An automatic block and spot indexing with $k$-nearest neighbors graph for microarray image analysis

Ho-Youl Jung and Hwan-Gue Cho

Department of Computer Science, Pusan National University, San-30, Jangjeon-dong, Keumjeong-gu, Pusan, 609-735, Korea

Received on April 8, 2002; accepted on June 15, 2002

ABSTRACT

Motivation: In this paper, we propose a fully automatic block and spot indexing algorithm for microarray image analysis. A microarray is a device which enables a parallel experiment of ten to hundreds of thousands of test genes in order to measure gene expression. Due to this huge size of experimental data, automated image analysis is gaining importance in microarray image processing systems. Currently, most of the automated microarray image processing systems require manual block indexing and, in some cases, spot indexing. If the microarray image is large and contains a lot of noise, it is very troublesome work. In this paper, we show it is possible to locate the addresses of blocks and spots by applying the Nearest Neighbors Graph Model. Also, we propose an analytic model for the feasibility of block addressing. Our analytic model is validated by a large body of experimental results.

Results: We demonstrate the features of automatic block detection, automatic spot addressing, and correction of the distortion and skewness of each microarray image.

Contact: hyjung@pearl.cs.pusan.ac.kr

INTRODUCTION

A microarray is a device which enables a parallel experiment of ten to hundreds of thousands of test genes in order to measure gene expression (Duggan et al., 1999; Ekins and Chu, 1999). In order to compare expression levels using computer devices, microarray generates fluorescence images of regular arrays of spots containing target genetic material to which the differentially labeled samples are hybridized (Schena, 2000; Jain et al., 2002). Figure 1 shows a typical image of a microarray which consists of $4 \times 4$ blocks and each block is composed of $24 \times 24$ spots (Alizadeh et al., 2000).

In order to determine the amount of fluorescence, we must first identify the location of each block and spot. This process is called ‘grid fitting’ or ‘spot addressing’. Then we can estimate the intensity of the true spot and the background. This is called ‘spot fitting’ or ‘quantification of spot’.

There have been many software systems and methods to quantify the intensities of spots. Generally, we can classify the microarray image analysis systems as follows (Schena, 2000):

- Assistance software: This system provides Graphic User Interface (GUI) tools for assisting the user to lay down a template and manually adjust the positions and the sizes of the spots.

- Semi-automated processing system: User specifies the bounding area or guide spots of the array, and system automatically locates each spot. GUI tools for manual corrections of any possibly misidentified spots are provided.

- Fully automated processing system: No human needs interactions. The grid and spots are automatically found and quantitated.
Almost all microarray image analysis systems in wide use allow for human interaction in block indexing and spot indexing as shown in Table 1. ScanAlyze (Eisen et al., 1998) and GenePix (Axon Instruments Inc., 2001) require manual work at the level of both block and spot indexing. AutoGene (BioDiscovery, 2001) needs manual input for minimum information (e.g. the number of rows and columns of spots) in spot indexing. All these systems assume that block indexing is already done by human operators. So, most of the analysis systems are hardly called a ‘fully automated processing system’.

Steinfath et al. proposed an automatic analysis method for microarray experiments from image analysis to clustering (Steinfath et al., 2001). Their system is designed to automatically analyze images from hybridization experiments with various arrangements: different kinds of probes, different supports, different labeling of probes. In order to analyze the intensities of spots in a block, they assume that the targets (spots) are arrayed in a grid, which can be approximately transformed to an orthogonal equidistant grid by a projective mapping. However, they did not mention the problem of block identification.

Jain et al. developed a system concerned with an automatic quantification of microarray image data (Jain et al., 2002). The system automatically locates and segments each spot and estimates ratios, eliminating the requirement for user identification of any image coordinates. However, as mentioned by the authors, this system works for automatic analysis only if the spot expression rate is more than 80%. So, if the expression rate is less than 70%, they did not guarantee the performance and manual work is required. Furthermore, in this system, estimation of spot spacing and block spacing is accomplished by using image profiling which is summing the signal intensities in the X and Y direction of the image. If the given microarray image is skewed, then this profiling method can not be applied.

The objective of this paper is to present a method for full automatic identification of each block and spot. In our method, we exploit graph models (k-Nearest Neighbors graph and $\epsilon$-graph) by mapping highly expressed spots as a vertices set and edges by considering the adjacency. Our graph-based method has some advantages; it is robust regarding the slope and distortion of image data. The only required knowledge for our method is the number of blocks and spots - namely the resolution of grid structure.

