Pattern Recognition Letters 10 (1989) 81–85
North-Holland

August 1989

An efficient algorithm for extrema detection in digital images

B.B. CHAUDHURI and B. Uma SHANKAR
Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India

Received 12 January 1989

Abstract: Computationally attractive serial algorithms for detecting extrema of 2-D discrete functions are considered for image processing applications. It is shown that the algorithms are essentially linear in K where K denotes the number of pels to be considered. Experimental results are presented on typical image sets. A parallel algorithm on a pyramid-like architecture is also discussed briefly.

Key words: Algorithms, extrema detection, texture analysis, image processing, fuzzy components.

1. Introduction

Detection of 2-D local maxima and minima is useful in many image processing problems. One of the application areas is texture analysis (Rosenfeld and Troy, 1970; Ledley, 1972; Rotolo, 1973; Michell et al. 1977) where the number of local gray level extrema per unit area, i.e., the extrema density and related statistics are used as texture properties. Recently, Rosenfeld (1984) introduced the concepts of fuzzy geometry and topological properties on image subsets. It can be shown that connected components and holes in the fuzzy sense are the connected components of these extrema. Similarly, morphological operations like propagation and shrinking may begin from the extremas of the image and lead to different shape manipulation and skeletonization.

The extreme components of an image are defined as follows. Let S denote the set of all pels of an image. Let R be a 8-connected component in S. If $\bar{R}$ is the complement of R in S, then the *border of* $\bar{R}$ *with* R is defined as the subset $B(\bar{R}, R)$ of $\bar{R}$, any element of which is an 8-neighbor of at least one element of R. Now, a component R in S is a *local maximum* or more generally, a *plateau* in S if the gray values of all elements in R are equal and this value is greater than the gray value of any element in $B(\bar{R}, R)$. Similarly, R is a *local minimum* or *valley* in S if the gray values of pels in R are equal but less than the gray value of any element in $B(\bar{R}, R)$. Typical extreme pels in 2-D are shown in Figure 1.

This correspondence is concerned with the algorithm for detecting the extrema. A straightforward approach may be the use of level sets in an image. The t-level set $L(t)$ consists of pels whose gray levels are greater than or equal to t. Let t_1 and t_2 be consecutive gray levels in the image where $t_2 > t_1$. Consider a connected component C of $L(t_1)$. Now, C is a maximum or plateau of gray level t_1 only if $C \cap L(t_2) = \emptyset$, i.e, pels of C completely vanish from the level set of gray level t_2, as explained in Figure 2.

This approach can be easily modified for minima detection. The main drawback of the algorithm is that component labeling is necessary at each level set and these components are to be compared with those in the previous level set. Hence a better approach is described below.

9	10	8	9	8	6	7	8	7	9	7	8
8	7	7	6	6	$\underline{5}$	$\underline{5}$	$\underline{5}$	11	7	7	7
8	8	8	8	6	6	6	6	9	7	8	7
9	8	10	9	10	6	7	7	8	8	9	7
8	9	9	8	7	8	7	9	8	8	8	8
9	8	9	7	$\underline{6}$	$\underline{6}$	$\underline{6}$	7	10	8	8	9
10	11	10	8	8	8	8	9	10	10	9	11
11	9	$\underline{7}$	8	$\underline{6}$	$\underline{6}$	$\underline{6}$	8	8	8	8	11
8	11	$\underline{7}$	8	7	8	9	8	9	9	10	7
9	8	$\underline{7}$	8	9	10	10	9	$\underline{7}$	10	10	6
9	9	8	9	9	11	11	8	9	8	6	5
$\underline{8}$	9	10	9	10	9	11	9	6	5	$\underline{4}$	$\underline{4}$

Minima

9	$\underline{10}$	8	$\underline{9}$	8	6	7	8	7	9	7	$\underline{8}$
8	7	7	6	6	5	5	5	$\underline{11}$	7	7	7
8	8	8	8	6	6	6	6	9	7	8	7
9	8	$\underline{10}$	9	$\underline{10}$	6	7	7	8	8	$\underline{9}$	7
8	9	9	8	7	8	7	9	8	8	8	8
9	8	9	7	6	6	6	7	$\underline{10}$	8	8	9
10	$\underline{11}$	10	8	8	8	8	9	$\underline{10}$	$\underline{10}$	9	$\underline{11}$
$\underline{11}$	9	7	8	6	6	6	8	8	8	8	$\underline{11}$
8	$\underline{11}$	7	8	7	8	9	8	9	9	10	7
9	8	7	8	9	10	10	9	7	10	10	6
9	9	8	9	9	$\underline{11}$	$\underline{11}$	8	9	8	6	5
8	9	$\underline{10}$	9	10	9	$\underline{11}$	9	6	5	4	4

Maxima

Figure 1. Typical extreme pels. Maxima and minima are underlined.

