Meas. Sci. Technol. **19** (2008) 095501 (16pp) doi:10.1088/0957-0233/19/9/095501

Automatic inline defect detection for a thin film transistor–liquid crystal display array process using locally linear embedding and support vector data description

Yi-Hung Liu[1], **Yu-Kai Huang**[1] **and Ming-Jiu Lee**[1,2]

[1] Department of Mechanical Engineering, Chung Yuan Christian University, Chungli 32023, Taiwan, Republic of China
[2] Array Integration Engineering Department, Longtan Plant, Chunghwa Picture Tubes Ltd, Tauyuan, Taiwan, Republic of China

E-mail: lyh@cycu.edu.tw, g9473049@cycu.edu.tw and leemji@mail.cptt.com.tw

Received 22 November 2007, in final form 28 May 2008
Published 25 July 2008
Online at stacks.iop.org/MST/19/095501

Abstract
Defect detection plays a critical role in thin film transistor liquid crystal display (TFT-LCD) manufacturing. This paper proposes an inline defect-detection (IDD) system, by which the defects can be automatically detected in a TFT array process. The IDD system is composed of three stages: the image preprocessing, the appearance-based classification and the decision-making stages. In the first stage, the pixels can be segmented from an input image based on the designed pixel segmentation method. The pixels are then sent into the appearance-based classification stage for defect and non-defect classification. Two novel methods are embedded in this stage: the locally linear embedding (LLE) and the support vector data description (SVDD). LLE is able to substantially reduce the dimensions of the input pixels by manifold learning and SVDD is able to effectively discriminate the normal pixels from the defective ones with a hypersphere by one-class classification. After aggregating the classification results, the third stage outputs the final detection result. Experimental results, carried out on real images provided by a LCD manufacturer, show that the IDD system can not only achieve a high defect-detection rate of over 98%, but also accomplish the task of inline defect detection within 4 s for one input image.

Keywords: thin film transistor liquid crystal display, defect detection, one-class classification, support vector data description, manifold learning, locally linear embedding

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Thin film transistor liquid crystal displays (TFT-LCDs) have become more and more popular in recent years. In today's competitive market, LCD manufacturers must provide panels of highest possible quality for returning a favorable market position. To achieve this, one of the most effective and efficient ways is to inspect defects on the panels directly.

TFT-LCD manufacturing consists of three fabrication processes, including TFT array, cell and module assembly processes. Over the past few years, several approaches to LCD defect detection have been proposed, and most of them

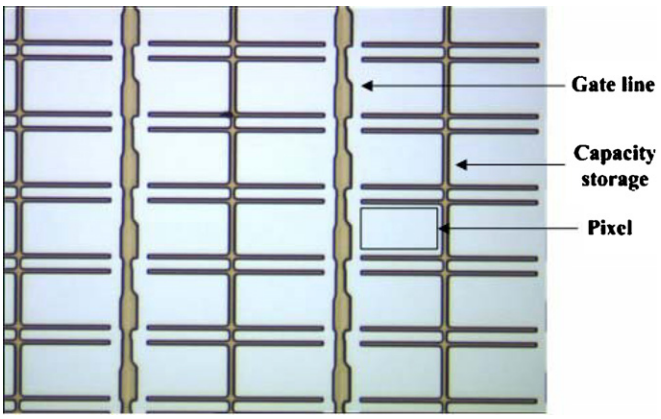


Figure 1. A normal GE image.

focused on the MURA defect which is executed in either a cell process or a module assembly process, for example, the works of [1–9]. Several works have also been proposed for defect detection on glass substrates [10, 11]. As for the defect detection for a TFT array process, Lu *et al* [12] proposed a singular value decomposition-based scheme by which four kinds of surface defects of panels can be detected. However, their scheme is performed after the entire TFT array process is over.

The above works deal with the problems of defect inspection in a cell/assembly process, or after the TFT array process is complete. However, inspecting the defects 'in' the TFT array process is also very important. The TFT array process consists of five successive engineering processes, including (1) gate-electrode, (2) semiconductor-electrode, (3) source-and-drain, (4) contact-hole and (5) pixel-electrode engineering. In each engineering, a panel will go through the same seven processes: cleaning, thin-film deposition, photo-resist coating, exposure, developing, etching and stripping. Due to different physical factors, various kinds of defects would occur during the TFT array process, and such defects are called inline defects. They would damage the panels, thus lowering the yield rate. Fortunately, most of the defective panels can still be fixed by rework if the inline defects are detected before the etching process. Therefore, detecting the inline defects on panels accurately before the entire TFT array process is complete becomes the first problem to be solved. This paper proposes an inline defect-detection (IDD) system which can quickly and accurately tell whether a captured image contains an inline defect.

Based on the idea that the earlier the inline defects are detected, the higher the probability that the defective panels can be fixed, this paper aims at dealing with the problem of defect detection for gate-electrode (GE) engineering. A normal GE image, as shown in figure 1, contains gate lines, capacity storages and pixel regions. The pixel regions are the rectangular regions enclosed by the gate lines and the capacity storages. For the sake of simplicity, the 'pixel regions' are called 'pixels' hereafter.

In GE engineering, the inline defects such as *connection between gate line and capacity storage* (CGC), *abnormal photo resist coating* (APC), *scratch* (SCR) and *particle* (PAR) can generally be observed from the surfaces of the panels. Examples of defective images are shown in figure 2. The images used in this paper are 768 × 576 colored images with resolutions of around 1.15–1.20 pixels μm^{-1} (the 'pixel' mentioned here means the spatial unit of a digital image, not the pixel region in the GE image). All the images are captured between developing and etching processes by high-resolution cameras.

From figure 2, we can observe that the defect in a defective image would appear in some pixels only, called the defective pixels, while other pixels in the same image are normal.

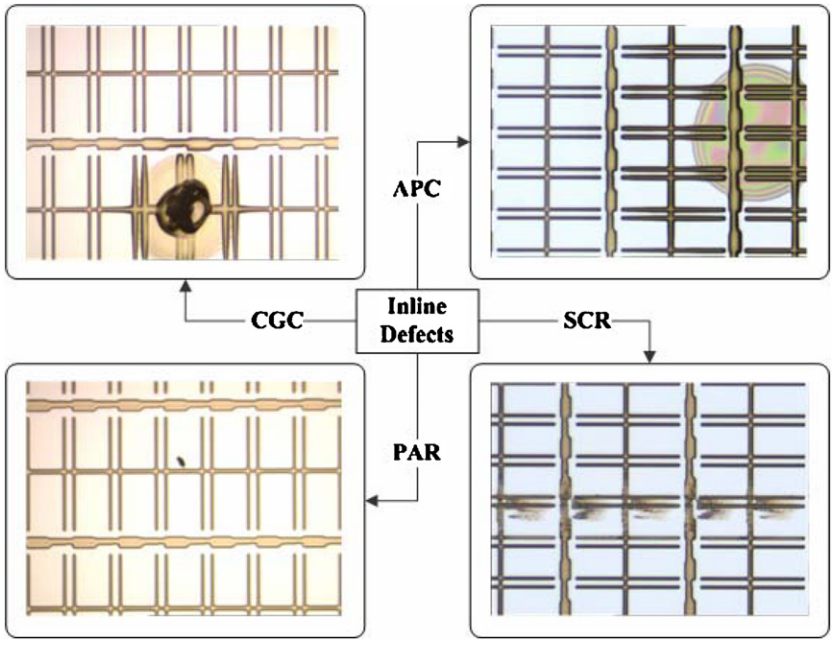


Figure 2. Examples of defective images in GE engineering.

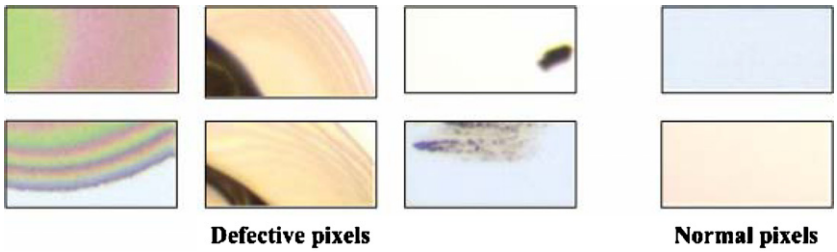


Figure 3. Defective pixels and normal pixels.

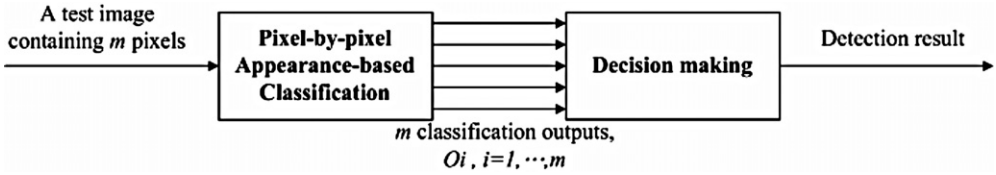


Figure 4. The proposed idea for inline defect detection.

Figure 3 shows some defective and normal pixels. The main difference between a normal pixel and a defective image is that their appearances are apparently different: the surface of a defective pixel contains a specific kind of texture, while the color of a normal pixel is nearly uniform, implying that the variation of the gray-level distribution of a normal pixel is small. Therefore, the appearance of a pixel can be a discriminating feature.

Based on the above observation, we have a straightforward idea, as illustrated in figure 4, that the goal of inline defect detection can be achieved by appearance-based classification. Assuming that an input GE image contains m pixels, the m pixels are sent into the system for appearance-based classification, one at a time. That is, the task of appearance-based classification will be performed m times for the input GE image. For appearance-based classification, the gray-level image of a pixel will be sent into the classifier directly. If the gray-level pixel is a $p \times q$ matrix, the input of the classifier will be a $p \times q$-dimensional vector in which each element is a gray value within the range of $[0, 255]$. The classification output for the ith pixel is denoted by O_i, where O_i will be either 0 or +1. If the ith pixel is classified as the non-defect class, then $O_i = 0$; otherwise (classified as the defect class) $O_i = +1$. The final decision can be made by the rule: if $\sum_{i=1}^{m} O_i > 0$, the input GE image is defective; otherwise $\left(\sum_{i=1}^{m} O_i = 0\right)$ it is a normal GE image.

