An Advanced Machine Vision System for VFD Inspection

Yi Lu[1], Tie-Qi Chen[1], Jie Chen[1] Anthony Tisler[2] and Jianxin Zhang[1]

[1]Department of Electrical and Computer Engineering
The University of Michigan-Dearborn
Dearborn, MI 48128-1491
yilu@umich.edu
Phone: 313-593-5420, fax: 313-593-9967
[2]Jabil Circuit, Inc.
1700 Atlantic Boulevard, Auburn Hills, MI 48326

ABSTRACT

In this paper, we present an advanced machine vision system for VFD inspection. The system includes two procedures—off-line learning and on-line testing. The learning procedure can be either automatic or manual to give user more flexibility to control. A template is generated from the intelligent learning, which includes all features needed by the VFD inspection. The automatic testing procedure loads the template and captures the images of the VFD boards on the manufacturing line, then gives the accurate results quickly. Some machine vision algorithms such as image binarization, image tilt detection, and component detection are described in the paper.

1. INTRODUCTION

Vision inspection has broad applications in industry automation and covers the full range of technical difficulty in machine vision [BKD95, DaJ88, SNW95]. Because of its diverse application environment, it has been recognized that there is no pervasive generic solution in machine vision, each application requires a careful study of alternatives and perhaps even the invention of a new technique [Mun88, DMW94] in order to satisfy the practical requirements.

Vacuum Florescent Displaying(VFD) boards are widely used in the automobile industry, appliances and many other instruments to display a broad range of information. VFD is illuminated by circuit boards specially designed to disclose information such as speed, mileage, fuel level, temperature, various symbols, etc. In general, a VFD board should correctly display a number of different functions. Each function can be specified by a set of ON segments and a set of OFF segments. On the manufacturing line, every VFD board must be tested for its proper functions.

In the current manufacturing, the screen test of VFD boards is visually inspected manually. The fuctional test forces the display to show pre-specified patterns which are to be verified by the test operator. Manual inspection is labor intensive, far from accurate and complete.

This paper describes an intelligent machine vision system for automatic inspection of VFD boards. The system incorporates interactive learning with machine vision algorithms to make the system more powerful and efficient. We will present the system architecture that consists of a learning and test procedure, discuss the major algorithms employed by the system including image binarization, image tilt detection, and component detection.

2. VFD VISION INSPECTION SYSTEM

Some types of VFD faults are described in [LuT96]:

- *Missing segment* - When a voltage is applied to a segment's pin, that segment fails to light.
- *Dim segment* - When an ON segment is dimmer than the other ON segments. It is caused by lack of sufficient florescent material or an external voltage drop.
- *Segment void* - This is a black or dim spot in an ON segment. This is caused by missing internal florescent material to display.
- *Extra segment* - When a test pattern is turned on, an extra segment that does not belong to the test pattern is also turned on.

Besides the capability of inspecting the above faults, our vision inspection system also can

- *inspect the dim ratio of a pattern* - The light intensity of some VFD is adjustable, the dim pattern also need to be inspected.
- *inspect print symbol faults* - Some VFD boards also have print symbols need to be inspected as well as VFD segments. The faults of symbol include scratch and smear.
- *invert the image* - For LCD boards, the ON segments are black and the OFF segments are white. The image needs to be inverted first.

The learning procedure is to learn feature through both interactive learning and automatic algorithms. For every new type of VFD boards, the learning procedure will capture the image of a normal VFD board and produces a symbolic list of inspection features call IRL (Inspection Reference List) to be used in the on-line inspection procedure. On the manufacturing line, the inspection procedure will read in the IRL for the VFD boards in production, capture the image of the VFD board under current inspection, and use the IRL to determine whether the displaying patterns of the VFD board is correct.

We implemented a user-friendly interface which allows the user to interact dynamically such as adjusting the binarization threshold, changing the tilting angle, moving, resizing, merging or splitting the bounding boxes of the small segments.

3. VISION INSPECTION ALGORITHMS

3.1 Binarization

Image binarization is often an important process in many machine vision systems. The accuracy of the binarization thresholds directly impacts the resulting binary image. In this application we need to find the threshold point that correctly separates the gray scale values of the ON segments from the others.

