1-1-1977

A max-min measure for image texture analysis

Robert Mitchell

University of Missouri--Rolla

C. R. Myers

W. Boyne

Follow this and additional works at: http://scholarsmine.mst.edu/faculty_work

Part of the Electrical and Computer Engineering Commons

Recommended Citation

Mitchell, Robert; Myers, C. R.; and Boyne, W., 'A max-min measure for image texture analysis' (1977). Faculty Research & Creative Works. Paper 1756.

http://scholarsmine.mst.edu/faculty_work/1756

This Article is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. For more information, please contact weaverjr@mst.edu.
A Max–Min Measure for Image Texture Analysis

Owen R. Mitchell, Charles R. Myers, and William Boyne

Abstract—A new technique for image texture analysis is described which uses the relative frequency of local extremes in grey level as the principal measure. This method is invariant to multiplicative gain changes (such as caused by changes in illumination level or film processing) and is invariant to image resolution and sampling rate if the image is not undersampled. The algorithm described is computationally simple and can be implemented in hardware for real-time analysis. Comparisons are made between this new method and the spatial dependence method of texture analysis using 49 samples of each of eight textures. The new method seems just as accurate and considerably faster.

Index Terms—Digital image processing, feature extraction, pattern recognition, smoothing algorithms, texture analysis.

I. INTRODUCTION

The use of texture information in pattern recognition applications is increasing. Some objects are best described by means of their texture. These texture analysis problems are found in such diverse areas as biomedical imagery [1], industrial monitoring of product quality (such as a steel mill) [2], and in high altitude aircraft and satellite imagery (such as geoscience textures formed by drainage patterns in different rock types) [3].

Several researchers have described algorithms for texture analysis using both strictly statistical measures and heuristic techniques [4]–[9]. These techniques are generally computationally demanding and/or work only for limited conditions such as constant illumination and size, well defined edges, and uniform coarseness. A new technique is described here which is computationally simple and appears to be as accurate as the more complex methods.

II. BASIC MAX–MIN MEASURE

The new measure for texture analysis is based on the intuition that the important texture information for the human visual system is contained in the relative frequency of local extremes in intensity. The principal measurement in this process is the determination of the number of local grey level maxima and minima along a one-dimensional scan direction. The grey level values are first sent through a smoothing process which eliminates reversals of small amplitude, thereby retaining only the principal extrema. The smoothing algorithm is the digital equivalent of the familiar analog mechanical process known as gear backlash and it was originally described as a preprocessing method for character recognition [10].

The smoothing algorithm is described as follows. Let $x_k$ be the grey level of the $k$th point along the scan line and let $y_k$ be the "smoothed" value. Let $T$ be the value of a preassigned threshold parameter. Let us start with $y_1 = x_1$ and proceed according to the algorithm shown below:

\[
\begin{align*}
\text{If} & \quad y_k < x_{k+1} - \frac{T}{2} & \quad y_{k+1} = x_{k+1} - \frac{T}{2} \\
& \quad x_{k+1} - \frac{T}{2} < y_k < x_{k+1} + \frac{T}{2} & \quad y_{k+1} = y_k \\
& \quad x_{k+1} + \frac{T}{2} < y_k & \quad y_{k+1} = x_{k+1} + \frac{T}{2}
\end{align*}
\]

REFERENCES


Manuscript received October 25, 1975; revised March 20, 1976. This work was supported by the Advanced Research Projects Agency under Contract F-30602-75-C-0100.

The authors are with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.
Fig. 1. Backlash smoothing and extrema identification for a threshold distance of \( T \).

![Diagram](image)

Fig. 2. Extrema detected for three different thresholds.

![Diagram](image)

Fig. 3. Samples of each texture in random order. Each is \( 64 \times 64 \) points. The third row in order is wood, water, cork, sand, fur, grass, and paper.

![Diagram](image)

Fig. 4. Extrema for 2 of the 8 textures. All \( 64 \) lines of each texture block were used which gave a possible maximum of \( 64 \times 32 = 2048 \) extrema pairs. Note that as the threshold is increased, fewer extrema are detected and when the threshold exceeds the dynamic range of the picture, no extrema are detected. Since the ordinate is the log of the number of extrema, the slope of each straight line drawn between adjacent data points is proportional to the ratio of the number of extrema at these two thresholds.

![Diagram](image)

Fig. 5. Extrema for 2 of the 8 textures. All \( 64 \) lines of each texture block were used which gave a possible maximum of \( 64 \times 32 = 2048 \) extrema pairs. Note that as the threshold is increased, fewer extrema are detected and when the threshold exceeds the dynamic range of the picture, no extrema are detected. Since the ordinate is the log of the number of extrema, the slope of each straight line drawn between adjacent data points is proportional to the ratio of the number of extrema at these two thresholds.

