Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

A novel Fourier descriptor based image alignment algorithm for automatic optical inspection

Chin-Sheng Chen [a,*], Chun-Wei Yeh [a], Peng-Yeng Yin [b]

[a] Institute of Automation Technology, National Taipei University of Technology, NTUT Box 4325, 1, Sec. 3, Chung-Hsiao E. Rd., Taipei, Taiwan 10608, Taiwan, ROC
[b] Department of Information Management, National Chi-Nan University, Taiwan, ROC

A R T I C L E   I N F O

Article history:
Received 4 April 2008
Accepted 5 November 2008
Available online 14 November 2008

Keywords:
Fourier descriptor
Image alignment
Automatic optical inspection
Component detection
Contour tracing
Run length encoding
Blobs tables
Phase-shifted technique

A B S T R A C T

This paper presents a Fourier descriptor based image alignment algorithm (FDBIA) for applications of automatic optical inspection (AOI) performed in real-time environment. It deliberates component detection and contour tracing algorithms and uses the magnitude and phase information of Fourier descriptors to establish correspondences between the target objects detected in the reference and the inspected images, so the parameters for aligning the two images can be estimated accordingly. To enhance the computational efficiency, the proposed component detection and contour tracing algorithms use the run length encoding (RLE) and Blobs tables to represent the pixel information in the regions of interest. The Fourier descriptors derived from the component boundaries are used to match the target objects. Finally, the transformation parameters for aligning the inspected image with the reference image are estimated based on a novel phase-shifted technique. Experimental results show that the proposed FDBIA algorithm sustains similar accuracy as achieved by the commercial software *Easyfind* against various rotation and translation conditions. Also, the computational time consumed by the FDBIA algorithm is significantly shorter than that by *Easyfind*.

1. Introduction

Image alignment is a fundamental technique for many applications of machine vision and image processing, including image retrieval, object recognition, pose estimation, industrial inspection, target tracking, just to name a few. As the applications in industrial inspection are usually undertaken with well-controlled environments (such as fixing the distance between the lens and the inspected objects), it is common practice to perform image alignment with constrained two-dimensional (2-D) rigid transformation, like rotation-invariant with limited scaling compensation, in this context. This creates a possibility to establish a fast image alignment algorithm for the applications in industrial inspection, which is the focus of this paper.

Image alignment techniques are generally divided into two major categories, area-based (or intensity-based) and feature-based (or geometry-based) methods [1]. The area-based method is sometimes called the correlation-like or template matching method and it has been very popular in the past decades due to its basic concepts. Firstly, the small reference template is applied in a large scene image by sliding the template window on a pixel-by-pixel basis and the normalized cross correlation (NCC) is computed between the template and the scene image. Then, the maximum values or peaks of the computed correlation values indicate the matches between a template and sub-images in the scene. The NCC metric is often used to deal with the registration of images which differ only by a translation. If images are deformed by more complex transformations, the template window cannot cover the same regions of interest in the reference and scene images. Therefore, several researchers have proposed modified NCC metrics to circumvent the image registration problem with complex transformations. In order to make the image matching invariant to rotation, a ring-projection transformation was proposed that transforms a two-dimensional (2-D) gray level image into a rotation-invariant representation as in the 1-D ring-projection space [2]. Tsai and Chiang [3] further used the ring-projection to represent a target pattern in wavelet-decomposed sub-images, using only the pixels with high wavelet coefficients at lower resolution level to compute the NCC between two compare patterns. In addition, Choi and Kim [4] presented a two-stage image alignment method that first finds candidates by comparing the vector sums of ring-projection, and then further matches these candidates based on the rotation-invariant moments.

Among the feature-based methods, Huttenlocher et al. [5–6] applied the directed Hausdorff distance to develop several efficient algorithms for image alignment. Kown et al. [7] proposed a robust hierarchical Hausdorff distance to compare edge maps in a multi-level pyramid structure. In addition, Chen et al. [8] used the Hausdorff distance for image alignment in a PCB inspection system. On

* Corresponding author. Fax: +886 2 87733217.
E-mail address: saint@ntut.edu.tw (C.-S. Chen).

1047-3203/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2008.11.003

the side, local descriptors are commonly employed in a number of real-world applications such as object recognition. Lowe [9] proposed a scale invariant feature transform (SIFT), which combines a scale invariant region detector and a descriptor based on the gradient distribution in the detected regions. SIFT focuses on local feature at particular interest points and are invariant to scale and rotation. Mikolajczyk and Schmid [10] compare the performance of descriptors computed for local interest regions. The above invariant descriptors of image interest points are resistant to partial occlusion, and are relatively insensitive to changes in viewpoint. The above algorithms are efficient in general applications, however, they could waste excrescent processing when the image alignment only involves rotation and limited scaling compensation as is the application of AOI.

Overall, Table 1 summarizes the features, approaches and disadvantages for each category of image alignment methods. The area-based category adopts image templates as matching features and uses cross correlation or ring-projection to register the images, but it could induce expensive computation time for applications involving complex image transformation. On the other hand, the feature-based category proposes edge maps, interest points, shape contours and invariant descriptors as alignment features. The embedded approaches are Hausdorff distance, feature correspondence, and transformation estimation. The disadvantage of feature-based category is the alignment error caused by inaccurate feature extraction.

Although many researchers have investigated the feature-based image alignment techniques, relatively few works have been developed based on shape-based alignment which can be considered a special type of the feature-based methods. The shape information could be described with contour-based and region-based methods [11]. The contour-based shape description exploits only boundary information about objects and involves less computation time than that incurred by the region-based shape description. Contour-based shape descriptors can be conveniently applied to applications where shape contours are available by using the contour tracing technique [12]. Many contour-based shape descriptors have been proposed. The improved moment invariants [13], which are computed by using the shape boundaries only, reduce the computations incurred by the traditional moment invariants. Zhang and Lu [14] evaluated and compared two important and promising shape descriptors, namely, the Fourier descriptor (FD) and the curvature scale space descriptor (CSSD). The effectiveness and efficiency of FD have been found superior to that of CSSD for image retrieval. The FD is a convenient shape descriptor and there are a number of modified FD methods [15–17]. However, the above descriptors are designed for rotation-invariant matching and they are not attempting to estimate the rotation angle between two corresponding shapes in different images for performing image alignment as requested in the applications of industrial inspection.

In automatic optical inspection (AOI) applications, the geometric relationship between the reference image R and inspected image S is constricted as shown in Fig. 1. There usually exists an indicated inspection mark (such as the diamond marks as shown in Fig. 1) in the images to alleviate the computational burden for performing real-time image alignment. Because the working distance between the lens and the inspected target is fixed in AOI applications, the scaling variation between R and S is tiny and can be neglected. Hence, the image alignment problem for AOI can be formulated as finding the optimal 2-D rigid transformation between R and S constrained by translation vector $\vec{T}$ and rotation angle θ.

Table 1
Summary of image alignment methods.

Category	Features	Approaches	Disadvantages
Area-based	Image templates	Cross correlation Ring-projection	Expensive computation time
Feature-based	Edge maps Interest point Shape contours Invariant descriptors	Hausdorff distance Feature correspondence Transformation estimation	Inaccurate feature extraction

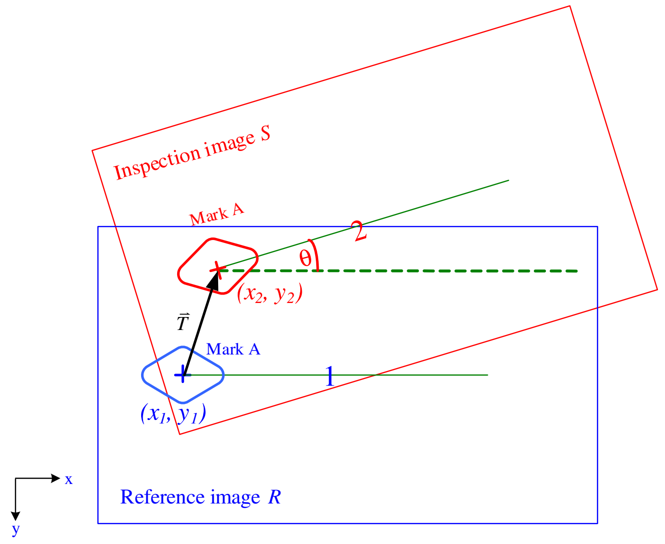


Fig. 1. The geometric relationship between reference image R and inspection image S.

This paper proposes a Fourier descriptor based image alignment (FDBIA) algorithm for AOI applications. The method first applies a component detection algorithm based on the run length encoding (RLE) and Blobs to represent the regions of interest. The pixel information stored in the RLE and Blob tables is fed into our contour tracing method for identifying the boundaries which are then described by sequences of FD. As the scaling variation is tiny for AOI applications, the area of the aligning marks is similar and can be used as a constraint to match the most probable candidate objects and expedite the process. The magnitude and phase information of FD are used to establish correspondences between the aligning target objects detected in the reference and the inspected images, so the parameters for aligning the two images can be estimated accordingly.