Though the idea is simple, two critical problems may follow. Firstly, the large-input-dimensionality (LID) problem would occur if the appearance-based classification were adopted [17, 18, 36]. Secondly, though the classification task mentioned above can simply be realized by two-class classification, a large number of *missing defects* may be produced, which refers to the large-positive-margin (LPM) problem. The two critical problems will be detailed in section 2.

In order to solve the LID problem, a powerful nonlinear dimensionality reduction technique called locally linear embedding (LLE) [32, 33] is introduced into our system. LLE is able to map the high-dimensional input data into a single low-dimensional embedding coordinate system by manifold learning. By LLE, the input dimensionality of the classifier can be reduced substantially, therefore reducing the complexity of the classifier.

In the classification stage, we need to select an appropriate classifier as the defect detector in order to return a high defect-detection rate. Instead of using two-class classifiers such as the support vector machine (SVM) [13, 14], this paper proposes the use of a one-class classification to solve the LPM problem. A novel one-class classifier called support vector data description (SVDD) [23, 44] is adopted as the defect detector. By using SVDD, the LPM problem can be solved, thus being able to reduce the number of missing defects.

By using LLE and SVDD, the appearance-based classification stage can be constructed. However, we need to first obtain the pixels from an input GE image before this stage. Therefore, we also construct an image preprocessing stage. Finally, we develop the IDD system by combining the following three: image preprocessing, appearance-based classification and decision making. The overview of the IDD system will be given in section 3. By the IDD system, the inline defects can be detected from the given GE images effectively and efficiently.

The remainder of this paper is organized as follows. In section 2, the LPM and LID problems will be elaborated, as well as their corresponding solutions. The overview of the IDD system is given in section 3. Section 4 details the appearance-based classification including LLE-based dimensionality reduction and SVDD-based defect and non-defect classification. Experimental results are presented in section 5. Conclusions are drawn in section 6.

2. Problem statements and solutions

In this section, we will detail the reasons why we adopted LLE and SVDD as the basis of the appearance-based classification stage in the proposed IDD system.

2.1. Solving the large-input-dimensionality (LID) problem with LLE

As aforementioned, the appearance-based classification approach is adopted in this paper, that is, the gray-level pixel image will be fed into the classifier directly. The input dimensionality, however, would be very large. For example, the input dimension equals 900 if a gray-level pixel is a 30×30 matrix. Often, appearance-based classification systems would suffer from the large-input-dimensionality (LID) problem. The most typical examples are face detection [16] and face recognition [17, 18, 36]. Therefore, it is necessary to perform the task of dimensionality reduction on the input pixels before they are sent into the classifier.

Projection-based methods are widely used for dimensionality reduction. The most popular one would be principal component analysis (PCA) [27]. PCA computes a linear transformation by diagonalizing the covariance matrix of data. However, its performance is generally limited due to its linear nature [28]. Its nonlinear version, kernel PCA (KPCA), has been proposed to overcome this shortcoming [29]. Unfortunately, KPCA is computationally expensive [30]: the kernel matrix needs to be stored after the training, and as a test datum arrives, a large number of dot-product calculations is needed to obtain the projection of the test datum. The self-organizing map (SOM) is an unsupervised neural-network approach to dimensionality reduction [31]. Though SOM possesses the property of a neighborhood-preserving map, it tends to involve more free parameters such as learning rate, neighborhood size and network structure, implying more computational complexities and the problem of local optimum.

Besides the projection-based and neural network approaches, there is also an approach to dimensionality reduction, called manifold learning. Manifold learning aims at discovering low-dimensional manifolds embedded in high-dimensional input space. Some promising manifold-learning methods include locally linear embedding (LLE) [32, 33], isomap [34] and Laplacian eigenmap [35].

LLE provides nonlinearly dimensionality reduction in an unsupervised manner and maps the high-dimensional inputs into a single global coordinate system of lower dimensionality, called embedding. During the LLE mapping, not only can the global data structure be recovered by locally linear fits, but also the intrinsic geometry of local neighborhoods can be preserved by a neighborhood-preserving map. Also, its optimizations do not involve local minima. LLE has been a promising tool for nonlinear dimensionality reduction and has successfully been applied in tasks that require reducing the input dimensionality [36–39]. Therefore, for solving the LID problem in this study, LLE is introduced into our system.

2.2. Solving the large-positive-margin (LPM) problem with SVDD

The defect-detection problem may be treated as a two-class (non-defect and defect) classification problem. Assuming that the two-class classification strategy is adopted, we need to define the two classes: all normal pixels belong to the non-defect class (defined as positive class), and all defective pixels belong to the defect class (defined as negative class). Then, we need to choose a good two-class classifier. SVM can be a candidate due to its great generalization ability [13, 14].

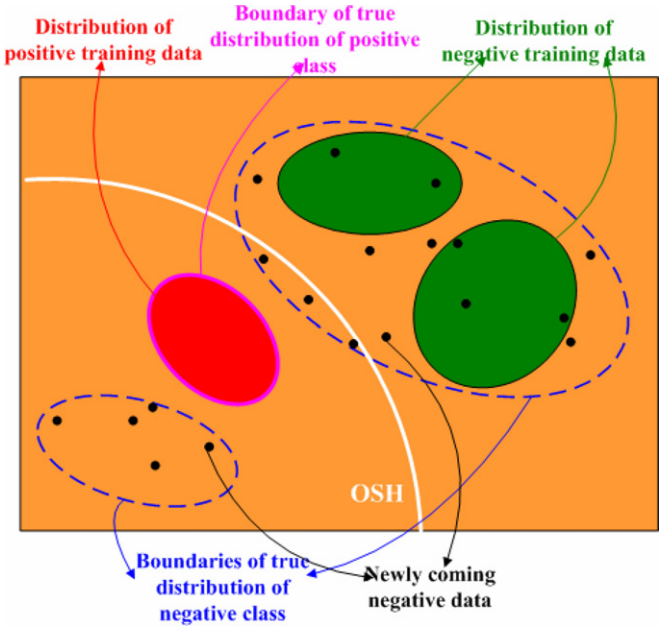


Figure 5. An example illustrating the LPM problem caused by the two-class classifier SVM.

Based on the principle of structural risk minimization (SRM), SVM can learn an optimal separating hyperplane (OSH) that not only minimizes the empirical error, but also maximizes the margin of separation simultaneously during the training, thus gaining better generalization performance than other traditional learning machines learning by empirical risk minimization (ERM) principle, such as the multilayer neural networks trained by the error back-propagation algorithm [14, 19]. The success of SVM has recently been shown in face detection [15, 16], face recognition [17, 18] and biomedical signal classification [19]. However, this great advantage, capable of maximizing the separation margin, would become a drawback for our study instead. We give an account of this point in the following.

For training an SVM, two sets must be prepared in advance: the positive (non-defect) training set and the negative (defect) training set. Figure 5 is an illustrative example, where the positive and the negative training sets are distributed in the red region and the green regions, respectively, and the OSH resulted from SVM is plotted with the white curve.

In a real TFT array process, the patterns of the defective pixels are generally quite different, and some kinds of inline defects do not appear frequently in practical manufacturing, for example the CGC defect. That is, the number of available defective images is actually limited. Therefore, it is difficult to represent the 'true' negative-class distribution by means of limited negative training data. That is, the available negative training set is generally not very representative. In other words, the variation of the true distribution of the negative class would be larger than that of the distribution of the negative training set. Therefore, as we can see from figure 5, the true distribution

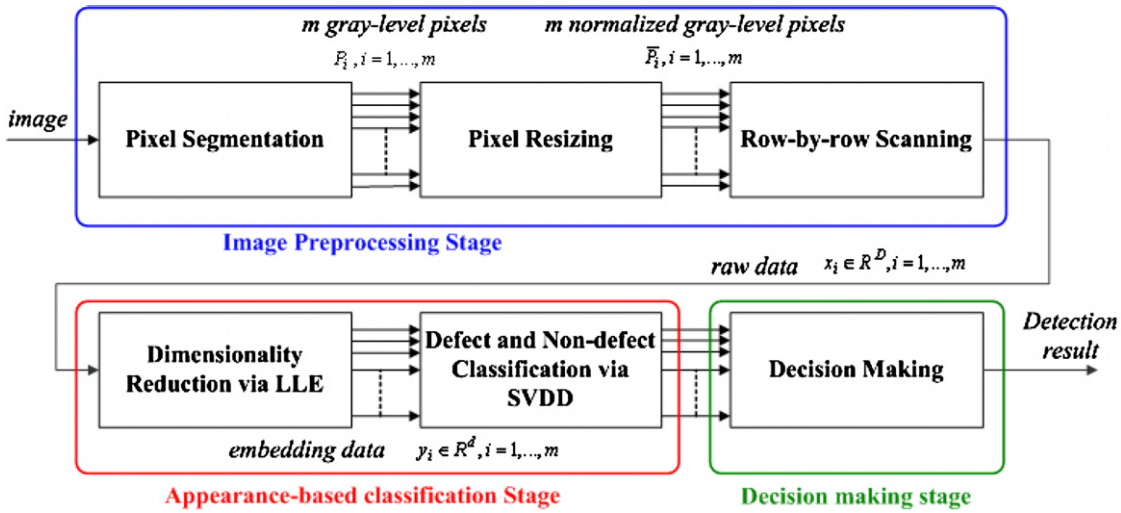


Figure 6. Block diagram of the proposed inline defect-detection system.

of the negative class is the region enclosed by the two blue dashed curves, and the green regions do not stand for the regions enclosed by the blue curves. Consequently, some of the newly coming defective pixels, plotted with black dots, may fall on the wrong side of OSH, since the margin on the side of the positive class is very large, resulting in considerable false positives (missing defects). A false positive means that a negative datum is classified as a positive datum. A defect-detection system is unreliable if there are too many missing defects. The problem stated above refers to the LPM problem. In summary, when the negative training set does not represent the true negative-class distribution, the LPM problem stated above would occur if the two-class classification approach were adopted.

In contrast with defective pixels, the normal pixels have much more uniform appearance though there would still exist a small variation in the positive-class distribution due to the lighting condition. Fortunately, collecting the normal pixels is easier because most of the panels are normal. Therefore, it is reasonable to make the assumption that the available positive training set can approximate the true distribution of the positive class.

Based on the reasonable assumption, the solution to the LPM problem can easily be obtained: if we can find a tight and closed boundary to enclose the positive class, e.g., the pink curve depicted in figure 5, then this boundary can be used as a discriminating function to distinguish defective pixels from normal pixels. Also, only the positive data are needed for training such a boundary. Such an idea meets the central concept of novelty detection, also called *one-class classification* [20–23, 30, 44, 45].

