David McAuley

Row-by-Row Dynamic Image Analysis of a Matrix of Scanned Points

A practical image analysis algorithm is described that incorporates features to permit it to carry out its analysis on a row-by-row basis, in contrast to a full image matrix basis. A significant storage saving is realized, since the total image to be scanned is substantially larger than that used by this method. The dynamic build-up of the image on a scan-by-scan or row-by-row basis is carried out by a series of connectivity, pivot, chain set-up, and chaining algorithms used in conjunction with a dynamic memory allocation strategy.

Introduction

This paper describes in some detail an image analysis algorithm which incorporates several significant and important features not associated with other image recognition and analysis systems. These special features arise mainly because of the method in which the analysis is carried out on a row-by-row basis.

In commercial applications of image recognition systems involving microprocessors, the design engineer is faced with the problem of balancing throughput capability and hardware costs. This usually translates into a trade-off between memory size and I/O rates. In the discussion which follows, we use the testing of a plasma display panel as the point of reference. Such a panel consists of many crosspoints, each of which can be used to display a picture element (pixel). Current methods of testing display panels check each crosspoint to determine whether it can be switched on and off (erased). Clearly the usability of a panel containing defective cells depends on a good knowledge of the pattern of failed crosspoints or cells.

In the case of the plasma display panel comprising 960 rows by 768 columns of addressable pixels, a total of 737 280 pixels must be considered. To obtain the required resolution, and thus be able to differentiate the status (on/off) of adjacent pixels on the same line, each row has to be scrutinized independently of the others. Thus, the video information on row 1 could be relevant to the video information on row 960, yet the system can only *see* one row at a time. Since the panel cannot be viewed in its entirety as a large 960 × 768 matrix, the "compressed" storage of the video information is crucial.

In addition, to record the data after a single total scan of the panel would involve several seconds of processing and require at least 92K bytes of memory [(960 × 768)/8—each cell being represented by a single bit]. This time was unacceptable, and thus an algorithm incorporating overwhelming savings in storage and time had to be devised, thus enabling video data to be recorded and analyzed dynamically on a row-by-row basis.

In the design of a cost-effective solution to a practical problem, the following method was devised. The panel is scanned and a program is called by the control processor to determine the acceptability of the panel on the basis of a predetermined set of criteria. These criteria are used to specify the number and proximities of allowable defective cells on the panel. The allowable defects are also known as *cosmetic defects* and they can appear as solitary failures or as varying types and sizes of patterns under certain circumstances. The purpose of the cosmetic analysis software is therefore to detect and analyze these solitary and/or clusters of failing cells.

The following sections describe how this analysis is carried out for the plasma panel. However, the author believes that the algorithm, and adaptations of it, can be used in almost all

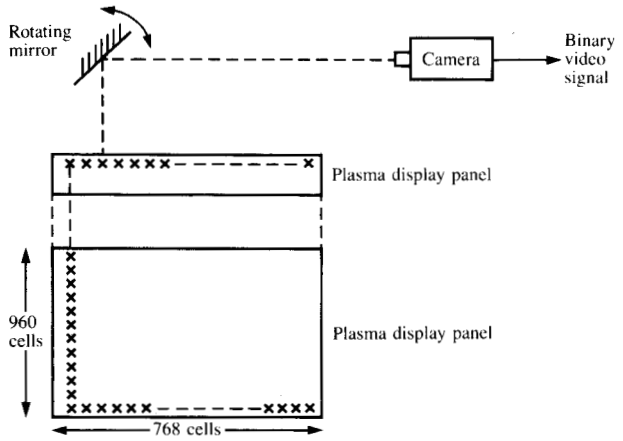

Figure 1 Schematic of plasma display panel test rig. The camera scans one row at a time.

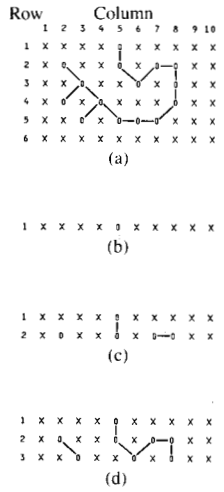

Figure 2 (a) Six rows of matrix constructed from a scanned image. The zeros represent defective cells, while the Xs represent good ones. (b)–(d) Illustration of the build-up of the matrix as each successive row is scanned.

areas of image analysis and inspection, remembering that cell elements with connecting entities (which would result in the building-up of a picture) can be replaced with picture elements. The same type of analysis is applicable to problems in

- pattern and character recognition,
- document handling,
- robotics, mechanized or automated handling, and test and inspection (e.g., front-of-screen testing),
- image analysis, and
- image orientation.

