The effect of window mask shape on algorithm design in pipeline architectures

Amelia Fong
Department of Computing and Information Science
University of Guelph
Guelph, Ontario
Canada N1G 2W1

Abstract

Due to the large amount of data and the speed requirement of many vision applications, parallel architectures are increasingly being used. Recently, there has been much research in algorithms for Pipeline Architectures. In this paper, we discuss how we can take advantage of the shape of the masks and the spatial relationships between the data points in designing algorithms for pipeline architecture. This design methodology allow individual fast algorithm to be designed for different local window masks. We shall illustrate this concept by giving several algorithms for different masks for median filters. The classes of median filters considered in this paper are cross shape, X-shape, all samples median filter and the separable median filter. The correctness of these algorithms are also proven.

KEYWORDS: median filters, window mask shape, algorithm design, parallel computation, pipeline architectures.

A cause du grand nombre de donnees et de la vitesse requise par plusieurs applications visuelles, les architectures paralleles sont de plus en plus utilisees. Recemment, beaucoup de recherches sur les algorythmes pour architectures en pipeline ont ete effectuees. Cet article examine la facon de tirer avantage de la forme des masques de meme que des relations spatiales entre les points de donnees dans le design d'algorythmes pour architecture en pipeline. Cette methodologie permet le design d'algorythmes individuels rapides pour differents masques en fenetre locaux. Nous illustrerons ce concept par des algorythmes pour differents masques pour filtres medians. Les filtres medians en forme de croix, ceux en forme de X, les filtres medians separables et tout autre echantillonage de filtres medians constituent les differentes categories etudiees dans cet article. Nous avons egalement fait la preuve de l'exactitude de ces algorythmes.

MOTS CLES: filtres medians, forme des masques en fenetre, design d'algorythmes, compilation parallele, architectures en pipeline.

Support for this research from the Natural Science and Engineering Research Council of Canada is gratefully acknowledged.

I. Introduction

Computer based image analysis have a wide range of applications. These include industrial applications such as automatic visual inspection and robotic vision. Biomedical applications include computer tomography, X-ray imaging, and Electron Microscopic imaging. Due to the large amount of data and the speed requirement, parallel architectures are often being used. Recently, several vision systems have been proposed using pipeline architectures [Petkovic, 86], [Persoon, 88], [Sanz & Petkovic, 88]. Pipeline architectures have fast performance and are commercially available, allowing applications to be tested before dedicated equipment need to be built. Currently there are much research interest and publications in parallel algorithms using pipeline architectures, including various geometric features [SHD, 87], [Sanz & Dinstein, 87], polygonal masks [SDP, 87], [Sanz & Petkovic, 88], projections [Sanz & Hinkle, 88], local maximum and minimum [Dinstein & Fong, 88], and non-linear hybrid filters [Fong, 89] In this paper, we propose a new approach for algorithm design for this type of parallel architectures.

In image processing applications, many operations are defined based on a local window. The window is often square in shape, sometimes rectangular, and the data points chosen within the window may vary as well. There has been a large amount of literature on image enhancement filters, where properties and utilities of filters of various size and shape are studied. In this paper, we shall consider the best known nonlinear order-statistical filters, the median filter. It was first introduced by Tukey[TU, 74], and has been observed to be very effective for removing noise, especially impulse noise, for one or two dimensional signals, while satisfying the usually conflicting goal of preserving information-bearing edges [GW, 81], [H, 81]. The properties of the median filters had been studied in details [AAW, 81], [AG, 82], [KW, 81], [NG, 84], [BHM, 87], [B, to appear]. They have been successfully applied to many signal and image processing tasks.

In standard median filters, the median is taken over all samples inside the window. When the number of samples is large, the ordering procedure is computationally intensive. Masks for the median filters other than the entire window have been proposed. These include the cross-shape and the X-shape windows, the separable median filters, and the multilevel median filters [NHN, 87]. In this paper, we propose an algorithm design methodology by taking advantage of the

relationship of the data points in the mask definition and the underlying pipeline architecture. Hence, separate algorithms can be designed for fast computation of different classes of median filters, based on the shape of the mask definition. In this paper, our ideas will be demonstrated by using the 3×3 window.