As shown in Fig. 5, each texture produces a characteristic curve which can be used to identify the texture. However, as seen in Fig. 4, each different sample of the same texture produces a slightly different curve. These variations in the same texture are more pronounced as the texture samples get smaller. In fact the \( 64 \times 64 \) size used for these tests is close to the limit of human recognition of the various textures as can be seen by studying Fig. 3. Obviously, a larger size region would give less variance in the features and more accurate classification results.

III. EXTREMA COUNTING RESULTS

A set of eight texture pictures from a book by Brodatz [11] was used. The images were scanned and digitized with 8 bits of log data retained for each point (512 \( \times \) 512 points). Forty-nine samples (each \( 64 \times 64 \)) from each texture were used for the testing. Several samples of each texture are shown in random order in Fig. 3. The average grey level for each texture has been normalized so that only texture information remains. (The Max−Min Method does not use average grey level information.)

The results shown in Fig. 4 indicate the logarithm of the number of extrema for various threshold sizes for 2 of the eight textures. All \( 64 \) lines of each texture block were used which gave a possible maximum of \( 64 \times 32 = 2048 \) extrema pairs. Note that as the threshold is increased, fewer extrema are detected and when the threshold exceeds the dynamic range of the picture, no extrema are detected. Since the ordinate is the log of the number of extrema, the slope of each straight line drawn between adjacent data points is proportional to the ratio of the number of extrema at these two thresholds.

As shown in Fig. 5, each texture produces a characteristic curve which can be used to identify the texture. However, as seen in Fig. 4, each different sample of the same texture produces a slightly different curve. These variations in the same texture are more pronounced as the texture samples get smaller. In fact the \( 64 \times 64 \) size used for these tests is close to the limit of human recognition of the various textures as can be seen by studying Fig. 3. Obviously, a larger size region would give less variance in the features and more accurate classification results.

For each point (512 \( \times \) 512 points). Forty-nine samples (each \( 64 \times 64 \)) from each texture were used for the testing. Several samples of each texture are shown in random order in Fig. 3. The average grey level for each texture has been normalized so that only texture information remains. (The Max−Min Method does not use average grey level information.)

The results shown in Fig. 4 indicate the logarithm of the number of extrema for various threshold sizes for 2 of the eight textures. All \( 64 \) lines of each texture block were used which gave a possible maximum of \( 64 \times 32 = 2048 \) extrema pairs. Note that as the threshold is increased, fewer extrema are detected and when the threshold exceeds the dynamic range of the picture, no extrema are detected. Since the ordinate is the log of the number of extrema, the slope of each straight line drawn between adjacent data points is proportional to the ratio of the number of extrema at these two thresholds.

As shown in Fig. 5, each texture produces a characteristic curve which can be used to identify the texture. However, as seen in Fig. 4, each different sample of the same texture produces a slightly different curve. These variations in the same texture are more pronounced as the texture samples get smaller. In fact the \( 64 \times 64 \) size used for these tests is close to the limit of human recognition of the various textures as can be seen by studying Fig. 3. Obviously, a larger size region would give less variance in the features and more accurate classification results.
Fig. 4. Number (log_e) of extrema pairs versus threshold size for 15 samples of (a) cork, (b) wood.
Fig. 4. Continued.
IV. Feature Selection

Using the type curve shown in Fig. 4 as the data, various features have been tested as to their suitability for texture classification. Two methods have shown good results.

1) Extrema Ratios: The ratios of the number of extrema at each selected threshold to that at other thresholds were used as features. This is equivalent to using the slopes of the straight lines drawn between the ordinate values at the selected thresholds. The threshold settings were chosen empirically to be 130, 110, 90, 70, 50, 30, and 10. (The data were already in log form, ranging from 0 to 255.)

2) Normalized Extrema Ratio: This measure is similar to that above, except the thresholds at which extrema ratios are calculated are determined by the curve itself. In this particular test the first threshold was the one at which the number of extrema pairs was ≥90. Each succeeding threshold was 20 lower and ratios were calculated using each adjacent pair. This technique has the effect of sliding the curves in Fig. 4 horizontally until the points where the curves cross log, 90 = 4.5 on the ordinate are all aligned. Then the straight line slopes between various points on the curves are compared.

V. Classification Results

The 49 samples of each texture were divided into 36 training samples and 13 test samples. Then six features were calculated for each sample and two classification techniques were used: 1) a simple normalized Euclidian distance measure with all features weighted equally, and 2) a three nearest neighbor decision rule. Each texture was classified as a point in six-dimensional feature space. For the first measure, a mean and standard deviation in each dimension were calculated for each texture from the training samples and the distance for an unknown sample from the test set was measured from the mean in standard deviation units. The results are shown in Table I. Also included in the table are results using another technique discussed in the next section.

The classification matrix for the method using the three-nearest neighbors decision rule is shown in Table II. The most common confusion using the max-min method occur among wood, fur, and water, and between paper and cork.