### GRIDDING PROBLEM

The final output of an image analysis is the expression profile, including the mean or median value of intensities in the region of each spot on the microarray image. In order to compute the intensity of each spot, we must consider the following three problems:

- **Problem 1 (BLOCK INDEXING)**: Find the block index of a specified spot.
- **Problem 2 (SPOT INDEXING)**: Find the index $(i, j)$ of a specified spot in an already indexed single block.
- **Problem 3 (INTENSITY COMPUTATION)**: Compute the signal intensity of a specified spot.

There are many existing techniques for Problem 3, intensity computation. There is an intensity computation method that is applicable only after the grid fitting is done (Brändle et al., 2000). Buhler also proposed a method of quality control on the shape and position of the spot (Buhler et al., 2000). The method exploits the physical information of the microarray and the average position of several microarray images in the grid fitting process. If there are large amounts of the image data produced by heterogeneous microarrays, then it is difficult for this method to support full automation of image processing.

The ideal microarray image should have the following properties: (i) all the subgrids (blocks) are the same size; (ii) the spacing between two blocks is uniform; (iii) the distance between two spots is uniform; (iv) the size and shape of the spots are perfectly circular and the same size; (v) the location of the grids is fixed in images for a given type of slides; (vi) no dust or other contamination is on the slide; (vii) there is a minimal and uniform background intensity across the images.

If all these conditions are satisfied, then image analysis can be done easily. However in real data some of these conditions are violated. These violations may be conceptualized in four categories: (i) spot position variation, (ii) spot shape and size irregularity, (iii) sample contamination, and (iv) global problems that affect multiple spots (Schena, 2000). Since various violations exist, fully automatic analysis of microarray images is very difficult work. So most of the existing analysis systems allow for several.
types of manual input, for example, the number of blocks, the number of spots in each block, the unit distance between two neighboring spots, the gap distance between two adjacent blocks, the size of the spots, etc.

Since the amount of data is not as excessively large in traditional tissue-based experiments, the ratio of analysis time to experimental time is small. However, in microarray experiments this ratio is enormous (Schena, 2000). Therefore, reducing the cost of image analysis procedures is desirable for the future. The major limitation of building fully automated systems is that it is difficult to address whole spots in the microarray image data, because the grid fitting needs a lot of human visual inspections. Therefore, fully automated systems must be developed for the whole process of image analysis including the identification of the blocks and the spots.

**BLOCK INDEXING ALGORITHM**

Most previous microarray analysis systems did not seriously enough consider the importance of block indexing. In this Section, we explain a block indexing algorithm which shows good performance at the low expression rate and for skewed image data.

Let $I$ denote given image data and $\{s_i\}$ be a set of segmented spots in $I$. $\{s_i\}$ is a connected component of pixels whose value of intensity is more than the predefined threshold. We can determine the value of the threshold as a predefined static value or consider the histogram of the whole image $I$. Let $c_m(s_i)$ denote the center of mass of $s_i$. Then we make a graph model, $G = (V, E)$ for $I$. At first, we isolate the components of the whole spot image by image segmentation techniques (8-connected components). Then, all the segments are mapped to vertex set, $V(G) = \{c_m(s_i)\}$.

### Modified k-nearest neighbors (MKNN) graph model

Given $|V|$ points in the plane, the ‘nearest neighbor’ is a relation on a set $V$ of points as follows: point $b$ is a nearest neighbor of point $a$, denoted $(a, b)$, if
\[
|(a, b)| = \min_{x \in V-a} \{|(a, x)|\},
\]
where $|(a, b)|$ denotes the $L_2$ metric length of the distance from $a$ to $b$. Especially, if a pair satisfies symmetry ($(a, b)$ and $(b, a)$), then it is called a reciprocal pair. A $k$-Nearest Neighbors (KNN) graph is a directed graph, $G_k = (V, E)$, where all vertices are adjacent up to $k$th nearest vertex in the plane (Preparata and Shamos, 1985).