While further work is required to fully substantiate this belief, an example of a more complicated pattern type, an object within an object, is shown at the end of this paper to be just as easily detected and identified by the implemented algorithm.

System analysis

Individual cell information is passed to the software via a binary *video* signal from the camera (Fig. 1). This video signal reflects the status of one row of cells, since the panel is scanned one row at a time by means of a rotating mirror. Typically, there may be 768 rows with each row comprising 960 cells, giving a total of 737 280 cells to be tested. Data transfer is achieved by successive reads of a first-in/first-out (FIFO) register where the binary video signal is held, with each read accessing one eight-bit byte of information, or 120 reads of the FIFO register per row ($8 \times 120 = 960$).

The binary data can then be compared to a previously generated mask of expected data in order to detect failing cells within a row (parts of images in a different application). When a defective cell is encountered, the main body of the software is invoked, and it proceeds to analyze this cell with respect to all other defective cells and with reference to the specified criteria.

Since a large panel cannot be scanned or viewed in its entirety, a method of "building up a picture" of the panel, row by row, is used. Since defective cell data in one row may be pertinent to defective cell data in the following row (not yet scanned), a method of logging or storing this information is required. Because of the size of the panel, and therefore the number of possible defects, a dynamic memory allocation strategy for recording failing cell data is incorporated in the algorithm to avoid unnecessary saturation of storage.

An important feature of this algorithm is the ability to recognize the *connectivity* of failing cells. This is accomplished by constructing a series of inter-relating tables and pointers which can interpret the cell information and build up a *network of defective cells* that can best be described as a type of skew matrix [Fig. 2(a)].

Note that the panel is scanned one row at a time. Figures 2(b)–(d) represent the information which is available after scanning rows 1 to 3, respectively. In the example shown, after row 1 has been scanned, a solitary failing cell can be detected. After row 2 has been scanned, a solitary cell and two clusters of two adjacent failures can be detected (with the previous solitary defect becoming one of a cluster of two). Only after scanning row 3 does the algorithm (or the program) detect a connection which links up the two clusters of two, making (at this stage) a combined group of six failures, along with the newly formed cluster of two. This scanning is repeated until the entire panel is analyzed, or

until a failing cell violates one of the defined criteria, thus rendering the panel unacceptable.

Criteria governing acceptability of the panel

Having established an algorithm which analyzes the cosmetic defects and determines their *connectivity*, we need a set of guidelines which monitor the acceptability of the panel. These guidelines take the form of a predetermined set of criteria which control the number of interconnected failing cells allowable on the panel. The criteria are programmable; i.e., the test engineer or programmer responsible can alter the parameters of the criteria, such as the number of defects allowed within any given area of the panel, or the number of allowable clusters of two defective cells (three defective cells, etc.), or the maximum number of allowable adjacent defective cells.

If any of these conditions are not satisfied, the software "signals" the control processor, which in turn must decide whether to fail the panel or to instigate further tests. In a different application, the interconnected *defective* cells constitute the image to be scanned, and/or the object to be registered, identified, or analyzed.

System design

There are a multitude of permutations in which the defective cells can appear on the panel. The algorithm therefore must be robust (i.e., perform well in a variety of environments) and capable of detecting even the most obscure pattern arrangement. At this point it is appropriate to define several of the system entities which are created and used within the algorithm. These take the form of tables, which are outlined briefly.

- *Network table*

This table is used in determining the "connectivity" of defective cells, and characteristics relative to this feature are logged in it. Each defective cell encountered therefore has a network table entry, and this also permits processing of adjacent defects on the same row. The table format is shown in Table 1.

- *Node table*

Each isolated group (two or more) of defective cells is treated as a node in a connecting network. This information is logged in the node table, which ensures the processing of the varying number of clusters allowable; e.g., the criteria may allow three groups of three defects; five groups of two, etc. Table 2 illustrates the format of the node table. The MAX table entry is used by an *out-of-range* algorithm which decides whether the cluster of defects represented in the node table is isolated from the current row of cells under scrutiny, thus freeing up the node table (now flagged for deletion) for use by other possible defective clusters, and storing the results in a more compact log table. The node table need not be large. (The table currently in use enables the handling of clusters of up to five defects, but this can be expanded if necessary.) The interaction between the network table and the node and scan tables is outlined briefly later.