The contributions of this paper for AOI application include the following. (1) Our contour tracing method takes advantage of the pixel information stored in RLE and Blob tables so we do not have to retrace the whole boundaries of the regions of interest in the original image. (2) While in the context of AOI the working distance between the lens and the inspected objects is fixed, it allows us to apply the area constraints to match the target objects with similar sizes only. (3) The FD transformation is applied to conduct the object matching robustly. (4) A bounding criterion is used to expedite the computation for the object matching, and (5) a novel phase-shifted technique is proposed to accurately estimate the rotation shift between the aligning objects identified by the FD. The above salient features make our method suitable for AOI applications where real-time processing and accurate alignment are rudimentary.

The remainder of this paper is organized as follows. In Section 2, the architecture of the FDBIA algorithm is described. Section 3 presents the details of the procedures performed in feature extraction for conducting subsequent object matching. Section 4 describes the novel phase-shifted orientation estimation. The experimental results are presented in Section 5 and the conclusions of this work are given in Section 6.

2. Architecture of the FDBIA algorithm

Fig. 2 shows the architecture of the FDBIA algorithm. The FDBIA algorithm consists of three phases: (1) the training phase, (2) the matching phase, and (3) the alignment phase. In the training phase, the reference image R is input and the region of interest (ROI) containing the inspection mark in R is specified manually. The algorithm then detects the inspection mark in the ROI by the connected-component detection method and produces the RLE and Blob tables tallying the component information. With this information, the area of the inspection mark can be computed and will be used as a constraint to locate the most probable candidate components in the inspected image. The contour tracing technique identifies the pixels $r^R(n = 0, 1, \ldots, N-1)$ along the object boundary by reference to the information stored in the RLE and Blob tables. Then, the FDs $\phi_i^R(i = 1, 2, \ldots, p)$ for the boundary pixels are derived, where p is the number of Fourier descriptors. These FDs are fed into the matching phase and the alignment phase, respectively. For the matching phase, the inspected image S is input and all the connected components in S are detected. However, only the candidate components with similar size to that of the inspection mark are subject to performing contour tracing to save the computation. Let $r_m^S(m = 1, 2, \ldots, k;\ n = 0, 1, \ldots, N-1)$ be the contour boundary information of the k candidate components, the corresponding FDs $\phi_i^S(m)(i = 1, 2, \ldots, p; m = 1, 2, \ldots, k)$ can be thus computed. We apply a fast minimum distance matching to identify the target component based on a bounding technique to expedite the computation. Finally, in the alignment phase, the translation estimation is performed by using the vector between the centers of the target components detected in R and S. The orientation estimation is fulfilled by using a novel phase-shifted technique by reference to the FDs.

The details of the three phases performed in our FDBIA algorithm are articulated in the following subsections.

2.1. Training phase

The training phase is applied to obtain the area and the FD information of the interest object (inspection mark) in the reference image R. The precise steps are presented as follows.

Step 1: Selecting the region of interest (ROI) containing the inspection mark in the reference image R.

Step 2: Selecting an appropriate threshold for different inspection conditions.

Step 3: Our connected-component detection method is applied to attain the RLE and Blob tables for the interest object in the ROI, and the area of the interest object in terms of the number of pixels is sent to the matching phase for activating the area constraint.

Step 4: After the connected-component detection process, the contour $r^R(n = 0, 1, \ldots, N-1)$ of the interest object is traced by using the contour tracing technique by reference to the RLE and Blob tables, where N is the number of boundary points.

Step 5: The FDs $\phi_i^R(i = 1, 2, \ldots, p)$ for the interest object are calculated, where p is the number of Fourier descriptors. These FDs are fed into the matching phase and the alignment phase for minimum distance matching and the translation and rotation estimation, respectively.

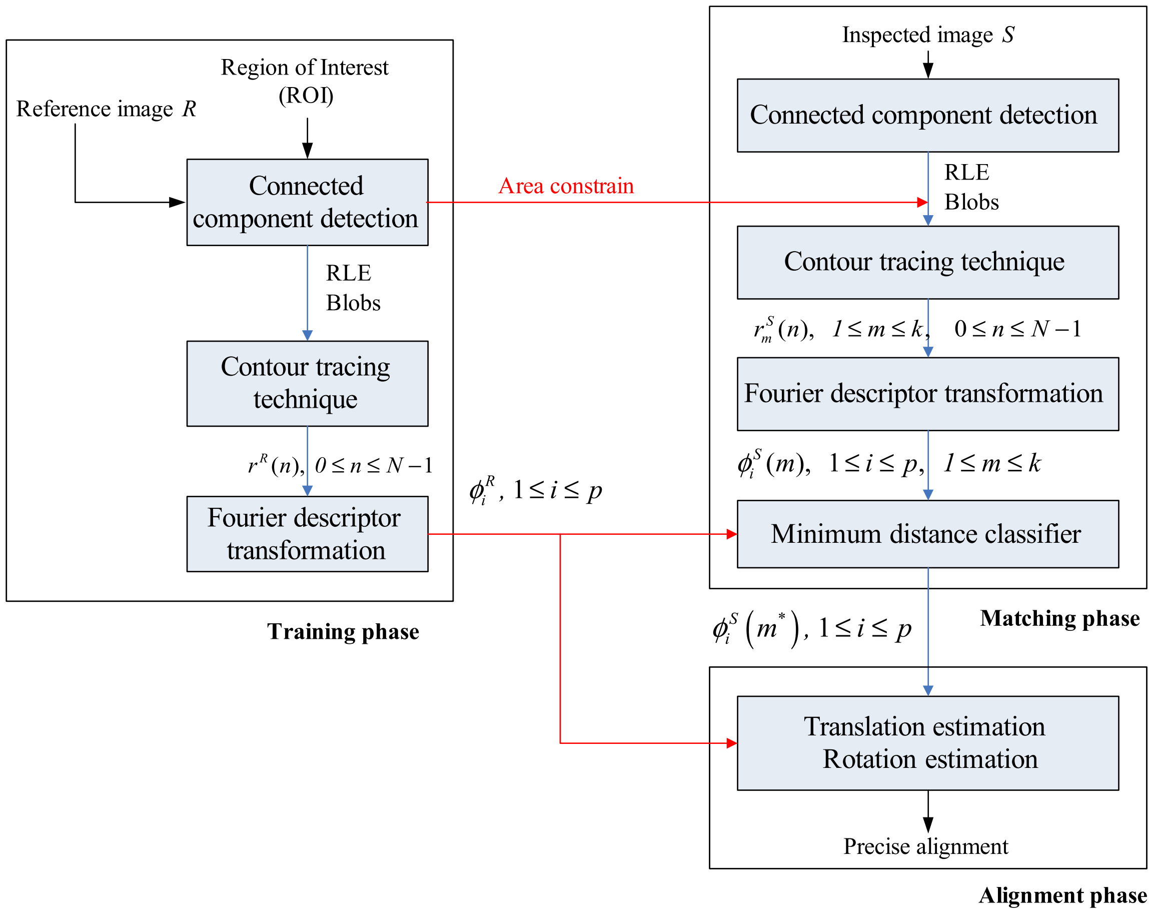

Region of Interest (ROI)
Reference image R
Connected component detection
RLE Blobs
Contour tracing technique
$r^R(n),\ 0 \le n \le N-1$
Fourier descriptor transformation
Training phase
Area constrain
$\phi_i^R,\ 1 \le i \le p$
Inspected image S
Connected component detection
RLE Blobs
Contour tracing technique
$r_m^S(n),\ 1 \le m \le k,\ 0 \le n \le N-1$
Fourier descriptor transformation
$\phi_i^S(m),\ 1 \le i \le p,\ 1 \le m \le k$
Minimum distance classifier
Matching phase
$\phi_i^S(m^*),\ 1 \le i \le p$
Translation estimation Rotation estimation
Precise alignment
Alignment phase

Fig. 2. Architecture of the proposed FDBIA algorithm.

2.2. Matching phase

The matching phase is applied to obtain the area and the FD information of the candidate objects in the inspected image S for conducting the area constraint and minimum distance matching. The precise steps are described as follows.

Step 1: The proposed connected-component detection is used to detect the objects in S. A predetermined threshold is embedded in the detecting process to determine whether the gray level pixel belongs to an object pixel or a background pixel on-the-fly.

Step 2: The contours $r_m^S(m = 1, 2, \ldots, k;\ n = 0, 1, \ldots, N-1)$ of the k candidate objects subject to the area constraint are traced, and the FDs of the contours of the candidate objects are computed. Let $\phi_i^S(m)(i = 1, 2, \ldots, p; m = 1, 2, \ldots, k)$ denote the p Fourier descriptor features for a candidate object indexed by m in the inspected image S.

Step 3: The minimum distance matching is performed for shape localization using the FDs. The normalized distance between the FDs of the interest object in R and those of a candidate object in S is defined as follows.

$$d_m = \sqrt{\sum_{i=1}^{p}\left[\frac{\phi_i^R - \phi_i^s(m)}{\phi_i^r}\right]^2}, \quad m = 1, 2, \ldots, k \tag{1}$$

The candidate object having the minimum value of Eq. (1) is identified as the target object. Because the minimum distance matching applied in Eq. (1) is based on a non-decreasing function, we employed a bounding technique to expedite the computation. The distance between $\phi_i^R(i = 1, 2, \ldots, p)$ and $\phi_i^s(m)(i = 1, 2, \ldots,\ p; m = 1)$ is first computed and is used as a bound to terminate the distance computation for the next candidate object $\phi_i^s(m)(i = 1, 2, \ldots, p; m = 2, \ldots, k)$. That is, the distance computation for the next candidate object is terminated if the intermediate distance value is already greater than the current bound. If the complete distance value is less than the current bound, then the bound is replaced by the new distance value and the best matching is updated to the corresponding object. The computing and bounding process is repeated until the distance computation for the last candidate object is finished. Finally, the FDs of the target object, $\phi_i^s(m^*)$, are fed into the alignment phase for estimating the translation and rotation parameters.