If we use the KNN graph model, block identification is very troublesome work because of the reciprocal pairs. Our block identification algorithm is based on the intersection of the MBRs of the components of the graph. If there are many reciprocal pairs, the probability of the intersection between two MBRs is very low. Therefore, in order to increase the probability of the intersection, we propose the Modified KNN (MKNN) graph, $G_k^M = (V, E)$, which can be constructed as follows:

**Algorithm: Construct MKNN from $V(G_k)$**

**Input:** $V(G_k)$ and $k$

**Output:** $G_k^M = (V, E)$

\[
V(G_k^M) = V(G_k);
\]

\[
\text{for } i = 1 \text{ to } |V(G_k^M)|
\]

\[
\text{for } j = 1 \text{ to } |V(G_k^M)|
\]

\[
S_i = k \text{ neighbors of } v_i \in V(G_k^M) \text{ which satisfy following:}
\]

\[
|(v_i, v_j)| = \min_{x \in V-\{v_i, f(S_j, v_i)\}} \{|(v_i, x)|\};
\]

\[
\text{end_for}
\]

\[
\text{end_for}
\]

$f(S_j, v_i)$ represents whether $v_i$ is an element of $S_j$ or not. $f(S_j, v_i)$ is computed as follows:

\[
f(S_j, v_i) = \begin{cases} 
 v_j & \text{if } S_j \cap v_i = v_i \\
 \phi & \text{otherwise}
\end{cases}
\]

![Fig. 2. A visual description of making KNN and MKNN graph](image)

Figure 2 shows a visual description of making KNN and MKNN graph when $k = 1$. $a$, $b$, $c$, and $d$ are edges of KNN and $aa$, $bb$, $cc$, and $dd$ are those of MKNN.

Figure 3 shows a KNN graph representation of a sample microarray image using $k = 1$. In Figure 3a, using the KNN graph, we can see some reciprocal pairs.
H.-J. Jung and H.-G. Cho

(a) KNN graph of a sample image: the number of components = 48

(b) Modified KNN: the number of components = 6

Fig. 3. A comparison of KNN and modified KNN graphs \((k = 1)\).

Figure 3b shows the modified KNN graph whose number of connected components is 6. As is shown, the number of components of MKNN is much smaller than that of KNN graph.

Identification of block index

In order to separate each block, we should know the resolution of the blocks, e.g. how many blocks are in rows \((b_r)\) and columns \((b_c)\). For this, we use the MBR (Minimum Bounding Rectangle) merging technique for each connected component, \(C_i\). Let \(C_i\) and \(C_j\) be two disjointed components of the MKNN graph. If MBR\((C_i)\) overlaps MBR\((C_j)\), then we regard \(C_i\) and \(C_j\) as belonging to the same block. This MBR merging procedure repeats until there are no more updates. The algorithm for block identification is given as follows:

Algorithm: Identification of Block Index

Input: \(\{C_i\}\), connected components from \(G_k = (V, E)\)

Output: \(\{\hat{C}_i\}\), connected components after MBR merging

\[
\{\hat{C}_i\} = \{C_i\}
\]

no_intersection \rightarrow false;
while no_intersection is false {
    no_intersection \rightarrow true;
    for \(m = 1\) to \(|\{\hat{C}_i\}|\)
        for \(n = m + 1\) to \(|\{\hat{C}_i\}|\)
            if there exists intersection\((\hat{C}_m, \hat{C}_n)\)
                then {
                    take union operation\((\hat{C}_m, \hat{C}_n)\);
                    no_intersection \rightarrow false;
                }
        end_for
    end_for
} end_while

This procedure is shown in Figure 4. There are 5 components in MKNN in the beginning. Let \(\square x\) denote an MBR of component \(x\). In the first step, \(\square a\) and \(\square b\) will be merged into one MBR\((\square a\overline{b})\), and \(\square c\) and \(\square d\) will be merged into one MBR\((\square c\overline{d})\). Then Figure 4c shows the results of the first stage. Figure 4d shows the third step. Figure 5 shows a \(2 \times 2\) blocks separation procedure with real chip image.
Fig. 5. An example of $2 \times 2$ blocks separation.

Probability model for block separation

In this Section, we describe the probability model for block separation. Let $e_p$ denote the spot expression probability (number of expressed spots/number of whole spots) and $d_B$ denote the distance between two adjacent blocks. Now, we can compute the probability that two adjacent blocks are merged using MBR merging procedure. Let inter-block edge, $(u, v)$ denote an edge which connects two different blocks in $G_k^M = (V, E)$ as follows:

$$(u, v) \in E \rightarrow u \in B_a, v \in B_b.$$ 

Figure 6 shows two adjacent blocks ($B_a$ and $B_b$). Let $r$ denote the radius of the circle at a boundary point. $R_a(r)$ and $R_b(r)$ represent the intersection regions of $B_a$ and $B_b$ with the inside of the circle, respectively. We assume that there are $n \times n$ spots in each block. Let $P_I$ be the probability that there exists at least one inter-block edge between two adjacent blocks. In Figure 6, there is at least one expressed spot in $R_a$ and there is no expressed spots in $R_b$, except for the one boundary spot. $P_I$ can be computed as follows:

