Efficient algorithm for image enhancement

B.B. Chaudhuri, M.Tech., Ph.D., Mem. I.E.E.E.

Indexing terms: Algorithms, Image processing

Abstract: The problem of smoothing, point, line and edge detection in image processing is considered. It has been shown that computation in each case can be speeded up considerably by looking at the redundancy. The methods are simple and can be implemented easily in software or hardware.

1 Introduction

Different spatial domain techniques have found wide application in image-enhancement problems because of their simplicity and ease of implementation in both software and hardware. Of these, neighbourhood averaging, or mean filtering, is used for smoothing and noise cleaning [1]. Other techniques of smoothing include median filter and standard-deviation-based mean filters [2–4]. For edge detection and enhancement, different kinds of thresholds are used. Detection of line, point, gradient and average as the projection onto one of the nine orthogonal complete basis vectors has also been proposed [5].

A 3 x 3 window is usually used in all the techniques. For smoothing using mean filters and detection of line, point and gradient, each of the nine positions in the window is weighted and the projection of the candidate pixel and its neighbours is calculated. In many cases, the weights are either ± 1 or 0, and hence addition is the main concern in the calculation. Since addition requires very little time in a computer or processor and the computer time complexity is of linear order $0(n)$, where n is the window size, little effort has been reported to reduce it. However, the operation on a 512 x 512 pixel picture makes the processing time reasonable. The number grows bigger if the task is to smooth and detect point, line and gradient successively, and it becomes respectable when the number of picture to be processed is large, as in fingerprint or script processing. Again, median filtering requires comparison instead of addition, and a straightforward comparison involves time complexity $0(n^2)$. For a 3 x 3 window, the number of operations is at least four times that of any of those stated above.

2 Procedure

It is, therefore, useful to find algorithms that reduce the number of operations needed to implement the above techniques. It is shown below that considerable savings can be achieved by looking at the redundancy for all the techniques.

The window is shown in Fig. 1 where 5 is the position to which the candidate pixel is aligned. For mean filter, the weight $w_5 = 0$ and $w_i = 1\ \forall\ i \neq 5$, needing seven additions at each pixel to implement it. But as shown in Fig. 2, addition of b and l can be shared in smoothing pixels f, g and h. Hence the average number of additions for each pixel is 1/3. Three such additions are needed for each pixel. For example, for candidate g, ak, bl and cm are necessary, where for convenience it is assumed that $x + y = xy$. Now, the addition of bl and cm can be shared by two candidates g and h, the average operations for each being 1/2. The redundancy in these four operations for each candidate, therefore, is $3 \times (1 - 1/3) + (1 - 1/2) = 2.5$, making a speed improvement of $7/(7 - 2.5) = 1.55$. The additional storage requirement is three; two for additions like bl, cm and one for $blcm$. The operation can be continued row or columnwise.

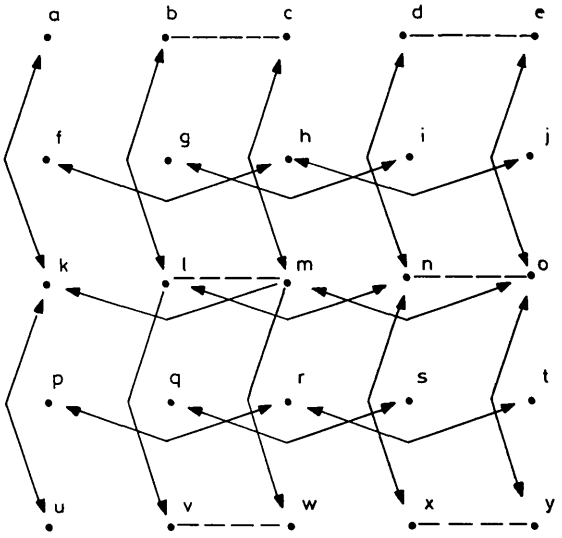

Fig. 1 *Window for image enhancement*

Fig. 2 *Redundancy in 8-pixel neighbourhood mean*

——— redundant addition of two points
– – – – redundant addition of previous 2-point additions

An additional 10% improvement in speed can be attained by increasing the storage space and implementing as follows. For candidate g, additions ak, bl, cm and fh; and for candidate q, additions ku, lv, mw and pr are required. It may be noted that fh and pr, already calculated, can be used for candidate l in the manner shown in Fig. 2. Actually, additions like fh and pr, but not km, are shared by three candidates, and this is the source of additional redundancy. It can be shown in the same way that the average number of operations necessary is 4.25 and the speed is improved to $7/4.25 = 1.64$. The operation can be continued row or columnwise, but three rows or columns have to be taken at a time and the storage requirement is increased by $2N$, where N is the width of the picture.