2. The algorithms

First we describe a method that is efficient if extrema at one gray level are required. It is obvious that pels with gray level t_1 are the only candidates for extrema at gray level t_1. Consider the case of only maxima. If any pel $P \in L(t_1)$ has a 8-neighbor in S whose gray level exceeds t_1, then the connected component to which P belongs cannot be a maximum. In Figure 2 the pels A,B,E,F are possible candidates of maxima at gray level t_1. Of them, B

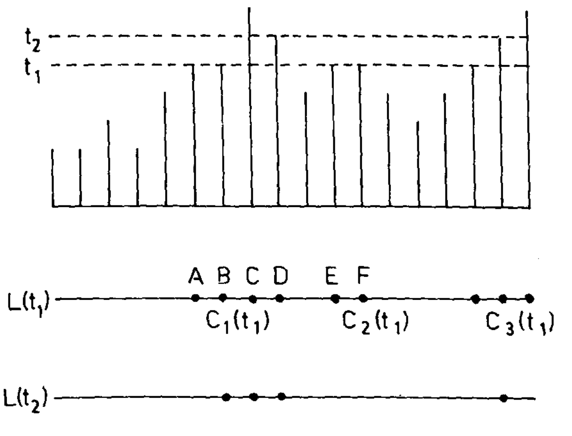

Figure 2. Maxima detection in one dimension. $C_2(t_1)$ is a plateau since $C_2(t) \cap L(t_2) = \emptyset$.

has a neighbor C whose gray level exceeds t_1 and hence the connected component consisting of pels A and B cannot be a maximum. The true maxima pels are E and F.

Implementation of this idea in 2-D is explained by Figure 3. The pels having gray value t_1 and no higher gray value neighbor are labeled 1, those having gray value less than t_1 are labeled 0, those having gray value greater than t_1 are left blank while those having gray value 1 but also an 8-neighbor whose gray value is greater than t_1 are labeled 2. Next, pels with label 2 are propagated into the pels with label 1. The remaining pels with label 1 are

0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	
0	0	0	0	2	1	0	0
				0	1	1	0
				2	1	1	0
				0	0	0	0

a

0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	
0	0	0	0	2	2	0	0
				0	2	2	0
				2	1	1	0
				0	0	0	0

b

Figure 3. Detection of 2-D maxima by thresholding. (a) Before propagation of label 2 into label 1. (b) After propagation of label 2. The remaining pels with label 1 are maxima.

maximal pels at level t_1. The approach can be modified to find minima as well as minima and maxima simultaneously. Let this algorithm be called EXTREMA-0.

If the image contains many gray levels and if extrema should be found at all levels, then this algorithm is not very efficient. However, this basic concept can be extended to find a modified algorithm as follows.

In this algorithm we consider the detection of maxima and minima simultaneously and indistinguishably. The idea is to find pels that are not extrema, find connected components of same gray level and delete them. The remaining pels are extreme pels. The algorithm, called EXTREMA-1 can be easily modified to find maxima or minima separately.

Here the pels are considered in a row scanning manner. First, each *candidate pel* (CAP), is labeled as either (a) an extremum or (b) neither a minimum or maximum in its 3×3 neighborhood. In case (b) the CAP is called a *collapsible pel* (CP). Examples of these cases are shown below.

$$\begin{array}{ccc} 2 & 4 & 8 \\ 3 & \underline{8} & 7 \\ 6 & 6 & 6 \end{array} \qquad \begin{array}{ccc} 4 & 2 & 3 \\ 6 & \underline{2} & 3 \\ 5 & 4 & 2 \end{array}$$

Maximum Minimum

$$\begin{array}{ccc} 2 & 4 & 8 \\ 4 & \underline{8} & 7 \\ 7 & 7 & 9 \end{array}$$

Collapsible pel (CP)

If a CP is encountered, its connected component of pels of the same gray level is found and all pels belonging to this component are collapsed. The result is stored in a matrix whose dimensions are the same as that of the original image matrix. The collapsed pels are labeled by, say, -1 in the corresponding positions of the *result matrix* while the extrema are labeled by, say, 1. Initially the result matrix is empty, i.e., labeled as 0.

In actual implementation, for each CAP, the corresponding position in the result matrix is examined if the label there is -1, i.e., if it is already collapsed by any previous CP. In case the CAP is already collapsed, no test of maximum or minimum is made. In case the CAP is not collapsed and it is found to be a CP, its corresponding connected component is found and collapsed. Otherwise it is considered as an extremum. All results corresponding to the current CAP are stored in the result matrix.

