The Enhancement of Computer Classifications by Logical Smoothing

Frederick E. Townsend
CAI a Division of RECON/OPTICAL, Inc., Barrington, IL 60010

ABSTRACT: The classification techniques by which features are discriminated in multispectral remote sensing data produce images with a salt-and-pepper appearance. This 'noise' can be removed with a logical smoothing operator, but undesirable loss of information may result. Better results are obtained by constraining the logical smoothing operator, by the addition of a connectivity rule, so as to act on the elementary regions in images. The behavior of the unconstrained and constrained operators are presented. An algorithm is presented by which the constrained operator is applied iteratively until all possible changes are effected in an image, while limiting input/output (I/O) to a single read/write of even large images which cannot be held in a computer's main memory. Process time approximates that required by one iteration with the unconstrained operator.

INTRODUCTION

SEGMENTATION is the general term defining the decision-making or pattern recognition process whose objective is to partition an image space (Figure 1) into its parts, regions, or meaningful entities. It is these meaningful entities, regions, or features which are typically the items of interest in remote sensor images. Classification is a pattern recognition process which attempts to partition, or segment, a two-dimensional feature space into regions representing different classes. The objective of classification is to associate each pixel (picture element) with a feature class by some unique discrimination function. The multispectral approach, commonly used for the analysis of remote sensing data, attempts to discriminate among types of materials on the basis of spectral signatures assigned to each material type of interest. However, classification (Figure 2) does not, in general, produce homogeneous regions. Rather, it produces images with a salt-and-pepper appearance; i.e., features which are depicted by one or few connected pixels which do not

FIG. 1. Aerial photograph of scene to be classified and smoothed.

FIG. 2. 10-class computer classification of the scene in Figure 1. (241 rows by 226 columns)

PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING,
Vol. 52, No. 2, February 1986, pp. 213–221.

0099-1112/86/5202-213$02.25/0

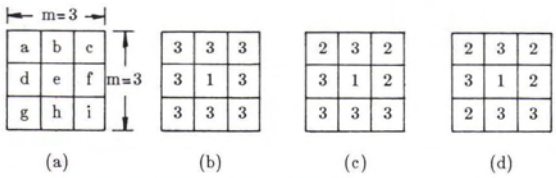


FIG. 3. Smoothing neighborhood and majority cases.

correlate with the scene. (see Appendix for a discussion of the theory of connectedness in digital images, as developed by Rosenfeld (1970)). This 'noise' can stem from several sources (e.g., nonuniform response of the sensor, processing errors, the mapping unit or pixel size, and the classifier), the specific source and degradation function(s) being unknown.

Before a description or other use of a classified image is attempted, it may be necessary to process it by some other segmentation method(s) which reduces noise and produces regions more representative of the features of interest. Enhancement is the term applied to such a process, which has as its objective the improvement of an image so as to make it more suitable for further processing, e.g., for feature mapping. One technique for reducing noise and thereby enhancing classifications is smoothing. But this technique can produce undesirable results, the character of which is presented in this paper. A two-decision rule smoothing operator is presented which obviates these undesirable traits of smoothing; and an algorithm is presented by which this operator is applied in a simulated parallel process while performing a single read/write of an image.

IMAGE ENHANCEMENT BY SMOOTHING

Two segmentation methods commonly used to combat noise in digital images are spatial smoothing and logical smoothing. The first of these methods is the technique whereby the value of a pixel is replaced by the average value of pixels in its immediate neighborhood. The method may be appropriate for quantized but unclassified images. The method is not appropriate for classified images because the values of pixels in classified images are arbitrary and intended merely to differentiate the pixels of feature classes. Averaging may produce pixels with values not previously present in an image and, therefore, not representative of defined classes; and the averaging pixels in a neighborhood may result in a pixel value represented in the neighborhood. Logical smoothing does not suffer these shortcomings. With this method each pixel and its neighbors are examined and a decision rule(s) applied to determine if a pixel is to retain its original value or is to be changed to the value of one of its neighbors.

