COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 44, 279–295 (1988)

Multilevel Thresholding Using Edge Matching

LOIS HERTZ AND RONALD W. SCHAFER

School of Electrical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332

Received August 18, 1987; accepted June 22, 1988

Thresholding is an effective method for simplifying images of solid objects on a solid background. Multiple thresholds are needed when thresholding images containing several objects of different brightness or reflectivity. Also, to account for variations of grey levels due to nonuniform illumination, it is often necessary to allow the thresholds to vary across the image. We describe a technique called *edge matching*, which provides a direct method of adjusting multiple thresholds so that the edges of the thresholded image closely match the edges of the original grey-tone image. The result is a threshold image that preserves the shape and geometrical structure of the objects in the image. © 1988 Academic Press, Inc.

1. INTRODUCTION

In many image analysis problems, it is necessary to simplify an image while retaining information about its shapes and geometric structures. Reducing the number of grey levels needed to represent each pixel will greatly simplify the image since each pixel may originally take on 256 or more possible values. Thresholding is a simple approach to this problem which involves selecting a value, T, such that for each pixel, (i, j), in the original image with grey value, G_{ij},

$$P_{ij} = \begin{cases} 0 & \text{if } G_{ij} \le T \\ 1 & \text{otherwise,} \end{cases}$$

where P_{ij} is the grey level of the output pixel. The resulting image, P, is a binary image where each pixel may take on only one of the two possible values 0 or 1.

The threshold, T, is usually chosen from the histogram of the original image (referred to as the "grey level" image). Consider an image of single, solid object in a solid background. A solid object is one in which the grey values of its pixels are approximately the same. That is, the distribution of pixels representing the object has a small variance around the mean, μ_o. Similarly, the distribution of pixels representing the background has a small variance around its mean, μ_b. The histogram of the image is then the superposition of these two distributions. For this case, most threshold selection techniques assume that μ_o and μ_b are far enough apart that these two distributions remain distinct and that the histogram is bimodal. The threshold is then chosen as a value between the peaks of these two distributions.

If the image can be divided into more than two distinct solid regions, multiple thresholds may be needed. For example, if there are N objects with distinct grey levels, a coarsely quantized image may be created, defined by

$$P_{ij} = \begin{cases} N & \text{if } G_{ij} > T_N \\ N-1 & \text{if } T_{N-1} < G_{ij} \leq T_N \\ \vdots & \vdots \\ 1 & \text{if } T_1 < G_{ij} \leq T_2 \\ 0 & \text{if } G_{ij} \leq T_1. \end{cases}$$

Using these thresholds, a desired region (object) may be "sliced" out by setting to zero all pixels that are not in the desired threshold band. The thresholds may be chosen directly from the histogram by a method analogous to the single threshold case [1]. Also, Wang and Haralick [2] suggested a method of multiple threshold selection based on the histogram of the edge image.

For many images, a global threshold or set of thresholds chosen by any of these methods will yield unsatisfactory results due to noise in the image or gradual variations in grey level either in the objects themselves or as a result of non-uniform lighting. To minimize these effects in the single threshold case, Chow and Kaneko [3] replaced the global threshold by a spatially varying threshold. This was accomplished by dividing the image into overlapping blocks, determining which blocks had bimodal histograms, and selecting a threshold for each such block. Thresholds for the remaining blocks were chosen by interpolating between previously assigned values. Finally, each pixel was assigned a distinct threshold by interpolating between blocks. Nakagawa and Rosenfeld [4] experimented with this technique and extrapolated it to the trimodal case. The overriding assumption with selecting spatially varying thresholds is that the histogram (of the entire image or an individual block) will be sufficient to obtain correct thresholds. However, often this is not true. In fact, especially in the multiple threshold case, the information gleaned from these histograms may be misleading, contradictory, or even incorrect.

The work to be presented here combines the histogram approach with edge information. The technique, which we call *edge matching*, may be used in the case of a single threshold but is most beneficial with multiple thresholds where it helps circumvent some of the previously mentioned problems. The next section will explain the motivation behind this method. Section 3 examines the algorithm in detail while the implementation details and examples are presented in Section 4.

2. EDGE MATCHING

As previously stated, the goal of thresholding is to obtain a simplified image which retains shape information and geometric structure. One measure of this information may be found in the edge image, a binary image which indicates the locations of all edges. If the shape and geometric information have been preserved in the process of thresholding, the edge image should remain virtually unchanged. Therefore, we may say that a thresholded image is a *good* representation of the original image if their corresponding edge images are approximately coincident. These ideas have been used before in thresholding (and general segmentation) methods. Examples include measures such as percentage coincidence of edge pixels, conformity of edge shape, and contrast of edge pixels [5, 6]. However, we have taken a different approach in an attempt not only to isolate those edge points that do not match, but also to use these values to determine better thresholds.

An image pixel is said to be an edge point if it is located on a boundary. That is, while travelling in a given direction, an edge point is the last point encountered which may be considered to be part of a particular object. This edge information may be determined through a variety of techniques which rely on such quantities as simple differences between adjacent pixels, statistics of regions, and properties of neighboring pixels. Summaries of many of these methods may be found in [7–11].

The *edge image* is defined as:

$$E_{ij} = \begin{cases} 1 & \text{if } (i, j) \in \text{set of edge points} \\ 0 & \text{otherwise.} \end{cases}$$

We define edge image subtraction as

$$SE_{ij} = OE_{ij} - TE_{ij},$$

where OE and TE represent the edge images of the original and thresholded images, respectively, and SE refers to the subtracted edge image. Since an edge image may contain only the values $[0, 1]$, the subtracted edge image consists of the values $[-1, 0, 1]$. For each pixel in the subtracted edge image,

$SE_{ij} = 0$ indicates one of two possibilities:

1. pixel (i, j) does not belong to the edge set in either the original or the thresholded image;
2. pixel (i, j) belongs to the edge set in both images.