- *Pivot table*

The pivot table processes information relating to the number of failing cells allowable within any given area of the panel (this number is specified in the criteria).

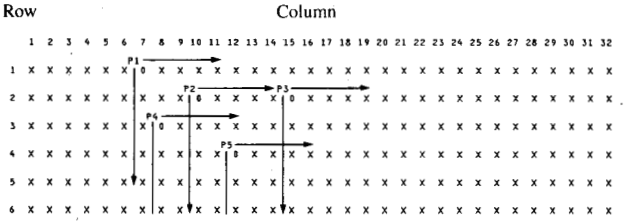

Figure 3 Pivot corners being formed for each defect. The rejection criterion sets the maximum extent for a defect cluster, here shown as five cells by five cells.

Table 1 Network table format and definition.

ROW	2 bytes	Row number of defective cell.
CELL	2	Cell/column number of defective cell.
LHADJ	1	Set if preceding adjacent cell was also defective.
C1 C2 C3	2 2 2	Possible defective cell connections in next row.
NODE TABLE POINTER	2	Address of node table for this particular defective cell.

Table 2 Node table format and definition.

ROW	2 bytes	Row number of defective cell.
CELL	2	Cell/column number of defective cell (repeated as many times as there are interconnected defects in this same cluster).
AA	2	Flags the last node in the table.
CNT	1	Number of defective cells in table.
MAX	2	Maximum row number of cells in table.
FSCELL	2	First cell number detected in a cluster of two or more (used to indicate where on the panel the defective cluster is situated).

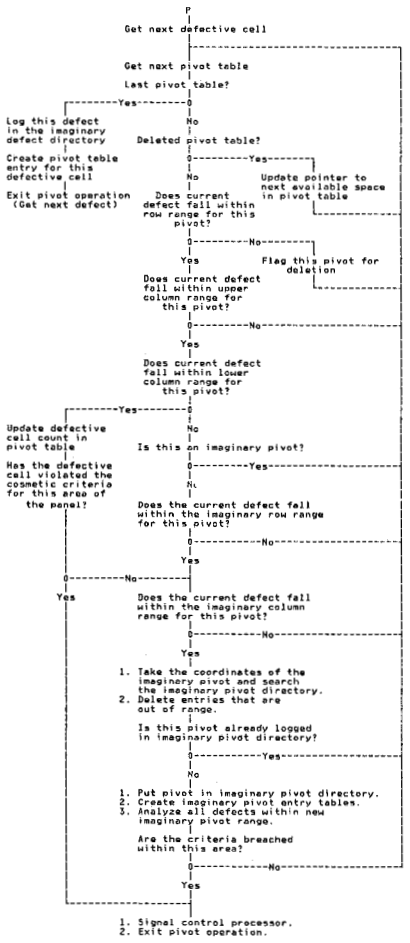

Figure 4 Pivot algorithm.

Table 3 Pivot table format and definition.

ROW	2 bytes	Row number of defective cell (pivot row).
CELL	2	Cell/column number of defective cell (pivot column).
CNT	1	Number of defective cells within this area.
ROW RANGE	2	Upper row limit for the area.
CELL RANGE	2	Upper cell limit for the area.
IMAGINARY PIVOT FLAG	2	This flag is set if the pivot table entry has been constructed around a defect which does not exist at that particular coordinate on the panel. However, the pivot entry has to be made at this point because there is no guarantee that any 100-mm-square defective area will be aligned with a defective cell as its corner pivot.

Each defect encountered forms the corner pivot of an area which is bounded geometrically by a discrete displacement from the pivot (for example 100 mm × 100 mm square). In other applications this might represent the expected size of the object to be labeled. Figure 3 illustrates five defects with each defect forming the corner pivot of an area five cells by five cells square. For example, the row and column ranges which fall within the jurisdiction of pivot P1 are 1 to 5 and 7 to 11, respectively, with three defects in this area (P1, P2, and P4) and so on. It can also be seen from this diagram that the pivot itself can be processed if it lies within another pivot's area, e.g., P2 and P4.