The smoothing neighborhood is defined by a window dimensioned $m \times m$ and centered on pixels as shown in Figure 3a. The minimum dimensions of the window are 3×3. Logical smoothing is based on treating the pixels within the window as Boolean or logical variables; and the smoothed image function at the pixel in the center of the window is defined by a Boolean function of these variables. It may, for example, be specified that the central pixel is to be changed if, and only if, all of its neighbors are of a like class (value); that is, only if an elementary region (a region consisting of a single pixel) is contained, as shown in Figure 3b. The specification can be relaxed to allow noncontained pixels (pixels of nonelementary regions) to be changed: the central pixel is to be changed if, and only if, at least five (i.e., a majority) of the pixels in a neighborhood are of a like class, as shown in Figure 3c. The 5-majority rule is the limit to which the specification can be relaxed without introducing additional decision rules. As an example, the central pixel might be assigned to a class which is a simple majority (i.e., four) of the eight neighbors of the central pixel. A decision rule is then required to resolve those instances when ties occur, e.g., if the central pixel is a member of one of the simple majorities, it retains its class value. Other rules can be devised.

Because logical smoothing can be implemented in many ways (e.g., Duda and Hart (1973), Kimerling (1976), design specifications are required which optimize the concerns of an algorithm which is to be used to enhance classified images:

- Because smoothing tends to remove detail or cause information to be lost, the first objective is to maximize enhancement while minimizing the loss of detail or information.
- Because a smoothing operator can be applied iteratively to an image, a stopping criteria is needed. In order to satisfy the first specification, the decision rule should be applied as long as enhancement is proceeding and stop when further application of the decision rule fails to produce enhancement.
- Because input/output (I/O) between main memory and mass storage is slow relative to processing, the algorithm should minimize the data transfer function—large images (e.g., Landsat scenes) cannot be held in a computer's main memory; and this implies a significant amount of I/O if intermediate results are to be stored and input for successive application of decision rules.

IMAGE CHARACTERISTICS

Having defined some decision rules, it is well to enquire as to the results to be realized from their application. But first it is necessary to establish a basis for evaluating those results in light of the first design specification. Because the objective is to remove noise, some minimal specification for noise is required so that the effectiveness of its removal may be judged. This need not, and cannot, be an absolute specification because the degradation function(s) is unknown, but need only advance the objective, which is enhancement. Any region in a

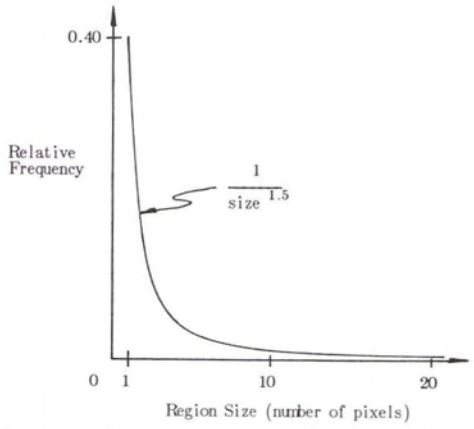


FIG. 4. Approximate distribution of region sizes in computer classified images.

classified image which is smaller than the smallest feature of interest might be considered to be noise; but the smoothing operator does not operate on region entities but pixels individually. Unless the interest is in identifying features consisting of single pixels, which would preclude smoothing, the interest must be, at least, in eliminating these elementary regions.

If most of the elementary regions were to be eliminated, by merging, would this constitute an enhancement sufficient to satisfy the first design specification? It has been found empirically (Nishikawa *et al.*, 1965) that the number of regions, in television pictures, of constant gray level that have area N is roughly proportional to $1/N^4$. It is expected that this proportion would hold for multispectral scanner images and quantized photographic images. But because classification segments images (to a degree), the proportion $1/N^{1.5}$ is a more reasonable upper limit on the approximation for classified images. The distribution of region sizes for this proportion is shown in Figure 4. The true proportion for a particular image is dependent on the quality of the classification, the number of objects in a scene, and the designated number of classes. Whatever the true proportion may be for a particular image, it is apparent that, by eliminating the elementary regions, the number of regions can be greatly reduced and an enhanced image achieved without a significant loss of information.