For mean filters, actually, a division by eight is necessary. Division or multiplication by a power of two is very easy in a digital machine and may be implemented by decreasing or

Paper 2397E, first received 1st July and in revised form 2nd December 1982

The author is with the Electronics & Communication Science Unit, Indian Statistical Institute, Calcutta, India

increasing the significance of bit location, virtually taking no time.

The procedure is also useful in point, edge and line detection. For point detection, again, fortunately $w_5 = 8$, a power of 2, and $w_i = -1$ $(i \neq 5)$. Thus, the number of operations in this case is $4.25 + 1 = 5.25$ whereas usually it is $7 + 1 = 8$. In edge detection template $w_4 = w_5 = w_6 = 0$, $w_1 = w_3 = 1$, $w_2 = 2$, $w_7 = w_9 = -1$ and $w_8 = -2$ for edges in the horizontal direction. The number of operations is usually five, whereas in the present method it is 2.25, and speed improvement to 2.2 times the usual speed is possible. This is also true for edges in the vertical direction.

For median filtering, the determination of the median of eight neighbours by the straightforward method requires $8 \times 7/2 = 28$ comparisons. This number can be reduced using the Mergesort algorithm [6] which partitions the elements successively as a power of two and works by repeatedly selecting the larger of the largest elements while merging two partitions. For sorting eight numbers, the maximum number of comparisons needed in Mergesort is 17. If, however, the redundancy discussed for mean filtering is utilised, the maximum average number of comparison is reduced to 12.25 and the speed is improved at least to $28/12.25 = 2.3$ times the usual speed.

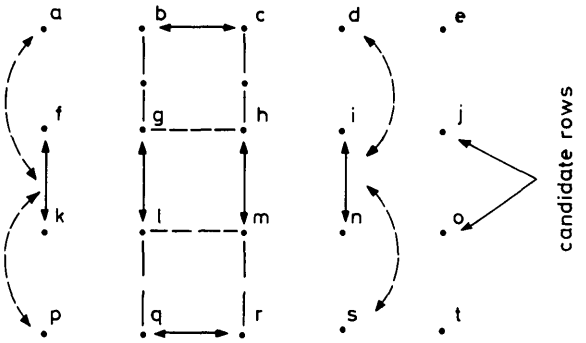

Fig. 3 *Redundancy in 9-pixel median*

Continuous vertical lines in candidate row denote redundant 2-point sorting; broken straight lines denote redundancy in 4-point sorting; broken curved lines denote nonredundant sorting. Redundancy in 6-point sorting is shown by vertical lines interrupted by a single point

Sometimes, mean or median filtering also includes the candidate pixel. In such a case, the determination of the median of nine numbers by the straightforward method requires $9 \times 8/2 = 36$ comparisons. Using Mergesort, this number is reduced to 21. But, if again the redundancy is considered, this number can further be reduced to a maximum average of 11.25. The redundancy is explained in Fig. 3 where four candidates g, h, l and m are considered. Let $(a, b) \rightarrow R$ signify that R is an array with elements a and b, but sorted in decreasing order or magnitude. Similarly $(R_1, R_2) \rightarrow R$ signifies an ordered array R whose elements are the elements of the ordered arrays R_1 and R_2. Denote $(f, k) \rightarrow R_1$, $(g, l) \rightarrow R_2$, $(h, m) \rightarrow R_3$, $(i, n) \rightarrow R_4$, $(b, c) \rightarrow R_5$ and $(q, r) \rightarrow R_6$. Sorting each of R_i, $i = 1, 6$ requires one comparison. Each sorting is redundant over the candidates. For example, R_3 is useful for the median evaluation of g, h, l, m as well as i, n. Actually, each R_i, $i = 1, 4$, can be shared by six candidates. Next, $(R_2, R_3) \rightarrow R_7$ is obtained. R_7 is shared by all four candidates g, h, l, m. Next $(R_7, R_5) \rightarrow R_8$ and $(R_7, R_6) \rightarrow R_9$ are obtained. R_8 and R_9 are shared, respectively, by the candidates g, h and l, m. However, arrays $(R_1, a) \rightarrow R_{10}$, $(R_1, p) \rightarrow R_{11}$, $(R_4, d) \rightarrow R_{12}$, and $(R_4, s) \rightarrow R_{13}$ are shared by none but the respective candidates g, l, h and m. Finally, $(R_8, R_{10}) \rightarrow R_{14}$, $(R_8, R_{11}) \rightarrow R_{15}$, $(R_9, R_{12}) \rightarrow R_{16}$ and $(R_9, R_{13}) \rightarrow R_{17}$ gives the median of g, l, h and m respectively.