The pivot table is illustrated in Table 3 and the pivot operation is briefly outlined in Fig. 4.

• *Scan table*

The scan table holds details concerning complete areas of failing cells; it facilitates the linking of several of these areas in order to scan the affected area and apply the appropriate test patterns.

Table 4 outlines the format and defines the entries of the scan table. The forward-chain (CHAINF1, CHAINF2, CHAINF3) and backward-chain (CHAINB1, CHAINB2, CHAINB3) entries are used in linking up connecting groups of failing cells, as discussed in the next section. Each scan table currently allows the linking of three other scan tables (but once again this can be expanded, if necessary).

Fast fail, connectivity, and chain set-up algorithms

These algorithms are used during scan and during cosmetic operations. The only difference is that during scan, the panel fail routines are disabled and chain set-up is included. Also, the defective cells are not stored, since during scan, whole areas of failing cells are analyzed, which would be wasteful of time and storage. Instead, the failing cell coordinates are updated each time, as necessary. To simplify the explanation of the steps involved, the current defective cell under scrutiny is called the NEW (see Fig. 5) and the defective cell connected to it on the previous row is called the OLD. Some other definitions are also required.

When we say that the left-hand (LH) adjacent cell is defective, we mean the cell immediately to the left of the current cell under scrutiny on the same row; similarly, the right-hand (RH) adjacent cell means the cell immediately to the right on the same row. This meaning is also illustrated in Fig. 5. In this example NEW has a LH adjacent defective cell, and OLD has a RH adjacent defective cell. On the other hand it can also be said that NEW is RH adjacent to a previously scanned defect.

The fast fail, connectivity, and chain set-up algorithm is outlined briefly in flowchart/tree diagram form in Fig. 6.

Chaining algorithm

Having constructed the individual scan tables and set up the chain addresses and other defective cell information, we require a method of linking these scan tables, or groups of failing areas, in order to analyze the defective portion of the panel.

This method of linking is called the *chaining algorithm.* Before this algorithm is described in flowchart/tree diagram format, consider first the defective area of cells outlined in Fig. 7(a). This illustrates the layout of the defects and how they are connected to one another. This diagram shows that a link can be traced from one defect to any other in this example.

Now consider Fig. 7(b), which details the connectivity or network characteristics of this particular panel. Figure 7(c) shows how the cosmetic analysis program partitions this area into individual groups of varying shapes and sizes *row by row.* Each area illustrated in this diagram is represented by a unique scan table in the system.

Continuing from this, Fig. 7(d) illustrates how the individual groups or tables are linked to one another. In order to complete the analysis for the entire area, we must progress down the panel, but also be able to retreat through the panel and gradually exhaust all of the possible routes or connections to and from all defective areas.

Figure 7(e) summarizes even further by representing each defective area as a node in a tree diagram. Only after every branch in the tree has been traversed and each node processed is the analysis complete for this group of failing cells.

As with the previous algorithm, some clarifying is needed before referring to the flowchart/tree diagram of Fig. 8, which outlines the chaining algorithm briefly. During the linking cycle it is possible that several areas could be omitted from the processing due to the route taken through the tables. If we then arrive at a terminal node and have also exhausted the backward chains, but other unprocessed areas remain, we must have some method of re-entering the cycle and completing the analysis. This is achieved by building a backward-chain address queue. This queue holds the addresses of the backward pointers in the network and can be accessed if we get lost while in the process of reaching a terminal node, having exhausted all of the backward chains. This situation arises quite often if a scan table has two or more backward pointers. Only when the queue is empty can we be sure that the job is complete.

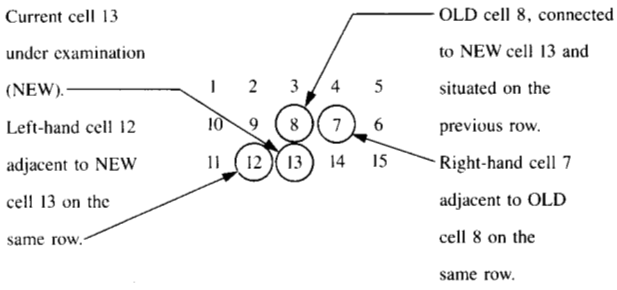

Figure 5 Definitions of adjacency in scan table.

Table 4 Scan table definition (each entry is two bytes).