UNCONSTRAINED LOGICAL SMOOTHING OPERATOR

The smoothing decision rules and the results obtained from their application to classified images are now evaluated. Application of the containment decision rule eliminates only elementary regions. But this condition occurs in only a relatively small proportion of observed cases; and the greater the number of classes, the less likely the chance of occurrence. This decision rule does not satisfy the first design specification because a majority of the elementary regions will be unaffected. Similar observations apply in the case of the tie breaking decision rule. Both have inherent stopping criteria because their repeated application will have no affect. The containment and tie breaking decision rules may be considered to be special cases of the majority decision rule. The 5-majority decision rule contains no stopping criteria and its application produces less predictable results, which are now presented.

In applying the 5-majority rule there are two processing choices, sequential or parallel. In the discussion that follows the order of processing is assumed to be row by row in the sequence of digitization or data acquisition. In parallel processing, local operations are performed independently, that is, the variables are always the original image pixel values. New values are stored in another image, but are not used until the operation has been performed for every pixel, when they become the variables for the next operation or iteration. Because this is equivalent to operating on all pixels simultaneously, the operation can be thought of as being performed in parallel. With sequential processing the intermediate results are not stored in another image; but, rather, as soon as a pixel is processed, its new value is used in processing succeeding pixels which have it as a neighbor. It has been argued (Rosenfeld and Pfaltz, 1966) that parallel and sequential processing are mathematically equivalent, and that any picture transformation that can be accomplished by a series of parallel local operations can also be accomplished by a series of sequential local operations and conversely. It will be seen that these two processing methods are not equivalent when the logical smoothing operator is applied to classified images. And, whereas the order of processing is immaterial when operations are performed in parallel, the results from sequential processing are dependent on the order of processing.

The 5-majority rule was applied to the test image of Figure 5, which is 'clean' except for a small cluster of elementary regions contained by a region of class-7. The other regions are 4-connected in a binary figure/ground relationship. The results after a single iteration are as shown in Figures 6a and 6b for sequential and parallel processing, respectively. There are a number of observations that can be made about these results. The most obvious of these are that, having started with a clean image, several of the non-elementary regions were eliminated; and clearly, the results from parallel and sequential processing are not equivalent. When processed until no further changes are possible, the results are as shown in Figures 7a and 7b; and these also are not equivalent. The 5-majority rule clearly fails to satisfy the first two design specifications. It fails to satisfy the first

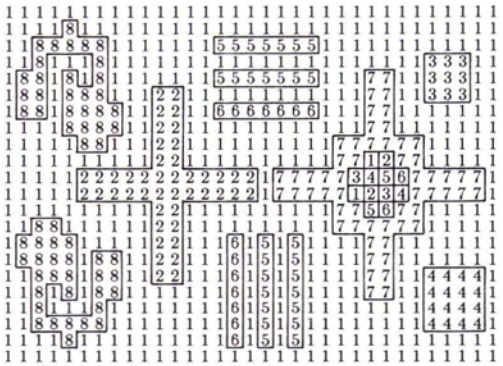
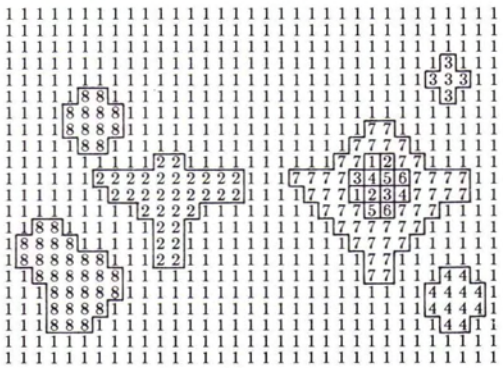
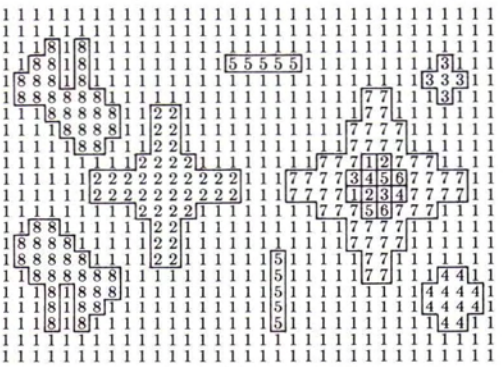


FIG. 5. Test pattern for evaluation of logical smoothing operators.

