QUATORZIEME COLLOQUE GRETSI - JUAN-LES-PINS - DU 13 AU 16 SEPTEMBRE 1993

A FAST L1-METRIC VECTOR MEDIAN FILTER

M.Barni, V.Cappellini

Università di Firenze, Dipartimento di Ingegneria Elettronica
Via S.Marta, 3 - 50139 Firenze (Italia)

A.Mecocci

Università di Pavia, Dipartimento di Elettronica
Via Abbiategrasso 209, 27100 Pavia (Italia)

ABSTRACT

The major drawback of vector median filters is their high computational complexity. In order to reduce it, two main approaches are possible: in the first one an alternative faster filter which has roughly the same features of the vector median is used; according to the latter an exact solution is employed. In this paper an exact fast algorithm is presented for the case of vector median filters adopting L1-metric. First it is proved that the L1-metric-based median filter can be interpreted as an approximation, over the filter window, of the componentwise application of the scalar median. Then a fast algorithm is proposed that considerably reduces the computational complexity. The effectiveness of the proposed algorithm is discussed by means of experimental results.

RESUME

Le major énconveniant du filtre "vector median" est sa complexitée de calcul. De façon de reduir cette complexitée, c'est possible operer en deux maniéres: en prémier on peut utiliser un filtre qui a presque les mêmes caratteristiques du filtre "vector median"; par contre on peut disposer d'une solution exacte. Dans cet article on represent un algoritme rapide et exacte dans le cas d'utilise de metrique L1. De prémièr fait on prouve que le filtre "vector median" qui utilise la metrique L1 peut étre consideré come une approximatione de l'application du "scalar median" une component à la foi. De consequence on propose un algoritme rapide qui reduit la complexitée du calcul. L'efficace de l'algoritme proposé vient analisée parmi les résultats de experiments.

1. INTRODUCTION

Since their introduction in 1974 [1], median filters have gained popularity as powerful tools for impulsive noise removal. Two important aspects of median filters are: the ability to preserve image edges, and the existence of fast algorithms for their implementation [2].

The early median filters (scalar median filters) were restricted to black and white images. Recently the concept of statistical median has been extended to vector images; filters that use this median are called vector median filters and maintain the edge preserving capabilities of the scalar filters [3-4]. A major drawback of the median when extended to the vector case is its high computational complexity. Let DxD and NxN be, respectively, the image and the filter window dimensions. For each point of the image the vector median is defined as the point inside the window which minimizes the sum of the distances from all the other points of the window. For the direct application of this definition the computation of N^3D^2 distances is needed for the overall image. In order to reduce the filter complexity two main approaches are possible: according to the first one, an alternative faster filter which has roughly the same features of the vector median filter is used; in the second case an exact solution is employed.

A rough estimate of the vector median can be obtained by means of cross-shaped filters [2]. For these filters the number of points contained in the window increases linearly with the window size thereby reducing the filter complexity from D^2N^3 to D^2N^2. A disadvantage of cross-shaped filters is that the relation between the actual vector median and the estimate given by such filters is not well understood. Moreover such filters are not satisfactory since the estimates of the vector median may be biased towards the values of points far from the center of the filter window.

The vectorial extension of the median depends on the distances among points which belong to a certain neighborhood of the current pixel. The distance depends on the metric which is adopted. In [5] the impact of the metric on the performance and the computation time is investigated. In the same paper by starting from a geometrical interpretation of the filter operations, a fast approximated algorithm is proposed for the case of L1 metric and the relationship between the exact and the approximated solution is discussed. An exact fast algorithm is also proposed for the case of squared euclidean metric. Unfortunately the use of the squared euclidean metric gives the poorest performance in term of edge preserving capabilities [5].

This paper is an extension of the analysis in [5]. A fast and exact algorithm for the L1-median filter is proposed which reduces considerably the computation complexity. It is also proved that the L1 median can be interpreted as an approximation to the componentwise application of the scalar median over the filter window.

In the next section the behavior of L1 median is analyzed. Section 3 fully describes the fast algorithm and section 4 presents experimental results.

2. L1 VECTOR MEDIAN

Given N vectors $\{x_1, x_2 \dots x_N\}$ in R^M, representing N points inside the current filter window, the output x_{VM} of the L1 metric vector median filter is defined by the following equations

$$\boldsymbol{x}_{VM} \in \{\boldsymbol{x}_1, \boldsymbol{x}_2 \dots \boldsymbol{x}_N\} \qquad (1)$$

$$\sum_{i=1}^{N} \|\boldsymbol{x}_{VM} - \boldsymbol{x}_i\|_{L_1} \leq \sum_{i=1}^{N} \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_{L_1} \qquad j = 1,2 \dots N$$

$$(2)$$

where $VM\{\cdot\}$ denotes the vector median operator.