In the past few years, one-class classification has attracted increasingly attention in the fields of pattern recognition and machine learning because some practical two-class classification applications suffer from the problem that one of the two classes is under-sampled. Several promising one-class classifiers have recently been proposed, and one of them is the support vector data description (SVDD) [23, 44].

SVDD is capable of finding a hypersphere to tightly enclose all or most of the positive data (usually referred to as the target data) by solving a constrained optimization problem. After the optimal hypersphere is obtained, its boundary can be used as the discriminating function. If a test datum falls inside or on the boundary, it is accepted as a target datum; otherwise, it is rejected as a negative datum (usually being referred to as an outlier). SVDD has gained wide acceptance in one-class-classification-related applications, such as anomaly detection in hyperspectral imagery [24], bearing fault detection [25] and T-cell epitopes prediction [26].

Target data may not be spherically distributed in the input space. By means of the kernel trick, SVDD can still find a flexible boundary to enclose the target data even if the target class is not spherically distributed in the input space. The success of SVDD should be attributed to the use of the kernel trick which had also been successfully introduced to SVM before SVDD was proposed. Therefore, using SVDD as the classifier in the appearance-based classification stage can deal with the LPM problem, and result in a satisfactory classification performance.

3. Overview of the inline defect-detection system

Figure 6 shows the block diagram of the proposed inline defect detection (IDD) system consisting of three stages: image preprocessing stage, appearance-based classification stage and decision-making stage. In the following, we show how the IDD system processes an image by taking a real defective image as its input.

3.1. Image preprocessing stage

The image preprocessing consists of three components: pixel segmentation, pixel resizing and row-by-row scanning.

3.1.1. Pixel segmentation. For pixel segmentation, a projection-based segmentation method containing six steps is designed. Based on the fact that the gate lines and capacity storages are distributed perpendicularly and periodically, projection can be a good approach to segmentation. The six

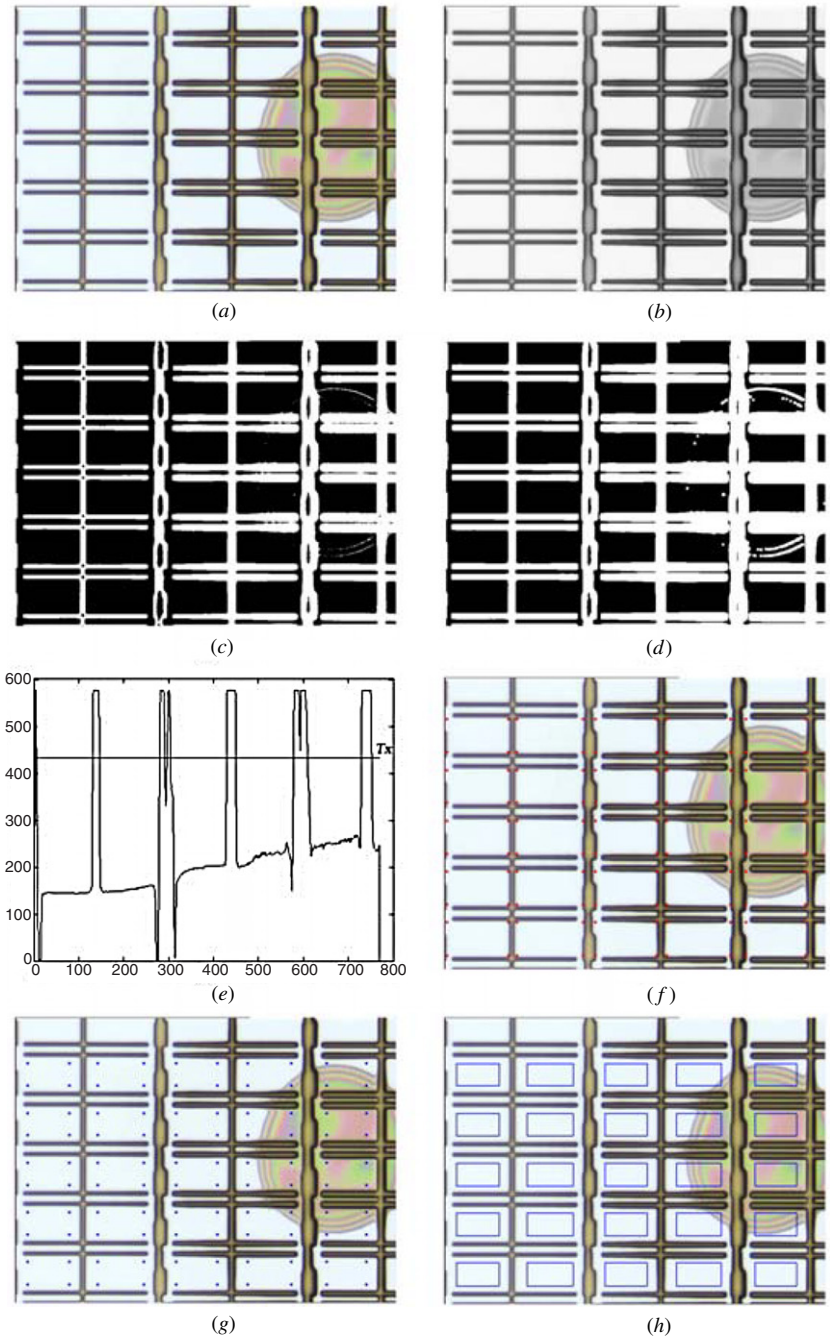


Figure 7. An example of projection-based pixel segmentation: (*a*) original defective image; (*b*) gray-level defective image; (*c*) binarized defective image; (*d*) dilated defective image; (*e*) horizontal projection histogram; (*f*) corner points (red dots) of the pixels; (*g*) modified corner points (blue dots); (*h*) segmented pixels (blue rectangles).

steps are introduced as follows, and the corresponding results are shown in figure 7.

- *Step* 1 (image transformation). First, a colored defective image (figure 7(*a*)) is transformed into a gray-level one $\mathbf{G}[i, j]$, as shown in figure 7(*b*). Then, $\mathbf{G}[i, j]$ is binarized to $\mathbf{B}[i, j]$ by the Otsu thresholding technique [40], as shown in figure 7(*c*).
- *Step* 2 (dilation). The morphological operation dilation with a squared structuring element (width $= 5$) is performed on $\mathbf{B}[i, j]$. The dilated image $\mathbf{D}[i, j]$ is shown in figure 7(*d*).
- *Step* 3 (projection). By projecting $\mathbf{D}[i, j]$ on the *x*- (horizontal) and *y*- (vertical) axes, two projection histograms are obtained. Here we use the horizontal projection histogram as the illustration (figure 7(*e*)).
- *Step* 4 (raster scanning). An *x*-directional raster T_X is used to scan the horizontal projection histogram. The height of T_X is 0.75 times the maximum height of the horizontal projection histogram. The *x*-directional coordinates of the crossing points (interactions of T_X and the histogram) are recorded. The same procedure is also applied to the vertical projection histogram. The coordinates

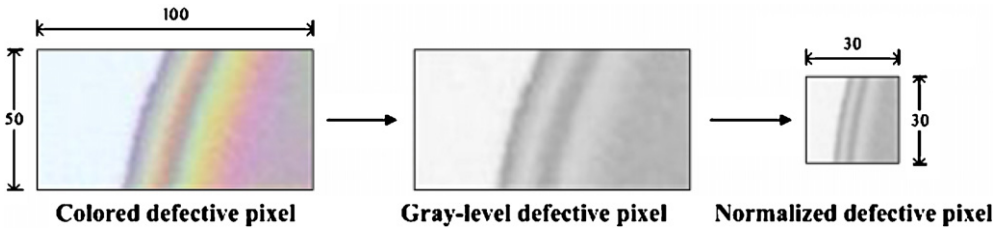


Figure 8. Resizing a defective pixel to a 30 × 30 normalized pixel.

of the crossing points along the *y*-axis can also be obtained.

- *Step* 5 (corner point marking). Each pixel has four corners. Based on the *x*- and *y*-coordinates of the crossing points, the corner points (marked with red dots) of each pixel can be obtained, as shown in figure 7(*f*). By linking the four corner points with straight lines for each pixel, the pixels in the defective image can therefore be found. However, the obtained pixels may cover a bit of gate lines. Some modifications are needed.
- *Step* 6 (corner point modification and segmentation). Therefore**,** the corner points of each pixel are moved inward, until the area of each new pixel surrounded by the moved corner points is 9/16 times the one surrounded by red dots. The result is shown in figure 7(*g*). By linking the newly obtained four corner points with four straight lines, each pixel in the defective image is found and can be segmented, as shown in figure 7(*h*).

From this example, we can observe that there are 25 pixels segmented from the defective image. Then, all the 25 colored pixels are transformed into gray-level ones denoted by P_i, $i = 1, \ldots, m$ where $m = 25$.

3.1.2. Pixel resizing. In this step, the gray-level pixels are resized to $p \times q$ normalized pixels. In this study, $p = q = 30$. An example is shown in figure 8. The normalized gray-level pixels are denoted by $\bar{P}_i$, $i = 1, \ldots, 25$.

3.1.3. Row-by-row scanning. After row-by-row scanning, each normalized gray-level pixel $\bar{P}_i$ can be represented by a vector $x_i \in R^D$ where $D = 900$. The vector x_i is called the raw data. If x_i represents a defective/normal pixel, it is called defect/non-defect raw data. In this example, 25 raw data are obtained. They will be sent to the appearance-based classification for further defect and non-defect classification.

3.2. Appearance-based classification stage

This stage is composed of two components: LLE and SVDD.

3.2.1. Dimensionality reduction via LLE. After the raw data are received, LLE would start to perform the task of nonlinear dimensionality reduction on these data, one at a time. By LLE, the input raw data are mapped into a *d*-dimensional embedding space R^d where $d \ll D$. The mapped data $y_i \in R^d$ are called embedding datum. Then, if x_i is a defect/non-defect raw datum, its corresponding y_i is called a defect/non-defect embedding datum. Then, in this example, the embedding data y_i, $i = 1, \ldots, 25$, are fed into SVDD.