CHAIN F1–3	These are the addresses of the scan tables which will be directly accessed and "chained to" whenever a link between the tables is discovered.
CHAIN B1–3	These are the addresses of the scan tables which already have been linked to the current table, but which may not necessarily result in continuous chaining until termination of the link. Therefore it may be necessary to "retreat through the chain" until termination of the link is completed.
MAX/MIN ROW-CELL	These are the maximum row and cell numbers and minimum row and cell numbers within the table. They are updated each time a defective cell is encountered and provide the coordinates within which the failing area lies.
#CHF	This holds the number of forward-chain addresses.
#CHB	This holds the number of backward-chain addresses.
DUPFLG	This flag is set to avoid duplicate processing of the defective cell area.
QUEBCK	This flag indicates whether or not the scan table address has been stacked on the backward-chain address queue.

Pivot algorithm

The function of the pivot operation is to process and analyze the cell data on the basis of the criteria which specify the number of defects allowable within any given square area of the panel. The basic pivot operation has already been illustrated diagrammatically in Fig. 3, but there is an additional procedure which forms an important part of the pivot process.

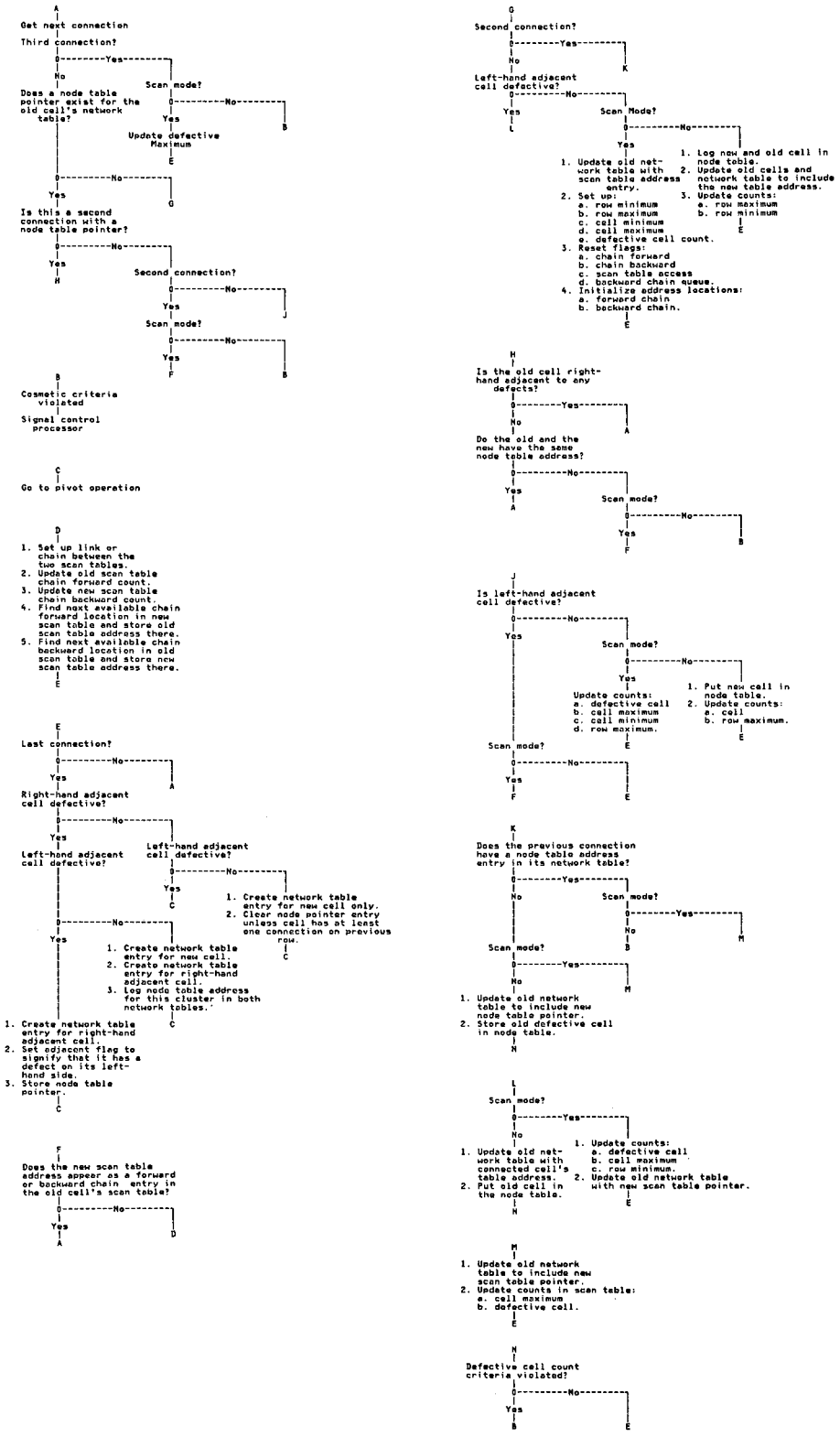

Figure 6 Fast fail, connectivity, and chain set-up algorithm.

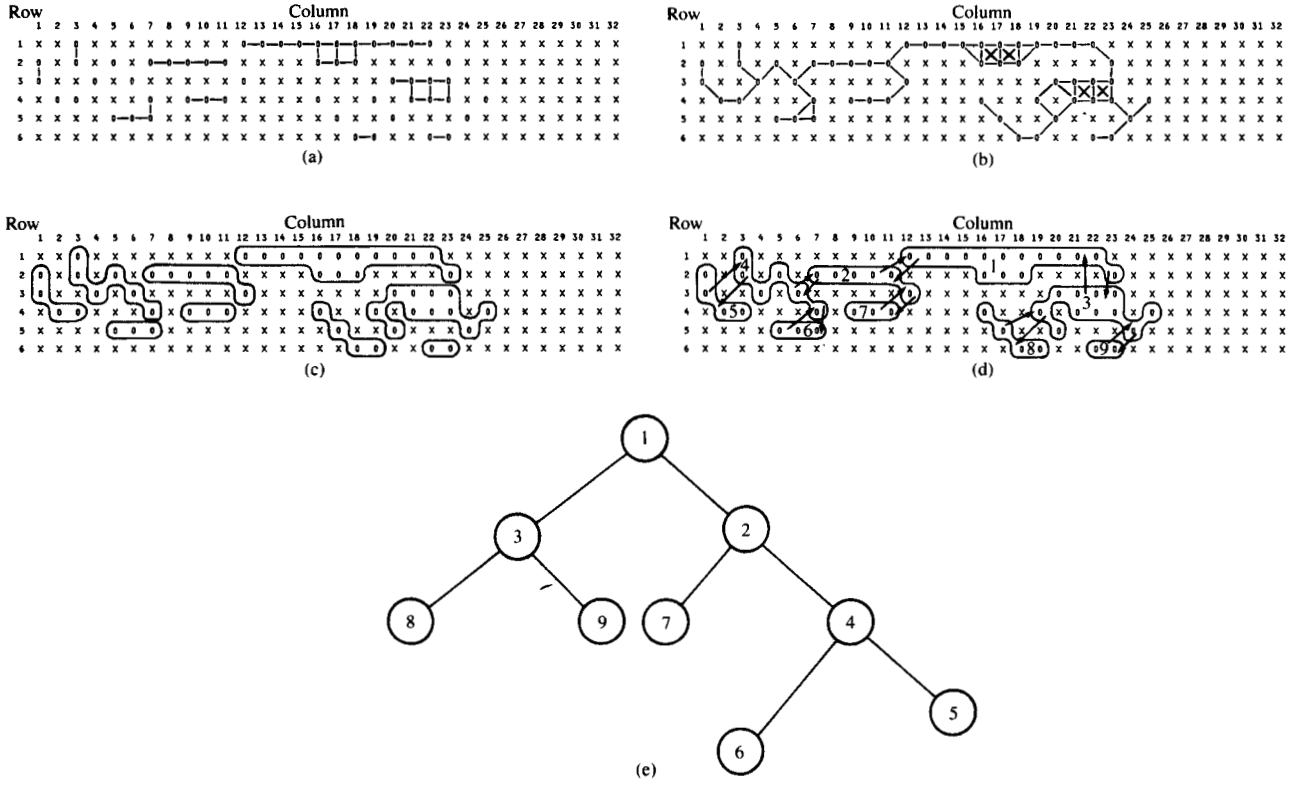

Figure 7 (a)–(e) Defect connectivity tracing illustrated (see text). Forward chaining is depicted by an up arrow; backward chaining by a down arrow.