The paper is organized as follows. Section II describes the underlying hardware architectures, namely pipeline architectures. Section III contains algorithms for the Cross-shape and X-shape median filters. Section IV contains an algorithm for the all sample median filter. Section V contains algorithm for the separable median filter followed by Section VI contains discussion and conclusion.

II. Pipeline Architecture

In this section, we describe the abstract machine architecture we are assuming. We assume a high speed pipeline processor which takes input from a number of image refresh memories or from a digitizer. The processor typically consists of several parallel stages of processing, such as described below. An example of such an architecture is the DeAnza IP8500 [De, 83]. Other typical features include various display capabilities, input/output devices such as joysticks, lightpens and trackballs, programmable cursor units, etc. which are not relevant in our present discussion.

We describe a prototype general purpose pipeline image processor with more details. It typically consists of an input selection stage, a multiplier stage which (optionally) multiply pairs of selected inputs. This is followed by one or more ALU stage(s) which performs arithmetic and/or logical operations. These are followed by a comparator stage, a shifter stage and a function table stage which can be used as a look-up table. We assume also a feed back loop from the pipeline processor to refresh memories whose number can vary depending on the particular configuration. A schematic diagram of a pipeline architecture machine model is shown in Fig. 1. We assume each refresh memory has on board look up table for performing the basic transformations such as translations in the x or y or both directions as the data streams from the memory. In short, what we have assumed is a general purpose pipeline architecture such as in [Sanz & Dinstein, 87], [Dinstein & Fong, 88], so that our algorithms can be easily implemented on any pipeline systems. The only processing unit we assume for the ordering operations is a hardware comparator. Most of the commercially available general purpose pipeline architectures are equipped with such comparator. For example, the DeAnza has two such comparators which can be set independently to output either the maximum or the minimum of the two input pixel values.

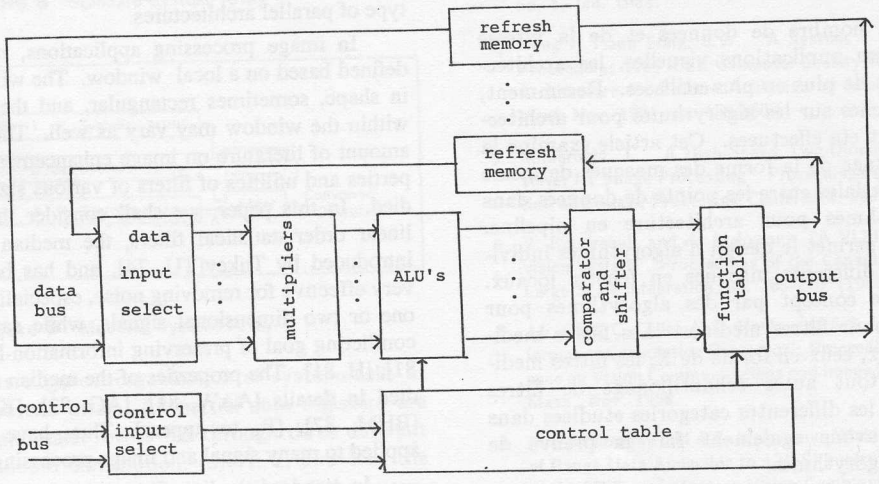

Fig. 1 Schematic Diagram of Pipeline Architectures

III. Cross-shape and X-shape median filters

The masks for the cross-shape and the X-shape median filters for the 3 × 3 windows are illustrated in Fig. 2(a) and 2(b). That is, the output of the median filter
y(m,n) = median { x[i,j] | [i,j] in the mask centred at [m,n] }.
These two filters are comparable in their degree of noise suppression. The cross-shape filter are more effective for images containing mostly edges in the horizontal/vertical directions, whereas the X-shape filter produce better results with respect to diagonal edges [BMH, 1987].

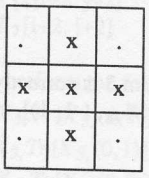

Fig. 2(a)

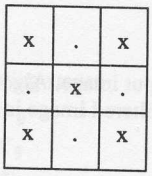

Fig. 2(b)

Let us define $Tr[A,(p,q)]$ to be the resulting image of translating the image A p pixels in the x direction, and q pixels in the y direction. When p (or q) is negative, it indicates that the translation is in -x direction (-y direction).

The cross shape median filter can be computed in Algorithm 1 below.