2.3. Alignment phase

After the matching phase, the information of rotation θ and translation $\vec{T}$ are calculated between the interest object (inspection mark) in R and the target object in S. The translation information can be gained from the positional difference between centers of gravity of the interest object and the target object. Then, the rotation information is obtained by using the proposed phase-shifted technique as noted in Section 4.

3. Feature extraction

The feature extraction process consisting of three steps, namely the connected-component detection, contour tracing technique, and Fourier descriptor transformation, is performed in both of the training phase and the matching phase. The complete procedures of the feature extraction process are described as follows.

3.1. Connected-component detection

The traditional connected-component labeling algorithm uses a label matrix where all pixels belonging to the same connected component are assigned a unique label identifying that component. However, the FDBIA algorithm does not use a label matrix, we only need partial information of the components depending on our AOI application to proceed with. We first review the existing general connected-component labeling algorithms as follows.

The Line-Scan Cluster (LSC) algorithm, a one-pass algorithm for labeling arbitrarily connected components, was first introduced [18]. Then, Suzuki et al. [19] used a one-dimensional table that could memorize label equivalences and unit equivalent labels successively during the two-pass scanning in forward and backward raster directions. Chang et al. [20] further presented a one-pass component labeling algorithm that uses a contour tracing technique to detect the external contour and possible internal contours of each component. A general approach to handle the label equivalence information is to use Union-Find algorithm with pointer-based rooted trees [21]. Wu et al. [22] used an array instead of the pointer-based rooted trees to expedite the connected component labeling process. Hu et al. [23] proposed the component labeling algorithm based on iterative recursions, where one directly labels an original binary image while the other labels the boundary voxels followed by one-pass labeling of non-boundary object voxels. Martín-Herrero proposed a hybrid object-labeling method [24,25] which combines recursion with iterative scanning and is able to mitigate the entailed cost by efficiently reducing the depth of recursive calls and the stack size, hence applicable to real-time and 3-D labeling applications. Two run-based connected-component labeling algorithms have been proposed with run length encoding. Shima et al. [26] proposed the propagation-type algorithm by use of block sorting and tracing of runs, and label propagation to the connected runs. The other algorithm, proposed in [27], is a run-based two-scan labeling algorithm. It records run data in a queue during the first scan, and the run data are used for detecting the connectivity in the further processing. When the first scan is finished, all provisional labels that were assigned to each connected component in the given image will be combined in the same provisional label set, and will have a unique representative label. In the second scan, each provisional label is replaced by its representative label.

Our connected-component detection algorithm also uses the run length encoding (RLE) but differs with the above noted algorithms in several aspects. The existing algorithms are the outstanding labeling algorithms for general cases. However, the FDBIA algorithm does not use a label matrix to assign a unique label identifying the component, we only need partial information of the components. For this reason, we proposed a component-detecting algorithm which uses two data structures, the RLE and Blob tables, to gain the information we need. Assume that our algorithm works with binary images containing either *object* or *background pixels*. We use $I(x,y)$ to denote the pixel value at coordinates (x,y) in the image. A run is a block of contiguous *object pixels* in a row. A run from $I(x_s,y)$ to $I(x_e,y)$ is described by $r(x_s,x_e,y)$. Let $r(x_s,x_e,y)$ be the run currently scanned, all its connecting runs contained in the preceding row can be easily determined by reference to the eight-connectivity criterion. Formally, a run $r(x_a,x_b,y_n)$ satisfying $x_a \geqslant x_s - 1, x_b \leqslant x_s + 1$, and $y_n = y - 1$ is connected to the current run $r(x_s,x_e,y)$. With this simple connectivity property, we can perform raster scan to the input image and obtain connected run information between two consecutive rows and the connected component information based on all connected runs. The two types of information are stored in the RLE and Blob tables, respectively.

The *RLE table* records four attributes (Xs, Xe, Y, Next) of each run, where Xs and Xe denote the starting and ending *x*-coordinates, Y is the *y*-coordinate and Next is the index of the connected run contained in the proceeding row (clearly, the value of Next is NULL when the current run is being processed but can be determined when the scan of the next row is finished). When a longer run has two shorter adjacent runs in the lower row, the smaller run number would go in "Next". On the other hand, the *Blob table* records four attributes (Parent, Size, First, Last) of each blob, where Parent indicates the index of the parent blob, Size is the number of runs contained in the current blob, First and Last are the *y*-coordinates of the beginning and ending runs of the current blob. Note that a connected component with concave contour may be represented by multiple blobs due to the raster scan fashion, and in this case the connected component will be represented as a blob tree whose root is the first blob in the connected component encountered in the raster scan. Then all the blobs, belonging to the same blob tree, are merged by the ***Union Find*** operation into a single one, and RLE table corresponding to final run of blobs will be modified. The Parent attribute is NULL if the corresponding blob is the root of the blob tree.

Fig. 3 shows the variations of the RLE and Blob tables before and after scanning a given row. Assume we have scanned the first three rows (indexed from 0 to 2) of the image. Using the previously noted connectivity property, the connected run and connected component information can be derived and stored in the RLE and Blob tables as shown in Fig. 3(b) and (d). The Next attribute of the entries 2 and 3 in the RLE table is NULL because the next row (row 3) has not been processed yet, and the Parent attribute of the first two entries in the Blob table is NULL because the two blobs are encountered first in the individual connected components. Now we proceed with the scanning of row 3. It is easy to find that run 2 and run 3 are connected to run 4, so their Next attributes are replaced by 4 (see Fig. 3(c)). Analogously, we observe that blob 0 and blob 1 are actually belonging to the same connected component and they are merged by the ***Union Find*** operation into a single one as shown in Fig. 3(e). Furthermore, the Next attribute of run 2 is replaced by 1 as shown in Fig. 3(c).

Fig. 4 illustrates a more complex case of the connected-component detection process where the connected component has concave contours. Before scanning row 9 of the original image, the entry values of the RLE and Blob tables are shown in Fig. 4(a). Twenty runs (indexed from 0 to 19) have been identified in the RLE tables and the three blobs with NULL parent in the Blob table indicated the three connected components currently perceived. Blob 2 is the child of blob 1 because they were found connected through run 11. Similarly, blob 0 is the parent of blob 3 since they were connected through run 12. After scanning row 9 of the original image, the updated entry values of the RLE and Blob tables are shown in Fig. 4(b). Twenty-one runs (indexed from 0 to 20) have been identified in the RLE tables and the only blob (indexed 0) with NULL parent in the Blob table revealed that there is actually just one connected component existing in the original image. The other four blobs (indexed from 1 to 4) are found connected to the same component due to run 20.

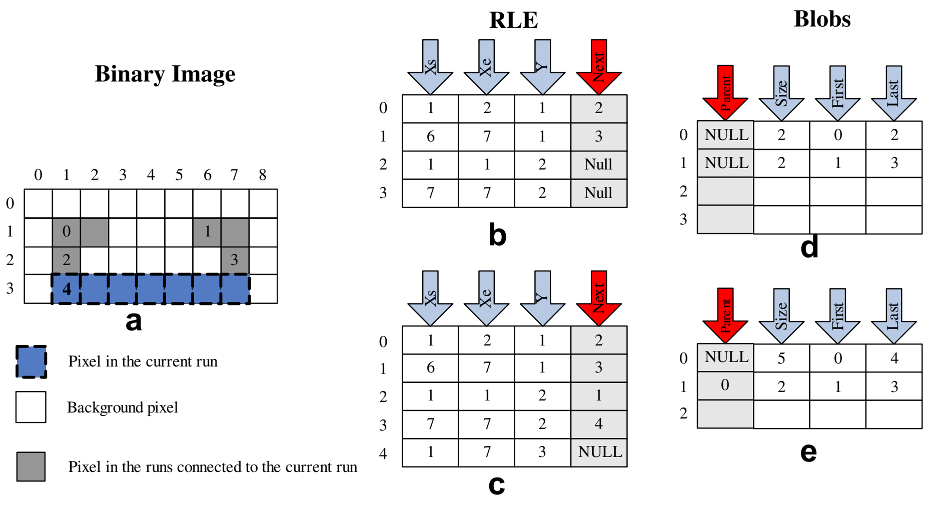
From the above steps, with a single pass scanning of the original image we get all the connected component information we need to proceed with the contour tracing. The advantage of using the RLE and Blob tables having the location information of all pixels in each component is remarkable for our AOI applications because the detected blobs are very small compared to the original image, and this means a significant amount of time saving.