3.2.2. Defect and non-defect classification via SVDD. For each embedding datum y_i, SVDD will generate a classification output O_i. In other words, there will be *m* outputs generated by SVDD. If y_i is classified as the non-defect class by SVDD, then $O = 0$, which means that the *i*th pixel P_i is classified as a normal pixel; if y_i is classified as the defect class, then $O_i = 1$, meaning P_i is a defective pixel. The *m* SVDD outputs will be sent into the decision-making stage for a final check.

3.3. Decision-making stage

Finally, the detection result can be determined by the simple rule: if $\sum_{i=1}^{m} O_i = 0$, the input GE image is normal; otherwise, it is defective. Then, the inline defect detection is complete.

4. Appearance-based classification

4.1. Dimensionality reduction via LLE

4.1.1. Training. Assuming that we have a training set $S = \{x_i\}_{i=1,\ldots,n}$, in which each raw datum is *D*-dimensional, $x_i \in R^D$, the training of LLE [32, 33] can be divided into two steps as follows.

- *Step* 1 (find the optimal reconstruction). In the space R^D, we expect each datum point and its neighbors to lie on or be close to the linear patch of a manifold. Hence, every datum point x_i can be considered as a linear combination of its *k* nearest neighbors. Let w_{ij} denote the *j*th weight of the linear combination for point x_i. The weights w_{ij} summarize the contribution to the reconstruction of the *i*th datum point. In addition, let *W* be the $n \times n$ weight matrix whose *i*th row corresponds to the weight vector $w_i = [w_{i1}, w_{i2}, \ldots, w_{ij}, \ldots, w_{in}]$ for the *i*th datum point x_i, and KNN_i be the set containing the *k* nearest neighbors of x_i. To find the best reconstruction, we need to minimize the reconstruction errors, measured by the cost function:

$$\Omega(W) = \sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|^2 \tag{1}$$

subject to

$$\sum_{j=1}^{n} w_{ij} = 1, \quad \forall i \tag{2}$$

$$w_{ij} = 0, \qquad \forall x_j \notin KNN_i \tag{3}$$

Since x_i is reconstructed only from its k nearest neighbors, the weight matrix W would be very sparse. Further, let $NN(1), NN(2), \ldots, NN(k)$ denote the indices of the k nearest neighbors for a fixed datum point; then each term in Ω can be re-expressed as

$$\begin{aligned}\left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|^2 &= \left\| x_i - \sum_{j=1}^{k} w_{iNN(j)} x_{NN(j)} \right\|^2 \\ &= \left\| \sum_{j=1}^{k} w_{iNN(j)} (x_i - x_{NN(j)}) \right\|^2 \\ &= V_i^T C_i V_i \end{aligned} \tag{4}$$

where $V_i = [w_{iNN(1)}, \ldots, w_{iNN(k)}]^T$ and $C_i = \left[c^i_{jj'}\right]_{j,j'=1,\ldots,k}$ is the local covariance matrix of $k \times k$ for point x_i:

$$c^i_{jj'} = (x_i - x_{NN(j)})^T (x - x_{NN(j')}). \tag{5}$$

Now the primal constrained least-squared problem can easily be solved; to get V_i, we just need to solve the linear system of equations, $C_i V_i = 1_{k\times 1}$, and to rescale the weights so that $\sum_{j=1}^{k} w_{iNN(j)} = 1$. Then, the reconstruction weights for x_i can be obtained. However, the matrix C_i may be singular if $k < D$. It can be solved by adding a sufficiently small value to the diagonal elements (see the appendix of [32]):

$$C_i \leftarrow C_i + \mu I \tag{6}$$

where μ is small compared to the trace of C_i, and I is the $k \times k$ identity matrix. The result would not be influenced if the value of μ were small enough. We set $\mu = 1\mathrm{e}^{-4} \times \mathrm{tr}(C_i)$ where $\mathrm{tr}(C_i)$ means the trace of C_i.

- *Step* 2 (find the optimal embedding). The main idea behind LLE is to embed the input data x_i, which are in a high-dimensional input space R^D, into a lower dimensional embedding space R^d. The embedding data $y_i \in R^d$ represent global internal coordinates on the manifold embedded in R^D. The embedding data in R^d should be reconstructed using the same weights of the linear combination determined in R^D. Therefore, using the same weights w_{ij} obtained in step 1, this step optimizes the coordinates of y_i, which is equivalent to minimizing the embedding cost:

$$\Psi(Y) = \sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2 \tag{7}$$

where the unknown matrix Y contains the output embedding data y_i as its columns. In order to make the resulting embedding data invariant to translation, the constraint $\sum_{i=1}^{n} y_i = 0$ should be added. Also, the resulting y_i should also be forced to have unit variance in all directions in order to avoid degenerate solutions: $(1/n)\sum_{i=1}^{n} y_i y_i^T = I$, where I is the $d \times d$ identity matrix. Therefore, the optimization also becomes a constrained least-squares problem.

To solve this, Roweis *et al* have suggested a more efficient way by introducing the matrix M:

$$M = (I - W)^T (I - W) \tag{8}$$

where M is the $n \times n$ sparse matrix. Then, the optimal embedding can be found by computing the bottom $d + 1$ eigenvectors of the matrix M. The bottom eigenvector associated with the smallest eigenvalue should be discarded since it represents a free translation mode. The remaining d eigenvectors form the embedding Y.

4.1.2. Testing via the linear generalization (LG) method. LLE lacks an explicit transformation for test raw data x_{n+1} since LLE operates in a batch mode. Two methods have been proposed to deal with this problem: one is the linear generalization (LG) method [33] and the other is the incremental LLE [41]. The LG method is adopted in this paper. The corresponding embedding data y_{n+1} for a test raw data can easily be found by LG, which is summarized to two steps as follows.

- *Step* 1. To find y_{n+1}, we need to first find the k' nearest neighbors of x_{n+1} among the training set S, and then compute the weights w_{n+1} that best reconstruct x_{n+1} from its k' neighbors with the sum-to-one constraint, as expressed by equation (2). Note that here the number of nearest neighbors k' can be smaller than that used in the training phase, i.e., $k' \leqslant k$.
- *Step* 2. Finally, the output in the embedding space can easily be found by a linear combination:

$$y_{n+1} = \sum\nolimits_j w_{(n+1)j} y_j \tag{9}$$

where the sum is over the y_i that correspond to the k' nearest neighbors of x_{n+1}.

4.2. Defect and non-defect classification via SVDD

4.2.1. Training. For SVDD, only the target data are needed for its training. In this study, the non-defect class is defined as the target class. Supposing that the training set is $T = \{y_i\}_{i=1}^{L}$, where $y_i \in R^d$ are non-defect training embedding data, and L is the size of the training set, SVDD aims to find a minimum-enclosing hypersphere such that all or most of the non-defect data are accepted, which is formulated as the constrained optimization problem [23]:

$$\begin{aligned} &\text{minimize} \quad R^2 + C \sum_{i=1}^{L} \xi_i \\ &\text{subject to} \quad \|y_i - a\|^2 \leqslant R^2 + \xi_i; \qquad \xi_i \geqslant 0 \quad \forall i \end{aligned} \tag{10}$$

where R is the radius of the hypersphere and a is its center; the slack variables ξ_i allow for some non-defect data outside the hypersphere, and the penalty weight C controls the tradeoff between the volume of the hypersphere and the number of errors. Equation (10) formulates a linear SVDD since it finds the hypersphere in the input space R^d.

However, the linear SVDD may not obtain a tight boundary due to the fact that the training set T may not be a spherically shaped distribution in the input space. A more

flexible boundary is required. Motivated by the success of the kernel trick used in SVM, SVDD has a nonlinear version also: if there exists a nonlinear mapping $\Phi : y \in R^d \mapsto \Phi(y) \in F$, which maps the data y_i into a higher dimensional feature space F (also called the Hilbert space) where the mapped data form a spherically shaped area. The nonlinear SVDD solves the following:

$$\begin{aligned} &\text{minimize } R^2 + C\sum_{i=1}^{L}\xi_i \\ &\text{subject to } \quad \|\Phi(y_i) - a\|^2 \leqslant R^2 + \xi_i; \quad \xi_i \geqslant 0 \quad \forall i. \end{aligned} \tag{11}$$

By introducing the non-negative Lagrange multipliers α_i and β_i, the Lagrangian is obtained:

$$\begin{aligned} Q(R, a, \xi_i, \alpha_i, \beta_i) &= R^2 + C\sum_{i=1}^{L}\xi_i \\ &- \sum_{i=1}^{L}\alpha_i(R^2 + \xi_i - \|\Phi(y_i) - a\|^2) - \sum_{i=1}^{L}\beta_i\xi_i. \end{aligned} \tag{12}$$

Taking the partial differentials of Q with respect to R, a and ξ_i, and setting them to zero yields

$$\frac{\partial Q}{\partial R} = 0 \quad \Rightarrow \quad \sum_{i=1}^{L}\alpha_i = 1 \tag{13}$$

$$\frac{\partial Q}{\partial a} = 0 \quad \Rightarrow \quad a = \sum_{i=1}^{L}\alpha_i\Phi(y_i) \tag{14}$$

$$\frac{\partial Q}{\partial \xi_i} = 0 \quad \Rightarrow \quad \alpha_i = C - \beta_i. \tag{15}$$

Since $\alpha_i, \beta_i \geqslant 0$ and $\alpha_i = C - \beta_i$, the lower and upper bounds for α_i can be obtained: $0 \leqslant \alpha_i \leqslant C$. Substituting (13), (14) and (15) into (12) yields the dual problem:

$$\begin{aligned} &\text{maximize } \quad \sum_{i=1}^{L}\alpha_i(\Phi(y_i)\cdot\Phi(y_i)) \\ &- \sum_{i=1}^{L}\sum_{j=1}^{L}\alpha_i\alpha_j(\Phi(y_i)\cdot\Phi(y_j)) \\ &\text{subject to } \quad \sum_{i=1}^{L}\alpha_i = 1; \qquad 0 \leqslant \alpha_i \leqslant C \quad \forall i. \end{aligned} \tag{16}$$

By the kernel trick, the dot product of any two data points in the feature space can be calculated with Mercer's kernel [14] defined by $K(x, y) = \Phi(x)\cdot\Phi(y)$. Finally, the nonlinear SVDD becomes

$$\begin{aligned} &\text{maximize } \quad \sum_{i=1}^{L}\alpha_i K(y_i, y_i) - \sum_{i=1}^{L}\sum_{j=1}^{L}\alpha_i\alpha_j K(y_i, y_j) \\ &\text{subject to } \quad \sum_{i=1}^{L}\alpha_i = 1; \qquad 0 \leqslant \alpha_i \leqslant C \ \forall i. \end{aligned} \tag{17}$$

The solutions are classified into the three categories: (1) for $\alpha_i = 0$, the corresponding data points fall inside the hypersphere; (2) for $0 < \alpha_i < C$, the data points lie on the boundary of the hypersphere and (3) for $\alpha_i = C$, the data points fall outside the hypersphere. The data points y_i with $\alpha_i > 0$ are called support vectors (SVs).