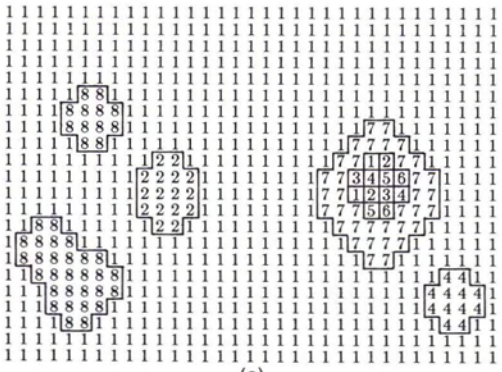


(a)

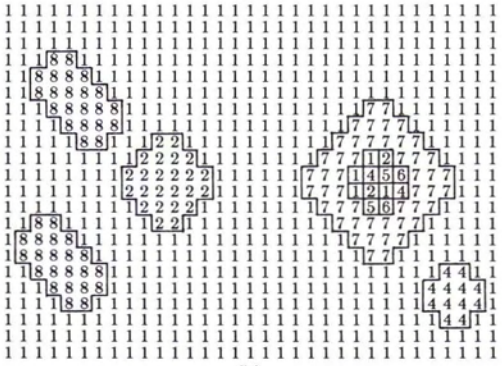


(b)

FIG. 6.(a) Test pattern after one iteration of logical smoothing with sequential processing. (b) Test pattern after one iteration of logical smoothing with parallel processing.



(a)



(b)

FIG. 7.(a) Test pattern after transformation until no changes effected by application of smoothing operator — sequential processing. (b) Test pattern after transformation until no changes effected by application of smoothing operator — parallel processing.

specification because it causes a significant loss of information (regions, features). It fails to satisfy the second specification because, having started with an image which could not be enhanced, it failed to stop immediately.

One characteristic of the operator is that it will affect only regions or parts of regions with a least dimension less than or equal to that of the window; and it is determined that the potential for information loss is dependent on the size of the window. Therefore, the minimum (3 × 3) window should be used in order to inhibit information loss. Another characteristic of the operator is that it tends to effect change until region borders are reduced to a series of horizontal and vertical straight edges and 45 degree diagonals—observance of this tendency is somewhat masked in noisy images. Neither of these characteristics of the majority rule argue for it adoption as an enhancement operator, as it obviously does much more than remove 'noise.' Even if any region of a size less than or equal to that of the window is defined to be noise, this single decision rule would be unacceptable because of its effects on larger regions.

It can be seen from the results obtained from one iteration that the results to be achieved with sequential processing are dependent on the order of processing. Feature components first encountered in processing are smoothed more than those encountered later. This is vividly illustrated by the two class-8 regions which are identical in configuration but rotated 180 degrees with respect to each other and the order of processing. Parallel processing of

these regions produces regions which are still identical in configuration but rotated 180 degrees. Sequential processing produces significantly different results, which are dependent on the order of processing and the relative placement of regions in an image or, equivalently, the organization of features in the object space. Clearly this is not a suitable processing method for the general case.

There is considerable variation in the number of iterations required to reduce a region to a state where no changes result from further application of the smoothing operator. Considering the parallel processing case only, the number of iterations required to transform the regions of Figure 5 into those of Figure 7b are as follows: the cluster of elementary regions, zero iterations; class-2 region, two iterations; class-3 region, three iterations, class-4 region, one iteration; class-5 regions, two iterations; class-6 regions, one iteration; class-7 region, two iterations; and class-8 region, nine iterations. Given the variability in processing, how would a stopping criteria be specified and enforced? Suppose that a case were to be made for a single iteration with parallel processing, which would limit the adverse effects of the majority decision rule. This can still result in the loss of linear features, which are often the features of interest in remote sensing records, and it can result in the creation of regions, as can be seen in the case of the class-5 regions in Figure 6b.

From the preceding examples, and understanding of the behavior of the 5-majority logical smoothing operator has been gained. The results from sequential processing have shown the method to be unsuitable for the purpose of enhancement. The remaining discussion is restricted to the parallel processing method.