3.2. Contour tracing technique

The goal of the contour tracing technique [12] is to find the location information of the external contours of a connected component. With the location information of all pixels in each component stored in the RLE and Blob tables, we are able to find the external contours. Our tracing algorithm firstly finds blob 0 from the Blob table, and then retrieves the first run belonging to blob 0 in the RLE table. Using the adjacency information of consecutive runs stored in the RLE table, we can obtain the location information of boundary pixels. The successive blobs will go through the same process. Finally, we can get all the boundary information as needed. For example, the contour information of the connected component in Fig. 4 is shown in Fig. 5. The boundary contour represents the shape signature of the interest object and we will use the Fourier descriptor to transform the signature.

3.3. Fourier descriptor transformation

The contour-based shape description exploits only boundary information about the object and involves less computation than that required by the region-based shape description. So we use



(b) RLE

	Xs	Xe	Y	Next
0	1	2	1	2
1	6	7	1	3
2	1	1	2	Null
3	7	7	2	Null

(c) RLE

	Xs	Xe	Y	Next
0	1	2	1	2
1	6	7	1	3
2	1	1	2	1
3	7	7	2	4
4	1	7	3	NULL

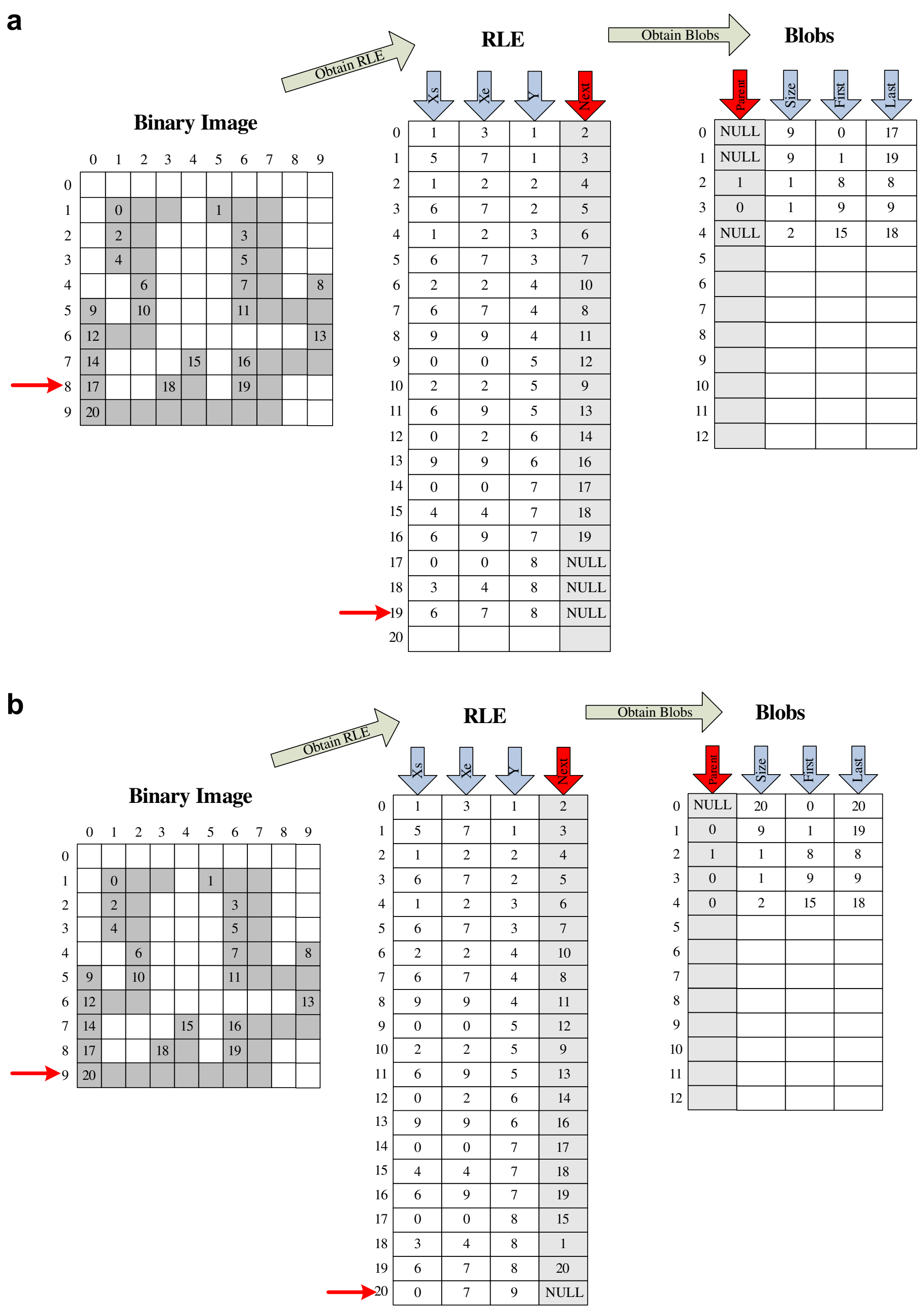
(d) Blobs

	Parent	Size	First	Last
0	NULL	2	0	2
1	NULL	2	1	3
2				
3				

(e) Blobs

	Parent	Size	First	Last
0	NULL	5	0	4
1	0	2	1	3
2				

Fig. 3. Connected-component detection: (a) a binary image; (b) RLE table before scanning row 3 of the original image; (c) RLE table after scanning row 3 of the original image; (d) Blobs table before scanning row 3 of the original image; (e) Blobs table after scanning row 3 of the original image.



(a) RLE

	Xs	Xe	Y	Next
0	1	3	1	2
1	5	7	1	3
2	1	2	2	4
3	6	7	2	5
4	1	2	3	6
5	6	7	3	7
6	2	2	4	10
7	6	7	4	8
8	9	9	4	11
9	0	0	5	12
10	2	2	5	9
11	6	9	5	13
12	0	2	6	14
13	9	9	6	16
14	0	0	7	17
15	4	4	7	18
16	6	9	7	19
17	0	0	8	NULL
18	3	4	8	NULL
19	6	7	8	NULL
20				

(a) Blobs

	Parent	Size	First	Last
0	NULL	9	0	17
1	NULL	9	1	19
2	1	1	8	8
3	0	1	9	9
4	NULL	2	15	18
5				
6				
7				
8				
9				
10				
11				
12				

(b) RLE

	Xs	Xe	Y	Next
0	1	3	1	2
1	5	7	1	3
2	1	2	2	4
3	6	7	2	5
4	1	2	3	6
5	6	7	3	7
6	2	2	4	10
7	6	7	4	8
8	9	9	4	11
9	0	0	5	12
10	2	2	5	9
11	6	9	5	13
12	0	2	6	14
13	9	9	6	16
14	0	0	7	17
15	4	4	7	18
16	6	9	7	19
17	0	0	8	15
18	3	4	8	1
19	6	7	8	20
20	0	7	9	NULL

(b) Blobs

	Parent	Size	First	Last
0	NULL	20	0	20
1	0	9	1	19
2	1	1	8	8
3	0	1	9	9
4	0	2	15	18
5				
6				
7				
8				
9				
10				
11				
12				

Fig. 4. Connected-component detection: (a) RLE and Blob tables before scanning row 9 of the original image; (b) RLE and Blob tables after scanning row 9 of the original image.

Fourier descriptor (FD) to describe the shape boundary. Since the candidate objects in the inspected image may have translation and rotation differences with the interest object in the reference image, the similarity measure for matching the objects should be invariant against translation and rotation. A comparison between shape descriptors can be found in [14] and the FD is more malleable than other descriptors in the above aspect.

The FD is derived by applying the Fourier transformation to a shape signature. The set of normalized Fourier transformed coefficients is called the FD of the shape. The shape signature is a 1-D function representing 2-D boundaries, and it usually uniquely describes a shape. Different shape signatures have been exploited to obtain FD. Complex coordinates, the curvature function, cumulative angular function, and centroid distance are the commonly used shape signatures. Among these, the centroid distance has been empirically shown to be the most effective one [28]. Assume that the coordinates of the pixels along the shape contour boundary are $(x(n), y(n))$, $n = 0, 1, 2, \ldots, N-1$, where N indicates the total number of boundary points. The *centroid distance function* is defined by the Euclidean distance between the boundary points and the centroid (x_c, y_c) of the shape, *i.e.*,

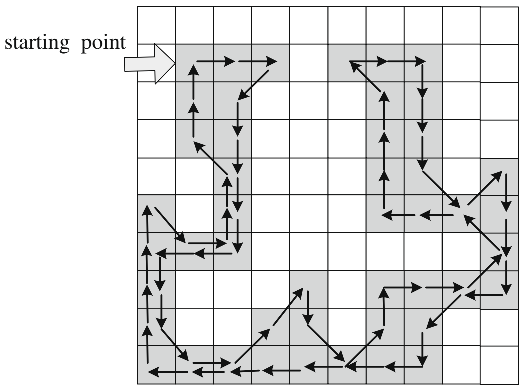


Fig. 5. The contour information of the connected component in Fig. 4.

$$r(n) = ([x(n) - x_c]^2 + [y(n) - y_c]^2)^{1/2} \tag{2}$$

where

$$x_c = \frac{1}{N}\sum_{n=0}^{N-1} x(n), \quad y_c = \frac{1}{N}\sum_{n=0}^{N-1} y(n)$$

The centroid distance signature $r(n)$ is invariant to translation. However, the rotation of shape will cause a circular shift in set $r(n)$, and the scaling of shape will increase or decrease the value of $r(n)$ linearly.