4.2.2. Testing via the modified decision rule. After the optimal hypersphere is obtained, the output for a test datum y can easily be obtained by the decision rule:

$$y \in \begin{cases} \text{non-defect class} & \text{if } \quad D(y) \leqslant R \\ \text{defect class} & \text{if } \quad D(y) > R \end{cases} \tag{18}$$

where $D(y)$ is the distance between y and the center of the hypersphere, which is calculated by

$$\begin{aligned} D(y) &= \sqrt{\|\Phi(y) - a\|^2} \\ &= \sqrt{\left(\Phi(y) - \sum_{y_i \in SVs}\alpha_i\Phi(y_i)\right)^T\left(\Phi(y) - \sum_{y_i \in SVs}\alpha_i\Phi(y_i)\right)} \\ &= \sqrt{K(y, y) - 2\sum_{y_i \in SVs}\alpha_i K(y_i, y) + \sum_{y_i \in SVs}\sum_{y_j \in SVs}\alpha_i\alpha_j K(y_i, y_j)}. \end{aligned} \tag{19}$$

Only training data points y_i with $\alpha_i > 0$ are needed in the testing phase because (1) from equation (14), we can see that the center a is a linear combination of the non-defect training embedding data whose $\alpha_i > 0$, and (2) the radius R can be obtained by calculating the distance between any training data points that lie on the boundary and at the center, i.e., the data points with $0 < \alpha_i < C$.

Two kernel functions are popular [14]: one is the polynomial function and the other is the Gaussian function. Generally, a polynomial kernel would result in a sparse data distribution in the feature space, which would drop the performance of SVDD [23, 44]. Therefore, the polynomial kernel is not a good choice. In contrast, a Gaussian kernel is more suitable for SVDD because the Gaussian kernel has the property of translation invariance. It has recently been shown that the SVDD with a Gaussian performs better than the one with a polynomial kernel [45]. Therefore, in this paper, the Gaussian function is chosen as the kernel, expressed by

$$K(x, y) = \Phi(x)\cdot\Phi(y) = \exp\left(-\frac{\|x - y\|^2}{s^2}\right) \tag{20}$$

where s is the width of the Gaussian and is a user-defined kernel parameter.

The decision rule expressed by equation (18) is the regular decision rule of SVDD. By using this rule, a satisfactory classification result can be obtained as long as SVDD is well trained. But, it can be predicted that a few test target data would still fall outside the boundary due to the fact that the boundary of the hypersphere is a hard boundary. Yet, it can also be predicted that most of the target data that fall outside the hypersphere would be very close to the boundary because the normal pixels have uniform appearances, implying a small variation. Therefore, we propose an idea that if the resulting boundary can further be enlarged a bit, then those misclassifying target data can be classified correctly. By this idea, we propose a *modified decision rule* given by

$$y \in \begin{cases} \text{non-defect class} & \text{if } \quad D(y) \leqslant R + \Delta \\ \text{defect class} & \text{if } \quad D(y) > R + \Delta \end{cases} \tag{21}$$

where the extension variable Δ is a small non-negative real value, and is user defined.

5. Experimental results

Two experiments will be presented in this section. The first experiment is to test the performance of the appearance-based classification stage. In this experiment, the inputs are the pixels segmented from a set of real GE images. The second experiment is to test the performance of the proposed IDD system, for which the inputs are the GE images.

5.1. Performance test of appearance-based classification

For the three stages of the IDD system, only the second stage, the appearance-based classification stage, needs to be trained in advance. Therefore, this subsection aims to test the performance of the appearance-based classification first. A set of GE images has been prepared for this experiment. This set contains 270 defective GE images, provided by the TFT-LCD manufacturer, Chunghwa Picture Tubes Ltd, Taiwan, in 2005.

5.1.1. Data preparation. First, the projection-based pixel segmentation method is performed in order to obtain the pixels from the 270 defective images. After segmentation, we obtain 4790 pixels from the 270 images. Among the 4790 pixels, 420 are defective and the remaining 4370 are normal. After pixel resizing and row-by-row scanning, 420 defect raw data and 4370 non-defect raw data are obtained from the 420 defective pixels and the 4370 normal pixels, respectively. Each raw datum is a 900-dimensional vector. The appearances of the normal pixels are so similar that we do not need to use all the 4370 non-defect raw data to train LLE and SVDD. Therefore, we randomly choose 420 non-defect raw data from the 4370 ones. Finally, we have two different sets at hand: the non-defect set S_N and the defect set S_D, where $|S_N| = |S_D| = 420$.

5.1.2. Training and testing for LLE. Two parameters need to be adjusted in LLE: the number of neighbors k and the dimension of the embedding space d. Different pairs of (k, d) would generate different dimensionality-reduction results. A comparison example is presented in the following.

We randomly choose 20 non-defect raw data from the non-defect set S_N and 50 defect raw data from the defect set S_D, and use them to train LLE with the two different pairs $(k, d) = (10, 2)$ and $(k, d) = (20, 2)$, respectively. After training, two sets of 150 2D embedding data are obtained, and their distributions in the 2D embedding space are shown in figure 9, where the non-defect and defect embedding data are plotted with '+' and '•', respectively. From figure 9, we can see that compared with the result of $k = 10$, there is a larger class separability between the defect and non-defect classes when $k = 20$. On the other hand, as the value of k is fixed, changing d would also change the result. However, it is difficult to visualize the data distribution as the value of d is larger than 3.

From the above example, it can be seen that to achieve the best dimensionality-reduction result, we need to find the

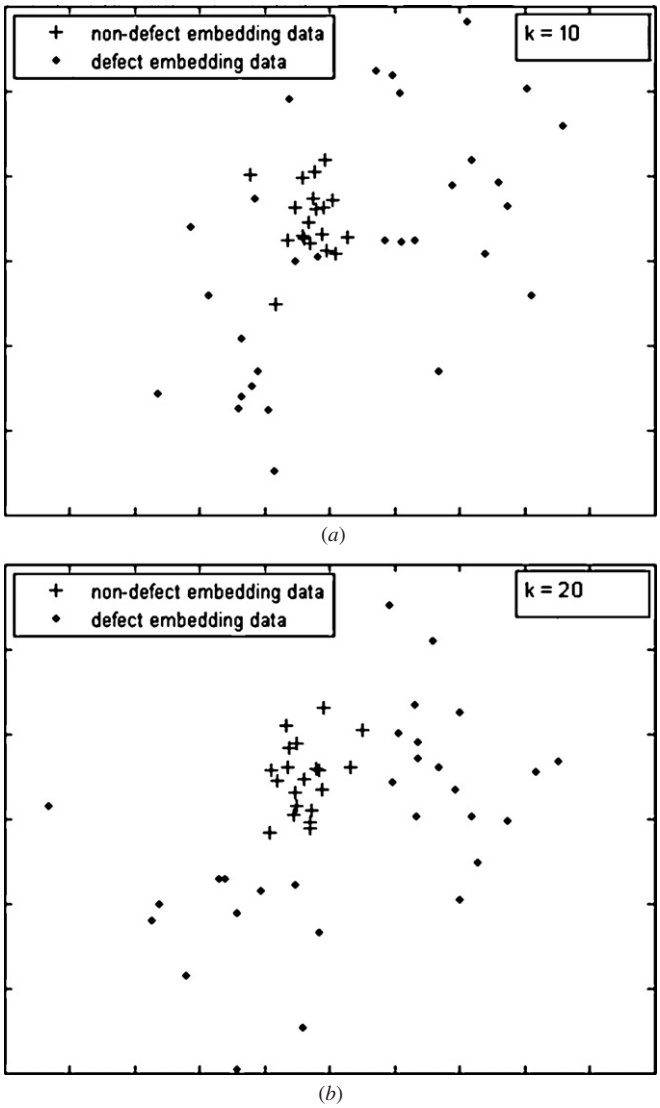


Figure 9. Distributions of defect and non-defect embedding data in the LLE-based 2D embedding space under different numbers of nearest neighbors: (*a*) $k = 10$; (*b*) $k = 20$.

optimal pair of (k, d). Choosing an appropriate and objective approach to optimal parameter selection is important. Two approaches are popular: one is the random sampling method [16–19, 28] and the other is the cross-validation method [23, 38, 43, 45]. The cross-validation method is more suitable for model/parameter selection because its predicted generalization accuracy is more objective, in particular the tenfold cross-validation [42]. Also, when searching for the optimal parameter, the tenfold cross-validation needs to be performed together with the grid searching technique [43] which partitions each parameter's range into several grids. Accordingly, this approach (tenfold cross-validation plus grid searching) is adopted in this paper. For LLE, we estimate the generalized classification accuracy using different pairs of (k, d): $k = [2^1, 2^2, \ldots, 2^8]$ and $d = [2^1, 2^2, \ldots, 2^9]$. Here a nearest neighbor (NN) classifier is used to measure the classification accuracy. To avoid the data tweak problem, we conduct the tenfold cross-validation to estimate the generalized classification accuracy on the sets S_N and S_D. Therefore,

Table 1. Comparisons of the NN-classifier-based classification accuracy and the testing time between with and without the LLE dimensionality reduction.

	Without LLE	With LLE
Error rate (%)	12.57	9.44
Average testing time (s/datum)	0.0610	0.0525

we need to try $8 \times 9 = 72$ combinations, and a tenfold cross-validation will be conducted on each combination. The optimal parameter pair will result in the best cross-validation rate. Finally, the optimal pair of (k, d) is found to be $(2^5, 2^4)$, which means that the optimal dimension of the embedding space is 2^4. That is, the dimensions of all input raw data are reduced substantially, from 900 to 16.