$$P\{E_I \neq \emptyset\} = P_I = 1 - \prod_{i=1,j=1}^{n} p_{ij},$$

where $E_I$ is a set of inter-block edges and $p_{ij}$ is the probability that there is no inter-block edge along the vertex $u_{ij} \in B_b$. If we were to make the complete probability model, we would consider all vertices in each block. However, in this paper, we only consider the boundary spots of each block, since the probability of an inter-block edge between two internal vertices (not in the boundary location) is negligible. So, we only consider $\hat{P}_I$ (probability model of block merging only between boundary spots) rather than $P_I$. Let $S_a$ and $S_b$ denote the area of the region $R_a(r)$ and $R_b(r)$, respectively. $\hat{P}_I$ is computed as follows:

$$\hat{P}_I(e_p, d_B, n) = 1 - \left( \prod_{r=d_B}^{n-1} (1 - e_p^{S_a} \cdot (1 - e_p^{S_b})) \right)^n.$$ 

Table 2 shows the $\hat{P}_I$ of an artificial data set, which was made for experiments. It shows that successful block separation is possible using our algorithm. If $e_p \geq 0.6$ and $d_B$ is 2, it is enough to separate blocks. If it is known that $e_p \geq 0.4$, then it is recommended to make $d_B \geq 3$ for a fully automatic block separation.

SPOT INDEXING ALGORITHM

In this paper, we convert the microarray image data into the graph structure for the automation of spot indexing. In the following, we explain the method of the graph
Let \( \epsilon \) be a guide spot. The spot addressing algorithm is as follows:}

**Algorithm:** Spot Addressing with guide spot

**Input:** \( V \) in \( G_\epsilon = (V, E) \) and \( \delta \)

**Output:** \( V(G_\epsilon)(\epsilon) \)

\[
V(G_\epsilon)(\epsilon) = \{ V_{\epsilon} \} \\
\text{for } i = 1 \text{ to } |V(G_\epsilon)| \\
\text{Find } V_{\epsilon} \text{ for } k = \{ u, v, x, y \}; \\
\text{Find } |(u, v)| = \min_{v \in V_k} |(u, v)|, k = \{ u, v, x, y \}; \\
\text{end_for}
\]

If the \( \epsilon_p \) is 100%, then \( G_\epsilon \) is a perfect grid. So, \( G_\epsilon \) is dependent on the expression rate and positional error rate of each spot. Figure 7 is an example of \( G_\epsilon = (V, E) \) obtained from real microarray image data (Alizadeh et al., 2000).

### Identification of spot address

The basic idea of spot addressing is that we can deduce the coordinates of other spots from the guide spots (anchor spots) by graph searching. Normally, guide spots are located in the corner vertices and are strongly expressed. If we know the position of \( \exists v \in V \), all other vertices, \( u \in V - v \), are addressed by applying the general graph traversal algorithm of \( G_\epsilon = (V, E) \). \( V(G_\epsilon) \) consists of the strong spots in which pixels \( I_{i, j} > T_0 \), where \( I_{i, j} \) is the intensity value of a pixel \((i, j)\) in the image data \( I \).

Let \( V_p \) denote a set of vertices which are not yet indexed and \( V_o \) denote a set of vertices which are already indexed. Let \( \hat{v} \) be a guide spot. The spot addressing algorithm is as follows:

**Algorithm:** Construct \( G_\epsilon = (V, E) \) from \( V(G_k^M) \)

**Input:** \( V(G_k^M) \) and \( \epsilon \)

**Output:** \( G_\epsilon = (V, E) \)

\[
V(G_\epsilon)(\epsilon) = V(G_k^M); \\
\text{for } i = 1 \text{ to } |V(G_\epsilon)(\epsilon)| \\
\text{Find } V_k \text{ for } k = \{ u, v, x, y \}; \\
\text{Find } |(u, v)| = \min_{v \in V_k} |(u, v)|, k = \{ u, v, x, y \}; \\
\text{end_for}
\]

Table 2. The results of computing \( \hat{P}_l \) in many artificial data set. In here the number of spots in each image data is 1600, \( T_0 = 30 \), and \( n = 20 \)

| Name | \(|V|\) | \(\epsilon_p\) | \(d_B\) | \(\hat{P}_l\) |
|------|------|------|------|------|
| ART1 | 174  | 0.11 | 3    | 0.999 |
| ART2 | 333  | 0.21 | 3    | 0.986 |
| ART3 | 491  | 0.31 | 3    | 0.975 |
| ART4 | 649  | 0.41 | 3    | 0.954 |
| ART5 | 808  | 0.51 | 3    | 0.932 |
| ART6 | 966  | 0.60 | 3    | 0.910 |
| ART7 | 1125 | 0.70 | 3    | 0.888 |
| ART8 | 1283 | 0.80 | 3    | 0.866 |
| ART9 | 1441 | 0.90 | 3    | 0.844 |
| ART10| 1600 | 1.00 | 3    | 0.822 |