For the centroid distance signature $r(n)$, its discrete Fourier transform is given by

$$a_u = \frac{1}{N}\sum_{n=0}^{N-1} r(n)\exp(-j2\pi un/N), \quad u = 0, 1, \ldots, N-1 \tag{3}$$

This results in a set of Fourier coefficients $\{a_u\}$, which is a representation of the shape signature $r(n)$. Since a shape generated through rotation, translation and scaling comes out a similar shape, the shape representation based on FD should be invariant against these conditions.

The general form for the Fourier coefficients of a contour generated by translation, rotation and scaling is given by:

$$a_u = \exp(ju\tau)\cdot s\cdot a_u^{(o)}, \quad u \neq 0 \tag{4}$$

where $\exp(ju\tau)$ and s are the rotation and scaling factors, respectively, and the basic phase shift τ is given as:

$$\tau = \frac{360^\circ}{N}\cdot n \tag{5}$$

Now, consider the following expression,

$$b_u = \frac{a_u}{a_0} = \frac{\exp(ju\tau)\cdot s\cdot a_u^{(o)}}{s\cdot a_0^{(o)}} = \frac{a_u^{(o)}}{a_0^{(o)}}\exp[ju\tau] = b_u^{(o)}\exp[ju\tau] \tag{6}$$

where a_0 is the DC component; b_u and $b_u^{(o)}$ are the normalized Fourier coefficients of the inspected shape and the original shape, respectively. From Eq. (6), there is only a phase shift $u\tau$ between b_u and $b_u^{(o)}$, and the magnitude of $|b_u|$ and $|b_u^{(o)}|$ are the same. In other words, $|b_u|$ is invariant to translation, rotation and scaling. Hence, the magnitude set of the normalized Fourier coefficients corresponding to any shape, $\{|b_u|, 0 < u < N\}$, can be used as the shape descriptor, denoted as $\{SD_u, 0 < u < N\}$. The normalized *SD* feature is in [0,1] because a_0 is normally the largest coefficient in *SD*. More detailed definitions of the Fourier descriptor can be found in the literature [14,28]. We retain 10 Fourier coefficients to represent the shape boundary and empirically found that they are quite robust against rotation and translation.

4. Translation and rotation estimation

As in AOI applications (Fig. 1) the translation parameter $\vec{T}$ and rotation parameter θ should be estimated between the interest object and the target object contained in the reference and the inspected images, respectively. The translation parameter can be estimated from the difference between the centers of gravity of the two objects. Assume that the coordinates of pixels in the objects contained in the reference and inspected image are $(x_{RA}(i), y_{RA}(i)), i = 0, 1, 2, \ldots, q-1$ and $(x_{SA}(i), y_{SA}(i)), i = 0, 1, 2, \ldots,$

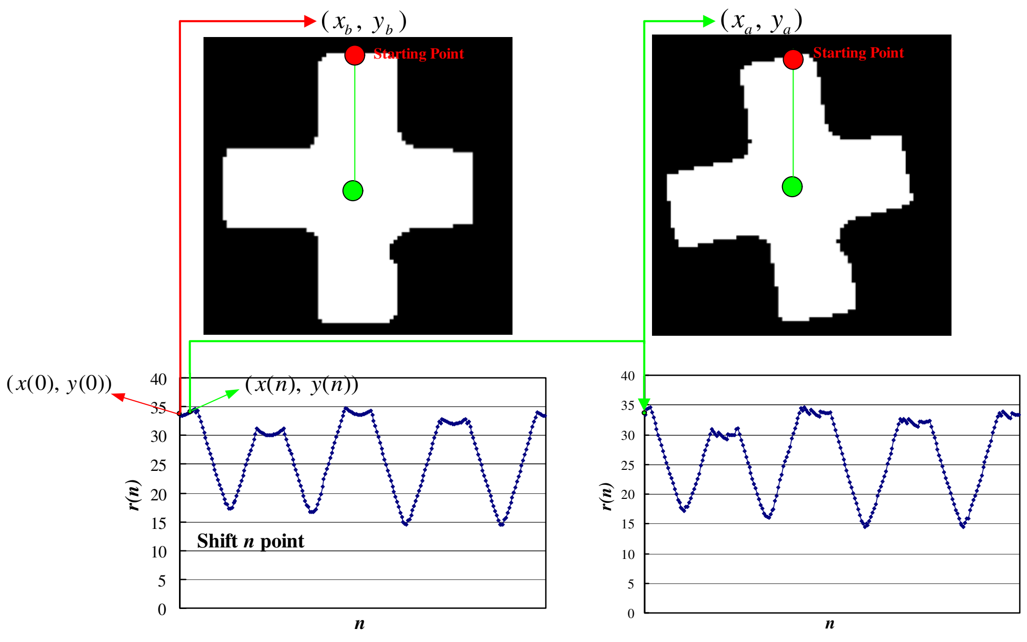


Fig. 6. The example for the rotation of a shape.

$q-1$, where q is the number of pixels in each object, the centers of gravity of the individual object can be calculated by

$$x_{Rc} = \frac{1}{q}\sum_{i=0}^{q-1} x_{RA}(i), \quad y_{Rc} = \frac{1}{q}\sum_{i=0}^{q-1} y_{RA}(i) \tag{7a}$$

$$x_{Sc} = \frac{1}{q}\sum_{i=0}^{q-1} x_{SA}(i), \quad y_{Sc} = \frac{1}{q}\sum_{i=0}^{q-1} y_{SA}(i) \tag{7b}$$

Then, the translation parameter $\vec{T}$ is derived by

$$\vec{T} = (x_{Sc}, y_{Sc}) - (x_{Rc}, y_{Rc}) \tag{8}$$

On the other hand, the phase information of FD is adopted to estimate the rotation angle of the shape. As mentioned in Section 3.3, the normalized Fourier coefficients of the inspected shape, b_u, and the original shape, $b_u^{(o)}$, exist only as phase-shifted $\exp[ju\tau]$ due to the rotation of the inspected shape. The starting point in the centroid distance signature $r(n)$ of the shape will be changed due to image rotation. Then, we can derive the phase information from the Fourier coefficients of $r(n)$. Then, the rotation angle is estimated from the different $\exp[ju\tau]$ between b_u and $b_u^{(o)}$. The phase-shifted $\exp[ju\tau]$ is obtained between b_u and $b_u^{(o)}$ in the u frequency; for instance, the phase-shift is 30° when phase information τ is set as 5° and frequency u is set as 6.

Fig. 6 shows that the coordinates of the shape boundary point (x_a, y_a) are equal to $(x(n), y(n))$ when the coordinates of the starting point $(x(0), y(0))$ are (x_b, y_b), where n is the number of shift in starting point. When the phase information τ was known in the frequency u, then the number of shift n can be derived from Eq. (5) as

$$n = \frac{N\tau}{360^\circ} \tag{9}$$

The rotation angle θ_d of the shape in the spatial domain is obtained with the phase-shifted information by using the least-squares estimation method and linear interpolation. In the following, we assume the shape boundary coordinates are $P(x(k), y(k)), k = 0, 1, 2, \ldots, N-1$, and the boundary coordinates of rotated shape are $P'(x(k), y(k))$, $k = 0, 1, 2, \ldots, N-1$, k and N indicate the index and the total number of boundary points. Then the transformation between the original and rotated shapes is computed by

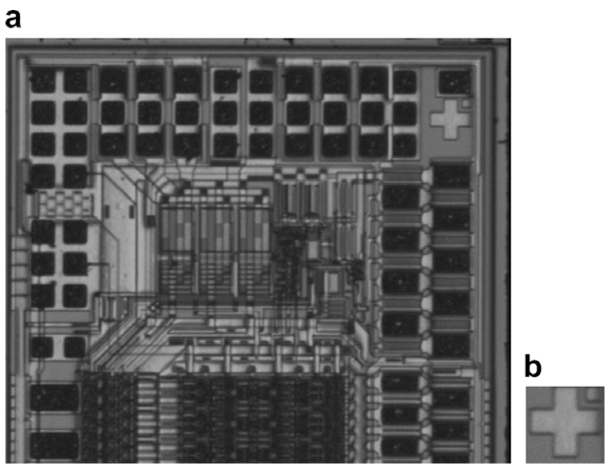


Fig. 7. An LCD driver image with interest object captured by a CCD: (a) LCD1 image, (b) the image of the interest object used as the matching template.

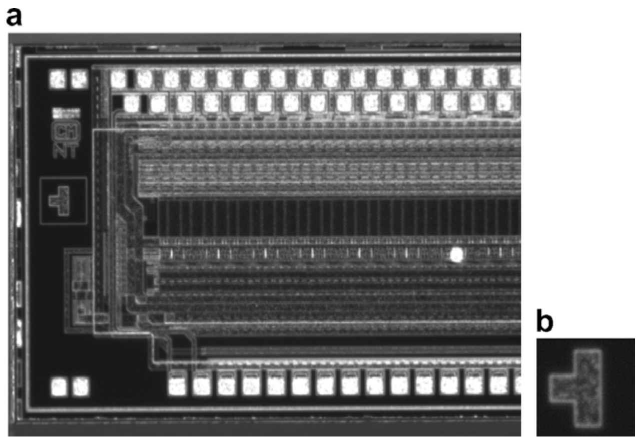


Fig. 8. An LCD driver image with interest object captured by a CCD: (a) LCD2 image, (b) the image of the interest object used as the matching template.