It is noted that the examples presented are not representative of the cases found in classified images, as represented in Figure 2. Figure 5 presents only 'clean' contained regions, except for the one cluster of elementary regions, and does not contain the noise normally present in classifications. The result after three iterations of Figure 2 is shown in Figure 8. Eleven, six, and three percent of the pixels were changed in the three successive iterations.

From what has been presented, the results to be achieved from transforming classified images with the 5-majority logical smoothing decision rule can be characterized as being dependent on the topological properties of the source image; but these need not be the properties of interest. Some of the topological properties which influence the results of the smoothing transformation are the number of pixels in regions; the number of connected regions, which is dependent in part on the number of classes represented; the presence of contained regions; and the spatial association of pixels in regions. These properties are useful for describing the spatial distribution of feature classes obtained by classification, but have been shown to provide an insufficient basis for the enhancement of classified images by the single decision rule smoothing operator.

FIG. 8. Computer classification after three iterations with the majority rule smoothing operator — parallel processing.

CONSTRAINED LOGICAL SMOOTHING OPERATOR

The preceding examination of the behavior of logical smoothing operators suggests that their application to classified images is most appropriate if restricted so as to preserve as much as possible the information obtained by classification. This can be done by preventing the operator from acting on connected sets of pixels, i.e., regions. This can be accomplished by introducing a second or, more properly, a first decision rule: if the central pixel of the operator window is connected to another pixel in its neighborhood, do not apply the second (majority) decision rule. With this restriction on the operator, it can only modify elementary regions. In practice, the connectivity test need not be applied to unclassified pixels. Because these pixels contain no information, an attempt can be made to assign them to a class by application of the majority decision rule.

Does this modified operator satisfy the first two design specifications? Yes it does, because, while removing most noise (elementary regions and unclassified pixels), most of the information obtained by classification will be retained. Recall from Figure 4 that a large proportion of the regions in classified images are elementary regions. But these may represent a relatively small proportion of the pixels in an image (on the order of perhaps 15 percent or less). Also, a stopping criteria derives from the new

decision rule: if no changes are made during the processing of an intermediate image (or image row), stop processing. While a maximum number of changes is ensured by processing to this extent, it may appear that such a stopping criteria would result in an excessive amount of I/O and processor time. This need not be the case, as is shown in the following sections.

CONSTRAINED LOGICAL SMOOTHING ALGORITHM

An algorithm which uses the two-decision rule operator is now presented. It is noted as a preface to the algorithm that, in implementing the connectivity test, it can be either a 4- or 8-connectivity test. The results to be realized will differ in that fewer iterations will usually be performed when the 8-connectivity test is used, and less smoothing will result.

It was shown that, of the two processing methods, parallel processing was most suitable. This method requires storage for intermediate results. It has been found that most classified images will smooth to completion with less than a dozen iterations. The algorithm to be presented ensures that all intermediate results are maintained in main memory, and only the original image and the final results of processing are maintained on mass storage. The algorithm requires but a single reading and writing of an image from/to mass storage, which is an optimization of the third design specification. It also turns out that the algorithm for applying the two-decision rule operator incurs approximately the same processing costs as one iteration of the majority decision rule alone.

Because memory space will always be needed for two copies of an image, assign two one-dimensional arrays, denoted *O* (original) and *I* (intermediate), for their storage. Let the dimensions of these arrays be 14 times the number of columns in the largest image to be processed. This will allow for up to 12 iterations and space for the leading and trailing rows of the moving neighborhood window. (The leading row contains the elements *a*, *b*, and *c* of Figure 3a. The trailing row contains the elements *g*, *h*, and *i* of the same figure.) When processing begins, the rows of data read from mass storage are stored in the *O* array. The intermediate results from the first of the several serial transformations are stored in the *I* array. For the second iteration, the original and intermediate designations of these arrays are reversed. That is, the storage allocated to the original image is allocated for the storage of the results of the next iteration, and the results from the completed transformation are designated to be the original image. The roles of the two arrays are reversed after each iteration.