VI. Comparison with Spatial-Dependence Technique

The most common texture classification techniques use statistical measures based on spatial dependence probabilities. In order to make the techniques comparable, we limited the technique described by Haralick et al. [4] to 6 features and to one dimension. The six features were angular second moment, contrast, correlation, entropy, and two information measures of correlation. We used 64 grey levels in the spatial dependence matrix. The best results we could obtain in using this technique was 83 percent accurate on training samples and 66 percent accurate on test samples as shown in Table I. The classification matrix for the result is shown in Table III. With 14 features in each of two dimensions the spatial dependence technique accuracy was increased to 94 percent accurate on training samples and 89 percent accurate on test samples. However, the computation times on the CDC 6500 were indicative of the relative efficiency of the max-min technique. The total feature generation and classification for 49 samples of each of 8 textures required 250 CPU seconds for the 6 feature max-min technique, 845 CPU seconds for the 6 feature spatial dependence technique, and 1690 CPU seconds for the 28 feature spatial dependence technique. However, it should be noted that little effort was made to minimize the running time of any of the above techniques.
TABLE I
Classification Results Using 36 Training Samples and 13 Test Samples of Each of 8 Texture Patterns

<table>
<thead>
<tr>
<th>Features Used</th>
<th>3-Nearest Neighbor Decision Rule</th>
<th>Weighted Distance Decision Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Test</td>
<td>Training</td>
</tr>
<tr>
<td>1. Max-Min Method</td>
<td></td>
<td>93.8</td>
</tr>
<tr>
<td>Extrema Ratios, 6 Features</td>
<td></td>
<td>86.8</td>
</tr>
<tr>
<td>2. Max-Min Method</td>
<td></td>
<td>82.7</td>
</tr>
<tr>
<td>Normalized Extrema Ratios, 6 Features</td>
<td></td>
<td>94.5</td>
</tr>
<tr>
<td>3. Spatial Dependence Method</td>
<td></td>
<td>89.8</td>
</tr>
<tr>
<td>8 Features, One Dimension</td>
<td></td>
<td>86.8</td>
</tr>
<tr>
<td>4. Spatial Dependence Method</td>
<td></td>
<td>94.5</td>
</tr>
<tr>
<td>28 Features, Two Dimensions</td>
<td></td>
<td>94.5</td>
</tr>
</tbody>
</table>

TABLE II
Classification Matrix for Max-Min Method Using 6 Examples Ratio Features and 3-Nearest Neighbor Decision Rule

| Assigned Category | Training Samples | Test Samples | |
|-------------------|------------------|--------------|
| Textures           | 1 2 3 4 5 6 7 8 |              | 1 2 3 4 5 6 7 8 |
| 04 Cork            | - - - - - - - - | 9 - - - - 4 - - |
| 070 Wood           | - 34 - - - - - 2 | - 12 - - - 1 - |
| 069 Wood           | - - 32 4 - - - - | - - 10 3 - - - |
| 093 Fur            | - - - 35 - - - 1 | - - - 13 - - - |
| 029 Sand           | - - - - 34 2 - - | - - - 11 2 - - |
| 057 Paper          | 2 - - - 2 32 - - | - - - - 13 - - |
| 038 Water          | 2 - - - - 34 - | - 2 5 - - - 6 - |
| 09 Grass           | - - - 1 - - - 35 | - - - - - 13 - |

TABLE III
Classification Matrix for Spatial Dependence Method Using 6 Features and 3-Nearest Neighbor Decision Rule

| Assigned Category | Training Samples | Test Samples | |
|-------------------|------------------|--------------|
| Texture            | 1 2 3 4 5 6 7 8 |              | 1 2 3 4 5 6 7 8 |
| 04 Cork            | 29 - - - 7 - - | 7 - - - 2 3 1 - |
| 070 Wood           | - 25 10 - - - 1 | - 9 3 - - - 1 - |
| 069 Wood           | - 4 32 - - - - | - 4 9 - - - - |
| 093 Fur            | - - 1 34 - - - 1 | - 3 2 8 - - - |
| 029 Sand           | 5 - - - 30 1 4 | - - - 6 1 6 - |
| 057 Paper          | 1 - - - 5 29 1 - | - - - 4 9 - - |
| 038 Water          | - - - 2 - - 34 | - - - - - 13 - |
| 09 Grass           | 31 - - - 4 1 - 8 | - - - 4 1 - 8 |

VII. DISCUSSION

The max–min appears to be a promising measure of texture characteristics. The method has presently been used in one dimension only. One two-dimensional extension would be to measure the max–min features in several directions and use that direction which maximizes some criteria plus the orthogonal direction. This would make the algorithm invariant to texture rotation as well as adding a measure of the rotational symmetry of the textures. The methods might also be used to detect texture boundaries by measuring features in two opposite directions from a suspected boundary.