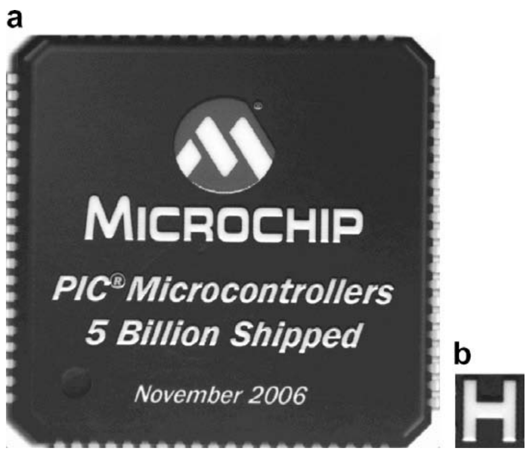


Fig. 9. An IC image with interest object captured by a CCD: (a) IC image, (b) the image of the interest object used as the matching template.

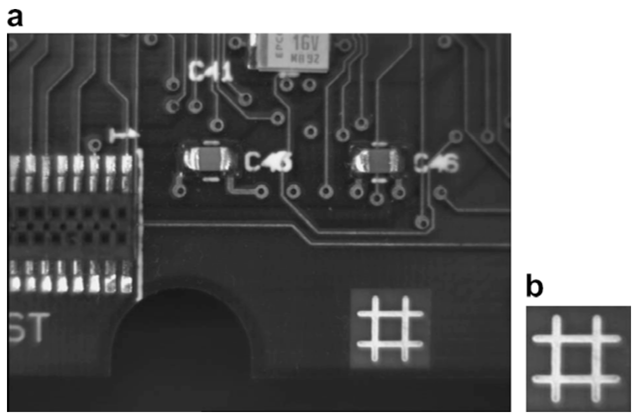


Fig. 10. A PCB image with interest object captured by a CCD: (a) PCB image, (b) the image of the interest object used as the matching template.

$$P''^{\mathrm{T}} = \begin{bmatrix} x''(k) \\ y''(k) \end{bmatrix} = RP^{\mathrm{T}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} = \begin{bmatrix} x(k)\cos\theta - y(k)\sin\theta \\ x(k)\sin\theta + y(k)\cos\theta \end{bmatrix} \tag{10}$$

where R is the matrix of rotation with angle θ, hence, P'' is a set of boundary points rotated from P. Therefore, we can define the error function E as follows.

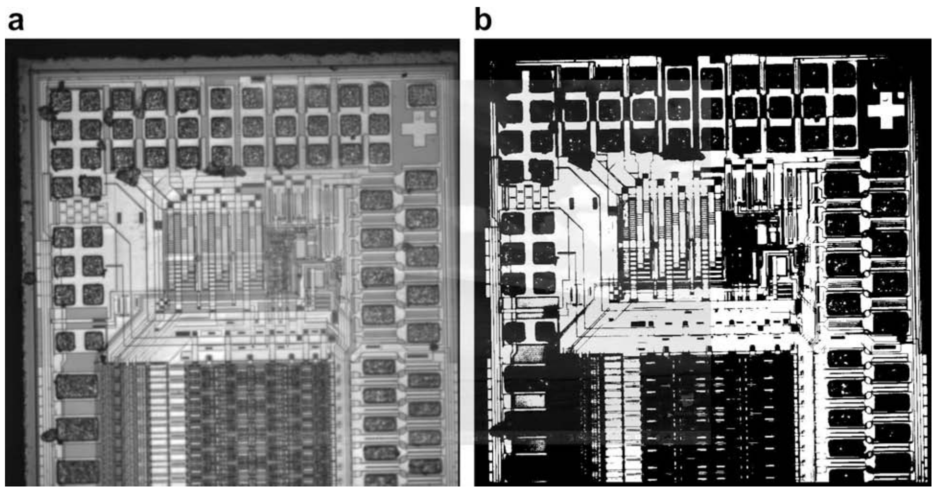


Fig. 11. (a) The LCD driver IC image. (b) The driver IC image after image binarization.

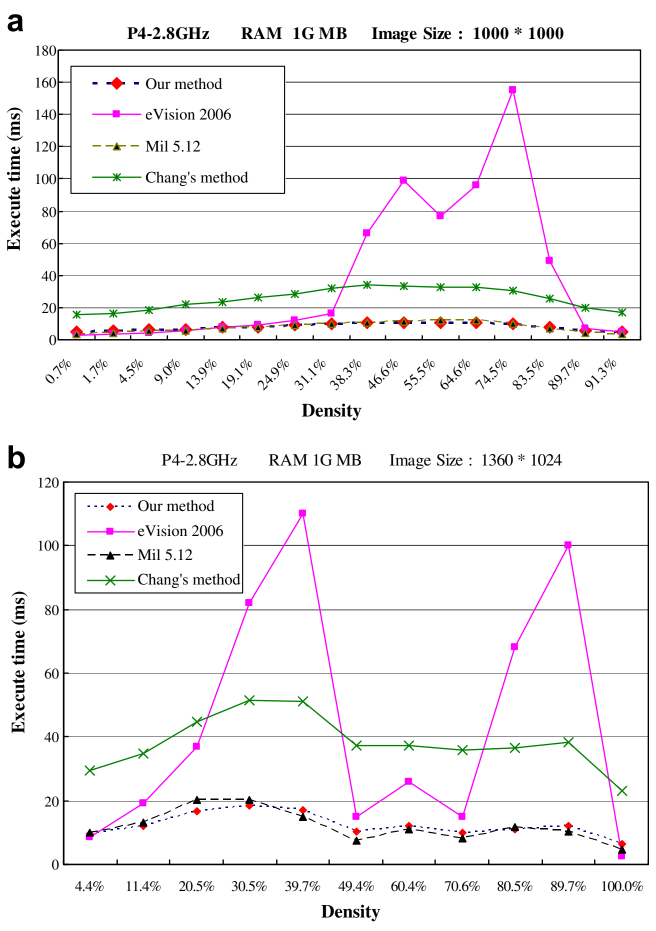


Fig. 12. Performances of the competing algorithms for the (a) 1000 × 1000, (b) 1360 × 1024 driver IC images.

Table 2
The computation times cost by the three parts of the FDBIA algorithm.

Part	LCD1	LCD2
Component detection & tracing	5.149	9.263
FD descriptor transformation	0.399	1.678
Translation & orientation estimation	0.521	0.521
Total processing time (ms)	6.069	11.462

$$E = \sum |P'' - P'|^2$$
$$= \sum_{k=0}^{N-1} (x(k)\cos\theta - y(k)\sin\theta - x'(k+n))^2 \qquad (11\ )$$
$$+ (x(k)\sin\theta + y(k)\cos\theta - y'(k+n))^2$$

The optimal value of θ can be derived by finding the minimum value of E according to the least-square estimation criterion. This implies that $\frac{\partial E}{\partial \theta} = 0$, the equation can be expressed by

$$\frac{\partial E}{\partial \theta} = 0$$
$$\Rightarrow 2\sum_{k=0}^{N-1}\begin{bmatrix}(x(k)\cos\theta - y(k)\sin\theta - x'(k+n))(-x(k)\sin\theta - y(k)\cos\theta)\\ +(x(k)\sin\theta + y(k)\cos\theta - y'(k+n))(x(k)\cos\theta - y(k)\sin\theta)\end{bmatrix} = 0$$
$$\Rightarrow \left[\sin\theta\sum_{k=0}^{N-1}(x(k)x'(k+n) + y(k)y'(k+n))\right] +$$
$$\left[\cos\theta\sum_{k=0}^{N-1}(x'(k+n)y(k) + x(k)y'(k+n))\right] = 0$$
$$(12\ )$$

Let $A = \sum_{k=0}^{N-1}(x(k)x'(k+n) + y(k)y'(k+n))$ and $B = \sum_{k=0}^{N-1}(x(k)x'(k+n)$ $+y(k)y'(k+n))$, Eq. (12) can be rewritten as

$$A\sin\theta + B\cos\theta = 0 \qquad (13)$$

The angle θ is computed as:

$$\theta = \tan^{-1}\left(-\frac{B}{A}\right) \qquad (14)$$

From Eq. (9), the number of shift n was calculated but the value is in general not an integer. Therefore, we utilized the linear interpolation to transfer this non-integer into its integer neighbors. Let n_1 and n_2 be the integral neighbors lying immediately below and over n, respectively. Assuming θ_1 and θ_2 are obtained from Eq. (14) when the number of shift points are equal to n_1 and n_2. The real angle θ_d is estimated as

$$\theta_d = Tr \times \theta_1 + (1 - Tr) \times \theta_2 \qquad (15)$$

where Tr is the difference between n and n_1 .