$SE_{ij} = 1$ indicates that pixel (i, j) is an edge pixel in the original image only. Such pixels will be referred to as *excess original* edge pixels.

$SE_{ij} = -1$ indicates that pixel (i, j) is an edge pixel in the thresholded image only. Such pixels will be called *excess thresholded* edge pixels.

As previously discussed, spatially varying thresholds may be used to counteract the effects of gradual change in grey level. If the image is divided into overlapping blocks, analyzing the subtracted edge image yields information about the thresholds in each block. This information is summarized below:

1. The number and values of the thresholds in an image block are correct if the corresponding block in the subtracted edge image contains only a small number of both excess original and excess thresholded edge pixels.
2. A large number of both excess original and excess thresholded edge pixels in a block in the subtracted edge image indicates that one of the threshold values is such that a valid edge appears in the thresholded image but this edge is not in the correct place. That threshold should be adjusted within this block to force the edge in thresholded image to lie as close to the original edge as possible.
3. If a block contains a large number of excess thresholded edge pixels with a negligible number of excess original edge pixels, a false edge has been introduced within that block. The thresholds should be changed to remove this edge.
4. If a block contains a large number of excess original edge pixels, but a small number of excess thresholded edge pixels, a threshold has probably been omitted in

that block. Suppose that an image has N thresholds and the thresholded block has pixels that fall into k distinct grey levels. If $k \neq N$, the thresholds should be changed so that pixels in the thresholded block fall into $k + 1$ levels. However, if $k = N$, there are not enough grey levels available to represent all grey levels. For example, in the case of a single threshold, a large number of blocks made up of both white and black pixels but containing a large number of excess original edge pixels would suggest that at least two thresholds are needed to successfully segment this image.

The edge matching method to be described below first thresholds the image with a set of global thresholds and then uses the corresponding edge information to adjust thresholds in each block. This has several advantages over histogram methods. First, it provides a means for immediate comparison to the original image and it allows for iteration. Second, it circumvents a problem encountered in the extrapolation of Chow and Kaneko's variable thresholding algorithm. In the binary case, their procedure determines whether the histogram of a given block is unimodal or bimodal. However, in the multiple threshold case, the solution is not as clear. For example, if the global histogram suggests two thresholds, but the histogram of a block is only bimodal—to which of the two global thresholds does this single value correspond? Nakagawa and Rosenfeld [4] suggested a specific test for this case based on the proximity of the new threshold to each global threshold. However, there are instances where this test is not reliable and, for more than two thresholds, there is no general rule that may be applied. In the edge matching algorithm discussed here, adjusting existing values rather than searching for new values avoids this dilemma.

The next section explains the details of the algorithm. In this discussion, it is important to keep in mind the significance of the excess edge pixels in each block as described above.

3. EDGE MATCHING ALGORITHM

The edge matching procedure is outlined below, followed by an explanation of each computational step.

1. Determine a set of global thresholds. Using these values, threshold the image. Then determine the edge images of both the original and the thresholded images and subtract them to obtain the subtracted edge image.

2. Divide the image into overlapping regions. Starting with the global set of thresholds assigned to each block, use the edge pixel analysis presented in the previous section to adjust the thresholds in each block. The following operations (a)–(d) are performed sequentially. After each step, the image is thresholded with the new threshold set, the edges are determined and a new subtracted image is computed.

(a) Adjust the thresholds to remove false edges.
(b) Move one or more thresholds up or down to insert edges where none appear.
(c) Adjust the new threshold values to make edges coincide.
(d) Repeat step (a) in case false edges were introduced in step (c).

If the information about the thresholds within a block is inconclusive, do not adjust any value. This may indicate that the global thresholds are good values in the block or that information about neighboring blocks may be necessary to adjust the thresholds for this block.

3. If new thresholds for a particular block were not set in the previous step, interpolate between neighboring blocks to determine these values.

4. Threshold the image.

5. Fill in small holes and clean up noise using morphological operations [12, 13].

The details of these procedures are explained more fully below.

Determination of global thresholds. The easiest way to determine a set of global thresholds is to search the grey level histogram. The underlying assumption is that this histogram is a superposition of distinct distributions, each centered at a particular grey value, corresponding to the objects in the image. Therefore, in simple cases, the number of desired thresholds, N, may be determined by counting the number of peaks in the histogram. However, for more than two thresholds, these underlying distributions tend to overlap. If this occurs, it may be difficult to determine the appropriate number of thresholds. Therefore, it may be desirable to provide the needed number of thresholds as input. Using this value, the histogram is then iteratively searched until the desired number of thresholds is obtained.

After these global values have been determined, the image is thresholded following the procedure described in Section 1. Using an edge detector, the edge images of both the original and thresholded images are computed. There is no restriction on the edge operator used and many methods have been proposed [7–11]. However, it is desirable that the method chosen yield thin edges; that is, the majority of represented edges should be no more than two to three pixels wide. Finally, following the method outlined in Section 2, the two edge images are subtracted yielding the subtracted edge image.

Adjustment of thresholds. The image is divided into overlapping blocks. For each block, the following computations are performed.

(a) *Removal of a false edge.* As previously mentioned, a false edge may be signaled by a large number of excess thresholded edge pixels. This condition indicates that one or more thresholds is either too high or too low. For a particular block, a small number of original edge pixels usually indicates that the block is solid (i.e., all pixels in the corresponding thresholded block should be of the same grey level.). To force it to be solid, let

$\{B_{\mathrm{ne}}\}$ = the set of all neighboring blocks which are not only solid in the original image but also in the thresholded image (i.e., it contains few excess thresholded edge pixels). These are the *non-edge* blocks.