Consider the case in which the output is not constrained to be one of the points inside the filter window. In this case the output is calculated by looking for the vector $\boldsymbol{x}$ which minimizes the function $f(\boldsymbol{x})$: $R^M \rightarrow R$ defined as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \|\boldsymbol{x} - \boldsymbol{x}_i\|_{L_1} = \sum_{j=1}^{M} \sum_{i=1}^{N} |\boldsymbol{x}(j) - \boldsymbol{x}_i(j)| \qquad (3)$$

where $\boldsymbol{x}(j)$ denotes the j-th component of vector $\boldsymbol{x}$. For each j, the corresponding component of the output-vector can be obtained by minimizing the quantity

$$\sum_{i=1}^{M} |\boldsymbol{x}(j) - \boldsymbol{x}_i(j)| \qquad (4)$$

which corresponds to the componentwise application of a scalar median filter to the vector image [2][6]. From the previous discussion it is clear that the only difference between the componentwise application of the scalar median filter and the vector median filter is that in the first case the output is not restricted to be one of the points inside the filter window. In other words, the vector median filter with L_1 norm approximates the componentwise application of the scalar median by constraining the selection to one of the points inside the filter window.

3. FAST VECTOR MEDIAN FILTER

A straightforward way to minimize the function $f(\boldsymbol{x})$ of equation (3) over the set of points inside the filter window, consists in computing the values of $f(\boldsymbol{x})$ at all points, and then in choosing the smallest one. In such a way, given an NxN window, $(N^2 - 1)$ distances must be calculated for each point. Since the window contains N^2 points, $N^2(N^2 - N)$ distances must be computed at each window position. The number of evaluated distances can be reduced by storing the distances which have already been calculated; in this way every time the filter window shifts by one pixel, only $N(N^2 - 1)$ distances need to be computed. Nevertheless the algorithm complexity is still too large.

The basic idea on which the fast algorithm is based is quite simple. Let us suppose that $f(\boldsymbol{x})$ assumes its minimum value on point $\boldsymbol{p}$. If such a point is known and the difference $f(\boldsymbol{x}) - f(\boldsymbol{p})$ is easier to calculate than the value $f(x)$, then a fast algorithm can be achieved by minimizing such a difference instead of $f(x)$. In our case the absolute minimum of $f(x)$ is known since it has been demonstrated in section 2 that such a minimum can be obtained by applying the scalar median filter separately to each component of the vector image. Besides, the time required for the computation of the scalar median is trifling with respect to the computation time of the vector median [5].

Hence, stated that $f(x) - f(p)$ can be easily calculated, the following scheme can be adopted for the fast vector median filter:

1. For each point in the image
   1.1 Componentwise apply the scalar median filter. Let p be the output of such an operation.
   1.2 For each point x_i of the filter window calculate $f(x_i) - f(p)$.
   1.3 The point which minimizes the quantity calculated in 1.2 is the vector median.

To investigate how the quantity $f(x_i) - f(p)$ can be calculated, let us consider the shape of the function $f(x)$ in more detail.

$f(x)$ is continuous since L1-distances change continuously in the R^M space. Besides, $f(x)$ is piecewise linear. To prove it consider the partial derivatives of $f(x)$ with respect to the components of vector x, we have

$$\frac{\partial f(x)}{\partial x(j)} = \sum_{i=1}^{N} sgn(x(j) - x(j)) \tag{5}$$

$$\begin{cases} sgn(x) = 1 & x>0 \\ sgn(x) = -1 & x<0 \end{cases}$$

which is a piecewise constant function. Hence $f(x)$ is piecewise linear. In particular, the breakpoints between the different regions of linearity are the points where $[x(j) - x_i(j)]$ changes sign for at least one i or j.

As an example look at the situation depicted in fig. 1. In it a set of 9 points $\{x_1, x_2 \ldots x_9\}$ in R^2 is shown. Point p represents the output of the componentwise application of the scalar median to the set, i.e. the point where $f(x)$ assumes its minimum value. As said before, $f(x)$ is piecewise linear and the horizontal and vertical lines passing through the points of the set define the borders between regions with different gradients. Moreover the partial derivative of $f(x)$ with respect to the i-th coordinate can be calculated very easily since it is equal to the number of points x_j for which $x_j(i) < x(i)$ minus the number of points for which $x_j(i) > x(i)$. Thereby, a practical way to calculate $f(x_i) - f(p)$ consists in integrating the gradient of $f(x)$ along a path which joins x_i and p and which is made of segments having the direction of the coordinate axes. In fig. 1 an example of such a path is illustrated for the 2D case. It is worthwhile to note that the gradient magnitude can be easily obtained from the histograms of the image components. Besides such histograms have yet been calculated during the componentwise application of the scalar median.