**\( \epsilon \)-graph model of microarray image**

Now, we propose a spot indexing method for a single block. In order to address each spot, we use a new graph model. We define this model as \( \epsilon \)-graph, \( G_\epsilon = (V, E) \). Each edge of a vertex is classified as ‘up’, ‘down’, ‘left’, or ‘right’. Let \( P_u = (u, x, u, y) \) denote the \( x \) and \( y \) position of a vertex \( u \). Let \( V_u, V_d, V_l, \) and \( V_r \) denote four directional vertex sets of \( V_u \in V \) which satisfy one of the following conditions:

\[
|u.x - v.x| < \epsilon \text{ or } |u.y - v.y| < \epsilon,
\]

where \( v \in V_u, V_d, V_l, \) or \( V_r \). If \( u^x, u^d, u^l, \) and \( u^r \) are the neighbor vertices of \( u \), then

\[
|(u, u^k)| = \min_{v \in V_k} |(u, v)|,
\]

where \( k = \{ u, v, x, y \} \). Note that the degree of a vertex \( v \in V(G_\epsilon) \), \( \rho(v) \leq 4 \) and \( G_\epsilon \) is a directed graph. The algorithm for constructing \( G_\epsilon = (V, E) \) is as follows:
Fig. 8. An example of spot indexing procedure: (a) a guide spot is located at (1,1); (b) at the second step the vertex (1,1) links to the vertex (1,2); (c) at the third step vertex the vertex (2,3) links to the vertex (4,3); (d) all vertices are visited.

Output: Spot index, \( C_{ij} \) corresponding to \( V = V_p \); /* the set of not-indexed vertices */
\[ V_a = \hat{v} \] /* the set of indexed vertices */
do 
Find \( |(u, v)| = \min_{u \in V_p, v \in V_a} |(u, v)| \);
add \( u \) into \( V_a \);
compute grid index for \( u \) based on grid index of \( v \);
while \( |V_p| \neq 0 \)
If the spot address of a vertex \( v \), which is connected by a current visited vertex \( u \), is \( (i, j) \), then the address of \( u \) is \( (i + a, j + b) \) which is determined by the directional factor (up, down, left, and right) and the average distance, \( d_\mu \). If \( u \) is located at higher position than \( v \), then
\[ a = -\left[ \frac{|(u, v)|}{r_\mu} + d_\mu \right], \quad b = 0, \]
where \( r_\mu \) is the average radius of the spots. Positions that are lower or more to the left or right can be computed in a similar way. Figure 8 shows an example of a spot indexing procedure. Solid spots are represented by \( V_a \) and vacant spots by \( V_p \). As you can see, our algorithm always selects the shortest vertex in \( V_p \).

It is desirable to select the shortest vertex in \( V_p \), since it guarantees the correct spot addressing. Figure 9 shows the result of spot addressing based on the graph traversal algorithm.

If the resolution of a grid is \( M = r \times c \), where \( r \) denotes the number of rows and \( c \) the number of columns, then the time complexity of our algorithm is \( O(M^2) \). It is natural to assume that the number of strong spots is more than \( \frac{M}{3} \) and their positions are quite random. So, we can address every spot using our method.

**CORRECTING SKEWED IMAGE**
Correcting the skewed image is another important feature of microarray image analysis. Figure 10 is a typical image.

In order to solve this problem, Steinfath proposed a method where the slope of \( I \) is computed by averaging...
the slopes of the four sides that compose the quadrangle of the block (Steinfath et al., 2001). However, in this method, if the positional error rate is high, then the probability of misidentifying the real slope extremely increases. In this paper, in order to cope with this problem, we propose using the average slopes of the edges of \( G_{\epsilon} \), which connect the vertices of the grid structure.