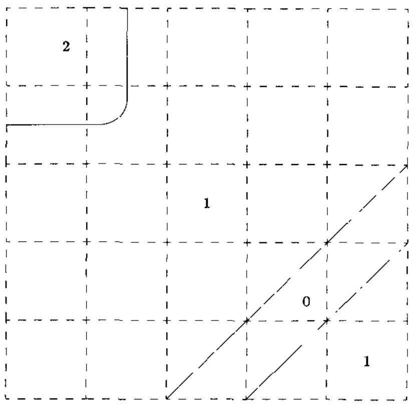
$\{B_{\mathrm{fe}}\}$ = the current block plus all neighboring blocks which also have a false edge. Call this the set of *false edge* blocks.

M = threshold band in which the majority of blocks $\in \{B_{\mathrm{ne}}\}$ fall.

```
For each block $\in \{B_{\mathrm{fe}}\}$,
    determine $G_{\min}$ and $G_{\max}$, the minimum and maximum grey level in the block.
    Set $T_M = G_{\min} - 1$, $T_{M+1} = G_{\max}$
    Set $T_1 \ldots T_{M-1}$ to 0
    Set $T_{M+2} \ldots T_N$ to 255 (maximum grey level)
```

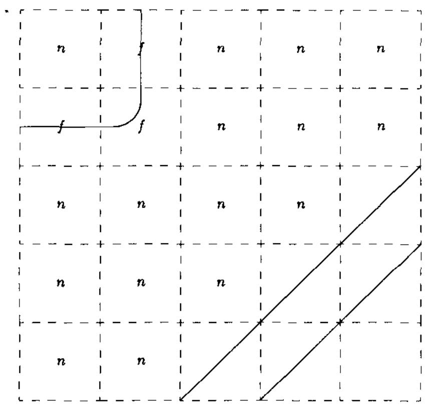
For each block $\in \{B_{ne}\}$,
if pixels fall into a threshold band other than M, then repeat above procedure on this block.

As an example, consider the following image:

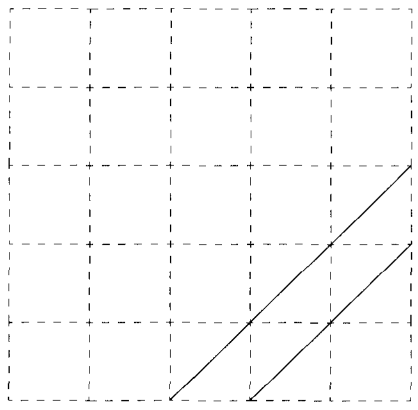


Thresholded image

This is an example of an image which was thresholded into a 3-level image with grey bands, 0, 1, and 2. Suppose its corresponding edge images are



Thresholded edge image



Original edge image

By comparing these edge images, we find that there are three false edge blocks (labeled f). Surrounding these blocks are 16 nonedge blocks (labeled n), 15 occupying threshold band 1 and one located in level 2. Therefore, this strongly indicates that the three false edge blocks should be forced to entirely occupy band 1. Let $G_{\min}$ and $G_{\max}$ denote the minimum and maximum grey values in a particular false edge block. To force the entire block into threshold band 1, set T_1 to $G_{\min} - 1$ and T_2 to $G_{\max}$.

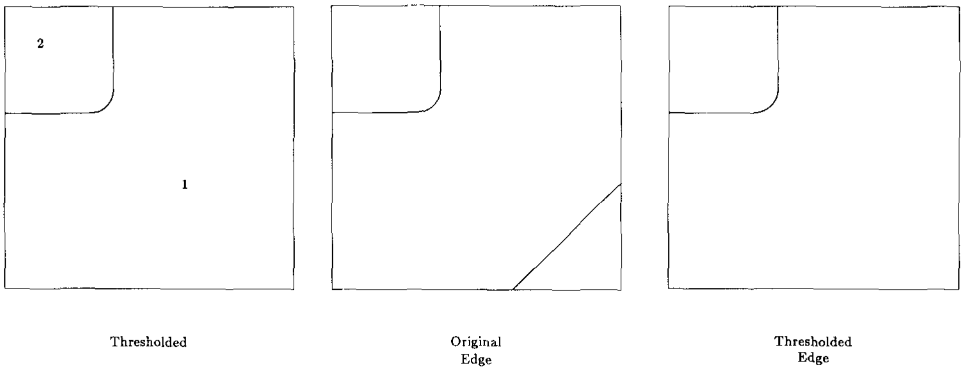
The only remaining problem is the single nonedge block at level 2. Since all of its neighbors occupy level 1 *and* there are no original edge pixels nearby, this block also must be forced into level 1 in an analogous manner.

(b) *Addition of a threshold*: Suppose an image requires N global thresholds but the subtracted edge image of a particular block has a large number of excess original edge pixels but relatively few excess thresholded pixels. This indicates that the block is missing a threshold. Therefore, one of the current thresholds must be deleted and a new value inserted into the threshold list. Since the next step will fine tune the location of this threshold, only an estimate is needed here. The simplest way to obtain this estimate is

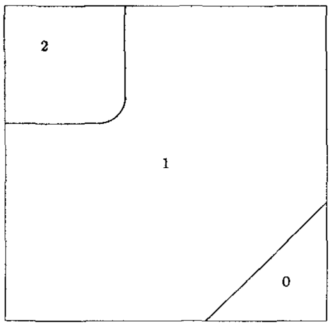
1. Let G = average grey level of all excess original pixels in block.
2. Insert G into the threshold list, temporarily assigning $N + 1$ thresholds to this block.
3. Let τ = the original threshold which is closest to the new threshold and delimits the band containing the smallest number of pixels.
4. Remove τ from the threshold list.