Here we make a comparison between with and without the LLE dimensionality reduction. The condition 'without LLE' means that the 900-dimensional test raw data are directly sent into the NN classifier without LLE mapping. The condition 'with LLE' means that the 900-dimensional test raw data are first mapped into the 16-dimensional embedding space via the LG method, and then these embedding data are sent into the NN classifier for testing the error rate. The results are listed in table 1.

From table 1 we can see that the error rate on the test embedding data (9.44%) is lower than that on the test raw data (12.57%). The testing time for the condition 'with LLE' is composed of two parts: the part of LG and the part of NN-classifier-based classification. We find that most of the testing time for this condition is spent on the LG part where the number of neighbors k' is set as 10. For test raw datum, LG takes around 0.0412 s to find its corresponding embedding data, which occupies 77% ($0.0412/0.0525 \times 100\%$) of the total testing time. The testing time is measured with a MATLAB program. Actually, the real classification time for an embedding datum is less than 0.012 (0.0525–0.0412) s, which is around 1/5 of the classification time of a raw datum (0.0610 s). The above results show that (1) the classification accuracy can be enhanced by using LLE, and (2) the classification time can be reduced after the LLE dimensionality reduction. The validity of the use of LLE for solving the LID problem has been verified.

5.1.3. Comparison between LLE and other methods. In the following, we compare the optimal LLE with two other popular dimensionality-reduction methods; one is the eigen-subspace approach, PCA [16, 27], and the other is the unsupervised neural network, Kohonen's SOM [31]. Note that for a fair comparison, the classification error rates for both PCA and SOM are measured by the nearest neighbor classifier, like the one used in LLE testing.

For PCA, there is only one free parameter, the number of the eigenvectors of data covariance matrix. Since each pixel is represented by a 900-dimensional raw datum, we search for the optimal number of eigenvectors within the range of [1, 891] with the interval of 10 (the searching range is partitioned into 90 grids). The optimal number of eigenvectors means the one resulting in the lowest tenfold cross-validation classification error rate. The optimal number is found to be 111, for which the corresponding error rate is 10.23%.

SOM is able to find a winner on the 2D net to represent the original data which are in high dimension. The learning process of SOM is composed of three steps [46]: (1) initializing the weights of all output neurons; (2) for an input datum, searching for its winner on the 2D net by minimum-distance-based similarity matching; and (3) updating the weights of all neurons. Steps 2 and 3 need to be performed repeatedly until no changes in the weights are observed. In the experiment, the number of input nodes of SOM is 900 because the input raw data are 900-dimensional. The output topological net is arranged as a 40-by-40 2D net. Therefore there are 1600 output neurons in total. During the learning, two SOM parameters need to be determined: one is the learning rate and the other is the topological neighborhood function. The setting of the two SOM parameters here mainly follow the suggestions in [46, 47].

(1) For the learning rate, a time-varying form $\eta(t)$ is adopted here:

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_1}\right), \quad t = 0, 1, 2, \ldots, \tag{22}$$

where t denotes the learning cycle, τ_1 is the time constant and is set as 200 and the initial learning factor η_0 is set as 0.9. The learning rate $\eta(t)$ decreases at an exponential rate. The minimum $\eta(t)$ is set as 0.01.

(2) The topological neighborhood function is also assumed to take a time-varying form as

$$h_{j,y_c}(t) = \exp\left(-\frac{d_{j,y_c}^2}{2\sigma^2(t)}\right) \tag{23}$$

where $h_{j,y_c}(t)$ is the neighborhood function centered around the winner y_c, d_{j,y_c} is the lateral distance between the winner y_c and the excited neuron j in the 2D output space and $\sigma(t)$ is defined as

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_2}\right) \tag{24}$$

where τ_2 is another time constant and σ_0 is the value of σ at the initiation. As the time t increases, the width $\sigma(t)$ decreases at an exponential rate, and the topological neighborhood shrinks in a corresponding manner. The initial neighborhood size should be equal to the number of output neurons, such that all output neurons will have the chance to be the winner of the corresponding input at the beginning. Therefore, the initial size σ_0 is set to be equal to the radius of the 2D output net. The time constant for the neighborhood function is $\tau_2 = 200/\log\sigma_0$.

The SOM network structure as well as the time constants (τ_1 and τ_2) are determined after thorough trial-and-error testing for achieving the best performance, i.e., the lowest MSE (mean-squared error). The SOM's learning stops when the MSE almost keeps constant. The minimum learning cycle is set as 10 000 in the experiment. Finally, the results of optimal PCA and SOM as well as that of the optimal LLE are summarized in table 2. The results of LLE here are the ones reported in table 1.

Table 2. Comparisons of the NN-classifier-based classification accuracy and the testing time between LLE, PCA and SOM.

	LLE	PCA	SOM
Error rate (%)	9.44	10.23	10.97
Average testing time (s/datum)	0.0525	0.0267	0.0311

From table 2 it can be seen that the classification error rates for LLE, PCA and SOM are 9.44%, 10.23% and 10.97%, respectively. Clearly, LLE outperforms the other two in terms of the classification accuracy. However, the classification speed of LLE is the slowest among the three. It is because the testing phase of LLE requires solving the constrained optimization problem again (see the first step of the LG method), which is the shortcoming of the manifold learning method. In contrast, PCA is the fastest among the three. It is not surprising because there is only one step during the PCA transformation: supposing that $x \in R^{900}$ is the input testing raw data, the output data $y \in R^{111}$ are obtained by $y = W^T x$, where W is the 900×111 transformation matrix for which the column vectors are the 111 eigenvectors chosen from the solutions of the data covariance matrix.

In the testing phase of SOM, for an input datum x, we need to find its corresponding output neuron (winner node) on the output 2D net. This procedure is a bit time consuming, because to find the winner we need to calculate the distances between x and all the output neurons first. Then, the winner is the one whose distance to the input datum x is the shortest. In this experiment, there is a total of 1600 output neurons. If the number of output neurons is reduced, the time for searching for the winner can be reduced. However, the SOM classification accuracy is poorer if the number of output neurons is reduced. Nevertheless, the SOM's testing speed (0.0311 s/data) is still faster than LLE's (0.0525 s/data). But this comparison is only valid for the testing phase. The training of SOM is much more time consuming than that of LLE. First, SOM has three parameters to be tuned (τ_1, τ_2 and the size of the output layer) while LLE has only two (k and d). Second, given fixed τ_1, τ_2 and the size of the output layer, the convergence of MSE of SOM network can only be reached after a huge number of learning cycles. In our experiment, it takes at least 20 min. But, given a pair of (k, d), LLE only needs to solve the constrained optimization problem expressed by equations (1)–(3) during the training. It can be accomplished within 10 s in our experiment. Hence, LLE is much faster than SOM in terms of the training speed.

5.1.4. Training and testing for SVDD. After the optimal LLE is obtained, all the non-defect raw data in S_N and the defect raw data in $S_D$are mapped into the 16-dimensional embedding space. Therefore, we obtain two sets: a non-defect embedding set S_{NE} and a defect embedding set S_{DE}, where $|S_{NE}| = |S_{DE}| = 420$. Next, we need to find the optimal SVDD for obtaining the best result of defect and non-defect classification.

There are two free parameters in SVDD: the penalty weight C, and the kernel parameter s. The penalty weight C controls the number of non-defect embedding data that would be rejected by the hypersphere. The larger the value of C the smaller the number of non-defect embedding data rejected. The value of C should be smaller than 1 [44]. In this study, we intend all the non-defect training embedding data to be accepted (within or on the boundary). Therefore, C should be set as large as possible. After taking trial-and-error, we find that as C is set to 0.8, the non-defect training embedding data are accepted.

As for another parameter s, it should be tuned carefully because it would affect the classification performance of SVDD significantly. For a very large s, the solution approximates the original spherically shaped solution. Namely, the nonlinear SVDD becomes the linear SVDD, which can be seen from the Taylor expansion of the Gaussian kernel [23]. We take the 30 non-defect embedding data shown in figure 9(b) to train an SVDD with a very large s ($s = 50$), and the result is shown in figure 10(a). The boundary of the resulting hypersphere is plotted with the white curve. Red dots are the defect embedding data shown in figure 9(b). As we can see, the resulting hypersphere is a rigid one, and is constructed only by two support vectors (the two with white circles). Though all the non-defect embedding data are accepted, one defect embedding datum is also accepted. That is, there is one 'missing defect' when s is set to 50.

In contrast, for very small s the training data would be orthogonal to each other in the space F since

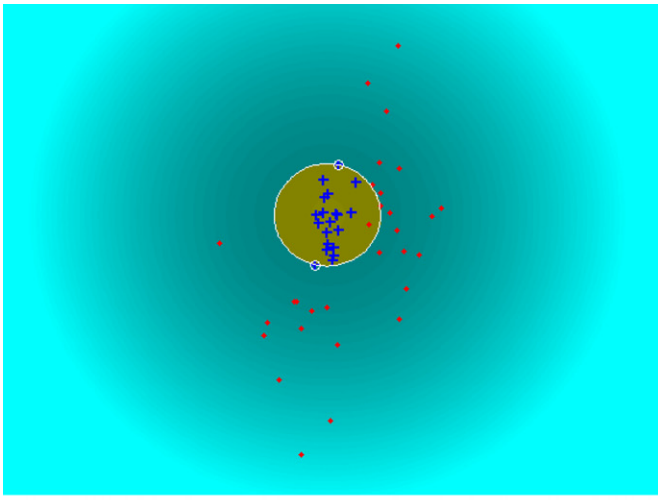
$$\text{as} \quad s \to 0 : K(y_i, y_j) = \Phi(y_i) \cdot \Phi(y_j) = \exp(-\|y_i - y_j\|^2/s^2) \approx 0, \qquad \forall i \neq j. \tag{25}$$

In such an extreme case, all data would become the support vectors. For example, when $s = 1$, the result shows that all the non-defect training embedding data become support vectors, as shown in figure 10(b), and the hypersphere becomes very tight. This result seems to be good because all the defect embedding data are rejected while all the non-defect training embedding data are accepted. However, it is only good for training data; for such a very tight hypersphere, some unseen non-defect embedding data would fall outside the boundary, which increases the errors.