Algorithm 1:

Let P be the original image. The following image processing passes will be taken:

1. $\max(P,Tr[P,(-1,1)])=X_1$
2. $\min(P,Tr[P,(-1,1)])=X_2$
3. $\max(X_2,Tr[X_2,(-1,-1)])=X_3$
4. $\min(X_1,Tr[X_1,(-1,-1)])=X_4$
5. $\max(X_3,X_4)=X_5$
6. $\min(X_3,X_4)=X_6$
7. $\max(Tr[X_6,(1,0)],P)=X_7$
8. $\min(Tr[X_5,(1,0)],X_7)=X_8$

Output image: X_8

Note that the hardware comparators behave differently than comparisons in sequential machines. This is because in sequential machine, a comparison of say a < b results in determining whether a or b is larger, which is then being used to guide further testing. In our image processing operation max(A, B), the resulting image C is the maximum of A and B pixelwise. We do not know for each pixel C[i,j] whether it is from A or from B. In pipeline architectures one does not have random access to the image planes, nor the hardware logic to perform branching pixelwise as can be done in sequential machine. In fact, the correctness of the algorithms need to be proven.

In the following, we shall prove that Algorithm 1 is correct. We need to prove that, for each pixel [i,j] of X_8, it is the median value of the pixels [i,j], [i-1,j], [i+1,j], [i,j-1], [i,j+1] of P (except for boundary pixels). We shall use 'a' to denote a pixel of image A, 'x_1' to denote a pixel of X_1, and so on. In this proof, it is important that we distinguish between the pixel itself and the value of the pixel. This is because our algorithm evaluate only the value of the median, and cannot provide the median pixel itself, as in algorithms for sequential machines. We use 'p' to denote a pixel of the image P, and v(p) to denote the value of p.

From steps (1) and (2), we can deduce that $v(x_2) \le v(x_1)$. That is, x_1 = max(P[i-1,j], P[i,j-1]) and x_2 = min(P[i-1,j], P[i,j-1]). Notice that max(P[i,j+1], P[i+1,j]) and min(P[i,j+1], P[i+1,j]) have also been computed. They are simply $Tr[X_2,(1,1)]$ and $Tr[X_1,(1,1)]$ respectively. Considering steps (3) and (4), we may assume, without loss of generality, that P[i,j+1] = $v(x_3)$ and $v(x_1) = v(x_4)$. Hence, $v(x_2) <= v(x_3)$ and $v(x_2) \le v(x_1) \le$ P[i+1,j]. Hence $v(x_2)$ cannot be the median value, as there are at least three values greater than or equal to it. Similarly, at least three of the five pixels in question are less than or equal to P[i+1,j], which cannot be the median value. Note that steps 1 to 4 compute $v(x_1)$ to $v(x_4)$ in the pixel [i-1, j] for the local window centred at [i,j]. Steps (5) to (8) compute the median of $v(x_1)$, P[i, j+1]) and P[i,j] at [i,j] by translating X_5 and X_6 one piexl in the x-direction. Hence X_8[i,j] is the median value of P[i,j], P[i-1,j], P[i+1,j], P[i,j-1], P[i,j+1].

The X-shape median filter can be computed in similar fashion. Again we take advantage of of the shape of the mask defined for this filter.

Algorithm 2:

1. $\max(P,Tr[P,(0,2)])=X_1$
2. $\min(P,Tr[P,(0,2)])=X_2$
3. $\max(X_2,Tr[X_2,(-2,0)])=X_3$
4. $\min(X_1,Tr[X_1,(-2,0)])=X_4$
5. $\max(X_3,X_4)=X_5$
6. $\min(X_3,X_4)=X_6$
7. $\max(Tr[X_6,(-1,-1)],P)=X_7$
8. $\min(Tr[X_5,(-1,-1)],X_7)=X_8$

The proof is similar and we omit it here.

IV. All sample median filter

This algorithms is the most interesting. This mask contains 9 data points. Again we design an algorithm that is based on the relationships between the data points. The algorithm is not obvious and its correctness need to be proven.

In this paper, we are not concerned with the properties and utility of the various filters. Median filters have been extensively studied in the literature and have been widely used in various image analysis applications. They are chosen as examples to illustrate how one can take advantage of the definition of mask to design individual fast algorithms, which are of interest in is own right.