It is important to determine the effects of this operation. To understand this, count the number of grey levels represented in the thresholded block (before the thresholds were changed). Call this number k. If $k \neq N$, we know that there is an available grey level to accommodate the new edge. That is, the missing edge in the thresholded image may be recovered while retaining all other information. However, if $k = N$, we may assume that some information will be lost with the changing of the thresholds. That is, judging from this block alone, the image should require $N + 1$ thresholds. Unfortunately, since only N values are possible, this may cause the disappearance of small features in the image. One alternative to this problem would be to increase the number of global thresholds and repeat the entire process.

As an example, consider the following blocks:

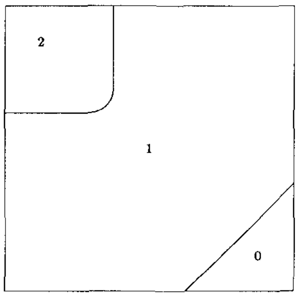


From the original edge block, it is obvious that an edge is missing. Assuming that there are 2 global thresholds T_1 and T_2, this edge may be recovered by forcing either the pixels above or below it to have a thresholded grey level of 0. To do this, one of the two thresholds in this block must be changed. Suppose that the average grey value of the pixels representing this edge is X. The problem is to determine which of the two global thresholds this value should replace. If $X \leq T_1$, set $T_1 = X$. Similarly, if $X \geq T_2$, set $T_2 = X$. If $X \in [T_1, T_2]$, the simplest test is to determine if X is closer to T_1 or T_2 and assign it that threshold. Other tests may be used if desired. The new thresholded block shown below would be the result if the pixels in the lower right corner had grey values less than X:

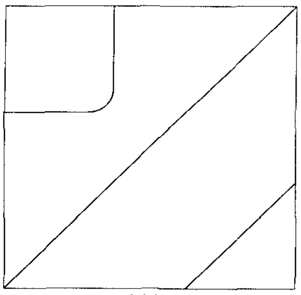


New
Thresholded

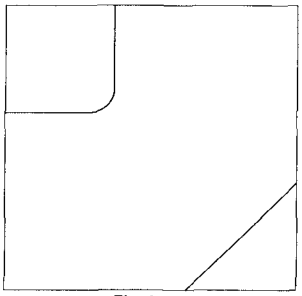
Next, consider these blocks:



Thresholded

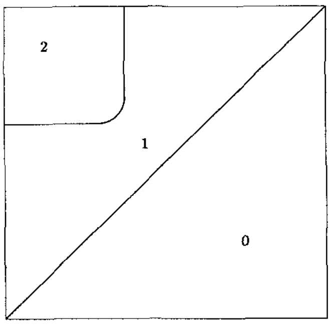


Original
Edge



Thresholded
Edge

Comparison of the original and thresholded edge blocks seems to indicate that an additional threshold is needed. Proceeding as in the previous example, a threshold may be chosen to restore the missing edge. However, only three grey levels are allowed. In the resultant block shown below, the thresholds were changed to incorporate the missing edge at the expense of the distinct region in the lower right corner. If several blocks in the image have this condition, it may be necessary to repeat the entire process, forcing the number of global thresholds to $N + 1$. This number may be incremented several times. However, as the number of global thresholds increases, the number of edge pixels necessarily increases. The resulting edge image can become very complex yielding contradictory information. The edge matching algorithm will tend to break down when this occurs. However, the actual cap on the number of global thresholds will be image dependent.



New
Thresholded

(c) *Relocation of a threshold.* If a block contains large numbers of both excess original and excess thresholded edge pixels, the thresholds simply need to be moved. The grey value of the excess original pixels suggests the correct value using the following procedure, assuming the image requires N global thresholds.

1. For each excess original pixel in the block, determine to which threshold the grey level of this pixel is closest (in absolute distance).
2. For each threshold, $\lambda(\lambda = 1 \dots N)$, if a large number of excess original pixels are near this threshold,
   (a) τ_λ = average grey level of all of these associated excess original pixels.
   (b) Consider the set of values surrounding τ_λ; e.g., $[\tau_\lambda - 10, \tau_\lambda + 10]$. Using each of these values, threshold the block. T_λ is set to the value that minimizes the number of excess pixels.

It will usually be the case that there will be sufficient data to change only one or two thresholds.[1]

Determination of thresholds from neighbors. If a block contains a significant number of one or both types of excess pixels but its thresholds were not changed in the previous step, new values may be determined by averaging those of the neighboring blocks whose thresholds were previously adjusted. This process may take several iterations before all blocks have been assigned new values. Any block with few of either type of excess pixel remains untouched.

Smoothing of the image. Since the underlying assumption is that the objects and the background are solid, small holes which appear in the thresholded image are probably artificial and should be removed. Also, by its design, the morphological edge detector does not find edges around noise points. Therefore, it is desirable to locate and remove noise found, especially in solid blocks. For both cases, a morphological clos-opening (a closing followed by an opening) using a square structuring element will smooth the image. This behaves similarly to median filtering [12].

This is the basic edge matching algorithm. The next section explores the details of the implementation and some results. It has been suggested [3] that, in order to ensure a smooth transition between blocks, the thresholds for each pixel be obtained by interpolating between blocks. This operation is not recommended since the thresholds chosen are based on the actual grey values in the block. Interpolation would counteract this process and introduce noise into the image. However, the transitions between blocks are automatically smoothed by the large degree of overlap of the blocks.

[1]Throughout the discussion of this algorithm, the actual quantities used to determine "sufficient data" or "significant numbers" are not constants. They are parameters set within the program and may vary due to the implementation of the algorithm, the quality of digitizing equipment, the object(s) in the image, and the size of the blocks. The next section provides some sample values.