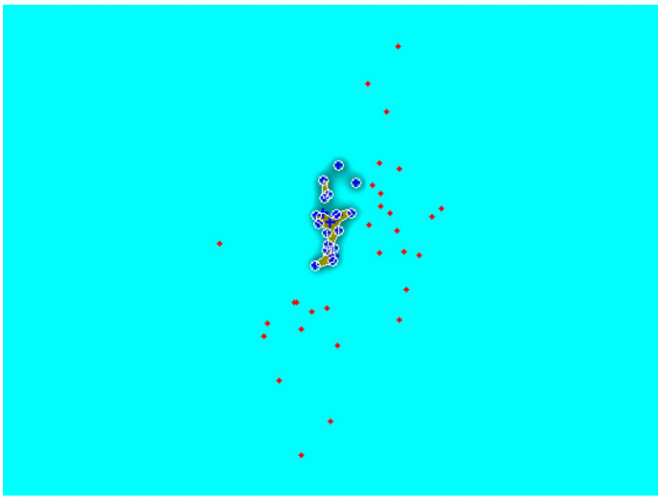
To get a satisfactory classification result, s should be tuned within a moderate range. As we can see from figure 10(c), a flexible boundary that is not too tight or too loose is obtained when $s = 20$. In this case, not only all non-defect embedding data are accepted but also all defect embedding data are rejected.

From the above comparison, it can be seen that the kernel parameter s must be adjusted carefully. Therefore, we search for the optimal value of s ranging from 10 to 40 with an interval of 2, i.e., there are 16 possible values of s in total: $[10, 12, 14, \ldots, 40]$. For each s, the tenfold cross-validation is performed on the two sets: S_{NE} and S_{DE}. Finally the optimal s is found to be 26, and the corresponding result is listed in table 3. In table 3, a false positive/negative indicates that a defect/non-defect test embedding datum is classified as the non-defect/defect class.

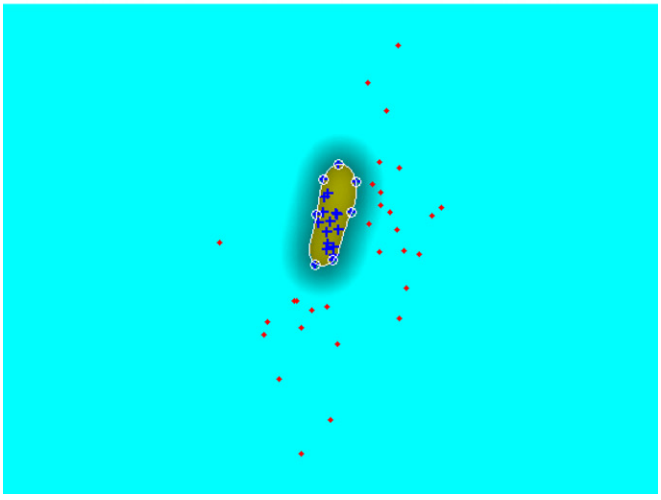
In addition to testing SVDD, we also test the performance of defect and non-defect classification by SVM on the two sets S_{NE} and S_{DE}. Here we briefly review the basics of SVM



(*a*)



(*b*)



(*c*)

Figure 10. SVDD training results under different Gaussian widths: (*a*) $s = 50$; (*b*) $s = 1$; (*c*) $s = 20$.

[17, 19]. Let $T' = \{y_i, z_i\}_{i=1}^{L'}$ be the training set for SVM, where $y_i \in R^d$ are the training data, z_i is its class label being either +1 (non-defect class) or −1 (defect class) and L' is the size of T'. SVM is to find the OSH that maximizes the margin of separation and minimizes the training errors, formulated as

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|w'\|^2 + C'\sum_{i=1}^{L'}\xi_i' \\ &\text{subject to } z_i(w'^T\Phi(y_i) + b') - 1 + \xi_i' \geqslant 0, \quad \forall i \\ &\xi_i' \geqslant 0, \quad \forall i \end{aligned} \tag{26}$$

where w' and b' are the weight and the bias of the hyperplane, respectively, $\Phi(y) : R^d \to F$ is a nonlinear mapping function which maps the data from R^d into a higher dimensional feature space F, ξ_i' are slack variables representing the error measures of data points and the error weight C' is a parameter to be chosen by the user, which measures the size of the penalties assigned to the errors. By introducing the Lagrangian, the primal optimization problem can easily be solved with its dual form, the details of which can be found in [13, 14].

For SVM, the Gaussian function is also adopted as its kernel. So, there are also two free parameters: the error weight C' and the kernel parameter s. The optimal pair of $(C', s) = (100, 0.2)$ is determined after the tenfold cross-validation is performed. The best cross-validation rate of SVM is listed in table 3.

Table 3. Comparison of the classification results between SVDD and SVM (%).

	False-positive rate	False-negative rate	Error rate
SVM	11.92	2.43	7.17
SVDD	2.41	8.76	5.87

From table 3 we can see that compared with SVM, SVDD obtains a lower false positive rate (2.41%). This result shows that SVDD is able to obtain many fewer false positives (missing defects), which indicates that as far as the defect detection is concerned, SVDD outperforms SVM. However, SVDD obtains a higher false-negative rate (8.76%) compared with SVM (2.43%). That is, by SVDD, more non-defect test embedding data are classified as defect ones because the decision rule of SVDD used here is the regular one expressed by equation (18). Therefore, we conduct another experiment in which the decision rule for SVDD is the modified one expressed by equation (21). We discuss the results in the following.

5.1.5. Sensitivity test on an extension variable. The values of the extension variable Δ are set as $0, 0.05R, 0.10R, \ldots, 0.25R$, respectively, where R is the radius of the hypersphere learned by the SVDD with the optimal pair $(C, s) = (0.8, 26)$. The results are listed in table 4.

First, as $\Delta = 0$, the modified decision rule becomes the regular one, and the corresponding results shown in table 4 are the same as those reported in table 3. From table 4, we can see that when $\Delta = 0.05R$ and $\Delta = 0.10R$, the obtained false-negative rates are lower than when $\Delta = 0$. It indicates that properly enlarging the hypersphere can reduce the false negatives effectively. However, the false-positive rate becomes larger as the value of Δ increases further (from $\Delta = 0.15R$ to $\Delta = 0.25R$). This is because the margin between the non-defect class and the enlarged boundary becomes too

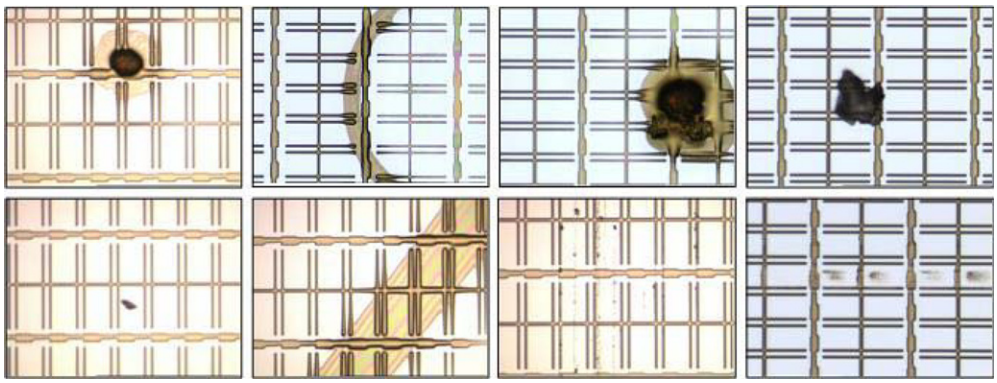


Figure 11. Some examples of defective GE images used for testing the IDD system.

Table 4. Classification results of SVDD with different extension variables (%).

	False-positive rate	False-negative rate	Error rate
$\Delta = 0$	2.41	8.76	5.87
$\Delta = 0.05R$	2.41	7.02	4.65
$\Delta = 0.10R$	2.41	2.35	2.37
$\Delta = 0.15R$	11.84	0	5.97
$\Delta = 0.20R$	13.79	0	7.34
$\Delta = 0.25R$	18.96	0	9.83

large so that some defect test embedding data are accepted. Nevertheless, we can see that the SVDD achieves the best results in both false-positive rate and the false-negative rate when $\Delta = 0.10R$, and the error rate is only 2.37%. The validity of the modified decision rule for reducing the false negatives is demonstrated. Finally, based on the results reported in this subsection (tables 1, 3 and 4), it can be concluded here that in terms of defective-pixel and normal-pixel classification, SVDD is superior to other classifiers including the SVM and NN classifiers.

5.2. Performance test of the IDD system

The last subsection has shown that the appearance-based classification stage is able to achieve a low error rate of 2.37%, for which the inputs are the raw data obtained from the pixels. This subsection further tests the performance of the proposed IDD system for which the inputs are the GE images.

For this experiment, we collected a set of new GE images in 2006, also provided by the same TFT-LCD manufacturer. This set contains 150 images including 50 normal images and 100 defective images. Some defective images are shown in figure 11. The 150 images are fed into the IDD system, one at a time. Each input GE image would go through the three stages (image preprocessing, appearance-based classification and decision making) in IDD systems. The modified decision rule is used for SVDD, and the extension variable Δ is set as $0.1 \times R$. After the test, the results, including the detection rate and the testing time, are obtained and are listed in table 5.

From table 5, we can see that the classification rates for normal images and defective images are 100% and 98%, respectively. The overall detection rate is 98.67%. The effectiveness of the proposed IDD system for the GE inline defect detection has been shown. In addition, for an input GE image, the IDD system would take around 3.5 s to accomplish the task of inline defect detection. The image preprocessing stage takes 2.2323 s to segment the pixels from the input image, while the appearance-based classification stage takes only 1.2365 s to generate the classification outputs for the input pixels. It is believed that the IDD system can be much faster if the algorithms are implemented with C++ program.

In the TFT-LCD manufacture we cooperate with, existing inspection equipment is used to capture the surface images of the panels automatically. The inspection equipment would take around 4 min (240 s) to scan a sheet of a glass substrate containing six panels. During the scanning, 30 images will be captured. The areas to be scanned are randomly determined. After scanning, the acquired images are stored in an image database. The proposed IDD system is separated from the inspection equipment, and the IDD system is able to download the images from the image database, one at a time. Then, the system would start to detect the defect from the image. Since the proposed IDD system is capable of processing an image within 4 s, the time for detecting the inline defects on the 30 scanned images can be less than 120 s, which is much less

Table 5. Detection results and testing time of the IDD system.