The binarization threshold is found by analyzing the histogram of the image. For a typical image of a VFD board, the number of the pixels belonging to the ON segments is small in comparison with the total pixel number of the image. Therefore in the histogram of a typical VFD image, the highest peak should appears near 0 which represents the background and OFF segments. Let us denote this peak as P_L. The bright pixels belonging to the ON segments result in a peak in the higher intensity area. Let us denote it as P_H. Because the dark pixels usually have different intensities (or pixel values) and the pixels which form the segments at OFF state are usually brighter than the pixels belonging to the background, the distribution near P_L is not uniformed. Let us denote the peaks around P_L as $\{P_{L'}\}$. Similarly, for the bright pixels, some of them are brighter than the others, which result in a number of peaks around P_H. Let us denote them as $\{P_{H'}\}$. As defined above, P_L is the highest peak among $\{P_{L'}\}$ and P_H is the highest peak among $\{P_{H'}\}$. In order to separate the pixels belonging to the foreground from the background, the binarization threshold must locate in a valley between $\{P_{L'}\}$ and $\{P_{H'}\}$. However, the border between $\{P_{L'}\}$ and $\{P_{H'}\}$ are hard to determine. Figure 1 shows two typical histograms of VFD images. Based on these considerations, we developed an algorithm to compute an effective threshold point. The algorithm consists of the following steps:

[Step-1] Generate the histogram $f(X)$.

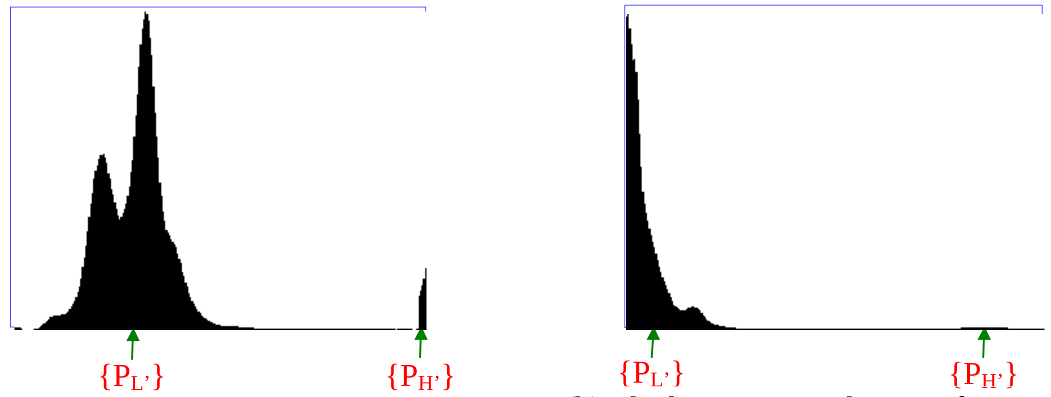
[Step-2] Smooth the histogram using an averaging filter. In most cases, the peaks around P_L and the peaks around P_H will not be removed.

[Step-3] Find the highest peak P_L. Suppose P_L locates at X_L.

[Step-4] Define a function $V_{MIN}(X)$: for any location $X \in (X_L, 255)$, $V_{MIN}(X) = \mathrm{MIN}\{f(x) | x \in (X_L, X]\}$.

[Step-5] Search from X_L to 255, find the location corresponding to the maximum of $[f(X)/V_{MIN}(X)]$. We denote it as X_H. We consider X_H the location of P_H because this peak has the largest peak-to-valley ratio to P_L. By finding the peak according to $[f(X)/V_{MIN}(X)]$, we can skip the peaks around P_L which are usually higher than P_H.

[Step-6] Find the location where the histogram equals to $V_{MIN}(X_H)$. We denote it as X_t. X_t is the binarization threshold we will use. The reason is that it is the location of the deepest valley between P_L and P_H and should be the most suitable border between $\{P_{L'}\}$ and $\{P_{H'}\}$. But why don't we find the deepest valley directly? Because the valleys outside (X_L, X_H) are usually deeper.