5. Experimental results

In this section, the performance of the proposed FDBIA algorithm is evaluated. Our connected-component detection algorithm is firstly compared with Chang's method [20] and two commercial software products, *Mil* and *eVision*, to verify that our algorithm is more efficient than general component labeling algorithms because we only need partial connectivity information (the boundary pixels) in the AOI context. Later, several reference images with different translations and rotations will be used to verify the efficiency and accuracy of the proposed FDBIA algorithm. The experimental result manifests that the FDBIA algorithm is suitable for real-time application of AOI. Figs. 7–10 show the original reference images and the corresponding interest objects (inspection marks). The original reference images (referred to as LCD1, LCD2, IC, and PCB) have the sizes being 1024×768, 1280×1024, 859×818, and 768×576, and the corresponding interest objects are of sizes 80×80, 90×90, 80×80, and 120×120, respectively. For each reference image, we simulate 20 inspected images by applying various rotations with angles from 0° to 20° to the original image. All the experiments were performed with Borland C++ Builder 6.0 on a P4 2.8 GHz with 1 GB of memory, and the computation times were recorded.

5.1. Performance of connected-component detection

In this section, the proposed component-detecting algorithm is compared with the other three algorithms including one recently published method (Chang's algorithm [20]) and two commercial software products, *Mil* and *eVision*. In the first experiment, an LCD driver IC image with two different resolutions (1000×1000 and 1360×1024) is used. Its original image and one instance of binarized image are shown in Fig. 11. Various binarization thresholds used in our connected-component detection algorithm would detect different densities of objects in the original image (the density is defined as the ratio of detected objects to the original image in terms of the number of pixels) and definitely affect the needed execution time. The comparative performances between our proposed algorithm and the other algorithms on the image with two resolutions are shown in Fig. 12. No matter which image resolution is used, our algorithm significantly outperforms Chang's algorithm and *eVision* in terms of execution times with various densities. Further, our method is as efficient as *Mil*.

5.2. Computation bottleneck of the FDBIA algorithm

As previously noted, our FDBIA algorithm consists of three parts: component-detecting and tracing, FD descriptor transformation, and translation and orientation estimation. The computation times for the reference images, LCD1 and LCD2, consumed by the three parts of the FDBIA algorithm are listed in Table 2, and it can be seen that the component-detecting and tracing part consumed the most computation time. As a consequence, it is critical to devise an efficient component-detecting and tracing algorithm to obtain a viable alignment method, as is the proposed FDBIA method, for the AOI applications. A thorough comparative computation times between the FDBIA algorithm and other commercial software will be presented in the next section. There are some conditions under which the FDBIA method and most commercial software may not work well. First, a low-contrast target object with large variations in illumination is difficult to recognize, and the target object cannot be fully segmented by the binarization threshold. Second, the alignment process will fail if there is an overlap between the target object and the other objects in the inspected image. This overlap causes the false segmentation of the component-detection process.

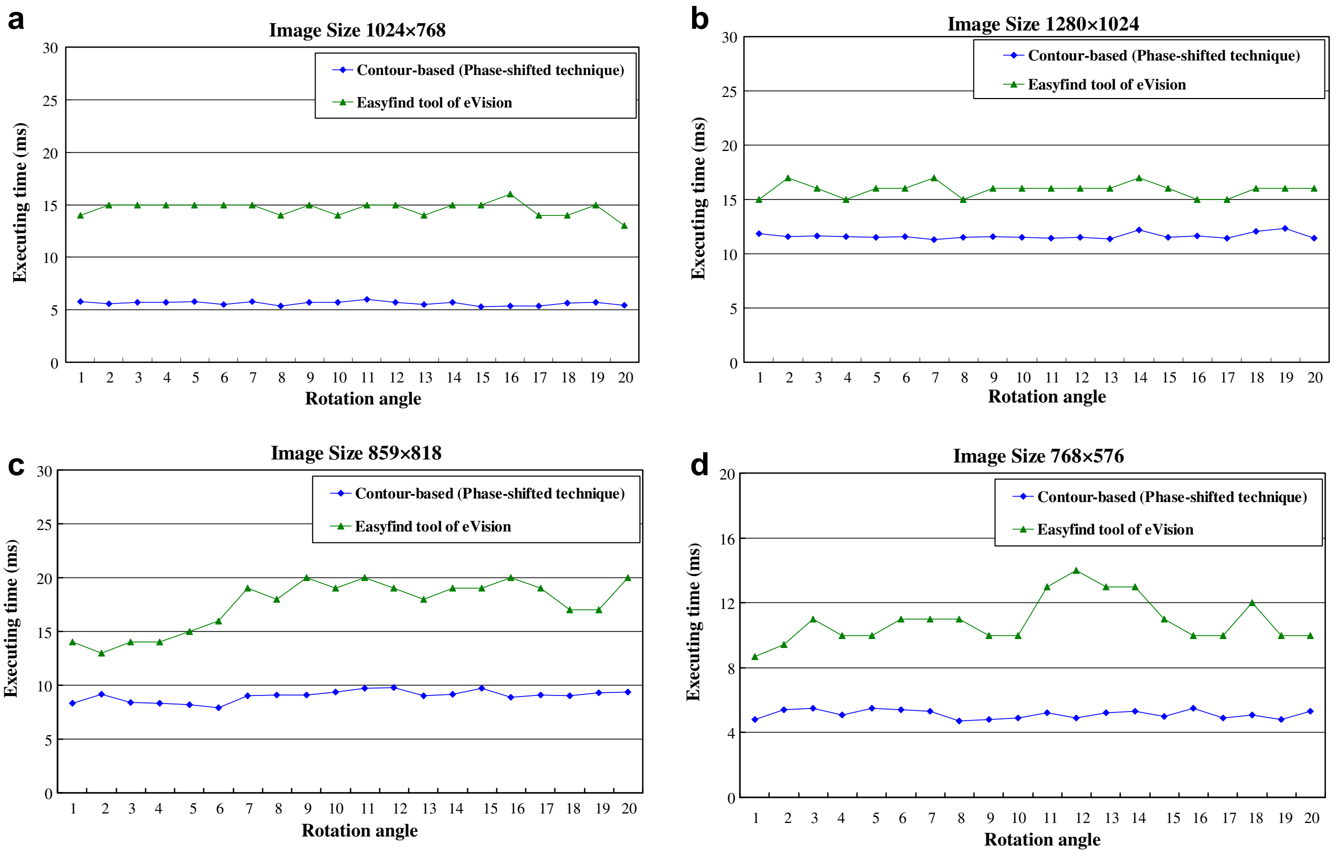


Fig. 13. The alignment computation time for different reference images using the FDBIA algorithm and the *Easyfind*. (a) LCD1, (b) LCD2, (c) IC, and (d) PCB.

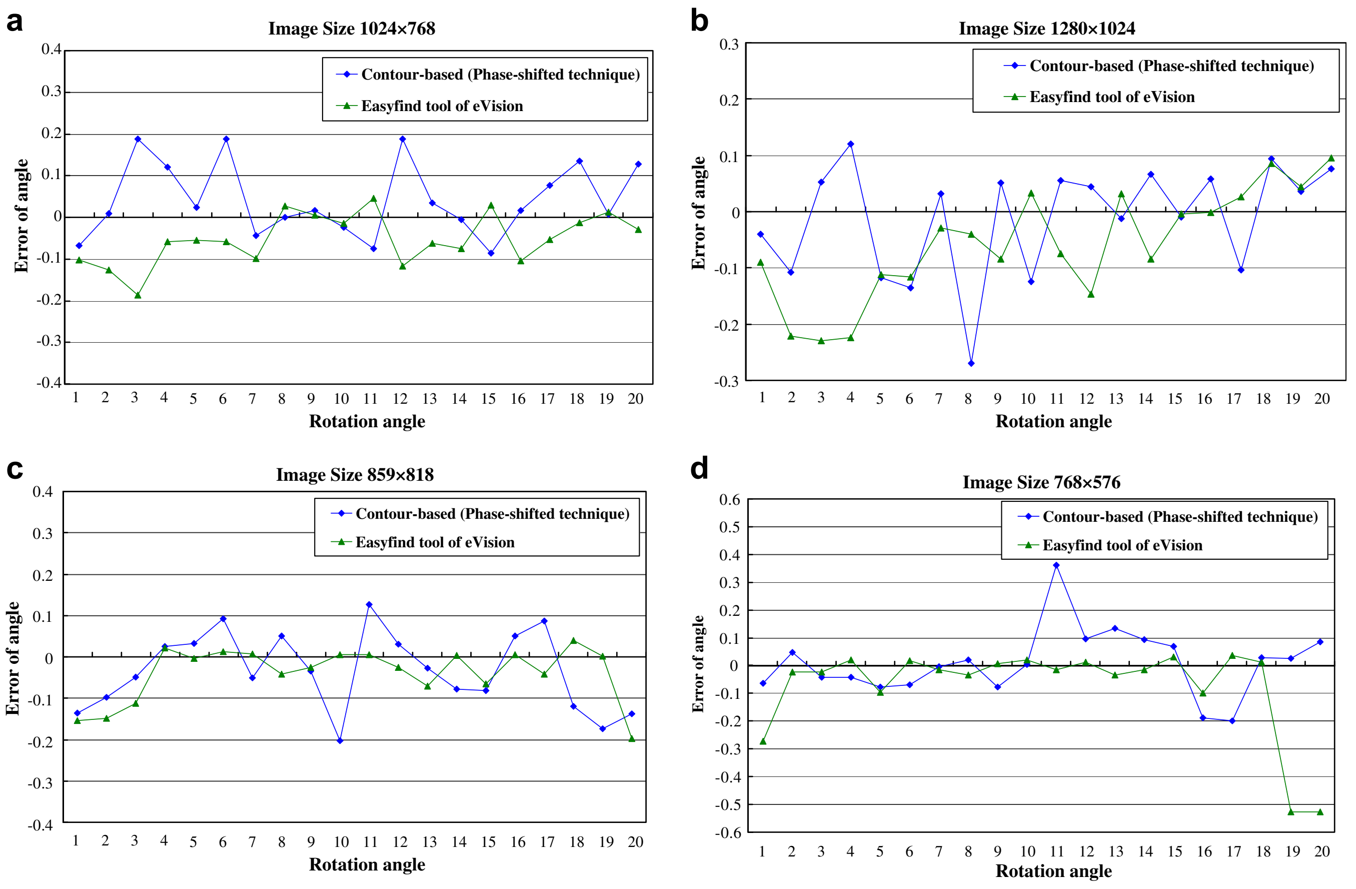


Fig. 14. Comparative alignment accuracy for different reference images. (a) LCD1, (b) LCD2, (c) IC, and (d) PCB.