The following is an algorithm for the 3×3 all samples median filter.

Algorithm 3:

Let P be the input image

$P_1 = Tr[P, (1,0)]$

$P_2 = Tr[P, (2,0)]$

The following pipeline passes are performed:

1. $\max(P, P_{1)} = X_1$
2. $\min(P, P_{1)} = X_2$
3. $\max(X_2, P_{2)} = X_3$
4. $\min(X_1, X_{3)} = X_4$
5. $\max(X_1, P_{2)} = X_5$
6. $\min(X_2, P_{2)} = X_6$
7. $\max(X_6, Tr[X_6, (0,1)]) = X_7$
8. $\max(X_7, Tr[X_6, (0,2)]) = X_8$
9. $\min(X_5, Tr[X_5, (0,1)]) = X_9$
10. $\min(X_9, Tr[X_5, (0,2)]) = X_{10}$
11. $\max(X_4, Tr[X_4, (0,1)]) = X_{11}$
12. $\min(X_4, Tr[X_4, (0,1)]) = X_{12}$
13. $\max(X_{12}, Tr[X_4, (0,2)]) = X_{13}$
14. $\min(X_{11}, X_{13)} = X_{14}$
15. $\max(X_8, X_{10)} = X_{15}$
16. $\min(X_8, X_{10)} = X_{16}$
17. $\max(X_{16}, X_{14)} = X_{17}$
18. $\min(X_{15}, X_{17)} = X_{18}$

Then $Tr[X_{18}, (-1,-1)]$ is the median filtered image of P.

Before proving the correctness of the above algorithm, we shall first prove the following lemma.

Lemma

Assume that $a \le x$, $b \le y$ and $c \le z$. Apply a permutation t to a, b and c such that t(a) = a', t(b) = b' and t(c) = c' such that $a' \le b' \le c'$. Similarly, apply a permutation s to x, y and z such that s(x) = x', s(y) = y' and s(z) = z' and $x' \le y' \le z'$. Then $a' \le x'$, $b' \le y'$ and $c' \le z'$.

Proof

First assume $a \le x$ and $b \le y$. In permuting a with b and x with y to obtain a sorted order, there are four possible cases. We shall prove that in each case, $a' \le x'$ and $b' \le y'$. The first case is when $a \le b$ and $x \le y$; in which case a = a', b = b', x = x' and y = y'. The second case is when $a \le b$ and $x > y$. That is a = a' and b = b', y = x' and x = y'. Now $b' = b \le y < x = y'$. Also $a' = a \le b \le y = x'$. The third case is when $a > b$ and $x \le y$. That is a' = b, b' = a and x = x', y = y'. In this case, $b' = a \le x \le y = y'$. Also $a' = b < a \le x = x'$. The last case is when $a > b$ and $x > y$. In this case, we interchange a with b and x with y. Clearly $a' = b \le y = x'$ and $b' \le y'$. Now assume also that $c \le z$. By applying the same argument as above, we can permute b with c and y with z while maintaining $b' \le y'$ and $c' \le z'$. By doing such pairwise interchanges, we can obtain a sorted sequence $a' \le b' \le c'$ and $x' \le y' \le z'$, while maintaining $a' \le x'$, $b' \le y'$ and $c \le z'$.

Theorem

If P is the input image, Algorithm 3 is correctly computes the 3×3 median filtered image in $Tr[X_{18}, (-1,-1)]$.

Proof

We name the 9 data points in the mask as P[i:i+2, j:j+2] with the obvious interpretation, i.e. the first subscript indicates the x-coordinate and the second the y-coordinate.

Steps (1) to (6) compute the median image pixelwise of P[i,j], P[i+1,j], P[i+2,j] in the pixel X_4[i+2, j], the maximum of the same three pixels in X_5[i+2, j] and the minimum of the same three pixels in X_6[i+2, j]. However, the maximum, minimum and median pixel for the pixels P[i,j+1], P[i+1,j+1], P[i+2,j+1] and for the set of three pixels P[i,j+2], P[i+1,j+2] P[i+2,j+2] have also been simultaneously computed in X_5, X_6 and X_4 respectively.

Note that for a particular 3×3 window, the 9 data points in question are X_5[i+2,j:j+2], X_4[i+2, j:j+2] and X_6[i+2, j:j+2], with the property that $X_6[i+2,k] \le X_4[i+2,k] \le X_5[i+2,k]$ for $j \le k \le j+2$.