To work on CAP's of the first and last row as well as columns, two rows and columns with value, say, 0 may be appended before and after the first and last row as well as columns, respectively. In our algorithm we used a stack for component growing and collapsing. In brief, the algorithm has the following steps.

Algorithm EXTREMA-1

Step 1: Initialize: Current CAP = first pel in the first row of the image matrix.

Step 2: If the current CAP is collapsed go to Step 5. Else, continue.

Step 3: Check the 8-neighborhood to find if the CAP is a CP or an extremum. A CAP is a CP if at least one neighbor has a higher gray level and another neighbor has a lower gray level than that of the CAP. Else, it is an extremum. If an extremum, enter the corresponding labels in the result file and go to Step 5. Else continue.

Step 4: Create the connected component of the CP using the subroutine GROW-STACK. By this subroutine, the result file is also filled with label -1 at corresponding pels of the component.

Step 5: If the last pel in the image is encountered, Stop. Else move to the next pel in raster scan mode and consider it as the current CAP. Go to Step 2.

Subroutine GROW-STACK

Parameters: Co-ordinates of the CP which has to grow; gray level of the CP.

Step 1: Check all 8-neighborhoods for connectivity with same gray level and collapse the connected pels, i.e., label the corresponding positions in the result file by -1. Store the co-ordinates of the collapsed pels in a stack.

Step 2: If the stack is empty, stop. Else, take the topmost entry in the stack and go to Step 1.

The result file contains the labels of the extrema. An alternative form of result file may be created which initially contains a copy of the image file. The gray level at the collapsed pels are changed by the label, say, -1 and all other pel gray levels are left intact. Then all pels with non-negative gray levels in the result file are the extrema.

If we want maxima and minima distinguished, then in the result file we have to lable maxima by, say, 1 and minima by, say, 2. Step 3 and Step 5 of algorithm EXTREMA-1 should be modified as follows:

Step 3: Check the 8-neighborhood. If the CAP gray level is not less than that of any of its 8-neighborhoods, then the CAP is called a maximum and labeled 1. On the other hand, if the CAP gray level is less than that of at least one neighbor but not greater than that of any neighbor, then it is called a minimum and labeled 2. The condition of CAP to become a CP is already given in EXTREMA-1. The labels are entered in the result file. If a minimum or maximum is encountered, go to Step 5. If a CP is encountered, go to Step 4 of algorithm EXTREMA-1.
Step 5(a): If the last pel in the image is encountered, go to Step 5(b). Else move to the next pel and go to Step 2.
Step 5(b): Generate the connected component of same gray level corresponding to each minimum pel. Positions corresponding to the pels of this component are labeled 2 in the result file.
Step 6: Stop.

For Step 5(b) a growing algorithm like GROW-STACK may be used. The reason for introducing Step 5(b) is the fact that in Step 3 if all gray levels of pels in a 3×3 neighborhood are equal, then the CAP is called a maximum although it may be a minimum as well. Step 5(b) changes the states of these pels in the result file. This modified algorithm may be called EXTREMA-2.

3. Algorithm complexity

First, let us find the computer complexity of the algorithms. We consider the worst case performance on an $N \times N$ image for the evaluation and present typical results on a set of images in the next section.

For algorithm EXTREMA-1, Step 2 requires one comparison per pel. Hence N^2 comparisons are needed. If Step 3 is encountered, it may need at best $8+7$ comparisons to find if the CAP is an extremum or CP. If all CAP enter Step 3 then $(8+7)N^2$ comparisons may be needed. The total is now $16N^2$.

Let M be the total number of extreme pels. The number of collapsed pels is $N^2 - M$. To collapse the pels by GROW-STACK, at best eight comparisons are needed for each pel. In fact, except for the CP with which the subroutine GROW-STACK is entered, all collapsed pels can be detected by seven or less comparisons. However, the program becomes more complicated and we accept $8(N^2 - M)$ comparisons in the worst case. In addition, the stack is checked once for each pel. So, the total number of comparisons for collapsing the pels is at most $9(N^2 - M)$. The grand total of number of comparisons cannot exceed $16N^2 + 9(N^2 - M)$.