	Output	
	Normal	Defective
Input		
50 normal GE images	50	0
100 defective GE images	2	98
Detection rate	$(50 + 98)/150 \times 100\% = 98.67\%$	
Average testing time (s/image)	(stage 1: 2.2323) + (stage 2: 1.2365) + (stage 3: 0.001) = 3.4788	

than that for scanning one sheet of a glass substrate. Therefore, the proposed IDD system is able to fulfill the requirement of real-time inline defect detection for the TFT array process.

6. Conclusions

This paper has presented an inline defect-detection (IDD) system for a TFT-LCD array process. The core of the system is the appearance-based classification stage. To enhance the performance of the whole system, the manifold-learning-based LLE and the one-class-classification-based SVDD are introduced into this stage such that the goals of high-accuracy and high-speed defect detection can be achieved. The effectiveness and the efficiency of the proposed IDD system have been demonstrated by the experimental results carried out on a set of real images captured in the TFT array process. Additionally, the results not only indicate that the one-class classifier SVDD outperforms the popular two-class classifier SVM and the nearest neighbor classifier in terms of defect and non-defect classification, but also show that the classification performance can be improved by further modifying the decision rule of SVDD.

Though the success of the proposed IDD system has been demonstrated in the experiment, there are some requirements and limitations.

(1) The proposed pixel segmentation method highly relies on the projection operations. Therefore, the clearer the images, the better the segmentation result. For best image acquisition, the image plane of the camera must be focused on the surface of the panel. In addition, to successfully segment the pixels from the images, the panels to be scanned must be aligned very well such that the direction of the gate-electrode line in an image can be (nearly) parallel to the horizontal (or vertical) axis of the camera's image plane. For example, in figure 1, the directions of the two gate-electrode lines are nearly parallel to the vertical axis of the image.

(2) Though SVDD is powerful for defect detection, it is not suitable for defect classification because it is essentially a one-class classifier. For defect classification, choosing other adequate classifiers is necessary, for example the SVM.

This work leaves some issues worth studying. Firstly, for the purpose of system diagnosis, it is necessary to further solve the problem of inline defect classification. Secondly, removing the redundant ones from the given training samples (non-defect training pixels) can not only maintain the high classification performance of SVDD, but also speed up the training of a SVDD detector. Therefore, minimum training sample selection is a topic worth studying for SVDD. In addition, due to possible environmental changes such as the change of the process recipe, it is also necessary to develop an online learning mechanism that can facilitate the retraining of the IDD system in real time and increase the adaptability of the system. Finally, the TFT array images used in this work were supported by a LCD manufacturer in Taiwan. In order to evaluate the robustness of the proposed IDD system, we will continue to test the system on the images from other LCD manufacturers in the future.

Acknowledgments

The authors thank the referees for the valuable comments. This work was supported by National Science Council of Taiwan, ROC, under grant NSC-95-2218-E-033-004.

References

[1] Chen L-C and Kuo C-C 2008 Automatic TFT-LCD mura defect inspection using discrete cosine transform-based background filtering and 'just noticeable difference' quantification strategies *Meas. Sci. Technol.* **19** 015507

[2] Zhang Y and Zhang J 2005 A fuzzy neural network approach for quantitative evaluation of mura in TFT-LCD 2005 *Int. Conf. on Neural Networks and Brain* pp 424–7

[3] Ryu J S, Oh J H, Kim J G, Koo T M and Park K H 2004 TFT-LCD panel blob-mura inspection using the correlation of wavelet coefficients *IEEE Region 10th Ann. Int. Conf.* pp 219–22

[4] Lee Y J and Yoo S I 2004 Automatic detection of region-mura defect in TFT-LCD *IEICE Trans. Inf. Syst.* **E87-D** 2371–8

[5] Kim W S, Kwak D M, Song Y C, Choi D H and Park K H 2004 Detection of spot-type defects on liquid crystal display modules *Key Eng. Mater.* **270** 808–13

[6] Oh J H, Kwak D M, Lee K B, Song Y C, Choi D H and Park K H 2004 Line defect detection in TFT-LCD using directional filter bank and adaptive multilevel thresholding *Key Eng. Mater.* **270** 233–8

[7] Song Y C, Choi D H and Park K H 2004 Multiscale detection of defect in thin film transistor liquid crystal display panel *Japan. J. Appl. Phys.* **43** 5465–8

[8] Mori Y, Tanahashi K and Tsuji S 2004 Quantitative evaluation of mura in liquid crystal *Opt. Eng.* **43** 2696–700

[9] Pratt W K, Sawkar S S and O'Reilly K 1998 Automatic blemish detection in liquid crystal flat panel displays *IS&T/SPIE Symp. on Electronic Imaging: Science and Technology*

[10] Tsai D-M and Chao S-M 2005 An anisotropic diffusion-based defect devotion for sputtered surfaces with inhomogeneous textures *Image Vis. Underst.* **23** 325–38

[11] Tsai D-M, Lin P-C and Lu C-J 2006 An independent component analysis-based filter design for defect detection in low-contrast surface images *Pattern Recognit.* **39** 1679–94

[12] Lu C-J and Tsai D-M 2004 Automatic defect inspection for LCDs using singular value decomposition *Int. J. Adv. Manuf. Technol.* **25** 53–61

[13] Vapnik V N 1998 *Statistical Learning Theory* (New York: Springer)

[14] Burges C J C 1998 A tutorial on support vector machines for pattern recognition *Data Min. Knowl. Discovery* **2** 121–67

[15] Osuna E, Freund R and Girosit F 1997 Training support vector machines: an application to face detection *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* pp 130–6

[16] Liu Y-H and Chen Y-T 2006 Face detection using kernel PCA and imbalanced SVM *Advances in Natural Computation (Lecture Notes in Computer Science* vol 4221*)* (Berlin: Springer) pp 351–60

[17] Liu Y-H and Chen Y-T 2007 Face recognition using total margin-based adaptive fuzzy support vector machines *IEEE Trans. Neural Netw.* **18** 178–92

[18] Kim K I, Jung K and Kim H J 2002 Face recognition using kernel principal component analysis *IEEE Signal Process. Lett.* **9** 40–2

[19] Liu Y-H, Huang H-P and Weng C-H 2007 Recognition of electromyographic signals using cascaded kernel learning machine *IEEE/ASME Trans. Mechatronics* **12** 253–64
[20] Markou M and Singh S 2003 Novelty detection: a review: Part I. Statistical approaches *Signal Process.* **83** 2481–97
[21] Markou M and Singh S 2003 Novelty detection: a review: Part II. Neural network based approaches *Signal Process.* **83** 2499–521
[22] Marsland S 2003 Novelty detection in learning systems *Neural Comput. Surv.* **3** 157–95
[23] Tax D and Duin R 2004 Support vector data description *Mach. Learn.* **54** 45–66
[24] Banerjee A, Burlina P and Diehl C 2006 A support vector method for anomaly detection in hyperspectral imagery *IEEE Trans. Geosci. Remote Sens.* **44** 2282–91
[25] Zhang J, Yan Q, Zhang Y and Huang Z 2006 Novel fault class detection based on novelty detection methods *Intelligent Computing in Signal Processing and Pattern Recognition (Lecture Notes in Control and Information Science* vol 345*)* (Berlin: Springer) pp 982–7
[26] Nanni L 2006 Machine learning algorithms for T-cell epitopes prediction *Neurocomputing* **69** 866–8
[27] Jolliffe I T 1989 *Principal Component Analysis* (New York: Springer)
[28] Lu J, Plataniotis K N and Venetsanopoulos A N 2003 Face recognition using kernel direct discriminant analysis algorithms *IEEE Trans. Neural Netw.* **14** 117–26
[29] Schölkopf B, Smola A and Müller K R 1998 Nonlinear component analysis as a kernel eigenvalue problem *Neural Comput.* **10** 1299–319
[30] Hoffmann H 2007 Kernel PCA for novelty detection *Patt. Recognit.* **40** 863–74
[31] Kohonen T 1990 The self-organizing map *Proc. IEEE* **78** 1464–80
[32] Roweis S T and Saul L K 2000 Nonlinear dimensionality reduction by locally linear embedding *Science* **290** 2323–6
[33] Saul L K and Roweis S T 2004 Think globally, fit locally: unsupervised learning of low dimensional manifolds *J. Mach. Learn. Res.* **4** 119–55
[34] Tenenbaum J B, Silva V and Langford J C 2000 A global geometric framework for nonlinear dimensionality reduction *Science* **290** 2319–23
[35] Belkin M and Niyogi P 2003 Laplacian eigenmaps for dimensionality reduction and data representation *Neural Comput.* **15** 1373–96
[36] Hadid A and Pietikinen M 2004 Selecting models from videos for appearance-based face recognition *17th Int. Conf. on Pattern Recognition (ICPR'04)* vol 1 pp 304–8
[37] Varini C, Nattkemper T W, Degenhard A and Wismüller A 2004 Breast MRI data analysis by LLE *IEEE Int. Joint Conf. Neural Netw.* **3** 2449–54
[38] Pang S N, Kim D and Bang S Y 2005 Face membership authentication using SVM classification tree generated by membership-based LLE data partition *IEEE Trans. Neural Netw.* **16** 436–46
[39] Charles M B, Thomas L A and Robert A F 2005 Exploiting manifold geometry in hyperspectral imagery *IEEE Trans. Geosci. Remote Sens.* **43** 441–54
[40] Otsu N 1979 A threshold selection method from gray-level histograms *IEEE Trans. Syst. Man Cybern.* **9** 62–9
[41] Kouropteva O, Okun O and Pietikäinen M 2005 Incremental locally linear embedding *Pattern Recognit.* **38** 1764–7
[42] Witten I H and Frank E 2000 *Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementation* (San Mateo, CA: Morgan Kaufmann)
[43] Hsu C-W and Lin C-J 2002 A comparison of methods for multiclass support vector machines *IEEE Trans. Neural Netw.* **13** 415–25
[44] Tax D and Duin R 1999 Support vector domain description *Pattern Recognit. Lett.* **20** 1191–9
[45] Lee K, Kim D W, Lee K H and Lee D 2007 Density-induced support vector data description *IEEE Trans. Neural Netw.* **18** 284–9
[46] Haykin S 1999 *Neural Networks: A Comprehensive Foundation* (Englewood Cliffs, NJ: Prentice-Hall)
[47] Huang H-P, Liu Y-H, Liu L-W and Wong C-S 2003 EMG classification for prehensile postures using cascaded architecture of neural networks with self-organizing maps *IEEE Int. Conf. Robotics and Automation (ICRA'03)* vol 1 pp 1497–502