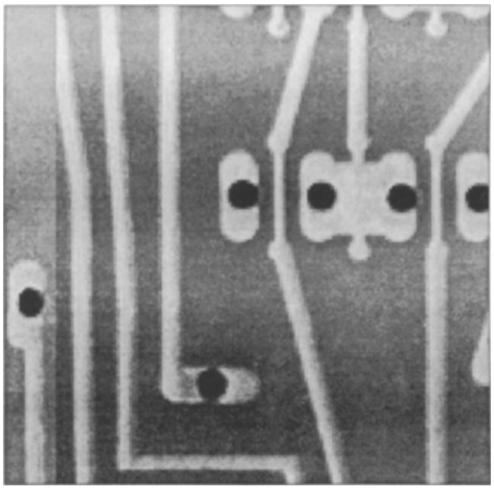


FIG. 1. Digitized PC board (G_{ij}).

4. IMPLEMENTATION AND RESULTS

As an example of an application of this algorithm, consider, the 160 × 160 image of the printed circuit (PC) board in Fig. 1 along with its grey histogram shown in Fig. 2. From the nature of the image, it appears that two values will be needed to successfully threshold this image, with the holes occupying the lowest grey band, the background in the middle level, and the traces and pads in the highest band. This may also be seen from the histogram which appears to contain three distinct, overlapping distributions with peaks at 32, 107, and 190. First, a simple histogram method [1, 14] was used to set the two global thresholds at 62 and 159. Using these thresholds, the 3-level image shown in Fig. 3 results. For display purposes, the thresholded image is assigned the grey levels 0, 128, and 255 in place of 0, 1, and 2. This result points out the problems encountered with global thresholding. The area of background in the upper left corner appears to be part of the traces while, in the lower region, several traces appear to be part of the background. This is due to effects of nonuniform illumination of the board while digitizing. Clearly, the goal of

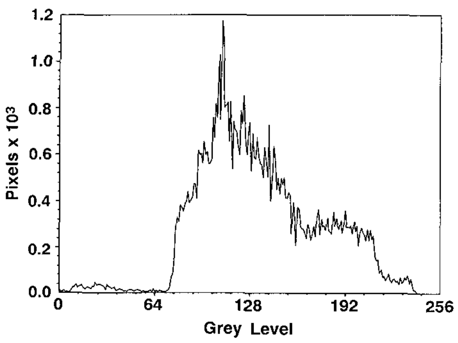


FIG. 2. Grey level histogram of PC board.

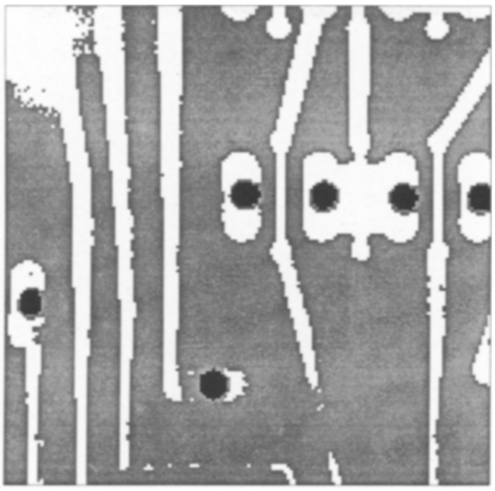


FIG. 3. Three-level PC board (P_{ij}).

simplifying the image while retaining the essential geometric structure has not been achieved by thresholding with two fixed global thresholds. Therefore, the thresholds must be locally adjusted to compensate for these problems. The next step involves computing the edge images of both the original and globally thresholded images, yielding the results shown in Fig. 4. (Note that, for display purposes, all edge images will be shown in their complemented forms. That is, edge points will appear as black and nonedge points as white.) The edge detector used was a morphological edge detector [11]. It was chosen because its resultant edges may be arbitrarily thin and it is insensitive to noise. However, any edge detector with these properties may be successfully used here for comparing edges. Note that the edges in the thresholded image are sharp, with 0–128 and 128–255 transitions; however, the corre-

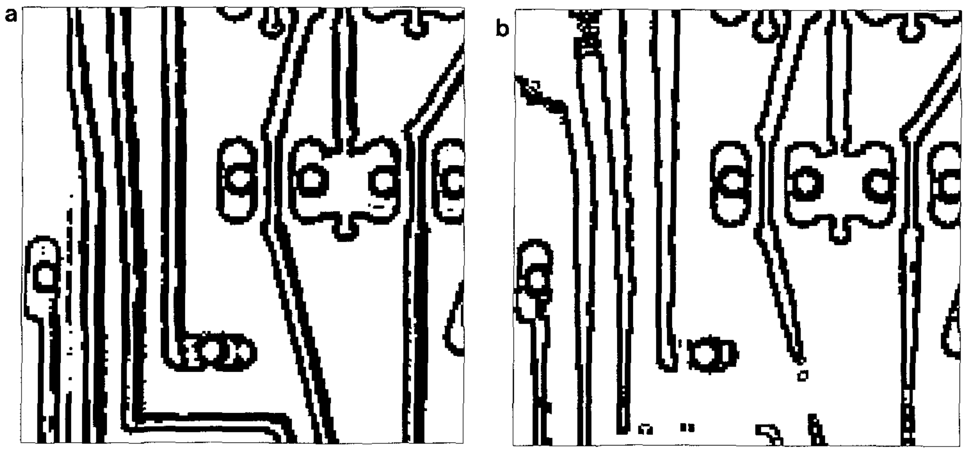


FIG. 4. Edge images of original and thresholded PC board (OE_{ij} and TE_{ij}).