Because entire images are not maintained in main memory, a procedure is needed whereby parallel processing is preserved while performing several local image transformations. Data are read from mass storage in the order from the first through the *n*th row. Processing is in the reverse order. This may be thought of as a queue in which rows enter at the rear and are removed after they reach the front position. The *O* and *I* arrays are used for the storage of these queues. The storage location of the first pixel in each row is computed by the following hash function:

Array index = mod (image row number, maxrows) × row length + 1

where maxrows = queue array dimension divided by row length and

mod () denotes the remainder when row number is divided by maxrows.

Processing is initiated by loading the first three rows of data into the *O* queue. This queue, referenced to the source image, is shown in Figure 9a. Because the first (and last) row is not transformed, copy it to the front position of the *I* queue. The other positions in this queue are initially empty. The initial states of the queues are as represented in Figure 9b. In processing, the transformation function is applied to rows in the queue *between* the rear and front. Therefore, initial processing will apply to row number 2 only; and its transform will be stored in the *I* queue. The state of the queues after initial processing is as shown in Figure 9c. Row number four is

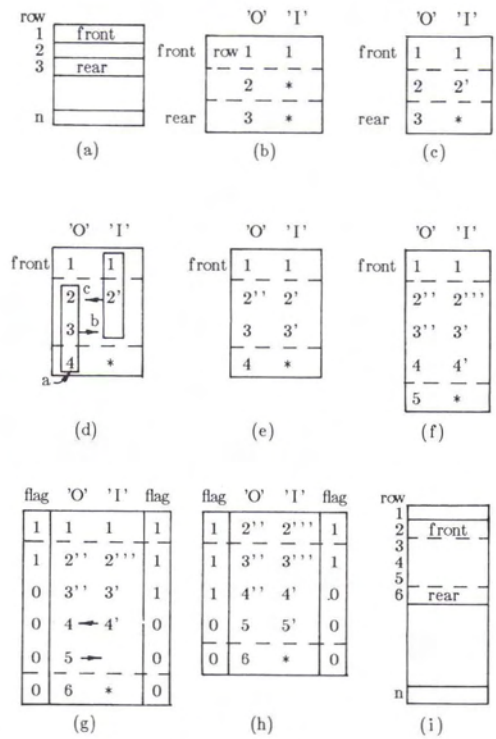


FIG.9. Queuing of image rows for smoothing.

then read from mass storage and inserted at the rear of the *O* queue, and the process is repeated. This time row number three is processed first and the transformed fow is stored in the *I* queue. Row number two (in the *I* queue) is then processed using the newly created intermediate results for row three; and the results of this transformation are stored in the *O* queue. The processing is represented in Figure 9d and the state of the queues after processing is as shown in Figure 9e. Repeat the process one more time and produce the queues shown in Figure 9f.

It should be apparent by now that some bookkeeping is requied in order to keep track of the storage locations of the 'original' and 'intermediate' images and of other processing requirements. This is done with a status table which is dimensioned to accomodate the largest expected value of 'maxrows.' After a row is processed, a table entry is made which indicates which of the storage arrays contains the most current results of transformation, i.e., the soure image for the *next* iteration on the row.

Because the repeated insertion of rows at the rear of a queue will result in the queue's eventual overflow, a decision rule is required which will result in the row at the front position of the queue being written to mass storage. When the front position row is written out, the remaining entries in the queue can be advanced one position. This does not necessitate the movement of the data in storage. The next row in a queue is merely designated to be the front row.

An image will not be uniformly clean when processing begins; and the number of iterations required to complete smoothing of a row will vary with local image conditions (a table of iteration counters may be used for recording the number of times a row is processed). The processing of a row is to be inhibited when it is known that processing will not affect any of the pixels in the row. This will be the case if no changes were made to a row and its neighbors when last processed. When this condition pertains to the row at position "front plus one," the front position row is written to mass storage and the queue advanced. A table of process flags is used to keep track of the processing state of each row. This table must also be dimensioned to accommodate at least 'maxrows' rows. The entries in the table are binary flags which are initialized to zero when rows are read into the queue. The flag is set to one if no changes are made to a row during an iteration. A row's flag is reset to (or left at) zero if processing results in a new class value being assigned to any of its pixels during an iteration.