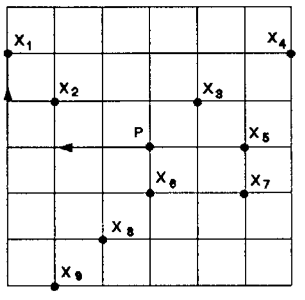

Fig. 1 At point p the function $f(x)$ assumes its minimum value. The vertical and horizontal lines define the breakpoints for the gradient of $f(x)$. The difference $f(x_i) - f(p)$ can be calculated by integrating the gradient of $f(x)$ along the path outlined in the figure.

4. FAST ALGORITHM COMPLEXITY AND EXPERIMENTAL RESULTS

In section 2 it has been demonstrated that the direct application of the vector median definition leads, in terms of computed distances, to an algorithm complexity of N^3 for each point in the image. For the fast algorithm, by neglecting the time spent for the application of the scalar median to each image component, only N^2 differences $f(x_i) - f(p)$ must be calculated.

To get more insight about the algorithm complexity, the exact way in which the differences are calculated must be considered. In table I an algorithm is reported which accomplishes the difference calculation by integrating the gradient of $f(x)$ along a path made of segments having the direction of the coordinate axis; the gradient intensity is obtained from the histograms of the vector components.

In order to quantify the performance of the fast algorithm several tests have been carried out.

```
- For each point of the scene
    - Componentwise apply the scalar median filter. Let
    vector p be the output of such an operation, ist[j] a
    vector containing the histogram of the j-th component
    of the image, nwin the number of points contained in
    the filter window and istosum[j] = $\sum_{r=0}^{p[j]}$ ist[j][r]
        - For each point x[i] of the filter window
            - diff[j] = x[i][j] - p[j]
            - if diff[j] > 0
                - sum[i] = diff[j] * istosum[j] * 2 - nwin
                - k = p[j]
                - for(m = 1; m < diff[j]; m++)
                    - k = k + 1
                    - sum[i] = sum[i] + (diff[j] - m) *
                    ist[j][k] * 2
            - else
                - sum[i] += diff[j] * (istosum[j] - nwin -
                ist[j][p[j]])
                - k = p[j]
                - for(m = -1; m > diff[j]; m--)
                    - k--
                    - sum[i] += ist[j][k] * 2 * (m -
                    diff[j])
    - Let the output of the filter be the point x[i] for
    which sum[i] is minimum
```

Table I Fast L1 vector median algorithm. The calculation of $f(x) - f(p)$ is performed by exploiting the histograms of the vector components.

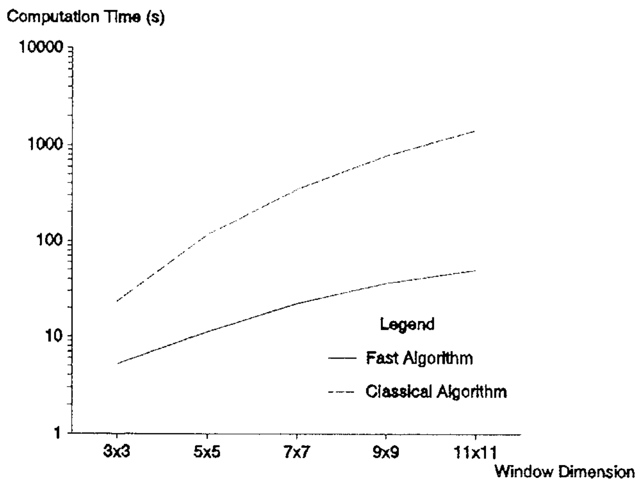

Fig.2 Comparison between the fast and the classical algorithms. The input image dimensions are 256x256.

In fig. 2 the comparison between the fast algorithm and an algorithm based on the vector median definition is shown. The adopted fast algorithm is the one outlined in table I. As it can be seen the proposed fast algorithm is significantly faster than the classical one. The performance improvement is more evident when large windows are adopted. The data reported on the diagram have been obtained by implementing the algorithms on a DECstation 5000/240 (40 MIPS).

REFERENCES

[1] **J.W.Tukey:** *"Nonlinear (nonsuperposable) methods for smoothing data"*; in Congr. Res. EASCON, p. 673, 1974.

[2] **I.Pitas, A.N. Venetsanopoulos:** *"Nonlinear digital filters: principles and applications"*; Kluwer Academic, 1990.

[3] **J.Astola, P.Haavisto, Y.Neuvo:** *"Vector Median Filters"*; IEEE Proceedings, Vol.78 no 4, p 678-689, April 1990.

[4] **Y.Neuvo, K.Oistama:** *"Video Signal Processing using Vector Median"*; SPIE Proceedings,Visual Communications and Image Processing", Vol.2, 1-4 October 1990, Lausanne.

[5] **M.Barni, V.Cappellini, A.Mecocci:** *"The Use of Different Metrics in Vector Median Filtering: Application to Fine Arts and Paintings"*; Proc. of EUSIPCO-92, Sixth European Signal Processing Conference, 25-28 August 1992, Brussels, pp. 1485-1488.

[6] **E.L.Lehmann:** *"Theory of point estimation"*; New York: Wiley, 1983.