	6.0	CCW 6.0	-6.0

The performance of the proposed approach on real microarray slides is hard to evaluate at the beginning because the ground truth is missing. That is, the exact tilted angles are very difficult to measure especially when they are small. To get around this problem, rather than comparing the detected angle with the 'actual' angle, we use the block boundary detection result as an indication of the effectiveness of the proposed method when applied to real microarray images. This is based on the fact that before the block boundary detection, microarray images must be corrected according to the detected tilted angle. Each microarray image used in our experiments contains 48 blocks with 5 vertical and 13 horizontal block boundary lines (see Figure 4).

The experiment consists of two phases. In the first phase, we apply our proposed approach on 5 real slides to detect the tilted angles, and then rotate the 5 images based on the detected tilted angles. The phase I results are

shown in Table 2. As can be gleaned from the results, the accuracy of block boundary detection, when performed on the corrected images, is 100% for all the 5 images.

In the second phase, we manually rotate these slides clock-wisely by 5 degrees and measure the tilted angles of the rotated images with the proposed approach. Subsequently, we correct the rotated images based on the tilted angles detected, and then perform the block boundary detection on the corrected images. The results of phase II are shown in Table 3. Again, we obtain 100% accuracy for all the 5 images.

Table 2 Phase I evaluation results

Slide ID	Detected Angle	Detected vertical lines	Detected horizontal lines
7743	0.01	5 / 100%	13 / 100%
7744	0.01	5 / 100%	13 / 100%
7745	0.01	5 / 100%	13 / 100%
7849	0.01	5 / 100%	13 / 100%
1040	0.01	5 / 100%	13 / 100%
Overall Accuracy		100%	100%

Table 3 Phase II evaluation results

Slide ID	Detected Angle	Detected vertical lines	Detected horizontal lines
7743	5.00	5 / 100%	13 / 100%
7744	5.00	5 / 100%	13 / 100%
7745	5.00	5 / 100%	13 / 100%
7849	5.00	5 / 100%	13 / 100%
1040	5.00	5 / 100%	13 / 100%
Overall Accuracy		100%	100%

It is worth noting that the proposed method works best for slightly tilted microarray images (within ±6 degrees). However, it still fits our needs since it is a very rare case that a slide tilts more than 5 degrees. In spite of that, the proposed method is time efficient when compared to the much more complicated algorithms as proposed in [7, 8].

3.2. Block detection and gridding results

To test the performance of our proposed method in handling uneven background and noise contamination, we select 8 slides in poor condition (a sample slide is shown in Figure 2) and evaluate the performance in terms of recall and precision values.

Based on our experiments on 8 different slides (48 blocks each, 384 blocks in total), the slide blocking module returns 384 correctly identified blocks. Thus, the recall value and precision value of the proposed method are both 100%. The results show the robustness of our proposed algorithm.

In our experiment to test the performance of automatic gridding on 8 slides (60 gridlines for each block, 23040 gridlines in total), our method missed 175 gridlines, and returned 1 false gridline. Therefore, the recall is 99.24% and the precision is 99.99%. It should be noted here that it is not appropriate to compare the proposed gridding algorithm with those proposed in [5, 6, 12] because the latter require the user to provide the grid template information (e.g., the number of rows/columns), which is discovered in a fully automatic manner with the proposed method.

4. Discussions, Conclusions, and Future work

In this paper, we proposed a Microarray Image Analysis system (MIA) which can automatically detect the block boundaries, generate grids, and segment spots for microarray images. As demonstrated by our experiments, the proposed system demonstrates a superb performance in handling microarray slides with (small) tilted angles, uneven background, and excessive noise.

In addition, the proposed model is fully integrated with the UIMA framework. The most impressive experience with UIMA is that it provides great flexibility for us to add primitive analysis engines in any order to form an aggregate analysis engine without any extra coding. Thus, we can implement more than one primitive analysis engine for a specific analysis task (e.g., block detection) and plug in the selected analysis engine to the aggregate engine for easy performance comparison.

In our future work, we will implement an agent-based information retrieval module for retrieving related information from online public databases like MeSH and store it into the microarray image database. Further, we shall focus on the subsequent data analysis and mining of the microarray experiment data since this is the ultimate goal of the microarray experiments.

5. References

[1] U. M. Braga-Neto and E. T. Marques, Jr., "From functional genomics to functional immunomics: new challenges, old problems, big rewards," *PLoS Comput Biol,* 2(7): e81, Jul. 28, 2006.

[2] G. A. Churchill, "Fundamentals of experimental design for cDNA microarrays," *Nat Genet,* v32, Suppl: 490-5, Dec. 2002.

[3] W.-B. Chen, C. Zhang, and W.-L. Liu, "An Automated Gridding and Segmentation Method for cDNA Microarray Image Analysis," *in Proc. of the 19th IEEE International Symposium on Computer-Based Medical Systems*, pp. 893-898, 2006.

[4] D. Ferrucci and A. Lally, "Building an example application with the Unstructured Information Management Architecture," *IBM SYSTEMS JOURNAL,* v43, pp. 455-475, 2004.

[5] M. Katzer, F. Kummert, and G. Sagerer, "Methods for automatic microarray image segmentation," *NanoBioscience*, *IEEE Transactions on*, v2, pp. 202-214, 2003.

[6] 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 Res*, v12, pp. 325-32, 2002.

[7] N. Deng and H. Duan, "An Automatic and Power Spectra-based Rotate Correcting Algorithm for Microarray Image," in *Proc. of the 2005 IEEE Engineering in Medicine and Biology Conference*, pp. 898-901, Shanghai, China, 2005.

[8] Y.-K. Wang and C.-W. Huang, "DNA microarray image analysis using active contour model," in P*roc. of IEEE Computational Systems Bioinformatics Conference (CSB)*, pp. 549-550, 2004.

[9] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology,* v24, pp. 417-441, 1933.

[10] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine,* v2, pp. 559-572, 1901.

[11] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics,* v9, pp. 62-66, 1979.

[12] Y. H. Yang, S. Dudoit, P. Luu, D. M. Lin, V. Peng, J. Ngai, and T. P. Speed, "Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation," *Nucleic Acids Res*, v30, pp. e15, 2002.