If we apply the following 6 steps:

1'. $\max(X_6, Tr[X_6, (0,1)]) = X_7$

2' $\min(X_6, Tr[X_6, (0,1)]) = T_2$

3' $\max(T_2, Tr[X_6, (0,2)]) = T_3$

4' $\min(X_7, T_{3)} = T_4$

5' $\max(X_7, Tr[X_6, (0,2)]) = X_8$

6' $\min(T_2, Tr[X_6, (0,2)]) = T_6$

Note that steps 1' and 5' are the same steps as step 7 and 8. The other steps are used to derive images necessary for the proof. The intermediate images generated from these steps are named T_i's and it will be shown that they need not be computed.

Steps 1' to 6' sort the pixels X_6[i+2, j:j+2] into T_6[i+2, j+2], T_4[i+2, j+2] and X_8[i+2, j+2] such that T_6[i+2, j+2] $\le T_4$[i+2, j+2] $\le X_8$[i+2, j+2].

Similarly if we perform the following passes

11. $\max(X_4, Tr[X_4, (0,1)])=X_{11}$
12. $\min(X_4, Tr[X_4, (0,1)])=X_{12}$
13. $\max(X_{12}, Tr[X_4, (0,2)])=X_{13}$
14. $\min(X_{11}, X_{13})=X_{14}$

14'. $\max(X_{11}, Tr[X_4, (0,2)])=T_7$

14''. $\min(X_{12}, Tr[X_4, (0,2)])=T_8$

The above 6 steps sort X_4[i+2, j:j+2] into T_8[i+2, j+2], X_{14}[i+2, j+2] and T_7[i+2, j+2] such that T_8[i+2, j+2] ≤ X_{14}[i+2, j+2] ≤ T_7[i+2, j+2].

Again it will be shown that steps 14' and 14'' will not need to be computed.

Then by applying the Lemma,
T_6[i+2, j+2] ≤ T_8[i+2, j+2]
T_4[i+2, j+2] ≤ X_{14}[i+2, j+2]
X_8[i+2, j+2] ≤ T_7[i+2, j+2]

Similarly if we perform the following steps:

1". $\max(X_5, Tr[X_5, (0,1)])=T_9$
2" $\min(X_5, Tr[X_5, (0,1)])=X_9$
3" $\max(X_9, Tr[X_5, (0,2)])=T_{10}$
4" $\min(T_9, T_{10})=T_{11}$
5" $\max(T_9, Tr[X_5, (0,2)])=T_{12}$
6" $\min(X_9, Tr[X_5, (0,2)])=X_{10}$

where steps 2" and 6" are the steps 9 and 10 in Algorithm 3 and it will be shown that the other steps need not be performed.

Steps 1" to 6" sort the pixels X_5[i+2, j:j+2] into X_{10}{i+2, j+2], T_{11}[i+2, j+2] and T_{12}[i+2, j+2] such that X_{10}{i+2, j+2] ≤ T_{11}[i+2, j+2] ≤ T_{12}[i+2, j+2].

By applying the Lemma to the set of 3 images T_8 and X_{14} and T_7, and the set of 3 images X_{10}, T_{11} and T_{12}, we have the following:
T_8[i+2, j+2] ≤ X_{10}[i+2, j+2]
X_{14}[i+2, j+2] ≤ T_{11}[i+2, j+2]
T_7[i+2, j+2] ≤ T_{12}[i+2, j+2]

In the following, all references to images refer to the pixel [i+2, j+2]. We omit the subscripts to improve readability. Since $T_4 \le X_{14} \le T_{11}$, and $T_4 \le X_8 \le T_7 \le T_{12}$, there are at least 5 elements larger than or equal T_4, which cannot be the median of the 9 points in question. Since $T_6 \le T_4$, it cannot be the median either. Similarly T_8 is ≤ X_{14} ≤ T_7, and also ≤ X_{10} ≤ T_{11} ≤ T_{12}, hence T_8 cannot be the median. Similarly, there are more than 5 points smaller than or equal to T_7, T_{11}, and T_{12}, none of which can be the median pixel.