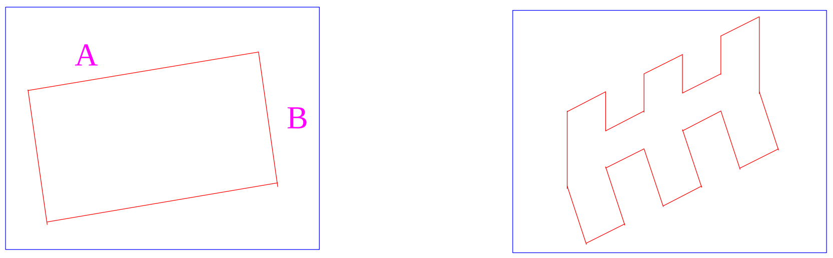
(a) The histogram with several significant peaks$\in\{P_{L'}\}$ existing in the left side of P_L.

(b) The histogram with a significant peak$\in\{P_{L'}\}$ existing between P_L and P_H.

Figure 1. Two types of histograms. Please note that small peaks are invisible in this figure.

3.2 Tilt Detection

VFD boards are not always placed properly therefore the VFD images are tilted. It is important to detect the tilting angle of a test VFD board image so the test features learnt during the learning process can be correctly mapped to this VFD board.



(a) A simple example.

(b) A complicate example.

Figure 2. Some examples of the tilting objects.

The algorithm we developed for detecting the tilting angle first extracts the WHITE pixels belonging to the top edge of the object. For example, the object is a tilted quadrilateral in Figure 2(a). The top edge of this object should be Side A. Often in VFD board images, the WHITE pixels belonging to Side A and the WHITE pixels belonging to Side B are not easy to separate, since

- the object edges are not sharp enough in the image due to noise;
- the angle between Side A and Side B can be an obtuse angle close to 180°;
- Side B may be longer than Side A or Side A may be broken, etc.

A more complicated example is shown in Figure 2(b) in which the top edge is not a straight line. Figure 3 shows a typical image of a VFD board, in which the top edge of the object has unconnected short line segments.

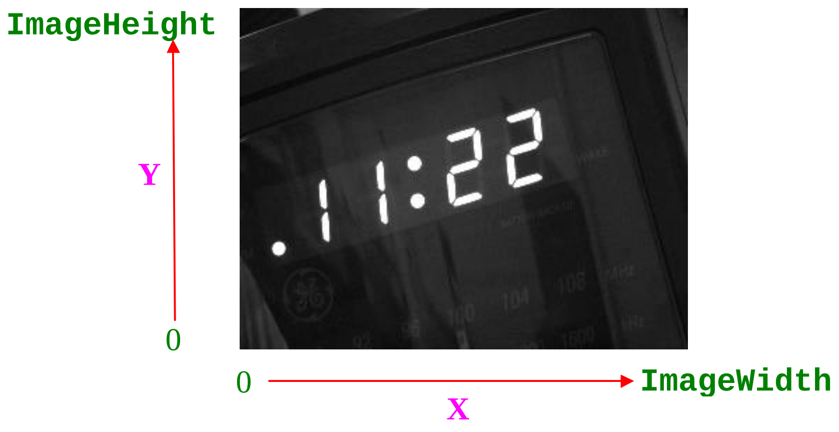


Figure 3. A typical image of a tilted VFD board.