Since \( G_{\epsilon} \) changes according to the value of \( \epsilon \), the slope of the grid structure can be found after determining the optimal \( \epsilon \) that best represents the grid structure. In order to measure how much \( G_{\epsilon} \) represents the grid structure, we propose defining the regularity of grid, \( R(G_{\epsilon}) \) as follows:

\[
R(G_{\epsilon}, r, c) = \frac{4 \cdot r \cdot c}{\sum_{i=0}^{4} \sum_{\text{dir}=1}^{4} q(v_i, \text{dir})},
\]

where \( r \times c \) means the resolution of grid, \( |V| \) denotes the number of vertices in \( G_{\epsilon} \), and \( q(v_i, \text{dir}) \) represents the difference of the addresses between \( v_i \) and four neighbor vertices. In the perfect grid structure, \( R(G_{\epsilon}, r, c) \) is always 1. To compute \( q(v_i, \text{dir}) \) the following is used:

\[
q(v_i, \text{dir}) = c_0 \cdot |h_r(v_i) - h_r(P(v_i, \text{dir}))| + c_1 \cdot |h_c(v_i) - h_c(P(v_i, \text{dir}))|,
\]

where \( h_r(v_i) \) denotes the row address of vertex \( v_i \) and \( h_c(v_i) \) the column address. \( P(v_i, \text{dir}) \) represents the vertex, which is connected to vertex \( v_i \) according to the direction \( \text{dir} \), e.g. \( \text{up}(\text{dir} = 1) \), \( \text{down}(\text{dir} = 2) \), \( \text{left}(\text{dir} = 3) \), and \( \text{right}(\text{dir} = 4) \). The weight constants, \( c_0 \) and \( c_1 \), are as follows:

\[
c_0 = \begin{cases} 
1 & \text{if } \text{dir} = 1 \text{ or } 2 \\
0 & \text{otherwise}
\end{cases}
\]

\[
c_1 = \begin{cases} 
1 & \text{if } \text{dir} = 3 \text{ or } 4 \\
0 & \text{otherwise}
\end{cases}
\]

where we set \( \epsilon = 20 \) by several experiments.

**EXPERIMENTAL RESULTS**

Our analysis system runs on Java platforms with any Java run time environment, having a minimum recommended RAM size of 512 Mb. It is written entirely in Java. A typical 1984×2004 16-bit image, containing 2688 spots, takes less than 10 s to analyze, including the block (spot) identification and intensity computation.

We tested our algorithm with several data sets (artificial and real data). There are four groups of test data in the following:

1. Artificial data (ART10, ART20, ···, ART100): their expression rates are from 10% to 100%. \( b_r \) and \( b_c \) is 2 and the number of rows \( (s_r) \) and columns \( (s_c) \) in each block is 20.

![Fig. 11. The number of connected components, \( |\hat{C}| \), with expression rate, \( \epsilon_P \), from 0.1 to 1.0 in artificial data.](image)

2. Real microarray data about cabbage gene (CBG1, CBG2, CBG3, CBG4, CBG5) : \( b_r = b_c = 4 \) and \( s_r = 12 \), \( s_c = 14 \), respectively.

3. Microarray image from NIH (NIH1, NIH2, NIH3) : Real data generated by Alizadeh et al. (2000) where \( b_r = b_c = 4 \) and \( s_r = s_c = 24 \).

4. Real microarray data from Stanford Univ. (STF1, STF2, STF3) : \( b_r = b_c = 2 \) and \( s_r = s_c = 44 \).

Now, we compute the number of components, \( |\hat{C}| \), to validate our probability model, \( \hat{P}_f \). Table 3 shows the results of our experiments. As is shown, the expected block joining (expected BJ) is quite similar to real block joining (real BJ). BJ represents the number of joined blocks from our algorithm. It validates our probability model in the real experiments. Since the quality of NIH data is very poor and block gap is so small, there are many joined blocks. If the quality of the image data is similar to NIH, then we strongly recommend that the block distance, \( d_B \), should be more than 3.

Figure 11 shows the variation of the number of components, \( |\hat{C}| \), from the artificial image data. We can see that if the expression rate is increasing, the accuracy of \( |\hat{C}| \) also increases. Therefore, if we want to adapt the fully automated system for image analysis, the expression rate must be at least 40% and the distance between two blocks must be more than three times the distance between the two spots. Figure 12 shows the probability that there is at least one inter-block edge between two blocks while varying the expression rate using artificial data whose \( n = 20 \). From this result, we can see that our method of block identification and that the probability model are feasible. Figure 13 shows a comparison of \( \hat{P}_f \) when \( d_B = 2, 3, 4, \) and 5.