Consider Fig. 9 as representing the cross section of a panel with criteria constraints defined as "... no more than four defects allowable within any area four cells by four cells square." In another application this could be interpreted as the bounds of an object.

It can be seen from the left-hand side of Fig. 9 that the existing pivots (defects) P1, P2, P3, P4, and P5 do not trap the error in the panel, since the defective area is bounded by Row 2 Column 4 (not a valid corner pivot), Row 2 Column 7 and Row 5 Column 4, Row 5 Column 7.

To circumvent this problem, therefore, a set of what are called *imaginary pivots* must be constructed. These are detailed in the figure and are created as normal pivot tables in the system, with a flag setting to indicate their unique status (see Table 3 for outline of pivot table format).

The imaginary pivot tables are created because there is no guarantee that any defective area will be aligned with a defective cell as its corner pivot. Therefore, a dummy pivot is constructed around a defect which does not exist at that coordinate on the panel. This dummy or imaginary pivot is determined by the intersecting point of two traces.

The first trace is backward from the first pivot and the second is upward from a second pivot which has a greater row number than the first, but a lesser column or cell number. The point or cell of intersection of these two traces is called the *imaginary pivot*. If the length of either trace is greater than the dimension of the defective area specified in the criteria, the imaginary pivot need not be constructed.

It can also be seen from Fig. 9 that duplication of imaginary pivot entries may take place (I1, 3/5 is an imaginary pivot constructed from the traces of P1 and P3 or P5). To avoid unnecessary duplication and consequently a longer processing time and additional storage allocation, an imaginary pivot directory is set up in the system. This directory holds the coordinates of all the actual and imaginary pivots and is updated as necessary, as the panel is scanned row by row.

Consider an extension of the right-hand side of Fig. 9. This would represent the worst possible case of a defective cell pattern with regard to pivot and imaginary pivot table allocation. Assume, for the moment, that all allowable defects (e.g., 16) in any sub-area were arranged in this diagonal fashion. The number of pivot tables required would

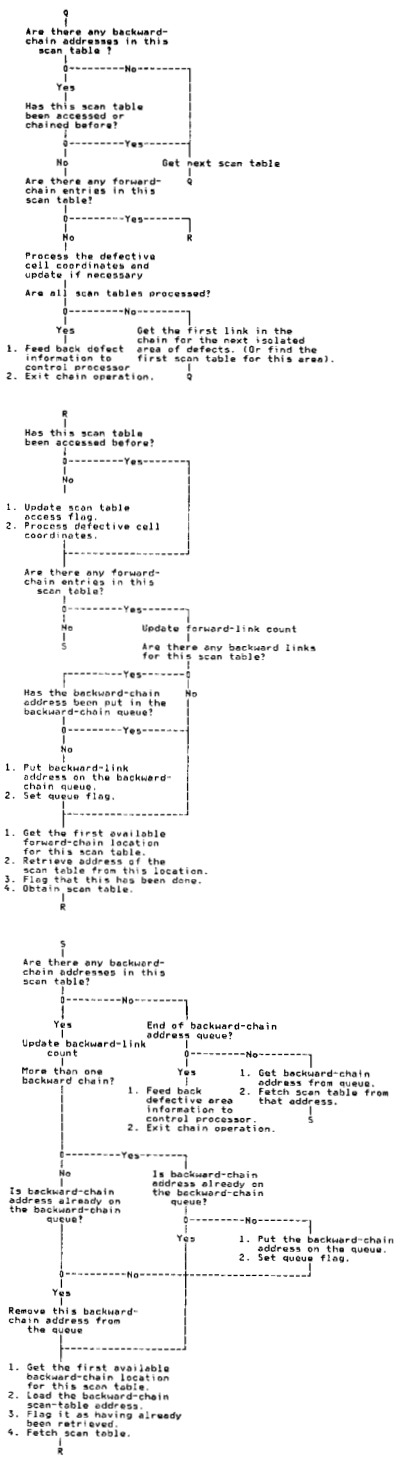

Figure 8 Forward- and backward-chaining algorithm.

be 16 (for the existing defects) + 15 + 14 + 13 + 12 + · · · + 1 (for the imaginary pivots).