R_7 requires three, each of R_8 and R_9 requires five, and each of R_{10}, R_{11}, R_{12}, R_{13} requires two comparisons. Since the median is the fifth largest element in the arrays R_{14}, R_{15}, R_{16}, R_{17}, four comparisons in each of them is enough. Also, R_3 and R_4 of candidates g, h, l, m can replace R_1 and R_2 of candidates i, j, n, o, and hence R_1, R_2 need not be sorted, except at the beginning of the candidates pair of rows. Taking all these into account, the total number of comparisons required for four candidates is at most 45, making the average 11.25 for each. The speed is improved to $36/11.25 = 3.2$ times.

Recently, Huang *et al.* [7] reported an algorithm which reduces the number of comparisons in median filtering by histogram updating. For a 3×3 window, in addition to histogram setting, this method requires a minimum of $7.5 + d$ comparisons, where d is the average difference of neighbouring medians of a picture. For a picture of moderate details, d ranges from 10 to 15. The present approach, on the other hand, requires only comparisons, and its number is independent of picture details. Furthermore, it requires a fewer number operations than in Reference 7 for a picture of moderate, as well as high, details.

It can be seen that the approach can also be used to find the mean. The only modification necessary in the program is to signify $(a, b) \rightarrow R$ as $a + b = R$. In this case, the number of additions can be reduced from 8 to an average of 3.75.

It is seen that the number of comparisons or additions required in case of 9 pixels is less than in case of 8 pixel mean or median evaluation in the present method. However, in the 8 pixel case, the candidate is at the centre of a 3×3 submatrix which has to be excluded from the computation. The spatial redundancy of a compact 3×3 structure is therefore reduced, and hence the number of operations is increased.

The present technique is quite simple and can be programmed or implemented in digital hardware very conveniently. The program listing of the algorithm in Applesoft Basic as well as Fortran language can be supplied by special arrangement.

3 Acknowledgments

The author wishes to thank Prof. D. Dutta Majumder, Indian Statistical Institute and B. Bates, Queen's University of Belfast, for their interest in the work. A visiting fellowship offered by Leverhulme Trust to support the present work is gratefully acknowledged. The work was supported partly by the Trust and the Indian Statistical Institute, and was conducted at Queen's University.

4 References

1 GONZALEZ, R.C., and WINTZ, P.: 'Digital image processing' (Addison-Wesley, London, 1977), pp. 136–175
2 FRIEDEN, B.R.: 'A new restoring algorithm for the preferential enhancement of edge gradients', *J. Opt. Soc. Amer.*, 1976, **66**, pp. 280–283
3 NAGAO, M., and MATSUYAMA, T.: 'Edge preserving smoothing', *Comp. Graph. Image Process.*, 1979, **9**, pp. 394–407
4 CHANDA, B., CHAUDHURI, B.B., and DUTTA MAJUMDER, D.: 'Image enhancement and edge detection for human visual system'. Indian Statistical Institute Tech. Rep. TR1/ECSU/81, 1981, pp. 1–16
5 FREI, W., and CHEN, C.C.: 'Fast boundary detection: a generalisation and a new algorithm', *IEEE Trans.*, 1977, **C-26**, pp. 988–998
6 AHO, A.V., HOPCROFT, J.E., and ULLMAN, J.D.: 'The design and analysis of computer algorithms' (Addison-Wesley, London, 1974), pp. 66–67
7 HUANG, T.S., YANG, G.J., and TANG, G.Y.: 'A fast two-dimensional median filtering algorithm', *IEEE Trans.*, 1979, **ASSP-27**, pp. 13–18