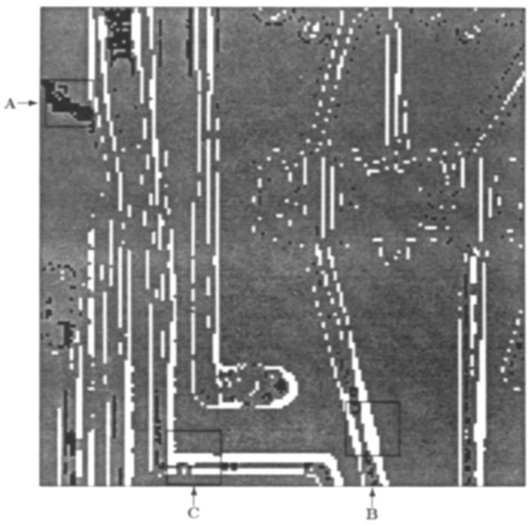


FIG. 5. Subtracted edge images (SE_{ij}).

sponding edges in the original image are much broader. Therefore, in order to perform a fair comparison, the thresholded image is blurred by averaging pixels in a 3 × 3 window.

Finally, these two edge images are subtracted, yielding the subtracted edge image shown in Fig. 5. For display purposes, the subtracted edge image contains the values 0, 128, and 255 which correspond to -1, 0, and 1, respectively.

The next step is to locate edges that are not present in the original image. To find the locations of these edges, the image is divided into 32 × 32 blocks which overlap by eight pixels on each side. Initially, the thresholds assigned to each block are 62 and 159, the global thresholds.

As an example, consider block A located in the upper left of Fig. 5, which has a large concentration of excess thresholded edge pixels. This area should be solid since it is part of the background. From the information about the neighbors, it appears that this block belongs in threshold band 2. Its minimum and maximum grey values are 152 and 182. To force the entire block into threshold band 2, 151 and 182 become the new thresholds for this block. This procedure is repeated for the entire image.

Step 2b deals with blocks having missing edges. Consider the block labeled B in Fig. 5. In the thresholded image (Figure 3), this block appears to be a part of the background when it is actually part of a trace. In the subtracted edge image, this block shows a large number of excess original pixels with mean grey value of 116. Since this value is closer to the global threshold of 159 and the threshold 159 removes only three pixels, the thresholds for this block become 62 and 116.

Again, this procedure is repeated for all blocks in the image, yielding Fig. 6. Block A is now entirely a part of the background and block B shows the presence of a trace. However, there are still smaller areas of background which appear as traces and traces which appear to be background. The next steps will clean up these regions.

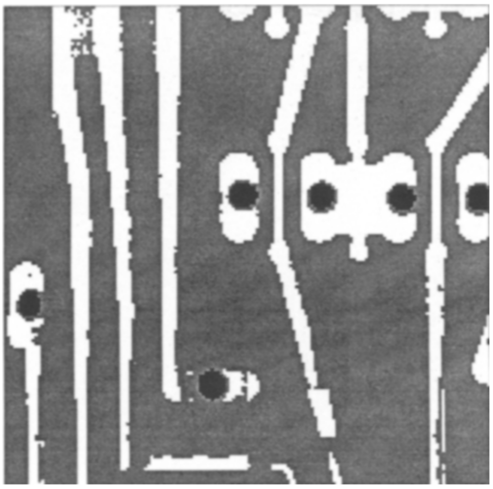


FIG. 6. Thresholded result after removing/inserting edges.

Step 2c adjusts thresholds to maximize coincidence of the edges. Consider Block C containing the trace along the bottom of Fig. 5. It contains both types of excess edge pixels. The majority of the excess thresholded edge pixels are closest to the highest global threshold (the second) and have mean 115. The block is then successively thresholded using the values in the range [105, 125]. The threshold that minimizes the number of both types of excess pixels is 121 so this number becomes the second threshold with the first remaining untouched. Again, this procedure is repeated for each block having sufficient information. As previously mentioned, the quantity which determines "sufficient" information is a global parameter whose value will be dependent on many factors in the particular type of image and digitizing environment. The value chosen here for the number of excess original and excess thresholded edge pixels was 1.5% of the number of pixels in each block, or 16, 12, and 10 for interior, border, and corner blocks, respectively. This number was chosen since it allows for the presence of noise. It may be decreased for systems with higher resolution and, therefore, fewer noise points or increased for noisy applications. This parameter is specific to our digitizing setup. Once chosen, it will remain constant.

For the remaining blocks, thresholds are assigned by interpolating between previously adjusted blocks yielding the thresholded image shown in Fig. 7. This image appears to be a good representation of the original image. Extracting threshold band 1 would yield only the holes and threshold band 3 would give the traces. However, these bands still contain some noise which we would like to remove. Figure 8 shows the result of the cleaning operations. The noise has been fairly well removed and the transitions between the holes and pads are much sharper. There are several places, however, where the image appears to be blocky. This arises during the clos-opening step. If the multilevel thresholding is being used as a preprocess to analysis where this blockiness presents a problem, the clos-opening step may be omitted or performed with a different structuring element.

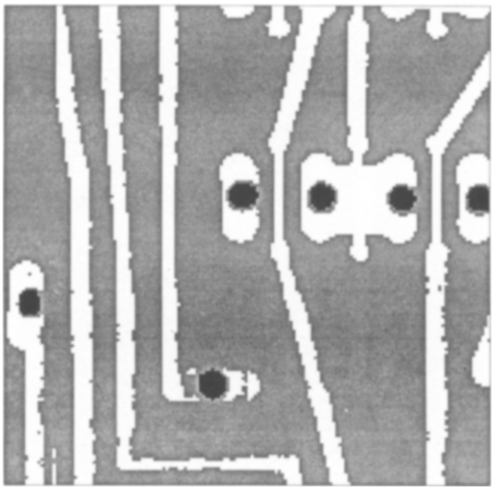


FIG. 7. Result of blockwise edge matching.

The final subtracted edge image still contains areas with high concentrations of either or both types of excess edge pixels. Unfortunately, due in part to the limitations of the edge detector, not all of this information could be used. In the future, improvements to the image cleaning process may be developed which will exploit this remaining information.