Any one of X_8, X_{10} and X_{14} can be the median. We claim the it is the median of these three imagepixels which is computed in steps 15 to 18 in Algorithm 3. The argument is as follows. Clearly T_6, T_4, T_8 are all ≤ X_{14}. Also X_{14} is ≤ T_7, T_{11} and T_{12}. Hence there is a group of 3 pixels elements which are all less than or equal to X_{14} and a group of 3 pixels elements which are all greater than or equal to X_{14}. We do not know where X_8 and X_{10} are in relation to X_{14}. If one is smaller (or equal) and the other is larger (or equal), then X_{14} is the desired median. If they are both larger (or equal) than X_{14}, then the smaller of the two will be the median. If they are both smaller than X_{14}, then the larger of the two will be the median. In all these three cases, the median of X_8,X_{10},X_{14} gives the desired median.

The above argument shows that for an input image P of size N x M, for any local window P[i:i+2, j:j+2] for 1≤i≤N-2 and for 1≤j≤M-2, the median pixel for those pixels within the window is computed in X_{18}[i+2, j+2]. Since the local window [i:i+2, j:j+2] is centred at [i+1,j+1], Tr[X_{18}, (-1,-1)] gives the desired median filtered image for P.

V. Separable median filter

In this section, we shall consider the separable median filter, A separable median filter [NG, 1983] is defined by passing first a horizontal oriented one-dimensional filter along each image row, the resulting image is then filtered by passing the filter (rotated 90°) over each column. It is not a true two-dimensional filter, but rather two one-dimensional median filters taken at orthogonal directions.

The algorithm for the 3 × 3 is simple as each mask consists of only three data points. The following is an algorithm for the 5 × 5 separable median filter. In this case, the relationships of the data points allow Algorithm 1 to be adapted and the details are as follows.

Algorithm 4:

Let P be the input image. The following image processing passes will be taken:

1. $\max(P, Tr[P, (0,1)])=X_1$
2. $\min(P, Tr[P, (0,1)])=X_2$
3. $\max(X_2, Tr[X_2, (0,-3)])=X_3$
4. $\min(X_1, Tr[X_1, (0,-3)])=X_4$
5. $\max(X_3, X_4)=X_5$
6. $\min(X_3, X_4)=X_6$
7. $\max(X_6, P)=X_7$
8. $\min(X_5, X_7)=X_8$
9. $\max(X_8, Tr[X_8, (1,0)])=X_9$
10. $\min(X_8, Tr[X_8, (1,0)])=X_{10}$
11. $\max(X_{10}, Tr[X_{10}, (-3,0)])=X_{11}$
12. $\min(X_9, Tr[X_9, (-3,0)])=X_{12}$
13. $\max(X_{11}, X_{12})=X_{13}$
14. $\min(X_{11}, X_{12})=X_{14}$
15. $\max(X_{14}, X_8)=X_{15}$
16. $\min(X_{13}, X_{15})=X_{16}$

Output image: X_{16}

VI. Discussion and Conclusion

Currently there has been much interests and research activities in designing algorithms for pipeline architectures. In this paper, we have presented fast algorithms for the cross-shape and X-shape median filters and proven their correctness. We have proposed an entirely different algorithm for the all sample median filter, which takes advantage of the relationship between the data points in the mask.

For the 5×5 separable median filter, an adaptation of the algorithm for computing the median of 5 points can be used for computing the medians row-wise, and then column-wise. However, the relationships in one orientation cannot be taken advantage of in the orthogonal direction.

In a pipeline image processing system that assumes video refresh rate , i.e. 30 frames/sec, all the above discussed median filters can be computed in well under a second. However, this is assuming that there is enough refresh memories for all intermediate images generated during the process. If not enough refresh memories are available in a given configuration of hardware, intermediate result has to be saved and reloaded from disk, thus increasing the total time required.

In [Fong 88], we discussed both time and space analysis of algorithms for pipeline architectures. We analysed space complexities in terms of the number of refresh memories required to implement the algorithm without disk transfer. In particular, algorithm 1 above requires 5 refresh memories if no disk transfer is performed. Some reordering of the steps is necessary and the content of the 5 refresh memories as we progress through the algorithm are shown in Table 1 below.

step 1	P	P	X_1	X_1	
step 6	P	P	X_1	X_1	X_4
step 2	P	P	X_2	X_2	X_4
step 3	P	X_3	X_2	X_2	X_4
step 5	P	X_3	X_5	X_2	X_4
step 6	P	X_3	X_5	X_6	X_4
step 7	X_7	X_3	X_5	X_6	X_4
step 8	X_7	X_3	X_5	X_8	X_4

Table 1.