Table 3
The maximum absolute error and rooted mean square error produced by the FDBIA algorithm and *Easyfind*.

	LCD1		LCD2		IC		PCB	
	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}	E_{max}	E_{rms}
FDBIA	0.189	0.096	0.269	0.098	0.203	0.094	0.361	0.121
Easyfind	0.187	0.078	0.229	0.112	0.198	0.066	0.526	0.168

5.3. Comparative computation time with commercial software

Fig. 13 shows the computation times of two competing alignment algorithms, the proposed FDBIA algorithm and a commercial software product, namely the *Easyfind* tool of *eVision*, applying to the references images LCD1, LCD2, IC, and PCB, respectively. These figures clearly show that our FDBIA algorithm is more computationally efficient than *Easyfind* tool by saving 5–10 ms depending on the degree of rotation variations. We also observe that the variation of computation time consumed by the proposed FDBIA algorithm is more conservative than that produced by *Easyfind*, meaning that our method is more suitable for time-bound applications.

5.4. Comparative accuracy with commercial software

Fig. 14 shows the comparative alignment accuracy of the FDBIA algorithm and the *Easyfind* for the four reference images LCD1, LCD2, IC, and PCB with various rotation angles ranging from 0° to 20°. It is seen that the FDBIA algorithm is comparative with *Easyfind* in terms of alignment accuracy under various rotation transformations, but consuming less computation time as we noted in the previous section.

To provide an overall accuracy evaluation, two performance indices, the maximum absolute error and the rooted mean square error, are defined as follows.

$$E_{max} = \max_i |E_i| \tag{14a}$$

$$E_{rms} = \sqrt{\frac{1}{N_{rot}} \sum_{N_{rot}} (E_i)^2} \tag{14b}$$

where E_i is the alignment error incurred by the transformation with rotation angle I ranging from 0° to 20°, and N_{rot} is the number of tested rotation transformations, here, $N_{rot} = 20$ in our case. As seen in Table 3, for reference images LCD1 and IC, the results indicate that the maximum absolute errors produced by the two competing algorithms are almost the same while the rooted mean square error obtained using *Easyfind* is a bit better than that of our algorithm. On the other hand, for the reference image LCD2, the maximum absolute error produced by *Easyfind* is less than that generated by our algorithm while *Easyfind* is defeated by our algorithm in terms of the rooted mean square error. Finally, the maximum absolute error and the rooted mean square error obtained using our algorithm are both better than that of *Easyfind* for reference image PCB. The experimental results therefore manifest that the FDBIA algorithm is comparable with the *Easyfind* tool with respect to the overall accuracy performance indices, but the former is more computationally efficient than the latter as previously noted. This indicates that the FDBIA algorithm is suited for real-time applications in the AOI context.

6. Conclusions

In this paper, a FDBIA algorithm for AOI applications has been presented. Our method can align reference images with inspected images efficiently and accurately against rotation and translation variations. The FDBIA algorithm elaborates connected component detecting and tracing, Fourier descriptors, and phase-shifted rotation estimation. The proposed FDBIA algorithm consists of three phases: training phase, matching phase, and alignment phase. In the training phase, the values of the Fourier descriptors for the interest object in the reference images are derived. While in the matching phase, an object matching method that uses the normalized minimum distance classifier with the bounding criterion is applied to find the most probable target object in the inspected image that is to be aligned with the interest object in the reference image. Finally, the rotation shift between the reference and the inspected images is estimated by using the novel phase-shifted technique. Experimental results manifest that the proposed FDBIA algorithm sustains the similar alignment accuracy to that of a commercial software *Easyfind*, and the computation time is more economic for the FDBIA algorithm. This indicates that our approach is suitable for accurate image alignment in real-time AOI industrial inspections.

References

[1] B. Zitova, J. Flusser, Image registration methods: a survey, Image Vision Computing 21 (2003) 977–1000.
[2] Y.Y. Tang, H.D. Cheng, C.Y. Suen, Transformation-Ring-Projection algorithm and its VLSI implementation, International Journal of Pattern Recognition and Artificial Intelligence 5 (1–2) (1991) 25–56.
[3] D.M. Tsai, C.H. Chiang, Rotation-invariant pattern matching using wavelet decomposition, Pattern Recognition Letters 23 (2002) 191–201.
[4] M.S. Choi, W.Y. Kim, A novel two stage template matching method for rotation and illumination invariance, Pattern Recognition 35 (2002) 119–129.
[5] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge, Comparing images using the Hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (9) (1993) 850–863.
[6] D.P. Huttenlocher, R.H. Lilien, C.F. Olson, View-based recognition using an eigenspace approximation to the Hausdorff measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (9) (1999) 951–955.
[7] O.K. Kwon, D.G. Sim, R.H. Park, Robust Hausdorff distance matching algorithms using pyramidal structures, Pattern Recognition 34 (10) (2001) 2005–2013.
[8] C.J. Chen, S.H. Lai, S.W. Liu, T. Ku, S.Y.C. Yeh, Optical PCB inspection system based on Hausdorff distance, Machine Vision Applications in Industrial Inspection 5679 (2005) 53–61.
[9] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 2 (60) (2004) 91–110.
[10] K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (10) (2005) 1615–1630.
[11] D. Zhang, G. Lu, Review of shape representation and description techniques, Pattern Recognition 37 (2004) 1–19.
[12] T.D. Haig, Y. Attikiouzel, M.D. Alder, Border following: new definition gives improved borders, IEE Proceedings on Communications Speech and Vision 139 (1992) 206–211.
[13] C.C. Chen, Improved moment invariants for shape discrimination, Pattern Recognition 26 (5) (1993) 683–686.
[14] D. Zhang, G. Lu, A comparative study of curvature scale space and Fourier descriptors, Journal of Visual Communication and Image Representation 14 (1) (2003) 41–60.
[15] D. Zhang, G. Lu, Shape-based image retrieval using generic Fourier descriptor, Signal Processing: Image Communication 17 (2002) 825–848.
[16] I. Kunttu, L. Lepisto, J. Rauhamaa, Fourier-based object description in defect image retrieval, Machine Vision and Applications 17 (2006) 211–218.
[17] I. Kunttu, L. Lepisto, Shape-based retrieval of industrial surface defects using angular radius Fourier descriptor, IET Image Process 1 (2) (2007) 231–236.
[18] Y. Yang, D. Zhang, A novel line scan clustering algorithm for identifying connected components in digital images, Image and Vision Computing 21 (2003) 459–472.
[19] K. Suzuki, I. Horiba, N. Sugie, Linear-time connected-component labeling based on sequential local operations, Computer Vision and Image Understanding 89 (2004) 1–23.
[20] F. Chang, C.J. Chen, C.J. Lu, A linear-time connected-component labeling using contour tracing technique, Computer Vision and Image Understanding 93 (2004) 206–220.
[21] C. Fiorio, J. Gustedt, Two linear time Union-Find strategies for image processing, Theoretical Computer Science 154 (1996) 165–181.
[22] K. Wu, E. Otoo, A. Shoshani, Optimizing connected component labeling algorithms, Medical Imaging 2005: Image Processing 5747 (2005) 1965–1976.
[23] Q. Hu, G. Oian, W.L. Nowinski, Fast connected-component labeling in three-dimensional binary image based on iterative recursion, Computer Vision and Image Understanding 99 (2005) 414–434.
[24] J. Martín-Herrero, Hybrid cluster identification, Journal of Physics A: Mathematical and General 37 (2004) 9377–9386.
[25] J. Martin-Herrero, Hybrid object labeling in digital images, Machine Vision and Applications 18 (2007) 1–15.
[26] Y. Shima, T. Murakami, M. Koga, H. Yashiro, H. Fujisawa, A high-speed algorithm for propagation-type labeling based on block sorting of runs in binary images, in: Proceedings of the 10th International Conference on Pattern Recognition, 1990, pp. 655–658.
[27] L. He, Y. Chao, K. Suzuki, A run-based two-scan labeling algorithm, IEEE Transactions on Image Processing 17 (5) (2008) 749–756.
[28] D. Zhang, G. Lu, Evaluation of MPEG-7 shape descriptors against other shape descriptors, Multimedia Systems 9 (2003) 15–30.