Figures 9 and 10 show two different types of grey level images (a), and the results after thresholding using (b) a set of global thresholds, and (c) a set of spatially varying thresholds determined by the edge matching algorithm. The first used the global thresholds of 102 and 147. The second required four thresholds with global values of 55, 77, 211, and 233. In both cases, the final image picks out many details

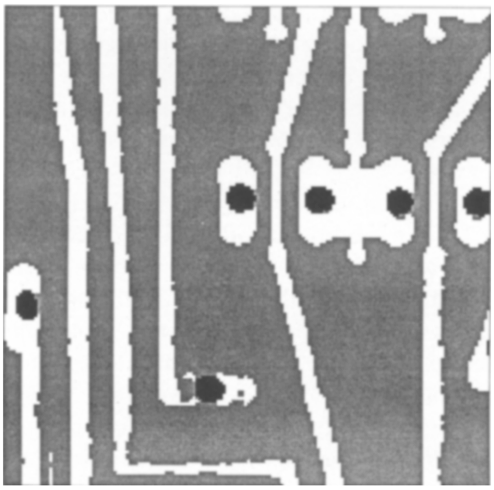


FIG. 8. Final thresholded image.

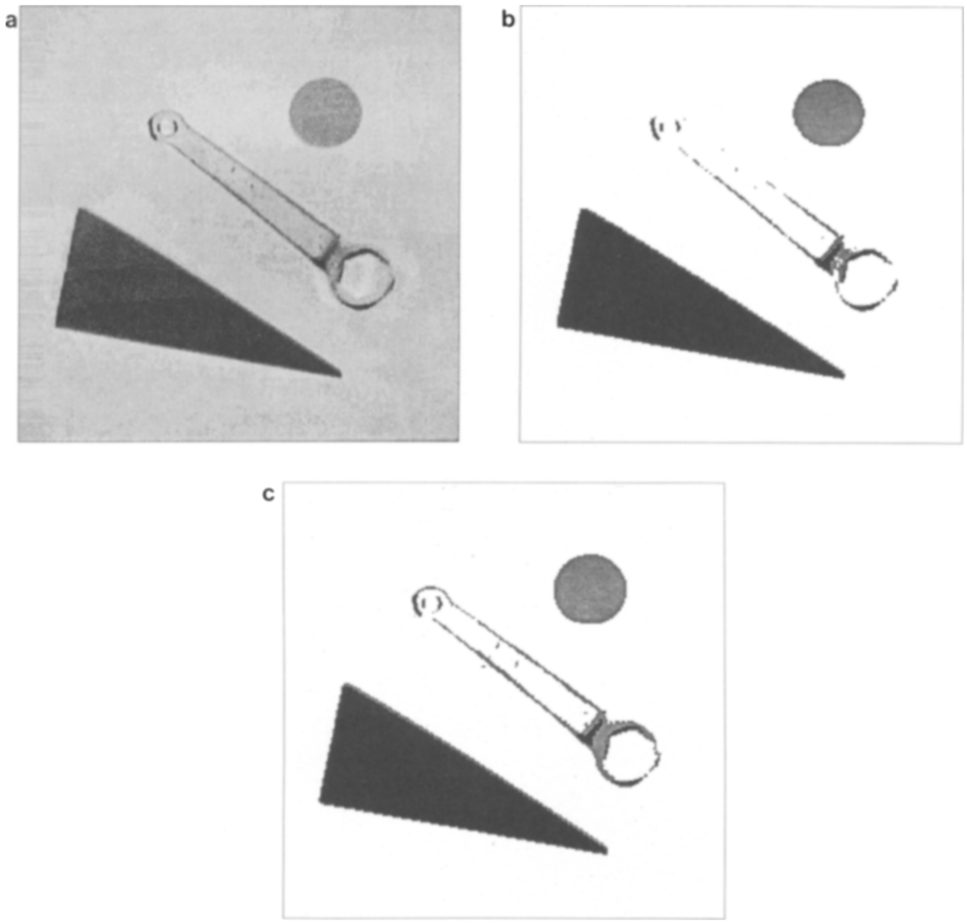


FIG. 9. Images of tools: (a) Original, (b) globally thresholded, (c) thresholded with spatially varying thresholds.

not present in the globally thresholded image. In the second image, the grass under the cameraman's feet is a "textured" region. As expected, the edge matching algorithm has difficulty here due to the large number of edges found in this area. If a more texture-insensitive edge detector replaced the morphological edge detector this noise would disappear. However, the use of such an edge detector would probably sacrifice some of the features identified and is, therefore, not advisable.

5. CONCLUSION

Given a grey tone image with one or more solid objects in a solid background, a set of global thresholds may be chosen. The image may then be divided into overlapping blocks and, by adjusting the thresholds in each block, a new image may be computed whose edge image closely coincides with the edge image of the original image, thereby preserving the geometrical properties of the image. Noise is then