Similarly, algorithm 2 can be implemented without disk transfer using 5 refresh memories.

As shown in [Fong, 88], Algorithm 3 can be implemented without disk transfer using 6 refresh memories. For the sake of completeness, the content of the refresh memories are included here as Table 2 below. A careful reordering of the steps in the algorithms are often necessary. There are also various techniques for space-time tradeoff, which we shall not discuss here. As for Algorithm 4, the number of refresh memories required without disk transfer is 5. It is a two stage algorithm, where the first stage of computing the one-dimensional median image in the horizontal direction requires 5 refresh memories. The output of the first stage is then used as input to the second stage, which computes the median image column-wise, using the same 5 refresh memories.

step 1	P	P_1	X_1			
step 2	P	P_1	X_1	X_2		
step 3	P	P_1	X_1	X_2	X_3	
step 6	X_6	P_1	X_1	X_2	X_3	X_6
step 7	X_6	P_1	X_1	X_7	X_3	X_6
step 8	X_6	P_1	X_1	X_7	X_3	X_8
step 5	X_5	P_1	X_1	X_5	X_3	X_8
step 9	X_5	X_9	X_1	X_5	X_3	X_8
step 10	X_5	X_9	X_1	X_{10}	X_3	X_8
step 4	X_4	X_4	X_1	X_{10}	X_3	X_8
step 11	X_4	X_4	X_{11}	X_{10}	X_3	X_8
step 12	X_4	X_{12}	X_{11}	X_{10}	X_3	X_8
step 13	X_4	X_{12}	X_{11}	X_{10}	X_{13}	X_8
step 14	X_{14}	X_{12}	X_{11}	X_{10}	X_{13}	X_8
step 15	X_{14}	X_{15}	X_{11}	X_{10}	X_{13}	X_8
step 16	X_{14}	X_{15}	X_{16}	X_{10}	X_{13}	X_8
step 17	X_{14}	X_{15}	X_{16}	X_{17}	X_{13}	X_8
step 18	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}	X_8

Table 2.

In conclusion, we like to note that each of the algorithms proposed takes advantage of the relationships between the data points in the particular local window mask in its design. Since it is separately designed to achieve maximum speed, it does not generalize to other shape masks. However, the same methodology can be applied in designing algorithms for other masks for pipeline architectures so that the parallelism in the hardware can be fully utilized. The implication of the space requirements for the algorithms are also briefly discussed.

References

[AAW, 81] E. Ataman, V.K. Aatre, and K.M. Wong, "Some statistical properties of median filters", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-29, pp. 1073-1075, Oct. 1981.

[AC, 84] G.R. Arce and R.J. Crinon, "Median filters: Analysis for 2-dimensional recursively filtered signals", in Proc. IEEE ICASSP-84, San Diego, CA, Mar. 1984, pp. 20.11.1-20.11.4.

[AG, 82] G.R. Arce and N.C. Gallagher, "State description for root- signal set of median filters", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-30, pp. 894-902, Dec. 1982.

[B, to appear] A.C. Bovik, "Streaking in median filtered images," IEEE Trans. Acoust., Speech, Signal Processing, to appear.

[BHM, 87] A.C. Bovik, T.S. Huang, and D.C. Munson, Jr., "Effect of median filtering on edge estimation," IEEE, Trans. of Pattern Analysis and Machine Intelligence, Vol. PAMI-9, Mar. 1987, pp.181-194.

[Chin, 86] R.T. Chin, "Algorithms and Techniques for Automated Visual Inspection", in Handbook of Pattern Recognition and Image Processing, ed. by T.Y. Young and K.S. Fu, Academic Press, 1986, pp. 587-612.

[De, 83] IP8500 Image Array Processor Programming and Software Manuals, published by Gould DeAnza Imaging and Graphics Division.

[Dinstein and Fong, 88] I. Dinstein and A.C. Fong (Lochovsky), "Computing Local Minima and Maxima of Digital Images in Pipeline Image processing Systems equipped with Hardware Comparators", Proceedings of the IEEE. Mar., 1988, pp. 286-287.