The other image data, which have the grid structure for spot identification can be categorized into 3 sets.
Table 3. This table lists the results of computing the number of components, |\hat{\mathcal{C}}|, and validation of our probability model, \( \hat{P}_I \). |V| denotes the number of vertices in \( G^M \), \( e_p \) spot expression probability, \( b_r \) (\( b_c \)) the number of rows (columns) of blocks, and \( s_r \) (\( s_c \)) the number of rows (columns) of spots. Unit block means a real block and |{\hat{\mathcal{C}}}| denotes the number of connected components after MBR merging procedure. Finally, \( \hat{P}_I \) denotes the probability model of block boundary spots and BJ represents the number of the block joining.

| Data   | |V| | e_p | b_r | b_c | s_r | s_c | d_B | # of unit block | |{\hat{\mathcal{C}}}| | \( \hat{P}_I \) | Expected BJ | Real BJ |
|--------|------------------|-------|-----|-----|-----|-----|-----|-----|----------------|-----------------|-----------------|-----------------|-----------------|------------------|
| ART1   | 174.14           | 0.11  | 2   | 2   | 20  | 20  | 2   | 4   | 12.59          | 0.72            | 2.88            | 2.27            |
| ART2   | 333.00           | 0.21  | 2   | 2   | 20  | 20  | 2   | 4   | 11.25          | 0.74            | 2.96            | 2.13            |
| ART3   | 491.14           | 0.31  | 2   | 2   | 20  | 20  | 2   | 4   | 7.31           | 0.63            | 2.52            | 1.78            |
| ART4   | 690.10           | 0.41  | 2   | 2   | 20  | 20  | 2   | 4   | 4.76           | 0.48            | 1.92            | 1.37            |
| ART5   | 807.88           | 0.50  | 2   | 2   | 20  | 20  | 2   | 4   | 3.52           | 0.26            | 1.04            | 0.89            |
| ART6   | 965.99           | 0.60  | 2   | 2   | 20  | 20  | 2   | 4   | 3.16           | 0.11            | 0.44            | 0.50            |
| ART7   | 1124.91          | 0.70  | 2   | 2   | 20  | 20  | 2   | 4   | 3.21           | 0.03            | 0.12            | 0.25            |
| ART8   | 1283.27          | 0.80  | 2   | 2   | 20  | 20  | 2   | 4   | 3.53           | 0.01            | 0.04            | 0.08            |
| ART9   | 1441.45          | 0.90  | 2   | 2   | 20  | 20  | 2   | 4   | 3.89           | 0.00            | 0.00            | 0.02            |
| ART10  | 1600.00          | 1.00  | 2   | 2   | 20  | 20  | 2   | 4   | 4.09           | 0.00            | 0.00            | 0.00            |
| CBG1   | 2343             | 0.87  | 4   | 4   | 12  | 14  | 5   | 16  | 16             | 0.00            | 0.00            | 0.00            |
| CBG2   | 2613             | 0.97  | 4   | 4   | 12  | 14  | 5   | 16  | 16             | 0.00            | 0.00            | 0.00            |
| CBG3   | 2519             | 0.94  | 4   | 4   | 12  | 14  | 5   | 16  | 16             | 0.00            | 0.00            | 0.00            |
| CBG4   | 2461             | 0.92  | 4   | 4   | 12  | 14  | 5   | 16  | 16             | 0.00            | 0.00            | 0.00            |
| CBG5   | 2577             | 0.96  | 4   | 4   | 12  | 14  | 5   | 16  | 16             | 0.00            | 0.00            | 0.00            |
| NIH1   | 2582             | 0.28  | 4   | 4   | 24  | 24  | 2   | 16  | 2              | 0.74            | 13.32           | 14              |
| NIH2   | 3191             | 0.35  | 4   | 4   | 24  | 24  | 2   | 16  | 2              | 0.64            | 11.52           | 14              |
| NIH3   | 2844             | 0.31  | 4   | 4   | 24  | 24  | 2   | 16  | 2              | 0.70            | 12.60           | 9               |
| STF1   | 7595             | 0.98  | 2   | 2   | 44  | 44  | 2   | 4   | 4              | 0.00            | 0.00            | 0.00            |
| STF2   | 7581             | 0.98  | 2   | 2   | 44  | 44  | 2   | 4   | 4              | 0.00            | 0.00            | 0.00            |
| STF3   | 7570             | 0.98  | 2   | 2   | 44  | 44  | 2   | 4   | 4              | 0.00            | 0.00            | 0.00            |

Fig. 12. The probability, \( \hat{P}_I \), that there exists at least one inter-block edge between two blocks while varying the expression rate, \( e_p \). The distance, \( d_B \), between two neighbor blocks is 2, 3, 4, and 5.

Fig. 13. The probability curve of \( d_B = 2 \), 3, 4, and 5. X-axis is the expression rate \( e_p \) and Y-axis is probability \( \hat{P}_I \).