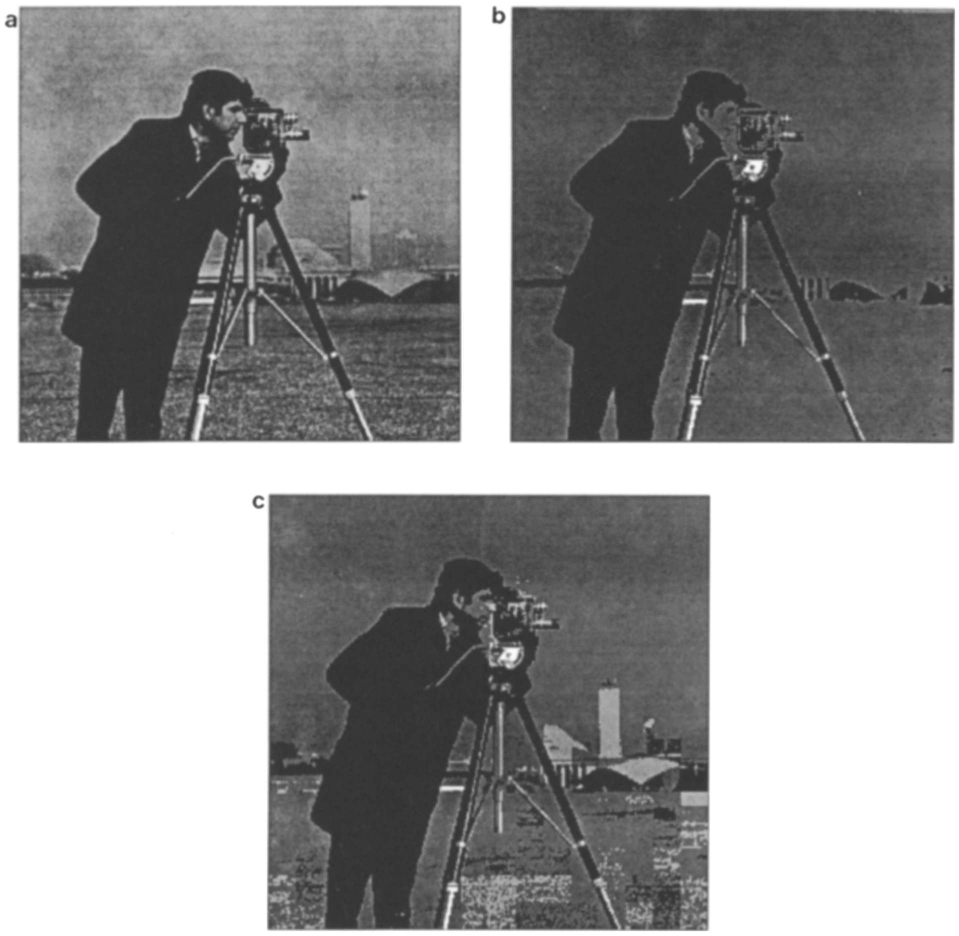


FIG. 10. Images of cameraman: (a) Original, (b) globally thresholded, (c) thresholded with spatially varying thresholds.

removed from this image yielding a multilevel image whose features may be extracted by removing specific grey levels.

It is important to consider some speed considerations of this algorithm. By far, the most time consuming process on a sequential machine is edge detection. Adjusting the thresholds for each block to make the edges coincide requires eleven edge detections per block. It is usually the case that only one threshold in a given block will require iteration. However, in the worst case, iterations could be required for all N thresholds. In our example, the image was divided into 225 blocks resulting, in the worst case, in $2475N$ block edge detections. This is another major advantage of using a morphological edge detector. Since morphological operations may be done in parallel, this will greatly reduce the time required for the edge detection step. In addition, there are many images in which we will be concerned mostly with noise problems. For these images, we may be able to omit this blockwise edge coincidence step entirely. In these cases, the speed of the algorithm is independent of N.

There are several constraints on this algorithm. By its design, it will only successfully threshold a given image if that image has distinct edges. Therefore, if the objects are blurry or textured, or shadows appear throughout the image, the extracted edges may not be distinct. If the image has too many objects, the edges may overlap and false edges may occur. Finally, as the number of thresholds, N, increases, the number of edge pixels necessarily increases, resulting in a complicated edge image. In all of these cases, this algorithm may not yield a good thresholded version of the original image. It is important to keep this in mind when increasing the number of global thresholds. There will be a limit to the number of thresholds a given image may have to achieve a good result. However, this number will be dependent on the particular image.

In the future, better thresholded images may be computed by exploiting the remaining information in the subtracted edge image. This data along with *a priori* information about the shapes found in the image may be built into a knowledge base and used to construct a very clean multilevel thresholded image. Finally, the result might also be cleaner by improving the digitizing setup to allow for more grey levels (such as 512 or 1024) or greater sampling density. In this manner, the edge image will be much sharper improving the performance of the edge matching algorithm.

REFERENCES

1. J. Weszka, A survey of threshold selection techniques, *Comput. Graphics and Image Process.* **7**, 1978, 259–265.
2. S. Wang and R. Haralick, Automatic multithreshold selection, *Comput. Vision Graphics Image Process.* **25**, 1984, 46–67.
3. C. Chow and T. Kaneko, Automatic boundary detection of the left ventricle from cineangiograms, *Comput. Biomed. Res.* **5**, 1972, 388–410.
4. Y. Nakagawa and A. Rosenfeld, Some experiments on variable thresholding, *Pattern Recognit.* **11**, 1979, 191–204.
5. D. Milgram, Region extraction using convergent evidence, *Comput. Graphics Image Process.* **11**, 1979, 1–12.
6. G. Medioni, Segmentation of images into regions using edge information, in *Proceedings, National Conference on Artificial Intelligence, Pittsburgh, PA, August, 1982*, pp. 42–45, American Association for Artificial Intelligence, Menlo Park, CA.
7. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
8. L. S. Davis, A survey of edge detection techniques, *Comput. Graphics Image Process.* **4**, 1975, 248–270.
9. T. Peli and D. Malah, A study of edge detection algorithms, *Comput. Graphics Image Process.* **20**, 1982, 1–21.
10. G. B. Shaw, Local and regional edge detection: Some comparisons, *Comput. Graphics Image Process.* **9**, 1979, 135–149.
11. J. S. J. Lee, R. M. Haralick, and L. G. Shapiro, Morphologic edge detection, *IEEE J. Robot. Automat.* **3**, 1987, 142–156.
12. P. Maragos, *A Unified Theory of Translation-Invariant Systems with Applications to Morphological Analysis and Coding of Images*, Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, July 1985.
13. J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
14. A. Rosenfeld and A. Kak, *Digital Picture Processing*, Vol. 2, Chap. 10, Academic Press, New York, 1982.