Therefore, a total of 136 tables would be required. Thus, the memory allocation for the pivot operation is dependent on

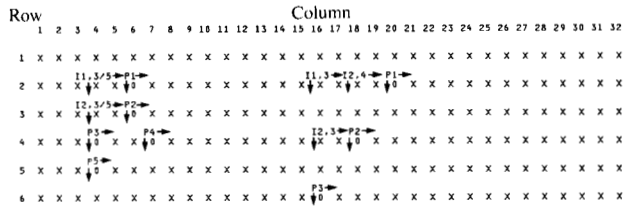

Figure 9 Imaginary pivot formation illustrated (see text).

the criteria. In summary, if no more than x defects are allowed within any given area, then the number of pivot tables required is $x + (x - 1) + (x - 2) + \cdots + 1$.

Note that the pivot operation routines are called immediately after a defective cell is encountered in Fig. 4. On exit from the pivot operation, assuming the criteria have not been violated, control is passed to the scan and cosmetic routines.

Complex applications

While many algorithms are available for complex patterns and applications, e.g., Kruse [1] or Danielsson [2], the algorithm described here has been used in detecting complex patterns. For example, Fig. 10 shows a pattern consisting of two separate areas, one within the other. It turns out that the technique is not only capable of detecting this type of pattern, but is also capable of determining its topology:

- 74 pixels within rows 2 to 12, and columns 2 to 19.
- 22 pixels within rows 5 to 8 and columns 7 to 14.

The algorithms do detect one area or object within the other, on a row-by-row basis, no matter what the alignment may be. While more rigorous experimentation is required for a generalized conclusion to this claim, indications are that the approach taken in this paper is very powerful in identifying complex shapes, yet it is simple and economical of space.

Conclusion

The algorithms described provide a solution for image analysis to the problem posed by the implementation of a row-by-row linear scanning application. This problem arises when the image to be analyzed cannot be viewed in its entirety but must be built up on a scan-by-scan basis, from which a *mirror image* is created in the processor memory and subsequently analyzed, an obviously time-consuming and costly method (in terms of storage capacity, a variable depending on the size of the image). Additional timing constraints (imposed by the very nature of the application, e.g., real time) may render the method undesirable if not impossible.

In these instances, a method which analyzes the image data on a row-by-row basis is essential. This paper has

described how this analysis is carried out. Algorithms are outlined which are capable of detecting the topology of solitary pixels as well as patterns of connected pixels in an image of any shape or size. The image can be reconstructed and analyzed dynamically without having the entire image presented for analysis at one time. The algorithms have been implemented and have been in use for several years for practical as well as experimental purposes. It is believed that the concepts presented, as well as the algorithms themselves, can be adapted and expanded to cover a wide range of image processing problems requiring economical solutions.

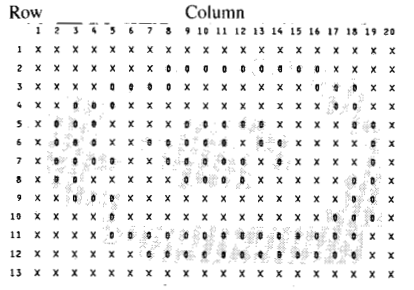

Figure 10 Example of an *object within an object* properly identified by the implemented technique (see text).

References

1. B. Kruse, "A Fast Stack-Based Algorithm for Region Extraction in Binary and Nonbinary Images," *Proceedings of EUSIPCO Conference, August 1980*, M. Kunt and F. de Coulon, Eds., North-Holland Publishing Co., Amsterdam, 1980, pp. 169–173.
2. Per-Erik Danielsson, "An Improved Segmentation and Coding Algorithm for Binary and Nonbinary Images," *IBM J. Res. Develop.* **26,** 698–707 (1982).

Received August 12, 1982; revised April 7, 1983

David McAuley *IBM United Kingdom, P.O. Box 30, Greenock, Renfrewshire PA 16 OAH, Scotland.* Mr. McAuley is a senior associate test design engineer working on automatic image inspection and analysis systems and disk file systems. He joined IBM United Kingdom in Greenock in 1978 after graduation from college and has worked on various test equipments, logic testers, and software support systems. In 1980, he developed the software system architecture and design for the plasma display panel automatic tester scanner and was subsequently assigned to work in Kingston, New York, for a year. Mr. McAuley received a B.Sc. in Computer Science from Paisley College of Technology, Paisley, Scotland, in 1978. He received an IBM special contributions award in 1979 for work on a keyboard logic tester.