Next we consider the number of assignments required in the algorithm. The result array requires N^2 assignments of label -1 or 1. The status of 1 may need be changed to -1 and hence $N^2 - M$ assignments may be necessary. However, if the result array is initially a copy of the image array, $N^2 - M$

Table 1

	Number of				Number of comparisons/pel					Number of assignments/pel		
Image number	Max.	Min.	Total	Total pels in extrema	For detec.	For pel growing	Extra	Total without extra	Total with extra	Detec. & pel growing	Extra	Total with extra
1	249	239	488	722	3.804	7.414	1.841	11.218	13.059	6.226	0.573	6.799
2	275	235	510	742	4.288	7.370	1.866	11.658	13.524	5.991	0.597	6.588
3	268	234	502	751	4.906	7.350	1.949	12.255	14.205	5.574	0.672	6.246
4	279	277	556	794	4.140	7.255	1.870	11.396	13.266	6.038	0.571	6.609
5	282	273	555	763	4.695	7.323	1.842	12.018	13.860	5.651	0.548	6.200
6	285	286	571	758	4.416	7.334	1.828	11.750	13.578	5.882	0.527	6.409

Average no. of comparisons/pel = 13.582, Average no. of assignments/pel = 6.475, Average extrema = 12.95%, Average pels in extrema = 18.43%.

assignments instead of $N^2 + N^2 - M$ assignments are at best necessary. In the stack the two co-ordinates and the serial number of each collapsed pel should be stored which needs 3 assignments per pel. For retrieval, again, 3 assignments per pel are required. Then, the total number of assignments for the stack can be $6(N^2 - M)$. The grand total of assignments can be $N^2 + 7(N^2 - M)$ if the result file is not initially a copy of the image file and $7(N^2 - M)$ otherwise.

To separate the maxima and minima by algorithm EXTREMA-2, the additional number of comparisons needed for collapsing the false maxima is $9M_1$ where M_1 is the number of minimal pels. In addition, N^2 comparisons may be necessary to detect the minimal pels, so that they may be propagated. The number of extra assignments necessary when the result file is not initially a copy of the image file is $8M_1$ in which $6M_1$ is needed for the stack and $2M_1$ is needed for transfer between arrays. In addition, to distinguish between minima and maxima we need another array in which $N^2 - M$ assignments are at best required.

It may be noted that the algorithms are of linear complexity in N^2 with the constant of proportionality not exceeding 25.

4. Results and discussion

To test how well the algorithms perform in practical cases, 6 sets of image arrays were considered for extrema detection. The images belong to a single frame of a band of LANDSAT-III imagery that covers the area around the city of Calcutta. The image arrays consist of 64×64 pels each and they are chosen so that the extrema cover less than 20% of the total pels. According to the previous analysis, the number of comparisons required by this algorithm increases with the increase in the number of collapsed pels and since the collapsed pels cover 80% of the area, the results presented here are closer to the worst case performance. The number of comparisons and assignments necessary by the two algorithms are scaled to N^2 and presented in Table 1. In this table, the 'Extra' columns refer to the additional computation needed for separating the maxima and minima pels by algorithm EXTREMA-2. It is seen that the totals are less than the worst case analysis of the previous section. The programs of the algorithms developed by the authors in Fortran-77 may be available to the readers on proper correspondence. It should be noted that EXTREMA-2 can return with the connected components of minima. For maximal components, however, the results should pass through a component labeling algorithm.

The algorithms discussed above are essentially serial. However, it may not be difficult to translate them into parallel algorithms. An alternative idea is to find a parallel algorithm in a pyramidal architecture, where extrema in $n \times n$ blocks are hierarchically merged to $2n \times 2n$ blocks until a single block of $N \times N$ results. The detail of the algorithm will be communicated in a separate correspondence.

It is rather straightforward to extend the EXTREMA-1 and EXTREMA-2 algorithms to extrema detection in 3-D with voxel representation of the image. Similarly, it is possible to extend the pyramidal parallel algorithm outlined in the previous paragraph to 3-D image data also.

Acknowledgement

The authors wish to thank Prof. D. Dutta Majumder for his encouragement while Mrs. S. De Bhowmick, Mr. J. Gupta and S. Chakraborty for help rendered in preparing the manuscript.

References

Ledley, R.S. (1972). Texture problems in biomedical pattern recognition. *Proc. 1972 IEEE Conf. on Decision Control and 11th Symp. on Adaptive Processes*, New Orleans, LA, Dec. 1972.

Michell, O.R., C.R. Myers and W. Boyne (1977). A max-min measure for image texture analysis, *IEEE Trans. Comput.* 26, 408–114.

Rosenfeld, A. and E. Troy (1970). Visual texture analysis, *Tech. Rep. 70-116.* University of Maryland, College Park, MD. [Also in *Conf. record for Symp. on Feature Extraction and Selection in Pattern Recognition*, Argonne, IL. IEEE Publication 70C-51C, 115–124.]

Rosenfeld, A. (1984). The fuzzy geometry of image subsets. *Pattern Recognition Letters* 2, 311–317.

Rotolo, L.S., (1973). Automatic texture analysis for the diagnosis of pneumoconiosis, *26th ACEMB*, Minneapolis, MN, Sept. 30–Oct. 4, 1973, 32.