The function of the processing flag is demonstrated by returning to the example in Figure 9. By the addition of processing flags to Figure 9f, and by insertion of the next row at the rear of the queue, the state of figure 9g is created. In establishing the flag values, it was assumed that no changes occurred in rows numbered two and three during their last processing. Say that the transformation of 4′ into 4″ results in no changes being made. The flag for this row is changed from zero to one. Row three (3″) is now to be processed. Because the processing flags for this row and its neighbors (4″ and 2″) are set to one, there is nothing to be gained by processing row 3″; and so it is simply copied to 3‴. Row number two (2‴) is now to be processed; and the conditions are the same as those existing for row three. In this instance, the front row is written to mass storage and row number two is advanced to the front queue position. The results after processing are as shown in figure 9h, and the queue of Figure 9a has been updated to that of Figure 9i.

A row is inserted at the rear of the queue each time the queue entries are processed. If the row at the front of the queue is not written to mass storage at the end of the process, the length of the queue will increase as shown in figures 9d through 9f. This cannot continue indefinitely because the length of the queue can never exceed 'maxrows'. If the estimation of the maximum number of iterations (queue dimension) is low, an overflow condition would occur at some point in the processing. There is a solution to this potential problem: if overflow would occur, write the row at the front of the queue out to mass storage each time the queue is processed. Smoothing may not be completed; but it can be completed by reexecuting the program and by using the initial result as the new 'original' image. In practice this is seldom necessary because few unconnected pixels will remain.

OPTIMIZED SMOOTHING ALGORITHM

The smoothing algorithm is now presented in a form which can be readily implemented in computer software. Begin processing by loading the first three rows of image data to the input queue; set labels in the status table indicating which array contains the input (original), and initialize iteration counters and process flags to zero. Table indices are computed by the following hash function:

table index = mod (image row number, maxrows) + 1

Because the first image row is not smoothed, copy it to the output queue and set process flag to one. Initialize image row counters:

FRONT = 1; EROW(end row) = FRONT + 1 = 2;
REAR = 3; SROW (start row) = REAR − 1 = 2;
QROWS(number of rows in queue) = 3;

(1) While SROW not equal to n (number of rows in image), R = SROW(where R is the next row to be processed).
Compute table and array indices for rows R, $R-1$, $R+1$ with hash function.
Process rows in queue *between* REAR and FRONT in bottom-up order.

(2) If the process flags for rows R, $R-1$, and $R+1$ re

are all set to 1, copy row R from input queue to output queue, and go to (5).

(3) While not at end of row R, apply decision rules of operator to pixels:
   (a) If pixel is unclassified, go to (e) (connectivity rule is not applied to unclassified pixels).
   (b) Else, if connected with pixel to the right, copy to the output queue and increment column. If unclassified go to (e).
   (c) If 4-connected with any of neighbor pixels copy to output queue; increment column and go to (a).
   (d) Else, if 8-connectedness is to be enforced, test the diagonal elements in the neighborhood. If connected, copy to the output queue; increment column and go to (a).
   (e) Perform neighborhood majority test. If 5 pixels of the same class not present, test for a simple majority of 4. If a majority (but not unclassified), write its class value to the output queue; increment column and go to (a). Otherwise, transfer unchanged to output queue; increment column and go to (a).

(4) Processing of a row completed. Increment iteration counter.

(5) If no changes made, set process flag for row R to 1, else set 0. For the row just processed, update the status table by reversing the input and output queue designations to apply the next time this row becomes row R. If R = EROW, go to (7).

(6) Move up one row in the queue: $R = R - 1$.
Compute table and array indices for the new R and its neighbors.
Go to (2).

(7) All rows in the queue between REAR and FRONT have been processed. If QROWS = MAXROWS, or if process flags for EROW and EROW + 1 are set to 1, write FRONT row to mass storage; and advance the queue:FRONT = FRONT + 1; EROW = EROW + 1; QROWS = QROWS − 1. Else, if EROW = $n-1$, and its process flag is set to 1, write EROW data to mass storage, and go to (9).

(8) If REAR = N, go to (1).
Else, read next row from mass storage:
SROW = SROW + 1; REAR = REAR + 1; QROWS = QROWS + 1
Go to (1).