[Fong, 88] Fong (Lochovsky), A., "Analysis and Comparison of Parallel Algorithms for Image Enhancement in Computer Vision Applications", International Computer Science Conference '88 on Artificial Intelligence (sponsored by IEEE), Hong Kong, Dec. 1988, pp.473-480.

[Fong, 89] Fong (Lochovsky), A.," Algorithms and Architectures for a Class of Non-linear Hybrid Filters using Pipeline Architectures", to appear, Computer Vision, Graphics and Image Processing.

[GW, 81] N.C. Gallagher, Jr., and G.L. Wise, "A theoretical analysis of the properties of median filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-29, pp. 1136-1141, Dec. 1981.

[H, 81] T.S. Huang, Ed., Two-Dimensional Digital Signal Processing II: Transforms and median filters. New York: Springer-Verlag, 1981.

[KW, 81] F. Kuhlman and G.L. Wise, "On second moment properties of median filtered sequences of independent data," IEEE Trans. Commun., vol. COM-29, pp. 1374-1379, 1981.

[NG, 83] T.A. Nodes and N.C. Gallagher, Jr., "Two-dimensional root structures and convergence properties of the separable median filter", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, pp.1350-1365, Dec. 1983.

[NG, 84] T.A. Nodes and N.C. Gallagher, Jr., "The output distribution of median type filters", IEEE Trans. Commun., vol. COM-32, pp. 532-541, May 1984.

[NHN, 87] A. Nieminen, P. Heinonen and Y. Neuvo, "A New Class of Detail-Preserving Filters for Image Processing", in IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, Jan. 1987, pp. 74-90.

[Persoon, 88] E. Persoon, "A Pipelined Image Analysis System Using Customed Circuits", IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-10, Jan., 1988, pp.110-117. pp. 74-90.

[Petkovic, 86] D. Petkovic *et al*, "An experimental system for disc heads inspection", Proc. of the 8th Int. Conf. on Pattern Recognition, Oct. 1986, Paris, France.

[RNR, 84] E.R. Ritenour, T.R. Nelson, and U. Raff, "Applications of the median filter to digital radiographic images", in Proc. IEEE ICASSP-84, San Diego, CA, Mar. 1984, pp. 23.1.1-23.1.4.

[Sanz & Dinstein, 87] J. L.C. Sanz and I. Dinstein, "Projection- based Geometrical Feature Extraction for Computer Vision Algorithms: Algorithms in Pipeline Architectures", In IEEE Transaction on Pattern Analysis and Machine Intelligence, Jan. 1987, pp. 160-168.

[Sanz & Hinkle, 88] J.L. Sanz and E. Hinkle, "Computing projections of digital images in image processing pipeline architectures", IEEE Trans. Acoust., Speech, Signal Processing, ASSP-35, No. 2, Feb., 1987, pp. 198-207.

[Sanz & Petkovic, 88] J.L. Sanz and Petkovic, D., "Machine Vision Algorithms for Automatic Inspection of Thin-Film Disk Heads", IEEE Trans. on Pattern Anal. & Machine Intell., PAMI-10, Nov. 88, pp.830-848.

[SDP, 87] J.L.C. Sanz, I. Dinstein and D. Petkovic, "A new procedure for computing multi-colored polygonal masks in image processing pipeline architectures and its application to automatic visual inspection", CACM, April, 1987.

[SF, 85] A. Stein and T.J. Fowlow, "The use of median filters for edge detection in noisy signals", in Proc. IEEE ISCAS-85, Kyoto, Japan, 1985, pp. 1331-1334.

[SHD, 87] J.L.C. Sanz, E. Hinkle, and I. Dinstein, "A new approach to computing geometric features of digital objects for machine vision, image processing and image analysis: Algorithms in pipeline architectures", IEEE Trans on Pattern Analysis & Machine Intelligence, PAMI-9, Feb., 1987.

[Tukey, 74] J.W. Tukey, "Nonlinear (nonsuperposable) methods for smoothing data," in Conf. Rec., 1974, EASCON, pp. 673.

[Uhr, 86] L. Uhr, "Parallel Architectures for image processing, computer vision, and pattern perception, in Handbook of Pattern Recognition and Image Processing, ed. Young and K.S. Fu, Academic Press, 1986.