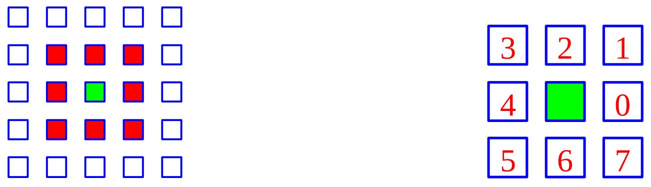
Here we introduce an effective algorithm for finding the tilting angle of a VFD image. Let us imagine a rigid rod falling down from the top of the image. Initially the rod is hanging horizontally at the top of the image. No matter what a tilting angle it may have later, the length of the projection on the **X**-axis is equal to the image width, i.e. **`ImageWidth`**. Assume we let the left end ($X = 0$) fall down first. The rod will stop falling when it hits the first WHITE pixel. Suppose the location of this pixel is ($X_0=0$, Y_0) and the rod is currently perpendicular to the **X**-axis. Let (X_0, Y_0) be the fulcrum and the rod rotate clockwise around this fulcrum until the rod hits another WHITE pixel. Suppose the location of this pixel is (X_1, Y_1). If $X_1 <$ **`ImageWidth`**-1-X_1, the mass center of the rod is at the right side of X_1. In this situation the rod is unstable, we take (X_1, Y_1) as the new fulcrum and continue to rotate the rod until we find the next WHITE pixel. This step repeats until we find an WHITE pixel (X, Y) such that $X \geq$ **`ImageWidth`**-1-X, namely, the mass center of the rod is between X_0 and X. In this situation the rod is stable and will not rotate further, and at this moment the rod and the image have the same tilting angle. This algorithm is fast and accurate and works on a broad range of tilting angle.

3.3 Component Detection

The location of a segment is specified by its bounding box. Once the bounding boxes of all the segments are obtained, we can compute a number of useful features including the occupancy ratio, the average pixel value, the standard deviation and the dim ratio. These features will be used later on to detect the VFD board defects. The bounding boxes of all the segments obtained during the learning program are then mapped to test images to detect defects.

One approach to find bounding boxes of segments is using connected component technique. Finding connected components in an image is a classic problem which has been intensively studied in the community of image processing and computer vision [HaS92]. Most algorithms are time consuming. Here we present a much more efficient algorithm to compute the connected components. This algorithm uses the technique of tracing the border of a complex polygon, therefore, not all the WHITE pixels are scanned and labeled belonging to a specific class, but the border of each class is found therefore the bounding box can be determined. In this algorithm we assume 8-connected ***adjacency*** (see Figure 4(a)), so a ±45°-tilted line will not be considered as separate pixels, and furthermore, while tracing the border of an arbitrary polygon, less border pixels need to be covered. The tracing directions for each pixel is defined in Figure 4(b).

This algorithm does not use any auxiliary storage. The second definition is ***tracing direction***. Each pixel has 8 neighboring pixels and each neighboring pixel represents a tracing direction. The 8 tracing directions which are used in our algorithm are defined in Figure 4(b). As mentioned above, some segments may be formed by connecting several components with thin lines, which means somewhere the width of a segment may be only 1 pixel. In this situation, some pixels must be scanned more than once while tracing the border. Therefore the tracing direction as well as the starting pixel of the border tracing must be used to determine when the border tracing should stop.



(a) 8-connected. (b) Definition of ***tracing direction***.

Figure 4. Border tracing.

Our algorithm first scan the binary image from top to bottom to look for a border pixel. A border pixel is a WHITE pixel which has BLACK pixels in its neighborhood. Suppose the current scanned pixel $\mathbf{P_s}$ is (x_s, y_s). If $\mathbf{P_s}$ is a border pixel outside all the bounding boxes of the segments already found, then we denote $\mathbf{P_s}$ as $\mathbf{P_0}$, let $x_0 = x_s$ and $y_0 = y_s$, and choose $\mathbf{P_0}$ as the starting pixel of a new border tracing. We scan the neighboring pixels of $\mathbf{P_0}$ counterclockwise until we find a BLACK pixel following by another border pixel $\mathbf{P_{-1}}$, extract the tracing direction d_0 from $\mathbf{P_{-1}}$ to $\mathbf{P_0}$ according to Figure 4(b). We initialize the new bounding box by letting $x_{min} = x_{max} = x_0$ and $y_{min} = y_{max} = y_0$, store the current border pixel $\mathbf{P_c}$ i.e. (x_c, y_c) and the current tracing direction d_c by letting $x_c = x_0$, $y_c = y_0$ and $d_c = d_0$. Label the segment as $\mathbf{P_c}$. The next border pixel (x_n, y_n) and the next tracing direction d_n is searched as follows: Let $d_{-c} = [(d_c+4)$ **MOD** $8]$ which is the opposite direction of d_c (see Figure 4(b)), scan the neighboring pixels of $\mathbf{P_c}$ clockwise from $[(d_{-c}+1)$ **MOD** $8]$ until we find another border pixel $\mathbf{P_n}$ i.e. (x_n, y_n), extract the tracing direction d_n from $\mathbf{P_c}$ to $\mathbf{P_n}$ according to Figure 4(b). If $x_n = x_0$ and $y_n = y_0$ and $d_n = d_0$, then accelerate the scan by moving the searching point to the right side of the current bounding box, which can be done simply by setting $x_s = x_{max}$. Otherwise, we update the new bounding box by setting $x_{min} = x_n$ if $x_n < x_{min}$; $x_{max} = x_n$ if $x_n > x_{max}$; $y_{min} = y_n$ if $y_n < y_{min}$; and $y_{max} = y_n$ if $y_n > y_{max}$; store the current border pixel $\mathbf{P_c}$ i.e. (x_c, y_c) and the current tracing direction d_c by setting $x_c = x_n$, $y_c = y_n$ and $d_c = d_n$. These steps of operation repeat until we have traced all border pixels. If we also want to label the pixels inside the borders. Scan the binary image from top to bottom. Assign each unlabeled WHITE pixel the label of its labeled neighboring pixel. Because all the border pixels have been labeled, scanning the image one time is enough.