1. Artificial data(ARTs0, ARTs1, \cdots, ARTs10): Perfect grid structure has \( s_r = 20 \) and \( s_c = 20 \) in \( 1 \times 1 \) block.

2. Microarray data from NIH(NIHs1, NIHs2, NIHs3): Real data generated by Alizadeh et al. (2000) have \( s_r = 24 \) and \( s_c = 24 \).

3. CCD data(CCD1, CCD2, CCD3): Images of tissues located in 96-well plates were captured by CCD(Charge Coupled Device) camera. They have \( 4 \times 6 \) spots in a block.
We have implemented a fully automatic image analysis system from block indexing to spot quantification. Figure 14 shows some snapshots of our system. Figure 14a and b show successful spot indexing for a skewed image. Figure 14a is an artificial data (ARTs0) and b is its 10° inclined image. Figure 14c and d show real image data from NIH and the photosynthesis of a rice leaf. Figure 14 shows results which have more precise regions than the results in Figure 10.

We have experimented with several data sets for computing the rotation angle using artificial data and real data. We also attempted to apply our spot indexing algorithm to the skewed image. The specifications of the test data and experimental results are shown in Table 4. Our algorithm works completely even though the images inclined 10° (ARTs10).

CONCLUSIONS
In this paper, we proposed a fully automatic block and spot indexing algorithm for microarray image analysis. Our analytic model and experimental results show the feasibility of using MKNN and spot indexing algorithm. A large body of microarray image data (artificial, NIH, Stanford, and CCD) were tested. The main features of our algorithm are as follows:

- Using the minimum precondition (e.g. the resolution of blocks in image data and the resolution of spots in a single block), we can index the block and spot successfully.
- Our algorithm can be adapted to not only microarray image data, but also other grid structured image data.
- Our algorithm is robust within a 10° rotation of the grid structure.

Fig. 14. The results of spot indexing for some image data. (a) is an artificial data (ARTs0) which has good quality and (b) is a skewed image of which the angle is 10°. (c) and (d) are the real image data from NIH and the photosynthesis of a rice leaf, respectively.
Table 4. The results of computing the rotation angle for several sets of image data: + means clockwise rotation, and − counter-clockwise

<table>
<thead>
<tr>
<th>Data</th>
<th>s_r</th>
<th>s_c</th>
<th># V</th>
<th># E</th>
<th>R(G_e)</th>
<th>ε</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTs1</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>2</td>
<td>1.009</td>
</tr>
<tr>
<td>ARTs2</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>3</td>
<td>2.048</td>
</tr>
<tr>
<td>ARTs3</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>3</td>
<td>3.015</td>
</tr>
<tr>
<td>ARTs4</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>3</td>
<td>4.003</td>
</tr>
<tr>
<td>ARTs5</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>4</td>
<td>5.015</td>
</tr>
<tr>
<td>ARTs6</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>5</td>
<td>6.013</td>
</tr>
<tr>
<td>ARTs7</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>5</td>
<td>7.012</td>
</tr>
<tr>
<td>ARTs8</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>7</td>
<td>7.98</td>
</tr>
<tr>
<td>ARTs9</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>7</td>
<td>8.719</td>
</tr>
<tr>
<td>ARTs10</td>
<td>25</td>
<td>24</td>
<td>599</td>
<td>1149</td>
<td>0.96</td>
<td>7</td>
<td>9.323</td>
</tr>
<tr>
<td>NIHs1</td>
<td>24</td>
<td>24</td>
<td>449</td>
<td>850</td>
<td>0.926</td>
<td>9</td>
<td>0.076</td>
</tr>
<tr>
<td>CCD1</td>
<td>4</td>
<td>6</td>
<td>24</td>
<td>38</td>
<td>0.49</td>
<td>4</td>
<td>-3.6</td>
</tr>
<tr>
<td>CCD2</td>
<td>4</td>
<td>6</td>
<td>25</td>
<td>39</td>
<td>0.392</td>
<td>9</td>
<td>-5.335</td>
</tr>
<tr>
<td>CCD3</td>
<td>4</td>
<td>6</td>
<td>24</td>
<td>38</td>
<td>0.41</td>
<td>4</td>
<td>-5.725</td>
</tr>
</tbody>
</table>

- If the expression rate is more than 40%, and the distance between two neighbor blocks, $d_B$, is more than three times the unit distance, $d_u$, then our algorithm can give a fully automatic image analysis.
- If $d_B \geq 5$ then our algorithm can index each block automatically even if the expression rate $\leq 10\%$.

REFERENCES