(9) Write row n, unchanged, to mass storage, and *stop*.

The processing of the classification shown in Figure 2 with this algorithm resulted in the image shown in Figure 10. Nine percent of the pixels in the image were changed, and the maximum number of iterations on any row was 12. The results were from an implementation of the algorithm in the 'C' language, on a Digital Equipment Corporation 11/45 computer.

It was previously stated that processing with this algorithm, which performs several image transformations, approximates that of a single transformation with the majority decision rule alone. Generation of the Figure 10 image, in which both 4- and 8-connectivity were enforced, required 59 seconds of process time. One iteration with the majority rule required 56 seconds. The reasons for the processing efficiency are now enumerated. The I/O requirements are identical because each row is read and written only once. There are three contributors to processing efficiency, and two of these derive from the testing for connectivity. In the first instance, the connectivity tests require fewer operations than the majority test, which requires that all pixels in a neighborhood be counted. The second instance where connectivity testing improves efficiency is in the application of the test first to the adjacent pixel to the right of the central pixel of the operator window. By this test, runs of pixels in nonelementary regions are identified and exempted from testing by the majority rule. Third, smoothing iterations are not applied to rows of data when it is known that processing will not result in change. This is known to be the case when a row to be processed and the adjacent rows were not changed when last processed.

FIG. 10. Computer classification smoothed with two decision rule operator.

DISCUSSION

Computer classification techniques segment multispectral remote sensor records but produce images with many small regions which do not correlate with the scene. A two-decision rule logical smoothing operator for the enhancement of computer classifications has been presented which simplifies classifications (compare Figures 2 and 10). Also, an algorithm for implementing this operator in computer software has been presented. Others (e.g., Scarpace *et al.* (1981)) have shown that this algorithm can improve the accuracy of classifications, and this is as would be expected if 'noise' is removed from an image.

REFERENCES

Duda, R. O., and P. E. Hart, 1973. *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons.

Kimerling, Arthur J., 1976. *Theoretical and Practical Relationships Between Remote Sensing and Cartography*, Ph.D. dissertation, Madison: University of Wisconsin.

Nishikawa, S., R. J. Mass, and Mott-Smith, 1965. Area Properties of Television Pictures, *IEEE Transactions on Information Theory* IT-11, pp. 348–352.

Rosenfeld, A., 1970. Connectivity in Digital Pictures, *Journal ACM*, Vol. 17, No. 1, pp. 146–160.

Rosenfeld, A., and J. L. Pfaltz, 1966. Sequential Operations in Digital Picture Processing, *Journal ACM*, Vol. 13, No. 4, pp. 471–494.

Scarpace, F. L., *et al.*, 1981. Wetland Mapping from Digitized Aerial Photography, *Photogrammetric Engineering and Remote Sensing*, Vol. 47, No. 6, pp. 829–838.

APPENDIX

CONNECTIVITY IN DIGITAL IMAGES

A pixel has eight surrounding or neighbor pixels, and these are of two kinds: four horizontal and vertical neighbors, and four diagonal neighbors. If a pixel is connected (i.e., adjacent) to any of the eight surrounding pixels, it is 8-connected. If it is connected to pixels sharing a common edge, i.e., the four horizontal and vertical neighbors, it is 4-connected. A connected region R is defined to be any subset of an image in which the pixels are connected. The pixels in R are connected if (1) all pixels in R are of the same value; and (2) there exists a 4-path (or 8-path) having its first and last pixels in R, i.e., any two pixels in R are connected by a chain of pixels each of which is in R. The simple binary image shown below illustrates the definitions and the ambiguities that may arise.

0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0
0 0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0

Considering first the idea of 8-connectedness, it is seen that, because the pixels of the figure (denoted by 1's) sharing only a common vertex are connected, the figure is connected. However, the background (denoted by 0's) is also connected (the hole in the figure is not separated from the background which surrounds the figure); i.e., the diamond-like figure has no inside. If 4-connectedness is taken to be the definition, the figure is not connected; and neither, however, is the background. The figure is not connected but has an interior.

(Received 5 March 1983; accepted 31 May 1985; revised 14 November 1985)