This algorithm guarantees to keep the interior of the segment at the right side of the tracer during the border tracing. According to the principle of topology, the border of any complicated polygon can be traced using this method as long as the starting pixel of the border tracing is at the outer border of the polygon (if the polygon also has inner borders). In our algorithm, the starting pixel of the border tracing is found by sequentially scanning the binary image, and therefore the first border pixel found is always at the outer border.

This algorithm is efficient in the sense that it only searches for new border pixels, therefore all the interior pixels and all the pixels inside the bounding boxes of the segments already found are skipped. After a new bounding box is found, the scan is accelerated by moving the scanning pointer to the right side of the bounding box. In many cases, this algorithm doesn't make even one entire pass through the image. The computational complexity of our algorithm is proportional to the length of the chain-code representation of the binary image.

4. CONCLUSION

In this paper we presented a VFD vision inspection system and three main algorithms. The system has been designed and implemented for reliable inspection of various types of defects of VFD boards. The system has two procedures, the learning and testing procedures. The three algorithms presented in this paper provide operations critical to the system. We implemented the algorithms above under Window95/NT. For an image of 640×480, the whole process takes about 1 second on a Pentium PC, and the results are very good.

The VFD inspection system has been deployed in three different manufacturing companies worldwide and has been proved to be robust to a plant environment.

REFERENCES

[BKD95] Jerry Bowskill, Tim Katz and John Downie, "Solder Inspection using an Object-Oriented Approach to Machine Vision," SPIE Proceedings, Machine Vision Applications in Industrial inspection III, pp. 34 - 45, 1995.

[DaJ88] A. M. Darwish and A. K. Jain, "Machine Vision Techniques for Inspection of Printed Wiring Boards and Thick Circuit," Journal of Opt. Society, Am., A3, pp. 1465-1482, 1986.

[DMW94] A. D. Dorundo, J. R. Mandeville, and F. Y. Wu, "Reference-Based Automatic Visual Inspection of Electronic Packaging using a Parallel Image Processing System," SPIE Machine Vision Applications in Industrial Inspection II, pp. 38 - 57, February, 1994.

[HaS92] R. M. Haralick and L. G. Shapiro, "Computer and Robot Vision," Addison-Wesley Publishing Company, 1992.

[LuT96] Yi Lu and Anthony Tisler, "Machine Vision Inspection of VF Boards," 13th International Conference on Pattern Recognition, Vienna, Austria, Vol. C, pp. 839-843, August, 1996.

[Mun88] Joseph L. Mundy, "Industrial Machine Vision — Is It practical?" Machine Vision — Algorithms, Architectures, and Systems, Edited by Herbert Freeman, Academic Press, Inc., 1988.

[SNW95] C. Sanby and L. Norton-Wayne, "Machine Vision Inspection of Lace using a Neural Network," SPIE Machine Vision Applications in Industrial Inspection III, Vol. 2423, pp. 315-322, 1995.