It is fairly easy to incorporate the computing required for the max–min feature extraction in special-purpose hardware. This would make real-time texture analysis possible. This is very important for applications such as steel mill output monitoring where a decision must be reached quickly as to whether to let the metal continue cooling or to reprocess it.

Also the quantities measured here (number of extrema versus threshold) might be called a first-order effect. The two curves for cork and paper in Fig. 5 are almost identical and confusions might necessarily be expected in a classification algorithm which uses only slopes of these curves. However, second-order measurements which include information
as to how the small extrema are interspersed among the large extrema would differentiate between these two textures. This might be the beginning of a hierarchical structure of texture primitives: those that differ in first-order measurements and those that differ in second-order measurements.

REFERENCES


An Algorithm for Testing Random Access Memories

JOHN KNAIZUK, JR., AND C. R. P. HARTMANN

Abstract—This correspondence presents an optimal algorithm to detect any single stuck-at-1 (s-a-1), stuck-at-0 (s-a-0) fault in a random access memory using only the n-bit memory address register input and n-bit memory buffer register input and output lines. It is shown that this algorithm requires $4 \times 2^n$ memory accesses.

Index Terms—Fault detection, optimal algorithm, random access memory, single stuck-at-fault.

I. INTRODUCTION

It is important to find methods for testing sequential networks that can produce a minimum test length to fully test a circuit [1], [2].

This correspondence presents an optimal algorithm to detect any single stuck-at-1 (s-a-1), stuck-at-0 (s-a-0), fault in a random access memory (RAM). The concept of detection is the prime concern when we are in a field testing situation where down time of a computer is costly. This algorithm can be used in the initial diagnostic to see if a fully contained RAM has a fault. With the advent of LSI circuits, a RAM that contains all its subsystems in one package is realistic.

This algorithm requires $4 \times 2^n$ memory accesses for a memory array with $2^n$ addresses. The RAM under test is entirely self-contained in that the only input/output connections will be the n-bit memory address register (MAR), the memory data register (MDR), and the associate power and timing inputs. This dictates that the main procedure of testing is by placing information through the MDR into the memory array and then checking if we can correctly read the same information out. An important assumption will be that the decoder is of a noncreative design [3]. That is, that a single fault within the decoder does not create a new memory address to be accessed without also accessing the programmed address.

In Section II the algorithm is presented. In Section III it is shown that this algorithm detects any single "stuck-at" fault and it is optimal.

II. OPTIMAL RAM TEST ALGORITHM

Before stating the algorithm let us introduce some notations. Let $A_i$ be the memory address $\mu$.

\[ 0 \leq \mu < 2^n. \]

Let

\[ \pi_0 = [A_i| \mu = 0(\text{modulo } 3)], \]
\[ \pi_1 = [A_i| \mu = 1(\text{modulo } 3)], \]
\[ \pi_2 = [A_i| \mu = 2(\text{modulo } 3)]. \]

Algorithm

Step 1: Write the all 0 word, $W_0$, at all locations $A_i \in \pi_0$ and $A_k \in \pi_2$.

Step 2: Write the all 1 word, $W_1$, at all locations $A_i \in \pi_0$.

Step 3: Read all locations $A_j \in \pi_1$;

if output $W_0 = \text{no fault indicated};$

$W_1 = \text{RAM fault indicated}.$

Step 4: Write the all 1 word $W_1$ at all locations $A_j \in \pi_1$.

Step 5: Read all locations $A_k \in \pi_2$;

if output $W_0 = \text{no fault indicated};$

$W_1 = \text{RAM fault indicated}.$

Step 6: Read all locations $A_i \in \pi_0$ and $A_j \in \pi_1$;

if output $W_1 = \text{no fault indicated};$

$W_0 = \text{RAM fault indicated}.$

Step 7: Write and then read the all 0 word $W_0$ at all locations $A_i \in \pi_0$.

If output $W_0 = \text{no fault indicated};$

$W_1 = \text{RAM fault indicated}.$

Step 8: Write and then read the all 1 word $W_1$ at all locations $A_k \in \pi_2$.

If output $W_1 = \text{no fault indicated};$

$W_0 = \text{RAM fault indicated}.$

END.

The above sequence requires a total of $4 \times 2^n$ memory accesses.

This algorithm can be viewed in a tabulated form as in Fig. 1.

III. VERIFICATION OF ALGORITHM

The RAM subsystems (MAR, MDR, decoder, memory array) will be separately analyzed using the above algorithm to show that all single s-a-1, s-a-0 faults are tested. This is possible since we are assuming a single fault condition and therefore no masking due to faults in different subsystems can occur.

In order to do this we will need the notion of Hamming distance. The Hamming distance between two binary memory addresses equals the number of binary digit positions in which the two disagree.